Framework-basiertes interaktives Prototyping als Methode der transdisziplinären Entwicklung interaktiver, direkt reaktiver Software-Systeme

Dissertation

zur Erlangung des akademischen Grades
Doktor-Ingenieur (Dr. Ing.)

vorgelegt dem Rat der Fakultät für Mathematik und Informatik
der Friedrich-Schiller-Universität Jena

von Dipl.-Inf. Alexander Karosseit
geboren am 11. September 1974 in Stollberg

Gutachter:

1. Prof. Dr. Wilhelm R. Rossak
2. Prof. Dr. Michael Fothe
3. Prof. Dr. Ernst Jonas

Tag der letzten Prüfung des Rigorosums: 15. März 2007
Tag der öffentlichen Verteidigung: 29. März 2007

Für meine Eltern

Danksagung

Die wesentlichen Ergebnisse dieser Arbeit entstanden während meiner Forschungstätigkeit am Fraunhofer-Institut für Software- und Systemtechnik (ISST) in Berlin. Ich möchte allen Kollegen danken, die meine Arbeit dort unterstützt und gefördert haben. Insbesondere Dirk Alban Adler, Anne Lämmer, Kerstin Roost, Heiko Schefter, Katja Virkus, Dennis Wendlandt vom ISST sowie Robert Erber vom Ernst Klett Verlag verdanke ich große Teile der produktreifen Realisierung meiner Ideen. Danken möchte ich ferner meinen Vorgesetzten am Institut. Dr. Jörg Caumanns und Frank Fuchs-Kittowski, gaben mir sowohl wissenschaftlich als auch administrativ immer den nötigen Rückhalt.

Besonderer Dank gilt meinem Doktorvater Prof. Dr. Wilhelm R. Rossak. Seine wissenschaftliche Expertise, seine fachlichen und methodischen Ratschläge sowie vor allem seine Geduld mit dem nicht immer einfachen transdisziplinären Entstehungskontext haben diese Arbeit überhaupt erst möglich gemacht. Den Herren Prof. Dr. Michael Fothe und Prof. Dr. Ernst Jonas danke ich sehr herzlich für die Begutachtung der Arbeit, aber auch für die anregenden Diskussionen und die hilfreichen Anmerkungen während der Entstehung.

Dank schulde ich nicht zuletzt meinen Freunden und meiner Familie, insbesondere meinen Eltern Martina und Frank, meiner Freundin Nicolle Böhme sowie in Erinnerung meinen Großeltern für ihre ausdauernde seelische und moralische Unterstützung.

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

Noch immer scheitert ein Großteil der Software-Entwicklungsprojekte. Das belegen beispielsweise die Ergebnisse des „Chaos Report“ der *Standish Group*, dessen Ergebnisse hier zitiert nach [Hay04] angeführt sind. Demnach waren im Jahre 2004 lediglich 28 % der IT-Projekte erfolgreich. Insbesondere für den Bereich von Software-Produkten, die ihren Zweck im Wesentlichen über die Interaktion mit dem Nutzer und dadurch die Informationsvermittlung an den Endanwender realisieren, Produkten also, deren Erfolg von der Akzeptanz durch den Nutzerkreis abhängen, kann dabei die mangelnde Einbeziehung von Nutzern in die Entwicklung als Hauptrisikoquelle gelten. Dieser Umstand wird nach den eben zitierten Ergebnissen des „Chaos Report“ als einer der Hauptgründe für das Scheitern von Software-Entwicklungsbemühungen angegeben.

Für Anwendungsbereiche, deren inhaltliche Ausrichtung im Bezug auf die vorherrschenden Praktiken, Denk- und Sichtweisen weit von denen der Ingenieurwissenschaften und insbesondere der Informationstechnologie entfernt sind, ist Konzeption, Realisierung und Einsatz von technischen Lösungen problematisch. Doch gerade in diesen Bereichen sind die Potentiale für den Einsatz der Informationstechnologie und digitaler Medien zur Unterstützung der Arbeit, zur Präsentation und zur Vermittlung von Ergebnissen unter anderem zur Unterstützung der Lehre noch kaum erschlossen und schon gar nicht ausgeschöpft. Praxisgerechte Lösungen fehlen auf Grund der Entkopplung zwischen den beteiligten Disziplinen. Zu groß sind die Differenzen in der Expertise, also den Wahrnehmungsformen, Vokabularen, Routinen und Praktiken, um effektiv und konstruktiv zusammenarbeiten zu können. So bleiben Gestaltungspotentiale, die sich aus einer fachübergreifenden Perspektive ergeben sollten, häufig unerkannt (vgl. [STMTK01]).

Manifest werden diese Differenzen in besonderem Maße auch in den unterschiedlichen Ansichten bezüglich des Werkzeugeinsatzes während der Entwicklung. Während die originären Entwickler der Technologie die ihnen vertrauten Modelle und Software-Werkzeuge auf der Basis medien- oder technologiezentrierter Abstraktionen routiniert nutzen, bleibt den technologiefernen Fachexperten oder gar designierten Nutzern der Zugang zu solchen Produktionsmitteln und damit zu einer tatsächlichen gegenständlich tätigen Mitgestaltung verwehrt. Gerade hier ist das Potential des vorliegenden Ansatzes zu sehen: mit auf die Domäne zugeschnittenen Methoden und Werkzeugen ergibt sich perspektivisch die Möglichkeit, aus der fachlichen Arbeit heraus, selbstständig den Einsatz von Software zu gestalten. Abhängigkeiten und damit Unwägbarkeiten innerhalb der Arbeitsprozesse der Domäne, die sich an den noch immer schwer kontrollierbaren Schnittstellen zwischen Technologie und Fachlichkeit ergeben, lassen sich auflösen.

Damit deutet sich für viele Domänen ferner das Problem an, was sich aus dem Einsatz der Technologie ergibt. Die Abläufe, für die der Einsatz von Technologie vorgesehen ist, werden sich ändern. Mit der Veränderung der Prozesse gehen veränderte Aufgaben, Verantwortlichkeiten und Rollenverständnisse einher. Dies wiederum erfordert die Anpassung arbeitsorganisatorischer Rahmenbedingungen und die systematische Angleichung der entsprechenden Kompetenzen der Protagonisten. Es ist davon auszugehen, dass gerade auf Fachgebieten mit sehr langer Tradition ein vorsichtiges Hineinwachsen in diese veränderte Arbeits- oder Erlebenswelt unabdingbar sein muss. Aber auch Geschäftsmodelle etwa der Software-Dienstleister werden sich ändern. Es kann beispielsweise die Befähigung der Domäne zur selbstständigen Arbeit als Dienstleistung die eigentliche Auftragsentwicklung ablösen.

Unabhängig von der Schwierigkeit der Konvergenz der Disziplinen wird die Arbeit dadurch erschwert, dass sich viele der hier relevanten Einsatzgebiete von Informationstechnologie methodisch auf Forschungsniveau bewegen. Die Bewältigung der Komplexität der an den Nutzer zu vermittelnden Inhalte und Abläufe in Verbindung mit der benötigten zielgruppengerechten fach- sowie mediendidaktischen Aufbereitung erfordert häufig die Expertise und das kreative Potential von Wissenschaftlern. In [Geu00] wird beispielsweise davon ausgegangen, dass überhaupt noch wenige gesicherte Erkenntnisse dezidiert über das Lernen mit modernen Computern existieren. Nach [Sch97] besteht ein entscheidendes Problem darin, dass das Potential der Interaktivität nicht ausgenutzt wird. Es ist nahe liegend, dass die Reduktion auf bloße Präsentation kaum geeignet ist, Praxiswissen und insbesondere Handlungskompetenz zu vermitteln. Interaktivität praxisgerecht zu gestalten erfordert die entsprechende Fach-Kompetenz in der Entwicklung. Auch aus technischer Sicht ergeben sich in entsprechenden Projekten immer wieder Fragestellungen, die einer wissenschaftlichen Aufklärung bedürfen. Sowohl infrastrukturelle bzw. architektonische Aspekte, etwa zur Unterstützung kollaborativer Aspekte der Nutzung als auch offene software-ergonomische Problemstellungen unterlaufen den Wunsch nach produktionsorientierter Software-*Erstellung*.

Die nun folgenden Abschnitte werden die Problematik des Bruchs zwischen der technischen Umsetzung von interaktiven direkt reaktiven Systemen und ihrer praxisrechten Ausgestaltung konkretisieren. Diese Einführung in die Problematik wird schließlich münden in die Idee zu einem Brückenschlag, also zum Lösungsansatz, den diese Arbeit verfolgen wird.

1.1. Engineering vs. Authoring

Die Schritte vom Erkennen eines Bedürfnisses nach informationstechnologischer Unterstützung innerhalb einer Fachdomäne, über die Konkretisierung zu einem wirtschaftlich zu rechtfertigenden Bedarf und schließlich einer tatsächlichen Übertragung fachlicher Strukturen und Abläufe in adäquate software-technische Abstraktionen sind seit Bestehen der Disziplin *Software-Technologie* eine ihrer größten Herausforderungen. Insbesondere in der Spezialisierung des *Requirements-Engineering* (vgl. etwa [Rup02]) wer-

den immer wieder Wege gesucht die entsprechenden Prozesse systematisch, nachvollziehbar und vor allem reproduzierbar zu gestalten. Durch die weitgehende Universalität von Software-Systemen als Medien der Kommunikation, Vehikel der Produktion etc. sind sie in der Lage, beinahe jede Form von Wirklichkeit modellhaft nachzuahmen (vgl. [Wei90] Seiten ff.). Damit kann das Streben der Software-Technologen in gewisser Weise als Versuch betrachtet zu werden, die im Allgemeinen bezüglich jeglicher Taxonomie heterogenen Fachdomänen ingenieurmäßig zu normieren. Arbeits-, Denk-, und Sichtweisen potentieller Nutzer müssten demnach idealerweise in standardisierten Spezifikationsdokumenten fixiert werden können. Dies würde es erlauben, tatsächlich die durch Dijkstra [Dij89] beschrieben „Brandmauer“ zu errichten, um die von wechselseitigen Einflüssen „unbehelligte“ Ausübung der jeweiligen spezialisierten Tätigkeiten der Beteiligten zu erlauben. Beispielsweise in [Flo92] wird jedoch darauf hingewiesen, dass die Multiperspektivität innerhalb multidisziplinärer Projekte kaum „objektiv“ kommunizierbar ist.

Insbesondere für Systeme, die intensiv mit einem menschlichen Nutzer interagieren, ist unstrittig, dass singuläre abgeschlossne Phasen der Anforderungserhebung mit dem Ergebnis, einer vollständigen, widerspruchsfreien, sowie eindeutig interpretierbaren Spezifikation kaum realisierbar sind. So fordert etwa [Pre99] (auf den Seiten ff.) ganz grundsätzlich, dass die Entwicklung solcher Systeme den Nutzer in den Mittelpunkt stellen muss. Wobei die Spezifika der Disziplinen aller an der Entwicklung beteiligten jeder Zeit Berücksichtigung finden müssen – dem Interdisziplinären Charakter des Entwicklungsgeschehens ist also Rechnung zu tragen. Dies impliziert die Forderung nach einem kontinuierlich geführten Dialog, besonders zwischen den Systementwicklern und den designierten Anwendern. So geht [Pre99] davon aus, dass interaktive Systeme in der Regel iterativ zu entwickeln sind, wobei idealerweise Prototypen dabei helfen bereits sehr früh im Projekt schnelle Wechsel zwischen den Aktivitäten des Entwurfs und der evaluierenden Nutzung vollziehen zu können. So ist jederzeit eine direkte Rückkopplung zwischen dem fachlich Gewünschten oder Notwendigen und dem technologisch Machbaren möglich. De facto spielt jedoch offenbar weder die konsequente iterative Entwicklung, noch die Beteiligung von Nutzern eine Rolle in aktuellen Software-Projekten. Der bereits angesprochene „Standish Chaos Report 2004“ macht etwa neben der mangelnden Einbeziehung der Nutzer insbesondere auch die Neigung, vor allem große Projekte nach traditionellen, nicht iterativen Praktiken zu bearbeiten, für das Scheitern des größten Teils der Projekte verantwortlich (die Ergebnisse des Reports sind wiedergegeben nach [Hay04]).

Während diese Anforderungen an das Vorgehen zumindest theoretisch noch durch gängige Modelle und Methoden berücksichtigt werden, wird eine tatsächliche kontinuierliche und vor allem aktive Einbeziehung von Vertretern der Fachdomäne und Nutzern in den gesamten Prozess der Systementwicklung bisher kaum unterstützt. Die Notwendigkeit dazu ergibt sich beispielsweise in besonderem Maße aus der Tatsache, dass über bestimmte Klassen von Software- und insbesondere Informationssystemen ein Kommunikationsmodell realisiert wird, bei dem nur fachlich versierte Autoren in der Lage sind, die mit technischen Mitteln zu repräsentierenden Informationen effektiv für die Vermittlung an den Nutzer aufzubereiten. Diese notwendige aktive Rolle von Fachautoren innerhalb

des durch das Modell abgebildeten Vermittlungsprozesses erfordert also sinnvollerweise auch eine aktive Rolle bei der gesamten Gestaltung der unterstützenden Software.

Es können also nicht mehr die Sichtweisen und vor allem Tätigkeiten der Technologen und Ingenieure im Mittelpunkt stehen bei der Ausgestaltung und Optimierung des Arbeitsprozesses, sondern es müssen vielmehr die kommunikativen aber auch die operativen Aufgaben der Domänenexperten in den Fokus der Unterstützung gerückt werden. Die Bedeutung der Wirtschaftlichkeit der Entwicklung bleibt davon jedoch unberührt. Die Software- bzw. der Systementwicklung wird damit geprägt von einem Aspekt, den man nach [Mau85] als *Authoring* bezeichnen kann. Die Schwerpunkte der damit verbunden Aufgaben liegen dabei auf der Erarbeitung, der Strukturierung, der mediendidaktischen Aufbereitung sowie der Präsentation umfangreicher Sammlungen fachlicher Inhalte (Medien, Abläufe und Aufbaustrukturen). Dabei werden je nach Schwerpunktsetzung des Entwicklungsvorhabens verschiedene Rollen und damit Tätigkeiten den Entwicklungsprozess im Detail prägen: in Projekten beispielsweise, die explizite kommerzielle Lehrsoftware für den Einsatz an Schulen zum Ziel haben, werden Fachautoren Inhalte lediglich auswählen und zusammen mit Fachdidaktikern und Pädagogen für die zielgruppengerechte Präsentation und Interaktion gemäß den programmatischen Vorgaben durch Verlage oder Rahmenlehrpläne aufbereiten. Dabei ist bei den meisten Fachdomänen davon auszugehen, dass die Personen, die diese Rollen ausfüllen, originär keinen Bezug zur Informationstechnologie haben.

Um diese Aktivitäten jedoch tatsächlich mit der technischen Umsetzung – also auch die fachlichen Notwendigkeiten mit den technischen und medialen Potentialen – harmonisieren zu können, sollte die Arbeit dieser Rollen in direkter gestalterischer Auseinandersetzung mit dem Zielartefakt geschehen. Der nahe liegende Ansatz, den die Technologieentwicklung dazu hervorgebracht hat, besteht darin, dass die eingesetzten Werkzeuge in ihrer Anwendung von den zugrunde liegenden technischen Konzepten abstrahieren. Komplexe technologische Strukturen der Software werden reduziert auf ansatzweise lebensweltliche und vermeintlich intuitiv beherrschbare Metaphern.

Als problematisch erweist sich dabei die Vereinheitlichung der Entwicklungsschnittstellen, die mit der Forderung nach größtmöglicher Flexibilität schließlich doch wieder die Komplexität herkömmlicher Entwicklungssysteme induzieren. Dies wiederum zieht den Bedarf nach technologischer Spezialisierung und damit eine Isolation der Arbeit der Disziplinen nach sich.

Benötigt wird jedoch ein gesamtheitlicher Entwicklungsprozess bei dem im Diskurs zwischen den Domänenfachleuten, Nutzern und Technologieentwicklern Lösungen emergent werden. Nur im kontinuierlichen Dialog und der direkten Auseinandersetzung mit der entstehenden Software können Fachleute erkennen, was die Potentiale des Technologie- bzw. Medieneinsatzes sind und damit, aus welchen ihrer Bedürfnisse sich tatsächlich zu deckende Bedarfe ableiten lassen. Umgekehrt wird es so auch den Systementwicklern möglich, fachliche Belange – Motive der Domänenexperten und Nutzer – nachzuvollziehen, um schließlich individuell darauf eingehen zu können.

1.2. Multidisziplinarität, Transdiziplinarität

Verschärft wird die Problematik der Beteiligung unterschiedlicher Disziplinen am Entwicklungsprozess durch die Tatsache, dass insbesondere interaktive Anwendungen schon während ihrer Entstehung durch den designierten Nutzungskontext determiniert werden. Um tatsächlich praxisgerechte Software erstellen zu können, müssen die Erfahrung und das Wissen von Anwendern direkt in den Entwicklungsprozess fließen. Umgekehrt müssen die technologischen Gestaltungsspielräume systematische vermittelt werden, um Impulse bzw. Anreize für Anpassungen auch auf Seiten der Anwendungen zu setzen. Unterschiedliche Expertisen und somit beispielsweise unterschiedliche Vokabulare und Handlungsweisen der beteiligten Disziplinen müssen konvergieren als Basis der gemeinsamen Erschließung von Gestaltungspotentialen und -notwendigkeiten.

Der Kreis der Anwender wird dabei wiederum bezüglich der Disziplin und der Art und Weise, in der der Beteiligte durch die Nutzung betroffen ist, heterogen sein. Die Wirkung des Nutzungskontexts auf den Entwicklungskontext vollzieht sich jedoch auf zwei Ebenen: einerseits durch die Einbeziehung von Fachexperten oder „repräsentative“, designierte Anwender. Sie vertreten den Nutzerkreis in seinen fachlichen Bedürfnissen und evaluieren permanent das entstehende System im tatsächlichen, zum Teil prototypischen, produktiven Einsatz. Andererseits ist es für viele der hier betrachteten Produkte entscheidend, dass auch die Weiterentwicklung und Individualisierung der Software während des tatsächlichen Einsatzes Unterstützung finden. Ein elementarer Anwendungsfall dafür ist beispielsweise, die Anpassung, die ein Lehrer vornimmt, um mit dem System eine konkrete von ihm selbst gestaltete Unterrichtssituation unterstützen lassen zu können. Dies weitet die Entwicklung tatsächlich im Sinne von [FG04] in die Phase der Nutzung aus.

Im Idealfall kann eine weitgehende Kongruenz zwischen der durch das Zielsystem unterstützen fachlichen Tätigkeit und der Entwicklungsbeiträge der Fachexperten ausgenutzt werden. Das heißt, das was schließlich dem Anwender an fachlicher Unterstützung angeboten wird, unterstützt bereits einen der Nutzung fachlich vergleichbaren Prozess zur Zeit der Entwicklung. Diese auf den Anwendungskontext zugeschnittene Entwicklung wird die Grundlage sein für die tatsächliche Integration von Domänenexperten in den Entwicklungsprozess. Damit ergibt sich aber auch die Möglichkeit der kontinuierlichen iterativen Anpassung des entstehenden Artefakts und der Organisation seines Einsatzes an für die tatsächliche Anwendung effektive Szenarien.

1.3. Der Brückenschlag

Die benötigte Angleichung der Sicht- und Arbeitsweisen zwischen den unterschiedlichen Disziplinen aber auch zwischen den an der Entwicklung beteiligten Individuen birgt im Sinne eines planbaren, steuerbaren, reproduzierbaren Prozesses die Schwierigkeit die folgendes Zitat illustriert:

> „Als strukturdeterminierte Systeme sind wir von außen prinzipiell nicht gezielt beeinflussbar, sondern reagieren immer im Sinne der eigenen Struktur. So kann ich nicht steuern, wie meine Worte wirken: Jeder liest, was er oder sie liest, dafür trage ich keine Verantwortung! Nicht dieser Text legt fest, was Sie lesen, sondern Ihre Struktur, Ihre jeweilige Befindlichkeit. Dabei obliegt es jedoch allein mir, keinen Unsinn zu verzapfen, denn ich bin selbst verantwortlich für das, was ich schreibe - bloß bin ich nicht verantwortlich für das, was Sie lesen.“ [Mat01] (Seite 3)

Diese Einsicht belegt, den (mit Verweis auf [Flo92]) bereits angesprochenen Umstand, wie wenig es möglich ist, Multiperspektivität „objektiv“ zu kommunizieren. Es wird kaum ein deterministischer Prozess nach dem Muster eines klassischen Vorgehensmodells dazu führen, dass zwischen den Stakeholdern Konsens bezüglich gemeinsamer Zielvorstellungen sowie Sicht- und Vorgehensweisen entsteht.

Die aussichtsreiche Idee für diesen angestrebten Brückenschlag besteht darin, den Konsens in der direkten diskursiven Auseinandersetzung entstehen zu lassen. Im Sinne des Konstruktivismus, erzeugt sich jeder Beteiligte durch die Erfahrung aus der unmittelbaren konstruktiven Arbeit am Zielsystem seine eigene Sicht auf die Realität der Entwicklung und des entstehenden Artefakts in Abhängigkeit von seinen individuellen Erfahrungen, Kenntnissen und Möglichkeiten des Verständnisses. In Anlehnung an die Lernpsychologie, die Experten und Lernende betrachtet, soll hier angenommen werden, dass auch zwischen den Stakeholdern unterschiedlicher Disziplinen Wissen nicht wechselseitig „übermittelt“ wird, sondern – in der authentischen Situation der Arbeit am Zielsystem – jeweils neu „konstruiert“ und „ausgehandelt“ wird [Ker01] (Seite 80). Dieser Diskurs schafft also gewissermaßen gemeinsames neues Wissen – also beispielsweise Einsichten bezüglich des benötigten Funktionsumfanges oder auch der optimalen Abläufe der Produktion. Dies ist die Grundlage der Emergenz von Lösungen.

Als Aufgabe bleibt also, einen solchen diskursiven Prozess in Gang zu setzen und soweit dies möglich ist zu steuern. Das heißt, im Wesentlichen müssen die Selbstorganisation, und der evolutionäre Charakter des Prozesses sowie die Orientierung an der Tätigkeit der fachlichen Gestaltung angestoßen und gefördert werden. Zieht man wiederum die Parallele zum Bereich der Aus- und Weiterbildung und den dort etablierten Bestrebungen, Kompetenzvermittlung integriert in den operativen Arbeitsprozess informell zu betreiben (vgl. [RH02] oder auch [BR06]), müssen etwa Prinzipien wie das Schaffen lernförderlicher Rahmenbedingungen und die systematisch moderierte Selbstreflexion übertragen werden in Rahmenbedingungen, die die Co-Adaption zwischen den Disziplinen fördern.

Die Annäherung zwischen den Protagonisten der einzelnen Disziplinen bildet die Grundlage für die wechselseitige Adaption zwischen den Praktiken, Tätigkeiten und Arbeitsabläufen von Fachexperten bzw. Nutzern und den Werkzeugen zu deren Unterstützung. Ziel muss es sein in einem Prozess des interaktiven Designs [Wys97] und der Co-Adaption Software, Nutzer sowie die umgebenden Prozesse der Anwendung bzw. die arbeitsorganisatorischen Strukturen im Allgemeinen aufeinander abzustimmen. Wie in [Wys97] durch

das Beispiel des Eintragens von Wanderschuhen illustriert, muss die konsequente kontinuierliche Nutzung des Designgegenstandes (also der Software) in einem (weitgehend) realen Produktionskontext einerseits den Designgegenstand ausgestalten und aber andererseits auch den Nutzungskontext an die neuen Gegebenheiten anpassen. Die Idee ist es also, dieses Prinzip zu einem tragenden Teil des Entwicklungsgeschehens zu machen. Um sehr früh im Entwicklungsprozess die aktive, an den Tätigkeiten der Zieldomäne orientierte evolutionäre Gestaltungsarbeit zu ermöglichen, wird ein Software-Framework eine schnelle Entwicklung von Prototypen unterstützen.

Der Einsatz dieser instrumentellen Grundlage wird flankiert durch einen Vorgehensrahmen, des kollaborativen interaktiven Prototyping, das den gemeinsamen diskursiven Prozess sukzessive vorantreibt. Ausgegangen wird dabei von der initialen Entwicklung einer „Keimzelle“ für das gemeinsame Verständnis auf der Grundlage ethnographischer Methoden – der wechselseitigen Untersuchung der grundlegenden Charakteristika der jeweiligen Disziplin durch die aktive Teilnahme an bestehenden produktiven Abläufen.

Die angesprochene Anpassung der Prozesse, Organisationsstrukturen und Kompetenzen durch die kontinuierliche evaluierende Anwendung von Prototypen des entstehenden Systems, erfüllt Aufgaben der Organisations- und Personalentwicklung. Die unter Umständen nötige Verschiebung von Rollenverständnissen und Verantwortlichkeiten bedarf über die gewünschte Selbstorganisation hinaus in der Regel der systematischen Regelung in Abstimmung mit den Organisationen, die die Entwicklung tragen. Eine integrierte Organisations- und Technologie Entwicklung in Anlehnung an die Konzepte in [WR95] wird also die Administration und Moderation des Gesamtprozesses leiten müssen.

1.4. Übersicht

Im Anschluss an diese Einleitung, die die Problematik motiviert, das adressierte Aufgabenfeld umreißt sowie das Vorgehen während der Arbeit beschreibt, folgt in Kapitel 2 die systematische Charakterisierung des Diskursbereichs. Dazu werden zunächst die Anwendungsdomäne und die Spezifik der Entwicklung interaktiver, direkt reaktiver Systeme untersucht. Es werden Anforderungen an solche Systeme zusammengetragen und Qualitätsmerkmale in Analogie zu gängigen allgemeinen Kategorien des Software-Engineering konkretisiert und ergänzt. Den besonderen Einfluss den die Wechselwirkung zwischen dem Entstehungskontext und dem Anwendungskontext auf den Lebenszyklus und damit die Charakteristik der Systeme ausübt, wird untersucht.

Die Ergebnisse dessen münden schließlich in Abschnitt 2.1.3 in die Diskussion der Chancen und Risiken, die sich aus dem transdisziplinären Charakter von Software-Projekten unter dem Einfluss der unterschiedlichen mitwirkenden Disziplinen ergeben. Dazu werden typische Projekte und Projektsituationen untersucht und zu einer de facto Schablone für entsprechende Projektorganisationen verallgemeinert. Typische Rollenbilder einzelner Disziplinen und deren Wechselwirkungen werden beschrieben. Als Ergebnis der Betrachtung des Spannungsfeldes zwischen operativem wirtschaftlichen Einsatz des

Systems, dem Innovationsbedarf der Forschung sowie politischen und kulturellen Einflüssen werden anschließend der Einfluss des Nutzungskontexts und die sich ergebende transdisziplinäre Konstellation begrifflich präzisiert.

Der Abschnitt 2.2 widmet sich den Auswirkungen der Werkzeuglandschaft auf die Entwicklungsprozesse. Dazu werden etablierte Entwicklungsumgebungen eingehend analysiert: Eigenschaften wie die verwendeten Metaphern, die unterstützten mentalen Modelle, Interaktionsschemata und Navigationsmuster sowie Infrastrukturaspekte wie die physikalische Modularisierbarkeit und Verteilbarkeit der Artefakte werden zusammengetragen und verallgemeinert. Die Konsequenzen für die Kernprozesse der Software-Entwicklung aber auch für die Querschnittsaufgaben der Projektorganisation, der Qualitätssicherung oder des Konfigurations- bzw. Änderungsmanagement werden verdeutlicht. Sowohl die identifizierten instrumentellen Probleme, die sich aus den Eigenarten der so genannten Autorenumgebungen ergeben, als auch die damit verbunden prozessbezogenen Unwägbarkeiten und damit Risiken münden dabei jeweils in Anknüpfungspunkte für die Formulierung der Thesen und Zielsetzungen zu Lösungsansätzen am Schluss des Kapitels.

Sowohl die methodischen als auch die instrumentellen Defizite, die sich in gängigen Projekten, der hier untersuchten Kategorie beobachten lassen werden durch verschiedene Forschungsbemühungen adressiert. Der Abschnitt 2.3 wird die wichtigsten Bestrebungen zusammentragen. Dazu werden auf der einen Seite die für den betrachteten Bereich spezialisierten Ansätze vorgestellt. Es werden aber auch etablierte Lösungen und Paradigmen besprochen, die als Grundlagen in die Ideen der vorgestellten Lösung eingeflossen sind. Alle der Vorgestellten, Ansätze, Lösungen und Theorien werden erörtert in direktem Bezug zum Beitrag, den sie jeweils leisten werden und die ggf. erforderlichen konzeptionellen Erweiterungen, Interpretationen und Konkretisierungen für den Einsatz.

Die Thesen, die dem Problempotential konkrete Lösungsmöglichkeiten entgegensetzen sind in Abschnitt zusammenfassend dargelegt. In Ergänzung zu den eigenen Lösungsvorschlägen werden dazu verwandte Bestrebungen aufgegriffen, weiterentwickelt, modifiziert oder kombiniert. In Abbildung 1.1 ist das Anknüpfen an die einzelnen Aspekte der Problemstellung bzw. der Defizitbetrachtung verwandter Arbeiten als Ausgangspunkt für die Ableitung der Thesen symbolisiert durch die vertikal verlaufenden filigraneren Pfeile innerhalb von Kapitel 2.

Im zentralen Kapitel 3 wird der Beitrag der Arbeit im Detail entwickelt. Dazu wird zunächst das Software-Framework als instrumentelle Grundlage der Entwicklung vorgestellt. Es werden der Aufbau und die Funktionsweise sowie technologische Aspekte der Implementierung erläutert. Im sich anschließenden Abschnitt 3.2 wird das Konzept der Entwicklungsmethodik des interaktiven Prototyping vorgestellt. Diese ist einerseits durch die Anwendung des Frameworks geprägt und geführt. Das Vorgehen bedarf aber andererseits einer Reihe von Überwachungs- und Steuerungsmechanismen, die den gewünschten evolutionären und partizipativen Charakter des Vorgehens nachhaltig etablieren sollen. Über die Entwicklung des Entwicklungsvorgehens hinaus werden dazu die Rahmenbedingungen beschrieben, die die für den Erfolg des Ansatzes notwendigen

querschnittlichen Prozesse der Personal- bzw. der Organisationsentwicklung aber auch des Wissensmanagements unterstützen werden.

Die Tragfähigkeit der vorgestellten Lösungsvorschläge wird anschließend in Kapitel 4 untersucht und damit die Gültigkeit der am Ende des Kapitels 2 aufgestellten Thesen nachgewiesen. Dazu wird die Fallstudie einer konkreten Anwendung der Lösungsansätze im Rahmen einer tatsächlichen vollzogenen Produktentwicklung präsentiert. Die gemachten Beobachtungen in Bezug auf die Vorgehensweise unter Einbeziehung von IT-fernen Projektbeteiligten in originär technologische Aktivitäten der Entwicklung werden qualitativ untersucht und beschrieben. Die Wirkungsweise des Frameworks wird anhand der dezidiert vorgenommenen Ausprägung illustriert. Die jeweiligen Einzelaspekte der Konzeption, also etwa dezidierte Framework-Funktionsblöcke für Aufgaben wie die Etablierung der Interaktivität, werden in der konkreten Anwendung während der Entwicklung der exemplarischen Software demonstriert.

In Kapitel 5 schließlich werden die Ergebnisse und Erkenntnisse zusammengefasst. Die Lösungen werden abschließend bezüglich der durch sie eröffneten Gestaltungspotentiale aber auch bezüglich ihrer Defizite diskutiert. Die Anschlussfähigkeit der Arbeit wird in zwei Dimensionen erörtert. Einerseits werden die noch offenen Forschungsaufgaben und -potentiale betrachtet. Andererseits wird ein Einblick in mögliche Szenarien der wirtschaftlichen Verwertung gegeben.

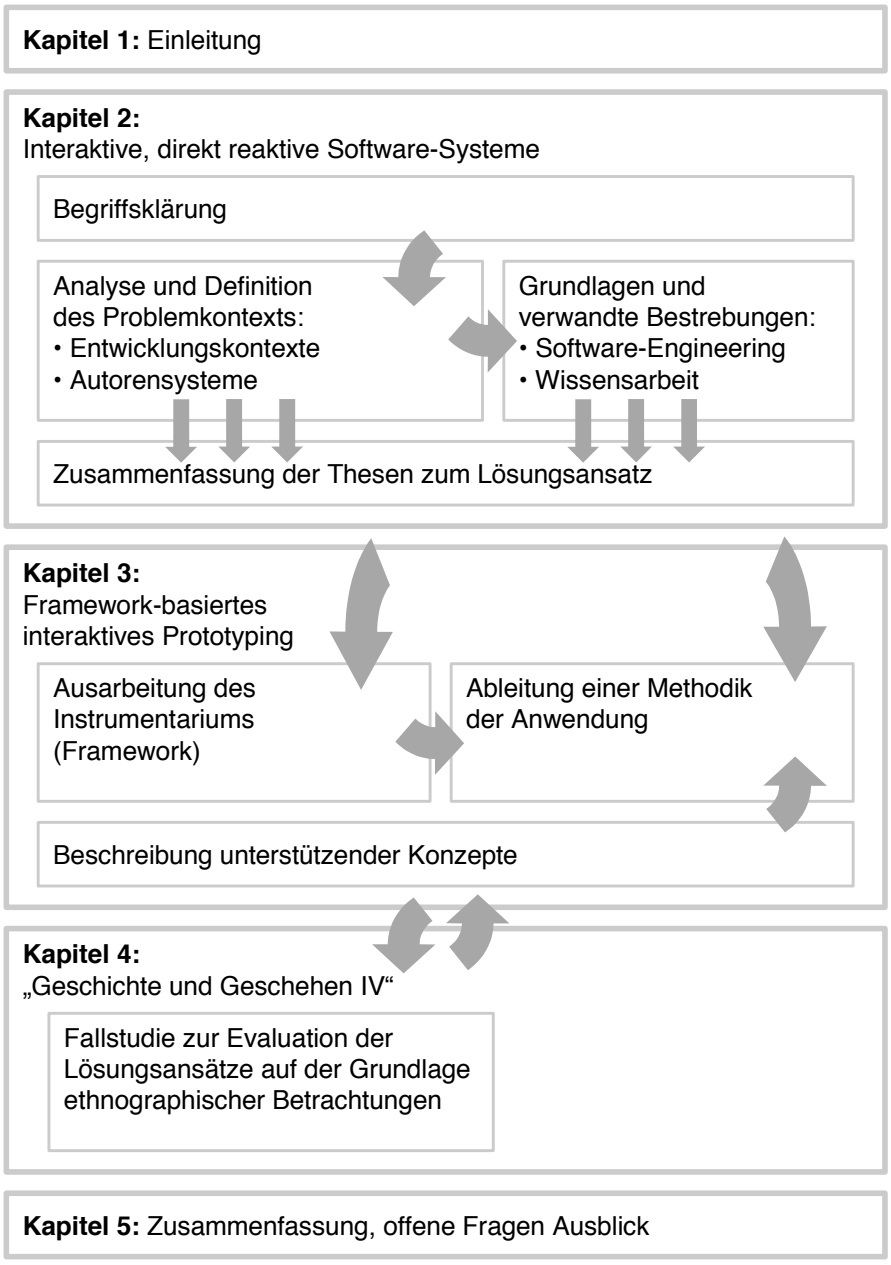


Abbildung 1.1.: Aufbau der Arbeit

2. Interaktive, direkt reaktive Software-Systeme

Das Erkennen von Bedarfen nach informationstechnologischer Unterstützung und daraus folgend das Antizipieren konkreter Anforderungen an ein zu entwickelndes System sowie schließlich die adäquate wirtschaftliche Umsetzung dessen, sind die zentrale Herausforderung der Software-Technik seit Beginn ihrer Existenz. Das von Dijkstra [Dij89] geforderte errichten einer „Brandmauer der Informatik“ zur Abschottung der Software-Entwickler von den nur „unpräzise“ zu fassenden lebensweltlichen Details der Anwendungsdomäne ist spätestens bei der Erstellung so genannter interaktiver, direkt reaktiver und medienorientierter Software, die eng in die kommunikativen und produktiven Prozesse von Menschen eingebunden ist, kaum mehr möglich.

Das Kapitel wird zunächst die Spezifik dieses Typs von Software präzisieren, indem allgemeine Anforderungen und Merkmale der Systeme selbst in Abgrenzung etwa zu betrieblichen Informationssystemen beschrieben werden. Besonderheiten bezüglich der Entstehungs- und Einsatzkontexte und deren transdisziplinäre Verquickung werden umrissen. Anschließend wird der Stand der Forschung und der alltäglichen Praxis bezüglich der methodischen Software-Entwicklung innerhalb der Entstehungskontexte ausgearbeitet: typische Entwicklungsszenarien beschrieben, der Einfluss gängiger Werkzeuge auf das methodische Vorgehen erörtert sowie erkannte Defizite und Ansätze zu deren Behebung vorgestellt. Außerdem werden Paradigmen erläutert, deren Konzepte die Problemstellung nicht unmittelbar adressieren, jedoch die Grundlage für die in Kapitel 3 entwickelten Konzepte bilden.

2.1. Charakterisierung der Anwendungsdomäne

Der Bedarf nach einer *neuen* Vorgehensweise zur Entwicklung von Software ist hier dahingehend relativiert zu betrachten, dass eine spezifische Charakteristik innerhalb einer Klasse von Software-Entwicklungsbemühungen dezidierte Lösungen bezüglich der Entwicklungsunterstützung benötigt.

Die Besonderheiten der Anwendungsdomäne und damit das Besondere an den Anforderungen sowohl an das zu entwickelnde Produkt und seinen Lebenszyklus, als auch an die damit verbundenen Produktionsprozesse werden in den folgenden Abschnitten beschrieben. Dazu wird zunächst der Begriff der *interaktiven, direkt reaktiven Anwendungen* entlang grundlegender typischer Anforderungen und Qualitätsmerkmale umrissen. Insbesondere durch Merkmale, wie beispielsweise das hohe Maß an Interaktivität und die

Komponente der didaktisch bewussten Präsentation ggf. großer Mengen an Inhalten aber auch wichtigerweise die Vermittlung von Handlungskompetenzen an die Nutzer, wird etwa die Abgrenzung zu betrieblichen Anwendungen oder gar eingebetteten Systemen konkretisiert.

Davon ausgehend werden typische und notwendige Projektkonstellationen beschrieben. Das umfasst zum einen die Stakeholder-Struktur, die den transdisziplinären Charakter der betrachteten Projekte prägt. Zum anderen werden die daraus resultierenden Aufgabenverteilungen und damit die Abläufe beschrieben, die die de facto Gradwanderung zwischen einer Software-Entwicklung nach dem klassischen Verständnis und den häufig durch Verlage und Redaktionen motivierten Produktionsprozessen schaffen müssen.

2.1.1. Grundlegende fachliche Anforderungen

Die Kategorie der interaktiven, direkt reaktiven Software-Systeme, die hier betrachtet werden soll, bildet eine Unterklasse Hypermedialer- bzw. Multimedialer Anwendungen. Der Begriff wurde geprägt in den Veröffentlichungen [SR02] bzw. [SRK00]. Das dort entwickelte Verständnis wird weitgehend übernommen. Eine weitere Konkretisierung und auch die Abgrenzung etwa gegenüber betrieblicher Anwendungs-Software oder eingebetteten Systemen werden durch die Beschreibung spezifischer Anforderungen und der daraus resultierenden üblichen Systemeigenschaften vorgenommen.

Informations- und Kompetenzvermittlung

Die hier betrachteten Zielsysteme erfüllen primär die Aufgabe von Informationssystemen im Sinne von [Fer03] (Seite 12). Nach diesem Verständnis sollen die hier adressierten Systeme Informationsbedarfe aus einem (persistent) materialisierten Wissensbestand befriedigen. Die im Dialog gewonnenen und übermittelten Informationen sollen schließlich durch Rezeption, Reflexion und Reproduktion als neues Wissen im Wissensbestand des Nutzers kontextualisiert werden [Fer03] (Seite 26 f.). Die hier betrachteten Systeme instanziieren dieses abstrakte Kommunikations- bzw. Informationsmodell, wobei der Sender in der Regel der domänensachverständige Autor sein wird und der Empfänger derjenige, der Informationen sucht bzw. bewusst Wissen und Kompetenzen erwerben möchte. Letzteres bricht die Gerichtetheit des Prozesses auf: der (Lern-)Prozess des Austauschs zwischen dem kodifizierten medialen Wissen und dem Lernenden bzw. Informationssuchenden ist in der Regel interaktiv (vgl. [Ker01] Seite 147, Abbildung 11).

Dabei sollen zunächst keine Annahmen über die zu Grunde zu legenden Basismodelle dieses Vermittlungs- bzw. Lernprozesses getroffen werden. Die Systeme, deren Entwicklung mit Hilfe des hier präsentierten Ansatzes unterstützt wird, sollen sowohl einfache Auskünfte geben können – etwa an Kiosk-Systemen, als auch ein möglichst breites Spektrum an Basismodellen und Typen von Lehr- bzw. Lernzielen bedienen können. Dazu gehören beispielsweise vordergründig (vgl. [NHHM$^+$04] Seite 76 bzw. 78):

- Entdeckendes Lernen und Problemlösen

- Begriffsbildung (z. B. Durcharbeiten von Lehrmedien, mediale Bestimmungsübungen)
- Konzeptbildung (Erschließung größerer Sachzusammenhänge, Analogiebildung; z.B. Entwicklung von geschichtskritischem Denken)
- Betrachtendes Lernen
- Lernen von Strategien
- Routinebildung, Training und Übung
- Hypertext-Lernen
- begrenzt auch Aufbau dynamischer Sozialbeziehungen

Neben den technisch, instrumentellen Aufgaben sollen die hier interessierenden Zielsysteme also auch soziale und Aufgaben im Sinne von [FRS89] übernehmen. Eine Klassifizierung von Systemen, die aus dem angebotenen Framework abgeleitete werden können differenziert nach dem Freiheitsgrad an Interaktionen, die einem Anwender zur Verfügung gestellt werden bzw. umgekehrt dem Grad der Kontrolle und der Führung, die ein System über den Nutzer übernimmt (vgl. [Sch97] Seiten 66 f.) – danach wird unterschieden in:

Courseware und Autorensysteme: Die zu vermittelnden Inhalte (Medien) werden mehr oder minder sequentiell, z.B. einem Curriculum folgend, präsentiert. Den Ablauf steuert der Nutzer in der Regel durch Weiterblättern oder andere einfache Schritte der Navigation. In einfachen Frage-Antwort-Dialogen, wird der Lernfortschritt des Nutzers ermittelt und entsprechend Feedback gegeben bzw. der weitere Ablauf bedingt.

Drill & Practice-Programme: Frage-Antwort-Dialoge nach festen Schemata wie, „Ja / Nein“, Multiple Choice, Lückentext, Freie Antwort ([Sch97] Seite 105), Drag & Drop etc. helfen insbesondere bei wiederholenden Übungen – etwa beim Vokabeltraining.

Präsentationen: Inhalte werden präsentiert, ohne größere Eingriffsmöglichkeiten durch den Nutzer. Häufig unterstützen mediale Präsentationen auch Vortragende.

Kiosk-Systeme: präsentieren Inhalte ebenfalls entlang von durch Autoren editierten Pfaden aber auch in beliebigen Hypermedialer Strukturen. Sie folgen dabei im Ablauf – den der Nutzer wiederum häufig durch Blättern bestimmt – eher einer narrativen Chronologie (vgl. [Sch97] Seite 326). Sie unterstützen insbesondere auch Suchmechanismen in umfangreichen Medienbeständen, sie dienen häufig Marketingzwecken in Warenhäusern und auf Messen, bieten ergänzende Informationen in Museen etc.

Hypermediale Systeme und Guided Tours: Beliebige Entitäten der Nutzerpräsentation können dabei verknüpft und inhaltlich in Beziehung gesetzt werden. So entstehen beliebig vernetzte Strukturen. In der Regel impliziert das Setzen eines solchen Links die Ausprägung von Programmablaufstrukturen, wie den Aufruf der Anzeige des assoziierten Elements – also etwa Schritte der Navigation. *Guided Tours* beschränken die Vernetzung zielgerichtet, um den Nutzer auf didaktisch intendierten Pfaden durch das Netz zu führen.

Direkt manipulative Programme: Darunter sollen Programme verstanden werden, mit deren Hilfe, Daten (Medien, Inhalte) durch den Nutzer weitgehend frei und im Sinne der Anwendungsdomäne operativ – generierend und konstruktiv – bearbeitet werden können.

In der Regel werden diese Paradigmen kombiniert eingesetzt - so können Knoten innerhalb einer Hyper-Media-Struktur beispielsweise vollwertige Programme zur Manipulation der Inhalte sein, die über den Link assoziiert sind. So kann das präsentierte Faktenwissen in eine direkte funktionale Verbindung gebracht werden mit dem Transfer in die operative Anwendung oder die repetierende Übung.

Nach [NHHM+04] (Seiten 110 ff.) und [Wei02] (Seiten 32 f.) sollen die hier betrachteten Anwendungen, unabhängig von der konkreten mediendidaktischen Auslegung, die klassischen sechs Anforderungen an Systeme zur Präsentation multimedialer Informationen bzw. zur Vermittlung von Wissen erfüllen können:

Motivation Aufmerksamkeit: Die Aufmerksamkeit und das Interesse des Rezipienten bzw. Nutzers sind zu wecken und aufrechtzuerhalten. Dabei ist Klarheit bezüglich der Lern- bzw. Vermittlungsziele zu schaffen bzw. zu wahren.

Information: Vermittlungszielrelevante Inhalte müssen adäquat zugänglich sein und präsentiert werden.

Informationsverarbeitung : Die Interpretation und das Verstehen der angebotenen Informationen muss unterstützt werden. Es bedarf je nach didaktischem Konzept der Anleitung und Führung.

Speichern und Abrufen: Um das Erfahrene bzw. Gelernte nachhaltig zu einem Teil des jeder Zeit abrufbaren Wissensschatzes des Rezipienten zu machen, werden Mechanismen der vertieften (z.B. wiederholenden, übenden) Auseinandersetzung mit den Inhalten benötigt.

Anwendung und Transfer: Auch das Ablösen des Gelernten von der Lernsituation und eine unmittelbare Operationalisierung durch die direkte Anwendung sollte unterstützt werden können. Dazu gehört auch die Übertragung des neuen Wissens in neue Kontexte z.B. durch die Anwendung einer Technik auf andere Inhalte und das Ziehen von Analogieschlüssen.

Steuerung und Kontrolle: Das Zusammenspiel dieser Funktionen muss angemessen reguliert werden. Die wichtigsten Komponenten dabei sind informative Rückmeldung, also etwa das Transparentmachen von Erfolgen und der angemessene Umgang mit Misserfolgen und Fehlern.

Es wird zu einem großen Teil die Aufgabe interaktiver Komponenten sein, die Erfüllung dieser Anforderungen zu unterstützen. Dementsprechend werden diese Anforderungen im Kontext der Beschreibung der Interaktivität nochmals aufgegriffen und präzisiert. Je nach Auslegung des Systems im Sinne der oben vorgestellten Paradigmen werden diese Funktionen mit mehr oder minder Gewicht zu Teilen interaktiver, direkt reaktiver Systeme nach dem Verständnis der vorliegenden Arbeit sein.

Die Charakterisierung der hier interessierenden Systeme nach den an sie gestellten Anforderungen in Bezug auf die effektive Vermittlung von Fakten, Fähigkeiten und Fertigkeiten soll durch eine von [Geu00] (Seite 135) gewonnene Erkenntnis zusammengefasst werden. Demnach gibt es nur sehr wenige gesicherte Erkenntnisse über das Lernen mit Hilfe moderner Computer. Insbesondere bei der Vermittlung von Handlungswissen, dem Aufbau echter Handlungskompetenz gibt es bezüglich vieler Fächer im Detail konkreter Projektbemühungen erhebliche Defizite – lediglich intuitive und zum Teil naive Versuche. Das ist mit Sicherheit zu allererst dadurch begründet, dass die meisten der Tätigkeiten in vielen Disziplinen überhaupt noch keine sinnvolle Unterstützung durch Informationstechnologie finden und überhaupt technische Abstraktionen für fachliche Konzepte der Domänen grundsätzlich unerforscht sind.

Interaktivität

Die Eigenschaft, die die hier betrachteten Systeme – bereits ausgewiesen durch den Titel – auszeichnet ist die Interaktivität. Entsprechend der sozialwissenschaftlichen Definition soll unter Interaktion „das wechselseitig handelnde aufeinander Einwirken zweier Subjekte“ [NHHM$^+$04] (Seite 109) verstanden werden. Es wird (etwa ebenfalls in [NHHM$^+$04]) als metaphorische Erweiterung des Begriffs verstanden, wenn eines der (menschlichen) Subjekte durch ein Software-System ersetzt wird. Pragmatisch betrachtet, ist ein System also dann interaktiv, wenn es in einer bestimmten Situation (in einem bestimmten Zustand) auf Eingaben des Nutzers in definierter Weise reagiert (vgl. [NBS$^+$99] Seite 17). Laut [Sch97] (Seiten 47) ist die Qualität der Interaktion dabei auf drei ebenen zu unterscheiden:

Reaktiv ist die Interaktion, wenn der durch eines der beiden Subjekte gesetzte Reiz durch ein entsprechendes Feedback beantwortet wird. Die Verkettung solcher Interaktionsschritte zu Dialogen wird auch als Interaktionskette bezeichnet [NHHM$^+$04] (Seite 110). Wenn die hier betrachteten Systeme als *direkt reaktiv* bezeichnet werden, deutet dies darauf hin, das diese Feedback-Zyklen und insbesondere die Reaktionen des Systems unmittelbar erfolgen – aus der Sicht des Anwenders synchron, ohne Zeitverzögerung durch Schritte der internen Verarbeitung.

Als *proaktiv* werden Interaktionsszenarien bezeichnet, in denen der Nutzer konstruktiv mit dem System arbeitet. Die Tätigkeit des Anwenders verändert die vorgegebene Struktur des Präsentierten durch eigene „einzigartige Konstruktionen und Elaborationen jenseits der vom Designer eingerichteten Regeln“ [Sch97] (Seite 47). Proaktive Interaktionsmuster sollen ohne Einschränkung der Allgemeinheit durch Bestandteile direkt reaktiver Systeme im Sinne des in dieser Arbeit verwendeten Verständnisses möglicher Zielsysteme unterstützt werden.

Echte *wechselseitige* Interaktion, die laut [Sch97] (Seiten 47) Mechanismen künstlicher Intelligenz voraussetzt und eine aktive (automatische) wechselseitige Anpassung impliziert, ist dagegen nicht Gegenstand der Betrachtungen.

Interaktionen dieser drei Qualitätsstufen werden weiterhin charakterisiert durch die Art der grundlegenden Transaktion, die sie realisieren (vgl. [Sch97] Seiten 47 und [MFCS92] Seite 12 f.):

- Bestätigung bzw. Unterbrechung (z.B. Mausklick auf eine Schaltfläche, um den Ablauf eines Films zu stoppen bzw. zu starten oder auch die Auswahl einer Multiple-Choise-Antwort zu treffen)
- Durchwandern / Traversieren (z.B. Vorwärtsbewegung durch eine Folge von Bildschirmen gemäß der durch die inhaltliche Struktur oder der didaktischen Intention vorgegebenen Ordnung. Dies kann wiederum durch Mausklicks auf entsprechende Schaltflächen gesteuert erfolgen.)
- Navigieren (z.B. freies Bewegen in einem virtuellen Raum das folgen mit einem Datenhandschuh, aber auch das Verfolgen von Hyper-Verweisen)
- Anfragen bzw. Befragungen (z.B. Volltextsuche, bei der Textdokumente nach einer über die Tatstatur eingegeben Zeichenkette durchsucht werden)
- Ausarbeitung / Ausführung / Vertiefung (z.B. visuelles Gestalten von Grafiken und Animationen mit Hilfe entsprechend komplexer Werkzeugpaletten)

Die in den Beispielen jeweils angedeuteten physikalisch, technischen Interaktionsprimitive (die Syntax der Interaktion im Sinne von [Sch97] Seite 43), die die hier adressierten Systeme unterstützen sollen, entsprechen dabei dem, was Anwender aktueller grafischer Nutzerschnittstellen erwarten:

- Das Klicken von Schaltflächen zum Auslösen von Funktionen entsprechend der implizierten Semantik der Schaltfläche (einschließlich der Gruppierung in Menüs und Dialogen etc.).
- Das Ergreifen und Verschieben von Bildschirmobjekten, einschließlich der Erzeugung von Objekten, aber auch der geführten Bewegung wie bei der Scrollbar.
- Eingabe von Text über die Tastatur und Verwenden von Tastenkürzeln.

- Es besteht die Möglichkeit der Kombination dieser Primitive, wobei Maus- bzw. Tastaturereignisse (Maus bzw. Taste gedrückt, los gelassen, etc.) abhängig vom Kontext (bei der Maus in der Regel abhängig von der aktuellen Zeigerposition und Zustand des Programms) ausgewertet werden.

Die *Interaktivität* bezeichnet laut [NHHM+04] (Seite 109) schließlich das Ausmaß, indem ein System Interaktionen ermöglicht bzw. erfordert, also die Häufigkeit, in der die oben beschriebenen Schritte der Interaktion ausgeführt werden. Für die hier untersuchten Systeme soll gelten, dass sie in einem sehr hohen Maße interaktiv sind – ihr Zweck an die Interaktion mit dem Nutzer gebunden ist. Die Funktionen, die sich erst in der Interaktion realisieren ergeben sich entsprechend unter anderem aus den fachlichen Anforderungen der Informations- und Kompetenzvermittlung (siehe vorangegangenen Abschnitt; vgl. [NHHM+04] Seiten 110 ff.):

Motivation und Aufmerksamkeit fördern: Beispielsweise durch spielerische Elemente, ermutigende Rückmeldungen Äußerungen zu Nutzereingaben wird die Aufmerksamkeit bzw. das Interesse der Nutzer gebunden und die Motivation aufrechterhalten. Es ist wichtig (Lern-)Erfolge transparent machen. [NHHM+04] weißt explizit darauf hin, dass sich abschätzige Bemerkungen im Feedback verbieten, die die Selbsteinschätzung des Anwenders negativ beeinflussen.

Informationen liefern: Die Gestaltung der Interaktionsketten muss die benötigen Informationen effizient zugänglich machen. Hinweise auf Defizite auf Seiten des Anwenders, die etwa aus Fehlersituationen ableitbar werden, fördern dabei ggf. die Selbständigkeit bei der Extraktion und Aneignung des Wissens.

Verständnis unterstützen: Auch die Anschaulichkeit und damit die Verständlichkeit von Inhalten kann durch Interaktion erhöht werden. So kann beispielsweise die (mutlimodal) abhängig vom Bedürfnis des Nutzers wählbare Darstellungsform auf individuelle Wahrnehmungsgewohnheiten eingehen.

Behalten, routinisieren fördern: Die Übung in der echten proaktiven kreierenden Auseinandersetzung mit Sachverhalten und Abläufen hilft bei der Vertiefung des erworbenen Wissens ebenso wie (spielerische) Übungen des Repetierens und die interaktive Umsetzung von Mnemotechniken wie Mind-Maps oder Lernkarten.

Transfer: Die Übertragung der neu gewonnenen Kenntnisse in die Anwendung zur Lösung von Aufgaben- und Problemstellung oder der Transfer in analoge bzw. verwandte Wissensgebiete wird beispielsweise unterstützt durch die Assoziation (etwa die Verknüpfung über Links) der Informationspräsentation mit Aufgaben und weiteren Themenbereichen. Die konstruktive Bearbeitung der Problemstellung kann dann etwa durch interaktive, direkt manipulative Werkzeuge unterstützt werden.

Regulation: Die Steuerung des Prozesses der Informationsvermittlung kann ebenfalls über Eingriffe in die Abläufe der Interaktion erfolgen. So können etwa Mechanismen der Nutzeroberfläche dafür Sorge tragen, dass bereits erarbeitete Inhalte entsprechend markiert werden bzw. nicht mehr zugänglich sind. Solche zu deren

Verständnis der Nutzer die Voraussetzungen noch nicht erfüllt, können gleichermaßen verborgen bleiben.

Navigation

Die zentrale der kommunikativen Aufgaben des Dialogs zwischen System und Nutzer stellt die Navigation dar. Diese realisiert die Bewegung (z.B. vorwärts und rückwärts) zwischen wahrnehmbaren, inhaltlich kohärenten Elementen (Knoten) innerhalb der Struktur des Systems abhängig vom aktuellen Kontext (vgl. [MFCS92] Seite 13). Rossi und Schwabe postulieren in [SR98], dass eine gute Navigationsstruktur einer der Schlüssel zum Erfolg ist bei der Gestaltung von *Hypermedia-Anwendungen*. Auch wenn der Begriff der hier adressierten interaktiven Anwendungen weiter gefasst sein soll als der von Hypertext-Systemen, etwa auf der Grundlage von HTML, wird die Unterstützung des Nutzers beim Auffinden von Funktionen und Inhalten, eine der zentralen Herausforderungen für das System sein. Es gilt in jedem Falle auch für interaktive, direkt reaktive Systeme:

> „Wenn der Nutzer weiß, wohin er gehen kann und wie er das gewünschte Ziel erreicht, kann er am besten von einer Anwendung profitieren.“ [SR98][1]

In der gleichen Publikation wird darauf hingewiesen, dass bereits die Kombination von Navigationsstrukturen mit Multimedia-Inhalten zu weiter reichenden Problemen führt. Erweitert man das Problemfeld, um die im hier betrachteten Kontext hinzukommenden eigenständigen Interaktiven (Teil-)Anwendungen, ist also davon auszugehen, dass die Navigation durch den vorgestellten Entwicklungsansatz explizit, sowohl instrumentell als auch methodisch in der Konzeption und der Umsetzung unterstützt werden muss. In Analogie zu [SR98] ergibt sich auch für diesen Diskursbereich beispielsweise die Frage: was passiert, wenn der Nutzer über die Navigation in eine eigenständige Anwendung mit eignen Kontrollflüssen und Interaktionsschemata gelangt? Wie findet er zurück? Wie behält er Übersicht und Kontrolle? In [EOOH94] (Seite 290 f.) wird der Begriff der *Navigation* als Nutzerdialog zur Ablaufsteuerung am Wechsel zwischen *Sichten* bezeichnet und gegen den Modus der Nutzerinteraktion zur Bearbeitung von Daten in so genannten *Bearbeitungsdialogen* abgegrenzt.

In [SRB96] entwickeln Rossi et al. den Begriff der Navigation als *Sicht* auf das konzeptionelle, also fachliche Modell (siehe Abbildung 2.1) was einer Anwendung zu Grunde liegt. Sie verstehen *Sicht* dabei tatsächlich im Sinne des Begriffs im Bereich der Datenbanken. Dies korrespondiert mit der in [EOOH94] vorgeschlagenen Typisierung von Sichten, nach der etwa *Einzelsichten* einer Selektion entsprechen oder *Containersichten* einer thematischen Aggregation. Die sich ergebenden unter Umständen unterschiedlichen Navigationsmodelle dienen also dem Zweck der Reorganisation der rein fachlichen Strukturen. Die im Sinne des adressierten Fachbereichs kanonischen Modelle für Abläufe und Aufbaustrukturen werden dabei so transformiert und in der Regel erweitert, dass das sich ergebende Modell und in letzter Konsequenz die daraus resultierende

[1] In einer Übersetzung des Autors

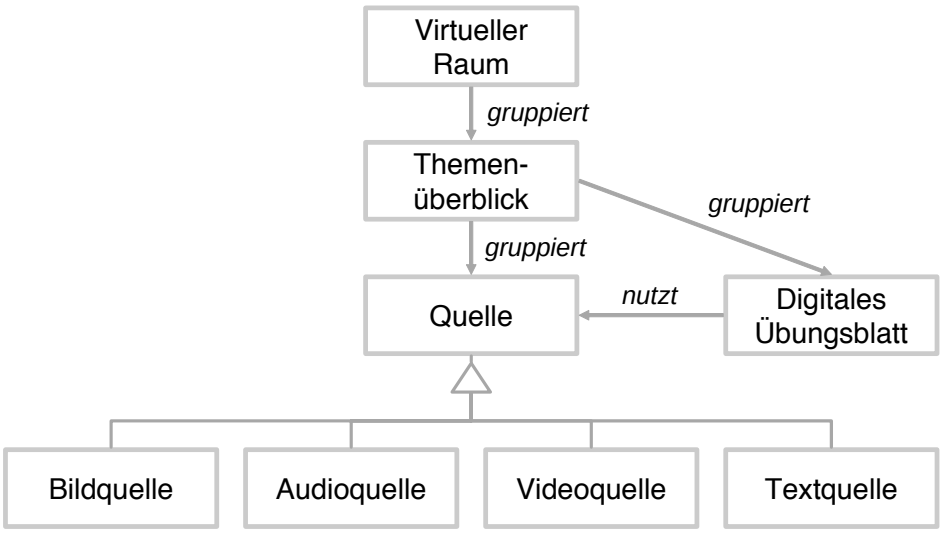


Abbildung 2.1.: Ausschnitt eines exemplarischen konzeptionellen Modells

Software-Umsetzung, die Anforderungen der Nutzer erfüllt. Rossi et al. gehen dabei davon aus, dass je nach Zielgruppe und einem entsprechenden Einsatzzweck auch unterschiedliche Navigationsmodelle für ein und dieselben fachliche Struktur entwickelt werden können.

So werden die auf den Inhalten definierten Abläufe beispielsweise konkreten Aufgaben angepasst oder in Abhängigkeit von der Zielgruppe ganz neue Ordnungskriterien zur Steuerung des Ablaufs und der Gliederung eingeführt. So kann etwa einfach durch die Zweckanpassung der Sicht die Effizienz der Nutzung etabliert werden. Über neu geschaffene Zusammenhänge und gesteuerte Ablaufregime können aber auch übergeordnete Intensionen, wie ein didaktisches Vorgehen und eine entsprechend gezielte Führung im System manifestiert werde.

Die Korrespondenz zwischen den Abbildungen 2.1 und 2.2 verdeutlicht den Zusammenhang zwischen fachlichem, konzeptionellem Hintergrund einer Anwendung und der Abbildung in Navigationsstrukturen einer interaktiven Anwendung. Die Darstellung des Navigationsmodells ist dabei lediglich von [SRB96] inspiriert. Auf formale syntaktische Präzision der dort entwickelten Modellierungstechnik wird hier zu Gunsten der Anschaulichkeit verzichtet. Inhaltlich orientiert sich das Beispiel an den Anforderungen des in dieser Arbeit vorgestellten Evaluationsprojekts (siehe Kapitel 4 auf Seite 233).

Das in Abbildung 2.2 wiedergegeben Navigationskontextschema ordnet dabei auf seiner rechten Seite jedem fachlichen Konzept einen so genannten Navigationskontext [SRB96] zu, der die Gliederung der (Bildschirm-)Präsentation der einzelnen fachlichen Elemente in Form von Knoten und deren Beziehungen, etwa über Links (Pfeile) oder Schachtelungen beschreibt. Für das Beispiel bedeutet das, dass es definierte Lernwege gibt, die mehr oder minder feste Abfolgen von *Themen* vorgeben, um etwa eine sukzessive Ver-

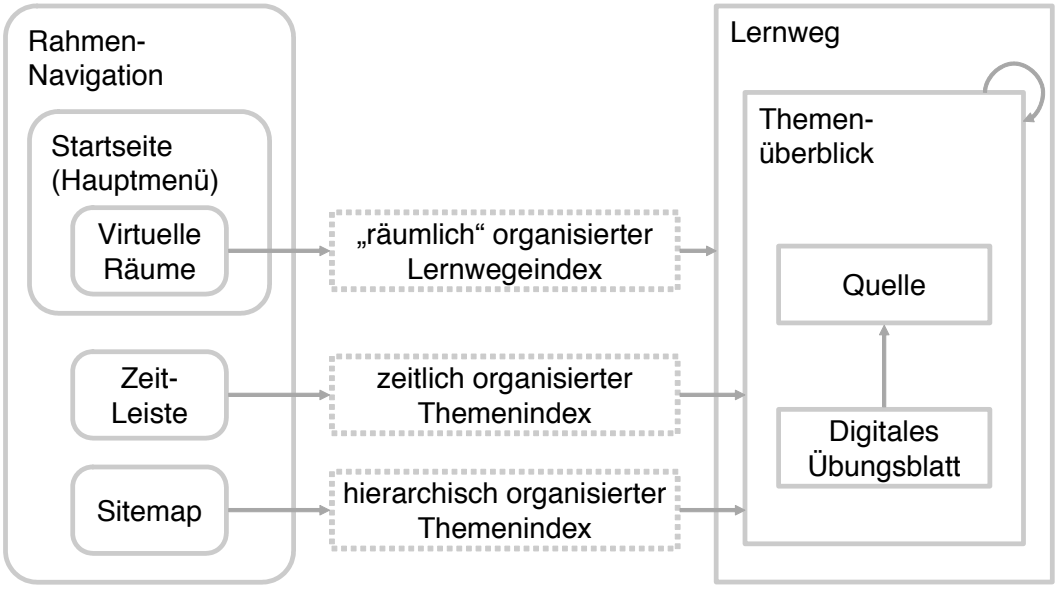


Abbildung 2.2.: Beispiel für ein Navigationskontextschema in Anlehnung an [SRB96]

mittlung von Fakten entlang einer didaktischen Konzeption zu führen. Diese *Themen* können dabei wiederum in sich geschachtelt sein – also Unterthemen von gleichem Aufbau und Erscheinungsbild besitzen. *Themen* fassen *Quellen* und *digitale Übungsblätter* zusammen und geben inhaltlichen Überblick.

Die aufrufenden Steuerelemente und deren Anordnung in Präsentationsbereichen (in Menüs etc.), sind als Platzhalter auf der linken Seite des Schemas durch Boxen mit abgerundeten Ecken dargestellt. Komplexe Ansichten zu Zwischenschritten für die gezielte Auswahl einzelner Ausprägungen der Kontexte sind gestrichelt wiedergegeben. In Analogie zur in [SRB96] entwickelten Darstellungsform, ist das die vereinfachte Abbildung von immer wieder ähnlich aufgebauten Übersichtsdarstellungen von Sammlungen von Knoten.

Diese Darstellung macht deutlich, dass sich für den Aufbau einer aufgabenabhängigen Navigationsstruktur (z.B. zur schrittweisen Wissensvermittlung) tendenziell Konzept-Konzept-Beziehungen ergeben, die sich vom kanonischen konzeptionellen Aufbau unterscheiden oder zumindest alternative Perspektiven darauf etablieren. Die Navigation muss als eigenständiger querschnittlicher Aspekt der Konstruktion eines interaktiven Systems über die rein fachliche Strukturierung hinaus betrachtet und bewusst entworfen werden.

Der in dieser Arbeit vorgestellte Entwicklungsansatz bedient sich nicht der in [SRB96] bzw. [SR98] entwickelten modellgetriebenen Entwicklungsvorgehen. Die dort definierten Modellierungstechniken erscheinen für den Einsatz bei der intensiven endnutzerorientierten Entwicklung als ebenso wenig praktikabel wie alle anderen formalisierenden

Herangehensweisen. Die explizite Unterstützung der prototypengetriebenen Entwicklung erfolgt hier durch die Vorgaben, die das Technologie-Framework mit dem so genannten Navigationsrahmen macht (siehe dazu den entsprechenden Abschnitt auf Seite 137). Explizite konzeptionelle Phasen der Entwicklung werden anstatt durch etablierte Modellierungstechniken unterstützt durch die kollaborative Ausarbeitung von Mock-Ups und gegebenen Falls die Bereitstellung abstrahierender Werkzeugsichten der Prototyping-Umgebung.

2.1.2. Qualitätsmerkmale

Im Folgenden wird eine Charakterisierung interaktiver, direkt reaktiver Software-Systeme entlang gängiger Kategorien (nach [Kro97] Seiten 21 ff.) vorgenommen. Dabei werden lediglich die Eigenschaften betrachtet, die die Spezifik der betrachten Produkte ausmachen. Die getroffenen Aussagen stützen sich neben der Analyse der Literatur auf die qualitative Untersuchung entsprechender Projekte und dabei schwerpunktmäßig die Fallstudie *LeMOLernen*.

Neben der Ökonomie der Entwicklung und des sich ergebenden Produktes, sind für die betrachteten Entwicklungsbemühungen oft auch pädagogische, politische Aspekte oder auch Forschungsinteressen (Erkenntnisprozess) von zentralem Interesse und damit (häufig orthogonale) Dimensionen für Zielgrößen der Optimierung.

Mehr als bei allen anderen Software-Kategorien hängt der Erfolg interaktiver, direkt reaktiver Systeme von der Akzeptanz durch ihre Nutzer ab. Dabei ist der Einsatz häufig sogar von der intrinsischen Motivation oder dem reinen Interesse des Anwenders abhängig. So lässt sich ein Fokus setzen innerhalb des von [Nie93] (Seite 25) beschriebenen Modells zur Akzeptanz von Software-Systemen. Der Ausschnitt dieses Modells gilt in besonderem Maße für die hier betrachtete Software und unterstützt eine weitere Konkretisierung des Systembegriffs: so leben die hier betrachteten Systeme von ihrer soziale Akzeptanz. Inhalte oder etwa auch die grafische Gestaltung dürfen den Wertvorstellungen, ethischen, moralischen oder gar religiösen Grundsätzen von potentiellen Nutzern nicht zuwiderlaufen. In Bezug auf alles was der Anwender im Rahmen der Mensch-Maschine-Interaktion an persönlichen Dingen offenbart, müssen Diskretion und Integrität zum Schutz der Persönlichkeitsrechte garantiert sein.

Um die praktische Akzeptanz der hier interessierenden Systeme zu erzielen, ist ein klarer Schwerpunkt zu setzen: innerhalb der *Nützlichkeit* wird die *Nutzerfreundlichkeit* darüber entscheiden, ob Anwender diese Systeme nutzen. Faktoren wie die Kompatibilität und die Zuverlässigkeit spielen dabei im Gegensatz etwa zu missionskritischen betrieblichen Anwendungen eine vergleichsweise untergeordnete Rolle. Während einem Sachbearbeiter beispielsweise in der Regel keine Wahl bleibt, ob er eine Geschäftsanwendung nutzt, ist es dem Besucher eines Museums völlig frei gestellt, ob er sich für ein Kiosksystem interessiert und von sich aus motiviert ist in einen längeren Dialog mit diesem zu treten.

Nutzerfreundlichkeit

Durch die grundsätzliche Fokussierung der hier betrachteten Systeme auf die Interaktion mit dem Endnutzer, ergibt sich bezüglich der Ergonomie der Nutzeroberflächen eine Schwerpunktsetzung innerhalb der Anforderungsdiskussion.

Was [Nie93] (Seite 26) als Anforderungen an die Benutzbarkeit von Software im Allgemeinen zusammenträgt, macht den eigentlichen Charakter der hier betrachteten Software-Kategorie zu großen Teilen aus. Die Schwerpunktsetzung auf diesen Aspekt soll als zentrales Unterscheidungsmerkmal zur Abgrenzung zu betrieblichen Anwendungen oder gar zu eingebetteten Software-Systemen dienen.

Die folgenden fundamentalen Eigenschaften interaktiver Systeme nach [Nie93] müssen also in den zu entwickelnden Zielsystemen besondere Berücksichtigung finden. Dies wiederum impliziert die Notwendigkeit, den Entwicklungsprozess so zu gestalten, dass die mit den Eigenschaften verbundenen Anforderungen erfüllt werden können. Der Entwicklungsprozess muss dabei der Tatsache Rechnung tragen, dass diese Merkmale für einen partikulären Einsatzkontext (Zielgruppe, Betriebsumgebung, etc.) nur sehr schwer vollständig, korrekt und widerspruchsfrei antizipiert oder gar formal spezifiziert werden können. Damit ist auch eine Evaluation der Entwicklungsergebnisse etwa in einem herkömmlichen Abnahmeprozess mit einer Prüfung gegen eine Definition kaum möglich. Nur eine kontinuierliche Determinierung des Entwicklungskontexts durch die Bedarfe bzw. Bedürfnisse der Anwendung erlauben die erfolgreiche Etablierung der folgenden Eigenschaften:

Erlernbarkeit: Nach dem Verständnis von [Nie93] muss es dem Nutzer primär sehr einfach möglich sein, die Bedienung des Systems beherrschen zu lernen. Er muss im Idealfall in der Lage sein, die Möglichkeiten der Interaktion, seine Handlungsoptionen und die entsprechenden Reaktionen des Systems intuitiv zu erfassen, in die zu unterstützenden fachlichen Abläufe einzubinden und ggf. nachhaltig zu routinisieren. Auf Grund der Tatsache, dass die hier betrachteten Systeme Informationen vermitteln oder gar explizite Lernprozesse bezüglich fachlicher Inhalte unterstützen werden, soll dieses Verständnis ausgeweitet werden. Auch bezüglich der präsentierten inhaltlich fachlichen Aspekte soll allgemein eine möglichst hohe kognitive Qualität, also eine möglichst gute Verständlichkeit und Eingängigkeit, gefordert werden (vgl. [Wei02] Seite 57). Sowohl die konkret vorauszusetzende Kompetenz der potentiellen Zielgruppe in Bezug auf die Nutzung des Mediums als auch das fachliche Vorwissen müssen dabei Berücksichtigung finden, um Frustration durch Überforderung oder aber Unterforderung zu vermeiden. Eigenschaften wie die *Selbstbeschreibungsfähigkeit* und die *Erwartungskonformität* (vgl. [fN03]) sind entscheidende Voraussetzungen, einer effektiven Erlernbarkeit der Nutzung von Software.

Effizienz und Effektivität: Ein im Umgang mit dem System routinierter Nutzer sollte in der Lage sein alle zu bewältigenden, durch das System zu unterstützenden Aufgaben möglichst effizient zu erledigen. Das bedeutet für die hier betrachtet

Domäne primär, dass die Rezeption von Informationen (einschließlich des Auffindens, Repetierens, trainierenden Anwendens etc.) und die Aneignung von Wissen mit möglichst geringem Zeitaufwand und ohne Frustrationen möglich sein sollten. Neben der bestmöglichen, also effizienten Unterstützung dieser Aufgabenstellung ist es jedoch auch wichtig, dass das System *effektiv* genutzt werden kann. Das heißt es ist auch entscheidend, die richtigen Aufgaben mit der richtigen Zielsetzung zu adressieren. Das Vorwissen des Nutzers und die individuellen Ziele (in der Regel explizite Lernziele) müssen bei der Gestaltung der Nutzerschnittstelle Berücksichtigung finden.

Wiedererkennungswert: Nachhaltigkeit in der Nutzbarkeit interaktiver Software ist dadurch zu etablieren, dass die Nutzerschnittstelle so gestaltet wird, dass Anwendern auch nach einer längeren Periode ohne Nutzung ein Wiedereinstieg leicht möglich ist. Dabei kann davon ausgegangen werden, dass Systeme, deren Nutzung leicht zu erlernen ist auch leicht zu erinnern sind.

Fehler und Frustration: In der Regel ist davon auszugehen, dass die hier betrachteten Systeme in Bezug auf die Gefährdung von Leib und Leben von Menschen eine eher geringe Kritikalität besitzen. Nichtsdestoweniger entscheidet die Fähigkeit eines Systems Ausnahmesituationen angemessen zu handhaben über den Erfolg. Es sollten grundsätzlich wenige Bedienschritte dazu führen können, dass das angestrebte Ziel nicht mehr unmittelbar erreicht werden kann. Die Möglichkeit katastrophale Fehler auszulösen, Fehler die nicht direkt durch den Nutzer korrigiert werden können und (im schlimmsten Falle unbemerkt) zu falschen Arbeitsergebnissen führen, sind ganz zu vermeiden (vgl. [Nie93] Seiten 32 f.) Damit kommt dem klassischen Qualitätsmerkmal der *Robustheit* bezogen auf die Toleranz gegenüber (fehlerhaften) Nutzereingaben als der zentralen Determinante des Systems ein besonderer Stellenwert zu.

Befriedigung Insbesondere die kognitive Qualität, also die Verständlichkeit der Nutzerschnittstellen, aber auch die ästhetische Qualität sind entscheidend dafür verantwortlich, dass das System auch subjektiv, angenehm zu benutzen ist.

Um ein System gemäß dieser Aspekte möglichst optimal zu gestalten, sind beispielsweise die folgenden Anforderungen zu erfüllen (vgl. [Pre99] (auf Seite 302)):

Grafische Darstellung Die Interaktion zwischen Nutzer und den hier interessierenden Systemen basiert zentral auf einer grafischen Nutzerschnittstelle. Auch die präsentierten Inhalte setzen innerhalb der Multimedialität einen Schwerpunkt auf visuelle Medien. Die Bedeutung der Visualisierung bei der Informationsextraktion und Vermittlung unterstreicht auch [BYRN99] (Seite 258). Demnach ist offensichtlich, dass viele Informationen durch eine grafische Darstellung viel leichter und schneller durch den Nutzer zu erfassen sind. So werden auch wissenschaftliche Zusammenhänge (eine zentrale der hier adressierten Domänen) erst durch grafische Visualisierungen, also etwa neue bildgebende Verfahren erkennbar. Selbst die Präsentation von Tondokumenten und Texten bzw. die Unterstützung der

Auseinandersetzung damit wird immer unterstützt durch visuelle Hilfsmittel, wie farbliche Hervorhebungen etc.

Direkt manipulative Interaktion (Konstruktion): Die aktive Auseinandersetzung mit den präsentierten Inhalten sollte möglich sein. Insbesondere im Zusammenhang mit der Darstellung bzw. der Vermittlung von Handlungswissen ist es sinnvoll, die präsentierten Methoden und die damit verbunden Tätigkeiten aktiv (ggf. simuliert) nachvollziehbar zu machen. Auch die Einbindung von Inhalten in spielerische Anwendungssituationen kann diesen Zweck erfüllen.

Informatives Feedback: Dem Nutzer muss jeder Zeit verdeutlicht werden, welche Konsequenzen seine Interaktion mit dem System hat. Unklarheiten bezüglich von durch den Nutzer ausgelösten Zustandsänderungen darf es nicht geben. Der Nutzer behält so das Gefühl, die Kontrolle über das System zu haben. Der Zusammenhang zwischen Eingaben und Ausgaben muss transparent und nachvollziehbar sein [BYRN99] (Seite 258).

Multimodalität: Eine Möglichkeit, die Software an individuelle Bedürfnisse und Gewohnheiten anzupassen besteht darin, die gleiche Funktionalität über verschiedenen Formen (Modi) der Interaktion bzw. die gleichen Inhalte über unterschiedlich Formen der Präsentation zugänglich zu machen.

Trennung von Präsentation und Schichten der Verarbeitung: Aus technologischer Perspektive erreicht man die für die Individualisierung benötigt Flexibilität der Software durch die saubere Trennung der Komponenten der Nutzeroberfläche von denen der Verarbeitung bzw. der Infrastruktur (Persistenz und Kommunikation) durch klare Schnittstellen und die Anwendung der dafür konzipierten Entwurfsmuster.

Standardkonformität: Die Akzeptanz des Systems kann von den Gewohnheiten der Nutzer profitieren – mediale Vorkenntnisse ausnutzen, indem standardisierte Bedienmuster und Style Guides verwendet werden, um die Software weitgehend erwartungskonform zu gestalten.

Individualisierbarkeit: Die Zielsoftware sollte den Anforderungen der Zielgruppe möglichst gut gerecht werden. In Abhängigkeit von Vorkenntnissen (fachlich wie auch bezogen auf die Mediennutzung), von Wahrnehmungsgewohnheiten sowie angestammten Handlungsmustern und Begriffsbildungen sollte die Software möglichst schon während der initialen Entwicklung individuell ausgestaltet werden. Um jedoch auch auf Varianzen innerhalb größerer, nicht präzise vorab zu charakterisierender Zielgruppen eingehen und insbesondere die progressive Entwicklung ihrer Kompetenz bezüglich der Nutzung des Systems berücksichtigen zu können, ist es wünschenswert, auch das ablauffähige System während des Einsatzes an die sich individuell ergebenden Bedarfe anpassen zu können. Im Idealfall sollte dies über einfache Schnittstellen selbstständig durch die Anwender vorgenommen werden können – ohne dass dazu vertiefte technische Kenntnisse oder gar Programmierfähigkeiten vorausgesetzten werden müssten.

Informationsqualität

Eine zentrale Aufgabe der hier untersuchten Systeme ist die Kodifizierung des Wissens von Fachleuten zu dem Zweck, dieses als Information für den Nutzer zugänglich zu machen, so dass dieser wiederum in der Lage ist, die Informationen in den Kontext des eigenen Wissensschatzes einzugliedern – sich die Informationen als Wissen anzueignen. So ist in Erweiterung der Interpretation klassischer Merkmale von Software wie der Angemessenheit, Richtigkeit, Reife oder der Fehlertoleranz bezogen auf technologisch funktionale Aspekte des Systems, die Qualität der fachlichen Informationen [Wei02] (Seite 56) adäquat zu charakterisieren. Verantwortlich für die Sicherstellung der Qualität auf dieser Ebene sind in der Regel die Fachredaktion oder die Autoren – deren Arbeit es mit der technischen Ausgestaltung zu verzahnen gilt. In Anlehnung an [Wei02] (Seite 57) soll Informationsqualität durch die hier betrachteten Systeme gemäß der folgenden Dimensionen berücksichtigt werden:

Interaktionsqualität: Die Bedienabläufe, die dem Nutzer die Informationen zugänglich machen, müssen seinen individuellen Bedarfen bzw. Bedürfnissen derart entgegnen kommen, dass er die ihn interessierenden Inhalte möglichst effizient auffinden, rezipieren und verinnerlichen kann. Die Abläufe sollten dabei an seine Gewohnheiten bzw. an die in der Domäne üblichen Tätigkeiten angelehnt sein bzw. diese optimal unterstützen.

Medienqualität: Medien wie etwa Audio-, Video- oder Text-Dokumente sollten in einer Qualität angeboten werden, die die Rezeption und auch die Weiterverarbeitung bestmöglich unterstützen. Dabei sind die angestrebten Einsatzszenarien und die vorausgesetzten Ressourcen zu berücksichtigen.

Konvergenz: Um den entscheidenden Vorteil des Einsatzes von Informationstechnologie bei der Präsentation und Vermittlung komplexer Zusammenhängen ausnutzen zu können, ist die Konvergenz unterschiedlicher Medien bewusst zu gestalten. Oft verdeutlicht erst das Zusammenspiel unterschiedlicher formen der Präsentation (z.B. das Tondokument in Verbindung mit dem Transkript des gesprochenen Textes) neue Zusammenhänge.

Ästhetische / stilistische Qualität: Die internistische Motivation des Nutzers, sich mit dem System auseinanderzusetzen, hängt wie beschrieben, auch vom subjektiv empfundenen Grad der Befriedigung ab [Nie93] (Seite 26). Dafür ist zu großen Teilen auch die Gestaltung der Oberfläche verantwortlich und die Wirkung auf das ästhetische Empfinden des Nutzers. Aber auch kulturelle Konnotationen, wie die Verwendung religiös belegter Farben oder eines zielgruppengerechten Stils der Ansprache in Texten sind dabei zu beachten, ebenso wie die Konsistenz der verwendeten Symbolik und die Einheitlichkeit bzw. Durchgängigkeit des Screendesigns.

Kognitive Qualität: Letztere Eigenschaft und eine konsistente Repräsentation leistete darüber hinaus auch einen Beitrag zur Verständlichkeit und Interpretierbarkeit der angebotenen Informationen.

Konsistenz: Die Informationen müssen entlang der folgenden Eigenschaften konsistent sein:

- Sicherheit,
- Vertraulichkeit (persönlicher Daten der Nutzer), Vertrauenswürdigkeit,
- inhaltlich Präzision,
- Objektivität,
- Reputation, Glaubhaftigkeit,
- Relevanz,
- Aktualität,
- thematische Vollständigkeit

Strukturierung: Die Struktur der Information und damit in der Regel auch der Navigation ist ebenfalls in Abhängigkeit von den individuellen Bedarfen der Anwendungsdomäne und den Zielsetzungen der Autoren zu Gestalten. Neben fachsystematischen Gliederungen werden dabei häufig auch räumliche oder auch zeitliche Strukturen – Metaphern ganzer Mikrowelten (vgl. [Sch97] Seiten 50 f.) – Anwendung finden. Dabei definiert die Qualität wiederum die Effizienz mit der der Nutzer der Syntax der Gliederung eine Bedeutung abgewinnen kann (Inhalte finden, verstehen, verinnerlichen etc.) (vgl. [Sch97] Seiten 44 f.). Die Problematik in der Nutzung von Hypertextstrukturen besteht im Finden der Balance zwischen klaren aber einseitigen Strukturen auf der einen Seite und möglichst vernetzten Informationen auf der anderen – verbunden mit dem Risiko sich in diesem Hyperspace sprichwörtlich zu verlieren [Fer03] (Seite 12).

Methodische / didaktische Qualität: Bewusst auszuprägen sind auch die Merkmale, die das Lernen mir dem System unterstützen: dazu gehören Elemente der Motivation, der Führung und des Feedbacks. Entscheidend ist ferner die Vollständigkeit und Korrektheit des Präsentierten in Bezug auf Vermittlungsziele – idealerweise ausgehend von der Vorbildung des Rezipienten.

Weitere, klassische Qualitätsmerkmale

Neben der Nutzerfreundlichkeit und der Informationsqualität, deren Konkretisierungen den Charakter der hier betrachteten Systemklasse am deutlichsten prägen, erfahren weitere der klassische Software-Qualitätsmerkmale erweiterte Interpretationen, bezogen auf die spezifischen Anforderungen, interaktiver, direkt reaktiver Software-Systeme.

Leistung / Effizienz: Der Einsatz von Medien wie Video oder Audio aber auch die Handhabung intensiver Nutzerinteraktionen erfordern grundsätzlich intensiven Ressourcen-Einsatz. Der Speicherbedarf aber auch die Rechenleistung, die etwa zur Dekodierung digitalen Medienmaterials benötigt wird, steht dabei noch immer

häufig im Widerspruch zur eingesetzten Infrastruktur. So lässt sich beispielsweise an Schulen beobachten, dass mediale Systeme kaum nutzbar sind, da beispielsweise Reaktionszeiten bei Nutzerinteraktionen zu lang sind oder aber Animationen oder Videos nicht flüssig wiedergegeben werden können. Die Intention des Präsentierten wird verfehlt (siehe oben: Interaktionsqualität). Die hier betrachteten Systementwicklungsbemühungen können eher selten von der Verfügbarkeit neuester Hardware ausgehen. Im Gegensatz zum allgemeinen Markt für Computer-Spiele muss hier der Spagat geschafft werden zwischen aktueller medialer Qualität (angemessenen Antwortzeiten, zeitgemäße Interaktions- und Darstellungsformen etc.) und begrenzten Ressourcen.

Wartungsfähigkeit: Neben der üblichen korrigierenden Fehlerbeseitigung, der Anpassung an veränderte Rahmenbedingungen und der Erweiterung im Sinne der Verbesserung des Produkts bzw. der Berücksichtigung neuer Anforderungen [Kro97], werden die hier betrachteten Systeme insbesondere auch bezogen auf die präsentierten Inhalte regelmäßiger Pflege und Erweiterung bedürfen. Eigenschaften wie Aktualität und inhaltliche Konsistenz der Informationen sollten sich dabei idealerweise im laufenden Betrieb sicherstellen lassen. Transparente Schnittstellen und eine explizit für Änderungen ausgelegte Architektur werden benötigt. Über sauber strukturierten, gut lesbaren und dokumentierten Code hinaus wird dazu ein integriertes Entwicklungskonzept benötigt, dass auch die Entwicklung der Kompetenz von nichttechnischen Stakeholdern bei der Ausführung von Wartungsarbeiten berücksichtigt.

Evolutionsfähigkeit: In Erweiterung der Forderung nach einfacher und damit effizienter Wartbarkeit sollen die Zielsystemen, der hier präsentierten Entwicklungsmethodik, über die Entwicklungszeit hinaus evolutionsfähig bleiben. Das heißt, sie sollen einerseits bezüglich der Inhalte ständig erweitert werden können. Es soll aber auch möglich sein, auf die durch die Nutzung des Systems induzierte Weiterentwicklung der Kompetenzen und damit des Bedarfs der Zielgruppe, mit funktionalen Erweiterungen des Systems reagieren zu können. Auch dabei wird angestrebt, Änderungen möglichst einfach vornehmen zu können. Es werden Entwicklerschnittstellen angestrebt, die die Nutzer selbst oder zumindest Domänenexperten – ggf. auch ohne vertiefte technologische Kenntnisse – in die Lage versetzen, die Anwendung an die individuelle Entwicklung ihrer Fähigkeiten und damit Bedürfnisse anzupassen.

Wiederverwendbarkeit: Ein großer Teil der Gestaltungsleistung interaktiver direkt reaktiver Systeme steckt in der Aufbereitung bzw. der Erstellung der Inhalte. Zum Schutz der damit verbundenen Investitionen sind Konzepte erforderlich, um den wiederholten Einsatz in anderen Systemen – etwa für andere Zielgruppen oder andere Einsatzkontexte – zu ermöglichen. Die Verwendung von Standardformaten und eine Entkopplung der Aspekte des eigentlichen Inhalts von der verarbeitenden Funktionalität sowie von der Darstellung sind dazu unabdingbar. Aber auch die Wiederverwendung funktionaler Komponenten im klassischen Sinne wird im vorgestellten Konzept mit Hilfe eines Frameworks instrumentalisiert, um durch die resultierende Reduktion des technischen Entwicklungsaufwandes den Fokus

der Entwicklungsbemühungen von den technischen auf die fachlich inhaltlichen Aspekte verlagern zu können.

Portabilität: Bezogen auf die Plattformabhängigkeit – also die Abhängigkeit der Zielsoftware von der Betriebssystemumgebung und der zu Grunde liegenden Hardware – setzen reine Web-Anwendungen zunehmend die Maßstäbe auch für interaktive Software im Allgemeinen. Die hier betrachtet Systemklasse versucht dementsprechend dem Anspruch zu genügen, überall zu funktionieren, wo aktuelle Web-Anwendungen (innerhalb eines Browsers) ablauffähig sind. Dem erweiterten Anspruch bezüglich der Interaktivität geschuldet, wird allerdings eine als Browser-Plugin verbreitet, für die meisten Plattformen verfügbare Laufzeitumgebung vorausgesetzt.

Interoperabilität: Die Zusammenarbeit unterschiedlicher Systeme wird auf mehreren ebenen benötigt: Zum einen zur Etablierung kommunikationsbezogener Interaktionskonzepte. So wird es Nutzern ermöglicht, Informationen über Systemgrenzen hinweg auszutauschen. Im Rahmen der Entwicklung des Evaluationssystems war es eine explizite Anforderung, die innerhalb des Systems genutzten bzw. bearbeiteten Medien auch im Zusammenhang mit anderen Anwendungen nutzen zu können. Die Verwendung von Standardformaten muss es beispielsweise ermöglichen, beliebige Standardwerkzeuge zur Bearbeitung von Dokumenten nutzen zu können. Aber auch die Wiederverwendung von Inhalten und Funktionalitäten bedarf der Entkopplung durch Standardprotokolle und -formate.

2.1.3. Entstehungs- und Anwendungskontexte

Aus den spezifischen, im vorangegangenen Abschnitt beschriebenen Anforderungen an interaktive, direkt reaktive Systeme, sowohl bezügliche der Struktur und des Verhaltens als auch in Bezug auf die zu erfüllenden Qualitätsansprüche, ergeben sich für die Software-Klasse charakteristische Konstellationen und Abläufe während der Entstehung. Wobei der Kontext des eigentlichen Einsatzes der zu entwickelnden Systeme dabei nicht nur als beobachtetes bzw. berücksichtigtes Objekt eine Rolle spielt, sondern in der Regel aktiv als Subjekt auf den Entstehungskontext ein- bzw. zurückwirkt. Dies schließlich macht den transdisziplinären, in Abgrenzung zum multidisziplinären Charakter der Soziographie innerhalb der Stakeholder des gesamten Produktlebenszyklus' aus. Diese Charakteristik gilt es in der vorgeschlagenen Unterstützung zu berücksichtigen.

In gleicher Weise wie für die Definition allgemeiner Anforderungen an die hier betrachteten Systeme wirkt auch hier die Tatsache, dass es für viele Fachbereiche wenige gesicherte Erkenntnisse über das Lernen mit Hilfe moderner Computer gibt ([Geu00] Seite 135). Entsprechend unbestimmt gestaltet sich in der Regel der Entwicklungsprozess. Sowohl qualitative, in Interviews gewonnene Aussagen aus bezüglich der Domäne sehr

unterschiedlichen aber bezogen auf die Zielstellung typischen Projekten belegen dies[2]. Eine im Kontext der vorliegenden Arbeit im Rahmen einer Diplomarbeit entstandene Untersuchung [Roo04]) auf der Grundlage einer strukturierten Befragung systematisiert diese Erkenntnisse.

Die Ergebnisse der quantitativen, nicht repräsentativen Analyse [Roo04] deuten daraufhin, dass es oft schwierig ist, Ziele im Sinne der Lösung partikulärer Probleme zu definieren, weil die Probleme nicht klar gefasst werden können und die technologischen Gestaltungsmöglichkeiten für die Lösung dem Großteil der Beteiligten, insbesondere den Fachleuten, die am ehesten den Problemraum verinnerlicht haben, unbekannt sind. Insbesondere im Kontext des Evaluations-Projektes konnte beobachtet werden, dass sich die Probleme erst in der Auseinandersetzung mit der Lösungsfindung und der permanenten Rückkopplung zur Anwendung entfalten. Damit werden sowohl Probleme als auch deren Lösungen erst im eigentlichen Entwicklungsprozess emergent im Sinne des in [Ste04] entwickelten Emergenzbegriffs. Typische hier adressierte Entwicklungsprozesse sind mithin im Wesentlichen selbst Erkenntnis- und Lernprozesse [FRS89] sowohl für Fachexperten und Nutzer (sie entdecken Gestaltungsspielräume und fachliche Optimierungspotentiale) als auch für die technischen Entwickler (sie verinnerlichen die Bedarfe der Domäne). Bei der Gestaltung dieser Prozesse zu berücksichtigen sind soziale und kommunikative Faktoren. Erst ein konstruktiver Diskurs und die reibungslose Rückkopplung zwischen der Anwendung des entstehenden Produkts sowie der damit verbundenen Tätigkeit auf der einen Seite und der Ausgestaltung des Zielsystems auf der anderen, ermöglichen die benötige Innovation – die Emergenz im Sinne von [Ste04].

Produktion vs. Entwicklung

Abbildung 2.3 verdeutlicht die Komplexität des Zusammenspiels der am Entwicklungsprozess beteiligten Rollen und der von ihnen typischerweise vertretenen Disziplinen und die Spezifik des Einflusses der Anwendungsdomäne direkt auf den Lebenszyklus des angestrebten Produktes. Die in der Abbildung verallgemeinerte Konstellation ist die idealtypische Essenz dessen, was intensive Analysen entsprechender Projekte ergeben. Vergleichbare Konstellationen beschreibt auch [Ker01] (Seite 321 ff.) als typisch für die *Produktion* didaktischer Medien. Augenfällig ist dabei die Entwicklung des Begriffs der *Produktion*, als integrierter Prozess, der die Erhebung der Informations- bzw. Bildungsbedarfe über die Beschaffung und Verwaltung von Medien und Nutzungsrechten bis hin zur technischen Realisierung, dem Roll-Out und der Wartung den gesamten Lebenszyklus eines Produktes bestimmt. Dies erfolgt offenkundig nach dem Verständnis des

[2] Zum Beispiel:
- Dr. Jörg Caumanns (Teachware on Demand: `http://www.isst.fhg.de/englisch/content/projects/2002_1999/Teachware_on_Demand/index.html`)
- Dr. Martin Majewski (SoSi: `http://www.isst.fraunhofer.de/deutsch/inhalt/Projektarchiv/archiv_03_04/Projekte_SoSi/index.html`)

Verlagswesens in Anlehnung an die Produktion klassischer Medien wie den Printmedien.

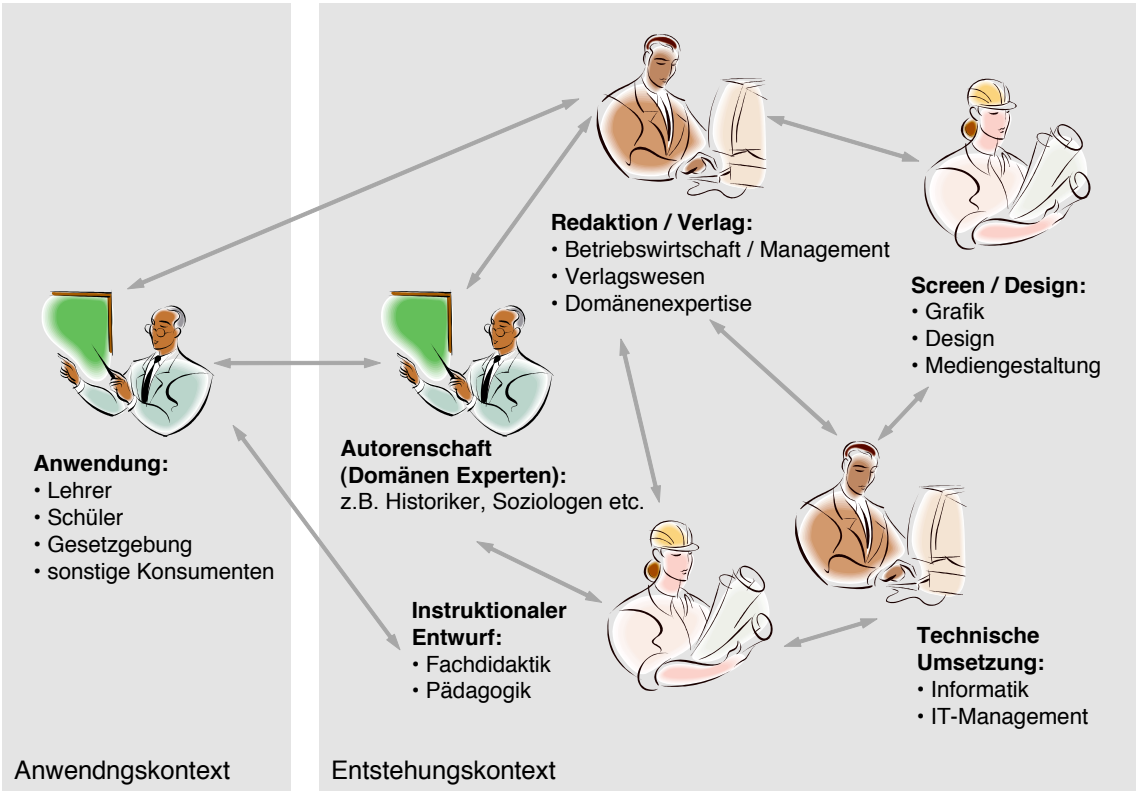


Abbildung 2.3.: Die am Entwicklungsprozess beteiligten Rollen

Insbesondere Produktionsaufgaben wie die Feststellung des Bildungsbedarfs, die Entscheidung für eine Vermittlungsstrategie aber auch die Planung und die Gestaltung des Einsatzkontexts deuten auf Konfliktpotentiale im Vorgehensverständis zwischen redaktioneller Medienproduktion und ingenieurmäßiger Entwicklung hin.

Es muss also als Herausforderung betrachtet werden, die zu erledigenden Aufgaben in der benötigten Gesamtheitlichkeit zu bearbeiten und kontrollieren zu können, ohne auf die Chancen evolutionärer Software-Entwicklungpraktiken verzichten zu müssen. Entscheidend wird sein, dass die Tendenz zur technischen Dominanz von Projekten mit rein technologischen Artefakten als Ergebnis systematisch minimiert wird zugunsten der Belange der Anwendungsdomäne. Der Enstehungskontext wird dabei kontinuierlich determiniert durch den Anwendungskontext. Um die benötigte Konvergenz zu erreichen, muss der Entwicklungsprozess des Produktes mit der Anpassung und Entwicklung sowohl der Entwicklungsorganisation als auch der Organisation der Zieldomäne verbunden werden.

Die üblichen Kernaufgaben der einzelnen in Abbildung 2.3 zusammengefassten Rollen werden nun kurz präzisiert:

Autorenschaft: Autoren sind die Träger des durch das Medium weiterzugebenden Wissens. Sie besitzen vertiefte Kenntnisse bezüglich der Fachdomäne. Dementsprechend wählen sie in Abstimmung mit den Vermittlungszielen (letztlich also auch abhängig von der Programmatik des Verlages) die zu repräsentierenden Inhalte aus, strukturieren diese im Idealfall gemäß der Vorgaben die das Instruktionsdesign bereits machen kann. Typisch ist dabei, dass die (medien-)didaktische Konzeption auch zu den Aufgaben der Autoren gehört – Fachlichkeit und Fachdidaktik bilden häufig eine Einheit. Außerdem arbeiten Autoren die Inhalte im Detail aus. Das heißt etwa, dass sie Texte verfassen, Quellenverweise auf Medienmaterial zusammentragen und Umsetzungsvorgaben (Regieanweisungen) für die technische Realisierung erstellen.

Redaktion, Produktion und Verlag: Der Verlag konzipiert das vermarktbare Produkt gemäß der Markterfordernisse (etwa auch beeinflusst durch Rahmenlehrpläne etc.) und der Programmatik des Verlags. Redakteure vermitteln dabei zwischen der inhaltlich fachlichen Expertise der Autoren und der technischen Expertise der Implementierung einschließlich des Screen-Designs, indem die Integration der Arbeitsergebnisse zu einem Produkt koordiniert wird. In der Regel wird das Gesamtprojekt durch den Verlag verantwortet und geleitet. Dazu gehören beispielsweise auch die Beschaffung von Medien und Rechten und die Organisation von Evaluation, Qualitätssicherung, Marketing und Vertrieb.

Content Aufbereitung und -Management: Es muss als eine eigene Aufgabe betrachtetet werden Inhalte aufzubereiten, strukturiert zu verwalten und zu pflegen, um etwa Medien in entsprechender Qualität und von den Autoren erstellte Inhalte für die Realisierung verfügbar zu haben. Die Einbindung der Inhalte in die Strukturen des Zielsystems einschließlich der Verwaltung von Versionen entspricht dabei sogar dem Aufgabenbereich klassischen Content-Managements.

Instruktionales Design: Die Erstellung der mediendidaktischen Konzeption in Abhängigkeit von angewandten Typen der Vermittlungsziele und -Basismodelle (Entdecken, Handlungskompetenz routinisieren, etc.), der Charakteristik der Zielgruppe sowie den Spezifika der Fachdomäne ist eine Aufgabe, die selten durch eine eigene Rolle ausgefüllt wird. Dabei birgt insbesondere die Abbildung der fachdidaktischen Konzepte in technische und mediale Konstrukte für viele Fachdomänen die meisten Unklarheiten, Risiken und somit Forschungspotentiale.

Technische Realisierung: Die Erstellung der eigentlichen technischen Lösung umfasst neben der Abbildung der fachlichen Konzepte und Anforderungen in Entwürfe und die Implementierung auch hier die Integration aller Systemteile, einschließlich der Inhalte und Medien sowie Aufgaben des Software-Tests. Wie bei der Entwicklung von Software-Produkten üblich, sind ferner die Distribution (Erstellung entsprechender Medien) bzw. Installation sowie die Wartung zu unterstützen.

Screen-Design und Ergonomie: Als eine Spezialaufgabe der technischen Realisierung kann die grafische und ergonomische Gestaltung der Nutzeroberflächen gelten. Neben der Ausrichtung auf Wahrnehmungs- und Tätigkeitsgewohnheiten der designierten Nutzergruppe sind dabei grundsätzlich die technologisch, medialen Grenzen der Realisierung, die Anforderungen der Inhaltepräsentation sowie der Didaktik und schließlich Aspekte der Produktgestaltung des Verlages (Corporate Design) zu berücksichtigen.

Der Einfluss des Nutzungskontexts

Die Eigenschaft, die die hier besprochenen Projektkontexte auszeichnet und zu einer besonderen Herausforderung macht ist die Transdisziplinarität. In Erweiterung zu den Ansprüchen üblicher Software-Projekte, die auch unter Einbeziehung der verschiedensten Disziplinen und unter dem Einfluss des Nutzerkreises stehen, betrachtet die hier vorgestellte Lösung zunächst die Risiken die sich aus ausgeprägter Transdisziplinarität ergeben, um daraus besondere Potentiale für ein Entwicklungsvorgehen abzuleiten und gezielt zu instrumentalisieren.

Neben der Multidisziplinarität, also der Mitwirkung der verschiedensten (zum Teil wissenschaftlichen) Disziplinen, impliziert Transdisziplinarität eine intensive Wechselwirkung des Entstehungskontextes mit dem Bereich der Anwendung, der wiederum von den unterschiedlichsten Disziplinen geprägt sein kann. Das bedeutet, dass Aktivitäten der Entwicklung mit denen der Evaluation und schließlich gar mit denen des tatsächlichen Einsatzes verschwimmen. Im Idealfall lässt sich der resultierende sich selbstorganisierende Prozess über das geschickte Gestalten der Rahmenbedingungen so steuern, dass er zielführend in Richtung einer individualisierten optimalen Lösung verläuft.

Dabei wirkt der Nutzungskontext auf verschiednen Ebenen auf den Lebenszyklus und das entstehende Artefakt ein. Um eine tatsächlich praxistaugliche Software zu erstellen, die in ihrer Interaktivität auch nuancierte Anforderungen an die unterstützten Bedienabläufe erfüllt, muss der Entwicklungsprozess Nutzerzentriert sein, das heißt zumindest, das Nutzer oder Nutzervertreter als Interviewpartner während der Analyse an der Entwicklung beteiligt werden. Interaktive Anwenden, für die die Akzeptanz durch den Nutzer missionskritisch ist, sollten jedoch darüber hinausgehen und das entstehende System oder auch Prototypen davon immer wieder entwicklungsbegleitend durch designierte Anwender evaluieren lassen. Ideal ist die hier angestrebte kontinuierliche evaluierende Nutzung in realen Produktivszenarien unter ständiger Abstimmung mit den Kompetenzen der Nutzer bzw. der umgebenden betrieblichen Organisation bzw. der Arbeitsorganisation.

Als Wirkung des Anwendungskontexts auf den Entwicklungskontext muss darüber hinaus der Einfluss von Reglungen und Bestimmungen betrachtet und berücksichtigt werden. Rahmenlehrpläne etwa geben vor, welche Lernziele zu verfolgen sind.

Der transdisziplinäre Charakter der hier adressierten Entwicklungszyklen ergibt sich jedoch schwerpunktmäßig auch aus der Anforderung, dass die fachbezogene, an der

Praxis der Domäne orientierte Weiterentwicklung und Individualisierung der Systeme zu den elementaren Anwendungsfällen zählen. Insbesondere Software, die explizit zur Lehre eingesetzt werden soll spielt dabei eine Sonderrolle. In dem Moment, in dem ein Lehrer, ein Coach oder beispielsweise auch ein Ausstellungsgestalter das System für die konkrete Unterrichts- oder Präsentationssituation vorbereitet, indem er Inhalte hinzufügt oder eigene Präsentations- und Interaktionsformen wählt bzw. ergänzt agiert er zwar einerseits als fachlicher Anwender aber übernimmt auch die Rolle eines Autors und damit fachlichen Entwicklers. Diese Nutzungssituation wird etwa bei [Ker01] (Seite 320) als typisch beschrieben. So verschwimmt die Grenze zwischen Entwicklung und Nutzung in einem von Kontinuum was etwa [FG04] als Ausgangspunkt für Prozesse des End-User-Development beschreibt.

Zielgruppenprofile

Eine Festlegung für vorauszusetzende Expertise-Niveaus der Stakeholder als Ausgangsbedingung für den Einsatz des in dieser Arbeit präsentierten Lösungsvorschlags soll zunächst bewusst nicht getroffen werden. Aus der Beschreibung des Entwicklungskontexts wird zumindest intuitiv deutlich, welchen Ausbildungsstand die direkten Entwicklungsbeteiligten haben. Die zu erwartenden Kompetenzen werden im Folgenden lediglich in Relation zueinander erörtert innerhalb der Dimensionen der Kompetenz nach [Nie93] (Seite 44):

- Domänenexpertise
- Technische Expertise
- Erfahrung im Umgang mit einem partikulären System

Die Feststellung des tatsächlich zu unterstützenden, konkreten Expertiseniveaus ist ein Teil der Aufgabe des Prozesses der Adaption innerhalb des durch den Lösungsvorschlag angestrebten Vorgehens. Als ein grober Rahmen zur Verdeutlichung der Verhältnisse der Expertise-Niveaus können die im Folgenden wiedergegebenen Beobachtungen aus einer Reihe typischer Projekte betrachtet werden.

Es ist davon auszugehen, dass die Vertreter der technologischen Entwicklung in der Lage sind auf professionellem Niveau Werkzeuge und Methoden der Entwicklung anzuwenden. Es soll jedoch vom allgemeinen, problematischen Fall ausgegangen werden, bei dem den Entwicklern der Technologie die Erfahrung mit der Fachdomäne noch weitgehend fehlt.

Genau umgekehrt soll es sich mit den Fachexperten verhalten. Um eine möglichst große Allgemeinheit der Fälle abdecken zu können sollen diese einerseits die Fachdomäne mit ihren Bedarfen und Konzeptualisierungen bestmöglich vertreten können. Andererseits soll davon ausgegangen werden, dass ihre Expertise im Umgang mit Informations- und Medientechnologie und insbesondere ihre Fähigkeit diese konstruktiv als Vehikel der Präsentation bzw. der Aus- und Umgestaltung von Domänenabläufen zu nutzen, kaum entwickelt sind.

Um wiederum den allgemeineren und damit schwierigereren Fall berücksichtigen zu können, soll für die eigentlichen Anwender gelten, dass sie weder über wirklich ausgeprägte Medienkompetenz im Umgang mit Informationstechnologie verfügen noch, dass sie Expertise bez. der Inhalte besitzen, die schließlich durch das zu erstellende System zu repräsentieren bzw. zu vermitteln sein werden.

Am ehesten eine vermittelnde Rolle spielen in der Regel Redakteure von Fachverlagen. Sie verfügen einen Überblick über die inhaltliche Programmatik des Verlages. Ihre Domänenkenntnisse sind in der Regel jedoch nicht auf dem häufig benötigen, spezialisierten Niveau der Fachautoren. Die technische Expertise ist bei Redakteuren insofern ausgeprägt, als dass sie den am Markt gängigen Stand der Technik kennen. Einblicke in die für die technische Realisierung notwendigen Schritte oder gar die Fähigkeit zur aktiv gestaltenden Mitarbeit fehlt ihnen jedoch in aller Regel. Auch technologische Potentiale, die über die im Fachbereich üblichen Standards hinausgehen werden häufig nicht erkannt bzw. bewusst im Sinne der vermeintlichen Vermarktungsrisiken nicht in Erwägung gezogen.

Die Dimension der Kompetenzen im Umgang mit dem Zielsystem steht hier auch zunächst für das Wissen um den benötigten Funktionsumfang und deren Einbettung in angestrebte Anwendungskontexte. So Betrifft alle Entwicklungsbeteiligten gleichermaßen, dass sie zu Beginn der Entwicklung einen in etwa gleichen Kenntnisstand über das angestrebte System und dessen Einsatzmöglichkeit haben. Versinnbildlicht ist dies in Abbildung 2.4 dadurch, dass alle Symbole für Stakeholder (gleich groß) im Vordergrund des Koordinatensystems abgebildet sind – alle eher wenig wissen über das benötigte Zielsystem. Am Ende der Entwicklung soll also die Konvergenz zwischen Fachkompetenz und technischer Kompetenz vor allem auch der Ausprägung des gemeinsamen Verständnisses für das Zielsystem dienen. Symbolisiert ist dies durch die Progression in die Tiefe des Raumes (kleiner werdende Symbole im Hintergrund).

Im Zusammenspiel: Transdisziplinarität

Bereits in den initialen Phasen der Entwicklung der Produktidee [Rup02] werden die Implikationen des transdisziplinären Charakters der betrachteten Projekte offenbar: der Impuls für die Entwicklung interaktiver Systeme geht in der Regel von einem fachlich orientierten Auftraggeber, etwa einem Fachverlag und den ihm zuarbeitenden Fachautoren, aus. Dieser besitzt wie beschrieben vertiefte Kenntnisse auf dem adressierten Problemfeld. Die Transformationen und Aufbereitung fachlicher Inhalte für die zielgruppengerechte Vermittlung und Präsentation – die Mediendidaktische Konzeption – sind oft noch stark auf Printmedien ausgerichteten. Autoren und Redakteure sind sich der Gestaltungsspielräume, die der Einsatz von interaktiven Medien dezidiert für ihre Domäne eröffnet, selten vollständig bewusst. Gerade die Möglichkeiten, die neueste Trends und Entwicklungen seitens der Informationstechnologie und der Medien bieten, sind ihnen nicht immer bekannt.

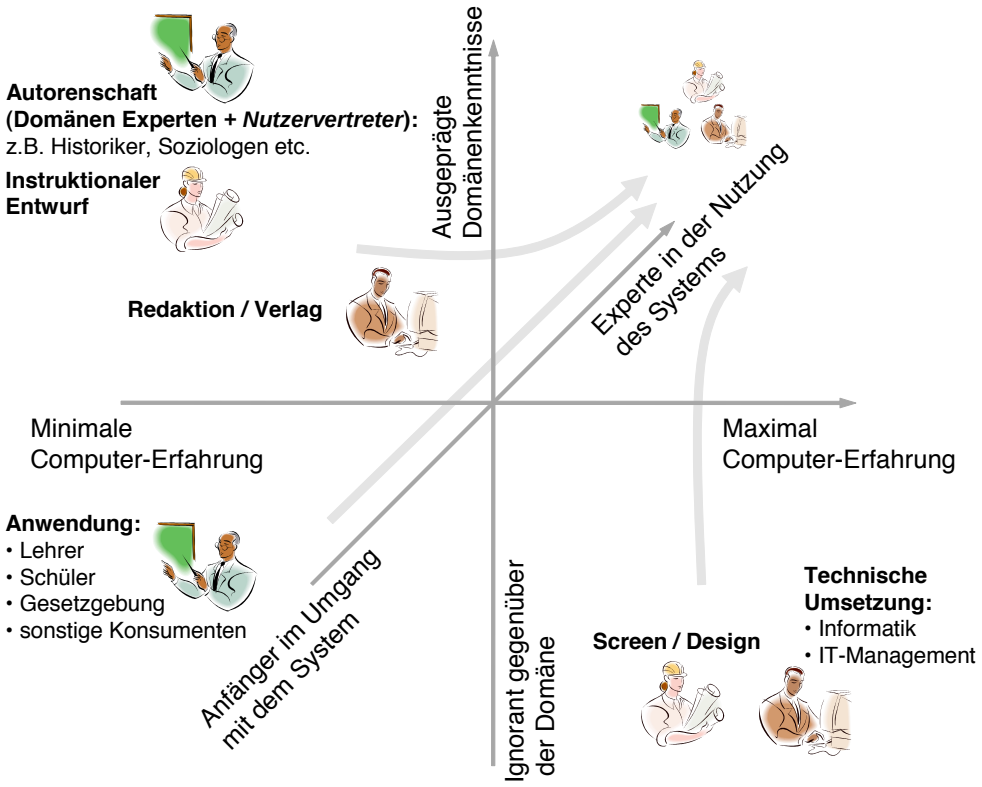


Abbildung 2.4.: Einordnung und Entwicklung der Stakeholder als Anwender nach [Nie93] (Seite 44)

Demgegenüber stehen technologisch versierte Systementwickler, die die fachlichen Anforderungen erfassen und in Konzepte der technischen Unterstützung bzw. Repräsentation übertragen sollten. Problematisch ist eben dieser Schritt der Vermittlung fachlicher Bedarfe, insbesondere auch auf Grund der Tatsache, dass ohne die Kenntnis des fachbezogenen technologischen Gestaltungspotentials Bedarfe nicht klar konturiert werden können.

Dies gilt in gleicher Weise für das Verhältnis der Technologie zum eigentlichen Anwendungskontext: wenn die Anwender nicht die Möglichkeit erhalten, ihre Bedarfe sukzessive parallel mit ihrer Kenntnis des Nutzungs- bzw. Gestaltungspotentials auszuprägen und weiterzuentwickeln, ist das Erreichen einer tatsächlichen aufgabengerechten Lösung unwahrscheinlich. Dieses Wirkgefüge zwischen Anwendung, fachlicher Expertise und technologischer Unterstützung bestimmt den gesamten Lebenszyklus der hier betrachteten Systeme in besonderem Maße.

2.2. Authoring Systems

Der Einsatz von Informationstechnologie zur Unterstützung der Lehre und zur Wiedergabe komplexer, auch nichttechnischer, fachlicher Sachverhalte geht bis in die siebziger Jahre des zwanzigsten Jahrhunderts zurück und kann damit als eine der ersten und zentralen Anwendungen von Computern in der Breite gelten. Arbeiten wie die von [Mau85] und [Kea82] zählen zu den ersten, die den Begriff des Authoring im Kontext der Erstellung entsprechender Software systematisieren. Erste Systeme zur Unterstützung des Erstellungsprozesses werden charakterisiert.

Die folgenden Abschnitte sollen einen kompakten Überblick über die wichtigsten Ergebnisse, Begriffe und Produkte zur Thematik „Autorensysteme“ geben. Dabei werden Ansatzpunkte für die in dieser Arbeit vertretene These konkretisiert, die davon ausgeht, dass die vorhandene Werkzeuglandschaft nur unzureichend auf die sehr individuellen Bedürfnisse unterschiedlicher Fachdomänen und ihrer jeweiligen Präsentationsform sowie Fachdidaktik eingehen kann. Die eingesetzten technologischen Abstraktionen, also etwa die zur Gestaltung der Nutzerinteraktion verwendete Metaphorik, ist in vielen Fällen zu weit von den Bedarfen entfernt, die sich aus den in den Domänen üblichen Handlungs- und Gestaltungsmustern ergeben. Eine direkte Unterstützung fachlicher Autoren – wie es der Name „Authoring System“ impliziert – ist damit kaum möglich. Lediglich mittelbar, durch den Einsatz von Spezialisten werden gängige Systeme einsetzbar. Entsprechende kommunikative Reibungsverluste, zumal an Disziplin- und damit häufig Verständnisgrenzen machen die Entwicklung problematisch.

Es sei kurz angemerkt, dass der englische Terminus „Authoring System“ hier bewusst für eine ganze Klasse von Software-Systemen – von einfachen Editoren-Werkzeugen bis hin zu integrierten Entwicklungsumgebungen – gewählt wurde, um der in der deutschsprachigen Literatur offenkundigen Uneinheitlichkeit des Begriffs der „Autorensysteme“ aus dem Weg zu gehen. So bezeichnet etwa [Sch97] damit eine spezielle Klasse interaktiver Anwendungen, die durch Autoren erstellt wurden. Dagegen nutzt [Ker01] den Begriff für tatsächliche Entwicklungsumgebungen.

2.2.1. Ansätze, Hintergründe und Anforderungen

Zunächst ist interaktive, direkt reaktive Software wie jede andere auch und kann entsprechend mit Hilfe der klassischen Methoden entwickelt werden. Gerade auch bei der Implementierung kann davon ausgegangen werden, dass der Einsatz von Programmiersprachen wie Java und C – die entsprechende Spezialisierung und Expertise bei den Entwicklern vorausgesetzt – die grundsätzlich größte Flexibilität bietet.

Jedoch bereits [Mau85] (Seite 557) wendet ein, dass es die Nutzung einer Programmiersprache (sogar einer spezialisierten) den nicht Nicht-Informatikern erschweren wird damit zu arbeiten, um ihre Inhalte und Anliegen zu repräsentieren bzw. zu vermitteln. Aus der Tatsache, dass die hier betrachteten Anwendungen im Kern immer wieder ähnliche Aufgaben erfüllen, leitet [Ker01] ab, dass es sinnvoll ist, vorgefertigte Module und

Bibliotheken zu nutzen, um den Entwicklungsaufwand zu reduzieren – zu dem Preis, dass „die Flexibilität der Entwicklung in Teilen eingeschränkt“ [Ker01] (Seite 363) wird. Auch die Verwendung fester Metaphern zur Organisation der Inhalte und zur Festlegung von Ablaufstrukturen eignen sich laut [Ker01] nur unterschiedlich gut für den Einsatz in verschiedenen Domänen. Der Autor kommt zu dem Schluss:

> „In der Praxis erweisen sich die gängigen Werkzeuge jedoch in vielfacher Hinsicht als wenig zufrieden stellend.“ [Ker01] (Seite 364)

Viele der weiterführenden Ansätze der technologischen Unterstützung fokussieren jedoch auf das technisch machbare. Offenbar scheitern aber gerade Versuche der Automatisierung auf der Grundlage formaler Repräsentationen (etwa abstrakter Instruktionsmodelle) an der Schwierigkeit, beliebige lebensweltliche Strukturen der Domänen und insbesondere mentale Strukturen, die der menschlichen Kommunikation zu Grund liegen, zu formalisieren. So verdeutlicht beispielsweise die Arbeit an Projekten wie Teachware-On-Demand [KW02], dass es in der Praxis kaum möglich ist, den Wissensstand eines Lernenden derart zu erfassen, dass man daraus automatisiert auf die Erfüllung der Voraussetzung für die Lektüre bestimmter Informationseinheiten schließen könnte.

Es erscheint eher aussichtsreich zu sein, mit der Entwicklungsumgebung und der -methodik, auf die Feinheiten der Begrifflichkeiten und der Tätigkeitsmuster innerhalb der jeweils adressierten Domäne individuell eingehen zu können. Insbesondere die Interaktivität des Systems, also die unterstützten Handlungsmuster, sowohl bei der Entwicklung als auch bei der Nutzung sollten auf die Bedürfnisse der Domäne eingehen können.

Die Spezifik der hier betrachteten Software-Klasse bezüglich des Vorgehens während der Umsetzung der Zielsysteme und damit die Abgrenzung zur Entwicklung von etwa betrieblichen Anwendungen, resultieren aus der Schwerpunktsetzung auf die Ausgestaltung der Aspekte der Nutzerinteraktion. Entsprechende Anforderungen sollte der Vorgang der Umsetzung erfüllen (in Anlehnung an [Pre99] Seiten 303 f.) und damit durch die Entwicklungsumgebung unterstützt werden.

- Es sollte ein schneller Wechsel zwischen den Aktivitäten des Entwurfs bzw. der Realisierung und denen der Simulation also der evaluierenden tatsächlichen Nutzung der Arbeitsergebnisse möglich sein. Über die von [Pre99] geforderte Möglichkeit eines schnelle Wechsels zwischen den entsprechenden Modi, also zwischen Entwurf und Ausführung der Zielanwendung hinaus wird mit dem in dieser Arbeit vorgestellten Ansatz untersucht, ob eine unmittelbare Verzahnung oder gar eine Verschmelzung der Modi der Entwicklung und der Anwendung effektiv sein kann.
- Das Dialogverhalten zwischen Entwurf und Ablauf des Zielsystems sollte übereinstimmen, damit Endanwender mit den gleichen Mitteln der Interaktion arbeiten können wie ursprünglich die Entwickler. Dies ist die Grundlage zur Ausnutzung der Kongruenz zwischen Anwendung und Entwicklung, um die traditionelle Kluft

zwischen Technikentwicklung und Fachkonzeption durch eine kollaborative transdisziplinäre Entwicklung zu überbrücken. So können Fachentwickler, deren fachliche Anforderungen durch das Zielsystem unterstützt werden sollen bereits an der Ausgestaltung mitwirken. Um dies gewährleisten zu können, bedarf es jedoch (so eine These dieser Arbeit) keiner generischen Interaktionsschemata für die Oberflächengestaltung sondern vielmehr individuell an die Fachlichkeit angepasste.

- Die direkt manipulative Erstellung statischer Aspekte der Anwendung (Texte, Grafiken, das Layout etc.) in einem Modus, der den Entwickler direkt, interaktiv das gestalten lässt, was er schließlich als Ergebnis erhält, ist bei den meisten der betrachten Ansätze Standard.
- Die Möglichkeiten der komfortablen Spezifikation dynamischer Aspekte beschränken sich dagegen in der Regel auf generische Metaphern (Filmrolle, Kartenstapel, etc.), deren Verständnis und Handhabung schließlich doch Spezialisten vorbehalten bleiben. Potential in einer tatsächlichen Vereinfachung ist dabei auch in der Individualisierung der Mechanismen der Spezifikation für spezifische Aufgaben, Fach- und Projektkontexte zu sehen.
- Das in [Pre99] geforderte einheitliche Entwurfsverfahren für alle Objekte der Nutzerschnittstelle und damit verbunden die Forderung nach einem durchgängigen Entwicklungswerkzeug, soll hier derart konkretisiert werden, dass zwar weitgehend einheitliche Interaktionsmuster für die Be- bzw. Verarbeitung aller Komponenten zum tragen kommen. Gefordert wird jedoch keine universelle Einheitlichkeit sondern lediglich die Durchgängigkeit und die Kohärenz der Interaktivität der Entwicklung bezogen auf die individuell zu unterstützende Entwicklungsaufgabe.
- Der Einsatz von Standardbausteinen soll es in Erweiterung des in [Pre99] entwickelten Verständnisses einerseits erlauben innerhalb eines entstehenden Systems konsistente, durchgängige Funktionalitäten der Nutzerschnittstelle zu bieten – sowohl an den individuellen Erfordernissen der Fachlichkeit orientiert als auch gängigen Paradigmen und Style Guides folgend. Andererseits sollte angestrebt werden, eine technologische Standardisierung der entsprechenden Software-Komponenten zu erreichen. Eine solche Vereinheitlichung und die transparente Gestaltung von Schnittstellen der internen Kommunikation bilden die Grundlage für die Wiederverwendung.
- Die technische Vereinheitlichung ist die Basis für nachhaltige Nutzung und eine vereinfachte Wartung.

Diese Forderungen werden in der Regel nicht oder nur unvollständig von den im Folgenden analysierten Produkten erfüllt.

2.2.2. Produkte

In den folgenden Abschnitten werden Entwicklungswerkzeuge beschrieben die den Markt dominieren und damit durch die von ihnen verwendete Technologie, die Gestaltung der

Nutzerschnittstellen und die Abstraktionsformen für die Erarbeitung der Aufbaustruktur und der Abläufe von Software das Vorgehen während der Entwicklung nachhaltig prägen. Die getroffene Auswahl der erläuterten Produkte erhebt keinen Anspruch auf Vollständigkeit, neben der Marktdurchdringung wird vielmehr Repräsentativität bez. der jeweils verwendeten zentralen Entwicklungs-Metapher angestrebt.

Ziel ist es also nicht, erschöpfende Produktbeschreibungen wiederzugeben, sondern vielmehr die Aspekte herauszuarbeiten, die eine integrative transdisziplinäre Arbeit an Software als Grundlage benötigt. Zu untersuchen sind also Einrichtungen mit deren Hilfe die Entwicklungsumgebungen die Erarbeitung, Zusammenstellung, Strukturierung und Aufbereitung der medialen Inhalte unterstützt. Es wird ferner betrachtet, welche Möglichkeiten die Systeme zur verteilten Arbeit in Teams bieten, inwieweit entstehende Artefakt interoperabel sind im Sinne der Übertragbarkeit in verschiedene Ablaufkontexte und damit wieder verwendbar sowie welche Deployment-Möglichkeiten (Distributionsformen) möglich sind.

Um diese Gesichtspunkte in den Fokus der Anforderungen transdisziplinärer Software-Entwicklung und der Einbeziehung von Domänenfachleuten in die technische Ausgestaltung von Software zu bringen, wird der Schwerpunkt der Betrachtungen querschnittlich auf der Usability liegen sowie auf dem zur Entwicklung nötigen technischen Sachverstand, der Adaptierbarkeit bzw. Erlernbarkeit, die durch die Hersteller avisierten Zielgruppen und die verwendeten Abstraktionen im Detail.

Authorware® (Macromedia, Inc.)

Die Authorware von der Firma Macromedia liegt zum Zeitpunkt dieser Niederschrift in der Version sieben vor und blickt damit zumindest nominell auf einen umfassenden Reifeprozess zurück. Sie gilt als letzte maßgebende (vgl. [HH02]) icon-orientierte Autorenumgebung (laut Macromedia-Terminologie: icon-based Authoring). Das bedeutet, dass die Elemente der Anwendung innerhalb eines Ablaufdiagramms organisiert und in entsprechender Reihenfolge zur Anzeige gebracht werden. Das Ablaufdiagramm folgt einer nativen Notation, in etwa vergleichbar mit Zustandsdiagrammen der UML, wobei ein Zustandssymbol der Wiedergabe bzw. Anzeige eines Elements entspricht; ein Zustandsübergang einer Änderung der wiedergegebenen Inhalte. Die Charakterisierung der Inhalte-Typen (Arten möglicher Zustände) wird durch jeweils spezifische Symbolik wiedergegeben. Auch die dynamischen Aspekte der Präsentation werden über Icons repräsentiert. Verzweigungen und Entscheidungen über Verzweigungen oder auch Pausen sind damit Modellelemente – ansatzweise vergleichbar mit Synchronisationspunkten in den Zustandsdiagrammen.

Die Abbildung 2.5 zeigt das Ablaufdiagramm einer einfachen Anwendung. Sie beginnt mit einem komplexen *Application Knowledge Object*, welches einen kompletten Eignungstest kapselt. Anschließend werden die Sachinhalte durch geschachtelte Mediengruppen (*Maps*), komplette Subanwendungen mit jeweils eigener Ablaufstruktur, präsentiert. Die Sachinhalte werden gefolgt von einem Kompetenztest und einem Verweis

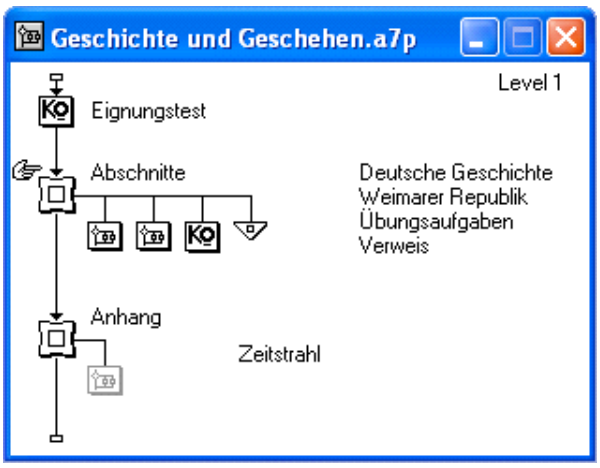


Abbildung 2.5.: Ein Ablaufdiagramm in Authorware

auf andere Inhalte. Der Rahmen, der diese vier Elemente gruppiert und durch das *Framework-Icon* symbolisiert ist, implementiert alle Funktionalitäten und das Design der globalen Navigation, wie eine Suchfunktion, eine Inhaltsübersicht sowie die „Vorwärts"- und „Rückwärts"-Bedienfelder zur Steuerung des Ablaufs entlang des vorgegeben Pfades. Der „Anhang" erhält im Beispiel einen eigenen Rahmen, mit einem eigenen „Look and Feel". Die Hand signalisiert dem Entwickler, an welcher Stelle des Ablaufs, sich die Anwendung während der Bearbeitung bzw. während der Testläufe befindet.

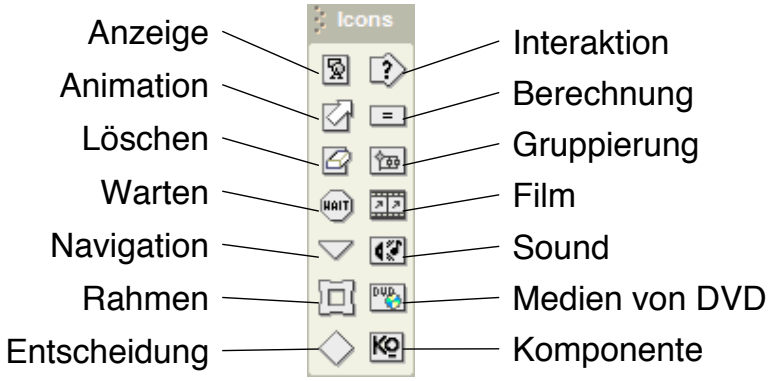


Abbildung 2.6.: *Icons* in Authorware

Die Elemente, mit denen eine Anwendung in Authorware definiert werden kann, sind im Schema 2.6 im Einzelnen aufgeführt und sollen nun kurz erläutert werden:

Anzeige Steht für ein Anzeigeelement, eine Bildschirmseite, auf der Grafiken und Texte arrangiert werden können.

Animation Realisiert die Bewegung von Objekten entlang von Führungslinien

Löschen Blendet unter Anwendung verschiedener Effekte nicht mehr benötigte Bildschirmobjekte aus.

Warten Fügt eine Pause in den Ablauf der Anwendung ein. Dabei sind die Länge der Pause und die Modalität der Nutzersteuerung für den Abbruch der Pause einstellbar.

Navigation Ermöglicht bedingt und unbedingte Sprünge zu anderen Elementen der Anwendung.

Rahmen Fasst alle Elemente der globalen Navigation innerhalb der Anwendung zusammen, etwa „Vorwärts“- und „Rückwärts“-Blättern im kanonischen Ablauf.

Entscheidung Schafft bedingte Verzeigungen im Ablauf der Anwendung und ermöglicht damit alternative Abläufe auch in Abhängigkeit von Nutzereingaben.

Interaktion Bietet die Möglichkeit der Nutzung gängiger Interaktionsformen mit dem Nutzer: Steuerelemente, wie Buttons und Menüs werden zur Anzeige gebracht und die Ergebnisse der Interaktion behandelt.

Berechnung Symbolisiert den Aufruf von Funktionen, die den Zustand der Anwendung verändern, also etwa Werte von Variablen manipulieren. Dahinter können unter anderem vollständige Skriptfragmente formuliert in JavaScript stehen.

Gruppierung Etabliert eine neue Gliederungsebene – einen „Unterablauf“.

Film Bindet einen Film ein und ermöglicht die Parametrisierung des Ablaufs durch das Festlegen von Start- und Endpunkt beispielsweise.

Sound Bietet analog zum Film-Element eine Einrichtung zum Wiedergeben von Tondokumenten.

Medien von DVD Kapselt den Zugriff auf Medienelemente (Bilder, Filme und Sound), die auf DVD vorliegen.

Komponente Fassen Funktionalitäten zu komplexeren Anwendungsteilen wie Wissenstests, aber auch standardisierten Dialogfunktionalitäten zusammen. Sie werden geführt durch so genannte *Wizzards* ausgeprägt.

Neben der Möglichkeit der Schachtelung der Flussdiagramme ergibt sich die Mächtigkeit der mit Hilfe dieser Notation formulierten Modelle und die so implizierte Flexibilität bzgl. der Erstellung von Anwendungen offensichtlich aus der detaillierten Konfiguration einzelner Elemente.

So hat jedes Element Eigenschaften, die über einen entsprechenden Eingabebereich manipuliert werden können. Die Abbildung 2.7 zeigt einen solchen vergleichsweise simplen Eingabebereich für ein einfaches Bildschirm-*Anzeige*-Element. Die Wertebereiche der einzelnen Eingabefelder — etwa die vielfältigen Möglichkeiten der Animation beim Einblenden des Anzeige-Elements (Auswahl-Feld `Transition`) — sowie die Semantik der

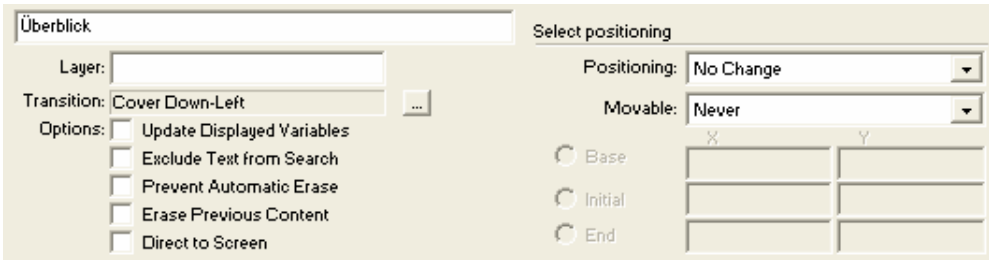


Abbildung 2.7.: Konfiguration von Bildschirmelementen in Authorware

jeweiligen Werte erschließt sich nicht intuitiv. Auch die Konsultation der Hilfe schafft keine praktikable Konkretisierung der sehr abstrakten technischen Begriffe.

Über die Konfiguration der Elemente hinaus, kann jedem Element ein Script-Fragment zugeordnet werden. Der wahlweise in der proprietären Sprache oder in Java-Script zu verfassende Code wird zur Ausführung gebracht, sobald das Element innerhalb des Ablaufs aktiv wird.

In der Beurteilung dieser Form der Abstraktion als Paradigma für eine vereinfachte intuitive Entwicklung interaktiver Systeme kommt man zu dem Schluss, dass die Vermischung von Datenstrukturierung und Ablauf- bzw. Verhaltenssteuerung zumindest bei größeren Projekten problematische Konsequenzen haben muss. Weder die ingenieurmäßige Software-Entwicklung findet saubere Anknüpfungspunkte für Ihre etablierten Modellbildungen, noch kann das System dem intuitiv arbeitenden, der Technik ferner stehenden Domänenexperten einen Zugang bieten. Auch visuell geprägten Screendesignern dürften die Spielräume bei der Gestaltung ansprechender Effekte zu eng sein.

Einrichtungen wie zahlreiche Wizzards, die den Nutzer etwa durch die Erstellung einfacher Übungen führen erlauben durchaus die schnelle Produktion kleinerer Lernanwendungen. Auch Adapter für den Import bereits vorhandener Präsentationsmaterialien prädestinieren die Entwicklungsumgebung für die kurzfristige Erzeugung einfacher E-Learning-Materialien mit einfachen Interaktionsmustern. Für die Arbeit an anspruchsvollen Multimedia-Produkten schränkt die enge Führung die Gestaltungsspielräume jedoch zu sehr ein.

Bezügliche der Distributionsform sind die auf der Basis von Authorware entwickelten Anwendungen sehr flexibel. Die ermöglicht eine Ablaufumgebung, die als Zusatzmodul (plugin) für alle gängigen Internet-Browser verfügbar ist. Ferner besteht für die Microsoft-Windows-Plattformen die Möglichkeit eine Ausführbare Datei zu erzeugen, die die Anwendung in die Ablaufumgebung eingebettet enthält. Die mit Authorware erstellten Anwendungen laufen also vollständig auf der Client-Seite ab.

Director® (Macromedia, Inc.)

Zum „Urgestein“ der Autoren-Werkzeuge gehört zweifelsohne der Director von Macromedia. Das Online-Magazin schreibt etwa zur Veröffentlichung der letzten Version: die langlebige Anwendung „stammt noch aus der Zeit, als Multimedia-Inhalte hauptsächlich auf CD-ROMs verbreitet wurden, und nicht über das Internet“ [Ern04]. Ein weiteres Indiz für die sehr weite Verbreitung der Entwicklungsumgebung ist die große Zahl an Bildungsangeboten rund um das Produkt. Studiengänge wie die „Medien-Informatik“ an der Technischen Fachhochschule in Berlin bieten Lehrveranstaltungen zu Autorensystemen an, in deren Zentrum die Arbeit mit Director steht.

Der letzte Aspekt deutet bereits an, wie sich die Zielgruppe der Anwendung zusammensetzt: es wird sich um Medienschaffende handeln mit vertieftem technischen Sachverstand, was man durch die Bemerkung: „Die wichtigsten Kunden des Director sind Software-Hersteller, die für ihre Kunden speziell zugeschnittene Weiterbildungs- und Trainings-Anwendungen erstellen.“ in [Ern04] (stellvertretend für die Fachpresse), bestätigt sehen kann.

Macromedia selbst bezeichnet das Produkt als „bewährtes Multimedia-Erstellungswerkzeug für Profis“, mit dem „sich ansprechende Inhalte entwickeln und überall bereitstellen“ [Mac04c] lassen. Die in der Hauptsache werblich anmutenden Passagen sind hier hauptsächlich wiedergegeben, um die Prämisse zu unterstreichen, das es sich bei Director a priori und intentional nicht um eine Software handelt, mit der technische Laien schnell und intuitiv Multimedia-Entwicklungen durchführen können bzw. die eine Integration unterschiedlicher Verständniswelten ausdrücklich unterstützt.

Die folgenden Betrachtungen des Aufbaus und der Funktionsweise des Systems werden diese These weiter konkretisieren. Einen Überblick über den Aufbau der Anwendersicht auf die Entwicklungsumgebung bietet das Bild 2.8. Der Filmmetapher folgend, erstellt der Autor Filme, die sich aus Einzelbildern zusammensetzen. Die in der so genannten *Besetzung* [Mac04b] organisierten *Darsteller* können in verschieden Ausprägungen als *Sprites* mehrer Rollen im Film übernehmen. Der untere Teil des Fensters in Abbildung 2.9 zeigt eine einfache Besetzung. Die *Darsteller* sind dabei Medienobjekte, wie Bilder, Sounds und Textbausteine. Ebenfalls als *Darsteller* bezeichnet Director Scriptfragmente, die etwa das Verhalten von *Sprites* bestimmen. Dementsprechend werden sie in der Bibliothek der *Besetzung* verwaltet. Das ist im Beispiel (Abbildung 2.9) durch die Methode `onClick` illustriert.

Darsteller übernehmen eine Rolle — werden ausgeprägt zu *Sprites* des Films, indem sie auf die so genannte *Bühne* gebracht werden (linkes, oberes Fenster in Grafik 2.8). Das *Drehbuch* legt den zeitlichen Ablauf des Films fest, indem es eine Folge von einzelnen Bühnenbildern und damit die Eigenschaften (z.B. Position, etc.) und das Verhalten der einzelnen Sprites über der Zeit definiert. So werden beispielsweise die Position oder auch die Reaktivität im Bezug auf Nutzerinteraktionen an Übergängen zwischen den einzelnen Bildern verändert. Ein flüssiger Filmablauf, etwa beim Einsatz von Animationen,

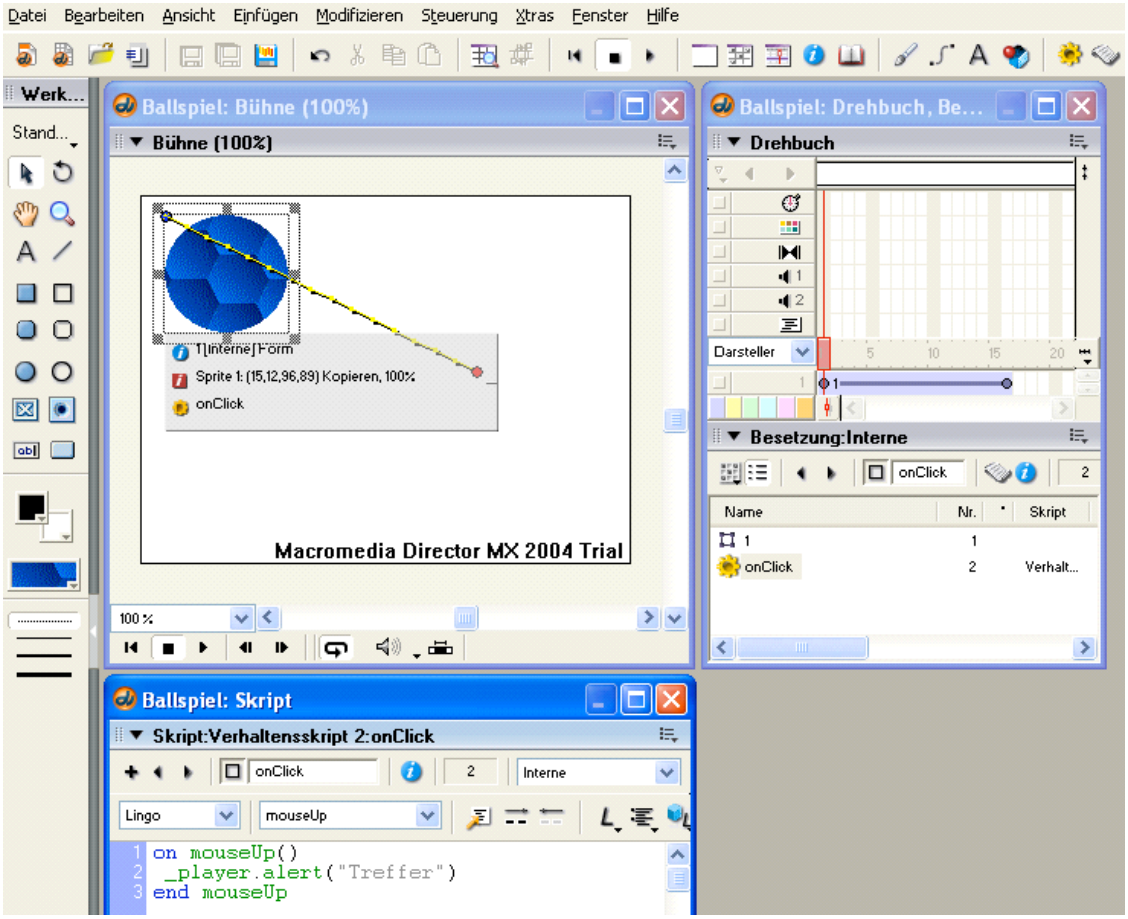


Abbildung 2.8.: Macromedia Director MX 2004 Überblick

ergibt sich nach Angaben in [Mac04b], wenn eine Bildrate von 30 Bildern pro Sekunde eingestellt wird.

Der obere Teil der Abbildung 2.9 zeigt wie die Spuren oder Kanäle [Wel96] des Films durch in Zellen unterteilte Zeilen repräsentiert sind. Jede Zelle entspricht dabei einem Bild des Films. Die Inhalte der Spalten bilden verschiedene Ebenen eines Bildes. Es gibt spezielle Spuren für das steuern zeitlicher Abläufe. Hier können Bilder etwa mit Pausen versehen werden. Die Soundkanäle ordnen den Bildern die Audioinformationen zu.

Die Verwaltung der eigentlichen Inhalte erfolgt wie erwähnt in der *Besetzung*. Medienelemente, wie Bilder, Filmsequenzen, Audiodaten und Texte können mit Hilfe eines einfachen Gruppierungsmechanismusmechanismus entlang verschiedener Besetzungen abgelegt werden. Eine Schachtelung von Besetzungen und ein Herstellen von qualifizierten Beziehungen zwischen Medienelementen auf der Typebene (also in der Besetzungsbibliothek) sind nicht möglich. Das bedeutet, dass sich eine systematische Organisation der Inhalte großer Projekte sehr unübersichtlich gestalten kann.

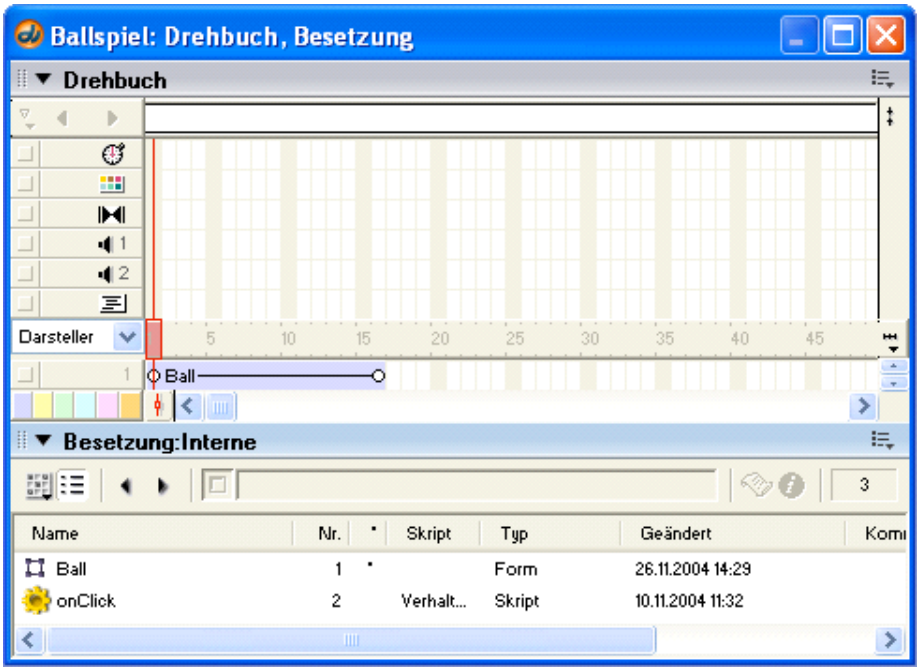


Abbildung 2.9.: „Drehbuch“ und „Besetzung“ in Macromedia Director MX 2004

Um über die reine Präsentation von Informationen hinaus, hoch interaktive Systeme erstellen zu können, wie sie etwa für die Unterstützung der Lehre nötig sind, ist es unabdingbar, Gebrauch von den Scripting-Fähigkeiten von Director zu machen. Nur so ist es möglich die zahlreich verfügbaren Funktionsbibliotheken zu nutzen und so den individuellen Bedarfen der Repräsentation von Domänensachverhalten gerecht zu werden. Director bietet die Möglichkeit, der Implementierung entsprechender Funktionalitäten einerseits mit der nativen Scriptsprache *Lingo* bzw. andererseits nunmehr auch mit JavaScript. Die Bezeichnung „Scriptsprache“ deutet bereits darauf hin, dass eine funktions- bzw. prozedurorientierte Programmierung ohne strenge Typisierung möglich ist.

Eine eindeutige Zuordnung von Funktionalitäten zu ihren Trägern wird in der Regel erst bei der Ausprägung konkreter Sprites vorgenommen. Eine bezogen auf die Datenstrukturen konsistente Kapselung von Funktionen, etwa im Sinne der Objektorientierung, ist damit nicht a priori sicherzustellen. Darstellerskripte, bzw. die Möglichkeit der Ableitung so genannter *Child-Objekte* von *Parent-Scripten* realisieren in Ansätzen Konzepte, die ein objektorientiertes Design umsetzen könnten (vgl. [Mac04a], Seite 62 ff.). Die zur Verfügung stehenden Konstrukte zur logischen Gliederung und zur Modularisierung von Daten und Funktionalität sind zusammengefasst:

- *Besetzungen*, von denen beliebig viele in ein Filmprojekt eingebunden werden können. Eine Strukturierung über den *Besetzungen* ist nicht möglich — neben der physischen Differenzierung in externe verlinkte und interne sind die einzelnen

Pakete lediglich durch Bezeichner zu unterscheiden.

- Mit dem Mechanismus der *eingebetteten Filme* lässt sich Modularisierung praktisch beliebiger Granularität realisieren. Problematisch gestaltet sich tendenziell das Design von Schnittstellen. Durch einen rudimentären Referenzierungsmechanismus auf der Basis der Nummern der *Kanäle* entsteht Datenhabhängigkeit. Nur klare Absprachen über die Nutzung bzw. Belegung einzelner *Kanäle* ermöglicht die Nutzung eingebetteter Filme im Sinne von Modulen oder Komponenten, wie sie in der Softwaretechnologie verstanden werden.
- Inhalte können als externe Ressourcen eingebunden werden. So können Text-*Darsteller* mit einer externen Textdatei korrespondieren, so dass, will man den Text ändern, lediglich die Textdatei bearbeitet werden muss, der Film an sich unberührt bleiben kann. Dies setzt allerdings eine disziplinierte Pflege der Datenbestände voraus, um die Entstehung von Inkonsistenzen bei den Verknüpfungen (tote Links) etwa durch die Veröffentlichung und die Distribution zu vermeiden. Dieser Ansatz entfällt praktisch als Option der Redigierung von Texten, wenn das von [Wel96] vorgeschlagene, offenbar gängige Vorgehen gewählt wird, bei dem Texte mit Hilfe eines explizit für den Schriftsatz vorgesehenen Systems erstellt und als Bilder eingebunden werden.
- Konstrukte der Scriptsprache *Lingo* bzw. von JavaScript ermöglichen einfache Type-Instanzbeziehungen und entsprechende Kapselungsstrategien.

Eine physikalische Dekomposition der Entwicklungsquellen, etwa um in Teams aus mehreren Personen zeitlich parallel daran arbeiten zu können, wird im Wesentlichen über die Verwendung externer *Besetzungen* und die geschachtelte Einbettung von separaten Director-Filmen realisiert. Das Auslagern von rein Quelltext-Elementen zur dynamischen Einbeziehung zum Zeitpunkt der Übersetzung der Quelltexte bzw. des Linkens ist nicht ohne weiteres Möglich. Das erschwert eine Versionsverwaltung von Quellcode auf der ebene von Textdateien, die etwa ein Mischen (Merge) der Ergebnisse paralleler Aktivitäten erlauben würde.

Zusammenfassend lässt sich feststellen, dass der Einsatz von Director sowohl grafisch, gestalterischen Sachverstand als auch vertiefte Kenntnisse im Umgang mit Scriptsprachen bzw. grundsätzlich der Programmierung bedarf. Wobei die entlang der Filmmetapher getaktete Ablaufsteuerung und das damit verbundene Nachrichtenmodell selbst von versierten Programmierern gängiger Sprachen der 3. Generation (3-GL-Sprachen) eine deutliche Spezialisierung erfordert. Die in der Sekundärliteratur (z.B. in [Wel96]) propagierten Vorgehensweisen, etwa für den präzisen Satz von Texten über spezialisierte externe Software lassen ferner erkennen, dass Director ein Werkzeug ist, um grafisch aufwendige repräsentative Multimedia-Anwendungen zu erstellen, die immer voraussetzen, dass der Autor den Umgang mit dem gesamten Repertoire an Software zur Aufbereitung der Einbezogenen Medien beherrscht.

Im Ergebnis ist es möglich umfangreiche Multimediasysteme beliebiger Distributionsformen zu erstellen. Der Direktorfilm wird dabei immer auf der Seite des Client im

Rahmen eines *Projektors* zum Ablauf gebracht. Der *Projektor*, der für alle gängigen Betriebssystemplattfomen verfügbar ist, kann als ausführbare Datei mit dem Film verbunden, beispielsweise als installationsfähiges Paket verteilt werden. Es gibt aber ferner auch die Möglichkeit, den Director-Film als Web-Anwendung auf der Basis eines Plug-In des Webbrowsers zu nutzen. Dabei können Teile des Films dynamisch über die Web-Verbindung nachgeladen werden.

HyperCard 2.4 (Apple Computer)

Das HyperCard-System ist zunächst als „eine Art Datenverwaltungsprogramm“ ([Dah95] Seite 43) charakterisiert, mit dem es möglich ist, Text, Grafiken, Audio- sowie Videodaten im Sinne von Hypermedia zu strukturieren und zu präsentieren, indem beliebige Verbindungen zwischen den Medien hergestellt werden können und damit die Navigationsstruktur festgelegt wird. Die Möglichkeit, in der objektorientierten Programmiersprache *HyperTalk* verfasste Funktionalitäten zur Steuerung der Abläufe zu hinterlegen, machen die Software schließlich zu einer „eleganten Programmierumgebung“ [Dah95] Sie ist gleichzeitig Ablaufumgebung für die entstehenden Programme. Das System soll es durch die Verwendung einer einfachen Metaphorik der graphischen Nutzerschnittstelle erleichtern, interaktive Software zu entwickeln. Die Entwicklungsumgebung soll in gleicher Weise intuitiv bedienbar und nutzerfreundlich sein, wie die entstehenden Anwendungen.

Die zentralen Elemente der Metapher sind [Dah95]:

- **Stapel**
- von **Karten**
- Für Karten können jeweils **Hintergründe** definiert werden.
- Die wichtigsten Steuer- bzw. Gestaltungselemente sind:
  - **Textfelder**
  - **Tasten** und
  - Medienelemente wie Bilder etc.

Entsprechend dem Namen *HyperCard* ist die Karte zentrales Strukturelement der Metapher. Sie fast Inhalte und Steuerelemente in der Art eines Bildschirms (bzw. Fensterinhalts) oder auch einer Website zusammen. Die inhaltliche Gruppierung entsprechend thematischer Schwerpunktsetzungen erfolgt über Stapel von Karten. Hintergründe fassen dabei im Sinne von Klassen diejenigen Funktionalitäten und deren Präsentation durch Gestaltungselemente zusammen, die in gleicher Weise für mehrere Karten eines Stapels benötigt, also wieder verwendet werden.

Damit entspricht die grundlegende Metapher am ehestem dem, was man sich intuitiv unter einem Dialogsystem oder aber einer Sammlung digitaler Medien vorstellt: eine

feste Abfolge von Bildschirminhalten, die jedoch abhängig von Bedingungen (Systemzustände, Nutzereingaben, etc.) nicht streng linear ablaufen muss, sondern Verzweigungen und damit auch Zyklen aufweisen kann. Entscheidend ist dabei, dass der Anwender sich mit solchen technischen Abstraktionen nicht befassen muss, sondern mit ihm vertrauten Bildern arbeiten kann.

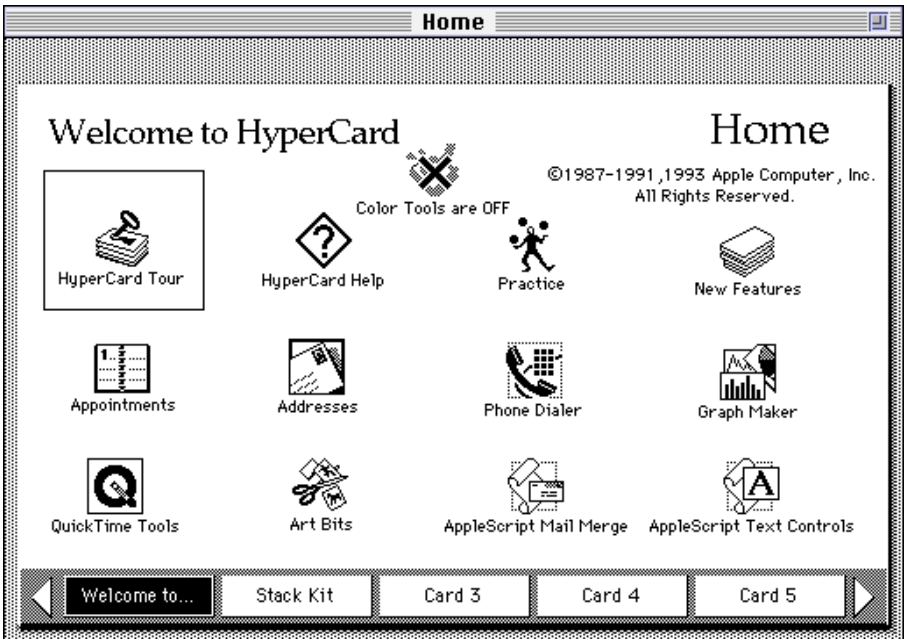


Abbildung 2.10.: Der *Homestack* von HyperCard

Abbildung 2.10 zeigt den so genannten *Start- oder Homestack* von Hypercard. Interessant ist dabei, dass auch die Anwendung selbst in einem Stapel von Karten organisiert ist. Die Symbole der abgebildeten ersten Seite dieses Stapels sind jeweils Knöpfe mit Verweisen zu wiederum Karten im selben oder einem anderen Stapel bzw. zu Script-Funktionalitäten.

Um tatsächlich anspruchsvolle Anwendungen für beliebige Aufgabenkontexte individuell und flexibel gestalten zu können, reichen jedoch einfache visuelle Mittel der Entwicklung nicht aus – etwa das Platzieren von Steuerelementen per Maus. In der so genannten *Umgebung* zu jedem Element können vielfältige Eigenschaften die Darstellung und das Verhalten betreffend individuell konfiguriert werden. So kann etwa ein Textfeld für Nutzereingaben geöffnet werden oder fixiert. Für einen Knopf lässt sich festlegen, welchem der gängigen Interaktionsmuster er folgen und damit in welchem Stil er dargestellt werden soll (beispielsweise als einfache Schaltfläche, Radiobutton für eine exklusive Auswahl oder aber als Checkbox). Schon an den einfachen Beispielen wird deutlich, dass die Terminologie „unter der Oberfläche“ an den gängigen eher technischen Verallgemeinerungen orientiert ist.

Die volle Mächtigkeit der Entwicklungsumgebung entfaltet sich jedoch erst mit dem Einsatz der Scriptsprache. Wie in objektorientierten Sprachen üblich und gemäß dem Model-View-Control-Paradigma können jedem Interaktionselement, welches in der Lage ist Ereignisse auszulösen entsprechende *Handler* zugewiesen werden. Diese implementieren die Funktionalität die als Reaktion auf das Ereignis ablaufen soll – sie verarbeiten die entsprechende *Botschaft*. Ein typisches Beispiel ist die Verarbeitung des Maus-Klicks auf einen Knopf. Dabei wird üblicherweise die Botschaft ausgewertet, die beim Loslassen der Maustaste über dem Objekt ausgelöst wird. Ein einfaches Beispiel für den Quelltext der einen solchen Handler umsetzt sieht folgendermaßen aus.

```
on mouseUp
    put "Hallo Welt!" into msg
end mouseUp
```

Nachdem der Knopf, mit dem dieser Handler assoziiert ist, geklickt (Maustaste gedrückt und wieder losgelassen) wurde erscheint ein Nachrichtenfenster mit dem Text „Hallo Welt!“. Was hier im Beispiel trivial erscheint, soll lediglich illustrieren, dass Kenntnisse in der Programmierung einer objektorientierten Sprache nötig sind, um das volle Potential des Entwicklungssystems auszuschöpfen. Die Sprache ist zwar an Smalltalk angelehnt, besitzt jedoch eine eigene Syntax und umfasst alle gängigen Konstrukte zur Steuerung des Ablaufs (Bedingungen, Schleifen, etc.) sowie zur Strukturierung (Methoden und Funktionen, etc.). Unabdingbar für einen Nutzer des Systems bzw. einen Entwickler ist ferner die Kenntnis der proprietären Architektur des Gesamtsystems einschließlich der Betriebssystemumgebung und der dadurch implizierten Hierarchie der Ereignisverarbeitung.

Die entstehenden Anwendungen sind an die Betriebssystemplattform gebundene lokale und allein stehende Anwendungen. Lediglich mit zusätzlichen Hilfsmitteln lassen sich web-fähige Präsentationen der Stapel generieren.

HyperCard findet hier einerseits Erwähnung, weil die verwendete Metapher viel versprechend war in der Hinsicht, dass sie es mit ihrer intuitiven Verständlichkeit einem breiten Kreis von Nutzern über den exklusiven Kreis professioneller Programmierer hinaus ermöglichen sollte, Anwendungen zusammenzustellen. Auf Grund der Tatsache, dass das Produkt als eines der ersten dieser Art gilt, welches zudem durch die enge Kopplung an die Produkte der Firma Apple massenhaft Verbreitung fand, sollte dies zumindest für ambitionierte Anwender tatsächlich zutreffen. Andererseits macht das Beispiel auch deutlich, dass viele der Bemühungen die Entwicklung von Software ergonomisch und „massentauglich“ zu gestalten sich erstens meist auf Versuche beschränkt die Implementierung direkt zu vereinfachen. Zweitens scheitern diese Versuche insofern, dass sie schließlich doch größtmögliche Allgemeingültigkeit anstreben und dem nicht technikaffinen Entwickler entsprechend technisch orientierte Abstraktionen zumuten müssen.

Produkte die sich der gleichen Metapher bedienen und mehr oder minder ähnliche Leistungsmerkmale aufweisen sind etwa SuperCard (IncWell Digital Media Group Ltd.) oder auch MetaCard 2.3 (MetaCard Corporation).

2.2.3. Verständlichkeit durch Metaphern

Um die Praktikabilität der vorstellten Systeme innerhalb des diskursiven Prozesses der Entwicklung beurteilen zu können, müssen die verwendeten Metaphern ins Verhältnis gesetzt werden zu den mentalen Modellen der Fachexperten bzw. Autoren. Dabei soll unter dem Begriff mentales Modell nach der Definition von [Mar98] die adäquate Organisation und Repräsentation von Daten, Funktionen, Aktivitäten und Rollen von Akteuren innerhalb sozialer Organisationen verstanden werden. So Umfasst etwa das mentale Modell von Historikern (z.B. innerhalb der sozialen Organisation eines Autorenteams):

- Historische Fakten in Form von (ggf. aufbereitetem, kommentiertem) Quellenmaterial (Daten),
- Einrichtungen zu dessen Betrachtung und Analyse (Funktionen von Filmprojektoren, aber auch das Blättern in Büchern etc.),
- Die Tätigkeiten des Sammelns, Strukturierens, Annotierens, Interpretierens historischer Quellen in historischen Zusammenhängen, Repetieren, Nachschlagen, Präsentieren (Aktivitäten)
- Mögliche Rollen sind Schüler, Studenten, Forscher, interessierte Laien.

Die Metapher der Oberfläche von Entwicklungswerkzeuge sollte laut [Mar98] in ihrer Bildsprache so gestaltet sein, dass Assoziationen mit bekannten Mustern leicht möglich sind. Die Möglichkeiten der Interaktion und der Inhalt der Präsentation müssen leicht erfasst, verinnerlicht und behalten werden können.

Die Beobachtung in der Praxis zeigt jedoch, dass selbst scheinbar intuitiv erfassbare Metaphern wie die Flussdiagramm-Metapher, oder die Karten-Stapel-Metapher etc. in den konkreten Ausprägungen der verfügbaren Entwicklungswerkzeuge und den damit verbundenen – der technischen Flexibilität geschuldeten – vielfältigen Konfigurationsmöglichkeiten zu weit von den mentalen Modellen vieler Disziplinen entfernt sind.

Problematisch dabei ist auch, dass viele der Produkte versuchen, alle Aspekte der Produkterstellung in die Metaphorik einzubeziehen. So sind Nutzer beispielsweise unter dem Bild des „Veröffentlichens eines Films“ mit den Aspekten der Erstellung des Distributionsmediums bzw. der Erstellung eines Releases konfrontiert. Spätestens mit der zu treffenden Entscheidung für eine abgeschlossene ausführbare Datei oder die Generierung von HTML-Rahmen und den dazugehörigen vielfältigen Wahlmöglichkeiten für Varianten ist die Vorstellungskraft der meisten Nichttechniker überstrapaziert.

Ein individueller Zuschnitt der Entwicklungsmetapher auf die mentalen Modelle der Zieldomäne müsste also von solchen Aspekten komplett abstrahieren können. Eine Spezialisierung der Metaphorik auf der Ebene der Tätigkeit und der Begriffsbildung der Anwendervertreter, also ein projektabhängiger Zuschnitt auf die von ihnen tatsächlich zu bewältigenden Aufgaben erscheint vor dem Hintergrund im allgemeinen wünschenswert. Die Lösung von der Universalität fester Metaphern zugunsten praxisgerechter

Arbeitsmittel wird allerdings für jede nötige Individualisierung einen zusätzlichen Kostenfaktor verursachen, der lediglich über die Wiederverwendung innerhalb der Domäne zu rechtfertigen sein wird.

2.2.4. Implizierte Vorgehensweisen

Die Vorgehensweisen bei der Erstellung direkt reaktiver, interaktiver Software sind in der Regel geprägt vom Projektverständnis der beauftragenden Fachdomäne, vom Pragmatismus der umsetzenden Agenturen bzw. letztlich auch von der Charakteristik der Werkzeuge und von der individuellen Arbeitsweise der Fachleute. In einer typischen Konstellation von Rollen und deren Interaktionen (wie sie in Abschnitt 2.1.3 beschrieben sind) muss davon ausgegangen werden, dass ein ingenieurmäßiges Vorgehen, das die für alle Beteiligten nachvollziehbare Entwicklung des Software-Artefakts in den Mittelpunkt stellt nur schwer etabliert werden kann.

Standard: Ad-hoc-Entwicklung

Den Einfluss den der De-facto-Standard der Werkzeuglandschaften auf die Arbeitsweise in Entwicklungsprojekten hat wird im Folgenden erörtert. Die Grundlage dafür bilden neben den in der Analyse der Werkzeuge gewonnenen Erkenntnissen die Beobachtungen die während der ethnographischen Untersuchung einer Reihe typischer Projekte[3] gemacht wurden.

Die Gestaltung der Entwicklungsumgebungen selbst geht offenbar davon aus, dass Produktionsprozesse deterministisch, linear und unter sauberer Trennung von Aufgabenbereichen ablaufen. Die ausschließlich auf die technisch-mediale Umsetzung spezialisierten Werkzeuge – die von ihnen verwendeten Metaphern und die damit unterstützten mentalen Modelle – manifestieren die Spezialisierung der Rolle ihrer Anwender. Lediglich hoch spezialisierte Entwickler mit dezidiert technischem Sachverstand sind in der Lage Anwendungen zu implementieren. Die Praxis zeigt dabei, dass begründet durch die Diversität der verschiedenen Paradigmen sogar eine Spezialisierung auf einzelne Produkte erforderlich wird, um tatsächlich größtmögliche Gestaltungsspielräume zur Verfügung zu haben.

Eine Besonderheit stellt darüber hinaus die Vermischung technisch konzeptioneller Aufgaben mit denen der Realisierung dar. Das bedeutete etwa für alle der beobachteten Projekte, dass die technische Entwicklung direkt mit der Umsetzung erster inhaltlicher Konzepte und Oberflächenideen (nicht von Entwürfen) in lauffähige Software auf der Grundlage eines der oben beschriebenen Autorensysteme begann. Technische Konzepte materialisiert in Skizzen, Modellen und Dokumente wurden nicht erstellt, weder als Diskussionsgrundlage noch als belastbare Spezifikation von Anforderungen, und

[3] LeMOLernen, Entwicklung von E-Learning-Formaten für die Bundeszentrale für politische Bildung, Entwicklung digitaler Karten – Software-Entwicklungsprojekte am Fraunhofer ISST

Lösungsansätzen. Ein Verständnis von Entwurf und Architektur war in der Regel lediglich implizit vorhanden bzw. durch die Vorgaben der Entwicklungswerkzeuge bez. der Distributionsformen weitgehend vorab determiniert. Viele der Werkzeuge zielen darauf ab, ein monolithisches Artefakt, bestehend aus einer Datei zu erzeugen, ohne die Möglichkeit zu bieten eine asynchrone Kommunikation mit externen oder gar verteilten Komponenten zu etablieren.

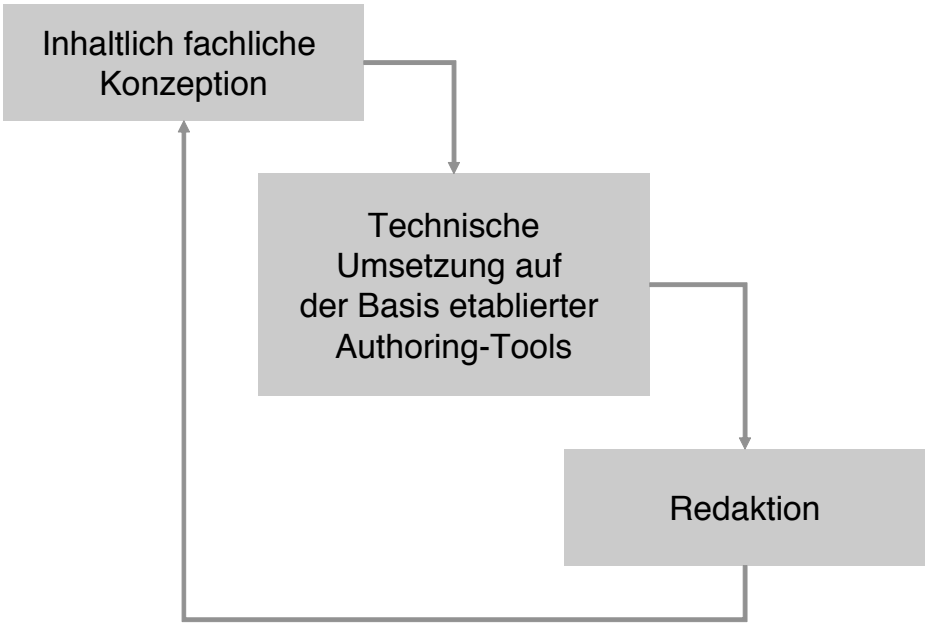


Abbildung 2.11.: Isolation von Phasen im üblichen Projektverlauf

Abbildung 2.11 zieht klar die Parallele zwischen dem Wasserfall-Lebenszyklus und den in der Medienbranche häufig „gelebten“ Prozessen. Der entscheidende Unterschied zwischen klassischem Wasserfallmodell und der beobachteten Projektrealität besteht jedoch im Fehlen konzeptioneller Phasen. Dieses offenbare Defizit ergibt sich unter anderem aus der Charakteristik der Autorenwerkzeuge. Sie vermischen konzeptionelle Aspekte der technischen Entwicklung, den Entwurf des Aufbaus bzw. der Struktur auf der Basis technische medialer Termini und Metaphern (etwa dem Auflaufdiagramm) mit den Aufgaben der eigentlichen Kodierung von Inhalten, Funktionalitäten sowie Screen-Designs. Die entsprechenden Aktivitäten lassen sich während der Arbeit mit den Entwicklungsumgebungen kaum trennen. Auf diese Weise wird die Entstehung einer sehr komplexen und isolierten Phase der technischen Umsetzung unterstützt.

Durch die angesprochene Vermischung von konzeptionellen mit technischen Aspekten, also der direkten Verquickung der Beschreibung von Aufbau- und Ablaufstruktur mit Hilfe technischer bzw. medialer Termini und Metaphern auf der einen Seite mit denen der Implementierung (Code) auf der anderen ist demnach auch eine Abbildung konzeptioneller Zwischenschritte in gängige Modellsprachen der Software-Entwicklung aus

systematischen Gründen kaum möglich. Die Dekomposition und damit die Gliederung auf der Ebene des Codes selbst sind nur unter Verwendung von Hilfskonstrukten dafür geeignet, zunächst in Phasen der Modellierung konzeptionell diskutiert zu werden.

Es ergeben sich monolithische Aufgabenblöcke, deren Zielartefakte sich kaum in Inkremente unterteilen lassen. Die Phasen der Entwicklung (unter Umständen auch in mehreren Iterationen), die sich diesen Aufgabenblöcken widmen sind bez. ihrer Dauer entsprechend Umfangreich, da Parallelisierung zumindest erschwert wird.

Eine weitere Konsequenz ist, dass Entwicklungsabläufe entstehen, die eine komplexe Schnittstelle zwischen einer Phase der technischen Realisierung und einer der fachlichen Konzeption aufweisen. In der Regel gibt es also einen zentralen Bruch zwischen der Antizipation der fachlichen (Nutzer-)Anforderungen und deren Transformation in technische Konzepte. Diese Prozessschnittstelle trennt mit den Aufgabenbereichen primär auch Kompetenzbereiche und damit Disziplinen. Dies ist als ebenso nahe liegend, wie schließlich in vielen Fällen auch als verhängnisvoll zu bewerten. Die Kluft die zwischen den Disziplinen besteht, wird also im Rahmen gängiger Produktionsprozesse auf der Basis etablierter Werkzeugumgebungen vertieft.

Die Durchgängigkeit der Entwicklung und damit ein über die gesamte Projektlaufzeit etabliertes gemeinsames konsistentes Verständnis für das Zielsystem kann kaum entwickelt werden. Die Arbeitsweisen innerhalb der beiden Phasen lassen sich insbesondere kaum im Sinne einer ingenieurmäßigen Vorgehensweise priorisiert verzahnen. So wie es erschwert ist, auf der Grundlage der bestehenden Werkzeuge und der dazugehörigen Entwicklungskultur systematisch ggf. modellgetrieben die eigentlichen Software-Artefakte zu produzieren, so wenig ist es denkbar, dass tatsächlich standardisierte Vorgehensweisen zu normgerechten Spezifikationen der fachlichen Anforderungen führen. Die Arbeitsergebnisse nichttechnischer Autoren und Redakteure, die die entsprechende Phase prägen, können den Ansprüchen einer Anforderungsspezifikation im klassischen Sinne als Ausgangspunkt für die kanonische Ableitung von Systemdesignentscheidungen nicht genügen. Die beim Wasserfallmodell umfassend kritisierte Voraussetzung der Abschließbarkeit von Entwicklungsphasen als sauberer Aufsetzpunkt für die Folgephase [BvK02] durch die vollständige Fertigstellung des jeweiligen Ergebnisses wird so in vielen realen Projekten auf der Grundlage der marktüblichen Autorenwerkzeuge in besonderem Maße zur Quelle von Missverständnissen und damit Projektrisiken.

Verschärfend wirkte bei den untersuchten Projekten die Dauer der Phasen. Die in jeweils sehr langen Phasen der inhaltlichen Konzeption erarbeiteten Drehbücher und so genannten Werkbeschreibungen, die zum Teil sehr detaillierte Festlegungen zu verwendeten Medien und deren Inszenierung machten, stellten in der Form ihrer Ausführung und auch durch ihre Loslösung von jeglicher Erwägung der technischen Machbarkeit und medialen Sinnhaftigkeit große Probleme für die Umsetzung dar. Waren die Unklarheiten mühevoll über die komplexe transdisziplinär geprägte Prozessschnittstelle (siehe oben) scheinbar ausgeräumt, zogen sich die technischen Entwickler ihrerseits in weitgehende Isolation (gleichsam hinter die Brandmauer der Informatik) zurück. Die dabei jeweils getätigten Investitionen waren entsprechend groß und proportional dazu das Verlustpotential im Falle des Scheiterns hoch.

Eine besondere Rolle im Komplex an Risiken, die sich aus langen isolierten Entwicklungsabschnitten ergeben, kommt der Einflussmöglichkeit der Kunden zu. Die häufig die frühere inhaltliche Phase in der Rolle von Redakteuren und Autoren bestimmenden Auftraggeber verlieren über weite Strecken des Projekts ihren Einfluss auf das entstehende System. Im Gegenzug fehlen den Auftragnehmern die Möglichkeit der Rückversicherung und damit die Kontrolle über die potentielle Kundenzufriedenheit. Entsprechende Beobachtungen lassen sich in fast allen der analysierten Projekte machen. Darüber hinaus befragte Protagonisten vergleichbarer Projektkontexte[4], sowohl von der fachlichen als auch der technisch orientierten Seite der Entwicklung sehen diese Verwerfung in der Kommunikation zwischen den Parteien als hauptsächlichen Anlass für die Unkalkulierbarkeit von Entwicklungskosten und Frustration der Projektbeteiligten.

Es ist allerdings kaum möglich festzustellen, was letztendlich ursächlich für diese Defacto-Entwicklung in der Produktion interaktiver präsentationsorientierter Software ist: das Wesen der Werkzeuge und die daraus resultierenden Rollenverteilungen und Spezialisierungen oder entsprechende historische Entwicklungen, die dazu führen, dass etwa klassische Verantwortungsverteilungen aus dem Printmedienbereich übernommen werden.

Evidenz erhalten diese Beobachten nicht zu letzt durch die Analyse klassischer phasenorientierter Vorgehensmodelle für die Entwicklung von Software im Allgemeinen. Diese manifestieren die Isolation der Nutzer vom Entwicklungsprozess gezielt. Die in der Abbildung 2.11 zusammengefassten Analyseergebnisse stellen eine Projektion der von [FK84] für die Software-Entwicklung im Allgemeinen erfasste Situation auf den Kontext der Entwicklung interaktiver, direkt reaktiver Systeme dar. Demnach findet eine Kommunikation mit dem Nutzer lediglich zu Beginn bzw. am Ende eines Projekts statt. Selbst iterative Phasenmodelle sehen nach dem von [FK84] auf Seite 37 entwickelten Schema vor, dass die vom Auftraggeber vorgelegten Nutzerforderungen (etwa in Form eines Lastenhefts) durch den Auftragnehmer mit einem Vorschlag zum technisch Machbaren und Nötigen beantwortet wird. Alle anschließend im Projekt erarbeiteten Dokumente – Modelle, Spezifikationen und Software-Artefakte – sind per se nicht für die Kommunikation mit dem Nutzer geeignet und gedacht. Erst die Präsentation fertig entwickelter Software ermöglicht wieder eine Rückkopplung zum Endnutzer. Dessen nicht seltene Reaktion [FK84] mit „Das habe ich nicht gewollt“ überspitzt verallgemeinert.

Die Transferleistung der Übertragung der Konzepte der Metaphern in die durch das Zielsystem wiederzugebenden Abläufe und Aufbaustrukturen der Domäne und damit

[4] Entsprechende Gespräche wurden u.a. geführt mit den folgenden Firmen bzw. Institutionen:
- bottled fish GbR Berlin (2003),
- bvm Gesellschaft für konzeption und gestaltung digitaler medien mbH Berlin (2003),
- KREAKTOR GmbH Hannover (2005),
- vr-fabrik virtual reality und multimedia GmbH Halle an der Saale (2005),
- Arbeitsgruppe zur Medienunterstützung bei der Vermittlung qualitativer Methoden und Triangulation in der Schul- und Bildungsforschung um Frau Dr. Jeanette Böhme in Berlin (2003) sowie
- Verschiednen Fachredaktionen des Ernst Klett Schulbuchverlag GmbH Leipzig (2002-2005)

letztlich der technische Ansatz der Lösung und deren Überführung in die Anwendung stellen in transdisziplinären Projektkontexten eine besondere Herausforderung dar. Umso problematischer wirkt sich der systemimmanente Bruch zwischen Anwendung und Technologieentwicklung aus. Eine zielgerichtete und über die gesamte Lebenszeit des Systems kontinuierlich forcierte Verzahnung der Beiträge aller beteiligten Disziplinen erscheint zwingend notwendig. Erst im Spannungsfeld der direkten (täglichen) Zusammenarbeit zwischen den Disziplinen ergeben sich die benötigten diskursiven Prozesse, die Lösungen emergent entstehen lassen. Enge Iterationen und damit die ständige Rückkopplung zwischen dem fachlich Wünschenswerten und dem technisch Machbaren, sowie die so sichergestellte Synchronisation der Aktivitäten unterstützt durch die Orientierung am gemeinsamen Artefakt werden die tragenden Konzepte der hier vorgestellten Vorgehensweise. Diese setzt damit in gewisser Weise eine Kontraposition zu bestehenden de facto Paradigmen – entsprechend kleinschrittig, aufmerksam moderiert und unter regelmäßiger Evaluation müssen die neuen Mechanismen eingeführt werden. Die technischen und organisatorischen Rahmenbedingungen müssen bewusst gestaltet werden.

Qualitätssicherung, Konfigurations-, und Projektmanagement

Die oben wiedergegebenen Ergebnisse der Analyse des üblichen Entwicklungsgeschehens legen nahe, dass sich die querschnittlichen Aktivitäten, die ein Software-Projekt üblicherweise begleiten in der Realität der Entwicklung direkt reaktiver Systeme oft ähnlich ad hoc gestalten oder gar gänzlich vernachlässigt werden.

Greift man den Aspekt der Konfigurationsverwaltung für die entstehenden Software-Artefakte heraus, wird erkennbar, dass auch dabei eine zumindest ungünstige Determinierung durch die Werkzeuge eine hemmende Rolle spielt. Die auf die visuelle Entwicklung ausgerichteten Werkzeugumgebungen erzeugen in der Regel keine textuellen Repräsentationen des Artefaktes, die es erlauben würden, klassische Mechanismen der kooperativen Software-Entwicklung, wie das Zusammenführen *merge* des gemeinsam bearbeiteten Quelltextfragments unterstützen zu lassen. Selbst Quelltextteile, wie Scripte sind dann in der Regel fester Bestandteil des monolithischen Projekts, welches nur zusammengefasst in einer proprietären, ausschließlich von der Entwicklungsumgebung selbst zu verarbeitenden, binären Datei abgelegt werden kann. Erst neuere Versionen von Autorenumgebungen erlauben es Quelltext extern als einfachen Text kodiert zu verwalten. Auch der Aufbau der entsprechenden Sprachen ist in der Regel nicht dazu geeignet den Quelltext zu modularisieren und auch physisch über mehrere Dateien zu dekomponieren.

Selbst Medienelemente werden von vielen Architekturen fest in das zentrale Artefakt eingebunden. Der Standpunkt, dass man auf diese Weise immer alle aktuellen Ressourcen in konsistenter Weise beisammen hat, ist in Anbetracht der Komplexität vieler Projekte sowohl bezüglich des entstehenden Systems als auch in Bezug auf die Zusammensetzung und ggf. die Verteilung des Projektteams kaum haltbar. Es ist nahezu unmöglich, pro Release (im Grunde nach jeder Änderung) jeweils eine komplette Version zu erzeugen und diese allen Betroffnen zur Verfügung zu stellen.

2.3. Softwaretechnologie

Auch wenn der hier vorgestellte Lösungsansatz vordergründig zum Ziel hat, die Belange der Anwendungsdomäne in den Fokus des Entwicklungsprozesses zu stellen und technologische Aspekte auf einen rein infrastrukturellen Charakter zu reduzieren, bleiben die adressierten Entwicklungsbemühungen im Kern Software-Entwicklungsprozesse. Die Ergebnisse müssen entsprechenden Qualitätskriterien genügen und das Vorgehen muss in gleicher Weise wirtschaftlich und zielorientiert sein. Es wird schließlich auch einer technologischen Anstrengung und einer konkreten Werkzeuggrundlage bedürfen, um das Ziel der Individualisierung des Entwicklungsgeschehens für den jeweiligen Einsatzkontext zu erreichen.

In den folgenden Abschnitten werden dementsprechend bestehende Ansätze besprochen, die Versuchen die Kluft zwischen der Fachlichkeit und der technologischen Abstraktion zu schließen – teilweise durch die Kaprizierung auf die Unterstützung formeller konzeptioneller Phasen zugeschnitten auf die Software-Klasse und die Entwicklung entsprechend spezifischer Sprachen. Aber vor allem auch die Ansätze zur organisatorisch, administrativen Lösung der betrachteten Problemstellung werden untersucht. Insbesondere gesamtheitliche Vorgehensmodelle und Methoden, die die Einbeziehung von Nutzern und Domänenexperten in die Entwicklung zum Ziel haben und Aspekte der Arbeitsorganisation und der Personalentwicklung ins Kalkül ziehen, werden beleuchtet.

Sowohl Aspekte etablierter Paradigmen als auch spezifische Ansätze werden auf Ihre Relevanz bezüglich der hier adressierten Aufgabenstellung untersucht. Damit werden sowohl Grundlagen für die entwickelten Ideen geschaffen als auch die Abgrenzung zu verwandten Bestrebungen beschrieben. Die Anknüpfungspunkte zu den entwickelten Konzepten und Lösungen der vorliegenden Arbeit werden jeweils konkretisiert.

2.3.1. Life-Cycles

Der Abschnitt zu den Vorgehensweisen, die als weit verbreitete gängige Praxis während der Entwicklung interaktiver, direkt reaktiver Systeme betrachtet werden müssen (Abschnitt 2.2.4) macht deutlich, dass gerade auch bei der Ausgestaltung des Entwicklungsprozesses Handlungsbedarf besteht. In den folgenden Abschnitten werden die für die Erarbeitung einer Entwicklungsmethodik im hier adressierten Problembereich benötigen grundlegenden Ergebnisse der Forschung zu Vorgehensmodellen vorgestellt.

Ingenieurmäßige Vorgehensweisen

Es steht außer Frage, dass die hier adressierten Projekte Software-Produkte und damit technische Systeme als zentrale Zielartefakte anstreben. Die adressierten Entwicklungs-Bemühungen haben komplexe interaktive Systeme zum Ziel, an deren Realisierung interdisziplinäre Teams mitwirken und auf deren Gestaltung der wiederum interdisziplinäre Einsatzkontext aktiv Einfluss nimmt. Damit ist klar, dass es grundsätzlich nötig wird,

eine nachvollziehbar ingenieurmäßige Vorgehensweise zu wählen. Aufgrund der Spezifik der in dieser Arbeit diskutierten Entwicklungsumfelder wird eine einfache Anwendung eines der etablierten Standardvorgehensmodelle jedoch nicht tragfähig sein.

Insbesondere die an den Bedürfnissen der Technologen orientierten Abstraktionen bezüglich der Arbeitsabläufe und der Strukturierung sowohl des Projekts als auch des Produkts und der häufig induzierte hohe „Verwaltungsaufwand“ sind sehr wahrscheinlich Hemmnisse für die Entfaltung des nötigen kreativen Prozesses. Die grundsätzlich durch die gängigen Modelle angestrebte Trennung der Problembereiche (*separation of concerns* vgl. [Boe81] Seite 14) und besonders zwischen der Nutzung und der Entwicklung widersprechen dem Anliegen, einen durchgängig an der Nutzung und Nutzbarkeit der Software ausgerichtete Entwicklung durchzuführen.

Eine Abweichung von den herkömmlichen Zielstellungen der Disziplin *Software Engineering* ergibt sich für die hier betrachteten Projekte aus dem Umstand, dass neben der reinen Wirtschaftlichkeit der Entwicklung und des sich ergebenden Produktes, pädagogische, oder gar politische Aspekte und Forschungsinteressen Einfluss auf die Prioritätensetzungen nehmen.

Nichtsdestotrotz muss sich die zu entwickelnde Vorgehensweise an den grundlegenden Motivationen und Prinzipien der etabliertesten Ergebnisse des Software-Engineering orientieren. Qualitätsmerkmale, die üblicherweise bei der Software-Entwicklung zu berücksichtigen sind bilden schließlich auch für die Systeme der hier besprochenen Kategorie die entscheidenden Maßstäbe. Neben Merkmalen wie etwa der *Leistung bzw. Effizienz* und der *Verifizierbarkeit* (siehe [Kro97] für eine umfassende Liste) kommt einigen der Merkmale eine besondere Bedeutung zu (siehe dazu die Betrachtungen in Abschnitt 2.1.2). Die Konsequenz dessen ist in erster Linie die Notwendigkeit der Verschränkung ingenieurmäßigen Arbeitens mit den Standard-Prozessen redaktioneller Arbeit und den kreativen Prozessen von Autoren und Designern.

Die Grundsätze denen Ingenieursdisziplinen folgen um die Komplexität der Aufgaben zu bewältigen und die es gilt in Einklang zu bringen mit denen der anderen Disziplinen sind nach [Kro97]:

Abstraktion und Modelle: Die Entwicklung des System erfolgt über verschiedene Stufen der Abstraktion. So werden in frühen konzeptionellen Phasen Details der Umsetzung – für die Überlegungen und Planungen zunächst nicht relevante Aspekte – vorerst ausgeblendet. In der Regel werden dazu formale Modelle genutzt, die die Möglichkeit bieten sollen, möglichst präzise und widerspruchsfreie Spezifikationen zu erstellen. Diese bilden dann die Grundlage zum Teil formalisierter Transformationsprozesse hin zur nächsten Stufe der Konkretisierung und Vervollkommnung des Systems. Auf Grund der Tatsache, dass viele der in der Software-Entwicklung verwendeten Modelle im mathematischen Sinne formal sind, kaum redundant und auf die Sicherung von Konsistenz fokussieren, ist deren Anwendung in transdisziplinärer Teamarbeit entsprechend anzupassen. Dabei ist die Konzentration auf die Interaktivität und damit die Nutzerzentrierung auch während der Konzeption zu berücksichtigen. Ansätze dazu liefern die Forschungen zu *agiler Modellierung*

und zur kooperativen Prototypenentwicklung auf der Basis so genannter *Mock-Ups* (siehe weiter unten im Text).

Präzision: Der Wunsch der Ingenieursdisziplinen nach möglichst eindeutigen Definitionen sowohl bezüglich des Produktes als auch im Bezug auf die Vorgehensweise zielt auf eine möglichst große Determiniertheit der Entwicklung durch die Beteiligten. Dabei ist es einerseits wichtig, dass die ablaufenden Prozesse objektiv Analysierbar und damit steuerbar werden bzw. „Baugruppen“ (Komponenten bzw. Module) sich passgenau integrieren lassen, auf der anderen Seite müssen sehr komplexe Zusammenhänge der Fachdomäne und sehr dynamische sowie kreative Prozesse der Systementwicklung gestaltet werden. Die Vereinbarkeit von formaler objektivierender Präzision, wie sie sich etwa auch im oben beschriebenen Bedarf nach formaler Modellierung niederschlägt, und der Notwendigkeit emergente Lösungen in experimenten-orientierten kreativen Prozessen zu finden, ist eine Herausforderung der sich der Beitrag dieser Arbeit stellt.

Reduzierung der Komplexität: Stellvertretend für den Bereich des Software-Engineering geht [Kro97] geht davon aus, dass die Größe des angestrebten Produktes und der für dessen Erstellung notwenige Zeitaufwand sowie die damit verbundene Verantwortung zu verringern sind, indem man geeignete Methoden der Dekomposition anwendet: das Produkt in Module zerlegt, das Projekt in Phasen unterteilt und dem Team eine entsprechende Personalstruktur gibt. Aus der Perspektive der hier adressierten Projektkontexte bzw. Produkte müssen diese Vorstellungen dahingehend erweitert werden, dass neben der „bloßen“ Reduzierung die Beherrschung oder gar die Instrumentalisierung der Komplexität Ziel der Gestaltungsbemühungen für den Entwicklungsprozess werden. Die enge Orientierung an den zu unterstützenden Tätigkeiten der Zieldomäne und an der Arbeitsweise und Terminologie der technikfremden Disziplinen durch die Etablierung integrierender kooperativer Entwicklungsprozesse werden dabei zum Vehikel für die eines kontrolliert bzw. moderiert selbstorganisierenden Prozesses. Die Anwendung des Frameworks, das heißt seine Ausprägung zum System, erfordert darüber hinaus eine bestimmte Vorgehensweise, und impliziert damit die Dekomposition des zeitlichen Ablaufs der Entwicklung. Das Framework gibt ferner die Gliederung des Produktes grundsätzlich vor.

Standardwerkzeuge: Die Nachvollziehbarkeit und Reproduzierbarkeit der Entwicklungsprozesse setzt den systematischen Einsatz standardisierter Technologien und Entwicklungswerkzeuge voraus. Auch in diesem Punkt lassen sich die Ansprüche der Ingenieure nicht direkt auf das gesamte transdisziplinäre Entwicklungsgeschehen übertragen. Aus technologischer Sicht nutzt und setzt das Framework entsprechende Standards (z.B. durch die XML-Basierung). Aus Sicht der fachlichen Zugänge zum Entwicklungssystem, soll Adaptivität der Spezifik der Arbeitsweise und damit der Werkzeuge der jeweiligen Disziplin Rechnung tragen.

Standard-Methodologie: Für die Nutzung bzw. Etablierung standardisierter Techniken und Methoden gilt das Gleiche, wie für die entsprechend benötigten Werkzeuge. Einen grundsätzlichen Rahmen geben das Framework und eine entsprechende

> Methodik zu seiner Ausprägung vor. Insbesondere die rein technischen Aufgaben werden sich dabei tatsächlich mehr oder minder zur Routine entwickeln können und hinter die inhaltlichen und gestalterischen zurücktreten können. Die mit diesen Aufgaben verbunden Tätigkeiten jedoch variieren mit dem Wechsel der Domäne so erheblich, dass auch dabei projekt- oder zumindest domänenspezifische Vorgaben individuell erarbeitet werden müssen.

Neben den Grundsätzen der Arbeitsweise ist es jedoch auch Ziel, die zu entwickelnde Vorgehensweise an bestehenden Standards zu Vorgehensmodellen zu orientieren. So ist es sicher erforderlich die Strukturierung des Vorgehens entlang der Vorgaben etablierter Modelle für Software-Lebenszyklen vorzunehmen. Das heißt, dass die grundsätzlich zu erfüllenden Aufgaben (entsprechend der *core workflows* im *Unified Software Development Process* (UP) [JBR99]) denen herkömmlicher Projekte entsprechen und in bewährter Weise über den Ablauf von Phasen koordiniert werden müssen.

Es sind also die *Anforderungen des Nutzers zu analysieren.* Aus diesen leiten sich technische *Anforderungen an das System* ab. Diese wiederum münden in *Entwürfe* des Systems, die dann die Grundlage für die *Umsetzung* bilden. Am Schluss steht die *Evaluation* der Ergebnisse der Aktivitäten. Orientiert man sich am *Unified Software Development Process* laufen diese Aktivitäten iteriert über verschiedene Phasen der Entwicklung ab – je nach Projektfortschritt mehr oder minder stark ausgeprägt.

So dominiert im Zuge der Phase des Projektauftakts (*Inception* im UP), währen der Visionen zu Zielen verdichtet werden und eine grobe Architektur festgelegt werden, die Erhebung von Nutzeranforderungen. Lediglich Ansatzweise werden, Überlegungen zu technologischen Konsequenzen angestellt, erste Designentscheidungen getroffen und ggf. erste Prototypen implementiert. Das kann in mehreren Iterationen geschehen, die dann beispielsweise jeweils im Rahmen eines Reviews evaluiert werden.

Während der Phase der Ausarbeitung der Konzeption ((*Elaboration*) dominieren zunächst die Aktivitäten zur Erhebung der von Nutzerforderungen und parallel dazu die Ableitung der Systemanforderungen. Dabei mündet diese Analyse mehr und mehr in Entwürfe für das System. Das bedeutet die investierten Aufwände verlagern sich langsam. Auch die Arbeiten an der Implementierung laufen an.

Die Implementierung bildet dann die Hauptaufgabe in der Phase der Ausführung (*Construction*). Es finden zwar immer noch Rückkopplungen zum Anwender statt und damit Arbeiten, die zur Analyse der Nutzeranforderungen und der Systemanforderungen gehören. Es sollte jedoch nach und nach dazu übergegangen werden, Featurewünsche der Nutzer einzufrieren. Die Aktivitäten der Evaluation der Ergebnisse jeder Iteration intensivieren sich in Form von Modul- und schließlich Integrationstests.

Im Rahmen der Phase der Überführung der Software in die Nutzung (*Transition*) schließlich erfolgen intensive Integrations- und Systemtests. Gegebenfalls müssen die Implementierenden hierbei nochmals stärker aktiv werden zur Behebung der erkannten Fehler.

In Abbildung 2.12 ist die Verteilung der Aufwände über alle Phasen für ein typisches Projekt nach den Modellvorstellungen des Unified Process dargestellt. Die Stärke der

Aktivität, wiedergegeben durch die Amplitude der Grafen, veranschaulicht dabei lediglich idealtypische Größenordnungen und erhebt keinerlei Anspruch auf empirische Präzision. Gleiches gilt für die Anzahl der Iterationen innerhalb jeder Phase. Abhängig von den jeweiligen konkreten Gegebenheiten unter denen ein Projekt bearbeitet wird, ist dies jeweils individuell zu planen. Das erfolgt beispielsweise auch dynamisch, abhängig vom Verlauf des Projekts im Rahmen der Meilensteine, die am Ende jeder Phase obligatorisch sind.

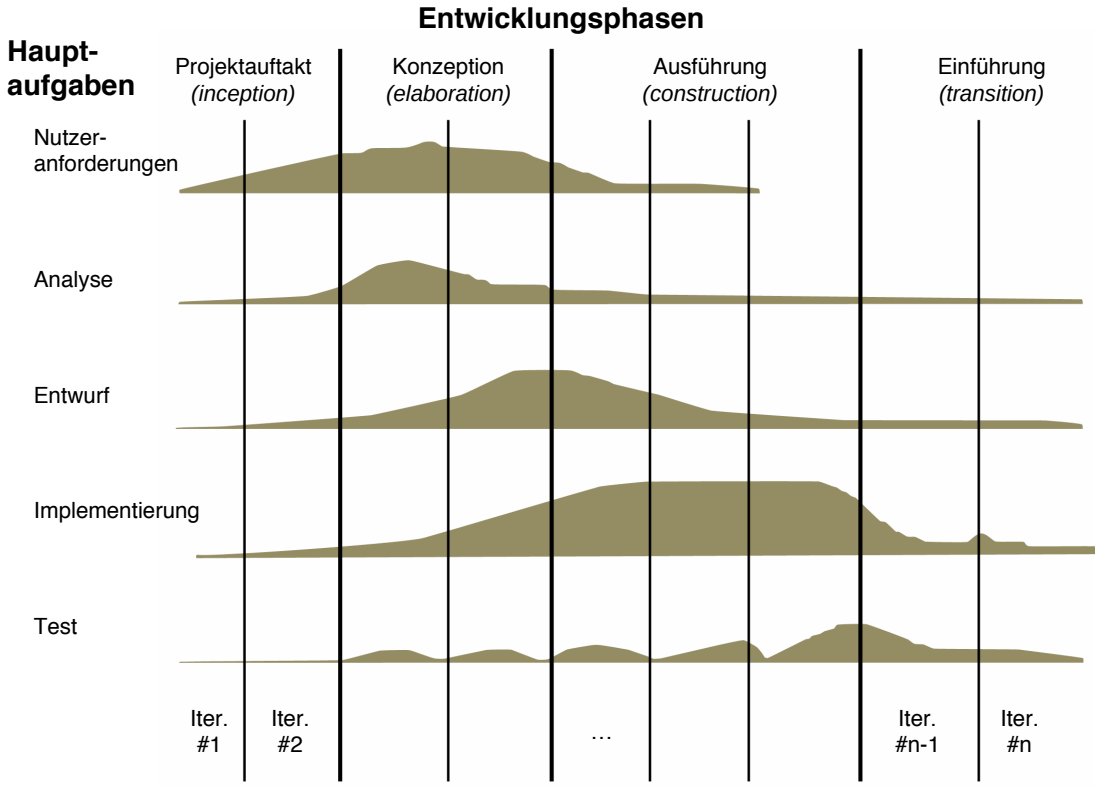


Abbildung 2.12.: Unified Software-Development Process nach [JBR99] Seite 11

Eine Methodik zur Entwicklung interaktiver direkt reaktiver Systeme in transdisziplinären Kontexten, also unter starkem Einfluss nicht technischer Anwendungs-Domänen und unter Mitwirkung entsprechender Disziplinen, hat also die Aufgabe die eben skizzierten bewährten Abläufe und Prinzipien der Software-Entwicklung zu vereinbaren mit den Arbeitsprozessen der anderen Disziplinen. Dabei handelt es sich unter Umständen um sehr formal gefasste Abläufe in Verlagen. Die eigentliche Herausforderung ergibt sich aber aus den Anforderungen der sehr dynamischen kreativen und damit kaum formalisierbaren Prozesse der fachlich-inhaltlichen sowie mediendidaktischen Konzeption und des Screendesign. Die angestrebte strenge Orientierung am Nutzungskontext und eine permanente Rückkopplung zu den Tätigkeiten, die es durch das System zu unterstützen gilt, machen es darüber hinaus erforderlich eine enge Verzahnung von Prozessen der Ent-

wicklung und Prozesse der Nutzung in der Gestaltung des Gesamtprozesses zu berücksichtigen.

Das bedeutet einerseits, dass die etablierten Softwareentwicklungs-Prozesse angepasst werden müssen. Andererseits werden aber auch die Methoden und der Instrumenteneinsatz der anderen beteiligten Disziplinen innerhalb eines co-adaptiven Prozesses umzugestalten sein. Der Einfluss der Nutzung des Frameworks und der dadurch implizierten Arbeitsschritte und insbesondere die Auswirkungen der intensiven Wiederverwendung von Infrastrukturkomponenten sind entsprechend zu untersuchen.

Agile Software-Entwicklung

Entwicklungsbemühungen, wie sie in Abschnitt 2.1 charakterisiert werden und die diese Arbeit adressiert, sind darauf angewiesen jeder Zeit flexibel auf veränderte Rahmenbedingungen – Sichtweisen, Erkenntnisstände etc. – reagieren zu können. Erfahrungen wie die in [GBL+04] präsentierten zeigen, dass agile Methoden auch in inhaltlich facettenreichen multidisziplinären Entwicklungsprojekten eine sinnvolle Alternative zu schwergewichtigen dokumentengetriebenen Entwicklungsverfahren bieten.

Während man bisher in Bezug auf übliche Software-Entwicklungsbemühungen davon ausgehen musste, dass die Kosten für Änderungen mit dem Projektfortschritt exponentiell steigen (vgl. [Boe81] Seite 40), die wichtigsten Entscheidungen als möglichst korrekt während der frühen Phasen der Entwicklung zu treffen sind, invertieren Autoren wie [Bec00] (Seiten 23 ff.) dieses Postulat einfach, indem sie die Konstanz der Kosten für Änderungen über die Zeit als Prämisse für die von ihnen entwickelte Methodik, das *Extreme Programming*, formulieren. Genau das ermöglicht es auch wichtige Entscheidungen erst dann treffen zu müssen, wenn sie tatsächlich anstehen.

Im Kontext forschungsorientierter transdisziplinärer Entwicklungen bedeutet das etwa, dass man sich auf die Einführung einer Funktionalität erst dann festlegen kann und muss, wenn man auf der Grundlage der in der Folge der Evolution von Kompetenzen und Einsichten erst spät im Projekt das Potential dieser Funktion erkannt hat. Es darf also in Projekten, die von der Emergenz von Ideen und Lösungen lebt keine Situation geben, in der es zu spät ist, um noch tief greifende, konzeptionelle Änderungen vornehmen zu können. Insbesondere auch die Implikation, dass damit tatsächlich nur die aktuell anstehenden Aufgaben gelöst werden müssen, es also nicht nötig ist, in vorauseilendem Eifer antizipieren zu müssen was benötigt werden könnte, hilft in diesem Kontext Hemmschwellen abzubauen. Mit der Orientierung der in dieser Arbeit entwickelten Vorgehensweise agilen Methoden wird dem Entwicklerkreis also möglich sich auf die Aufgaben zu konzentrieren, dem er mit dem aktuellen Kenntnisstand gewachsen ist.

Vor dem Hintergrund der weit verbreiteten Projektrealität, wie sie in Abschnitt 2.2.4 beschrieben wird, sind es insbesondere auch die durch XP etablierten Mechanismen der Minimierung von Risiken (von [Bec00] bzw. [GBL+04] als Grundproblem der Softwareentwicklung identifiziert), von denen die in Abschnitt 3.2 entwickelte Methodik

profitieren muss. Viele der von Beck angeführten Risiken ergeben sich aus der Tatsache das die klassische Software-Entwicklung monolithische Entwicklungsphasen abzuschließen: sowohl Zeitlich als auch bez. der Aufgaben. Die starke Entkopplung der beteiligten Disziplinen und damit in der Regel der Auftraggeber von den Auftragnehmern bietet zwangsläufig Anlass zu Missverständnissen.

Die Methodik Extreme Programmierung setzt sich also als primäres Ziel, Projektrisiken wie beispielsweise Terminverzögerungen, missverstandene Geschäftsziele oder falsche Funktionsfülle durch einen stark iterativen Prozess unter ständiger Rückkopplung zur Anwendungsdomäne und mit einer möglichst großen Evolutionsfähigkeit des entstehenden Systems zu minimieren. Revolutionär dabei ist das umstrittene Prinzip der Implementierungszentrierung. Das heißt dass umfangreiche Phasen der Konzeption, etwa Zwischenschritte der Modellierung, keine Rolle spielen. Es wird vielmehr möglichst früh im Projekt damit begonnen, am Zielartefakt zu arbeiten, also Quellcode zu formulieren. Nur damit scheint es möglich zu sein, gefundene Lösungen direkt gegen den Bedarf der Domäne zu prüfen bzw. die Machbarkeit nachzuweisen.

Die Methodik gründet sich auf vier Werten, die das Software-Entwicklungsgeschehen auf der Grundlage von XP prägen sollten:

Kommunikation: Die Erkenntnis, dass verbaler Kommunikation von Mensch zu Mensch zumal zwischen Vertretern unterschiedlicher kultureller Kontexte (Disziplinen, Organisationen, Funktionen, etc.) anfällig ist für das Entstehen von Missverständnissen führt zur Forderung nach dem Einsatz von Praktiken, die die Kommunikation fordern und damit fördern.

Einfachheit: In dem in dieser Arbeit betrachteten Entwicklungskontext sind viele der benötigten Lösungen kaum rein theoretisch zu antizipieren. Erst Experimente zeigen die Brauchbarkeit medialer bzw. informationstechnologischer Umsetzungen von fachlichen Konzepten. Umso wichtiger wird es sein, sich am von XP formulierten Wert der it „Einfachheit“ zu orientieren, der es gebietet zu vermeiden, alles technische Machbare und alles fachlich Wünschbare zu realisieren. Insbesondere der angestrebte evolutionäre selbstorganisierende Charakter des Entwicklungsprozesses (etwa im Sinne von [DR97]) braucht die Führung entlang dieses Wertes, um eine Konvergenz der Entwicklung in Richtung der Zielstellung sicherstellen zu können.

Feedback: Um Unklarheiten und damit Spekulationen jeglicher Art, beispielsweise bezüglich der Qualität des Entstehenden Systems, des Fortschritts der Entwicklung oder aber in Bezug auf Kundenwünsche zu vermeiden, soll es in XP-Projekten möglich sein, solche Größen jeder Zeit konkret zu „messen“. Das bedeutet etwa, dass der Umfang und die Qualität des Zielsystems permanent durch automatisierte Tests objektiviert werden können. Durch kurze Release-Zyklen kann die Tragfähigkeit der Funktionalitäten immer direkt im Einsatz in der Anwendungsdomäne evaluiert werden und die Ergebnisse dessen in die jederzeit mögliche Überarbeitung zurückfließen.

Mut: In einem Software-Entwicklungsumfeld, indem ein großer Teil der Projektbeteiligten keine systematische Ausbildung zum Software-Ingenieur oder Programmierer genossen hat oder gar Berührungsängste im Umgang mit Informationstechnologie hat, ist es wichtig it „Mut“ als Wert zu erkennen und hochzuhalten. Dazu ist es entscheidend, die von XP geforderte Projektkultur zu etablieren, in der es an der Tagesordnung ist Fehler zu bekennen. Wichtig ist aber in erster Linie auch, dass jeder in der Lage ist „Besitzstände“ aufzugeben. Das heißt alle Projektbeteiligten müssen jeder Zeit bereit sein, sich neuen Ideen zu öffnen und diesen ggf. auch größere Investitionen, die sich als weniger tragfähig erwiesen haben, zu opfern.

Die Werte bedingen einander: So sind beispielsweise Mut und eine Kultur des offenen Umgang miteinander eine wichtige Voraussetzung für eine konstruktive Kommunikation. Oder nur ein möglichst präzises kurzfristiges Feedback erlaubt es, von der zunächst einfachsten Lösung auszugehen und nicht über alle eventuellen Notwendigkeiten spekulieren zu müssen.

Um diese Wertvorstellungen zum bindenden Maßstab für ein Projekt machen zu können, bedarf es der Einhaltung einiger Grundprinzipien. Diese bilden, wie es [Bec00] formuliert, Richtlinien für den neuen Stil der Entwicklungsarbeit. In der folgenden Liste werden diejenigen näher betrachtet, die für die Entwicklung der in Abschnitt 3.2 beschriebenen Vorgehensweise am inspirierendsten sind.

Lernen lehren: Auch wenn man dieses Prinzip fast schon als Binsenweisheit bezeichnen darf, ist es gerade als integrierter Bestandteil einer Projektmanagementstrategie im Bereich der Software-Entwicklung bisher kaum etabliert. Informelles Lernen und das Einarbeiten in neue Themen werden zwar stillschweigend Vorausgesetzt und im Rahmen der Entwicklung expliziter Aus- und Weiterbildungsprogramme [RH02] auch systematisch diskutiert und instrumentalisiert. In der Praxis ist es jedoch schwierig, das völlig informelle und ungesteuerte Selbstlernen gezielt und nachhaltig zu instrumentalisieren.

Gezielte Experimente: Der Charakter der von dieser Arbeit adressierten Projekte an sich – die relative Unbestimmtheit der Ziele, die Emergenz und die Dynamik in der Erarbeitung von Lösungen – erfordern ein exploratives Arbeiten. Ideen müssen möglichst direkt erfahrbar werden, mögliche Lösungen ausprobiert werden können. Was bei [Bec00] zunächst in der Hauptsache dem minimieren von Risiken, etwa dem Sicherstellen der technischen Machbarkeit dient, kann auch als Vehikel für den systematischen Erkenntnisgewinn verstanden werden. Die direkte Erfahrung und das direkte Feedback zum Wirken des eigenen Schaffens im Sinne der Tätigkeitstheorie und letztlich des Konstruktivismus hilft Gestaltungsspielräumen auszuleuchten und Grenzen zu erkennen. Das Verständnis der Software-Entwicklung als *Konstruktion von Realität* – also die gezielte Anwendung konstruktivistischer Ansätze als Grundlage für den nötigen Erkenntnisgewinn – geht im Wesentlichen auf [Flo92] zurück.

Inkrementelle Veränderungen: Behutsamkeit ist bei der Steuerung komplexer Systeme, bei denen kleinste Veränderungen an den Eingangsparametern unvorhersehba-

re Effekte am Endzustand nach sich ziehen können, ein wichtiges Prinzip. Überträgt man das von [Bec00] genutzte Gleichnis des Steuerns eines Wagens – man sollte nie ruckartige und heftige Steuerbewegungen machen – auf die Software-Entwicklung und die Entwicklung in transdisziplinären Umgebungen im Speziellen, sollte man ein Entwicklungsvorgehen etablieren, bei dem sowohl am System kleinschrittig gearbeitet werden kann, als auch die Integration der Einzelpersonen und die Entwicklung deren Kompetenzen sukzessive vollzogen wird. Im von XP skizzierten Idealfall ist die Dekomposition als Vehikel zur Herstellung von Übersichtlichkeit und Beherrschbarkeit so weit zu treiben, dass einzelne Teile des Systems aber auch die Schritte ihrer Erstellung so elementar sind, dass ihre Handhabung bzw. Umsetzung beinahe trivial sind.

Veränderungen wollen: Als eines der größten Risiken, welches die Transdisziplinarität induziert kann die Unvereinbarkeit von Anschauungen, Begriffsbildungen und Arbeitsweisen gelten. Aus der Tatsache dass beteiligte Disziplinen, etwa aus dem wissenschaftlichen Umfeld, oft auf lange erfolgreiche Traditionen zurückblicken können, leitet sich in der Regel der Anspruch ab, dass sich das Vorgehen in Projekten nach den jeweiligen Notwendigkeiten richtet. In der Software-Entwicklung wird es jedoch nötig sein, die Flexibilität die XP [Bec00] von den Programmierern fordert, in Teilen auch der Fachdomäne abzuverlangen. Das setzt voraus, dass man die von Beck vorgeschlagene Strategie, immer zunächst die vordringlichsten Probleme zu lösen, um offen zu bleiben für Veränderung, auch auf die konzeptionelle Arbeit der Domäne übertragen muss.

Um diesen Prinzipien folgen und damit eine Software-Entwicklung entlang der oben beschriebenen Werte etablieren zu können, bedient sich Beck einiger simpler Verfahren (Practices). In der konsequenten Anwendung ermöglicht deren Zusammenspiel die Erfüllung der vier grundlegenden Aufgaben der Software-Entwicklung (nach [Bec00] bzw. [ext05]): Planen (in [Bec00] Zuhören), Entwerfen, Programmieren, Testen. Im Folgenden werden diejenigen von den zwölf Verfahren kurz vorgestellt, die essenzielle Beiträge zu Konzepten dieser Arbeit leisten. Die Relevanz des Beitrags wird jeweils kurz erläutert.

Gemeinsame Verantwortlichkeit: Dass Software-Entwickler auch innerhalb reiner Programmierteams häufig eine Politik der Abschottung bez. des selbst verfassten Codes betreiben ist ein bekanntes Problem. Die individuelle Verantwortlichkeit kann laut [Bec00] aber auch eine Strategie sein, um die Vermischung unterschiedlicher Programmierstile und Missverständlichkeiten unter Kontrolle zu bringen. Verallgemeinert auf die Trennung der Zuständigkeiten zwischen Domänenexperten und der Technikentwicklung entspricht das der Errichtung der Brandmauer der Informatik [Dij89]. Die entstehenden Abhängigkeiten liegen auf der Hand: verlässt der allein verantwortliche die Entwicklerfirma, ist es nur mit Mühe möglich Veränderungen und Weiterentwicklungen daran vornehmen zu lassen. Verschärft ist das Problem noch auf der Ebene der durch die Brandmauer vom Auftraggeber isolierten Entwicklerfirma. Entgegen aller gängigen Paradigmen der Portfoliobereinigung und der Effizienzsteigerung durch Outsourcing der nichtfachlichen Aufga-

benbereiche (der Software-Entwicklung) zeigt die Praxis, dass eine permanente Einflussmöglichkeit aller Projektbeteiligten auf das Endprodukt wichtig ist. Dabei ist die geteilte Verantwortung, die Beck für den Quellcode fordert auszuweiten auf die Konzeption und die Integration des Zielsystems auf der Grundlage eines Frameworks.

Metapher: Eines der wichtigsten Prinzipien der in Abschnitt 3.2 vorgestellten Vorgehensweise des interaktiven kooperativen Prototyping ist das *gemeinsame Material*. Die Hintergründe und die verwendeten Konzepte zu diesem Ansatz sind in Abschnitt 2.3.6 ausführlich erläutert. Die Arbeit an einem gemeinsamen Material erfordert eine gemeinsame Sprache. Anstatt also mit Hilfe technisch geprägter, abstrakter Sprachelemente – wie etwa den Konstrukten der UML – zu arbeiten, um die Grundkomponenten und deren Beziehungen, also die Architektur des Systems zu beschreiben, hält es auch [Bec00] für sinnvoller, eine eigene, bereits an der Domäne orientierte Sprache zu finden. So hat sich im Projekt LeMOLernen beispielsweise herausgestellt, dass es das einzig Praktikable ist, über die *interaktive historische Zeitleiste*, *virtuelle Räume* oder etwa *digitale Arbeitsblätter* zu sprechen als über die zu Grunde liegenden immer gleichen technischen Konzepte (Pages) in jeweils leicht veränderter und dennoch stereotyper Ausprägung. Die Übersetzung der der Metaphern in konkrete Konstrukte der Software muss bei XP „stimmig beschrieben werden Mut“ [Bec00]. Einen dementsprechenden Ansatz verfolgt die Idee der Anpassbarkeit von Entwicklungszugängen an einem adaptiven Framework als instrumenteller Kern der Entwicklungsmethodik.

Kurze Release-Zyklen: Um tatsächlich ein möglichst unmittelbares Feedback zu gefundenen Lösungen zu erhalten, ist es wichtig, möglichst frühzeitig lauffähige Funktionalitäten, am besten direkt im praktischen Einsatz nutzen und damit evaluieren zu können. Der Vorschlag von XP dazu ist, möglichst kleine Inkremente der Software als abgeschlossene Versionen zu veröffentlichen. Das setzt voraus, dass eine entsprechende Dekomposition der Software möglich ist. Der Einsatz eines Frameworks, wie ihn diese Arbeit vorschlägt eröffnet dieser Praxis neue Möglichkeiten. Auf Grund der Charakteristik des Frameworks, ein komplettes Software-Skelett zu sein – schon ohne die Konkretisierung zur Befriedigung von Domänenanforderungen lauffähig – ist es sehr schnell möglich, einfache domänenspezifische Funktionalitäten im Kontext eines schlüssigen Gesamtsystems, mit einer kompletten Infrastruktur (Persistenz, Kommunikation, etc) als Unterbau zu präsentieren.

Fortlaufende Integration: Dieses Vorgehen steht in engem Zusammenhang mit dem eben diskutierten Punkt der Einhaltung *Kurzer Release Zyklen*. Die Möglichkeit, möglichst zu jeder Zeit einen Ablauffähigen Stand der Software erzeugen zu können und Einzelfunktionalitäten im Zusammenspiel mit dem Gesamtsystem prüfen zu können, ist ebenfalls ein entscheidendes Vehikel für direktes Feedback zur technischen Tragfähigkeit. Auch dieses Vorgehen wird durch den Einsatz eines Frameworks gleichsam erzwungen. Die Arbeit am zentralen, gemeinsamen Artefakt auf der Basis der bereits integrierten Gesamtarchitektur ist geprägt durch permanente

Integration von Einzelfunktionen oder gar die Konfiguration und Konkretisierung bereits integrierter Komponenten.

Kunde vor Ort: Das im Abschnitt zur partizipativen Entwicklung detaillierter ausgeführte Verfahren bildet die Basis für eine echte kooperative und an der Tätigkeit der Nutzer orientierte Software-Entwicklung. Die konkrete Ausgestaltung der Zusammenarbeit und die Frage, ob ein Kunde bzw. Nutzervertreter wirklich permanent physisch im Kreise der Entwickler sein muss sind diskutabel. Dabei ist der Wunsch von XP, permanent einen solchen „echten Kunden“ [Bec00], einen der das System schließlich nutzt, für inhaltliche Fragen zur Verfügung zu haben, ihn zum schreiben von Funktionstests einzusetzen oder für Entscheidungen zu Prioritätensetzungen heranzuziehen für viele Projekte nicht weit reichend genug. Gerade in Bereichen, bei denen eine Analyse von funktionalen Anforderungen die intellektuelle Leistung der Erforschung der Möglichkeiten des Einsatzes von Informationstechnologie erfordert, ist es unabdingbar, gemeinsam direkt am Gegenstand des Diskurses zu arbeiten. Physische Anwesenheit aller Beteiligten ist dabei möglicherweise sehr gut zu ersetzen durch Mittel der Kollaboration auf der Grundlage entsprechender technologischer Infrastrukturen. Das von Beck thematisierte, politische Problem, der Durchsetzbarkeit einer solchen Forderung gegen Widerstände, die sich aus wirtschaftlichen Erwägungen ableiten, muss durch eine Korrektur der Kultur des Verhältnisses zwischen Auftragnehmer und Kunde gelöst werden. Die entsprechenden Probleme dabei – etwa mit zu großer Transparenz und dem daraus resultierenden Risiko des „Bloßstellens“ von Prozessproblemen auf Seiten des Auftragnehmers – thematisiert etwa [GP05].

Programmierstandards: Es ist laut Beck nur möglich an einem gemeinsamen Artefakt zu arbeiten – die Verantwortung für den gesamten Quellcode zu teilen – wenn zumindest die Form der Nutzung der gemeinsamen „Sprache“ gleich ist. Das Potential für Missverständnisse, was sich aus unterschiedlichen Stilen ergibt, muss ausgeräumt werden. Bezüglich konkreter Maßnahmen bleibt [Bec00] spekulativ (sinngemäß): wenn ein Programmierer erst Teil eines erfolgreichen Teams, wird er unter Umständen bereit sein, seinen Stil anzupassen. Die Evidenz, dass der Einsatz eines Entwicklungsframeworks mit definierten Entwicklungsschnittstellen und zumindest für das konkrete Projekt standardisierte Regeln der Nutzung (Ausprägung) entsprechend erfolgreich sein wird, ist wesentlich höher.

Die häufig als Schwäche diskutierte Eigenschaft von XP, bezüglich der konzeptionellen Phasen der Entwicklung kaum methodische Unterstützung bieten, ist durch die Ergänzung um Verfahren der *Agilen Modellierung (AM)* [Amb02] gut zu korrigieren. Da diese Methodik den grundsätzlich gleichen Werten folgt und dazu die Akteure nach den gleichen Prinzipien handeln lässt wie XP fügen sich beide Verfahren nahtlos ineinander. Da es also bei der Ausgestaltung sehr komplexer Systeme oftmals schlechterdings nicht möglich ist sofort an einer lauffähigen Software zu arbeiten (siehe zur Argumentation etwa [BvK02] Seite 63), ist es nötig, eine Phase der strukturierten inhaltlichen Stoffsammlung und Modellierung vor die eigentliche Implementierung zu schalten.

Die Verfahren (auch beim AM *practices* genannt) die dabei definiert werden, um die Modellierung entlang dieser Prinzipien zu leiten bedienen sich ähnlicher Mechanismen, wie die Ansätze zur kooperativen Software-Entwicklung, die die weiter unten vorgestellte „Skandinavischen Schule“ (siehe etwa [BGK93]) einführt. Diejenigen Verfahren die auch für die Initialisierungsphase eines Projektes auf der Grundlage des hier präsentierten Vorgehens relevante Grundlagen liefern und die über die analogen Verfahren im XP qualitativ hinausgehen sind (nach [Amb02]):

Entwickle verschiedene Modelle (Sichten) gleichzeitig: Der Umstand, dass man sich während der Entwicklung immer mit unterschiedlichen Aspekten des Diskursbereichs auseinanderzusetzen hat und man gerade in agilen Entwicklungskontexten auch oft auf unterschiedlichen Abstraktionsebenen agieren muss, ist durch die transdisziplinäre Arbeitsweise innerhalb der hier betrachteten Projekte in besonderem Maße zu berücksichtigen. So wird man wiederholt an den fachlich-inhaltlichen Konzepten arbeiten: beispielsweise inhaltliche Strukturen und damit etwa Navigationspfade und Präsentations- bzw. Interaktionsschemata anpassen. Während zur gleichen Zeit andere Teile des Teams das Design und die Ergonomie von Nutzerdialogen diskutieren und wieder andere an der Implementierung bereits spezifizierter Funktionalitäten arbeiten. Der von AM aufgenommene Impuls für die hier vorgestellte Methodik ergibt sich aus der Schlussfolgerung, dass die von der jeweiligen Aufgabe und dem entsprechenden Verständnis abhängigen unterschiedlichen Sichten auf das Zielsystem jeweils geeignet zu repräsentieren sind. Idealerweise werden die verwendeten Sichten den Anforderungen, die sich aus unterschiedlichen diversen Ansprüchen der beteiligten Disziplinen ergeben, individuell angepasst.

Beziehe Betroffenen aktiv ein: Diese Herangehensweise verschärft die Forderung von [Amb02] nach kollaborativer Modellierung in Teams wobei Beteiligte etwa wechselseitig voneinander lernen. Ambler bezeichnet das Verfahren ferner als eine Ausweitung des Extreme-Programming-Verfahrens der „Kunde vor Ort“ [Bec00]. Er geht davon aus, dass das permanente zur Verfügung stehen von Nutzern bzw. Nutzervertretern zur Beantwortung von Rückfragen für eine effektive Software-Entwicklung zwar notwenig, in vielen Organisationen jedoch nicht ausreichend ist. Dies gilt insbesondere dann, wenn komplexe, politisch bestimmte Projektkonstellationen eine Zielgerichtet Entwicklung, bei der alle Beteiligten an einem Strang ziehen, negativ beeinflussen. Die Agile Modellierung fordert viel mehr die echte aktive Einbeziehung aller maßgeblichen Stakeholder um tatsächliche Verbindlichkeit zwischen den Beteiligten in Bezug auf das gemeinsame Ziel und das entsprechende Bewusstsein zu schaffen. Dazu muss das Management den Nutzen und den zu erzielenden Mehrwert nachvollziehen können und anerkennen sowie schließlich voll zu dem Projekt stehen. Das setzt voraus, dass es über das Lesen von Dokumenten und die Teilnahme an Besprechungen hinaus, bereit ist, sich aktiv mit der Materie auseinanderzusetzen. Gleiches gilt für Personal, was perspektivisch für den betrieb, Wartung und Support zuständig sein wird: ist dieses direkt an der Entwicklung beteiligt, hat es am effektivsten die Möglichkeit das System intensiv kennen zu lernen und zu verstehen – ihre eigenen Anforderungen gezielt mit den Anforderungen des Systems in Übereinstimmung zu bringen. Diese von Ambler propagierte Ef-

fektivität dieser Praktik liefert ein zentrales Argument für die Charakterisierung der in dieser Arbeit vorgestellten Methodik als kooperativ und interaktiv mit der Implikation der Ausweitung dessen auf alle maßgeblich Beteiligten bzw. Betroffen.

Stelle einfache Inhalte (create simple content) einfach dar (depict models simply): Das von [Bec00]für das extreme Programming geforderte Verfahren immer das einfachste zu entwerfen, was gerade die bestehenden Anforderungen erfüllt, ohne über möglicherweise in der Zukunft Benötigtes zu spekulieren wird hier auf die ebene der fachlichen und inhaltliche Konzeption des Systems gehoben. Das heißt einerseits, dass verwendete Abstraktionen zur Repräsentation von Aufbaustrukturen (Diagramme für Datenstrukturen oder Formularschemata der Nutzerschnittstelle) sowie Abläufen (Skizzen für Interaktionsmuster etc.) und den entsprechend verwendeten Metaphern nicht überfrachtet werden sollten. Es bedarf also einer Darstellungsform, die in ihrer Syntax unkompliziert und idealerweise an die individuellen Rezeptionsweisen der Disziplinen anpassbar ist. Andererseits müssen die auf dieser Grundlage entstehenden Darstellungen inhaltlich möglichst einfach gehalten werden und nur die tatsächlich benötigten Ideen transportieren. Dabei ist für transdisziplinäre Kontexte sicher zu ergänzen, dass die Konzeption allen Verständlich expliziert sein sollte. So ist die Forderung, auf Redundanzen zu verzichten, zumindest in dem Sinne der Erhöhung der Verständlichkeit und Transparenz diskutabel.

Nutze einfachste Werkzeuge: Die größte Barriere für die aktive Einbeziehung von Projektbeteiligten, die nicht originär Software-Entwickler sind, in die Prozesse der Entwicklung ergibt sich oft auch aus dem Fehlen einer gemeinsamen Entwicklungsumgebung. Insbesondere der Einsatz integrierter Software-Werkzeuge, etwa zur Modellierung, setzt oft spezialisierte Kompetenzen und umfangreiche Erfahrungen voraus, die bei den nicht technisch orientierten Team-Mitgliedern nicht erwartet werden können. Um jedoch möglichst viele Betroffene einbeziehen zu können, ist es nahe liegend die Wahl der Arbeitsmittel auf möglichst Elementares zu beschränken. Ambler schlägt in [Amb02] dazu beispielsweise vor, mit Handskizzen an der Wandtafel zu arbeiten um Diagramme zu visualisieren oder mit Klebezetteln einfache Oberflächenprototypen zu entwerfen. Ambler geht davon aus, dass es möglich ist die Mittel der Modellierung derart zu wählen, dass jeder damit arbeiten kann. Die direkte taktile Erfahrung mit dem Diskursgegenstand hält er dabei für ein entscheidend. Den Erfolg seines Ansatzes sieht er darüber hinaus in der hohen Flexibilität begründet. Je nach dem individuellen Bedarf der jeweiligen Projektsituation können die Instrumente geeignet gewählt werden. Das bezieht sich auch auf die Handhabung direkt – es ist überall und jeder Zeit möglich mit Bleistift und Papier zu arbeiten (vgl. [Amb02] Seite 103). Es sind genau diese Eigenschaften, die es auf den hier entwickelten Ansatz zu übertragen und auszunutzen gilt. Zweischneidig ist dagegen das Argument Amblers, dass konzeptionelle Zwischenergebnisse in der Regel eine „Einweg“-Funktion haben, die lediglich der Illustration und Dokumentation dienen, und dadurch auch nur der Einsatz möglichst kostengünstiger Mittel angezeigt ist. Die Diskrepanz zwischen den Kostenerwägungen und der Etablierung der nötigen Nachhaltigkeit wird durch

den in der vorliegenden Arbeit popagierten Ansatz eher konstruktiv behandelt um der unbestritten notwendigen Nachvollziehbarkeit (Traceability siehe etwa [Rup02] Seite 407) des Weges von den Anforderungen bis hin zu den jeweiligen Realisierungen besser Rechnung tragen zu können – also jeder Zeit eine Verbindung zwischen den Artefakten der Explizierung von Ideen und der eigentlichen Umsetzung durch eine direkte transformationsorientierte Kopplung aufrechtzuerhalten.

Neben den inhaltlichen Gesichtspunkten der hier vorgestellten agilen Herangehensweisen an die Software-Entwicklung bilden strukturelle Aspekte ihrer Beschreibung eine wichtige Grundlage für die Erarbeitung der Methodik des interaktiven Prototyping. Die Konstruktion einer Entwicklungsmethodik entlang der Struktur:

Werte → Prinzipien → Verfahren

dient als Vorbild für die Ausarbeitung der Vorgehensweise auf der Grundlage eines Frameworks. Dazu werden im Wesentlichen die eben beschriebenen Kategorien genutzt und für den spezifischen Entwicklungskontext erweitert bzw. konkretisiert und interpretiert. So wird in Abschnitt 3.2.1 beispielsweise die Frage beantwortet, welche Bedeutung den Werten *Kommunikation, Einfachheit, Feedback und Mut* im Rahmen transdisziplinärer Projekte zukommt und welche Prinzipien und Verfahren zum tragen kommen müssen, um den daraus resultierenden Ansprüchen genügen zu können.

2.3.2. Spezialisierte Ansätze

Nachdem die allgemeinen Prinzipien und Ansatzpunkte für die Erarbeitung einer Vorgehensweise für die Entwicklung von interaktiver direkt reaktiver Software in transdisziplinären Entwicklungsumfeldern erörtert wurden, sollen nun Ansätze beleuchtet werden, die den Eigenarten der adressierten Problemdomäne durch konkretere Maßnahmen und Mechanismen Rechnung zu tragen suchen. Die Tatsache, dass der hier untersuchte Anwendungsbereich die Präsentation fachlicher Inhalte und die intensive Interaktionen zwischen System und Nutzer als zentrale Aufgabe der Software-Unterstützung betrachtet, setzt eine Orientierung an entsprechenden Bemühungen zur effektiven Gestaltung der Produktionsprozesse von Multimedia-Anwendungen, Angeboten elektronisch unterstützen Lernens sowie Hypertext-Systemen voraus.

Hypermedia-Storyboards

Im Rahmen der Dissertation von Geukes [Geu00] wird erstmalig dezidiert die „kulturelle Divergenz“ zwischen Autoren, Technikern und Designern als Kern der Problematik identifiziert und der Versuch unternommen, ein integrierendes Vorgehensmodell und eine gemeinsame Entwicklungsplattform zu schaffen, die von allen Beteiligten gleichermaßen genutzt werden kann und es insbesondere den der Technik fern stehenden Autoren ermöglichen soll, direkt in die Umsetzung eines Systems zur Vermittlung von Wissen einzugreifen.

Geukes geht bei seinen Ansatz ähnlich, wie es die hier vorgestellte Lösung vorschlägt, von einem Framework aus – einem allgemeinen Applikationsrahmen, der sowohl den technischen (Laufzeitumgebung) als auch den methodischen Rahmen für ein – hier ist die Einschränkung manifest – Lernsystem vorgibt. Darunter sind zusammengefasst: Regeln für die Interaktion und die Navigation, sowie Festlegungen für das Layout.

Ausgehend von einem so genannten Wissensmodell werden durch Prototyping in zwei Phasen entsprechende Konzepte spezifiziert. Jeweils automatisierte Schritte der Generierung sollen dann die Übergänge zwischen den Phasen unterstützen: Strukturprototypen, Interaktionsprototypen und Layoutprototypen als Ergebnis der ersten Phase sollen so etwa als Templates (Vorlagen wie Styleguides) für die Generierung so genannter Frames, einzelner entsprechend der Templates funktionierender konkretisierter Masken bez. Bildschirminhalte genutzt werden. Die Anwendung mit allen Masken, ihren Funktionalitäten, Navigationselementen und den grundsätzlichen Ablaufsteuerung ist damit spezifiziert. Diese Spezifikation wiederum wird dann im Schritt der eigentlichen Implementierung mit den Inhalten gefüllt. Der Übergang von der Spezifikation zu eigentlichen Softwaremodulen und Medienelementen soll dabei wieder weitgehend generativ unterstützt werden.

Um dieses Instrumentarium für den Produktionsprozess unter enger Einbeziehung von Fachautoren nutzbar zu machen, schlägt er eine formale Methode zur inhaltlichen und funktionalen Beschreibung des Systems durch die Autoren vor. Geukes' Strategie umfasst also eine eigens definierte Sprache, die es Autoren ermöglichen soll, fokussiert auf die für sie disponiblen Elemente, Funktionalitäten zu implementieren bzw. Inhalte aufzubereiten und zu inszenieren. Die dazu genutzte Abstraktion bezeichnet er als *Hypermedia-Storyboards*, die aufbauend auf einer formalen Notation, ähnlich einer Programmiersprache, Abläufe und mögliche Interaktionen im System beschreibbar macht. Die Schlüsselkonzepte orientieren sich dabei an den Aspekten der Präsentation. Das heißt beispielsweise, es können Interaktionselmente wie *Buttons*, entsprechende *Events* sowie Präsentationselemente wie *Pictures und Hypertext* innerhalb von *Frames* definiert bzw. arrangiert werden.

Mit folgendem Konstrukt etwa:

```
<frame_init> -> <play_video> "Bürgerinitiative"
```

konnte festgelegt werden, dass beim erstmaligen Anzeigen eines Frame (also einer Bildschirmseite) ein Video mit dem Titel „Bürgerinitiative“ abgefahren werden soll. Wie beschrieben, wurden die inhaltliche Einordnung des Bildschirms sowie das grundsätzliche Erscheinungsbild und die Interaktionsfähigkeit etwa des Videoplayers in der initialen Phase der Entwicklung konzipiert und in der zweiten grundsätzlich vorgegeben. Genau das sollte die fachliche Fokussierung an dieser Stelle ermöglichen.

Den Versuch, damit den Brückenschlag zwischen fachdidaktisch und inhaltlich motivierten Anforderungen und dem technisch Möglichen und Notwendigen zu schaffen erklärt Geukes allerdings selbst für gescheitert: „Den wissenschaftlichen Autoren gelang

es nur unzureichend, sich damit angemessen auszudrücken.“ ([Geu00] Seite 160). Seiner Einschätzung nach, ist dies dadurch zu begründen, dass einerseits die Abstraktionsstufe zu hoch ist, um tatsächlich flexibel arbeiten zu können. Andererseits hielten die angesprochenen Autoren die Sprache für zu kompliziert und technisch orientiert. Dementsprechend belässt er es bei der Formulierung dieser Vorgehensweise als Strategie und nimmt die Umsetzung einer entsprechenden Werkzeugumgebung nicht in Angriff.

Entscheidende Konsequenzen der Erkenntnisse von Geukes ergeben sich für die vorliegende Arbeit hauptsächlich bei der Setzung der Prämissen. In völliger Übereinstimmung mit den Ergebnissen der Analyse zu den de facto Standardvorgehensweisen auf der Grundlage bestehender Entwicklungssysteme geht Geukes davon aus, dass: „nur sehr wenige wissenschaftliche Autoren z.B. einer Hochschule eines der etablierten Autorensysteme zur Entwicklung komplexeren Lehrmaterials einsetzen würden“ ([Geu00] Seite 160).

Er rechnet aber andererseits damit, dass sowohl die Entwicklung der etablierten Werkzeuge der Zielgruppe der Autoren grundsätzlich entgegenkommen muss und wird. Dass auch die Autoren perspektivisch nicht umhinkommen werden, sich neuen Technologien zu öffnen, ihre Arbeits- und Denkweisen oder gar das gesamte Selbstverständnis ihrer Profession auf den Einsatz von Informationstechnologie einzustellen, hält Geukes für ebenso notwendig wie wahrscheinlich. Als Gründe nennt er die Notwendigkeit der Beschleunigung der Entwicklung in Zeiten immer kürzer werdender Halbwertzeiten des zu repräsentierenden Wissens. Außerdem analysiert er die Möglichkeit der fortschreitenden technologischen Dominanz über die fachlichen Notwendigkeiten und antizipiert das Risiko für die Wahrung der inhaltlichen Qualität der entstehenden Produkte.

Neben dem angesprochenen kulturellen Wandel erwägt Geukes eine systematische Annäherung der Fachdomäne und der Technikentwicklung primär auf der instrumentellen Ebene, also bei der Ausgestaltung der Werkzeuge. Er meint, eine möglichst große Vielfalt an vorgefertigt angebotenen kleinteiligen Bausteinen – „funktionellen Primitiven“ ([Geu00] Seite 161) – könne dazu führen dass die Autorensysteme universell genug würden, um den Anforderungen einer Vielzahl von Autoren verschiedener Disziplinen tatsächlich gerecht werden zu können.

Diese These ist aus zweierlei Gründen angreifbar: es darf zum einen bezweifelt werden, dass ein solcher Prozess der wechselseitigen Adaption ohne gezielte methodische Unterstützung tatsächlich erfolgreich verlaufen wird in dem Sinne, dass sich für vergleichbare Projektsituationen reproduzierbare effektive Entwicklungsprozesse intuitiv ableiten lassen. Zum anderen lässt der Autor offen, wie genau er den durch den Einsatz von Primitiven erreichten Grad an Flexibilität und Allgemeingültigkeit mit den hoch diversen und individuellen Ansprüchen einzelner Fachdomänen vereinbaren will.

An dieser Stelle setzt der Lösungsvorschlag der vorliegenden Arbeit an, indem er die systematische Konvergenz zwischen den Bedürfnissen und Fähigkeiten der Stakeholder der sowie den strukturellen und prozessualen Erfordernissen der Anwendungsdomäne einerseits und den Möglichkeiten und Notwendigkeiten der Technologieentwicklung andererseits systematisch und individuell für jede neue Projektkonstellation forciert. Den

wechselseitigen Anpassungsprozess den Geukes für die Zukunft als Grundsätzlichen Kulturwechsel erhofft – durch die fortschreitende allgemeine Durchdringung aller Lebensbereiche durch IT – nutzt der hier beschriebene Ansatz als zentrales Vehikel.

HDM

Die Arbeit [GPS93] stellt einen modellbasierten Ansatz für das Vorgehen bei der Entwicklung von Hypermedia-Anwendungen vor. Das Konzept der Hypermedia- bzw. Hypertext-Anwendungen kann grundsätzlich als äquivalent zu dem hier betrachteten Begriff der interaktiven, direkt reaktiven Anwendung gesehen werden. Mit der Ausweitung der Konzepte des so genannten *HDM* in [SR98] bzw. [SRB96] zum *OOHDM* wird vorrangig die Design-Phase der Entwicklung großer web-basierter Anwendungen adressiert. Es wird dabei von der Prämisse ausgegangen, dass traditionelle Entwicklungsmethoden für Software keine sinnvoll nutzbaren Abstraktionen bieten, um Applikationen die sich zentral der Hypertextmetapher bedienen zu spezifizieren. Beispielsweise wird bezweifelt, dass Konzepte wie *Links* oder Aspekte der Gestaltung des Nutzer-Interface adäquat mit Hilfe bestehender Modellsprachen — etwa der UML – repräsentiert werden können. Es werden neben Modellbildungen zur Wiedergabe von *Navigationsobjekten* Abstraktionsmechanismen zu deren Anordnung innerhalb so genannter *Navigationsräume* definiert. Aspekte der Nutzerschnittstellengestaltung werden dabei klar getrennt von den Belangen der Navigation. Wie der Name der Methodik nahe legt, wird an objektorientierte Vorgehensweisen angeknüpft – für das Design der Navigation und das abstrakte Design der Nutzeroberflächen werden neue Notationen erarbeitet, die sich in UML-Diagramme einbetten lassen.

Die Einführung zusätzlicher Sprachkonstrukte in die UML, die spezifisch sind für interaktive Anwendungen und etwa Interaktionselemente explizit versinnbildlichet, bewältigt das Problem der Verständlichkeit technischer Abstraktion ebenso unzureichend, wie die Metaphern der Autorenwerkzeuge selbst. Entscheidend ist jedoch die Einsicht, dass die Unterstützung von Phasen der Konzeption auch in agilen Entwicklungsumfeldern zur Bewältigung der Komplexität unabdingbar ist. Die zu wählende Modellsprache jedoch sollte sich an die Bedürfnisse und Sichtweisen der eigentlich Kreativen (also der Fachexperten) innerhalb dieses Prozesses anpassen lassen.

Die Erarbeitung der Nutzeranforderungen und deren Überführung in einen konzeptionellen Entwurf und weiterhin in das Navigationsdesign wird von der in [GSV00] vorgestellten Methode unterstützt. Sie versucht damit die Lücke zu schließen, die *OOHDM* zwischen der Anforderungsanalyse und einem Feinentwurf des Systems bzgl. des systematischen Vorgehens lässt. Die Synthese des Navigations-Designs (entsprechend dem Verständnis von [SR98]) erfolgt dabei in einfachen Phasen ausgehend von einer Szenario-Spezifikation und einer Use-Case-Spezifikation. Das geschieht im Wesentlichen durch die schrittweise Transformation der Modelle entlang einfacher heuristischer Regeln.

In Richtung Objektorientierung

Macromedia gelingt es mit der Einführung von Actionscript 2, dem technisch versierten Entwickler das Arbeiten nach objektorientierten Paradigmen zu ermöglichen. Damit ist zunächst überhaupt die Grundlage geschaffen, um aus einem systematischen Entwicklungsprozess (z.B. dem Rational Unified Process unter Einsatz objektorientierter Methoden und Notationen, wie der UML) nahtlos, ohne Modellbrüche und Hilfskonstruktionen zu einer Implementierung zu gelangen.

Es wird nun also zumindest bezüglich der Funktionalitäten möglich, die entsprechenden Methoden des Requirements Engineering – etwa das Object Engineering nach [Rup02] – zu nutzen. Konkrete Ansätze zum Schließen der Kluft zwischen den rein technikorientierten, ingenieurmäßigen Vorgehensweisen und der Arbeit von Autoren, Lektoren und Gestaltern bietet dies jedoch nicht. Im Gegenteil, die Fokussierung auf die Belange der IT-Experten verstellt mit großer Wahrscheinlichkeit den Blick auf die Probleme der interdisziplinären Integration in einem komplexen Projekt der hier adressierten Kategorie.

Der in [DEM+99] beschriebene Ansatz versucht mit einer ähnlichen Herangehensweise, gleiche Voraussetzungen schaffend, gezielt eine Verbindung zwischen den Modellbildungen der Entwurfsphasen und den meist monolithischen Implementationsmodellen handelsüblicher Autorenumgebungen zu schaffen. Diese konzeptionelle Kluft wird überwunden. Das behebt jedoch auch lediglich den Bruch, der sich ergibt, wenn Autorenwerkzeuge im Kontext klassischer objektorientiert betriebener Entwicklungsbemühungen zur Implementierung genutzt werden müssen. Die Konzeptualisierung, die das Metamodell für den objektorientierten Entwurf vorgibt, bleibt als Basis für den Austausch mit nicht technisch orientierten Projektbeteiligten problematisch.

2.3.3. Partizipation

Bei der Entwicklung interaktiver, hoch reaktiver Systeme ist es grundsätzlich nahe liegend, Nutzer intensiv in den Prozess der Entwicklung einzubeziehen. Darüber hinaus erscheint es sinnvoll zu untersuchen, inwieweit die entsprechenden methodischen Ansätze übertragbar sind auf die Integration von Domänenexperten unterschiedlicher Spezialisierung, die nicht notwendigerweise den Nutzungskontext vertreten, zu transdisziplinär arbeitenden Entwicklerteams. Partizipation der technikfernen Disziplinen auch bei den technischen Aspekten der Entwicklung ist das zentrale Instrument zur Beherrschung der Transdisziplinarität innerhalb der in dieser Arbeit adressierten Projekte. Die zu entwickelnde Methodik wird sowohl bei den in diesem Abschnitt beschriebenen fundamentalen Ansätzen Anleihen machen, als auch fokussierter bei den im Kapitel 2.3.4 vorgestellten Entwicklungs-Modellen.

Die Einbeziehung von Nutzern in den Produktionsprozess von Software hat eine lange Geschichte und entsprechend viele Facetten. Die bereits 1986 zusammengestellten Ansätze zum so genannten *User centered design* [NDH86] konzentrieren sich dabei ganz

grundsätzlich auf die Einbindung von Nutzern in die Prozesse der Systementwicklung während der Entwicklungszeit (*design-time*) [FGY+04]), weniger jedoch auf die Gestaltung „lebender“ Systeme, die auch in dynamischen Nutzungskontexten flexibel bleiben für individuell benötigte Änderungen durch den Nutzer (vgl. [FGY+04]). Der Nutzer bleibt also eher in einer reaktiven Rolle. Er ist Ansprechpartner für Rückfragen der Entwickler und die Beurteilung der durch die Techniker erstellten Lösungen.

Die präzise begriffliche Abgrenzung zwischen den Konzepten „einfacher“ Einbeziehung von Nutzern (Partizipation), echter Kooperation und dem End-user Development (siehe Abschnitt 2.3.4) ist für die vorzustellenden Konzepte von untergeordneter Bedeutung. Ziel der folgenden Abschnitte ist es viel mehr, Methoden und Herangehensweise zusammenzutragen, die explizit einen Beitrag dazu leisten transdisziplinär Software zu entwickeln und bewusst Synergieeffekte daraus zu ziehen, dass „die Beteiligten durch ihre verschiedenen Sichten das zu Lösende Problem vollständiger und allseitiger erfassen, konkretisieren und formulieren können“ [DR97] und durch eine möglichst direkte Umsetzung und entsprechende Rückkopplung zwischen fachlichen Anforderungen und technischen Möglichkeiten Risiken zu minimieren. Durch die Integration von Wissen und Methoden der beteiligten Disziplinen (vergleiche wiederum [DR97]) werden Gestaltungsspielräume verdeutlicht bzw. sogar eröffnet. Wie die einzelnen Konzepte in die Ansätze dieser Arbeit einfließen ist im jeweiligen Unterabschnitt kurz umrissen. Die konkrete Adaption und Umsetzung erfolgt dann in Kapitel 3.

XP-Verfahren: „Planungsspiel“, „Metapher“, „Kunde vor Ort“, „Test“

Die bereits in Abschnitt 2.3.1 aus dem Blickwinkel der Agilität vorgestellte Software-Entwicklungsmethodik des *Extreme Programming (XP)* implementiert augenscheinlich das momentan praktikabelste Verfahren zur Integration potentieller Nutzer in das eigentliche Entwicklungsgeschehen. Speziell unter dem Gesichtspunkt der Partizipation von Nutzern sollen Teile der Methode hier genauer beleuchtet und in die Thematik der Kooperativen transdisziplinären Entwicklung eingeordnet werden. Die Grundlagen, die XP zur Ausprägung des co-adaptiven Charakters der in Abschnitt 3.2 vorgestellten Vorgehensweise beiträgt, werden im Folgenden konkretisiert.

Das Vorgehen bezeichnet neben dem *Programmieren*, *Testen* und *Designentwerfen* das Zuhören als einen der grundlegenden Arbeitschritte (vgl. [Bec00] Seiten 43 ff.). Die Erkenntnis, dass die herkömmlichen Methoden der Anforderungserhebung und dabei insbesondere lange unstrukturierte Besprechungen zwischen Entwicklern und Domänensachverständigen oft zu divergenter unfokussierter Kommunikation führen lässt die Forderung nach „aktivem Zuhören“ [Bec00] aufkommen.

Dabei sollen Domänensachverständige bzw. potentielle Kunden oder Nutzer auf der einen Seite und die Entwickler der Technologie auf der anderen Seite in einem paritätischen, strukturiert interaktiven Kommunikationsprozess die zu lösenden Geschäftsprobleme gemeinsam erarbeiten. Der Domänenexperte formuliert dazu seine Anliegen aus

seiner Sicht. Der Technologieentwickler bewertet die Ansätze und Ideen sofort hinsichtlich der technischen Implikationen – der Machbarkeit, des Aufwandes der Umsetzung, etc. – und gibt entsprechend qualifiziertes Feedback. Vorgaben, die ein solches Vorgehen praktikabel machen sollen, bestimmen, dass die Kommunikation nur genau dann stattfindet, wenn sie tatsächlich benötigt wird und dass Unklarheiten möglichst vollständig beseitigt werden. Es soll ferner verhindert werden, dass unnötige weitschweifige Diskussionen geführt werden.

Diese Festlegungen lassen sich wiederum verdichten zu einer der Kernideen von XP, die besagt, dass durch sehr kurze Zyklen der Entwicklung, also auch der Planung und der Konzeption, immer Zwischenergebnisse und Artefakte entstehen sollen, die bei nötig werdenden Änderungen kein Risiko induzieren. Insbesondere die „Trägheit“ umfangreicher monolithischer Spezifikationen im Bezug auf Veränderungen als Reaktion auf geänderte Anforderungen bzw. angepasste Sichtweisen und damit neue Interpretationen des Spezifizierten soll durch iterierte kleine und unmittelbare Feedback-Zyklen und möglichst direkte Umsetzung aufgehoben werden.

Die enge, strukturierte und vor allem aktive Einbindung von Domänenexperten in den diskursiven Entwicklungsprozess und damit der fortwährende Dialog zwischen Technologie und Domäne in der oben charakterisierten Form, ist im Wesentlichen durch vier so genannte Verfahren manifestiert:

„Planungsspiel“: Die Kommunikation und insbesondere bei der Vermittlung zwischen dem technisch Möglichen und dem fachlich wünschenswerten bedarf einiger Regeln, die eine Atmosphäre gegenseitigen Respekts, Interesses sowie Vertrauens sicherstellen. Unabhängig vom Detail der Umsetzung spielerischer Regeln bei XP, kann es über die dort beschriebenen drei Phasen (der Erforschung, der Verpflichtung, und der Steuerung) gelingen, den konstruktiven Diskurs von interdisziplinären Ressentiments und entsprechenden Barrieren zu lösen. Dabei ist es wichtig diesen bewusst geführten Dialog über geeignete Moderation und vor allem die Iteration zum Teil des Selbstverständnisses innerhalb des transdisziplinären Entwicklungskontexts zu machen.

„Metapher“: Ein wichtige Voraussetzung für eine echte Zusammenarbeit ist eine Gemeinsame Sprache oder zumindest ein gemeinsames Verständnis von den Ergebnissen, aber auch den Aufgaben, die zu erledigen sind. Dazu ist es sinnvoll zumindest in der disziplinübergreifenden Kommunikation von jeweils disziplinspezifischen Aspekten zu abstrahieren. So ist es beispielsweise sicher einfacher ein gemeinsames Verständnis für eine *Themenvorlage* zu entwickeln als für eine *XML-Schema*, was ein technischer Aspekt der Implementierung der Themenvorlage sein kann.

„Kunde vor Ort“: Die für die Arbeitsorganisatorische Praxis eher utopisch anmutende Forderung nach physischer Präsenz von Kunden bzw. Endnutzern im Kreise der Entwickler wird in [Bec00] (Seiten 68 f.) dann auch dahingehend relativiert, dass es für Entwickler grundsätzlich wichtig ist, einen Vertreter der Anwenderseite auch in kleinem Rahmen für Entscheidungen über Prioritäten und Umfang

bei der Umsetzung einzelner Funktionalitäten zur Verfügung zu haben. Direkte Kommunikationswege auf der organisatorisch gleichen Ebene, und die Möglichkeit Ergebnisse ad hoc gemeinsam zu analysieren und überarbeiten zu können auch mit technischen Mitteln unterstützt werden.

„Test“: Ein echtes Mitwirken aller Disziplinen wird nur dann möglich sein, wenn die Ergebnisse der Arbeit für alle jeder Zeit transparent sind. Die von XP realisierten automatisierten Tests bereiten zumindest in Bezug auf die Nutzbarkeit interaktiver Systeme Probleme. So kann das Verfahren zumindest in einer Interpretation Anwendung finden, nach der ein ständig verfügbares, ablauffähiges Artefakt kontinuierlich evaluierend in einem Kontext genutzt wird, der dem der Anwendung entspricht.

Diese Verfahren bilden das Fundament von Grundwerten, die in adaptierter und zum Teil verschärfter Form den diskursiven Prozess innerhalb des in dieser Arbeit vorgestellten Entwicklungsvorgehens, unter direkter Beteiligung auch der Dämonenexperten und Nutzer leiten sollen.

JAD

Als eine der ersten Bemühungen, Nutzer systematisch in den Entwicklungszyklus oder zumindest in die Entwurfsphasen von Software-Systemen einzubeziehen, kann das so genannte Joint Application Development (abgekürzt JAD) betrachtet werden. Die Wurzeln der Idee von JAD reichen zurück bis 1977, als bei der IBM erstmals gezielt strukturierte Meetings unter Einbeziehung der Nutzer zur Ausgestaltung von Software-Systemen durchgeführt wurden.

Ausgehend von der Erkenntnis, dass der traditionelle Prozess des Entwurfs von IT-Systemen wegen träger Kommunikation und langer Feedback-Zyklen oft sehr zeitaufwändig ist und umso schwieriger zu handhaben wird, je mehr Stakeholder in die Entwicklung involviert werden, systematisieren [WS95] eine Methodik, die den Defiziten klassischer Vorgehensweisen an den organisatorischen, interdisziplinären Schnittstellen Rechnung trägt. Im Zentrum der Vorgehensweise steht ein Workshop (die JAD-Sitzung), während dessen alle wichtigen Entscheidungen bez. der gemeinsam gesteckten Projektziele im Konsens zwischen allen Betroffenen getroffen werden.

Das Revolutionäre dieses Ansatzes besteht dabei darin, dass Nutzer in einer aktiven konstruktiven Rolle am Entwicklungsprozess beteiligt werden – sie vom analysierten Objekt zum handelnden Subjekt innerhalb des Projektes werden. Das tragende Konzept um dies zu erreichen manifestiert sich in zwei der „Zehn JAD-Gebote“: „Keep technical jargon to a minimum.“ sowie „All participants are equal“ [WS95]. Das heißt, dass durch eine möglichst große Nähe der während der Systemkonzeption verwendeten Sprache und der Metaphorik von Modellbildungen zur Anwendungsdomäne, bestehende Barrieren und Berührungsängste abgebaut werden sollen. Auch organisatorische Brüche zwischen Technikern und Fachleuten aber auch zwischen Entscheidungsträgern und Ausführenden

müssen im Sinne der konstruktiven Arbeit am gemeinsamen Artefakt systematisch durch die Ausbildung einer entsprechenden Kultur ausgeräumt werden.

Die drei zentralen Anliegen des Ansatzes bestehen laut [WS95] in:

- Der Etablierung von Verbindlichkeit und Verantwortungsbewusstsein (*Commitment*) unter den beteiligten Nutzern. Durch eine frühe und direkte Einbindung in den Entwicklungsprozess soll das Interesse am angestrebten Produkt und am Erfolg des Projektes manifestiert werden. Die Autoren gehen davon aus, dass dies eine wesentliche Voraussetzung für die intrinsische Motivation und ein entsprechendes Engagement aller Beteiligten ist. Dies soll es ermöglichen innerhalb weniger Tage Konsens bez. aller Entwurfsentscheidungen zu erreichen, wobei die Verantwortung für das erhaltene Produkt unter allen Beteiligten gleichberechtigt geteilt wird.
- Die Schaffung enger sozialer Bindungen durch die forcierte persönliche Nähe innerhalb der JAD-Sitzungen soll dazu führen, dass im Arbeitsalltag verfestigte Positionen und Handlungsweisen für neue Perspektiven geöffnet werden. Dass fachliche, organisatorische und kulturelle Barrieren, die sich etwa aus der Zugehörigkeit zu unterschiedlichen Disziplinen ergeben, durch die Konstituierung einer eingeschworenen Gruppe (vgl. [WS95], Seite 7) leichter zu überbrücken sind, ist nahe liegend.
- Mit der Idee, Besprechungen grundsätzlich in Form eines stark strukturierten und eng geführten Workshops durchzuführen, adressiert der JAD-Ansatz a priori das Problem unfokussierter, unproduktiver zeitraubender Projektbesprechungen. Auf diese Weise werden die Reibungsverluste, die sich aus unzureichend gesicherten gemeinsamen Arbeitsergebnissen, nicht systematisch ausgeräumten Missverständnissen ergeben, als Quellen des Scheiterns transdisziplinärer Projekte ausgeräumt.

Der in [WS95] zusammengefasste Leitfaden zur Durchführung von JAD-Projekten strukturiert das Vorgehen zur Erreichung dieser Ziele in fünf Phasen und macht jeweils sehr feste Vorgaben zu deren inhaltlicher und formaler Ausgestaltung:

Phase 1 – JAD Projektdefinition: In Interviews mit den Verantwortlichen der Fachabteilung (der Anwender) werden der der grundsätzliche Zweck und die Ziele des Projekts erhoben. Grundanliegen, erste Anforderungen, grundlegende Voraussetzungen, Annahmen und Rahmenbedingungen (beispielsweise zu den benötigten Ressourcen, oder etwa administrativen bzw. juristischen Notwendigkeiten) werden fixiert.

Diese Phase erfüllt damit im Wesentlichen die Aufgaben der Anforderungsermittlung, wie sie etwa auch in [Sch02] (Seiten 34 ff.) beschrieben sind. Die Aktivitäten zur Identifizierung von betroffenen und beteiligten Parteien (Stakeholders) ist hierbei insofern spezifisch, dass bereits eine Auswahl an Personen getroffen werden muss, die relevante Beiträge im Rahmen der JAD-Sitzung leisten können und müssen. Ferner werden Planung der Sitzung wir vorgenommen. Schließlich werden die Ergebnisse dieser ersten Analyse in einem so genannten *Management Definition Guide* zusammengefasst.

Phase 2 – Analyse: Die im englischen Original des Modells nach [WS95] als *Research* bezeichnete Phase dient der Konkretisierung der erkannten Anforderungen aus fachlicher Sicht. Analog zur eigentlichen Anforderungsanalyse (diese Differenzierung zur Anforderungsermittlung macht wiederum [Sch02]) klassischer Vorgehensweisen werden hier im Detail Geschäftsprozesse analysiert und entsprechend das Potential für Optimierungen durch die Einführung von Informationstechnologie antizipiert. Die Struktur der zu verwaltenden Daten wird auf konzeptioneller Ebene definiert sowie erste Vorstellungen für Mensch-Maschine-Schnittstelle skizziert.

Das Ergebnis der Analyse ist eine Liste mit Anforderungen und Problempunkten, die der Erörterung und Klärung im Rahmen der eigentlichen JAD-Sitzung bedürfen. Vorgaben für den Inhalt und die Strukturierung der Dokumentation orientieren sich dabei an gängigen Paradigmen. So schlägt JAD beispielsweise vor, dass Entity-Relationship-Diagramme zur Modellierung der Anforderungen bez. der Strukturierung der Daten genutzt werden. Die Ergebnisse bestimmen schließlich die fachliche Ausgestaltung der Agenda für die Sitzung.

Phase 3 – Vorbereitung: Dass sich diese Phase der Vorbereitung dezidiert, den technischen und methodischen Aspekten der Durchführung der bevorstehenden Sitzung widmet, mach deutlich, welche Bedeutung die Autoren des JAD einer klaren Strukturierung des Treffens beimessen. Nur auf der Basis

- Einer sauberen Arbeitsgrundlage in Form des so genannten *Working Document* – einer Zusammenstellung aller diskutablen bisher erarbeiteten Ergebnisse,
- Eines gut vorbereiteten Teilnehmerkreises, einschließlich eines geschulten Protokollanten, sowie
- Einer inhalteangepassten „Inszenierung“ der Session durch die Wahl adäquater Präsentations-, Kreativ-, und Moderationstechniken.

kann die Sitzung zielführend abgehalten werden. Insbesondere die Vorbereitung der Beteiligten durch die frühzeitige Bereitstellung des Arbeitsdokuments und die in Vorbesprechungen *Pre-Session Meeting* erläuterten Zielstellungen und Zeitplanungen sind Grundvoraussetzung, um ausufernde Diskussionen zu vermeiden. Interessant ist, dass sogar der Einrichtung, also die Bestuhlung und die technische Ausstattung des Besprechungsraumes durch [WS95] Aufmerksamkeit geschenkt wird. Daran ist abzulesen, dass den Nuancen der Gestaltung des sozialen Entwicklungskontexts auch in der Erfahrung der Entwickler von JAD eine große Bedeutung zukommt.

Phase 4 – Die Sitzung: In Analogie zu den in [Sch02] (Seiten 41 f.) beschriebenen „herkömmlichen“ Aktivitäten zur Entscheidungsfindung zur tatsächlichen Umsetzung und einer Einigung über die Anforderungen werden in einer oder mehreren inhaltlich jeweils fokussierenden Sitzungen gebündelt Verständnisdifferenzen und Interessenskonflikte zwischen Stakeholdern ausgeräumt. Insbesondere durch den Einsatz von Kreativtechniken und die Veranschaulichung durch interaktive

Präsentationsformen (Flipcharts, Magnettafel, etc.) wird es möglich, allen Beteiligten auf fachlicher Ebene, losgelöst von technischen Abstraktionen die in der Analyse erarbeiteten Modelle zu diskutieren, konkretisieren und zu korrigieren. Die Präsentation erster Anmutungen möglicher Nutzerschnittestellen entspricht dabei schon ansatzweise dem, was [EK91] versuchen mit dem Einsatz einfachster Oberflächenprototypen zu erreichen (Siehe dazu auch den Abschnitt 2.3.3). Als wichtigstes Mittel der Fixierung von Ergebnissen wird dabei das Protokoll angesehen. Die darin festgehaltenen Entscheidungen und die offen bleibenden Diskussionspunkte münden bilden zusammen mit den in der Sitzung verfeinerten Anforderungsmodellen und dem konkretisierten Arbeitspapier die Grundlage für die abschließende Phase der zusammenfassenden Dokumentation.

Phase 5 – Abschlussdokumentation: Die Dokumentation zum Schluss des Prozesses fasst alle Ergebnisse der des JAD-Prozesses und in insbesondere der Konsensfindung zusammen. Die in der Gruppe verabschiedeten Beschlüsse zu Grundannahmen, zu Voraussetzungen, sowie zur Datenstrukturierung, zu ersten groben Screen-Entwürfen und zu Prozessmodellen erhalten dadurch einen verbindlichen Charakter, dass alle am Prozess beteiligten Entscheidungsträger ein so genanntes *JAD Approval Form*. Die tatsächliche Verbindlichkeit eines solchen Dokuments hängt dabei sehr stark von seiner Verankerung und der Bedeutung innerhalb der Organisation ab. Das rein psychologische Moment der der Abgabe der Verpflichtungserklärung wird ohne die Einbettung in eine Systematik der Verwaltung von Änderungen (Change Management) und ohne disziplinarische Verbindlichkeit wirkungslos bleiben.

Theoretisch lässt sich dieses Vorgehen auf jede Art von Entwicklungsbemühung unter Einbeziehung unterschiedlicher entscheidungstragender Institutionen und Disziplinen übertragen. Im Bereich der Software-Entwicklung ist die Erhebung und Strukturierung von Nutzeranforderungen an ein zu entwickelndes System das prädestinierte Einsatzszenario für JAD. Dabei ist der Anspruch der Autoren, die Ergebnisse in einem singulären Zyklus und innerhalb einer möglichst kurzen Sitzung zu generieren zumindest diskutabel. Insofern wird der in dieser Arbeit vorgestellte Ansatz methodisch zwar Anleihen bei JAD machen um das transdisziplinäre Entwicklerteam zu konstituieren und den diskursiven und sehr dynamischen Prozess der Entwicklung unter intensiver Nutzung der Emergenz kreativer Lösungen vorzustrukturieren und steuerbar zu halten.

Insbesondere die Einbindung der JAD-Ansätze in einen iterativen und evolvierenden Gesamtprozess und damit die Ausweitung der Idee auf alle Phasen der Entwicklung bedarf dazu jedoch einiger konzeptioneller Anpassungen bzw. neuer Interpretationen einzelner Schritte. Die Singularität und der monolithische Charakter des Prinzips, wie es in [WS95] beschrieben ist beispielsweise, muss aufgebrochen werden zugunsten der Möglichkeit, Aufgaben zu dekomponieren. Anstatt eine Vorlage für eine geschlossene obligatorische Abschlussdokumentation vorzugeben, werden vielmehr Muster für Zwischenergebnisse festzulegen sein, die Grundlage für eine systematische kontinuierliche Weiterentwicklung sein können. Eine klare Planung und Strukturierung der Moderation

und Steuerung der Kollaboration zwischen den Disziplinen bzw. eine adaptierbare Systematik zu entsprechenden Festlegungen muss Teil des Konzepts des framework-basierten interaktiven Prototyping werden.

Wikis

[LC01] Wissen zu externalisieren und internalisieren – gleichermaßen sowohl als Sender, als auch als Empfänger zu fungieren ist Nutzern so genannter Wikis [LC01] möglich. Sie kreieren damit auf sehr einfache Weise System von Hypertextseiten – letztlich einer simplen interaktiven Anwendung, ohne tatsächlich vertiefte Kenntnisse von den technischen Hintergründen des Gesamtsystems haben zu müssen. Dabei füllen Autoren und Redakteure einen bezüglich des Layouts und der Gestaltung weitgehend vorgegebenen Rahmen mit Inhalten und einfachen interaktiven Funktionen (nach dem Hypermedia-Prinzip). Der Rahmen implementiert (u.a. unter Nutzung der Möglichkeiten des HTTP-Protokolls und der Funktionen des Webbrowsers) ein zentrales Softwareartefakt mit allen Infrastrukturellen Aspekten – wie den Zugriff über das Netzt, und die Verarbeitung von Nutzereingaben etc. Das Ergänzen um Inhalte und Funktionen entspricht also der evolutionären gemeinschaftlichen Entwicklung eines Informationssystems.

Projekte wie Wikipedia[5] machen deutlich, welche Potentiale ein gemeinschaftlicher diskursiver Gestaltungsprozess birgt.

„Die skandinavische Schule“

Partizipative bzw. kooperative Design-Ansätze (zusammengefasst unter *participatory Design* [SNH93] und *cooperative Design* [GKH91]) versuchen potentielle Nutzer von Systemen, über die Rolle des passiven analysierten Anforderungsbeitragenden hinaus, sehr viel stärker in den Prozess der Software-Entwicklung einzubeziehen als beim *User-centered design*. In der Funktion von *co-designers* [FG04] sollen Domänensachverständigen ohne Bezug zur Technologieentwicklung Gestaltungsspielräume eröffnet werden, die es ihnen erlauben, selbst zu entscheiden, inwieweit sie ihre Arbeitsprozesse durch den Einsatz von Informationstechnologie effektivieren können (vgl. [BGK93]). Dies soll erreicht werden, indem der Design-Prozess – hier der Prozess der inhaltlich funktionalen Gestaltung der Anwendung – ganzheitlich betrachtet wird und nicht als bloße formale deterministische Umsetzung von Nutzeranforderungen in Systemanforderungen und schließlich das System. Bei der Entwicklung der partizipativen Methoden werden Aspekte berücksichtigt wie:

- Die organisatorische und politische Einbettung der zu entwickelnden Software und der Prozesse ihrer Entwicklung in die betrieblichen Kontexte. So werden etwa Konfliktpotentiale, die sich aus unterschiedlichen Machtpositionen heraus ergeben – etwa zwischen Management und Arbeitern – versucht zu kontrollieren.

[5] `http://de.wikipedia.org/`

- Die *Use situation* [BGK93] – die realen Arbeitsumstände für den potentiell tagtäglichen Einsatz des Systems.
- Die Kompetenzen der Beteiligten – insbesondere auch das implizite Wissen (*tacit knowledge*) und gemeinschaftliche Wissen von Gruppen. Entsprechend angepasst wird die Wahl der Instrumente und Vorgehensweisen während des Designs. Traditionelle Modellbildungen wie Ablauf- und Datenflussdiagramme oder gar gängige Programmiersprachen sowie entsprechende Werkzeuge hält [BGK93] für nicht tragfähig während der Design-Arbeit mit Nutzern.

Zusammengefasst bedeutet das, dass Arbeit, Technologie und die verbindenden soziotechnologischen Strukturen bewusst so gestaltet werden müssen, dass sie optimal vereinbar sind mit *menschlichen* Bedürfnissen. Die Entwicklung und die Einführung von Software-Systemen werden zu einem integralen Bestandteil der permanenten Ausgestaltung und Weiterentwicklung der unterstützten Arbeitsprozesse und sind damit auch federführend von denselben Akteuren zu gestalten.

Um die erforderlichen Gestaltungsprozesse in Gang zu setzen und im Sinne der gesteckten Ziele konstruktiv zu steuern werden bei [BGK93] abhängig vom konkreten Problemkontext verschiedene Techniken kombiniert. Der Grundgedanke ist dabei, eine systematische Annäherung der Entwicklung der Technik und der Domänenexperten herbeizuführen und einen wechselseitigen Lernprozess anzustoßen. Den Auftakt bildet in der Regel die Einarbeitung der Techniker in die Probleme des adressierten Arbeitsprozesses.

In Interviews und Demonstrationen der Abläufe der alltäglichen Arbeit versuchen die Techniker die Gesamtzusammenhänge zu verstehen und eine erste Sammlung potentieller Problem- und damit Optimierungspunkte zusammenzustellen.

Spätestens in der zweiten Phase werden die Nutzer zu den zentralen Akteuren des Umgestaltungsprozesses, der durch die Systementwickler lediglich noch moderiert wird. In so genannten *Future Workshops* [KHM91] werden dabei aus aktuell bestehenden Problemen Verbesserungspotentiale für den Arbeitsprozess abgeleitet. Entsprechend gliedert sich der Workshop in zwei Phasen: während der *critique phase* werden strukturierte Brainstormings durchgeführt. Dabei wird beispielsweise jedem der Beteiligten der Anwendungsdomäne in jeweils dreißigsekündiger Redezeit die Möglichkeit gegeben, aktuelle Probleme innerhalb der Arbeitsabläufe zusammenzutragen. Den Impuls dafür geben die von den Systementwickler während ihrer Interviews gemachten Beobachtungen. Die Kritikpunkte werden anschließend, wie in Brainstorming-Sessions üblich, gruppiert. In der im Anschluss folgenden *Fantasy Phase* werden die Kritikpunkte so umformuliert, dass sie als konstruktive Zielstellung fungieren können. Übertragen auf den Kontext der in Kapitel 3.2.2 entwickelten Konzepte zum Vorgehen, wird das beispielsweise bedeuten, dass Autoren, die starre Abstraktionen von Contentmanagement-Systemen nicht tolerieren können, formulieren müssen, mit welchen Mitteln bzw. unter welchen Umständen sie bereit sind, Ihre Inhalte direkt am System zu erfassen – etwa durch die Verwendung von Bedienmetaphern, die Ihnen vertraut sind.

Die Übertragung der in den *Futurworkshops* generierten Ideen in tatsächliche Systementwürfe, die als Grundlage für die Umsetzungsphase dienen können, würde in „klassischen“ Vorgehensmodellen beispielsweise darin bestehen, so genannte Businessobjekte zu identifizieren, diese zu klassen zu verallgemeinern, deren Interaktionen in Ablauf- und Kollaborationsdiagrammen zu modellieren. Unter der Prämisse, dass Nutzer bzw. Domänensachverständige auch in diese Schritte des Systementwurfs einbezogen werden sollen lässt sich beobachten, dass diese Form der Formalisierung, ungeeignet ist als Abstraktionsmittel während der transdisziplinären kooperativen Systementwicklung. Entstehende (formale) Modelle werden von Beteiligten, die der Informatik fern stehen, schlicht nicht verstanden bzw. akzeptiert. Das bestätigt die Erfahrung aus der Projektarbeit am Fraunhofer ISST und ist auch Ausgangspunkt der Betrachtungen, der in diesem Abschnitt vorgestellten Methodik [ES91]). Um den Nutzern die Transferleistung von einfachen Ideen hin zu Systementwürfen möglich zu machen, benötigt man vielmehr (vgl. auch [ES91]):

- Einfache Mechanismen die alle Beteiligten leicht beherrschen können und die in kooperativer Arbeit in der Gruppe anwendbar sind.
- Kostengünstige flexible infrastrukturelle Unterstützung – als kaum praktikabel erweisen sich umfangreiche integrierte CASE[6]-Werkzeuge.
- Die der Methodik zu Grunde liegenden Konzepte – Metaphern, Begrifflichkeiten und Sprachen – sollten individuell abgestimmt sein auf den aktuellen Anwendungskontext und die damit verbundene Entwurfsaufgabe.
- Die Beteiligung soll Spaß machen und dennoch zu ernsthafter Arbeit motivieren.
- Die Ergebnisse des Prozesses sollen tauglich sein um als Grundlage für die Umsetzung fungieren zu können und schließlich tatsächlich und sichtbar in die Umsetzung fließen, um die Sinnhaftigkeit der Aktivitäten zu manifestieren und den Anwendern transparent zu machen.

In so genannten *organizational games* [ES91] erhalten Nutzer und Domänenexperten die Möglichkeit die angestrebten Veränderungen ihres Arbeitsumfeldes zu simulieren, aus unterschiedlichen Perspektiven zu betrachten und zu hinterfragen sowie aktiv zu gestalten. Für jedes Entwicklungsvorhaben wird dazu ein Spiel entwickelt, welches die Sprache und die Metaphorik der jeweiligen Anwendungsdomäne zum Kern hat. Mit einfachen Mitteln werden Beispielsweise Spiel-Karten entwickelt, die etwa Artefakte, Materialien oder Funktionen der zu simulierenden Abläufe repräsentieren. Diese Karten sind dann Gegenstand der spielerischen Auseinandersetzung mit bestehenden Arbeitsprozessen und schließlich auch mit den veränderten Prozessen der Zukunft. Der Spielablauf gestaltet sich ebenfalls abhängig vom Einsatzkontext. Das in [ES91] vorgestellte Spektrum reicht dabei vom freien Strukturieren und Inbeziehungsetzen solcher Karten und einer entsprechend moderierten Diskussion über gesteuerte Spiele, die nachzuvollziehende Schlüsselsituationen auf Ereignis und Aktionskarten vorgeben bis hin zur Inszenierung kleiner szenischer Spiele mit klar definierten Rollen und Rahmenvorgaben

[6] **C**omputer **A**ided **S**oftware **E**ngineering

für Nachzustellende Situationen. Die Art und Weise, wie Anwender etwa in kritischen Situationen reagieren, aber auch wie sie alltägliche Verrichtungen ausüben, werden auf diese Weise transparent und von den Beteiligten reflektiert. Das bezieht sich insbesondere auch auf Abläufe, die zukünftig von dem zu entwerfenden System zu unterstützen sind.

Anforderungen an das System bzw. Vorschläge für die Umsetzung ergeben sich dabei auf zwei Arten:

Zum einen leiten die Systementwickler erste Vorschläge für Systemfunktionalitäten aus den Ergebnissen der Phase ihrer Einarbeitung bzw. der Future-Workshops ab und bringen diese in Form einfacher Prototypen und Anschauungsbilder in die Spiele eine. Sinnvollerweise werden beispielsweise Interaktionsmuster und Oberflächenmetaphern bei der Ausgestaltung des Spiels (beispielsweise der Karten und des Spielfeldes) genutzt. Auf dieser Grundlage können Anwender sich mit den prinzipiellen Möglichkeiten und auch Grenzen des Einsatzes der neuen Technologie im Rahmen ihrer Arbeit auseinandersetzen – zunächst unverbindlich, spielerisch und damit ohne Risiken, die Berührungsängste auslösen müssten. Die von den Systementwicklern gemachten Vorschläge werden so evaluiert und ggf. selektiert und überarbeitet.

Zum anderen können die Anwender durch ihr agieren, etwa durch die Reihenfolge in der sie Aktivitäten setzen und metaphorische Hilfsmittel nutzen (das entspricht dem Ausspielen einer entsprechenden Karte) das gestalten, was schließlich als Ablaufsteuerung bzw. Interaktionsmuster in der Software materialisiert wird.

Zur Fixierung der Ergebnisse im Anschluss an die Phase der *organizational Games*, schlägt [BGK93] die kooperative Entwicklung an Prototypen und Mock-Ups[7] vor. Mock-Ups bieten dem zukünftigen Nutzer eine sehr einfache Möglichkeit, sich eine konkretere Vorstellung von der Handhabung (hands-on experience) und der Optik des zu entwickelnden Systems zu machen – weit über die von den klassischen Systembeschreibung vermittelten Abstraktionen hinaus. Selbst Folgen von Bildern der Nutzerschnittestelle oder einfache interaktive Animationen ermöglichen einen allen verständlichen Einblick in die zu erstellenden Systemfunktionalitäten und der technischen Möglichkeiten des Einsatzes von Informationssystemen.

In Experimenten [EK91] konnte dabei nachgewiesen werden, dass es selbst mit den einfachsten Mitteln möglich wird, designierte Nutzer dazu zu bewegen die zukünftige Tätigkeit tatsächlich (wenn auch simuliert) auszuüben und sich in die neuen Arbeitsprozesse hineinzufühlen. Man ist also mit dem Einsatz einer Metaphorik, die sich an bekannten bzw. traditionellen Werkzeugen und Abläufen orientiert, in der Lage das Abstraktionsvermögen von Nutzern derart zu instrumentalisieren, dass sie sich auf eine möglichst realitätsnah nachgestellte Arbeit an ihren eigentlichen Aufgaben konzentrieren können, ohne sich dabei durch die Reflektion über neue mögliche, zukünftige Funktionalitäten eines Software-Systems überfordern lassen zu müssen. Dabei bleibt für alle Beteiligten transparent, dass es sich lediglich um eine einfache kostengünstige Simulationen handelt.

[7] Oberflächenprototypen und Grafiken zu Veranschaulichung der Anmutung.

Jeder besitzt die Kompetenz, Änderungen vorzunehmen, ohne Investitionen am eigentlichen Zielsystem zu gefährden. Dieses Einweg-Prototyping erhebt nicht den Anspruch Artefakte teilweise oder als ganze Module für die Entwicklung des Zielsystems beizutragen. Kriterien der Qualität und Korrektheit stehen klar hinter der partizipativen fachinhaltlichen Konzeption zurück. Problematisch ist der Aufwand, der dabei betrieben werden muss: kleine Änderungen, etwa an einer Menü-Struktur, führen unter Umständen dazu, dass alle involvierten Screenanmutungen neu gezeichnet werden müssen. Eine weitere Schwierigkeit besteht darin, den kreativen Prozess der Diskussion und Weiterentwicklungen der Mock-Ups innerhalb der Grenzen des technisch Machbaren zu steuern und zu moderieren.

Während des *cooperative prototyping* geht man über die Vermittlung erster Anmutungen hinaus. Lauffähige Prototypen werden in realitätsnahen ggf. auch simulierten zukünftigen Arbeitssituationen eingesetzt und in der Nutzung durch die potentiellen Anwender evaluiert. Auftretende Probleme, die auf fehlende bzw. falsch umgesetzte Funktionalitäten am Prototypen zurückzuführen sind, werden dabei zeitnah und möglichst noch direkt im Rahmen der Prototyping-Session von den Technikern behoben. Das erlaubt eine dem Nutzer nachvollziehbare Evolution des Prototypen innerhalb des Arbeitsprozesses, der letztlich die Kriterien für die Verbesserungen innerhalb der Session „live“ vorgibt.

Dass die Nutzer mit den Werkzeugen der Weiterentwicklung selbst so vertraut sind, dass sie die Änderungen selbst vornehmen können – in der Art der Arbeit an den Mock-Ups – hält [BGK93] noch für nicht machbar. Darüber hinaus gehen [BGK93] den Schritt zur Software-Produktion bzw. -Pflege und -Änderung unter Einbeziehung der Nutzer im Rahmen der Alltäglichen Nutzung nicht. Diese beiden Punkte bergen jedoch erhebliche Gestaltungspotentiale, die die hier vorliegende Arbeit versuchen wird auszuschöpfen.

Für die vorliegende Arbeit relevante Kernaspekte der Gesamtmethodik nach [BGK93] sind:

- Die Nutzung strukturierter Kreativtechniken zur systematischen Annäherung zwischen den beteiligten Disziplinen und insbesondere zwischen originären technisch orientierten Systementwicklern und den Fachleuten der Problemdomäne.
- Die Einbindung der Prototypen und Mock-Ups der zu entwickelnden Software-Systeme in die Anforderungserhebung (hier etwa in die jeweilige Spielsituation) zur Veranschaulichung der technologischen Aspekte der arbeitsorganisatorischen Umgestaltung bzw. zunächst zur Verdeutlichung der Gestaltungspotentiale, die sich aus dem Einsatz von Software ergeben.
- Die Nutzung interaktiver Prototypen zur kollaborativen, anschaulichen sowie explorativen Fixierung der Entwurfsentscheidungen. Von besonderem Interesse ist der Nachweis der Anwendbarkeit und der Effektivität der Mechanismen zur Einbeziehung von Nutzern bei der Erstellung und Überarbeitung von Prototypen.

Die Ansätze der „skandinavische Schule“ zur Nutzung von Prototypen in interaktiven Designprozessen unter Einbeziehung von Anwendern werden in einem separaten

Abschnitt (2.3.5) konkretisiert und in den Kontext der prototypenbasierten Software-Entwicklung im Allgemeinen gesetzt. Die Übertragung und Weiterentwicklung der tragenden Prinzipien auf das Problemfeld der interaktiven, direkt reaktiven Systeme unterstützt durch eine allgemeine Prototyping-Infrastruktur erfolgt in Abschnitt 3.2.

Ethnographische Feldforschung

Die Durchführung von *Future-Workshops* und *Organizational Games* zur partizipativen Erforschung und Umgestaltung von Arbeitsprozessen (wie oben beschrieben), ist aufwändig und erfordert sehr viel Erfahrung oder gar Spezialisierung bez. der Moderation kreativer Gruppenprozesse und bei der Anwendung von Kreativtechniken. Ein offenbar für viele Projektkonstellationen eher praktikables Vorgehen für die systematische Konvergenz der Arbeits- und Verständniswelten der Entwickler und der Domänensachverständigen bzw. Nutzer verspricht die Anwendung von Methoden der ethnographischen Feldforschung. Die in der Projektarbeit am Fraunhofer ISST gemachten Erfahrungen auf der Grundlage zunächst eher intuitiv ausgestalteter Erhebungsprozesse (siehe dazu Abschnitt 4.2.2) und die daraus abgeleiteten allgemeinen Lösungsansätze knüpfen dabei im Wesentlichen an die Betrachtungen in [BGMSW93] an.

Grundsätzliches Ziel der Methodik ist es, die Arbeitsweise der Akteure der Zieldomäne tatsächlich zu durchdringen und gesamtheitlich zu verstehen. Mit diesem Anliegen geht man weit über die üblichen Praktiken der Anforderungserhebung hinaus, die in der Regel lediglich die konkreten Eigenschaften eines partikulären Zielsystems untersuchen. Ausgehend von einer oft im voraus festgesetzten Zielstellung werden in der „herkömmlichen“ Anforderungsanalyse präskriptiv zukünftige optimierte Geschäftsprozesse antizipiert, ohne die aktuellen Abläufe oder gar einzelnen Aktivitäten sowie die daraus resultierenden tatsächlichen Notwendigkeiten der Verbesserung und Weiterentwicklung schrittweise zu entwickeln. Mit Hilfe ethnographischer Methoden versuchen Systementwickler die Sichtweisen, Verständniswelten, Begriffsbildungen und vor allem die Arbeitsabläufe innerhalb der Zieldomäne gleichsam von „innen heraus“ zu betrachten und zu verstehen.

Ethnographische Feldforschung ist ein gängiges Instrument der Sozialforschung zur Untersuchung menschlicher alltäglicher Verhaltensweisen in Abhängigkeit vom konkreten sozialen Kontext. Insbesondere die kleinen Dinge des Alltagslebens (z.B. Grüßen, Fragen, Verabreden) werden bei diesen qualitativen Analysen aus der teilnehmenden Betrachtung heraus untersucht und interpretiert (siehe dazu auch Stichwort „Ethnomethodologie“ [BD02], Seite 304). Das heißt, der Untersuchende wird zum Teil der beobachteten Gruppe – er nimmt Teil am sozialen Leben der Gemeinschaft. Er ist so einerseits in der Lage, die Dynamiken und Strukturen der Gruppe „aus großer Nähe“ und in entsprechendem Detail sowie über einen langen Zeitraum in möglichst vielen Facetten zu beobachten. Andererseits ist es dem Analytiker möglich, durch die Beobachtung der Reaktionen auf seine Aktivitäten und durch das Sammeln eigener Erfahrungen, also durch den eigenen Lernprozess, Einblicke in verborgene Zusammenhänge innerhalb der Domäne zu erlangen – in Fakten, die den Domänenprotagonisten zum Teil unbewusst

sind. Gerade in Bereichen, bei denen Routine, eine große Roll spielt, sind den beteiligten einzelne Abläufe und insbesondere die dazu notwendigen Kompetenzen nicht mehr bewusst. Das führt dazu, dass auch Defizite und daraus resultierend Optimierungspotentiale nicht mehr erkannt werden.

Die Herausforderung besteht nun laut [BGMSW93] darin, die so gewonnen Erkenntnisse bez. der Gemeinschaft – des kooperierenden Kollektivs der Akteure im adressierten Arbeitsprozess – in die Abläufe der Ausgestaltung von unterstützenden Systemen für eben diese Arbeitsprozesse zu tragen. Es geht also darum, auch die elementarsten, nicht offensichtlichen Aktivitäten zu untersuchen, die den Arbeitsprozess ausmachen – analog zu den einfachen Gesten (wie dem angesprochenen Grüßen), mit denen man dem Alltagshandeln Bedeutung verleiht (vgl. [BD02], Seite 304). Die Beschreibung und die Interpretation der Beobachtungen sollen entsprechende Gestaltungsspielräume zur Verbesserung der Prozesse durch den Einsatz von Informationstechnologie aufdecken.

Das vertiefte Verständnis für die Arbeitsweise und insbesondere das geschaffene Bewusstsein für die menschlichen Bedürfnisse und die sozialen Verquickungen innerhalb der Arbeitsabläufe erlauben eine neue Qualität in der Gestaltung adäquater Systeme zur Unterstützung der Abläufe. Man ist so beispielsweise in der Lage auf unbewusste bzw. unausgesprochene Bedürfnisse, Vorlieben und habitualisierte Arbeitsweisen einzugehen und die zu entwickelnden Systeme entsprechend auszulegen.

Die Methodik kommt im Rahmen dieser Dissertation auf zwei Ebenen zum tragen: zum einen bilden intensive Beobachtung bestehender Entwicklungsprozesse bzw. Medienproduktionsprozesse die Grundlage für die Erarbeitung und Evaluation der der vorgeschlagenen Lösungsansätze. Auf diesem Wege wurden der Gesamtkontext typischer Projekte und allgemeine Zusammenhänge innerhalb verschiedener Projektkonstellationen untersucht und – auf der Metaebene – schrittweise verallgemeinert. Zum anderen wird ein Teil dieses Analyseschritts zu einem integralen Bestandteil der hier entwickelten Methodik konkretisiert.

2.3.4. End-User Development

Konsequent weiterentwickelt und konkretisiert werden partizipative Entwicklungsansätze in den Forderungen des *End-User Development* (kurz *EUD*), nach „echter“ Teilnahme von Nutzern bzw. Domänenexperten am eigentlichen Software-Produktions- bzw. sogar Pflegeprozess durch tatsächliche Mitgestaltung auf der Grundlage systematisch angepasster Fähigkeiten und Rollenverständnisse. Einen entscheidenden Teil, des konzeptionellen Fundaments der vorliegenden Arbeit bilden die Ergebnisse der Forschungen dieser Bestrebungen. Dabei wird davon ausgegangen dass Nutzeranforderungen nicht effektiv im Rahmen eines singulären Entwicklungsschrittes von einem auf die Technologie spezialisierten Team abschließend zu einem System umgesetzt werden können.

Es wird vielmehr als wichtig angesehen, Systeme so auszulegen, dass sie flexibel an veränderte Anforderungen angepasst werden können. Das bezieht sich sowohl auf Änderungen der Domäne selbst als auch auf die Veränderungen der Nutzergewohnheiten

und Fähigkeiten, die sich aus der Anwendung des Systems ergeben. Entscheidend ist dabei, dass die Anpassung des Systems idealerweise vom Nutzer selbst vorgenommen werden können soll. Der Anwendungskontext (*use time* bei [FG04]) wird bewusst zum Teil der Entwurfs- bzw. sogar der Implementierungsphasen im Software-Lebenszyklus. Potentielle Anwender des Systems bzw. Domänenexperten, ohne originären Bezug zur Software-Entwicklung sollen so die Systementwicklung treiben und gestalten. [FG04], [FGY+04], [PS87]

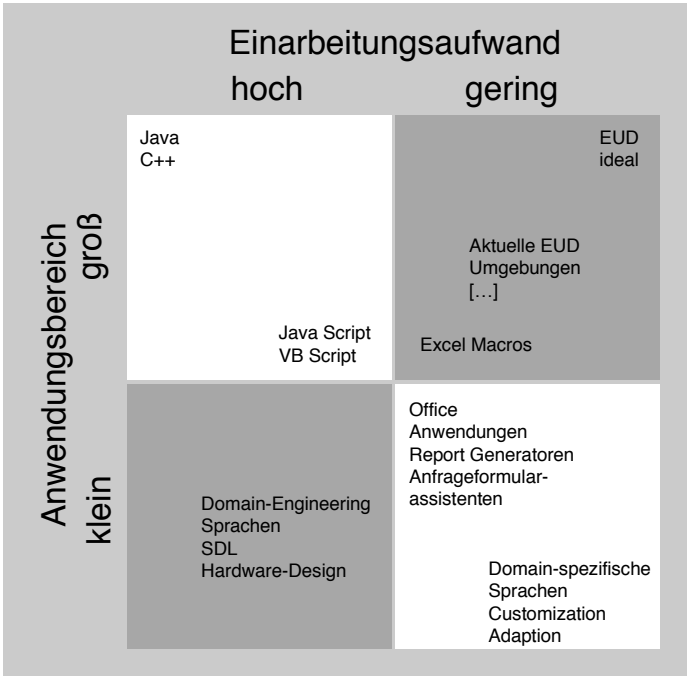


Abbildung 2.13.: Verhältnis der Einarbeitungskosten zum Einsatzbereich bei EUD-Werkzeugen nach [FGY+04]

Die Abbildung 2.13 verdeutlicht das Spektrum des Einsatzes von EUD. So ist es de facto als End-User Development zu bezeichnen, wenn technisch versierte Physiker eigene Algorithmen zur Aufnahme und Auswertung von Messwerten in C++ implementieren und evolutionär direkt durch umschreiben der Quelltexte an veränderte Versuchsaufbauten anpassen. Was die Darstellung ferner verdeutlicht, ist der Umstand, dass diese Arbeitsweise vertieftes Verständnis für die Informationstechnologie voraussetzt bzw. einen hohen Einarbeitungsaufwand erfordert. Demgegenüber bietet diese Entwicklungsplattform für das EUD die größte Flexibilität. Ist der Anwender „geschickt“ genug, kann er beliebig mächtige Software für seine Domäne entwickeln, unbenommen davon, ob er Physiker ist und sehr systemnahe technische Probleme zu lösen hat oder aber als Marketing-Assistent ein Adressverwaltungssystem mit dem Schwerpunkt bei effektiven Eingabemasken entwickeln muss.

Am gegenüberliegenden Ende des Spektrums stehen hoch spezialisierte Ansätze der Software-Entwicklung, wie etwa domänenspezifische Programmiersprachen, deren Sprachelemente und die sich ergebende Ausdrucksfähigkeit ganz auf die konkreten Belange der Anwendungsgebiet und deren Verständniswelt ausgerichtet ist. Die Sprache *LAMBDA* die als Teil des integrierten Bildverarbeitungssystems *Interactive Image Processing System (DIAS)* am Lehrstuhl für digitale Bildverarbeitung an der Friedrich-Schiller-Universität in Jena entwickelt wurde bietet Datenstrukturen wie *Bilder (Images)* und *Filter* als integralen Bestandteil der Sprache. Darüber hinaus ist die Sprache primär als Plattform für die Erforschung und die Lehre der Prinzipien der Bildverarbeitung entwickelt und damit dezidiert für diesen Einsatzkontext geeignet. Experten der Bildverarbeitung – im Sinne von EUD in diesem Falle Endnutzer – finden sich entsprechend einfach zurecht. Der nötige Einarbeitungsaufwand ist gering. Durch diese Spezialisierung sowie insbesondere auch durch die enge Kopplung an das DIAS-Ablaufsystem und dessen Eine- und Ausgabemodi bleiben die erstellten Anwendungen aber auf den Bereich der Verarbeitung von Bildern in Forschung und Lehre spezialisiert. Der Einsatz bereich ist beschränkt. An diesem Ende des Spektrums besteht die Entwicklungsleistung meist auch einfach in der Anpassung und Konfiguration bestehender Systeme.

Auch gängige Standard-Büro-Anwendungen, wie Textverarbeitungen und Tabellenkalkulationen bieten vielfältige Möglichkeiten zur Anpassung an individuelle Nutzerbedürfnisse. Neben der möglichen Zusammenstellung Von Werkzeugleisten und Ansichten, helfen Formulare, einfache Scriptsprachen bzw. der Einsatz von Makros – also der Aufzeichnung und Reproduktion einzelner Bedienabläufe – Routinearbeiten zu automatisieren. Diese Mittel sind einfacher zu erlernen als eine gängige Programmiersprache der dritten Generation. Im Gegenzug allerdings auch nicht in gleicher Weise mächtig.

Aktuelle EUD-Entwicklungsumgebungen versuchen sich dagegen dem Ideal der Endnutzerentwicklung zu nähern, indem aus der Nutzerinteraktion automatisch Bedienmuster und Programmabläufe abgeleitet werden. Damit soll erreicht werden, dass der Nutzer versucht die Probleme seiner Anwendungsdomäne zu lösen und sich das System systematisch lernend an die entsprechenden Erfordernisse nach Unterstützung anpasst.

Die Grenzen einer rein infrastrukturorientierten Unterstützung der Einbeziehung der Nutzer in den Entwicklungsprozess liegen auf der Hand. Selbst die im vorangegangenen Absatz vorgestellten adaptiven Ansätze benötigen für das ziehen sinnvoller Schlüsse aus den Nutzeraktivitäten eine entsprechende Faktenbasis, die umso universeller sein muss, je unabhängiger das System von einer partikulären Domäne arbeiten soll. Die Erarbeitung eines gemeinsamen Verständnisses für das Zielsystem und damit ein Domänen-Modells bleibt also laut [FGY+04] der schwierigste Part des End-User Development. Die folgenden Abschnitte präsenteren die Auswahl an konkreten Ansätzen, die als Fundamente direkt Eingang finden in die zu entwickelnde Methodik.

Meta-design

Das Ziel des Meta-design-Ansatzes ist es, durch das Verweben sozialer und technischer Rahmenbedingungen[8] die Voraussetzung dafür zu schaffen, dass Nutzer aktiv am Entwicklungsprozess mitwirken können. Im Mittelpunkt steht dabei die Idee, die Phase der Nutzung eines Systems (use-time) im tatsächlichen operativen Anwendungskontext bewusst zu einem Teil des Prozesses der Ausgestaltung des Systems zu machen. Der Meta-Design-Ansatz fordert dazu, dass sowohl der Entwicklungsprozess als auch das System so gestaltet werden, dass Nutzer während des gesamten Lebenszyklus des Systems zu *Co-Designern* werden können. Nutzer sollen in die Lage versetzt werden, das System jeder Zeit an die Erfordernisse ihres sich ständig ändernden Arbeitsalltags anzupassen, ohne in jedem Fall auf die eigentlichen Software-Entwickler zurückgreifen zu müssen.

Durch eine entsprechende Gestaltung von System und Entwicklungsprozess, werden so genannte *co-adaptive* Prozesse etabliert: das bedeutet einerseits, dass sowohl der Nutzer seine Fähigkeiten und Handlungsweisen an die Funktionalitäten des Systems anpassen wird, indem er lernt und gegebenenfalls die nötigen organisationsgestalterischen Maßnahmen trifft. Auf der anderen Seite soll der Nutzer von Beginn an befähigt werden, das System in seinem Sinne zu überarbeiten. Meta-design versucht den entsprechenden evolutionären Prozess der ständigen Rückkopplung zwischen System und Nutzer so zu steuern, dass sich beide weiter entwickeln im Sinne der sich kontinuierlich ändernden Anforderungen des Arbeitsalltags.

Der in der vorliegenden Arbeit vorgestellte Ansatz wird dieses Prinzip auf die Ebene der Entwicklungsumgebung und der Ausgestaltung des Entwicklungsprozesses selbst anwenden. Das heißt, nicht allein die Ziel-Anwendung selbst wird durch bzw. für den Nutzer adaptierbar, sondern zunächst die Entwicklungsumgebung und die Vorgehensmodelle für den technisch unerfahrenen Domänenexperten, der die Inhalte und Funktionalitäten des Systems initial entwickelt. Die im Sinne des EUD angepasste Anwendung ist also in einem ersten Schritt die Entwicklungsumgebung, die evolutionär und im Sinne von Meta-design zur eigentlichen Endanwendung ausgebaut wird. Der Begriff der Entwicklungsumgebung muss dabei gesamtheitlich verstanden werden: das Werkzeug und die umgebenden Prozesse der Nutzung umfassen. Die Grenze zwischen Entwicklungsumgebung und Anwendung wird fließend durch das Konzept des evolutionär entwickelten Prototypen (siehe 3.2).

Der eigentliche Endnutzer des der hier besprochenen Systeme spielt bisher eine im Bezug auf die Gestaltung eine vergleichsweise passive Rolle. Lernende bzw. zu informierende Nutzer müssen sich in der Regel den didaktischen und fachlichen Intentionen der Fachleute eher unterordnen als Nutzer eines betrieblichen Informationssystems. Das widerspricht dem steigenden Bedarf nach Unterstützung individueller Rezeptions- und Lerngewohnheiten im Sinne selbst gesteuerter auch informeller Lernprozesse. Entsprechend wird im zweiten Schritt die entstehende eigentliche Anwendung ebenfalls offen sein

[8] in [FGY+04]: socio-technical environments

für die Nutzung in co-adaptiven Prozessen des Lernens bzw. Lehrens auf der Grundlage der im ersten Schritt entwickelten Basis-Inhalte und -Funktionalitäten.

STEPS

Software Technology for Evolutionary Participative System Development (STEPS) stellt einen der ersten Ansätze dar, der die kooperative Arbeit unter direkter Einbeziehung designierter Nutzer zum methodischen Kern eines Vorgehensmodells für die Software-Entwicklung macht. Ausgehend von der Einsicht, dass Software-Entwicklungsprozesse geprägt sind vom Spannungsfeld zischen rein technischen Fragestellungen einerseits und sozialer Gestaltungspotentiale andererseits sowie der sich daraus ableitenden These, dass die Software-Entwicklung nicht im eigentlichen Sinne als Produktion (linear und deterministisch) bezeichnet werden kann, etabliert STEPS einen Rahmen für Projekte, die die Anpassung des Domänen-Verständnisses sowohl der Entwickler als auch der Nutzer im Zuge der Arbeit explizit unterstützen bzw. sogar ausnutzen.

Insbesondere die Spezifik, dass Software-Entwicklungsprozesse, immer auch den Charakter von Erkenntnisprozessen tragen wird ausdrücklich berücksichtigt. Der Entwicklungsprozess wir so gestaltet, dass Anwender die Möglichkeiten, die die Technologie eröffnet jeder Zeit erkennen und zur systematischen Effektivierung und Rationalisierung der Prozesse der Anwendungsdomäne nutzen können. Die Rückwirkung dieses Umstandes auf die Prozesse der technischen Ausgestaltung von Systemfunktionalitäten und die zunehmende Erfahrung der Entwickler auf dem Gebiet der Anwendung wird instrumentalisiert und entsprechend gesteuert. Um die nötige wechselseitige Anpassung von Arbeitsprozessen, Instrumenten und entsprechend benötigten Kompetenzen gewährleisten zu können macht STEPS den Software-Entwicklungsprozess als einen Lernprozess bewusst. Besonders die Individualität und die Multiperspektivität, die sich aus den immer wieder neuen Projektkonstellationen ergeben sowie die dadurch impliziert erforderlichen Mittel der Kommunikation und der Organisationsgestaltung sollen durch offene adaptive Entwicklungsstrategien unterstützt werden.

Dazu etabliert der Ansatz als zentrales Elemente ein Projekt-Modell (wiedergegeben in Abbildung 2.14), welches die Antizipation der Nutzung zum integralen Bestandteil des Entwicklungsprozesses macht, indem kooperative evolutionäre Entwicklungsprozesse in die eigentlichen operativen Arbeitsprozesse der Nutzer eingepasst werden (vgl. [FRS89]). Unter dem Terminus „Nutzung“ sind dabei in Abgrenzung zu den klassischen Vorgehensmodellen keine definierten Anforderungsmodelle zu verstehen, sondern der gesamte Kontext der Anwendung mit all seinen operativen Geschäftsprozessen, Organisations- und Kommunikationsstrukturen sowie den sozialen Gegebenheiten zwischen den Beteiligen. Ferner wird die Dynamik permanent – unter anderem durch die Software-Entwicklung bedingt – ablaufender Umgestaltungsprozesse innerhalb aller dieser Dimensionen kontinuierlich über die einzelnen Iterationen des evolutionären Prozesses berücksichtigt.

Der iterative Charakter des Modells stellt also eine entscheidende Voraussetzung für die Wirkungsweise von STEPS dar – materialisiert gleichsam die These, dass Software-

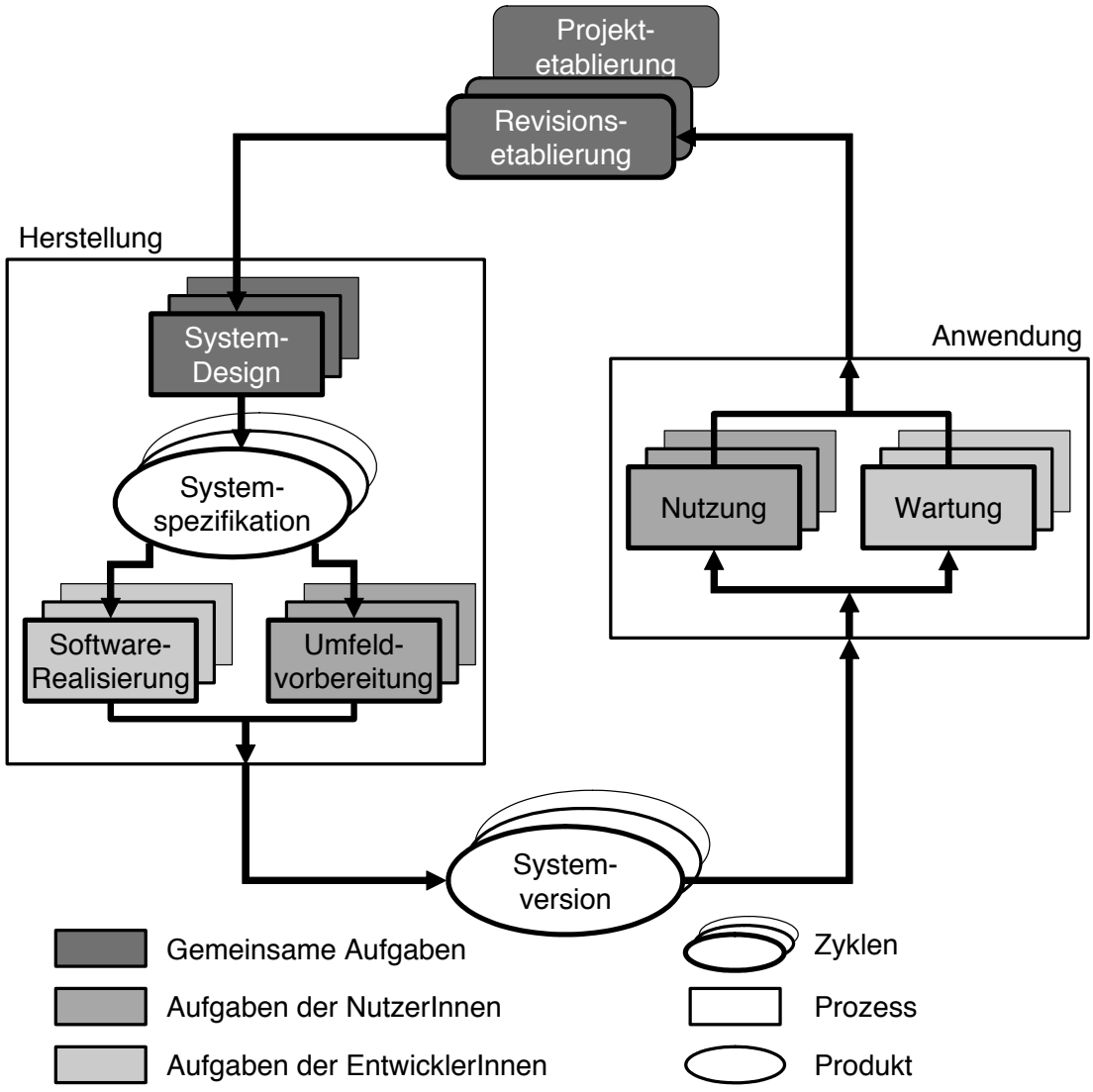


Abbildung 2.14.: Das Projektmodell von STEPS nach [FRS89] (Übersetzung ins Deutsche z.T. nach [WR95])

Entwicklung wegen der Unvorhersehbarkeit der Entwicklung von Erkenntnissen und Problemen keinen linearen deterministischen Prozessen folgen kann. Gegenstand der iterativen Be- bzw. Überarbeitung sind zwei so genannte *products* – die vordefinierten Artefakte des Projektzyklus'. Das sind zum einen definierende Dokumente (*System Specification*) mit den Beschreibungen der funktionalen Systemspezifikation als Grundlage für die technische Umsetzung der nächsten Systemversion sowie Konzepten für die arbeitsorganisatorische Umgestaltung. Dies alles mündet schließlich in neue *System versions*, die im Kern eine neue Software-Version enthält und entsprechende Dokumente zur

Unterstützung des Einsatzes. Sowohl die entstehenden Artefakte, als auch die Schritte, die im Zuge der Entstehung vollzogen werden, unterliegen einem Reifeprozess, also der kontinuierlichen Anpassung der Systemeigenschaften bzw. zunächst des Designs an das Verständnis und die Bedürfnisse des Einsatzkontextes. Umgekehrt entwickelt sich der Einsatzkontext mit den neuen ihm zur Verfügung stehen Möglichkeiten und Einsichten weiter. In Bild 2.14 wird dieser evolutionäre Prozess symbolisiert durch die in die Tiefe gestaffelten nach vorn hin stärker gerahmten Ausprägungen aller Teilschritte (Rechtecke) und Artefakte (*Product* - im Bild als Ellipse dargestellt) des Projektmodells. Die Verflechtung von Nutzung und Entwicklung des Systems ist auf drei ebenen zu sehen:

1. Die beiden Teilprozesse *Production* und *Application* gliedern den Entwicklungsprozess in die beiden zentralen Phasen: zum einen die Phase der iterativen Erstellung des Systems entsprechend einer kontextangepassten Vorgehensweise. Die entstehenden Systemversionen (Inkremente des Systems, Spezifikationen oder Prototypen) werden direkt in die operativen Arbeitsprozesse des Nutzungskontexts getragen, um etwa direkt durch die Anwendung oder durch systematische anwendungsnahe Reviews evaluiert zu werden.

2. Innerhalb dieser beiden Blöcke wird dann parallel auf den Ebenen der Anwendung (helles Grau in der Abbildung 2.14 und der Entwicklung (mittleres Grau) gearbeitet – in Prozessschritten die jeweils die Spezialisierung erfordern. Sowohl die inhaltliche als auch die zeitliche und räumliche Abhängigkeit beispielsweise zwischen den Schritten der Software-Umsetzung (*Software Realisation*) und der Vorbereitung der Einführung (*Embedment Preparation*) als Voraussetzung für die Erstellung von Systemversionen impliziert den Bedarf nach enger Kooperation an dieser Stelle des Zyklus'. *Use* bezeichnet eine Phase der Anwendung und Evaluation der *System Version* in tatsächlichen alltäglichen Arbeitsprozessen der potentiellen Nutzer. Die Nutzung wird quasi begleitet durch Wartungsaktivitäten (*Maintenance*) – die Fehler, Probleme bzw. die Notwendigkeit der Weiterentwicklung in der operativen Anwendung aufdecken. Und zum Gegenstand der weiteren Planung (*Revision establishment*) machen kann.

3. Die engste Verbindung zwischen Entwicklern der Technologie und den Experten der Domäne und damit den Gestaltern der zu unterstützenden Arbeitsprozesse besteht in den *participative Activities* (den Modellelementen mit der dunkelsten Färbung in Bild 2.14. Der Begriff der Partizipation wird von [FRS89] insofern konkretisiert, dass man nicht von bloßer Teilnahme der Nutzer am Entwicklungsprozess spricht, während die Techniker den Großteil der Verantwortung tragen sowie alle wichtigen Entscheidungen treffen. Während dieser Prozessschritte ist es vielmehr entscheidend wirklich paritätisch zu arbeiten auf gleichem Kompetenzniveau, natürlich vor komplementären fachlichen Hintergründen und in bewussten Prozessen gegenseitigen Lehrens und Lernens.

Dem *System Design* kommt damit eine besondere Rolle zu: diese Phase besteht, wie in allen gängigen Ansätzen der Systementwicklung aus der Analyse von Anforderungen, der Synthese von Lösungsansätzen und deren stetige Überarbeitung und Weiterentwicklung. Im Gegensatz zu den Grundsätzen etablierter Vorgehensmodelle stehen dabei nicht die

Formalisierung, etwa die Modellierung von Zwischenergebnissen und die strukturierte Dokumentation oder gar die Implementierung im Vordergrund. Als wichtiger wird es erachtet, den Kommunikationsprozess entlang der individuellen Bedürfnisse der konkreten Projektsituation zu gestalten, Methoden und Werkzeuge entsprechend zu wählen und anzupassen.

Für die Schritte *Project establishment* zum Auftakt des Projekts bzw. *Revision establishment* zu Beginn jeder neuen Iteration schlägt STEPS eine konkrete Methodik vor, um den Planungsprozess zu systematisieren und an die jeweils individuelle Projektsituation anzupassen. Ausgehend von der Projektidee werden im initialen Durchlauf ein Grobkonzept für das zu erstellende Systems erarbeitet und eine entsprechende Projektstrategie abgeleitet. Alle weiteren Durchläufe beziehen den Abstimmungsprozess zwischen den Nutzern und Entwicklern auf die jeweils durch den Projektverlauf veränderte Situation. So werden etwa nach jeder Iteration neue Requirements in die Konzeption einfließen und neue Planungsrisiken zu berücksichtigen sein. Um dies zu koordinieren arbeitet STEPS mit dem Konzept von *Reference Lines*. Diese definieren dynamisch Zwischenziele der Entwicklung (evaluierbare Projektzustände, wie das Vorhandensein bestimmter Artefakte) nur soweit im Voraus, wie es für die weitere arbeit nötig und möglich ist. Neue solcher Ziele werden immer erst dann definiert, wenn die ursprünglich gesteckten erreicht wurden. Das geschieht immer unter der Berücksichtigung der dann herrschenden spezifischen Gesamtsituation des Projekts.

Eine weitere methodische Maßgabe die STEPS macht ist das Prototyping. Diese Technik bietet die Möglichkeit, die in STEPS vorgeschlagenen Zyklen bereits in frühen Phasen des Projekts kurz zu halten und sehr schnell die während der *Production* entwickelten Konzepte und Lösungen in den operativen Arbeitsprozess zu tragen. Sie wird als wichtiges Mittel betrachtet, um explorativ Anforderungen zu erheben und auf experimentelle Weise den Entwurf der Nutzerschnittstelle zu entwerfen. Prototypen werden als Gegenstand der Diskussion beim Austausch zwischen den Entwicklern und den Domänensachverständigen formalen Beschreibungen vorgezogen.

Zusammenfassend lässt sich feststellen, dass STEPS als einer der ersten methodischen Ansätze versucht, die sozialen Gegebenheiten und die daraus resultierende Individualität eines jeden Software-Entwicklungsprozesses gezielt zu unterstützen und die aus intensiver Kooperation erwachsenden kreativen Potentiale zu verwerten. Das legt nahe, dass die gemachten Vorgaben lediglich allgemeine Rahmen bilden können, die auf einer Metaebene die Gestaltungsspielräume und -notwendigkeiten für die Entwicklung und Anwendung von Methoden sowie deren Anpassung an den jeweiligen Projektkontext aufzeigen. Was STEPS also nicht bietet ist eine Methode im Sinne einer tatsächlichen Handlungsanweisung, deren Einhaltung den Projekterfolg planbar macht oder gar garantiert. Als grundlegende Voraussetzung werden dementsprechend auch keine Einschränkungen bzgl. der Fachdomäne gemacht, Annahmen über Kompetenzen der Beteiligten, oder den Einsatz von Modellierungstechniken und Werkzeugen getroffen, sondern gefordert, dass die Natur des Menschen Berücksichtigung findet – seine Fähigkeit zur Kommunikation, Kooperation, gegenseitiger Akzeptanz sowie die Bereitschaft voneinander zu lernen und Wissen weiterzugeben.

Defizite für eine direkte Anwendung von STEPS erkennt [KRW96] besonders in der Fokussierung der evolutionären Entwicklung des technischen Artefakts. Organisatorische und qualifikatorische Aspekte finden demnach zwar Erwähnung, eine durch das Prozessmodell gesteuerte systematische Entwicklung etwa von Prozessen des Nutzungskontexts oder etwa von individuellen Qualifikationskonzepten für spätere Nutzer ist nicht konkretisiert.

Die ebenfalls durch [KRW96] analysierte Dichotomie zwischen Nutzung und Herstellung der Software führt nicht nur zu einer zu engen Bindung der entstehenden Artefakte an einen spezifischen Anwendungskontext. Übertragen auf die durch die vorliegende Arbeit adressierte Fragestellung der Transdisziplinarität von Medienentwicklungen, ist diese Zweiteilung darüber hinaus nicht weit reichend genug. Die Einbeziehung von Autoren und Redakteuren einerseits – Rollen also, die an sich dem Nutzerkreis nicht angehören – und letztlich einem wiederum aus vielen Rollen komplex strukturierten Anwendungskontext andererseits in den eigentlichen Entwicklungsprozess kann mit STEPS nicht ohne weiteres realisiert werden.

In [KRW96] wird weiterhin kritisiert, dass durch den Ansatz des Wechselspiels zwischen Entwicklung und Anwendung die Anpassung bestehender Software nicht Thematisiert wird. Gerade auch im Bereich der Entwicklung interaktiver, direkt reaktiver Software-Systeme spielt die Wiederverwendung zentraler technologischer Komponenten und ganzer Infrastrukturen eine wichtige Rolle. Viele der Projekte sind geprägt von der Inszenierung von Inhalten innerhalb relativ fixierter Rahmen der Navigation und auf der Basis einer Vielzahl ebenfalls vordefinierter Interaktionschemata. Dabei prägt insbesondere die Auswahl der entsprechenden Methoden und deren Arrangement im Zusammenspiel mit dem Inhalt das Entwicklungsgeschehen. Dabei führt nicht nur der Einsatz einer Infrastruktur in verschiedenen fachlichen Kontexten zur Notwendigkeit der Anpassung bestehender Software, sondern auch die Veränderung bzw. Weiterentwicklung des Anwendungskontexts selbst. Für den hier besprochenen Problembereich heißt das beispielsweise, dass Lehrsoftware, die für den Einsatz an Schulen konzipiert wurde, inhaltlich und methodisch an die durch den Föderalismus in Deutschland bedingt unterschiedlichen und sich ggf. ändernden Lehrpläne angepasst werden können muss.

Die zu entwickelnde Methodik 3.2 wird also bzgl. der Artefakt-Zentrierung und des inkrementellen sowie iterativen Vorgehens unter systematischer Einbeziehung von Nutzern zwar Anleihen bei STEPS machen. Darüber hinaus ergibt sich jedoch die Notwendigkeit von Weiterentwicklungen etwa durch den transdisziplinären Charakter der betrachteten Projekte und die Konkretisierung der in STEPS gemachten Vorgabe zur Arbeit an Prototypen durch das Angebot eines Frameworks. Die Auswirkung der Spezifik der Problemdomäne der Entwicklung interaktiver hoch reaktiver Systeme auf das Vorgehen wird ebenfalls zu untersuchen sein.

In Anlehnung an [Flo93] bzw. [FRS89] wird die bewusste Gestaltung des Software-Entwicklungsprozesses als Lernprozess für Entwickler und Anwender angestrebt unter systematischer moderierter Gestaltung der Aspekte der Kommunikation, der Arbeit und der sozialen Prozesse. Die Multiperspektivität wird dabei als Grundvoraussetzung instrumentalisiert. Laut Floyd „[sind] Anforderungen nicht gegeben, und können genau

genommen nicht analysiert werden“. Sie ergeben sich emergent im Zusammenspiel der Disziplinen. Im Diskurs werden Anforderungen antizipiert, in konstruktiven und evaluativen Schritten in der *Interaktion* von Nutzern und Entwicklern konsolidiert.

OTE

Besonders mit Blick auf die Vorgehensweisen, die sich tatsächlich am Dienstleistungsmarkt für die Entwicklung interaktiver Systeme etabliert haben (zusammengefasst in Abschnitt 2.2.4) scheint es notwendig zu sein, die Organisationsentwicklung innerhalb des Entwicklungsprozesses stärker und gezielter zu instrumentalisieren, um den nötigen Paradigmenwechsel hin zu „echter“ Partizipation (siehe dazu auch obigen Abschnitt 2.3.4 und [FRS89]) vollziehen zu können. Das Konzept der *Integrierten Organisations- und Technikentwicklung* (OTE) [KRW96], [VSE05b] und [WR95] erweitert die Idee von STEPS entsprechend. Das heißt, integriert in den partizipativen Prozess der Entwicklung der technologischen Artefakte werden Schritte definiert, die die organisatorischen Rahmenbedingungen auf den sich weiterentwickelnden Anwendungskontext abstimmen.

Unter Organisationsentwicklung soll dabei in Anlehnung an [KRW96] und [VSE05a] der permanent angestoßene, sukzessive, langfristige organisationsweite Wandel im Verhalten, den Einstellungen und den Fähigkeiten der Organisationsmitglieder entsprechend bestimmter Kriterien verstanden werden.

Übertragen auf den Anwendungskontext der in dieser Arbeit vorgestellten Methodik wird das etwa bedeuten, dass das Bewusstsein für die Arbeit in einem transdisziplinären Umfeld bei allen Beteiligten geschaffen werden muss. Das mit der Veränderung des technischen Entwicklungsprozesses verbundene anzupassende Rollenverständnis zieht eine Entwicklung der Anforderungen an die Kompetenzen jedes Einzelner nach sich. Auch umgebende Geschäftsprozesse, etwa zur Beschaffung von Medien, sind perspektivisch von den Maßnahmen der Organisationsentwicklung betroffen. Die Konsequenzen der Organisationsentwicklung können unter Umständen sogar zu einer strategischen Neuorientierung beteiligter Unternehmen führen, wenn sich etwa neue, effektivere Produkt- und Dienstleistungsangebote aus den neuen Prozessen ableiten lassen.

Organisationsentwicklung ist in der Regel ein Prozess der vom Management der Organisation betrieben wird. Einer Untersuchung der aktuellen Situation folgt dabei in der Regel eine Rückkopplung an die betroffenen Organisationsmitglieder, eine Planung so genannter Interventionen und schließlich die Anwendung der Interventionen. Typische solcher Interventionen sind:

- Feedback
- Veränderung des Normen- und Wertesystems
- Steigerung der Kommunikation und Interaktion
- Konfrontation mit Schlichtung und Aushandlung
- Qualifizierung durch die Vermittlung von Kompetenzen

Im Idealfall vollzieht sich der auf der Aktionsforschung beruhende Zyklus der Organisationsentwicklung iterativ und entwickelt die Organisation evolutionär entsprechend der definierten Kriterien weiter.

In der Praxis tragen diese Prozesse selten den nötigen selbstreflexiven Charakter und die Organisationsmitglieder selbst erhalten kaum den nötigen Gestaltungsspielraum. Es werden vielmehr die vom Management festgesetzten Ziele verfolgt – Veränderungen werden der Organisation eher „aufgezwungen".

Aus dieser Kritik heraus, versucht OTE den Gedanken der „Demokratisierung und des persönlichen Wachstums" [KRW96] der Mitglieder als Grundlage für den Wandel individueller und kollektiver Regeln zu forcieren, indem die partizipativen Aspekte der Softwareentwicklung (entsprechend dem STEPS-Modell) verwoben werden mit diskursiver Organisationsentwicklung. Dazu wird der Teilprozess *Application des STEPS-Projektmodells* insofern erweitert und konkretisiert, dass mit jeder Phase des Einsatzes einer neuen *Systemversion* ein Ausgestaltungsprozess für ein so genanntes Arbeitssystem in Gang gesetzt wird. Das Arbeitssystem repräsentiert den Gesamteinsatzkontext des entstehenden Systems und wird gleichberechtigt aufgespannt durch die Dimensionen der Technik, der Organisation und der Qualifikation.

Die Aus- bzw. Umgestaltung des Arbeitssystems läuft dabei wiederum iterativ und evolutionär. Die im übergeordneten Zyklus der Technikentwicklung entstehende Systemversion wird dazu auf der Basis der bestehenden organisatorischen Situation und dem aktuellen Qualifikationsstand der Anwender (also einem Arbeitssystem) genutzt, evaluiert und angepasst. Die Grafik 2.15 verdeutlicht durch den grau gefärbten, mit „STEPS" bezeichneten Pfeil, zu welchem Zeitpunkt eine neue Systemversion in den OTE-Zyklus eingebracht – also in ein Arbeitssystem eingebettet – wird. Die sich ergebende Problembeschreibung wird zum Gegenstand einer Ist-Analyse. Diese eruiert die Möglichkeiten bzw. Notwendigkeiten der Umgestaltung des Arbeitssystems, welche schließlich zu einem Soll-Konzept verdichtet werden. Das Soll-Konzept formuliert Ziele sowohl für die Weiterntwicklung der Technik, die daraus resultierenden Ziele für die organisatorische Umgestaltung, als auch für die nötigen qualifikatorischen Schritte einzelner Beteiligter. Die Umsetzung dieser Ziele mündet in eine neue Version des Arbeitssystems, welches in einen neuen Zyklus der Evaluation und Revision gebracht wird.

Entscheidendes Merkmal des Ansatzes ist, dass sowohl während der Konzeption als auch während der Umsetzung die Aspekte Technik, Organisation und Qualifikation verzahnt entwickelt werden. Aufgrund dieser Tatsache und auf der Grundlage der Prämisse, dass der Teilprozess der Nutzung partizipativ, unter enger Einbeziehung aller Entwicklungsbeteiligten bzw. Systembetroffenen abläuft, schafft OTE die Möglichkeit, Organisationsentwicklung tatsächlich aus der Organisation („von innen") heraus zu betreiben. Der sehr hohe Bedarf an zwischenmenschlicher Kommunikation und allgemein sozialer Interaktion im Rahmen der partizipativen Entwicklungsprozesse setzt entsprechend Ausgebildete soziale Kompetenzen bei den Prozessbeteiligten voraus. OTE trägt dem gezielt Rechnung, indem in Weiterbildungsmaßnahmen den Projektbeteiligten Problemlösestrategien und Methoden des Konfliktmanagement vermittelt werden. Dies geschieht kontinuierlich, parallel zur sachlich inhaltlichen Entwicklung des Arbeitssystems.

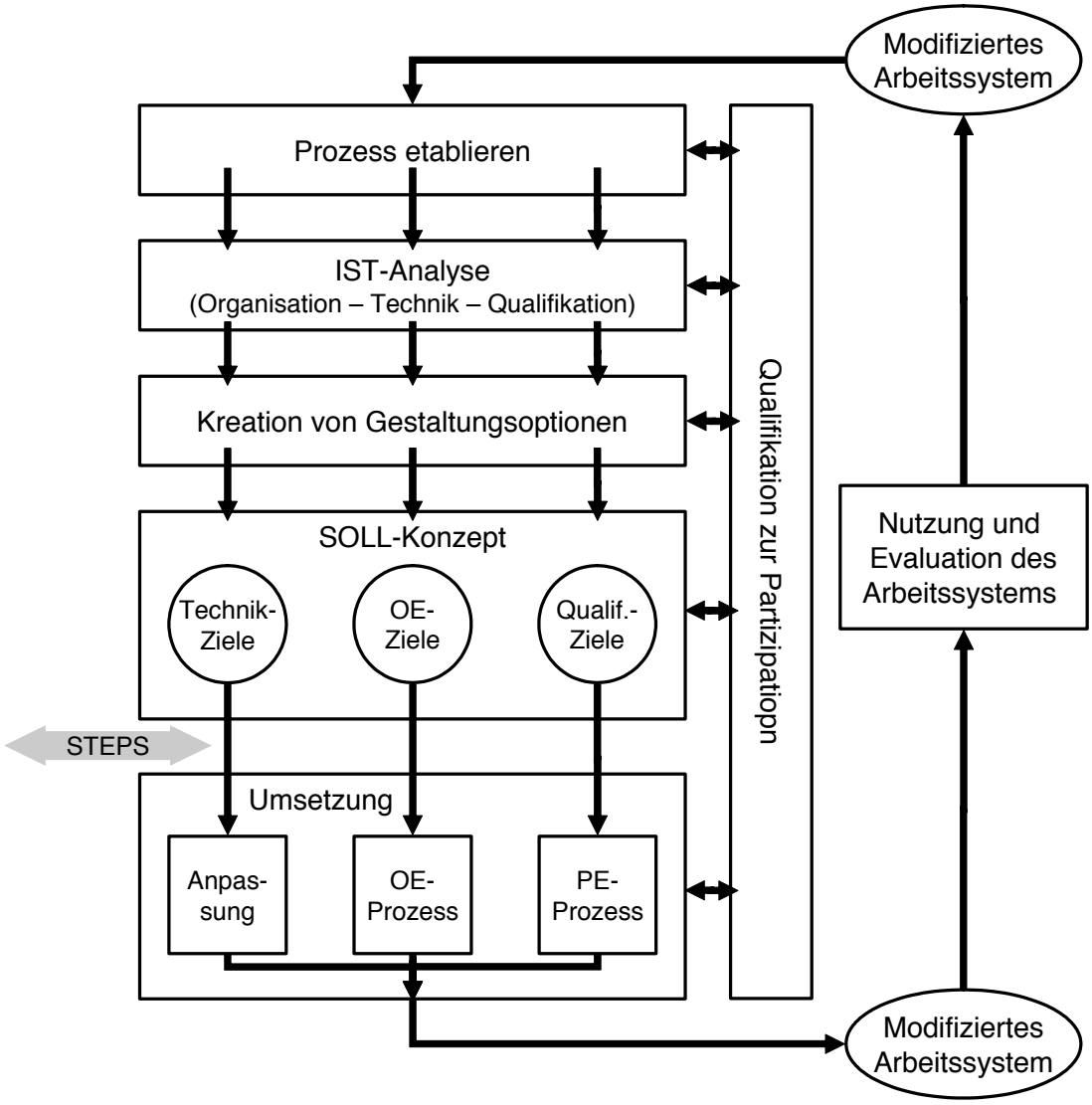


Abbildung 2.15.: Partizipative integrierte Organisationsentwicklung nach [WR95].

Die von OTE vorgeschlagenen Vorgehensweisen zur Etablierung tatsächlicher partizipativer Entwicklungsprozesse, unter engster Einbeziehung des Nutzungskontextes in alle Phasen der Entwicklung – in Abgrenzung zu regelmäßigen Projekttreffen mit Kunden und potentiellen Nutzern – und die damit verbunden notwenigen parallele Ansätze zur Weiterentwicklung der Organisation beteiligter Institutionen und der Qualifikation Ihrer Mitglieder werden auch für den hier vorgestellten Ansatz zum entscheidenden Erfolgsfaktor.

Es ist allerdings zu untersuchen, inwieweit das Modell angepasst werden muss und welche Konkretisierungen bzw. Erweiterungen an den methodischen und instrumentellen

Details notwendig werden, um den Eigenheiten der Anwendungsdomäne und der angestrebten Entwicklungsszenarien Rechnung tragen zu können. Insbesondere die hohe Interaktivität der zu entwickelnden Systeme und der zum Teil experimentelle explorative Charakter der Konzeption der Anwendungen sowie die transdisziplinäre, interorganisationale Fragmentierung von Entwicklungs- und Einsatzkontext sind durch OTE nicht ausdrücklich und im für den operativen Einsatz nötigen Detail berücksichtigt.

2.3.5. Prototyping

Prototyping gilt als unabdingbarer methodischer Bestandteil aller Entwicklungsbemühungen, die nutzerschnittstellenzenriert sind [Pre99] und bildet damit auch einen zentralen Aspekt des in dieser Arbeit vorgestellten Vorgehens. Im Folgenden werden die Aspekte des Prototyping erörtert, die als die entscheidenden Impulse für die Erarbeitung des methodischen Teils des Konzepts genutzt wurden. Neben einer Begriffsbestimmung erfolgt ein Fokussierung und Konkretisierung bekannter Verfahrensweisen für den avisierten Einsatzkontext. Insbesondere die für den interaktiven Charakter der Entwicklung und die Frameworkunterstützung im Einsatz in transdisziplinären Projekten benötigten Adaptionen werden durch bestehende Ansätze wie etwa Methoden des kooperativen Prototyping nach [BG91] motiviert.

Begriffsbestimmung

Das schnelle Ausprobieren von Lösungsalternativen in Verbindung mit intensivem Kontakt zu potentiellen Nutzern ist als Grundvoraussetzung für die Entwicklung interaktiver Systeme eine unumstrittene Notwendigkeit, die in einschlägigen Monographien wie [Pre99] propagiert wird. Das etablierteste methodische Vehikel zur Realisierung dessen ist das so genannte *Prototyping*. Darunter soll entsprechend einer Synthese aus den in [Ste91] (Seiten 49 f.) zusammengetragen Definitionen eine Vorgehensweise verstanden werden, die es unter enger Einbeziehung des potentiellen Nutzers erlaubt, Softwaresysteme oder Teile davon sehr schnell entwickeln, testen und überarbeiten zu können. Wobei entscheidend ist, dass im Gegensatz zu klassischen Vorgehensmodellen, sehr früh im Projekt ablauffähige Artefakte erstellt werden, deren Qualität oder zumindest dessen Aufgabenangemessenheit anhand der direkten Rückkopplung aus der Arbeit des Nutzers[9] mit der prototypischen Software überprüft werden kann. Prototypen bilden dabei laut [BKKZ92] (Seite 35) eine konkrete Basis der Diskussion von Schwierigkeiten, der Verdeutlichung von Problemen sowie der Vorbereitung von Entscheidungen. Der Vorteil von Prototypen gegenüber abstrakten statischen Modellen liegt dabei in der besseren *Anschauligkeit*, *Erfahrbarkeit* sowie *Bewertbarkeit* von Zwischenergebnissen der technischen Entwicklung (vgl. [Ste91] Seite 79).

[9] Prototyping, welches ausschließlich dem Ausräumen technologischer Unwägbarkeiten dient, das so genannte *„Breadboard“*-Prototyping (vgl. [BKKZ92] Seite 36) ist für den hier vorgestellten Ansatz von untergeordnetem Interesse und wird entsprechend nicht weiter vertieft.

Möglich wird die schnelle Umsetzung von Prototypen in der Regel durch die vorläufige Reduktion des Aufwandes. Dabei ergibt sich die erste Dimension der Charakterisierung der Prototypen: eine Variante bedient sich so genannter *Ein-Weg-* oder auch *Wegwerf-Prototypen*, mit deren Hilfe man lediglich exemplarische Anwendungen umsetzt. Diese zeigen in der Regel mit elementarsten Mitteln, oft ohne die Technologie der Zielarchitektur zu verwenden, die Funktionsweise des Zielsystems. Die entstehenden Artefakte dienen ausschließlich als Diskursgegenstand in der Auseinandersetzung mit den Bedürfnissen des Nutzers. Im Anschluss an die auf dieser Grundlage präzisierte Anforderungsdefinition wird der Prototyp nicht weiterentwickelt. Die gewonnenen Erkenntnisse fließen in Form der Anforderungsspezifikation in einen üblicherweise klassischen Software-Entwicklungszyklus. Dieses Vorgehen wird in der Regel angewendet, wenn über das zu erstellende System kaum Erkenntnisse vorhanden sind und wird auch als reines *exploratives Prototyping* bezeichnet (vgl. [Kro97] Seite 203).

Von Bedeutung für die in der vorliegenden Arbeit entwickelte Methodik ist der alternative bzw. weiterführende, *inkrementelle* Ansatz. Dabei wird der explorativ entwickelte Prototyp nicht verworfen, sondern entweder bezüglich des Funktionsumfangs vervollständigt *(evolvierend)* bzw. qualitativ vervollkommnet *(evolutionär)*. Das eröffnet darüber hinaus die Möglichkeit, die Software-Entwicklung als offenen Prozess zu gestalten, der es erlaubt, ein so entwickeltes System kontinuierlich an die sich ändernden Bedürfnisse der Anwendungsdomäne anzupassen. Die Software-Entwicklung nicht mehr als in sich geschlossenen, abschließbaren Prozess betrachten zu müssen, sondern vielmehr als einen Prozess der kontinuierlichen die Anwendung unterstützend begleitet (vgl. [BKKZ92] Seite 39), erlaubt es die Abläufe tatsächlich in einem transdisziplinären Kontext zu steuern. Die Kombination der Ansätze des *End-User Development* und dem entstehenden Kontinuum zwischen Anwendung und Software-Entwicklung macht den in dieser Arbeit angestrebte Veränderung der Rollenverständnisse zur Notwendigkeit: die Softwareentwickler sind damit nicht mehr länger die „Protagonisten“ eines in sich geschlossenen Prozesses sondern werden zu technologischen Beratern der Endnutzer, die im Rahmen ihrer täglichen Arbeit permanent gezwungen sind, ihr Instrumentarium (also auch die Software), weitgehend selbstständig an die sich ändernden Anforderungen anzupassen. Die Regel ist jedoch, wie die Verallgemeinerung von [Kro97] im in Abbildung wiedergegebenen Modell zeigt, dass der Anwender was die tatsächlich technologische Mitgestaltung betrifft, zunächst im Wesentlichen die Rolle des Evaluierenden und Beobachteten spielt.

Die Voraussetzung für ein solches Vorgehen bildet eine solide technologische Basis – ein „guter Gesamtentwurf der vorgesehenen Software-Architektur“ [Kro97] (Seite 204). Der Gesamtaufbau und die damit verbundene Anlage der grundlegenden Kontroll- und Datenflüsse sowie die Schnittstellen als Anknüpfungspunkte für Erweiterungen und Verfeinerungen müssen so gestaltet sein, dass die Konsistenz des Gesamtsystems in jedem Zyklus der Weiterentwicklung erhalten bleibt. Das impliziert auch klare Vorgaben für die Ausführung der Weiterentwicklung, um zu vermeiden, dass ein chaotisches, nichthomogenes System entsteht [Kro97] (Seite 204) und nicht frühzeitig antizipierte Anforderungen den Gesamtaufbau (zu) spät im Projekt doch noch zum „kollabieren“ bringen. Bereits hier deutet sich eine Stärke des framework-basierten Ansatzes an. Das Frame-

work mit seinen klaren Vorgaben bez. seiner Anwendung und die in potentiell vielen Einsätzen konsolidierte Architektur bietet in dieser Hinsicht eine optimale Voraussetzungen für ein inkrementelles Vorgehen.

Der grundsätzliche Ablauf eines Software-Entwicklungsprojekts auf der Basis inkrementellen Prototypings ist in Abbildung 2.16 in Anlehnung an [Kro97] (Seite 204) wiedergegeben. Dabei wird durch die Systementwickler aus den informellen Systemanforderungen zunächst die Definition des Prototypen abgeleitet auf deren Basis der Initiale Prototyp umgesetzt wird. Dieser wird dem Kunden zur Evaluation in der Anwendung übergeben. Das Feedback mündet in eine Liste von nötigen Veränderungen, die etwa in Abhängigkeit von einer durch den Anwender bzw. dem Auftraggeber vorgenommenen Priorisierung oder auch abhängig von der Machbarkeit in der nächsten Version des Prototypen umgesetzt werden. Das Ergebnis der Überarbeitung mündet schließlich erneut in die Anwendung und setzt damit die Iterationen des Reifeprozesses des Prototypen hin zum Endprodukt fort.

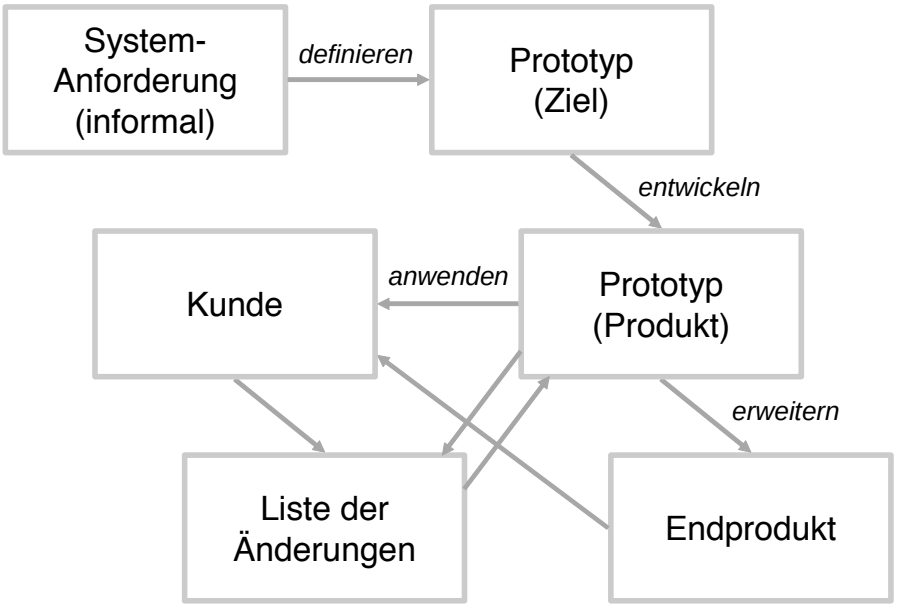


Abbildung 2.16.: Prototyping-Modell nach [Kro97] (Seite 204)

Zusammenfassend lässt sich in Anlehnung an die von [BKKZ92] beschriebenen Klassifikationskriterien für Prototypen, die sich aus dem Verhältnis des Prototypen zur Zielanwendung ergeben, für den hier betrachteten Kontext folgendes ableiten: Das auf der Grundlage einer soliden framework-basierten Architektur in frühen Phasen des Projekts zunächst explorativ entwickelte prototypische System wird schrittweise ergänzt und vervollkommnet zu einem Softwareprodukt. Der Prototyp und schließlich das daraus erwachsende Zielsystem sind das zentrale Artefakt. Sein Einsatz ist nicht beschränkt auf Aufgaben der Spezifikation bzw. der Verringerung technischer Risiken.

Ein weiteres, zur Beschreibung des in dieser Arbeit entwickelten instrumentellen Konzepts, interessantes Kriterium bildet die Einordnung bezüglich des horizontalen bzw. des vertikalen Charakters des Prototypen. Während bei den *horizontalen Prototypen* lediglich einzelne Schichten der Architektur, vorzugsweise die Nutzerschnittstelle, prototypisch umgesetzt werden, sind in *vertikalen Prototypen* komplette Teile des Zielsystems inklusive der Komponenten über alle Schichten der Architektur implementiert. Damit bieten letztere die Möglichkeit, von einem noch nicht vollständig definierten Funktionsumfang des Systems auszugehen (vgl. [BKKZ92] Seite 39). Sie bilden in der Regel den Ausgangspunkt für den inkrementellen Ausbau des Prototypen zum Zielsystem (vgl. [Ste91] Seite 79).

Dementsprechend ist das hier entwickelte interaktive framework-basierte Prototyping einzuordnen: insbesondere die Schichten die infrastrukturelle Aspekte der Architektur, wie etwa die persistente Datenhaltung, implementieren sind durch das Framework vorgegeben und per se Bestandteil, eines jeden auf dieser Basis abgeleiteten Prototypen. Eine initiale Version eines Prototypen ist auf der Basis des Frameworks zu zwar zu erzeugen und in gewisser weise auch ausführbar, jedoch ohne jede durch den Nutzer steuerbare Funktion (siehe dazu den Abschnitt 3.1.1 ab Seite 128). Sukzessive werden fachliche Strukturen, erste Funktionen der Navigation, der Interaktion und Inhalte hinzugefügt. Die initialen Schritte werden von technischen Entwicklern ausgeführt mit dem Ziel, Fachexperten und potentielle Nutzer in die Lage zu versetzen, explorativ an der weiteren Entwicklung direkt am Softwareartefakt mitzuwirken.

Werkzeugunterstützung

Die Werkzeugunterstützung des Prototyping muss im Wesentlich eines Leisten: die Erstellung und anschließende Manipulation des mit ihr erstellten Prototypen so flexibel und schnell wie möglich zu durchführbar zu machen (vgl. [Ste91] Seite 113).

Auf Grund der Tatsache, dass die Arbeit mit Oberflächenprototypen die sicher am weitesten verbreitete Form des Prototyping darstellt, hat sich seit den achtziger Jahren nichts an der Tatsache geändert, dass so genannte *Screen Generatoren* ([BKKZ92] 87 ff.) eine wichtige Kategorie in der Landschaft der Prototyping-Werkzeuge bilden. Sie erlauben es, sehr schnell und einfach, die Komponenten der Nutzerschnittstelle zu realisieren. In der Regel werden dabei nach dem so genannten WYSIWYG[10]-Prinzip Layouts und Abfolgen von Screens der Inhaltepräsentation und Dialogen der Nutzerinteraktion erstellt. Der Entwickler sieht dabei in der Bearbeitungsansicht Oberflächenkomponenten genau das, was schließlich der endgültigen Ansicht des Ergebnisses entspricht. Ein erster Eindruck der zur Wirkungsweise des Systems aus der Sicht der Handhabbarkeit durch den Nutzer, entsteht also bereits direkt während des Erstellungsprozesses. Unter anderem dieses direkte Feedback macht sich der in dieser Arbeit verwendete Ansatz des interaktiven kooperativen Prototyping zu nutze, um die *Handlungen* des Fachexperten bzw. des Nutzers in den Mittelpunkt des Gestaltungsprozesses zu stellen. Indem

[10] **W**hat **y**ou **s**ee **i**s **w**hat **y**ou **g**et. Prinzip nach dem die Bearbeitungsansicht eines Dokuments oder einer grafischen Nutzeroberfläche der endgültigen Ansicht des Ergebnisses entspricht.

der Nutzer in die Lage versetzt wird den Entwicklungsprozess, im direkten Wechselspiel zwischen seinen fachlichen Tätigkeiten und der eigenverantwortlichen Anpassung der unterstützenden Software-Werkzeuge, zu treiben, erreicht man die unmittelbarste Verbindung zwischen Anforderungsaufkommen und Realisierung.

Die für den Einsatz von Oberflächenprototypen notwendige Loslösung der interaktiven Komponenten von den Komponenten der Anwendungslogik stellt im Bereich interaktiver Systeme üblicherweise keinen limitierenden Faktor für den Einsatz der Methodik dar. Naturgemäß sind aufwendige Geschäftslogiken, -berechnungen und -modelle nicht Bestandteil der Anwendungsschicht der hier betrachteten Systeme. Selbst die dynamisch aufzubereitenden Inhalte, etwa aus dem Austausch in der Echtzeitkommunikation mit anderen Nutzern können in der Regel einfach durch statische Daten simuliert werden. So ist es etwa unerheblich, ob eine interaktive Übung an einem Textfragment zunächst mit Blindtext auf Ihre Mediendidaktische Wirksamkeit hin untersucht wird oder ob das Arrangement des Bildschirminhalts mit Medien gespeist wird, die von einer Netzwerkverbindung oder aber nur simpel von einem lokalen Speichermedium gelesen werden.

Wiederverwendbare Komponenten, die einfach und schnell etwa durch konfigurierende Schritte zu Ablauffähige Software zusammengestellt werden können, galten ebenfalls schon frühzeitig als eine technologische Basis für Prototyping [SSZ83]. Sie stellen die beste Möglichkeit dar, sehr schnell voll funktionsfähige Software zu erstellen. Dadurch, dass grundlegende standardisierte Anforderungen in vorgefertigten Komponenten bereits realisiert sind, bedarf es oft lediglich noch Schritte der Integration und der Konfiguration, um erste ablauffähige Software präsentieren zu können, die in weiten Teilen den („standardisierten“) Erwartungen der Nutzer entspricht. Das Framework als instrumenteller Teil der hier präsentierten Lösung erfüllt diese Aufgabe, mit der konzeptionellen Erweiterung, dass dabei die potentiellen Nutzer bzw. deren Vertreter in die Lage versetzt werden sollen, aus dem Fundus an angebotenen Komponenten zu schöpfen um ihre Ideen schnell und explorativ umzusetzen.

Integrierte Entwicklungsumgebungen unterstützen in gewisser Weise prototypisches Arbeiten dadurch, dass sie den Erstellungsprozess erleichtern und damit beschleunigen. Sie bilden laut den technologischen Rahmen bei der Erstellung von Prototypen, indem sie in enger Verzahnung Aktivitäten des Entwurfs, der Implementierung (einschließlich des kompletten Build-Prozesses), des Test, des Konfigurationsmanagements und zum Teil des Projektmanagements unterstützen. Die Entwicklungsumgebung führt damit Prototyping-Prinzipien, wie den Einsatz von wiederverwendbaren Komponenten oder die visuelle Erstellung von Oberflächenkomponenten instrumentell zusammen. Die von [BKKZ92] (Seite 145) geäußerten Zweifel daran, dass die Entwicklungsumgebungen mit Ihrer Fixierung auf jeweils *eine* Entwicklungsmethode für zur Unterstützung von Prototyping tauglich sein könnten, versucht der hier beschriebene Ansatz für seine Entwicklungsumgebung a priori zu entkräften. Eines der Tragenden Elemente dabei ist die gezielte Anpassung der Entwicklerschnittstellen an den Entwicklungskontext. Die Grundlage dafür ist eine adaptierbare Entwicklungsumgebung, um die Handhabung an die Bedürfnisse, die Tätigkeitsprofile und den Entwicklungsablauf innerhalb der von der

Fachlichkeit der Domäne geprägten Entwicklungssituation anzupassen und damit die Erstellungs- und Änderungsprozesse tatsächlich zu beschleunigen.

Durch ihre Fähigkeit zur Abstraktion von physischen Aspekten der Repräsentation und damit der Möglichkeit, sich auf fachlich konzeptionelle Aspekte der Entwicklung zu beschränken, etwa durch die Arbeit an rein konzeptionellen Modellen, werden in [BKKZ92] 102 ff. auch datenbankorientierte Entwicklungs-Systeme als Plattform für prototypisches Arbeiten betrachtet. Ähnlich Argumente sprechen für den Einsatz so genannter *Very High Level Languages*. Die Programmiersprachen versuchen den Fokus der Entwicklung auf die Formulierung dessen zu setzen, *was* realisiert werden muss. Dabei sollen die Aspekte dessen, *wie* es umzusetzen ist ausgeblendet werden. Die beiden Ansätze, passen auf Grund ihrer technologischen Spezifik nicht zu den hier präsentierten Lösungen und finden dementsprechend keinen direkten Eingang in die Ergebnisse. Lediglich die Einsicht, dass erst eine möglichst strikte Trennung des *„Was“* vom *„Wie“*, eine dezidiert fachlich orientierte Entwicklungsarbeit ermöglicht, fließt ein in das konzeptionelle Fundament der Methodik.

Interaktives, kooperatives Prototyping

Klassische Ansätze des Prototyping arbeiten wie angesprochen in der Regel innerhalb sehr enger Zyklen der iterierten der Erstellung und der Evaluation der Ergebnisse durch den potentiellen Anwenderkreis. Dementsprechend relativiert ist die Form der Einbeziehung der Nutzer auch zunächst im wesentlichen als beteiligungsorientiert [Ste91] zu bezeichnen. Dabei werden technische Teile der Entwicklung des Prototypen durch technisch spezialisierte Entwickler ausgeführt – entsprechend gestaltet sich die Werkzeuglandschaft. Der potentielle Nutzer oder Fachexperte konzentriert sich weiterhin weitgehend isoliert auf die Fachlichkeit. Der Übergang zu interaktivem und kooperativen Prototyping, direkt und aktiv betrieben von Nutzern wird unter anderem durch die Ideen von [BG91] vollzogen. Angeleitet durch die Entwickler der Technologie werden Fachleute und Endnutzer schrittweise in die Lage versetzt, ihre Ideen direkt prototypisch umzusetzen und damit in der permanenten Auseinandersetzung mit der Technik integriert in die Abläufe der vertrauten Tätigkeiten der Fachdomäne, Spielräume und Notwendigkeiten zu erkunden und nutzbar zu machen.

Innerhalb dieses diskursiven Prozesses dominieren die Bedürfnisse der Zieldomäne. Durch die eng verzahnte kooperative Arbeit ist der Wissensaustausch zwischen den Disziplinen jedoch grundsätzlich direkter und damit effektiver. Das heißt auch der Rückfluss von für die technische Unterstützung und Anleitung nötigen fachlichen Informationen erfolgt reibungsloser. Gestützt wird diese Behauptung etwa durch die von [Sch01] präzisierte Idee, kooperativ entwickelte Artefakte als so genanntes *Gemeinsames Material* und damit als Austauschplattform im Sinne von Wissensmanagement zu betrachten. Die entsprechenden Grundlagen werden weiter unten im entsprechenden Abschnitt 2.3.6 beschrieben. Die daraus abgeleiteten Lösungsansatze und insbesondere die für die Erstellung von Dokumenten erarbeiteten Konzepte werden in Abschnitt 3.3.2 auf die Entwicklung von Software übertragen.

Die Strategie, um das Ziel der interaktiven, kooperativen Entwicklung, orientiert an den zu unterstützenden Prozessen der Zieldomäne zu erreichen, muss sicher mehrstufig vorgehen. Es gilt den diskursiven Prozess möglich „behutsam“ in Gang zu setzen. Das bedeutet unter Umständen, dass der Prozess des interaktiven Prototyping nicht in jedem Falle, mit der Arbeit direkt am Artefakt beginnen kann. In vielen Fällen wird es nötig sein, die Art und Weise, in der das Prototyping schließlich ablaufen soll, mit empirischen Mitteln zu untersuchen. Dazu bietet sich etwa der Einsatz so genannter Mock-Ups an [EK91]. Diese besondere Form der Arbeit mit Oberflächenprototypen erlaubt es ohne umfangreichen technischen Aufwand und vor allem ohne, Berührungsängste mit der Technologie auszulösen, informationstechnologische Instrumente in den Entwicklungsdiskurs einzuführen. Beispielsweise durch die Nutzung von Nachbildungen von Interaktionselementen mit Papier, Stift und Schere. So können sehr einfach Elemente wie Dialoge, Formulare und ähnliche Software-Elemente repräsentiert werden. Auf Grund der extrem hohen Flexibilität und vor allem wegen der beinahe vollkommenen Disziplinunabhängigkeit dieses Mitteleinsatzes können alle Aspekte der Anwendungsdomäne Eingang in diese Form der Modellierung finden. Dieser Ansatz ist offen für beliebige Prozesse und Aufbaustrukturen sowie insbesondere für die subtilen Teile der mentalen Modelle aller Disziplinen. Es kann so etwa eingegangen werden auf fachspezifische Symbolik, Tätigkeitsmuster (etwa entsprungen aus Tradition oder Routine) Rollenverantwortlichkeiten und Medieneinsätze.

Aus diesen ersten „interaktiven Skizzen“ der Zielsoftware wird schließlich die Entwicklungsumgebung abgeleitet für das eigentliche interaktive Prototyping abgeleitet. Das geschieht durch die Konkretisierung des Frameworks und der entsprechenden Entwicklerschnittstellen. So gelingt eine beliebig „vorsichtige“ Transformation erster Ideen zu technischen Lösungen für die Unterstützung der fachlichen Tätigkeiten über die technologisch unterstützte Gestaltung der technischen Instrumente hin zum tatsächlichen Technologieeinsatz.

Welche Rolle der kooperative Charakter des interaktiven Prototyping und dessen Orientierung an der Tätigkeit der Fachexperten bzw. End-User bei der Vermeidung transdisziplinärer Reibungsverluste spielen kann wird in den kommenden Abschnitten präzisiert.

2.3.6. Wissens- / Content-Management

Der in dieser Arbeit präsentierte Ansatz gestaltet ganz bewusst einen Prozess systematischen Wissensmanagements. Dabei geht es zentral um die kontinuierliche Induktion, Diffusion und Anwendung von für die Entwicklung operativ benötigten Kompetenzen innerhalb eines transdisziplinären Entwicklungs-Teams. Die in den vorangegangenen Kapiteln entwickelte Begründung der Notwendigkeit einer engen Verzahnung der technischen Entwicklung mit sowohl der Entwicklung der umgebenden Organisation als auch der Kompetenzen der einzelnen Beteiligten also der gezielten Gestaltung eines soziotechnologischen Prozesses, wird in den folgenden Abschnitten auf die Grundlage der Notwendigkeiten und Lösungsansätze des Wissenmanagement gestellt. Es wird erläutert,

welche Forderungen dieser Disziplin durch das vorgestellte Vorgehen zu erfüllen werden – welchen Beitrag etwa der Einsatz kooperativen Prototypings zu einer personifizierten bzw. sozialisierenden Wissensmanagementstrategie (vgl. [FKS05]) leistet.

Wissensarbeit in transdisziplinären Software-Projekten

Die Entwicklung interaktiver, direkt reaktiver Software unter Beteiligung verschiedenster Disziplinen und unter intensiver Rückwirkung des Nutzungskontexts auf die Gestaltung des Systems vollzieht sich in einem bezüglich der Kompetenzen der Beteiligten sehr heterogenen Kontext. Die unterschiedlichen Fähigkeiten im Umgang mit den Methoden und Werkzeugen der Entwicklung und verschiednen Niveaus der fachinhaltlichen Kenntnisse führen zu einer sehr hohen Dynamik bei der Erarbeitung gemeinsamer Lösungen. Die zu erbringende Leistung entspricht in den nichttrivialen Fällen einer multidisziplinären Forschungsleistung. Dies birgt einerseits ein sehr hohes kreatives Potential, indem im wechselseitigen Diskurs – unter den Beteiligten und perspektivisch im Sinne der angebotenen Lösung in der Interaktion mit dem Zielartefakt – explorativ Lösungen und damit Wissen emergent entstehen.

Andererseits ergibt sich aus der Komplexität des Zusammenspiels der unterschiedlichen Wissensstände und der zum Teil unbestimmte Problemraum das Risiko chaotischer Projektverläufe, die geprägt sind von Missverständnissen und schließlich dem Verfehlen des Ziels, eine hochqualitatives Softwareprodukt effektiv zu erstellen.

Die Kanalisierung der heterogenen Wissensressourcen in Richtung effektiver Lösungen im Sinne der Projektziele wird zur zentralen Aufgabe innerhalb der hier adressierten Entwicklungsprojekte. Der sehr schwer zu fassende Begriff von *Wissen* soll dabei intuitiv und pragmatisch aus der diffusen Palette an Definitionen aus der Welt des Wissenmanagement ausgewählt werden. Nach [Wil01] etwa ist Wissen ein Fundus an Informationen, der sich, eingebaut in den Erfahrungskontext eines Systems, für dessen Entwicklung und Erhaltung als bedeutsam herausgestellt hat. Damit ist Wissen notwendiger Bestandteil eines zweckorientierten Produktionsprozesses [Wil01]. Als Beispiele für Ergebnisse derartiger produktiver Prozesse können so einerseits Leistungen einzelner Personen gelten, andererseits aber auch kollektiv gefertigte Güter, wie ein innerhalb eines transdisziplinären, interorganisationalen Teams entwickeltes Software-System.

In derartigen Projektkonstellationen, die die Belange der Anwendungsdomäne über den gesamten Entwicklungszyklus in den Mittelpunkt stellen, ist es entscheidend den wechselseitigen Lernprozess zwischen Systementwicklern und Domänenexperten — wie er etwa von der so genannten Skandinavischen Schule (siehe dazu 2.3.3 auf Seite 80) gefordert wird – kontinuierlich zu stimulieren. Das Wissen der Software-Ingenieure bezüglich der grundsätzlichen Möglichkeiten des Einsatzes von Informationstechnologie zur Abbildung und Unterstützung lebensweltlicher Zusammenhänge im Allgemeinen beispielsweise sollte im Verlauf des Projektes konkretisiert werden durch die Bedürfnisse der Domäne und der anstehenden Entwicklungsaufgaben. Das entsprechende Domänenwissen sollte durch die Einordnung der relevanten Informationen in den Erfahrungskontext

erworben werden. Nur durch ein vertieftes Verständnis für die Bedeutung der Kategorien und Strukturen der Zieldomäne wird es möglich die semantisch korrekte Abbildung zu den Abstraktionen der Informatik zu schaffen. Umgekehrt müssen technikferne Fachexperten lernen welche Gestaltungsspielräume der Einsatz der Technologie bietet. Am effektivsten wird dies möglich sein, wenn sie die Methoden und Werkzeuge selbst weitgehend zu beherrschen lernen. Lernen bedeutet auch hier, dass Informationsflüsse derart gezielt etabliert werden müssen, dass die Aneignung der Informationen zu eigentlichem Wissen der Fachexperten möglich wird.

Diese symmetrischen iterativen Wissensflüsse verallgemeinert die Disziplin des Wissensmanagement durch das in Abbildung 2.17 wiedergegebene Schema.

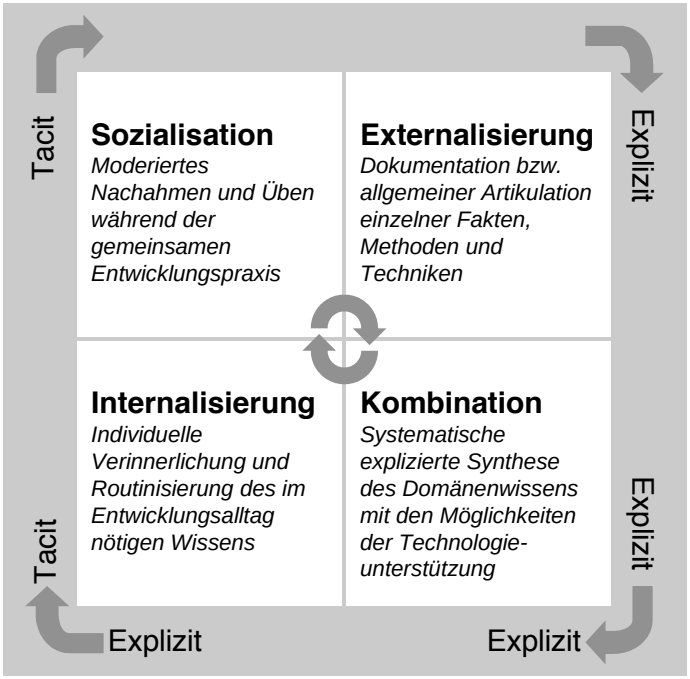


Abbildung 2.17.: Evolution des Wissens in der Software-Entwicklung adaptiert nach [NTB01] und [Wil01]).

Innerhalb der vier Phasen *Sozialisation, Externalisierung, Kombination, Internalisierung* gestaltet systematisches Wissensmanagement den Übergang des Wissens zwischen implizitem („tacit“) und explizitem („explicit“) Wissen und damit entsprechend des Modells der „*organizational knowledge creation*“ von [NTB01] die Generierung von organisationalem Wissen aus personalem Wissen und umgekehrt.

Eine beispielhafte Konkretisierung des Schemas für den Bereich der Software-Entwicklung in transdisziplinären Projektkonstellationen ist in der Abbildung bereits angedeutet und soll im Folgenden ausführlicher entwickelt werden. Den Ausgangspunkt bildet zu Projektbeginn wahrscheinlich eine Phase, die schwerpunktmäßig von der *Sozialisation*

impliziten Wissens geprägt ist. Die Projektbeteiligten mit ihren jeweiligen individuellen Erfahrungen und Kompetenzen begeben sich auf fachliches und operatives Neuland. Sie sehen sich allerdings sofort in einem Projektalltag in dem sie auf der Grundlage der bestehenden verinnerlichten meist zunächst handeln müssen. Insbesondere die sozialen kommunikativen Kompetenzen werden zunächst gebraucht, um das Team sozial zu formieren. Die Beteiligten werden sich auf ihre Intuition verlassen müssen und in einem eher improvisierten Umfeld das Projekt als solches zunächst konstituieren müssen, eine gemeinsame Vertrauensbasis aufbauen, innere Haltungen und Neigungen ergründen. Diese Phase entspricht in etwa dem was ein Projekt-Kick-Off bzw. informelle Vorgespräche leisten können.

Es wird der Punkt erreicht, an dem man formeller, auf inhaltlich sachlicher, aber auch auf methodischer und instrumenteller Ebene verabreden muss, welche Ziele zu verfolgen sind, welche Beiträge von wem zu leisten sind etc. In Präsentation und Projekt-Skizzen oder etwa auch in Exposés zu Kompetenzen werden erste Informationen explizit Kommuniziert und damit als Wissen *externalisiert*. In der Folge werden getroffene Festlegungen und erste gemeinsam erstellte Ergebnisse bzw. auch gemachte Erfahrungen zusammengestellt und traditioneller Weise in Dokumentationen, Spezifikationen und Datenbanken verdichtet. Idealerweise werden die Informationen so aufbereitet verfügbar gemacht, dass das entstehende Material als Grundlage für die Einarbeitung neu zum Projekt hinzukommender Personen dienen kann. Bezogen auf das Beispiel der Projektinitialisierung würde das etwa bedeuten, dass Festlegungen zu Begrifflichkeiten, Vorgehensweisen und zu erstellenden Artefakten in einem Projekthandbuch fixiert werden. Diese erste *Kombination* verschiedener Einzelaspekte von zum Teil orthogonaler Dimension lässt neue Zusammenhänge und neues Wissen entstehen und ist damit wichtiger Teil der Überbrückung von Verständnisbarrieren zwischen den Disziplinen.

Das in der Phase der *Kombination* auf der Basis von organisational vergemeinschaftetem, expliziten Wissen erlernte wird schließlich im Projektalltag, in der täglichen Anwendung zur Routine des Einzelnen verinnerlicht (*internalisiert*). So werden beispielsweise die definierten Prozesse „gelebt“ und das verabredete gemeinsame Vokabular zum „Argot“ des Projekts.

Der Übergang zwischen den vier Phasen wird jeweils nur bezogen auf einzelne Wissens-Assets vollzogen. Als Wissens-Assets bezeichnet [NTB01] Elemente des Wissens die als Ein- bzw. Ausgabe des Wissens-Erzeugungsprozesses eine gewisse Einheit bilden und als Ressource innerhalb eines Wertschöpfungsprozesses eine entscheidende Rolle spielen. Der Zyklus der Wissenserzeugung beginnt immer wieder neu für jeweils unterschiedliche Assets und verläuft parallel sowie verzahnt ohne scharfe Trennung zwischen den Phasen. So ist es beispielsweise denkbar, dass ein Systementwickler einem Fachredakteur ein Vorgehen zur Formatierung und Codierung von Medienmaterial erläutert — er also das entsprechende Wissen externalisiert -– während innerhalb der gleichen Diskussion die technischen Notwendigkeiten, die sich aus den Inhalten ergeben, dem Entwickler implizit klar werden, also das entsprechende Wissen dazu innerhalb seiner routinierten täglichen Arbeit in der diskursiven Auseinandersetzung damit sozialisiert wird.

Die Steuerung dieser Abläufe und eine bewusste Ausgestaltung der vier Phasen hängen ab von der verfolgten Wissensmanagementstrategie. Diese wiederum wird geprägt vom jeweiligen Arbeitskontext. Der Arbeitskontext ist im Wesentlichen bestimmt durch die verfolgten Ziele, die zu unterstützenden Aufgaben, das System an das das Wissen gebunden ist (Organisation oder Person) sowie die eingesetzte Technologie.

Entsprechend des in [FKS05] vorgestellten Spektrums lassen sich die hier adressierten Software-Entwicklungsprojekte folgendermaßen charakterisieren:

Gegenstand der Wissensarbeit: Software-Entwicklungsprozesse im Umfeld direkt reaktiver und interaktiver Systeme mit hohen Anteilen an inhaltlicher (Content) Entwicklung sind im Gegensatz zu den Abläufen während der Entwicklung betrieblicher Anwendungen kaum ausschließlich dokumentenzentriert. Die Entwicklung vieler Lösungsansätze im Spannungsfeld zwischen mediendidaktischer Konzeption, inhaltlicher Gestaltung und technischer Entwicklung auf Forschungsniveau ist viel mehr getrieben von intensivem interpersonellem Wissensaustausch und gemeinschaftlichem Diskurs.

Systembindung des Wissens: Träger des für das Projekt entscheidenden Wissens sind in erster Linie die einzelnen Protagonisten der Entwicklung. Sie bringen die Erkenntnisse und Erfahrung der jeweiligen Disziplin ein. Im Zuge der fachlichen Integration und des Diskurses der kooperativen Entwicklung kommt dabei insbesondere dieses personengebundene Wissen zum tragen. Im Verlaufe des Projekts wird sich eine Schnittmenge eines gemeinsamen Verständnisses und damit ein gemeinsamer Erfahrungs- bzw. Wissensschatz herausbilden. Dieses Wissen als soziales Produkt existiert dann auch nur im Zusammenwirken des Entwicklungs-Teams.

Unterstützte Aufgaben: Die zu erledigenden Aufgaben passen in kein festes Schema von Referenzprozessen, sie sind komplex (wie angesprochen nicht selten auf Forschungsniveau) und erfordern permanent kreative Leistungen aller Beteiligten. Oft werden Probleme erst während der Arbeit erkennbar. Entsprechend spontan müssen Lösungen gefunden werden. Aufgaben sind somit schlecht planbar — in den seltensten Fällen schematisch wiederkehrend oder gar formalisier- und damit automatisierbar.

Aus dieser Charakteristik des Arbeitsumfeldes lässt sich nun entsprechend der Ergebnisse von [FKS05] ableiten, dass eine Kommunikationsorientierte im Gegensatz zu eine Informationsorientierten Herangehensweise für die Verwaltung des Wissens im Rahmen der hier adressierten Entwicklungsvorhaben sinnvoll ist. Eine Differenzierung in eine Strategie, die die Personifizierung bzw. die Sozialisierung in den Mittelpunkt stellt bzw. eine Kombination aus beiden ist also sinnvollerweise zum Teil des Vorgehens zu machen.

Es muss also auf der einen Seite darum gehen, situativ erzeugtes Wissen einzelner Experten – Wissen also was zur Lösung der in partikulären Situationen auftretenden Probleme spontan benötigt wird – zwischen einzelnen Personen gezielt zu transferieren. Technisch soll diese Strategie der *Personifizierung* beispielsweise unterstützt werden durch die

Einrichtung von Skillmanagement-Systemen oder so genannte Expert-Maps. Diese sollen das Auffinden des Trägers des für die Lösung der anstehenden Probleme nötigen Wissens ad hoc ermöglichen. Die eigentlich Wissenserzeugung, also die Reflexionen des Experten zur vorgelegten Aufgabenstellung, sowie der Transfer zu anderen (anfragenden) Personen vollziehen sich dann informell.

Auf der anderen Seite ist die bewusste Entwicklung einer Strategie der *Sozialisierung* des Projektwissens gerade für kleinere, verteilt arbeitende Teams von großer Bedeutung. Für den gezielten Aufbau gemeinschaftlichen Wissens schlägt [FKS05] etwa vor, Foren, Wikis oder auch komplexe Wissens-Community-Plattformen als technische Unterstützung zu etablieren. Es bedarf also der Steuerung und Unterstützung des sozialen sehr dynamischen Prozesses der Kommunikation, also der Diskussion, der Bewertung und schließlich auch der Dokumentation und der Weitergabe individueller Problemstellungen, der dazugehörigen Lösungen und Lösungsstrategien einzelner Gruppenmitglieder. Die Selbstorganisation und der informelle Charakter sind dabei einerseits wichtige Voraussetzung zur Bewältigung der Aufgaben (entsprechend der Charakterisierung oben). Andererseits impliziert das Informelle als Voraussetzung für das Funktionieren der Strategie auch, dass es kaum Möglich ist, solche sozialen Prozesse bewusst zu induzieren bzw. von außen zu steuern. Laut [FKS05] ist es demnach für die umgebende Organisation lediglich wichtig, die adäquaten förderlichen Rahmenbedingungen zu schaffen.

Ein weiterer Beleg für die These, dass eine Strategie der gezielten Sozialisierung von Wissen ein wichtiger Aspekt der Ausgestaltung von Software-Entwicklungsprojekten ist, lässt sich sicher am Erfolg des Extreme-Programming ablesen. Eines der zentralen Prinzipien des *Programmierens in Paaren* (vgl. [Bec00] Seite 100 ff.) adressiert genau diesen Bedarf nach Austausch z.B. von einem Routinier zu einem Einsteiger. Neben dem unidirektionalen Transfer von fachlichem Wissen und methodischen Fertigkeiten während der Einarbeitung neuer Kollegen steht auch die soziale Integration, die Etablierung starker Teambindungen, im Mittelpunkt. Entscheidend am auf diese Weise forcierten kooperativen Diskurs ist die Ausnutzung des kreativen Potentials der kommunikativen Interaktion. Beck [Bec00] führt beispielsweise an, dass schon nach einigen Wochen davon auszugehen ist, dass neue Projektmitglieder Flüchtigkeitsfehler aufdecken werden, die der Routine versierter Entwickler entspringen. Das bedeutet auch, dass weniger wahrscheinlich effektive Lösungen der Betriebsblindheit einzelner Routiniers zum Opfer fallen.

Die Autoren des Artikels [HBM03] gehen sogar so weit, das inhärente personalisiende Wissensmanagementkonzept von XP zur didaktischen Grundlage eines Kurrikulums für die Lehre von Grundlagen des Software-Engineering zu machen. Sie machen die Erfahrung, dass sich durch die direkte und konkrete Konfrontation Techniken und Verfahren unmittelbar in Handlungskompetenz umsetzen lassen. Das Fehlen der notwendigen Erfahrung in Bezug auf zentrale Aspekte, die sich nicht ad hoc und im Diskurs erarbeiten lassen (wie etwa die Architektur), wurde dabei kompensiert durch ergänzende Maßnahmen des Coaching.

Welchen Beitrag die Ergebnisse der vorliegenden Arbeit zur Induktion eines solchen evolvierenden Prozesses der Wissensentwicklung leisten bzw. wie sie die bestehenden

Ansätze zum Teil einer Lösungsstrategie machen, wird in den Abschnitten zur Unterstützung der Wissensarbeit (3.3) beschrieben.

Grundlagen Co-Adaptiver sozio-technologischer Prozesse

Neues Wissen entsteht in den hier adressierten Entwicklungsvorhaben vor allem an den Schnittstellen zwischen den Disziplinen und insbesondere an den Berührungspunkten zwischen Fachdomäne und Technologie. In einem kontinuierlichen Prozess des wechselseitigen Lernens werden einerseits die technischen und medialen Möglichkeiten der Wiedergabe der Sachverhalte der Fachdomäne mit Hilfe von Informationstechnologien an Fachautoren, Redakteure und Grafik-Designer vermittelt. Andererseits lernen die Technologen und Programmierer nach und nach die Begrifflichkeiten und Zusammenhänge der Anwendungsdomäne kennen und damit den von der Aufgabe abhängigen individuellen Bedarf nach technische Hilfsmitteln der Präsentation, der Navigation, etc.

Gestaltungspotential für die Steuerung dieses Prozesses lässt sich über die im vorangegangenen Abschnitt vorgestellten Ansätze hinaus aus der Aktivitätstheorie ableiten. Diese stellt die Tätigkeit als vermittelndes Element in die Ringstruktur eines Wechselwirkungsprozesses zwischen das handelnde Subjekt und ein Objekt. Das Subjekt übt eine Tätigkeit aus, beeinflusst damit seine Umwelt (Objekt), generiert bzw. manipuliert Informationen aber auch Produkte. Die Ergebnisse dessen wirken auf die Tätigkeit zurück. Das heißt etwa die Arbeitsbedingungen verändern sich. Dies wiederum führt zu einer Rückkopplung zum Ausführenden.

Bezogen auf den Prozess der Software-Entwicklung im hier betrachteten Kontext bedeutet das, dass die Entwicklungsbeteiligten als Subjekte während ihrer Arbeit am zu Entwickelnden System immer wieder aufs neue Gestaltungsmöglichkeiten für das Zielssystem aber auch Möglichkeiten der entsprechenden Werkzeuge erkennen und im darauf folgenden Zyklus nutzen lernen. Die Unterstützung der aktiven kooperativen Arbeit möglichst aller Beteiligten am realen Zielartefakt stellt also eine effektive und flexible Alternative zu dokumentengetriebener Konsensfindung dar. Die Idee folgt damit den Ansätzen konstruktivistischer arbeitsprozessintegrierter Ansätze der Wissensvermittlung, wie sie etwa in aktuellen Systematiken der Weiterbildung propagiert werden (vergleiche etwa [RH02]).

Im Hinblick auf die Schwerpunktsetzung auf die Einbeziehung technikferner Disziplinen in technisch orientierte Schritte der Entwicklung bzw. die Ausweitung der Entwicklungs- oder auch der Reifephase der Software auf den Kontext der Nutzung, im Sinne von *End-user Development* (siehe Abschnitt 2.3.4) lässt sich dieses Modell zu einer Gestaltungsanforderung für die hier entwickelte Lösung konkretisieren. Die Methodik und die Entwicklungsumgebung sollten einen Prozess der Co-Adaption zwischen den Fähigkeiten und Bedürfnissen der Entwickler und den Entwicklungszugängen bzw. den der Entwicklungsumgebung induzieren. Entscheidend ist dabei, dass die gegenständliche Tätigkeit die vom Modell postulierte Vermittlerfunktion [FS99] (Seiten 50 ff.) einnimmt. Über die Idee selbst gesteuerten arbeitsprozessintegrierten Lernens hinaus, soll dabei tatsächlich

die Rückkopplung in beide Richtungen etabliert werden. Motiviert durch die explorative Tätigkeit der Entwicklung und das permanente Erkennen neuer Möglichkeiten der Operationalisierbarkeit von Tätigkeiten erweitert der Domänenexperte zwar einerseits seine Entwicklungskompetenzen, andererseits soll er in die Lage versetzt werden, den Gegenstand der Erkenntnis, also die Entwicklungswerkzeuge und ihre Wirkung, tatsächlich an seine Bedürfnisse anpassen zu können.

In Abbildung 2.18 ist die Ringstruktur der Tätigkeit nach Leontjew konkretisiert für den avisierten Kontext dargestellt. Entscheidend ist dabei die Dualität der Verfeinerung der Werkzeuge sowie der Fähigkeiten ihrer Anwendung einerseits und der damit verzahnten Entwicklung des eigentlichen Zielsystems und des Verständnisses für die technologischen Potentiale andererseits.

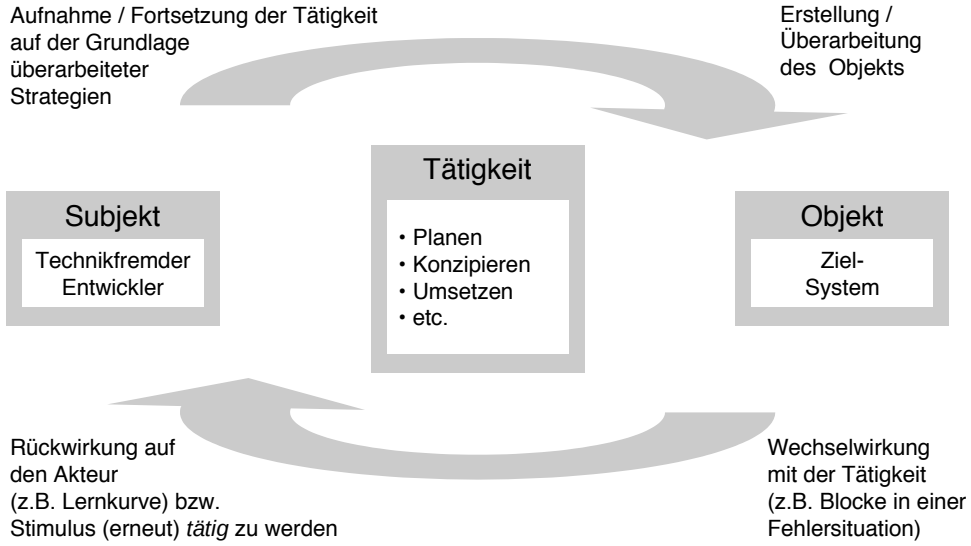


Abbildung 2.18.: Co-adaption durch Tätigkeit nach [FS99].

Die Bedeutung der Tätigkeit potentieller Nutzer als zentraler Analysegegenstand bei der Entwicklung brauchbarer Software, in Abgrenzung zum Geschäftsprozess, der in der Regel von menschlichen Motiven und Bedürfnissen abstrahiert, wird durch den in [DR97] beschriebenen Ansatz hervorgehoben. Über den in diesem Artikel vorgeschlagenen evolutionären, kooperativen Prozess hinaus soll der hier vorgeschlagene Ansatz einen Prozess der Selbstreflexion anstoßen und zielgerichtet in Gang halten.

Entwickler und Nutzer gleichermaßen sollen in permanenter Reflexion ihrer Tätigkeit, einfach durch wiederholtes zum Teil spielerisches Erfüllen und Überdenken ihrer Aufgaben entlang der Vorgaben des Frameworks und der daraus resultierenden Möglichkeiten, in Richtung der Ziele des Projekts ein Bewusstsein für die Motive ihrer Tätigkeiten entwickeln. Sie sollen dadurch dafür sensibilisiert werden zu erkennen, wann einzelne Handlungen im Rahmen ihrer Tätigkeit im Sinne von [DR97] operationalisierbar sind und damit das Potential für Automatisierung bzw. mediale Repräsentation besitzen.

Die für eine Analyse im herkömmlichen Sinne nötige Entäußerung oder zumindest strukturierte Betrachtung so genannter innerer, orientierender Tätigkeiten ist im Bereich kreativer oder gar wissenschaftlicher Arbeit kaum möglich. Daher erscheint es sinnvoll, die für eine mediale bzw. technologische Umsetzung nötigen Transformationen von Bildern, Symbolen und Sprachkonstrukten (nach [DR97] den Gegenständen innerer Tätigkeit) in Modelle oder schließlich gar Datenstrukturen und Funktionalitäten von Softwaresystemen möglichst direkt von den Trägern der Tätigkeit ausführen zu lassen. Dazu ist es notwendig die Kluft zwischen den Konzepten der realen Welt der Tätigkeit der Domänenexperten und den für eine Abbildung in Technik nötigen Abstraktionen zu verringern.

Zusammengefasst besteht eine der tragenden Ideen der in der vorliegenden Arbeit entwickelten Konzepte darin, dass Domänensachverständige durch den permanenten Umgang mit der Technologie, zunächst Berührungsängste abbauen und schließlich lernen technische Möglichkeiten für ihre alltägliche Arbeit zu instrumentalisieren. Der Prozess der Moderation, bzw. im Idealfall das Framework durch größtmögliche Intuitivität der Nutzung und Anpassbarkeit an die Begriffswelt der Domäne aus sich selbst heraus, sowie ein behutsames Vorgehen in kleinen Schritten schaffen die nötige Annäherung der Technik an die Domäne. Dieser Prozess der gegenseitigen Annäherung von Technologieentwicklung und Domäne kann entsprechend [KFK05] als co-adaptiv bezeichnet werden.

Es kann dabei ganz analog zu den von [DR97] beschriebenen kooperativen evolutionären Entwicklungsansätzen davon ausgegangen werden, dass durch die Orientierung an der Tätigkeit ein Lern- bzw. Selbstorganisationsprozess in Gang gesetzt wird. Dabei nähern sich Zielvorstellungen und Realität sowie Gewünschtes und Mögliches idealerweise systematisch aneinander an. Auf Veränderungen der Prämissen kann dynamisch reagiert werden. Dies wird möglich durch die forcierte häufige Rückkopplung zwischen der Nutzung des Zielssystems und der angepassten Entwicklungsumgebung – also der Tätigkeit der Anwender – auf der einen Seite sowie der Entwicklung der Technologie – der Tätigkeit der Systementerstellung – auf der anderen Seite. Diese enge Verflechtung zwischen Nutzung und Entwicklung ist eine entscheidende Charakteristik des interaktiven, kooperativen Prototypings.

Das Risiko von Divergenz oder gar chaotischer Zielzustände des Entwicklungs-Prozesses werden entsprechend [DR97] bereits durch die Tätigkeitsorientierung minimiert. Mit Hilfe gezielter Moderation und der permanenten Orientierung am gemeinsamen Artefakt, gestützt durch ein Framework wird dieser Bezugrahmen weiterhin konkretisiert. Damit werden die Voraussetzungen geschaffen, um eine strenge Zielorientierung kombinieren zu können mit den Vorzügen eines dynamischen kreativen und diskursiven Prozesses der kooperativen Entwicklung.

„Gemeinsames Material“ als Grundlage der Wissensarbeit

Um den Übergang von einer weitgehend durch Dokumente bestimmten und damit entkoppelnden Projektkommunikation zu einer echten Zusammenarbeit im Sinne der oben

geforderten Kooperation aller Beteiligten zu schaffen wird sich die in dieser Arbeit vorgestellte Methodik des Konzepts des *gemeinsamen Materials* [Sch01] bedienen. Das durch CSCW- Forschung für die Wissensarbeit adaptierte Prinzip geht von dem Problem aus, dass es in der „herkömmlichen“, zumeist verbalisierenden Kommunikation oft sehr aufwändig ist, individuelle Modellvorstellungen des Diskursbereichs in Übereinstimmung zu bringen. Das führt [Sch01] auf das Wesen persönlichen Verstehens und der damit verbundenen Subjektivität der Interpretation von Modellen zurück. Darüber hinaus macht er die Unzuverlässigkeit des menschlichen Gedächtnisses für die Tatsache verantwortlich, dass innerhalb der aktuellen Kommunikation häufig nur Kontextinformationen aus der unmittelbaren Historie der Kommunikation herangezogen werden.

Übertragen auf die Praxis der Software-Entwicklung lässt sich diese Beobachtung etwa dadurch bestätigen, dass anfallende Spezifikationen oft missverständlich sind und selbst Verfasser nach einiger Zeit nicht mehr wissen was sie bei der Erstellung zum Ausdruck bringen wollten. Zugespitzt ist diese Problematik in der Auseinandersetzung mit technikfernen Kunden, die als Fachexperten selten in der Lage sind, die technischen Termini oder auch die Konstrukte der Modellsprachen für die frühen Phasen der Software-Entwicklung zu verstehen.

Gerade in dem hier adressierten Bereich der Entwicklung medien- bzw. präsentationsorientierter Systeme innerhalb einer weit verteilten transdisziplinären Projektorganisation fehlt in der Regel ein systematisches Konfigurations- und Änderungsmanagement, was die Konsistenz zwischen den Ergebnissen einzelner Phasen sichern sollte. So ist es nahe liegend, dass in der Bearbeitung aktuell anstehender Aufgaben auch nur jüngst dokumentierte Festlegungen zu Rate gezogen werden und veränderte Bedingungen und Fakten nur in aktuelle Dokumente Eingang finden. Eine Rückkopplung auf ältere Dokumentation, etwa zur Verifikation im Rahmen von Abnahmen, wird durch die fehlende Aktualität früherer Diskussionsgegenstände oft unmöglich.

Den angesprochenen Übergang von in eben beschriebener Weise „anfälliger“ Kommunikation zu ergebnisorientierter diskursiver Zusammenarbeit, die das Freisetzen kreativer Potentiale und das entstehen neuen Wissens fördert, schafft [Sch01] durch den Einsatz des in Abbildung 2.19 dargestellten Modells. Dabei wird das klassische Kommunikationsmodell, welches den Austausch von Informationen vom Sender zum Empfänger über eine entsprechendes Medium etabliert (Kommunikationskanal – gestrichelt dargestellt im unteren Teil der Abbildung), erweitert um das *gemeinsame Material*. Dieses materialisiert *gemeinsame* Modellbildungen explizit. Es fungiert in Ergänzung zum weiterhin bestehenden Kommunikationskanal als gemeinsames externes Gedächtnis und Bezugspunkt für die Kommunikationsteilnehmer [Sch01]. So können die gemeinsam erarbeiteten Ergebnisse jeder Zeit wechselseitig überprüft und überarbeitet werden. Das schöpferische Potential der unterschiedlichen Perspektiven, die sich aus den verschiedenen fachlichen Hintergründen der Projektbeteiligten ergeben, kann entsprechend der Forderungen tätigkeitsorientierter kooperativer Entwicklungsansätze (siehe Abschnitte 2.3.6 bzw. 2.3.4) koordiniert ausgeschöpft werden.

Laut [Sch01] steht bei diesem Prinzip der Kooperation nicht die rein anschauliche Präsentation des Entwickelten im Mittelpunkt. Es ist vielmehr der spielerische Um-

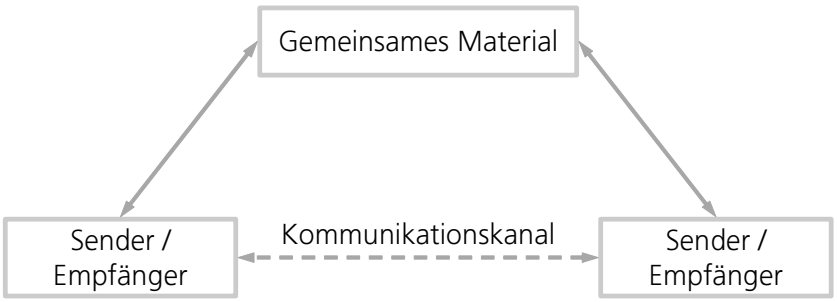


Abbildung 2.19.: Kommunikationsmodell der Zusammenarbeit nach [Sch90] und [Sch01].

gang mit Ideen und Informationen und die Interaktivität des Schaffensprozesses (schließlich des Software-Entwicklungsprozesses), die es schließlich erlauben, ein gemeinsames Verständnis zwischen den Disziplinen wachsen zu lassen und im Zuge dessen, Innovationen emergent zu entwickeln. Die Tragfähigkeit des Ansatzes bez. der Minimierung von Missverständnissen, lässt sich aus dem Umstand ableiten, dass die gesamte konventionelle Kommunikation durch die „Erdung“ im gemeinsamen Material zwangsläufig eindeutiger wird.

Bezogen auf die Arbeit mit Informationssystemen wurden diese Ansätze wie erwähnt zunächst vom das Wissensmanagement bzw. im Speziellen von der CSCW-Forschung instrumentalisierte, indem Infrastrukturen geschaffen wurden, die ein gemeinsames Arbeiten an digitalen Dokumenten ermöglichen. Auf derart verteilten Arbeitsplätzen wird es etwa möglich, in geographisch verteilten Gruppen simultan an Grafiken und Textdokumenten zu arbeiten. Eine entsprechende Software-Lösung, die mittlerweile Einsatzreife auf industriellem Niveau erreicht hat, ist etwa *Groove Virtual Office* [Gro05], mit dessen Hilfe man innerhalb so genannter *Workspaces* beispielsweise *gemeinsam* in Echtzeit auf der Basis von über das Netzwerk verteilten Client-Anwendungen an Skizzen und in Standard-Textverarbeitungen arbeiten kann.

Angewendet auf die Entwicklung von Software im Rahmen gängiger Vorgehensweisen würden solche technischen Einrichtungen etwa die Möglichkeit eröffnen Ansätze wie das Programmieren in Paaren (Extreme Programmierung [Bec00]) in geografisch verteilten Entwickler-Teams durchzuführen. UML-Modelle könnten direkt kooperativ diskutiert und bearbeitet werden. Es werden mehr oder minder identische Nutzerzugänge zum gemeinsamen Material geschaffen und den Beteiligten zur Verfügung gestellt. Eine Differenzierung nach unterschiedlichen Perspektiven (z.B. mathematisch, logisch, visuell, strukturell) [Sch01] wird zwar angestrebt; eine Individualisierung der Nutzerschnittstellen entsprechend der Disziplin und der dadurch implizierten Aufgaben bei der Bearbeitung des gemeinsamen Artefakts ist jedoch bisher höchstens in Bezug auf einfache Dokumente realisiert.

Möchte man ein Software-System in seiner Komplexität und der Mehrstufigkeit des Ab-

straktionsprozesses seiner Entwicklung als gemeinsames Material innerhalb eines transdisziplinären Entwicklungskontexts kooperativ bearbeiten lassen, benötigt man Mechanismen, die die individuellen Tätigkeiten im Rahmen der Aufgaben jeder einzelnen Disziplin reflektieren. Individuelle Entwicklungsschnittstellen zur Kooperation an einem singulären Artefakt müssen dazu im Rahmen eines Prozesses der wechselseitigen Anpassung (co-adaption [KFK05]) ausgeprägt werden. Die gemeinsame Arbeit benötigt einen methodischen Rahmen, um den Entwicklungsprozess über alle Stufen der Abstraktion also auch die konzeptionellen Phasen unterstützen zu können.

Wie eine Instanziierung des in Abbildung 2.19 dargestellten Modells auf der Grundlage der Nutzung eines Frameworks zu gestalten ist, wird in Abschnitt 3.3.2 ausgeführt.

Die gemeinsame fach-inhaltliche Wissensbasis

Wie in Abschnitt 3.3 herausgearbeitet, benötigen die hier angestrebten Projektkonstellationen Rahmenbedingungen, die den informellen spontanen Austausch fördern. Entsprechend der Vorschläge von [FKS05] sollen beispielsweise Wikis oder Foren helfen, Ergebnisse kooperativer Leistungen zu sichern und zu kanalisieren.

Da gerade auf der Seite der rein fachlich inhaltlichen bzw. redaktionellen Produktion jedoch auch viele der Ergebnisse in strukturierter Form Eingang in das angestrebte Produkt finden müssen, wird es notwendig sein, Anleihen bei herkömmlichen Content-Management-Systemen (zum Begriff vgl. [Chr03] Seite 15) zu machen. Insbesondere die Forderung nach medienneutraler Datenhaltung sollte die zu entwickelnde Entwicklungsinfrastruktur erfüllen, um den Brückenschlag zwischen einer nutzerindividualisierten ggf. informellen die Kollaboration unterstützenden Sicht und der formatierten strukturierten Ansicht des schließlich dem Anwender präsentierten Contents zu ermöglichen.

Gängige Features wie:

- Importschnittstellen für Inhalte und entsprechende Transformationsmechanismen zwischen den benötigten Formaten
- Intuitiv bedienbare Nutzerschnittstellen zur Erfassung von Inhalten
- Unterstützung räumlich verteilter Arbeit
- Unterstützung der Zusammenarbeit mehrer Nutzer

werden benötigt, um die Entwicklung der unter umständen großen Mengen an Content mit der Entwicklung der eigentlichen Anwendung verzahnen zu können.

Aufbau und Struktur der Wissensbasis sowie die Abbildungen zwischen sind dabei so flexibel zu gestalten, dass sie die angestrebten Adaptivität des Gesamtsystems unterstützen. Die permanente Weiterntwicklung und Anpassung der Strukturierung der Inhalte vollzieht sich dabei in gleicher Weise, wie die Evolution der verwendeten Werkzeuge und des Zielsystems entsprechend der Vorstellungen aus dem vorangegangenen

Abschnitt. In der permanenten aktiven Auseinandersetzung mit der Bedeutung der Inhalte – dem Erstellen und Modellieren und Codieren von Daten- und Ablaufstrukturen – näheren sich die Systementwickler systematisch dem Verständnis der Anwendungsdomäne an. Sie sind damit in der Lage die symmetrisch ablaufende Adaption zwischen Domänenexperten und Technologie und insbesondere den Anpassungsprozess der Werkzeuge gezielter zu unterstützen.

Die gemeinsame fachliche Wissensbasis ist Teil des Prototypen, der in seiner Rolle als *gemeinsames Material* (siehe vorangegangenen Abschnitt 2.3.6) und damit als spezielle Kommunikationsplattform der Wissensarbeit im Rahmen der kollaborativen Software-Entwicklung eine neue Qualität gibt. Von diesem Konzept der direkten objektbezogenen Kommunikation und dem so unterstützten Gruppen- und damit auch Projektgedächtnis entsprechend der Ideen von [Sch01] profitiert also im Speziellen auch die transdisziplinäre Zusammenarbeit bei der Entwicklung der Inhalte.

Der in dieser Arbeit entwickelte Ansatz leistet einen entsprechenden Beitrag dadurch, dass er den Prozess des Aufbaus und der Pflege der inhaltlichen Wissensbasis durch die Vorgabe sinnvoller fester Rahmen unterstützt. Das heißt ein Sockelsystem zur Verwaltung des Contents und eine elementare Richtlinie zur darstellungsunabhängigen Strukturierung der Inhalte ist bereits vorhanden. Die projektabhängige Ausprägung erfolgt dann jeweils durch Konfiguration und die Konkretisierung abstrakter Schnittstellen und die Festlegung der adäquaten inhaltlichen Struktur und des verwendeten Vokabulars. Die vorgeschlagene Lösung wird in Abschnitt 3.3.2 im Detail vorgestellt.

2.4. Der Beitrag der Arbeit

Nachdem nun die adressierte Anwendungsdomäne sowie typische Umfelder und Instrumentarien der Entwicklung von interaktiver Software primär zur Unterstützung von Prozessen der Wissensvermittlung -und Präsentation innerhalb dieser Domänen beschrieben sind, werden die Beobachtungen im Folgenden zu einer zusammenfassenden Beschreibung des Problemkontexts verdichtet.

Auf der Basis dieser Zusammenfassung der Problemstellung und den ebenfalls in der vorangegangenen Abschnitten zusammengetragenen wissenschaftlichen Grundlagen bzw. Arbeiten zu verwandten Bestrebungen sowie den dort bereits gegebenen Anknüpfungspunkten für Lösungsmöglichkeiten, werden anschließend die Kernthesen zur Wirkungsweise bzw. Wirksamkeit des vorgestellten Ansatzes formuliert. Es wird jeweils ein Ausblick auf das Vorgehen bei der tatsächlichen Ausarbeitung der Ansätze in Kapitel 3 als Betrag der Arbeit gegeben.

2.4.1. Der adressierte Problemkontext zusammengefasst

Für die Beurteilung der Entwicklung interaktiver, direkt reaktiver Anwendungen, die dem primären Zweck der Präsentation von Inhalten und der Vermittlung von Fakten

und Handlungskompetenzen bzw. der Unterhaltung dienen, gelten die Grundsätzlich gleichen Kriterien wie bei der Entwicklung von Software im Allgemeinen. Die Aufrechterhaltung der Effektivität der Entwicklung, also der Effizienz gepaart mit der Zweckdienlichkeit, stellt jedoch auf Grund von Besonderheiten üblicher Anwendungsdomänen und der entsprechenden Entwicklungskonstellationen eine besondere Herausforderung dar. Nicht selten verlaufen Entwicklungsvorhaben problematisch (vgl. [Hay04]) – Kosten und Zeitrahmen werden gesprengt. Eine Hauptursache für das Scheitern der hier betrachteten Projekte ist jedoch die häufig unzureichende Sicherstellung der Akzeptanz durch den Nutzer. Wobei es als erschwerend angesehen werden kann, dass die meisten der relevanten Anwendungsdomänen selbst durch komplexe Zusammenhänge und Abläufe (zum Teil auf wissenschaftlichem Niveau) geprägt sind. Diese Prägung unterscheidet sich dabei häufig von der technischer oder gar explizit Informationstechnologischer Disziplinen.

Dies führt zum Problem der fehlenden Praxistauglichkeit vieler Lösungen. Die Entkopplung zwischen fachlich inhaltlicher Konzeption und Anwendung auf der einen Seite und der Entwicklung der technologischen Unterstützung auf der anderen Seite ist zu überwinden. Die zu große Differenz in Expertise – Wahrnehmungsformen, Vokabularen, Routinen und Praktiken – führt dazu, dass der für die Emergenz von Lösungen nötige Diskurs nicht in Gang gesetzt werden kann. Gestaltungschancen werden nicht identifiziert (vgl. [STMTK01]). Insbesondere nicht erkannte bzw. nicht ausgeschöpfte Potentiale der Interaktivität (vgl. auch [Sch97]) induzieren Handlungsbedarf.

Die Spezifik der Anwendungsdomäne

Die betrachtete Kategorie von Anwendungsbereichen ist dadurch charakterisiert, dass das durch sie kodifizierbare Domänenwissen nicht unmittelbar mit den gängigen Mitteln der Medien- bzw. der Software-Entwicklung in multimediale Formen der Repräsentation zu übertragen ist. Das liegt in der Regel begründet in einer tatsächlichen Kluft zwischen den traditionellen Formen der Darstellung (z.B. der fachdidaktischen Darbietung) und den neuen Möglichkeiten bzw. auf der anderen Seite auch den Restriktionen elektronischer Medien.

Für diese Anwendungsbereiche ist dennoch klar, dass die neuen Medien dringend benötigte Zugänge schaffen etwa für potentielle Rezipientenschichten, die aufgrund veränderter Wahrnehmungsgewohnheiten mit den herkömmlichen Formen der Präsentation nur noch schwer zurecht kommen. Auch eine signifikante Steigerung der Effizienz bei der Vermittlung, Präsentation und auch der Sicherung von zum Teil großen Mengen von Domäneninhalten wird eine Motivation für den Einsatz interaktiver Systeme sein.

Eine besondere Rolle spielt dabei die Unterstützung fachbezogener Tätigkeiten durch Software-Systeme. Die Übertragung von Handlungsweisen auf Interaktionschemata an der Schnittstelle zwischen Nutzer und System kann sich nicht praktikabel auf die Abbildung bzw. Nachahmung bekannter Muster in Form von Metaphern beschränken.

Tatsächliche Steigerungen der Effektivität der Arbeit unter Ausschöpfung der Potentiale der Technologie ergeben sich erst aus der wechselseitigen individuellen Anpassung. Dazu gehört insbesondere auch die Überarbeitung bekannter Tätigkeitsmuster der Fachdomäne und der entsprechenden Handlungskompetenz. Das berührt den Kontext der Ausgestaltung und den der Nutzung gleichermaßen. Die Arbeits- und Sichtweisen der fachlichen Entwickler, also der Autoren und Redakteure beispielsweise, gleichen denen der Nutzer. Interaktionsschemata und Metaphern der Entwicklung und der Nutzung können daher als weitgehend kongruent betrachtet werden.

Die Spezifik der Entwicklungskontexte

Die hier adressierten Software-Entwicklungsbemühungen zeichnen sich etwa gegenüber der Entwicklung betrieblicher Software-Systeme durch die fachliche Diversität der Zusammensetzung des Entwicklungsteams aus. Neben der Multidisziplinarität des Entwicklungsgeschehens muss in vielen bereichen davon ausgegangen werden, dass die während der Entwicklung zu erbringenden Leistungen auf Grund der Unbestimmtheit des Lösungsspektrums Forschungsanspruch erheben. In Verbindung mit dem Umstand, dass durch die Multidisziplinarität die Vorstellung von Arbeitsabläufen, also Entwicklungsprozessen divergieren, ist so ein strikt planbares Vorgehen, etwa in fest abschließbaren Phasen kaum möglich.

In bestimmten Zieldomänen werden die interdisziplinären Differenzen zwischen den am Entwicklungsprojekt beteiligten so groß, dass eine systematische effiziente Projektabwicklung praktisch unmöglich wird. Diese Differenzen sind sowohl organisatorischer, sozialer, kultureller, sprachlicher als auch methodischer Natur. Problematisch wirkt sich die dadurch in der Regel induzierte, forcierte Entkopplung der Disziplinen aus.

Eine weitere Spezifik der hier untersuchten Software-Lebenszyklen besteht in der intensiven Rückwirkung des Anwendungskontexts auf das Entwicklungsgeschehen. Diese in Kombination mit dem multidisziplinären Charakter als Transdisziplinarität beschriebene Eigenschaft wirkt einerseits dadurch, dass fachlich orientierte Entwicklungsbeteiligte die Entwicklung aus der Sicht der Nutzung, geprägt durch die Handlungsweisen und -kompetenzen der Fachdomäne betreiben. Die geforderte Interaktivität im Sinne der Unterstützung und Vermittlung fachbezogener Tätigkeiten und die fachlich inhaltliche Zentrierung der Systeme erfordern eine gewisse Dominanz der Anwendungssicht.

Ein weiterer Einfluss des Anwendungsbereichs auf die Software als Produkt ergibt sich aus der Forderung, die Systeme so auszulegen, dass sie auch während der Nutzung noch an individuelle Bedürfnisse des Einsatzes anpassen oder gar weiterentwickeln zu können. Dies wird unter anderem notwendig, um Inhalte aktuell zu halten oder an konkrete Einsatzsituationen etwa im Unterricht anzupassen. Das entspricht einer Ausdehnung der Entwicklung und der Pflege der Software in den Kontext der Nutzung. Die Forderung danach ist mit abgeschlossenen und mit herkömmlichen Mitteln (etwa der Programmierung in einer Sprache der dritten Generation) entwickelten Systemen nicht zu realisieren.

Schließlich ist auch der Anwendungskontext in sich multidisziplinär wobei die Akteure dort insbesondere keinen Bezug zur Technologieentwicklung haben.

Zumindest die Bereiche der Organisation, der Modellbildungen als Facette der Sprache und die der Methoden sind relativ gut abzugrenzen, zu analysieren und damit zu beeinflussen. Insbesondere das Requirements Engineering, während dessen die Interaktion zwischen allen Projektbeteiligten am intensivsten ist, bietet hier Ansatzpunkte für die interdisziplinäre Integration, etwa indem man Aktivitäten der Anforderungserhebung als gesteuerten Entdeckungsprozess gestaltet, der die Entwicklung durchgängig begleitet.

Die Defizite bestehender Lösungsansätze

Die meisten der Lösungen, die den hier betrachteten Problemkontext adressieren, sind spezialisiert auf einzelne Aspekte des vielschichtigen Problemraumes. Viele der innovativen Konzepte sind zunächst theoretisch, konzeptionell beschrieben und kaum ad hoc zu vereinbaren mit den sehr praktischen Bedürfnissen der Zielgruppe der Entwickler.

Auch etablierte Lösungen wenden sich in der Regel nur einem Aspekt der Entwicklung zu: Entwicklungsumgebungen bzw. Autorenumgebungen fokussieren beispielsweise wie analysiert primär auf die Aspekte der Implementierung und der grafischen Gestaltung. Wobei auch die Zielgruppe der Anwender dieser Werkzeuge festgelegt ist auf entsprechende Spezialisten. Konzeptionelle Phasen der inhaltlichen Ausgestaltung werden kaum direkt unterstützt. Ebenfalls als lediglich punktuelle Lösungen zur Unterstützung der Sammlung, Strukturierung und Aufbereitung von Medien sind Content-Managementsysteme zu betrachten. Die interaktiven Elemente, mit denen die Rezeption und die Nutzung der Inhalte dort in die Tätigkeiten der Domäne eingebunden werden können sind in der Regel auf die Navigation – also die Fähigkeiten von Web-Anwendungen beschränkt.

Die Metaphern und Abstraktionen der Entwicklerschnittstellen sind ausgerichtet am Verständnis der Technologie- bzw. Medienentwicklung (z.B. bei der Filmmetapher). Die Abstraktionen, die eine universelle Einsetzbarkeit, unabhängig von der Fachlichkeit des Zielsystems versprechen aber auch die zum Teil vorgegebenen Metaphern, die zur Umsetzung der Interaktion verwendet werden sind häufig auf die technologisch üblichen Muster reduziert. Bedienelemente entsprechend der üblichen Toolkits und betriebssystemabhängigen Styleguids, die ausschließlich zur Erstellung von im klassischen Sinne erwartungskonformen Anwendungen abzielen, sind bei der Präsentation komplexer Inhalte und Abläufe nicht immer ausreichend.

Die Werkzeuge erzeugen darüber hinaus in der Regel abgeschlossene und monolithische ausführbare Dateien bzw. Programme, die ausschließlich in spezialisierten Laufzeitumgebungen zum Ablauf gebracht werden können. Auch die Artefakte der Entwicklung (quasi der Quellcode) selbst werden häufig in intransparenten binären Formaten verwaltet. So entstehen schwer wartbare und kaum portable Systeme, die die Isolation der technischen von der inhaltlichen Entwicklung befördern.

Ansätze wie XP, OTE oder andere beschriebenen kollaborativen bzw. partizipative Entwicklungsmethoden bieten sehr viel versprechende Ansatzpunkte. Sie sind jedoch oft noch zu allgemein ausgearbeitet, um unmittelbar einsatzbereit zu sein. Die Anknüpfung der Vorgehensweisen an konkrete instrumentelle Rahmen fehlt. Sie sind darüber hinaus zu sehr und vordergründig an den Ideen der Technologie-Entwicklung im Sinne von *Engineering* orientiert. Der fachliche inhaltliche geprägte Diskurs wird immer noch dominiert von technisch abstrahierenden Termini. Das resultiert in der Regel auch in einer Konzentration auf die technische Sicht auf den Entwicklungsprozess. Die Unterstützung deduktiver Vorgehensweisen, die ausgehend von allgemeinen Modellen der logischen und physischen Konzeption schrittweise konkrete Artefakte ableiten, beschränkt sich häufig auf die Anwendung von Techniken und Sprachen der Software-Technologie. Abstraktionen im Sinne der Fachlichkeit finden in allen Ansätzen kaum Beachtung. Der Schritt vom Planen und Entwerfen (z.B. Gliedern und Auswählen), also der inhaltlichen oder didaktischen Konzeption, hin zum Implementieren (Inszenieren und Gestalten) wird kaum bewusst und angepasst an die individuellen Bedürfnisse der Zieldomäne unterstützt. Dies stellt die Notwendigkeit einer ingenieurmäßigen Arbeitsweise nicht in Frage. Doch gerade auf der Ebene der fachlich inhaltlichen Ausgestaltung sollte diese Art die Arbeit zu organisieren insofern Transparent sein, als dass sie als Auslöser von Verständnisschwierigkeiten zwischen den Disziplinen und damit als Grund für die üblichen Friktionen weitgehend ausscheidet.

Fehlende Integration bzw. Integrierbarkeit von Einzellösungen und mangelnde Durchgängigkeit und Gesamtheitlichkeit im Hinblick auf die Zusammenführung von technologischen und fachlich inhaltlichern Entwicklungsaufgaben, erschwert die Bewältigung der Komplexität üblicher Entwicklungsbestrebungen und der a priori Unbestimmbarkeit der Resultate. Die Problemstellung ergibt sich aus der Vernetzung von Disziplinen, arbeitsorganisatorischen und sozialen Strukturen sowie technologischen Ansätzen. Die Isoliert Betrachtung der Probleme ist also im Grunde nicht relevant und isolierte Lösungen sind entsprechend kaum adäquat.

Damit bietet auch keine der Lösungen einen Ansatzpunkt für eine nachhaltig strategische Anwendung, bei der sowohl die Entwicklung der Technologie, der Inhalte als auch organisatorischer Strukturen und der Kompetenzen des Personals über eine einmalige Verwertung der Ergebnisse hinaus einen Mehrwert schaffen kann.

2.4.2. Thesen zum Lösungsansatz

Wie der in Kapitel 3 ausgearbeitete Lösungsansatz, die oben nochmals zusammengefassten Probleme adressiert, wird nun in zwei Kernthesen gefasst. Deren Verifizierung bzw. Falsifizierung, differenziert nach Details der Umsetzung der Lösung, schließlich die Aufgabe des Kapitels 4 also der Evaluation sein wird.

Ein Software-Framework und das von diesem implizierte Vorgehensmodell, sowie ein klares Verständnis für die Besonderheiten eines interdisziplinär geprägten MM-Entwicklungsprojektes besonders auch die entsprechende Sensibilisierung der Projektorganisa-

tion werden die Grundlage für diese Bemühungen bilden. Der grundsätzliche Charakter des Ansatzes lässt sich als *Framework-basiertes interaktives Prototyping* zusammenfassen. Die für den jeweiligen Aufgabenkontext individualisierte Ausprägung sowohl des Frameworks, als auch der implizierten Abläufe bilden den Schlüssel zur Integration der Disziplinen.

These 1: Die spezifische instrumentelle Unterstützung

Es bedarf einer instrumentellen Grundlage, die die Entwicklung im Sinne einer integrierten Entwicklungsumgebung trägt. Diese soll in der Lage sein, den Entwicklungsprozess ganzheitlich zu unterstützen – unter Berücksichtigung individueller disziplinspezifischer Aufgaben innerhalb des Projekts. Dabei soll insbesondere auch originär fachlich inhaltliche Entwicklungsarbeit implizit technische Aufgaben der Umsetzung erfüllen. Die Entwicklungsbeiträge ursprünglich nicht technisch orientierter Domänenexperten münden aus der Sicht dieser Experten weitgehend transparent in tatsächliche Software-Artefakte, wobei vertraute Tätigkeiten der Domäne die Art der Nutzung der Entwicklungsinstrumente determinieren. Mit dieser kontextabhängigen (etwa projektabhängigen) Anpassung der Werkzeuge an die Fachlichkeit der Entwicklung soll es ganz grundsätzlich gelingen technologische Aufgabenstellungen innerhalb typischer Projektabläufe in den Hintergrund treten zu lassen.

Ein Framework wird die Konvergenz der Sicht- und Arbeitsweisen zwischen den Disziplinen der Stakeholder stützen, indem es flexibel an die Bedürfnisse, die sich aus den jeweils im Projektkontext zu erfüllenden Aufgaben ergeben, angepasst werden kann. Darüber hinaus wird das Framework insofern die Rolle einer Kollaborationsplattform spielen, als dass es das kollaborative, experimentell prototypische Arbeiten an einem gemeinsamen Artefakt als Grundlage für den unmittelbaren, kontinuierlichen, konstruktiven Diskurs zwischen den Disziplinen unterstützt. Das Framework und das auf dessen Grundlage entstehende System dienen so als Kristallisationspunkt und Diskursgenestand im Sinne eines gemeinsamen Materials, zur Führung der Entwicklungsbemühung in Richtung der Projektziele.

Es ist notwendig, den Prozess der Anpassung auf evolutionäre Weise zu betreiben. Die benötigten technologischen Gestaltungsspielräume und fachlichen Notwenigkeiten ergeben sich erst nach und nach emergent aus der direkten kollaborativen Auseinandersetzung mit dem Artefakt und damit der Entwicklung eines wechselseitigen Verständnisses insbesondere zwischen Technikern und Domänenexperten. Die Schaffung und evolutionäre Konsolidierung kollaborativ nutzbarer Entwicklungsschnittstellen, die den Anforderungen der Arbeit innerhalb der Fachdomäne und damit der Anwendungsdomäne genügen, erlauben es darüber hinaus, die Individualisierung der entstehenden Produkte im eigentlichen Kontext der Nutzung durch den Nutzer selbst weiter vorantreiben zu lassen.

Ein für die Lösung ausgenutzter Aspekt ist die Kongruenz zwischen Anwendung und inhaltlich fachlicher Ausgestaltung. So wird es möglich über den Brückenschlag zwi-

schen Technologieentwicklung und fachlicher bzw. fachdidaktischer Konzeption hinaus die Anwendung mit der Entwicklung tatsächlich verschwimmen lassen. Während der Entwicklung ist so die direkteste Rückkopplung zischen evaluierenden Nutzung und Weiterentwicklung möglich; in der eigentlichen Anwendung die direkte Individualisierung der Software für Bedürfnisse, die sich aus der Nutzung ergeben. Nutzerschnittstellen der Werkzeuge werden jeweils bezügliche ihrer Interaktivität und des Aufbaus nicht nur für die Domänenaufgaben individualisiert sondern möglichst auch schon an der Zielanwendung ausgerichtet.

Die für Frameworks charakteristische Wiederverwendung und die Vereinfachung der Technologieentwicklung durch die Konzentration auf interaktive Systeme unterstützen die Fokussierung auf die fachlich, inhaltliche Arbeit. Die Adaptierbarkeit ist durch die Verwendung eines entsprechenden Architekturmusters inhärent gegeben. Der Einsatz dieses Muster in Verbindung mit einem breiten Spektrum möglicher Zielarchitekturen gewährleistet auch die benötigte Skalierbarkeit der entstehenden Systeme: die dynamische Verbindung großer Mengen hochwertiger Inhalte mit den interaktiven Elementen der Rezeption und der Manipulation wird möglich.

These 2: Der angeleitete Diskurs als Kern einer Lösung

Eine entwicklungskontextspezifische Vereinheitlichung bzw. gezielte Verzahnung von Modellbildungen, disziplinspezifischen Abläufen und Mustern der medialen Repräsentation führen zu einer Konvergenz der beteiligten Disziplinen, die auch die kaum kontrollierbaren Brüche innerhalb der Dimensionen Kultur und Sprache aber auch der Organisation zu überbrücken hilft. Eine für alle Beteiligten transparente, planbare, steuerbare und zielgerichtete Entwicklung wird möglich.

Die Entwicklungsmethodik, die dies garantieren soll ist charakterisiert durch direkte, interaktive, durchgängige Kollaboration zwischen den Beteiligten aller Disziplinen. Das Entwicklungsvorgehen orientiert sich an der Arbeitsorganisation und den Tätigkeiten der Zieldomäne. Das erlaubt die direkte Einbeziehung der Domänensachverständigen auch in die Erfüllung originär technischer Aufgaben der Entwicklung. Angestrebt wird ein leichtgewichtiger Prozess, der das Vorgehen innerhalb eine Rahmens von Werten und Prinzipien steuert und die nötige für den konstruktiven Diskurs benötigte Co-adaption ermöglicht.

Die zweistufige Anwendung des Frameworks prägt das Vorgehen. Der sukzessive Ausbau der Entwicklungsumgebung entlang der Bedürfnisse (Abläufe, Tätigkeiten, Kompetenzen, Zuständigkeiten) des fachlichen Entwicklungskontexts bildet die Grundlage der Produktion der eigentlichen Zielsysteme.

Nach der Initialisierung des Gesamtprozesses und des Frameworks beginnt der eigentliche evolutionäre Entwicklungsprozess. Sowohl die Entwicklung der Werkzeuge als auch die verflochten ablaufende, fachlich inhaltliche Ausgestaltung des Produkts nutzen dabei Methoden des interaktiven, kollaborativen Prototypings. Im Zuge dessen werden

Anforderungen und entsprechende Gestaltungspotentiale anhand des zentralen Artefakts kontinuierlich experimentell ermittelt, realisiert und evaluiert. Es entstehen gegenseitiges Verständnis und schließlich Lösungen emergent im direkten Diskurs zwischen Domänenexperten und Technologieentwicklern. Die sich nach und nach herausbildende selbstständige Handlungsfähigkeit der fachlich orientieren Entwickler bezogen auf die direkte Umsetzung von Ideen in technische Konstrukte des Produkts wird im Rahmen der kooperativen Arbeit durch die die Systementwickler und die fortschreitende Anpassung der Werkzeuge systematisch unterstützt. Umgekehrt können die technisch orientierten Entwickler durch das im Diskurs vertiefte Verständnis für Domänenspezifika immer sensibler auf Bedarfe und Bedürfnisse der Fachleute eingehen. Zur Konvergenz des Prozesses in Richtung eines wechselseitigen Verständnisses bzw. der daraus resultierenden konstruktiven Arbeitsfähigkeit und vor allem im Hinblick auf die Ziele einer effektiven Produktentwicklung trägt zum einen die permanente Orientierung am Artefakt und die damit verbundene Rückkopplung zwischen Entwicklung und direkt erfahrbaren Projektfortschritten bei. Zum anderen wird eine moderierende Instanz benötigt, die über die Initialisierung des Frameworks und des Vorgehens hinaus immer wieder sicherstellt, dass die während des Prototypings dynamische erkannten Einzelentwicklungsziele im Rahmen globaler Zielstellungen bleiben.

Eine Voraussetzung für das Funktionieren der wechselseitigen Angleichung von Werkzeugen und damit Tätigkeitsmustern im Rahmen der täglichen Arbeit ist die systematische Ausrichtung organisatorischer Aspekte auf die veränderten Prozesse. Auch wenn sich die angestrebten Entwicklungsprozesse an der Anwendungsdomäne orientieren erfordert der Prozess tatsächliche Co-Adaption. Wenn auch behutsam, so werden mit der Einführung von Technologie in angestammte Abläufe doch Weiterentwicklungen der Arbeitsorganisation und damit auch der Rollenverständnisse und schließlich der Kompetenzen der Protagonisten vorausgesetzt. Das heißt die Entwicklung der Organisation, also die Anpassung der umgebenden betrieblichen Strukturen und Prozesse und eine Bewusste Gestaltung eines Prozesses der Kompetenz- bzw. der Personalentwicklung, wird integriert mit der Entwicklung der Technologie und den aus dem Einsatz erwachsenden Anforderungen. Der vorgestellte Ansatz macht dazu die im Prozess angelegten Mechanismen des Wissensaustauschs und -Erwerbs bewusst. Eine personale Wissensmanagementstrategie auf der Grundlage des Dialogs zwischen den Entwicklern und Ideen des informellen Lernens kommen dabei zum Tragen, indem entsprechend förderliche Rahmenbedingungen geschaffen werden. Ein gezieltes Coaching und die Moderation müssen diesen Prozess steuern.

3. Framework-basiertes interaktives Prototyping

Ein Lösungsansatz zur Bewältigung der im vorangegangenen Kapitel untersuchten Problemstellung muss im Wesentlichen drei das Problemfeld aufspannende Dimensionen adressieren. Ausgehend von der *technisch instrumentellen Ebene*, dem Konzept der Konstruktion domänenspezifischer Entwicklungsumgebungen auf der Grundlage eines adaptierbaren Frameworks, wird ein *methodischer Rahmen* entwickelt. Dieser beschreibt einerseits den Einsatz des Frameworks im Sinne einer Konkretisierung dessen für einen partikulären Entwicklungskontext und andererseits den eigentlichen Einsatz der individualisierten Entwicklungsinfrastruktur für die Erstellung der jeweiligen Zielanwendungen.

Die Einbettung dessen in einen arbeitsorganisatorischen Rahmen, unter Anwendung von Prinzipien des Wissensmanagements und der Personal- und Organisationsentwicklung, ist schließlich das dritte tragende Element der vorgestellten Konzeption. Erst gezielte Interventionen mit Mitteln dieses Gebietes machen es möglich, die Vision von effektiver transdisziplinärer Entwicklung, kontinuierlich orientiert an der Tätigkeit der Domänenexperten und der Nutzer, umzusetzen und damit einen echt interaktiven, die Disziplinen und die Technik verzahnenden Entwicklungsprozess in Gang zu halten. Vor dem Hintergrund, dass viele der zentralen Fragestellungen in den betrachteten Projekt den Anspruch von Forschungsbemühungen an sich stellen, wird damit auch erstmals bewusst dem Aspekt der emergenten Wissenserzeugung innerhalb des diskursiven Prozesses der Softwareentwicklung Rechnung getragen.

Die Aufgabe des Instrumentariums ist es die Fokussierung auf die fachliche Arbeit durch die Reduktion auf die fachlich notwendigen Interaktionsschemata der Ausgestaltung des Systems zu unterstützen. Es soll ferner der Orientierung, der Führung und Unterstützung der Moderation des diskursiven Prozesses dienen. Es forciert die Entwicklung eines zentralen Zielartefakts – eines gemeinsamen Materials als Kristallisationspunkt für einen konvergenten kollaborativen und diskursiven Prozess. Darüber hinaus bietet das Framework die Vorzüge wiederverwendbarer Software-Systeme und damit erneut verwendeter Analyseergebnisse, Entwurfsentscheidungen sowie Implementierungsleistungen.

3.1. Das Framework: „Flash Markup Engine (FLAME)“

Der Einsatz von Software-Frameworks als konsequentestes Prinzip der Wiederverwendung von Arbeitsergebnissen in der Softwaretechnik ist heute fester Bestandteil fast eines jeden Software-Projekts. In den verschiedensten Formen ihres Aufbaus und der Art ihrer Anwendung liefern sie Funktionalitäten in vorgefertigten Architekturgerüsten und nehmen den Entwicklern damit lästige Routinearbeit bei der Erstellung immer wieder benötigter Komponenten und ganzer Designs ab. Sie etablieren Standards sowohl bezüglich der Daten- als der Ablaufstrukturen ganzer Anwendungen oder zumindest von Anwendungsteilen. Die in vielen Einsätzen erprobten und weiterentwickelten Architekturen und Funktionalitäten sowie die entsprechenden Schnittellen sichern so neben der Wirtschaftlichkeit auch ein hohes Maß an Interoperabilität, Austausch- und Wartbarkeit sowie schließlich Qualität.

Beispiele für weit verbreitete Frameworks sind etwa die verschiedensten Bibliotheken zur Entwicklung von graphischen Nutzerschnittstellen analog zu gängigen Bedienparadigmen von Fenster basierten Anwendungen. Die AWT- oder SWING-Klassen, als Bestandteil der Laufzeitumgebung von Java liefern beispielsweise Standardfunktionalitäten zur Gestaltung von Nutzer-Dialogen mit Standardbedienelementen, wie Buttons, Checkboxen und Ausgabebereichen (Panels) zur Erstellung und Gliederung der Ausgabe von Text Grafiken etc. Die grundsätzliche Funktionalität und die Art und Weise der Einbindung in den Kontrollfluss des Gesamtsystems ist dabei weitgehend vorgegeben etwa über das Model-View-Conroller-Architekturmuster. Diese Art von Framework bezeichnet man auch als *Anwendungs-Framework*. Es zeichnet sich dadurch aus, dass es zunächst unabhängig ist von konkreten Anwendungs-Domänen. Mit seinem in der Regel horizontalen Charakter bedient es aber lediglich eine spezielle Ebene innerhalb einer Mehrschichtigen Architektur – im Falle des angeführten Beispiels die Präsentationsschicht. Dem gegenüber stehen Frameworks mit vertikalem Charakter. Diese treffen Festlegungen zu allen Ebenen einer Systemarchitektur und schränken damit in der Regel auch den potentiellen Einsatzkontext ein. Dementsprechend ist eine verbreitete Bezeichnung *Domänen-Framework*.

Wie Frameworks die klassische Charakteristik der Software-Entwicklung verändern fast die Abbildung 3.1 vereinfacht zusammen. Während üblicherweise für jede zu entwickelnde Anwendung ein voller Zyklus von der Analyse der Anforderungen über die Konzeption des Systems bis hin zur Umsetzung durchlaufen werden muss – einschließlich aller Querschnittsaktivitäten wie der Qualitätssicherung sowie dem Änderungs- und Projektmanagement, kann ein Großteil dieser Aufgaben – zumindest für die im Framework fixierten Teile – einmalig erledigt und dann mehrfach, in jeder Ausprägung eines konkreten Systems, davon profitiert werden. Höhere Kosten, die die Rücksicht auf die Allgemeingültigkeit der Komponenten während der Entwicklung des Frameworks erfordern, werden durch entsprechend häufigen Einsatz Gerechtfertigt.

Das hier vorgestellte Framework nun wird einerseits die Stärken eines Anwendungs-Frameworks mit denen eines Domänen-Frameworks verbinden. Das heißt, es wird sowohl

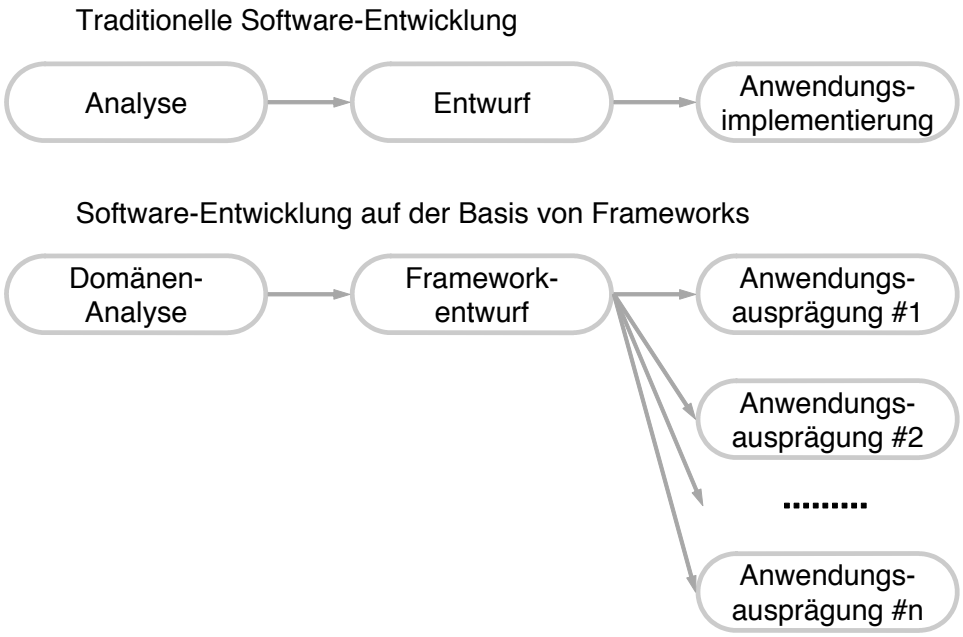


Abbildung 3.1.: Vergleich zwischen herkömmlicher und framework-basierter Software-Entwicklung

ein völlig domänenneutraler Anwendungsrahmen angeboten, dessen Einsatzmöglichkeiten dadurch effektiviert werden können, dass aus einem einfach erweiterbaren Fundus domänenspezifischer Bausteine geschöpft wird.

Andererseits wird das Vorgehen bei der Anwendung des Frameworks um einen Zwischenschritt der Abstraktion erweitert. Anstatt einen mehr oder minder starren Rahmen vorzugeben, der durch Konfiguration und die Ergänzung fachlicher Inhalte bzw. die Auswahl und Erweiterung einzelner Funktionalitäten vervollständigt wird, nutzt das hier konzipierte Entwicklungsprinzip einen flexiblen Rahmen, der zunächst für seinen konkreten Einsatzkontext jeweils individuell zu einem dedizierten Anwendungsrahmen ausgeprägt wird. Darin besteht das zentrale Konzept der technologischen Ebene des hier propagierten Ansatzes. Abbildung 3.2 illustriert den Innovationsschritt als Weiterentwicklung des in Abbildung 3.1 dargestellten herkömmlichen Einsatzprinzips von Frameworks.

Die Diskussion zum Nutzen von Frameworks und der Rentabilität ihres Einsatzes in Abhängigkeit von der Häufigkeit der Wiederverwendung scheint übertragen auf diese Idee zunächst klar zu dem Schluss zu führen, dass es schlechterdings unmöglich ist, Wiederverwendbarkeit über zwei Ebenen sinnvoll aufrechtzuerhalten. Es ist jedoch lediglich der sekundäre Zweck des hier entwickelten Frameworks, Entwicklungsaufwände durch den wiederholten Einsatz von Ergebnissen zu reduzieren. Die Art der Entwicklungsinfrastruktur dient primär als technologische Orientierungshilfe für die Gestaltung

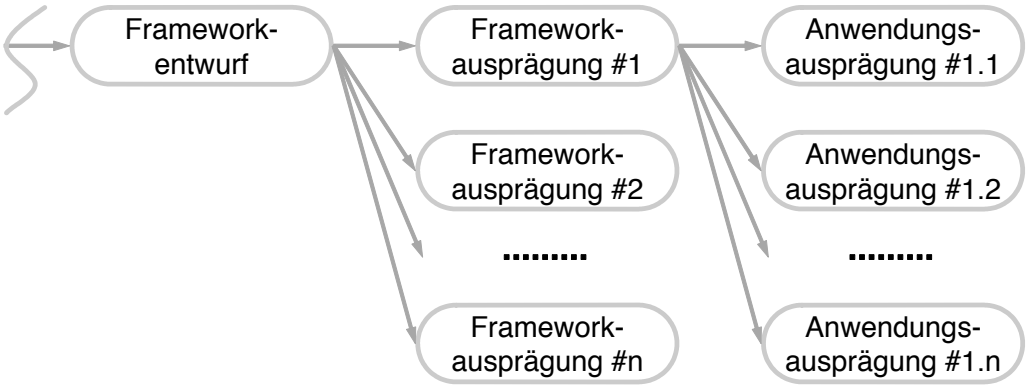


Abbildung 3.2.: Instanzenbildung vom abstrakten, über ein konkretes Framework bis hin zur Anwendung

eines Entwicklungsvorgehens, welches alle Beteiligten und insbesondere die ansonsten der Technik fern stehenden permanent in das Entwicklungsgeschehen einbezieht. Erst dieser Zwischenschritt der Individualisierung der Werkzeuge erlaubt es offenbar die nötige disziplinübergreifende Akzeptanz für die Technologiegestaltung zu schaffen, indem die Tätigkeit und damit die Kompetenzen der jeweiligen Stakeholder in den Vordergrund gerückt werden und nicht dominiert von den Möglichkeiten und Erfordernissen der Technologie.

3.1.1. Das Framework im Überblick

Die Konzeption

Das Framework insgesamt etabliert quasi die Schablone einer vollständigen Anwendung mit festgelegten grundlegenden Ablaufregimen und definierten Aufrufprotokollen für das Einbinden von Funktionalitäten. Es ist auch ohne konkretisierende Schritte übersetzbar und das Ergebnis grundsätzlich ausführbar, wenn auch ohne jegliche Funktion. Selbst das Beenden wird so lediglich mit Mitteln des Umgebenden Betriebssystems möglich sein.

Neben dem Rahmen für das zu entwickelnde Artefakt ist das Framework mit Schnittstellen versehen, die es ermöglichen Entwicklungszugänge zum System auszuprägen. Ganz analog zu den Funktionalitäten, die schließlich den Endnutzern präsentiert werden, ist es möglich, Komponenten zu konfigurieren und zu ergänzen, die zunächst die Arbeit der Gestaltung am eigentlichen Zielsystem unterstützen. Diese Besonderheit führt zu dem Zwischenschritt im Entwicklungsprozess, der in der Einführung zu diesem Kapitel oben erläutert wurde und in der Beschreibung des Vorgehens in Abschnitt 3.2 detailliert beschrieben ist.

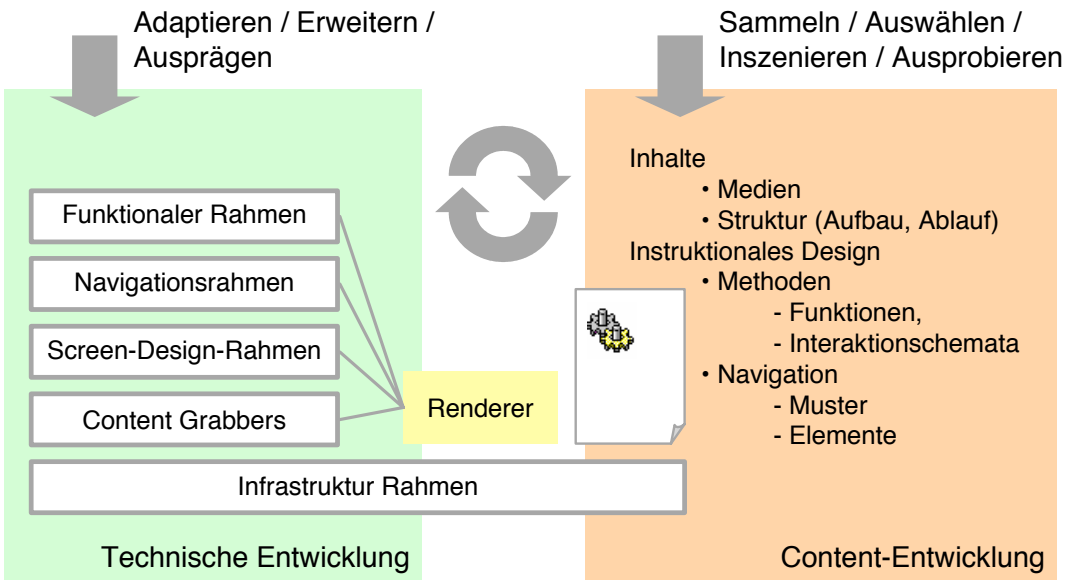


Abbildung 3.3.: Überblick über das Framework und seine Anwendung

Die Abbildung 3.3 illustriert die Funktionsweise des Frameworks im Überblick. Die Vermischung von Aspekten des Aufbaus mit denen der Anwendung erfolgt dabei bewusst, um den Zweck der einzelnen Teile im Zusammenspiel zu verdeutlichen. Das Framework selbst ist strukturiert in fünf Blöcke, die jeweils wieder Rahmen für die Ableitung von Komponenten auf verschiedenen Architekturebenen bzw. für verschiedene Aufgabenbereiche bilden.

Der Infrastrukturrahmen beinhaltet eine flexible Musterarchitektur, die die Grundlage für die Auslegung der Architektur des Zielsystems bildet. Durch den Einsatz von Web-Technologien, XML und leichtgewichtigen Basiskomponenten, kann zwischen verschiedensten Aufbauten gewählt werden. Echte Webapplikationen mit potentiell sehr vielen konkurrierenden Nutzerzugriffen oder die direkte Vernetzung lokale installierter Applikationen untereinander (Peer-To-Peer) sind dabei genau so einfach zu etablieren wie eine Standalone-Anwendung bestehend aus einer singulären Anwendungsschicht ohne Backend. Die dazu erforderlichen Funktionalitäten, etwa zur Verteilung von Systemteilen über ein Netzwerk oder auch alle nötigen Mechanismen zur persistenten Verwaltung von Daten, sind als Teil des Rahmens bereits angelegt. Sie können den Erfordernissen entsprechend konfiguriert oder aber über definierte Schnittstellen ergänzt oder überlagert werden.

Die Content Grabbers bilden die Basis der Technikunterstützung für die „sensibelste“ Schnittstelle innerhalb der avisierten Software-Entwicklungsprozesse. Auch bei dieser Komponente handelt es sich zunächst lediglich um einen allgemeinen Rahmen, der die Schnittstellen definiert, über die Inhalte in das System gelangen. Das ist der zentrale Anknüpfungspunkt für Werkzeuge, die es den potentiell nicht-

technischen Fachautoren und Redakteuren erlauben, Medien einzubinden und das System inhaltlich zu gliedern. Bei diesen Werkzeugen wird es sich sowohl um interaktive Anwendungen handeln, als auch um Adapter und Filter für die Anbindung von Systemen, die der adressierten Domäne vertraut sind. Ziel der Frameworkentwicklung ist es zwar perspektivisch eine Vielzahl nützlicher auch domänenübergreifend einsetzbarer Komponenten innerhalb dieses Frames zusammenzutragen. Hier besteht jedoch am wenigsten die Möglichkeit Vorgaben zu machen. Dieser Rahmen soll vielmehr eine möglichst allgemeingültige Schnittstelle als Ansatzpunkt für die fachliche Individualisierung der Entwicklungsumgebung bieten.

Der Funktionale Rahmen Wenn man bildlich gesprochen den Infrastrukturrahmen als das Skelett einer Anwendung bezeichnet, dann bildet der funktionale Rahmen einen Teil der Muskulatur. Die hier angelegten Komponenten etablieren zum einen die grundsätzliche Interaktivität der Anwendung. Elementare Bedienelemente wie Schaltflächen, Fenster und Textfelder sowie deren Verhalten sind hier grundsätzlich angelegt und können je nach bedarf ergänzt und angepasst werden. Zum anderen sind hier auch gängige komplexe Baugruppen zusammengestellt. So existieren etwa vorgefertigte Mechanismen zur einfachen Erstellung von Zuordnungsübungen – so genannter *Drag-and-Drop*-Aufgaben. Auch innerhalb dieses Rahmens sind Schnittstellen zur Ausprägung von Anwendungen definiert. Diese Programmierschnittstellen (APIs[1] im klassischen Sinne) erlauben die Integration von Funktionen in das System.

Der Navigationsrahmen Innerhalb der Domäne der interaktiven Systeme kommt den Funktionen der Navigation eine besondere Rolle zu. Das effiziente Auffinden und Anwählen von Inhalten bzw. Programmteilen sowie die Orientierung des Nutzers innerhalb der üblicherweise komplexen Strukturen erfordert immer wieder ähnliche querschnittliche Funktionen, die den globalen Ablauf der Anwendungen bestimmen. So ist etwa eine immer wiederkehrende Anforderung, ein Verzeichnis aller Inhalte mit einem Überblick über Aufbau und Struktur des Systems als eigenständige Funktion zu realisieren. Neben dem was man aus der Welt des WWW als Sitemap[2] kennt sind hier abhängig von der Domäne beliebige Strukturen und Metaphern denkbar. Im für die Evaluation betrachteten Projekt beispielsweise machte sich eine historische Zeitleiste, mit einer Ordnung von Inhalten bzw. Verweisen darauf über einer Zeitachse durch das zwanzigste Jahrhundert erforderlich. Es kann davon ausgegangen werden, dass eine Verquickung der inhaltlichen Struktur mit dem Aufbau des Systems generierende Schritte bei der Implementierung erlaubt.

Der Screen-Design-Rahmen bietet einen Mechanismus, um das Erscheinungsbild der Anwendung zentral verändern zu können, ohne die Funktion und Wirkungsweise von Bildschirmelementen verändern oder während der Implementierung auch nur berühren zu müssen. Die Trennung zwischen Darstellung und Funktionalität auf der Basis des so genannten Skinning-Mechanismus erlaubt es etwa, Druckknöpfen

[1] engl. **A**pplication **P**rogramming **I**nterface

[2] Einfaches, meist hierarchisch aufgebautes Verzeichnis der Unterdokumente einer Web Site.

jedes beliebige Aussehen zu verleihen, einfach indem eine entsprechende Grafik damit assoziiert wird. Die Ausprägung einer konkreten Anwendung bzw. zunächst einer individualisierten Entwicklungsumgebung im Hinblick auf das Look-And-Feel erfolgt aus technischer Sicht auf zwei Ebenen. Nach dem Whitebox-Prinzip erfolgt das Skinning zentraler Anwendungsteile (Fenster, Bedienelemente etc.) mit Mitteln (so genannte Bibliothekseinträge) der Client-Ablaufumgebung (Macromedia Flash). Die eigentlichen Inhalte werden über einen eigens geschaffenen dynamischen Styling-Mechanismus gestaltet, vergleichbar den CSS[3] für HTML.

Die Aufgabe der initialen Projektphase, einer Phase vorrangig technischer Entwicklung (siehe Abbildung 3.3), ist es also, durch eine erste Ausprägung aller dieser Rahmen eine prototypische Anwendung abzuleiten, die die Grundlage für experimentelles Arbeiten von Fachautoren und Redakteuren bildet. Mit der ersten Anwendung stehen entsprechende, an die individuellen Bedürfnisse der Projektkonstellation angepasste Entwicklungsschnittstellen zur Verfügung, die es den Autoren ermöglichen, Inhalte und Funktionalitäten einzufügen, zu strukturieren und zu arrangieren (Phase der Content-Entwicklung). Diese Schnittstellen werden systematisch diskursiv erarbeitet und in ggf. vielen Iterationen auf der Grundlage direkten Feedbacks verfeinert. Das Vorgehen ist in Abschnitt 3.2 im Detail beschrieben.

Entscheidend an dieser Idee des interaktiven Prototyping auf der Basis des Frameworks ist, dass die Autoren (also die nicht technikaffinen Projektbeteiligten) einerseits die Auswirkungen der Technikunterstützung ihrer Entwicklungstätigkeit direkt erfahren, an ihrer üblichen Arbeitsweise reflektieren und schließlich unmittelbar beeinflussen können. Andererseits können Autoren die Ergebnisse ihrer Arbeit als Entwicklungsbeiträge zum Zielsystem sofort sehen. Damit sind sie in der Lage, differenziert Feedback zu geben, sowohl bezüglich der Anlage der eigentlichen Zielanwendung als auch was ihre Arbeitsfähigkeit bei der Entwicklung betrifft. Das explorative, an der Tätigkeit der Entwickler und damit der Domäne orientierte Vorgehen unter ständiger Rückkopplung zwischen den Wünschen und Fähigkeiten der Fachleute auf der einen Seite und den Möglichkeiten der technischen Entwicklung auf der anderen führt gewissermaßen zwangsläufig zum Konsens unter den beteiligten Rollen. Welche gezielten Maßnahmen der Organisations- und Personalentwicklung sowie der Integration mit der Technologieentwicklung nötig werden, um diesen Prozess tatsächlich erfolgreich gestalten zu können, klären die Abschnitte zur Beschreibung der Vorgehensweise (Abschnitt 3.2) und zur Anwendung von Prinzipien des Wissenserzeugung (Abschnitt 3.3).

Die Ableitung konkreter Anwendungen besteht also zum einen im sukzessiven Ausfüllen der einzelnen Rahmen. Das Ergebnis dessen aus technischer Sicht ist eine vollständige Konfiguration des so genannten Renderers. Diese Konfiguration zur Steuerung des Renderers entspricht dem Prinzip der Meta-Architektur nachdem sich der Aufbau und der Ablauf eines Systems zur Laufzeit aus einer externen Beschreibung ableiten. Eine detailliertere Beschreibung dessen findet sich im Unterabschnitt 3.1.5. Zum anderen entsteht in gleicher Weise inkrementell ein Repository mit Inhalten und mit der Konfiguration für

[3] **C**ascading **S**tyle **S**heet beinhalten den Stil einzelner Teile von in HTML oder XML verfassten Web-Inhalten und separieren dadurch das Erscheinungsbild vom eigentlichen Inhalt.

die Gliederung der Anwendung und ihrer einzelnen dem Nutzer präsentierten Funktionalitäten. Die Ergebnisse der Entwicklung sind in Abbildung 3.3 durch den Renderer-Block und durch das Dokument mit dem Getriebe-Icon, symbolisch für die inhaltlich fachliche Konfiguration (das Repository), wiedergegeben.

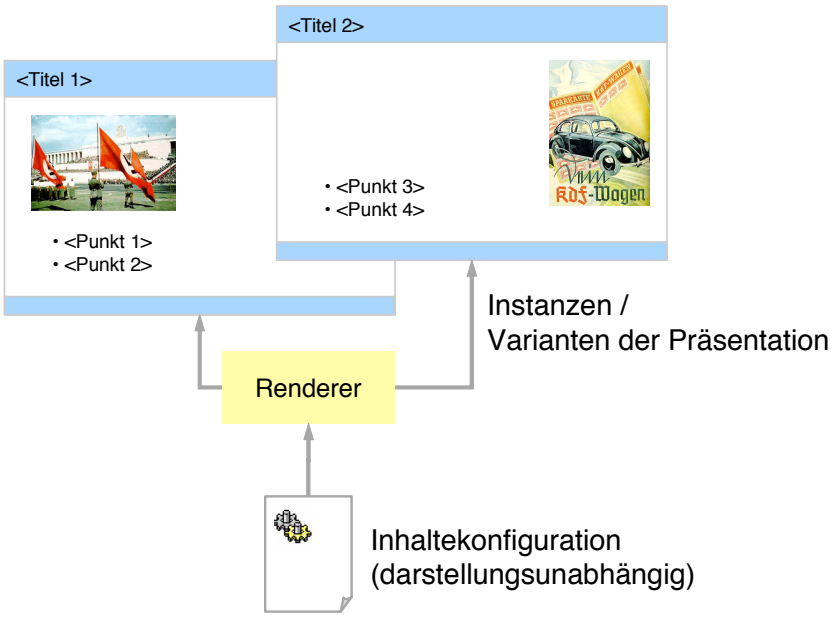


Abbildung 3.4.: Zusammenspiel der Ergebnisse der Ausprägung des Frameworks

Das Zusammenspiel dieser beiden Teile illustriert die Abbildung 3.4. Die eigentlichen Funktionalitäten sind im Renderer wie umrissen ausgeprägt. Das Repository und die darin enthaltene Konfiguration steuern die Funktionalitäten sowie deren Integration zum Gesamtsystem und die Inszenierung im Zusammenspiel mit den Inhalten. Diese Steuerung ist aus der Sicht der eigentliche Kontrollflüsse so zu verstehen, dass der Renderer Struktur-, Design-, Medien- und Ablaufinformationen aus dem Repository liest und in die Präsentation des Gesamtsystems umsetzt.

Ein Beispiel zur Wirkungsweise

Im Folgenden wird die Arbeitsweise der Frameworkanwendung, wie sie in Abbildung 3.4 dargestellt ist, ganz grundsätzlich illustriert. Der Prozess der Co-Adaption zwischen Technologieentwicklung und Fachdomäne wird dabei zunächst noch weitgehend ausgeblendet. Dieser Aspekt wird im Zuge der Beschreibung des Vorgehens separat betrachtet.

Als ein einfaches Beispiel soll eine Anwendung fungieren, mit deren Hilfe ein Filmausschnitt analysiert und interpretiert werden kann. Dabei soll eine Reihe von Schlüsselbildern des Films extrahiert und zu einer Art Storyboard angeordnet werden können. Jedes

der Bilder soll ferner zur Laufzeit des Systems mit einem interpretierenden Text versehen werden können. Eine solche Anwendung kann beispielsweise als interaktive Übung zur Filmanalyse als Teil eines Systems zur Vermittlung von Medienkompetenz zum tragen kommen. Dazu enthält der *funktionale Rahmen* eine Basiskomponente zur Anzeige von Filmen. Außerdem existiert ein einfacher Mechanismus, der es erlaubt beliebige durch den Nutzer verschiebbare (dragable) Bildschirmobjekte an einer Position einrasten zu lassen. Ein Textfeld als elementarstes Bedienelement ist ebenfalls ein disponibles Element des Frameworks.

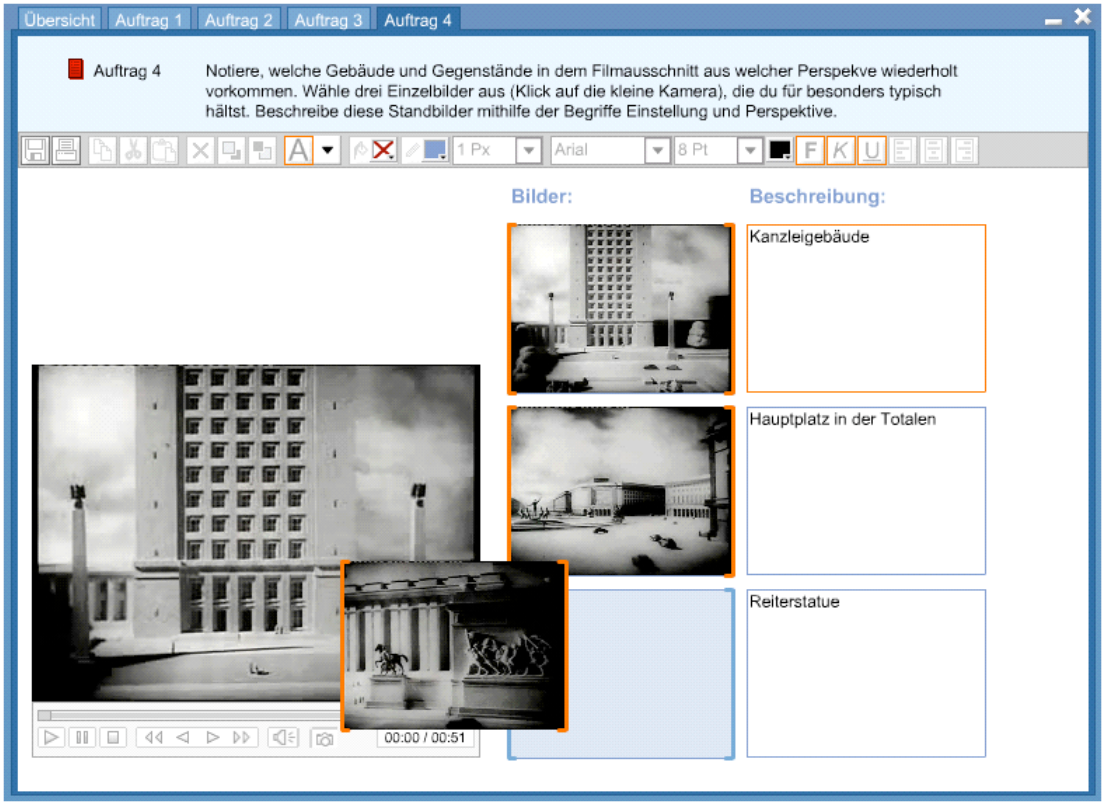


Abbildung 3.5.: Anwendung zur Analyse eines Filmausschnitts

Die Ergebnisse der initialen Phase der Entwicklung bestehen also in der Konkretisierung der Anwendung und der einzelnen verwendeten, funktionalen Elemente. Das bedeutet, dass der Rahmen für die Teilanwendung festgelegt wird, also an welcher Stelle des Ablaufs des Gesamtsystems sie sich eingliedert und damit auf welchen Pfaden der Navigation sie zu erreichen ist. Für das in Abbildung 3.5 gezeigte Beispiel ist charakteristisch, dass sich die Teilanwendung selbst in eine Reihe ähnlicher Anwendungen eingliedert – geordnete durch die Karteikartenmetapher. Es wird ferner festgelegt, wie sich das System beim Aufruf und Verlassen der Anwendung verhalten soll. So kann etwa beim Schließen der Anwendung der gängige Dialogmechanismus zum Wunsch nach persistenter Speicherung des letzten Zustandes der Anwendung ausgewählt werden. Dieser Mechanismus als

Teil des Infrastrukturrahmens kann auch standardmäßig für alle Teilanwendungen einer „Klasse“[4] von Anwendungen innerhalb des umgebenden Systems konfiguriert werden, also beispielsweise für alle interaktiven Übungen.

Die einzelnen innerhalb der Anwendung disponiblen Elemente werden im Falle des Beispiels vordergründig bezüglich ihres Erscheinungsbildes angepasst. Mit Hilfe des Skinning-Mechanismus wird unter anderem die Form und die Farbe der Bedienelemente des Mediaplayers festgelegt. Die Symbole der Tasten werden mit den vom Screendesign entworfenen Grafiken belegt. Das kann ebenfalls generell, etwa für alle Ausprägungen von Video-Playern im System, erfolgen oder individualisiert.

Ein Beispiel für die wirklich funktionale Erweiterung eines der Schablonenelemente ist die Erweiterung des Players um eine Einrichtung zur Erzeugung eines Standbildes. Diese sehr spezielle neue Funktionalität muss nach dem White-Box-Prinzip, direkt im Code des Frameworks realisiert werden. Die stärke des Frameworks besteht darin, diese Aufgabe durch die Bereitstellung klarer Programmierschnittstellen bestmöglich zu unterstützen und eine saubere Trennung zwischen allgemeingültigen Teilen des Frameworks und den anwendungsspezifischen Ergänzungen zu gewährleisten. Mittel der Objektorientierung, wie Mechanismen der Vererbung innerhalb von Klassenhierarchien und darauf aufbauend der Einsatz von Entwurfs-Mustern werden entsprechend genutzt.

Neben der Ableitung einer ersten prototypischen Rahmenanwendung für den Einsatzkontext werden die Entwicklerzugänge für die Autoren angelegt. Diese bestimmen die Art und Weise, wie die Autoren mit Primitiven wie Textfenstern verfahren, mit Hilfe welcher Metaphorik sie in der Bildschirm-Arbeit tatsächlich disponieren. So sollen sie beispielsweise einen Film (so bezeichnet und entsprechend grafisch repräsentiert) als Abstraktes Objekt wählen und in der Anwendung arrangieren, ohne einen Player-Komponente konfigurieren und mit dem Medium einer Mpeg-Video-Datei initialisieren zu müssen.

Auf der Grundlage dieser Projektspezifischen Entwicklungsschnittstellen wird schließlich in der zweiten Phase einer Entwicklungs-Iteration die Anwendung aus rein fachlicher Sicht entwickelt. Das geschieht so im Idealfall weitgehend unter Verwendung von am Fach orientierten Termini und Tätigkeiten. So wird der *Film* aus einem *Fundus* ausgewählt und dem *Arbeitsblatt* zugeordnet. Das kann, wie in der exemplarischen Entwicklung des Evaluationsprojekts, auf zwei Wegen erfolgen: zum einen durch das Auswählen und Einfügen im WYSIWYG[5]-Modus. Zum anderen kann die Gestaltung der Anwendung auch über eine explizite Autorenschnittstelle, wie dem im Evaluationsprojekt umgesetzten *LeMOLenen-Wizzard* erfolgen. Diese Loslösung von der eigentlichen Präsentation der Zielanwendung kann gerade bei inhaltlich umfangreichen Vorhaben eine Fokussierung auf rein fachliche Aspekte der Arbeit erleichtern, einfach durch das konsequente Abstrahieren von technischen Aspekten. So können Filme in großer Zahl

[4] Der Begriff der *„Klasse“* ist hier nicht im Sinne der Objektorientierung zu verstehen.

[5] **W**hat **y**ou **s**ee **i**s **w**hat **y**ou **g**et. Prinzip nach dem die Bearbeitungsansicht eines Dokuments oder einer grafischen Nutzeroberfläche der endgültigen Ansicht des Ergebnisses entspricht.

recherchiert und thematisch gegliedert werden, ohne mit der Modularisierung in einzelne Unteranwendungen in Berührung kommen zu müssen.

Einfachste Elemente, wie Textfelder und die entsprechenden vorgegeben Texte oder die so genannten Drop-Zones, die Felder als Ziel-Anker für die Objekte von Zuordnungsübungen, werden in der Regel in der direkten Ergebnisansicht eingefügt und inszeniert.

In der ersten Iteration des Projektablaufs wird die Phase der Content-Entwicklung noch spielerischen Gehversuchen entsprechen. Intensive Analysen der Arbeitsweise und der erzielten Ergebnisse fließen zurück in eine erneute Phase der Konkretisierung der technologischen Grundlage. Erkenntnisse über Defizite der Werkzeuge in der Handhabung durch die Fachautoren und fehlende und fehlerhafte Funktionalitäten sowie unbefriedigende Gestaltungsaspekte der Oberflächen der Zielanwendung münden als direkte Rückkopplung in die Überarbeitung der Ausprägung des Frameworks. So entsteht systematisch und ggf. in mehreren Durchläufen eine konsolidierte Arbeitsgrundlage für die Phase der fachlichen Gestaltung.

Der Aufbau und die Arbeitsweise der einzelnen Teile des Frameworks werden in den folgenden Unterabschnitten im Detail beschrieben. Die Gliederung in einen als Domänen-Framework und einen als Anwendungs-Framework bezeichneten Teil entspricht dem Versuch, die zum Teil der Komplexität der Gesamtzielsetzung geschuldet Vielschichtigkeit des Frameworks entlang gängiger Kategorien zu strukturieren. Innerhalb der aktuellen Ausbaustufe der technischen Lösung ist diese klare Trennung zwischen einem domänenspezifischen Teil (der Domänenbegriff in diesem Zusammenhang wird Eingangs des kommenden Abschnitts präzisiert) und einem rein querschnittlichen, infrastrukturellen zumindest nicht so konsequent ausgeprägt, als das es möglich wäre, beide Teile losgelöst voneinander einzusetzen.

3.1.2. Das Domänen-Framework

Der Begriff der Domäne, als abgrenzbares Problemfeld für die Zielstellungen einer angestrebten Softwareunterstützung, muss an dieser Stelle für die verschiedenen Ebenen der Framework-Nutzung, also die unterschiedlichen Stadien der Ausprägung, differenziert betrachtet werden. Ganz intuitiv soll dabei jeder Abstraktionsstufe der Entwicklung jeweils eine eigene Definition eines sich ergebenden Gestaltungsspielraums also eine Abstraktionsstufe des Domänenverständnisses zugeordnet werden.

Hinterlegt man die in Abbildung 3.6 wiedergegebene Struktur des Diskursbereichs mit den Schritten der Anwendung des Frameworks aus Abbildung 3.2 bedeutet das, dass das „FLAME“-Framework im Ausgangszustand die komplette Domäne der interaktiven, direkt reaktiven Systeme adressiert. Nach der initialen Ausprägung des Frameworks erhält man einen fachlich spezialisierten Rahmen, der zumindest für den Fachbereich noch immer weitgehende Allgemeingültigkeit besitzt und damit noch immer die Grundlage für eine ganze Klasse von Anwendungen des somit engeren Diskursbereichs sein kann. Erst

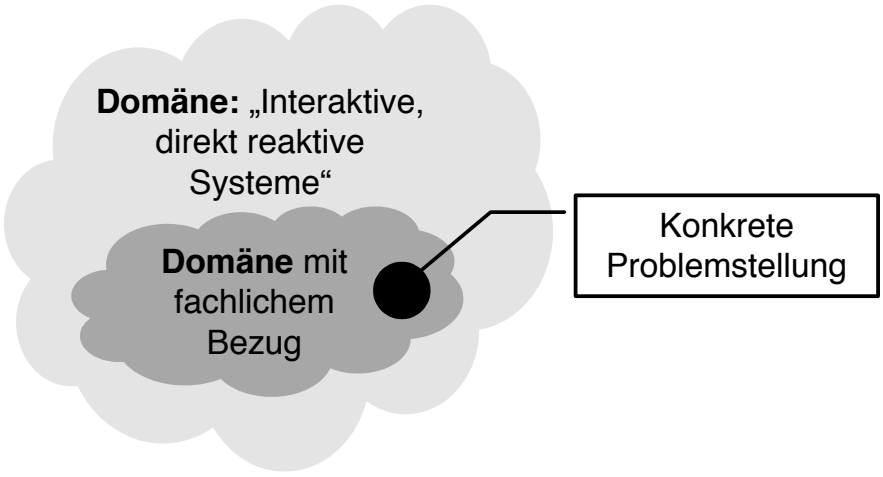


Abbildung 3.6.: Zweistufiges Domänenverständnis des Frameworks

durch den Schritt der fachlich, inhaltlichen Konkretisierung – etwa durch die Fachautoren – entsteht die eigentliche Zielanwendung als Beitrag zur Lösung der partikulären Problemstellung des Anwendungskontexts.

Illustriert durch ein Beispiel bedeutet dies, dass die allgemeinste Konfiguration des Frameworks, also sein Urzustand, bezüglich der Domäne der interaktiven, direkt reaktiven Systeme insofern spezialisiert ist, als dass:

- Die Möglichkeiten die Architektur auszulegen auf die Präsentation großer Mengen medialer Inhalte ausgerichtet ist und Aspekte wie beispielsweise Transaktionalität weniger eine Rolle spielen.
- Die Wahl der technologischen Grundlagen (Macromedia Flash, HTTP, SOAP, etc.) auf diesen Anwendungsbereich fokussieren.
- Das zu Grunde liegende Datenmodell zur darstellungsneutralen Strukturierung der Inhalte des Systems, sich stark auf den Aspekt der Präsentation und der Nutzerinteraktion ausrichtet.
- Viele der angebotenen Komponenten die Domäne direkt bedienen.

Diese Einschränkungen legen damit etwa einen Einsatz zur Erstellung betrieblicher Informationssysteme oder gar von Echtzeit-Steuerungs-Software eher nicht nahe.

Auf der Grundlage dieser allgemeinsten Konfiguration wird nach dem ersten Schritt der Ausprägung beispielsweise der Rahmen für die Entwicklung von CD-ROM-basierten Anwendungen zur Vermittlung historischer Fakten an Schüler stehen – wie im Projekt, das für die Evaluation des Ansatzes herangezogen wird. Die Entwicklerschnittstellen sind dann weitgehend auf die Arbeitsweise und die Begriffswelt von Historikern abgestimmt und die Navigationselemente und Interaktionsmuster entsprechend angelegt. Historische Zeitleisten und virtuelle historische Räume und deren Verhalten sind beispielsweise

grundsätzlich angelegt. Sie müssen für den letzten Schritt in Richtung der eigentlichen Anwendung dann lediglich noch mit den jeweiligen medialen Elementen ausgestattet und ggf. in der Handhabung verfeinert werden.

So können je nach inhaltlicher Ausprägung die verschiedensten Systeme zur Anwendung innerhalb der fachlich konkretisierten Domäne „Vermittlung historischer Fakten“ erstellt werden. Allein die Variationen über die verschiedenen Zielgruppen und die damit verbundene didaktische Diversität ergeben ein potenziell sehr großes Feld für die Wiederverwendung auch des bereits sehr spezifischen Entwicklungszwischenschritts.

Entsprechend zweistufig gegliedert ist der Aufbau des Anteils des „FLAME“-Frameworks, der domänenspezifische Software-Komponenten bereithält. Es gibt also zunächst einen Domänen-Framework-Kern, der wie beschrieben lediglich insofern Domänenspezifisch ist, als dass er sich auf die Belange interaktiver, direkt reaktiver Systeme konzentriert und entsprechende Vorgaben bezüglich der Architektur macht sowie in diesem Kontext immer wieder benötigte Funktionalitäten anbietet.

Es besteht darüber hinaus die Möglichkeit, die Zwischenschritte auf dem Weg zur konkreten Anwendung – und dazu gehören insbesondere die Anteile der Entwicklung fachlich individueller Entwicklerzugänge – als fachlich orientiertes Domänenframework einzufrieren, als Grundlage für fachlich verwandte Entwicklungen in vergleichbaren Projektkonstellationen.

Für das hier vorgeschlagene Konzept, welches den Brückenschlag zwischen den projektbeteiligten Disziplinen in den Mittelpunkt stellt, ist jedoch die kooperative Erarbeitung eben dieses für die Einbeziehung von Vertretern der Fachdomäne notwendigen Zwischenergebnises von entscheidendem Interesse. Die potentielle wirtschaftliche Effektivitätssteigerung durch den wiederholten Einsatz dieser Ergebnisse spielt dabei zunächst eine untergeordnete Rolle. So wird im Folgenden die Funktionsweise der Framework Kernkomponenten beschrieben und die Art und Weise, wie sie für die Erstellung fachlich vorgeprägter oder auch anwendungsspezifischer Komponenten eingesetzt werden. Die unterstützenden Auswirkungen der Wiederverwendung aus Sicht der fortschreitenden Adaption zwischen den Herangehensweisen der Disziplinen und der Technologieunterstützung werden dagegen als Teil des methodischen Konzepts im Abschnitt 3.2 erörtert.

Navigationsrahmen

Auf Grund der Eigenschaft interaktiver, direkt reaktiver Systeme umfangreiche mediale Inhalte zu präsentieren, und dem Nutzer vielfältige Funktionalitäten der Interaktion zu bieten, werden Konzepte und schließlich auch technische Einrichtungen der Navigation benötigt. Diese ermöglichen dem Nutzer die Orientierung innerhalb des Systems. Gerade in komplexen Systemen repräsentieren sie Strukturen, die der Ordnung und damit der Beherrschung der fachlich inhaltlichen Komplexität dienen. Mit Hilfe (ggf. abstrahierender oder filternder) Sichten auf das Gesamtsystem wird der Nutzer in die Lage

versetzt, die ihn interessierenden Funktionalitäten bzw. Inhalte zu finden und abzurufen. Sie erlauben aber etwa auch eine gezielte, redaktionell vorgegebene Führung durch den Systemablauf, die beispielsweise didaktischen Intensionen folgen kann. Navigation ist eine definierende Eigenschaft der hier betrachteten Systeme – in Abgrenzung etwa zu betrieblichen Informationssystemen. Eine eingehende Beschreibung der Anforderung „Navigation“ ist in Abschnitt 2.1.1 auf Seite 18 zu finden.

Das technologische Basisprinzip zur Realisierung der Navigation, also der elementarste vorgegebene Kontrollfluss des Navigationsrahmens, gestaltet sich folgendermaßen: Ausgangspunkt ist ein Knoten der Anwendung in Anlehnung an das Verständnis gängiger Hypermedia-Systeme (siehe etwa [SRB96]). Ein solcher Knoten ist eine eigenständige Einheit der Präsentation von Inhalten und Anwendungsfunktionalität, die insofern abgeschlossen ist, als dass sie durch eine dezidierte Konfiguration ausgeprägt wird. Das kann beispielsweise einer Bildschirmseite, einem modalen Dialog aber auch einem Panel innerhalb eines komplexen Knotens entsprechen. Ein Knoten ist die kleinste Einheit dessen, was der Renderer zur Anzeige bringen kann. Navigation von einem Knoten zu einem anderen vollzieht sich über so genannte Links. Der Navigationsrahmen gibt dazu einen Mechanismus vor, der die Interaktion des Nutzers mit einem Bedienelement innerhalb eines Knotens, also etwa die Anwahl eines Menüpunktes durch Mausklick, in den Aufruf der Anzeige des Zielknotens münden lässt. Das Framework stellt also einen Standard-Handler für die Ereignisbehandlung eines jeden Bedienelements bereit, der, konfiguriert mit der Zieladresse, das Laden der Inhalte und Funktionen des Zielknotens veranlasst. Eine Konkretisierung dieses sehr grundlegenden Prinzips für die betrachtet Domäne interaktiver, direkt reaktiver Systeme zur Einschränkung der Komplexität zugunsten der Adaptierbarkeit für die multidisziplinäre Entwicklung wird im Folgenden dargestellt.

Der Beobachtung nach ist allen interaktiven Anwendungen ein Aufbau gemeinsam, wie er in Abbildung 3.7 dargestellt ist. Selbst Websites trennen grundsätzlich zwischen einem mehr oder minder statischen Rahmen mit global verfügbaren Elementen der Navigation zu immer wieder unteranwendungsübergreifend benötigten Funktionalitäten bzw. Inhalten auf der einen Seite und dem Bereich für die Präsentation der eigentlichen fachlichen Inhalte und Funktionalitäten auf der anderen Seite.

Das hier vorgestellte Framework gliedert Anwendungen a priori, nicht jedoch zwingend, dementsprechend. Auch der grafische Aufbau der Unterteilung in einen Anwendungsbereich, der am oberen und am unteren Rand von rechteckigen Leisten für die Rahmenfunktionalitäten begrenzt wird (wie in Abbildung 3.7), ist lediglich dem weit verbreiteten Muster geschuldet, nach dem ein Hauptteil, ein Kopf- und ein Fußbereich die zentrale Ansicht einer Anwendung gliedern. Dieser grobe Aufbau ist für die Nutzung von fensterbasierten Standardanwendungen genauso erwartungskonform wie für Websites. Entsprechend gestaltet ist die initiale Ausprägung des Frameworks etwa für den Einsatz im Evaluationsprojekt. Nimmt man diese Ausprägung als Grundlage für die Ableitung ähnlich zu gliedernder Ziel-Systeme, ist es durch einfache Schritte der Konfiguration möglich, das Erscheinungsbild, etwa die farbliche Gestaltung oder die Dimensionierung anzupassen. Die für die Domäne der hier betrachteten Systeme benötigte

volle Flexibilität schafft das Framework, indem über definierte White-box-Schnittstellen des Renderers (direkt auf Ebene des Quelltexts), jeder beliebige Aufbau umgesetzt werden kann – globale Funktionalitäten beliebig platziert oder gänzlich weggelassen werden können.

Abbildung 3.7.: Schematischer Aufbau der Hauptansicht einer typischen Anwendung

Welche Funktionalitäten der Navigation, repräsentiert durch entsprechende Bedienelemente zu deren Aufruf, im globalen, statischen Steuerungsrahmen der Anwendung typischerweise gruppiert werden, ist in der Abbildung 3.8 zusammengestellt. Die in einer Art Explosionsschema dargestellten Bedienelemente sind in der Regel Schaltflächen oder Buttons, die die jeweilige Funktion auslösen. Lediglich die Angabe der aktuellen inhaltlich thematischen „Position“ wird in der Regel ein Textfeld bzw. sensitives Label sein. Die Bedienelemente sind sinnvollerweise über den gesamten Ablauf einer Anwendung hinweg statisch an derselben Position im Rahmen verfügbar.

Der Raum, durch den die Navigation führt, ist in der Regel bestimmt durch die fachlich inhaltlichen, potenziell beliebig ordnenden Strukturen – im Gegensatz etwa zu einer pauschalen Aufgabenorientierung des Aufbaus der Nutzeroberflächen, wie man sie in betrieblichen Anwendungen findet. Das gängige Navigationsmuster *Hypertext* beispielsweise verquickt üblicherweise Informationen eines Kontexts. Betrachtet man typische Vertreter interaktiver, direkt reaktiver Systeme, lässt sich darüber hinaus verallgemeinern, das die übliche auf den Inhalten definierte Ordnung, Funktionalitäten zum Vorwärts- und Rückwärts-„Blättern“ zwischen aufeinander folgenden Teilanwendungen sinnvoll und in vielen Fällen nötig macht. Wenn etwa zwei Teilanwendungen eindeutig aufeinander Folgen bezüglich des Einsatzes bzw. der Rezeption. Im Bereich des elektronisch unterstützten Lernens beispielsweise bauen einzelne digitale Lerneinheiten aufeinander auf. Das bedeutet, dass der Gegenstand der Wissens- bzw. Kompetenzvermittlung einer

Anwendung notwendige Voraussetzung für das Verständnis und die Anwendbarkeit der darauf folgenden ist.

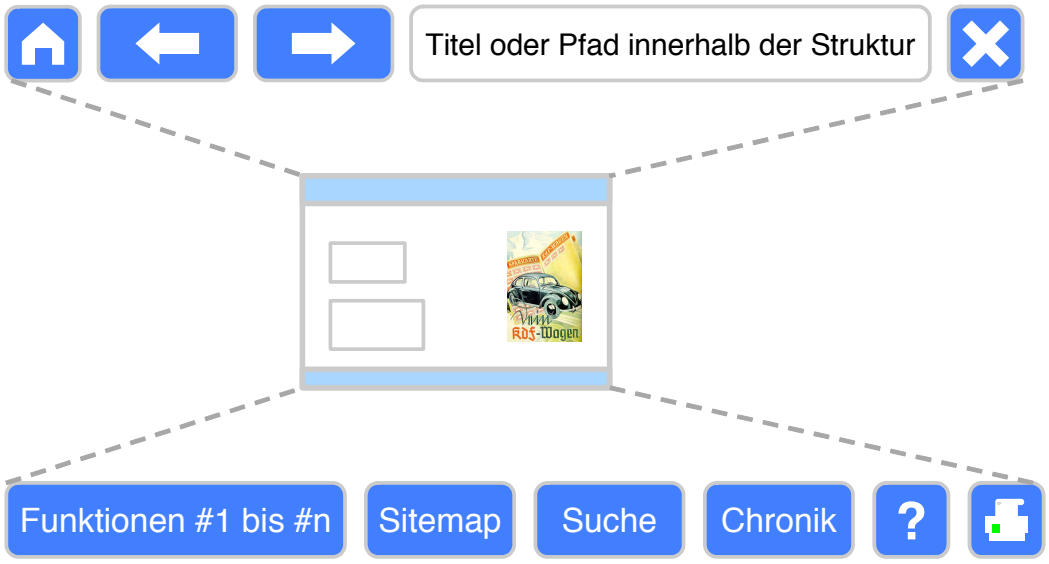


Abbildung 3.8.: Navigationselemente, die im Ablauf der Anwendung global zur Verfügung stehen

Auch auf Grund der Tatsache, dass dieses Navigationsprinzip, materialisiert durch den Browser, integraler Bestandteil jeder Webanwendung ist und damit bezüglich der Bedienung medienorientierter Software von einem gewissen Erwartungspotential seitens der Anwender ausgegangen werden muss, wurde ein entsprechender Mechanismus fest in den Kern des Navigationsrahmens des FLAME-Frameworks aufgenommen. Das impliziert auch in gewisser Weise die Übernahme eines seitenorientierten Aufbaus der Kern-Anwendung. Das FLAME-Konzept erweitert das Verständnis der Präsentation einzelner Seiten dahingehend, dass eine FLAME-Anwendung aus mehreren abgeschlossenen, mehr oder minder komplexen Einzelanwendungen besteht, die jeweils durch eine eigene XML-repräsentierbare Konfiguration im Repository definiert werden. Dabei ist die Umsetzung der Konfiguration in die Präsentation also die einzelne Anwendung eben nicht statisch und kanonisch wie bei HTML, sondern abhängig von der kontextspezifischen Ausprägung des Renderers (siehe auch 3.1.1).

Das heißt also, dass die Nutzung des „Blättern“-Mechanismus, die Vor- und Zurückbewegung innerhalb einer Abfolge abgeschlossener Teilanwendungen steuert. Die Nutzerinteraktion erfolgt üblicherweise mit Hilfe von Schaltflächen, die durch Pfeile rechts und links gekennzeichnet sind. Aus der sich ergebenden Sequentialität der Aufrufe lässt sich ein weiteres Navigationsmuster ableiten, welches ebenfalls die gängigsten Webbrowsern durch entsprechende Features unterstützen. Was in den Web-Browser die Orientierung an bereits besuchten Seiten erlaubt, in dem Sinne dass man durch Assoziationen zu unlängst gesehenen Inhalten Informationen *wieder findet*, hilft im Rahmen der hier

adressierten Anwendungen darüber hinaus nachzuvollziehen, welche Anwendungen man bereits bearbeitet, also welchen Lernpfad man beispielsweise bereits beschritten hat. Die entsprechende Funktion – in Abbildung 3.8 ist der auslösende Button mit „Chronik“ bezeichnet – handelt es sich dabei um ein Merkmal, welches klar zur Navigationsfunktionalität des Rahmens zählt.

Die Funktionalität an sich gestaltet sich aus der Sicht des Nutzers als ein separater Ausgabebereich in dem die bereits besuchten Programmbereiche aufgelistet werden. Die Voreinstellung des Frameworks etwa führt zum öffnen eines Fensters. Die einzelnen Listeneinträge beinhalten in der Regel einen textuellen Titel sowie ggf. ein Symbol, was inhaltliche Assoziationen unterstützt. Die Anwahl eines der Einträge, beispielsweise durch einen einfachen oder je nach Konfiguration doppelten Mausklick, ruft den damit verknüpften Programmbereich erneut auf. Sowohl die Fixierung auf ein Fenster als separierendes Element für Die Funktionalität als auch die Darstellung als vertikal fortlaufende Liste entspricht einem nahe liegenden Interaktionsmuster für die Funktionalität, ist jedoch im Zuge einer initialen Ausprägung eines Systems disponibel. Gruppierungen, etwa je Zeitintervall (im Sinne von „*Gestern* genutzte Anwendungen“ usw.) oder auch entlang der Medientypen, können relativ einfach nach dem White-Box-Prinzip ergänzt werden.

Komplett neue Formen der Interaktion können auch hier zumindest über definierte Schnittstellen auf die entsprechenden Historie-Informationen zugreifen um sie als Grundlage einer beliebigen Visualisierung nutzen zu können. Auch die Einbindung in die Kontrollstrukturen der Gesamtanwendung, die beispielsweise die Reaktionen auf Nutzeraktivitäten zentral registrieren, weiterleiten und schließlich für eine Reaktion bearbeiten, erfolgt nach den Regeln des Frameworks und über entsprechend definierte Schnittstellen und Aufrufregime.

In diese Reihe der Optionen der grundlegendsten Navigationsmechanismen, die sich in den permanent zugreifbaren Rahmen einfügen, gehört zweifelsohne die „Home“-Funktion. Die verbirgt sich hinter dem mit dem Häuschen markierten Knopf in der Illustration 3.8. Sie führt den Nutzer zu einem festen definierten funktionalen bzw. inhaltlichen „Anlaufpunkt“ innerhalb der Gesamtanwendung, etwa zu einem Überblick über das System und seine Möglichkeiten. Diese „Einsprungadresse“ ist frei festlegbar. Eine Festlegung, die zur Nutzungszeit nicht zu ändern ist ebenfalls möglich.

Ein ebenfalls aus der Welt der Web-Anwendungen übernommener Navigationsmechanismus ist die Sitemap. Während Websites mit Hilfe dieser speziellen Ansicht einen Überblick über die meist hierarchische Struktur geben, in der sich die einzelnen Unterdokumente und deren Beziehungen widerspiegeln, bietet das Framework die Grundlage zur Ausprägung beliebig aufgebauter Ansichten der Metanavigation. Ein solcher Inhalteüberblick repräsentiert Inhalte und Anwendungsteile potentiell gegliedert gemäß beliebiger Ordnungsprinzipien und der didaktischen Intension. Er schafft den Überblick über komplexe vernetzte Inhalte, indem er Informationen aggregiert, filtert und abstrahiert. Die Interaktivität der Navigation gestaltet sich dabei nach einem ähnlichen Muster, wie bei der „Chronik“-Ansicht: das Element des Interesses wird innerhalb der

Ansicht (ggf. separates Fenster) angewählt und damit der entsprechende Navigationsschritt ausgeführt – das Element geladen und in den Vordergrund gebracht. Dies Erfolgt unter Umständen zweistufig. Nach einer ersten Anwahl und dem damit verbunden Setzen eines Fokus' auf ein Element, erscheint zunächst eine Zusammenfassung zum zu erwartenden Inhalt bzw. der zu nutzenden Funktionalität. Erst nach einem bestätigenden zweiten Mausklick etwa wird der Navigationsschritt ausgeführt.

Das Framework bietet dafür einen offenen Mechanismus mit eigener darstellungsunabhängiger interner Repräsentation für den Austausch oder auch die persistente Speicherung der Übersichtsinformationen. Es gibt ferner Werkzeuge, die es erlauben, einen Teil dieser Metainformationen automatisiert aus den eigentlichen Inhalten zu extrahierenden. Neben der voreingestellten Möglichkeit der Abbildung einer einfachen fachsystematischen Gliederung, innerhalb der sich in beliebig tiefer Schachtelung Unterthemen einem Oberthemen unterordnen, ist es möglich, beliebig hypervernetzte Inhalte und Systemteile im Überblick wiederzugeben. Die technische Repräsentation der Struktur und deren Abbildung in Konstrukte der Nutzerinteraktion bzw. der Navigation auf der Basis des *Reflection*-Architekturmusters ist in Abschnitt 3.1.5 auf den Seiten 169 ff. ausführlicher beschrieben. Die Ausprägung der entsprechenden „View“[6] ist dabei auf White-Box-Ebene im Renderer nötig. Die Integration in den Rahmen der Gesamtanwendung, etwa zur Bereitstellung der nötigen Daten und die Einbindung in den Nachrichtenfluss zur Behandlung der Nutzerinteraktion erfolgt auch hier über definierte Schnittstellen.

Ein Suchmechanismus für die gezielte fachliche Filterung benötigter Inhalte und Funktionen ist ebenfalls fester Bestandteil des Navigationsrahmens und in der Ausprägung üblicherweise repräsentiert und anwählbar über einen Button im permanent verfügbaren Rahmen der Anwendung. Die eigentliche Funktionalität wird auch hier innerhalb eines eigenen Ausgabebereichs, also etwa einem modalen Fenster, präsentiert. Als elementarste Ausprägung einer Suchanwendung ist eine Volltextsuche über allen textuellen Beschreibungen von Inhalten und Funktionalitäten sehr einfach zu realisieren. Die Menge der gefundenen Ergebnisse kann dabei wie die Einträge der Sitemap oder auch der Chronik gestaltet werden. Auch hierbei können Gruppierungen etwa anhand von Aggregationen über Metadateneinträgen zum Medientyp vorgenommen und entsprechend präsentiert werden.

Die Erweiterung der Suche um eine attributgesteuerte Selektion wird ebenfalls direkt durch das Framework unterstützt. So kann beispielsweise eine Sucheanfrage spezifiziert werden, die lediglich diejenigen Inhalte und Teilanwendungen als Ergebnis liefert, die die Eigenschaft besitzen ein bestimmtes *Schlagwort* als *Teil* (Prädikat) ihres *Titels* (Attribut) zu besitzen. Es ist möglich die Attributmenge beliebig zu erweitern. Die für eine solche Adaption nötigen Schritte sind: die Erweiterung des Eingabeformulars für die Spezifaktion der Anfrage, die Anpassung der Überführung der Nutzerangaben (Zustände der Datenstrukturen des Renderers) in ein Anfrageobjekt zur Auslösung des entsprechenden Services, die Erweiterung der Indexierungsprozedur (diese ist im folgenden Abschnitt näher erläutert) sowie die Anpassung der Anfrage-Bearbeitung, also der Ergebnisgenerierung und Anzeige. Aus der Sicht der Nutzung des Frameworks bedeutet dies wie-

[6] Im Sinne des Model-View-Control-Prinzips zu verstehen

derum eine Konkretisierung bestehender Klassen und Schnittstellen (Interfaces) – in der Regel durch die Ergänzung um benötigte Attribute. Eine ganz elementare Klasse „Result“, die im Wesentlichen den Konvention von JavaBeans™[Mic97] entspricht und damit auch die für den Transfer nötig Fähigkeit zur Serialisierbarkeit besitzt, nimmt die Ergebnisse für den Austausch über die SOAP-Schnittstelle auf. Der Ablauf für die Initialisierung der Klassen zu Objekten, die entsprechende Speicherverwaltung sowie die für den Austausch der Informationen (Anfragen, Ergebnisse, etc.) zwischen Renderer und Infrastruktur- bzw. Anwendungsschicht benötigten Mechanismen, sind durch das Framework vorgegeben. Der Aufwand der Instanziierung der Suchefunktion beschränkt sich also auf die rein fachliche Anpassung des „Containers“ für die zu verarbeitenden Inhalte.

Als ein weiterer Fixpunkt für die Orientierung des Nutzers kann ein permanent verfügbarerer Hinweis auf seine aktuelle Position innerhalb der Anwendung gelten. Was zunächst nahe liegend und trivial erscheint muss dennoch als potentielle Funktionalität des Navigationsrahmens im Framework verankert werden. Die zur Unterstützung der Ausprägung konkreter Anwendungen angebotene Funktionalität erlaubt es, Metadaten des aktuell durch den Nutzer geöffneten Anwendungsteils auszulesen und im globalen Rahmen zu visualisieren. So kann beispielsweise, wie es etwa in fensterbasierten Systemen üblich ist, einfach ein Titel im Kopf der Gesamtanwendung eingeblendet werden. Es ist beispielsweise auch denkbar die Gestaltung des Rahmens komplett in Abhängigkeit vom Typ der momentan genutzten Funktion dynamisch umzugestalten – die Wichtigkeit einer Übungsaufgabe etwa durch Rotfärbung des Rahmens hervorzuheben. Eine sinnvolle interaktiv funktionale Lösung auf der Grundlage dieser Konstruktion ist auch die Wiedergabe des fachlichen Kontexts der aktuellen Anwendung. So könnte etwa der Pfad, der durch übergeordnete Themen zur aktuellen Position führt, angegeben werden. Die Möglichkeit auch dieses Steuerelement interaktiv zu gestalten und für die Navigation nutzbar zu machen ist gegeben. Auch die in diesem Bereich angezeigten Elemente können mit Handlern für die Bearbeitung von Mouse-Aktionen hinterlegt werden. Auf diese Weise wird es beispielsweise Möglich, die Knoten des eben beschriebenen Kontextpfades anwählbar zu machen, um die zugehörigen Anwendungsteile, etwa das direkt übergeordnete Thema, zu aktivieren.

Weitere grundlegende Funktionen, die im weitesten Sinne navigatorischen Charakter haben und üblicherweise im globalen Rahmen der Gesamtanwendung Platz finden sind der Zugang zu Hilfefunktionen (Button mit dem Fragezeichensymbol in Abbildung 3.8), die sowohl ein komplexes eigenständiges System, entwickelt auf der Grundlage des Frameworks sein kann als auch einfach ein Verweis auf externe Dokumente. Die Kontextinformationen zur aktuell genutzten Teilanwendung können grundsätzlich dazu genutzt werden, um die Hilfestellung dezidiert zu den entsprechend benötigten Funktionalitäten anzubieten.

In vielen der Produkte, die der Klasse der hier adressierten Systeme angehören, ist darüber hinaus eine eigenständige zentrale Komponente zur Begriffserklärung wichtig, um dem Nutzer die fachliche Beherrschung des Systems und insbesondere der Inhalte zu erleichtern. Eine entsprechende Komponente enthält das Framework. Auf Grund

der Tatsache, dass es sich dabei um eine relativ konkrete Funktionalität, etwa für die Ausprägung von Lernanwendungen handelt, ist ein entsprechender Button nicht in die Abbildung 3.8 aufgenommen worden. Dieses so genannte „Glossar“ kann vielmehr als ein Beispiel für diejenigen Funktionalitäten gelten, die generisch in den Rahmen eingebunden werden können. Die entsprechenden „Slots“ sind in derselben Abbildung durch die „Funktionen #1 – #n“ symbolisiert.

Zu guter Letzt und der Vollständigkeit halber sei noch auf den „Verlassen-Button“ hingewiesen. In Grafik 3.8 ist dieser im Stile der gängigen Symbolik zum Schließen von Fenstern mit einem Kreuz gekennzeichnet. Er erlaubt im weitesten Sinne die Navigation zum Verlassen des Systems. Gerade die Anforderung, dass viele der interaktiven, direkt reaktiven Systeme im Vollbildmodus laufen, um den Fokus nachhaltig auf die Inhalte zu lenken und eigene, von den Bedienmodi des umgebende Betriebssystems unabhängige Interaktionsformen zu etablieren, ist es wichtig den „Ausgang“ immer klar zu kennzeichnen. Insbesondere auch, die durch den Einsatz von Macromedia Flash erreichte Plattformunabhängigkeit der Clientanwendung lässt kaum Annahmen über andere allgemein durch das Betriebssystem initiierten Mechanismus des Schließens der Anwendung zu – etwa durch das Drücken der Tasten Alt und F4 unter dem Betriebssystem Windows der Firma Microsoft. Das Framework unterlegt diesen „Verlassen-Request“ standardmäßig mit einer Funktionalität, die einen aus Nutzersicht Erwartungskonformen Ablauf des Beendens der Anwendung ermöglicht. Neben der Nachfrage, ob eigene, noch nicht persistent gespeicherte Arbeitsergebnisse des Nutzers noch abgespeichert werden sollen, kann ein expliziter Dialog zwischengeschaltet werden, der sicherstellt, dass der Nutzer überhaupt intendiert die Software zu schließen.

Die weniger der Navigation zuzurechnende und aus diesem Grunde an dieser Stelle nicht erörterte Druckfunktion als letztes Schaltelement im Schema der Rahmennavigation wird im Abschnitt 3.1.3 als querschnittliche, weitgehend domänenunabhängige Funktion beschrieben.

Abbildung 3.7 auf Seite 139 verweist mit der Andeutung von Menüelementen, Schaltflächen und Hyperlinks im zentralen Anwendungsbereich darauf, dass die Navigation nicht auf die Metaebene des Rahmens der Anwendung beschränkt ist. Auch Interaktionselemente im Kontext der eigentlichen Inhalte bzw. Teilanwendungen erfüllen Aufgaben der Navigation. Das heißt, dass neben den Bedienelementen zur Manipulation der aktuell bearbeiteten bzw. betrachteten Daten ein Großteil der Funktionalität der Anwendungen dafür verantwortlich ist, Verbindungen zu anderen Systemteilen und Inhalten herzustellen. Auf möglichst ergonomische Weise sollen so Kontexte bzw. Sichten geschaffen werden, die die Rezeption im Sinne der Erkundung von Informations- und Interaktionsräumen ermöglichen – unabhängig von den kanonischen fachsystematischen Ordnungsprinzipien der Domäne. Das Framework unterstützt diese Anforderung dadurch, dass *jedes* Ereignis (*Event*), also etwa ein Mausklick auf eine Schaltfläche, anstatt einer Funktionalität wie eingangs erwähnt auch mit einem Verweis hinterlegt werden kann.

Diese grundlegende, immer wiederkehrende Anforderung nach der Ausprägung von Navigationsstrukturen durch die potenziell beliebige hypermediale Vernetzung, erfüllt das

Framework entsprechend durch einen aus der Sicht der Framework-Nutzung einfachen Konfigurationsmechanismus. Um also eine Verknüpfung zwischen einem Präsentations- bzw. Bedienelement innerhalb eines Knotens und einem Zielknoten herzustellen, bedarf es lediglich einer entsprechenden Auszeichnung des Elements auf der Ebene der XML-basierten Beschreibung des referenzierenden Knotens (siehe die Ausführungen zur darstellungsunabhängigen Datenhaltung, weiter unten in diesem Abschnitt). Das impliziert etwa auch, dass die Ableitung eines sehr simplen Systems mit vollständig ausgeprägter Navigationsstruktur, welches lediglich elementare hypermedial verknüpfte Medienelemente und die im Framework akkumulierten Funktionalitäten nutzt, ohne die Formulierung von ergänzendem White-Box-Code auskommt.

Die Ausgestaltung von Navigationsstrukturen gehört im Wesentlichen zu den Aufgaben von Autoren oder auch Fachdidaktikern. Nur diese sind in der Lage, inhaltlich sinnvolle Zusammenhänge herzustellen und mit der nötigen Sensibilität für die Eigenarten der Zielgruppe, den Nutzer durch die Anwendung zu leiten. Durch die starke Abhängigkeit von den individuellen technischen Möglichkeiten einzelner Entwicklungsparadigmen und die damit verbundenen Spezifika der Entwicklungsumgebungen, werden die Implementierung und damit in der Regel auch große Teile der Konzeption der Navigation weitgehend von den Technikern ausgeführt. Über den Zwischenschritt der Ausprägung eines interaktiven Prototyps, wird es mit dem hier vorgestellten Ansatz möglich, die Mittel der Ausgestaltung und der Umsetzung der Navigation so für Projektkontexte zu individualisieren, dass die jeweiligen Domänenexperten in die Lage versetzt werden, beliebige neue Sichten auf das System und insbesondere die Inhalte zu etablieren. So wird es ihnen beispielsweise möglich, Verknüpfungen zwischen Knoten herzustellen, Verzeichnisse und Übersichten anzulegen ohne sich mit dem Konzept eines Hyperlinks auseinandersetzen zu müssen. Die Werkzeuge, die den Fachleuten dezidiert für diese Aufgabe zur Verfügung gestellt werden, sind quasi „maßgeschneidert“ für die Unterstützung der etablierten Arbeitsweisen und mentalen Modelle der Fachdomäne. Dabei sind zum Zeitpunkt der initialen Ausprägung des Frameworks zur Prototyping-Umgebung lediglich noch passende Interaktionsmöglichkeiten zu entwickeln, die die eigentliche Konfiguration kapseln. Diese Idee der domänenadaptierenden individualisierten Entwicklungsumgebung wird im Abschnitt 3.1.3 unter dem Stichwort „Adaptierbare Entwicklungsmetapher“ ausführlicher beschrieben.

Interaktionskomponenten

Auch wenn der Nutzerdialog der die Navigation realisiert im Einzelnen von Interaktionen zwischen Nutzer und grafischer Schnittstelle getragen wird, also auch Interaktionskomponenten wie Buttons benötig werden, um Sichten zuwechseln, ist die Abgrenzung zwischen Navigation und Interaktion zur Manipulation von Daten oder zur Unterstützung der Rezeption von Inhalten im Sinne von [EOOH94] innerhalb des Frameworks sinnvoll und dementsprechend umgesetzt.

Das zentrale Konzept der Gliederung der Interaktionskomponenten ist ein Panel. Ein Panel steht letztlich für eine in sich abgeschlossene Sicht oder auch einen Knoten innerhalb

der Navigationsstruktur. Das Panel fasst Bedienelemente zur Manipulation und Präsentation von Daten zusammen. Es ist grundsätzlich vergleichbar mit einer singulären Webseite. Historisch bedingt ist die Bezeichnung des Wurzelknotens der XML-Repräsentation nach wie vor <Page>. Ein kanonisches Rendering für einen fest definierten Satz an Elementen, die ein Panel aufnehmen kann, gibt es jedoch nicht in der Art von HTML und der Umsetzung im Web-Browser. Auch die Kontexte der Einbindung des Panels sind beliebig und nicht auf eine Browserkomponente wie ein Popupfenster oder einen Frame beschränkt.

Dementsprechend exemplarisch ist die Abbildung 3.9[7]. Das Framework bietet zwar einerseits, wie in der Grafik angedeutet, einen Fundus an grundlegenden Bedienelementen oder auch ganzen Aggregaten an Interaktionseinrichtungen an, ermöglicht es aber andererseits sehr einfach über ein definiertes Entwicklungsschema (also einer Art Programmiermodell) beliebige disponible Komponenten zur Einbindung in ein Panel hinzuzufügen. Den technologischen Anknüpfungspunkt markiert dabei sowohl eine White-Box-Schnittstelle zur Realisierung spezieller „framework-naher“ Komponenten (etwa in enger Verzahnung mit der Rahmennavigation) oder aber eine Black-Box-Schnittstelle zur Einbindung relativ eigenständiger Interaktionselemente. Durch die Ableitung konkretisierter Klassen von Interaktionselementen von einer Basisklasse `Interaktionselement` erhalten die so eingebundenen Komponenten, als Instanzen der entsprechenden Klassen, alle grundlegenden Eigenschaften zur Einbindung in den Kontrollfluss der Gesamtanwendung. Dazu gehört die Verknüpfung mit allen benötigten elementaren Handler-Routinen zur Verarbeitung von Nutzerinteraktionen, wie Mausklicks. Auf diese Weise erhält etwa auch jedes Element die Möglichkeit (abhängig von der Konfiguration) innerhalb des Panels durch Nutzerinteraktionen manipuliert zu werden. Dazu gehören beispielsweise folgende grundlegende Eigenschaften:

- Freie Positionierbarkeit innerhalb des Panels. Das ist einerseits die Grundlage für das Layout der Panels, also das Arrangement der einzelnen Interaktionskomponenten und Medienelemente durch die Redaktion. Anderseits ist das Verschieben und Verankern von Elementen auch ein elementares Interaktionsmuster, das es beispielsweise erlaubt Zuordnungsspiele zu gestalten oder auch Layout als Grundlage freier gestaltender Arbeit des Endnutzers anzubieten. Ein so genannter `LayoutManager`, der Bestandteil des umgebenden Panels ist, ermöglicht es dabei beispielsweise Abhängigkeiten in der Positionierung der Elemente zu spezifizieren und in der Bildschirmdarstellung umzusetzen, um etwa ein Element direkt unter einem zweiten anordnen zu können.
- Freie Transformierbarkeit bezüglich der Größe und theoretisch auch der Orientierung (Es werden in der Regel bewusst nur die elementarsten Möglichkeiten, die Macromedia Flash bietet, an die Flame-Entwicklerschnittstelle weitergegeben),
- Die Möglichkeit der persistenten Speicherung der Daten und des Zustandes (also auch des Erscheinungsbildes und der Konfiguration) eines Elements, sowohl indi-

[7] Die Darstellung und insbesondere die Bezeichnungen abstrahieren leicht vom tatsächlichen Design des Framework-Prototypen zu Gunsten der Anschauligkeit.

viduell innerhalb des aktuellen Anzeigekontexts, als auch im Rückgriff auf global festgelegte Stile,

- Die Einbindung in ein globales Konzept der Steuerung bzw. Administration dieser Eigenschaften, um zur Erstellungszeit festlegen zu können, welche Möglichkeiten der Manipulation dem End-Nutzer tatsächlich zur Verfügung stehen – ggf. in Abhängigkeit von projektabhängigen Rechteregimen, die etwa die Unterschiede in den Zugriffsmöglichkeiten auf einzelne Komponenten zwischen Lehrern und Schülern aber auch zwischen End-Nutzern und Redakteuren abbilden.

- Dabei unterscheidet die Entwicklerschnittstelle zwei Modi der Bearbeitung: einen so genannten Autorenmodus, in dem alle Möglichkeiten der Disposition und Konfiguration von Elementen zur Verfügung stehen. Ein Inspektor- oder Eigenschaftsdialog, der im Kontext jedes Elements geöffnet werden kann, ermöglicht es den Autoren, sehr feingranulare Einstellungen an der Steuerung und der Gestaltung jedes einzelnen Elements vorzunehmen. Demgegenüber steht der Rezeptionsmodus, der dem Nutzer, in Abhängigkeit von den durch die Autoren eingeräumten Rechten, die grundsätzlich gleichen Funktionen der Disposition und Interaktion eröffnet. Details der Konfiguration, die der Inspektorendialog Autoren zugänglich macht, bleiben den Rezipienten üblicherweise verborgen.

- Durch die konsequente, vom Framework sichergestellte Anwendung des objektorientierten Programmiermodells, liegt jedes neu eingebrachte Bildschirmelement zunächst in Form einer Klasse vor, von der beliebig viele Instanzen erzeugt und wieder zerstört werden können. Das heißt, dass der Nutzer für jede Art von Element die Möglichkeit erhalten kann, neue Ausprägungen in Panels einzubinden und diese wieder zu löschen. Mit Hilfe des Reflection-Musters ([BMR+98] Seiten 193 ff.), wird es dabei möglich die Erzeugung von Instanzen zur Laufzeit dynamisch zu steuern und Quasi-Serialisierungen der Elemente persistent zu speichern (siehe dazu auch den Abschnitt zur Meta-Architektur).

- Die Handler zur Bearbeitung von Mausinteraktionen und Tastatureingaben sind mit einer Basisfunktionalität vorgeprägt, die es dem umgebenden Panel aber auch der Gesamtanwendung ermöglichen, das Zusammenspiel der Komponenten zu koordinieren. So verfolgt ein so genannter Fokusmanager, welches der angezeigten Elemente aktuell aktiv vom Nutzer bearbeitet wird. Das dient beispielsweise dazu, Menüzustände kontextabhängig anpassen zu können oder aber die Präsentation zu steuern, um etwa eine Hervorhebung einzublenden. Das Bewegen des Mauszeigers über ein Element kann standardmäßig zur Einblendung eines so genannten Tooltips führen. Kontextmenüs als Reaktion auf einen Klick der rechten Maustaste sind ebenfalls konfigurierbar. Entsprechende hinterlegte Funktionen und Aktionen der Navigation über den Standard hinaus sind über White-Box-Schnittstellen zu ergänzen. Das Hinzufügen oder Entfernen von Standardbefehlen wie *„Kopieren“*, *„Einfügen“* oder *„Löschen“* entspricht einer Konfigurationsaufgabe.

Grundlegende Nachrichten, die ein Objekt an das umgebende System sendet, um globale Funktion wie die Navigation zu anderen Systemteilen auszulösen oder ein benachbar-

tes Element zu steuern bzw. Informationen zu übertragen, folgen ebenfalls den durch das Framework vorgegebenen Konventionen. Die entsprechenden Design-Entscheidungen bleiben den Anwendern des Frameworks genauso erspart, wie deren Umsetzung und der Test.

Die Binnenstruktur eines Elementes kann ganz individuell gestaltet sein. Aus der Sicht des Frameworks können diese Komponenten wie angesprochen den Charakter von Blackboxes haben, die eigene interne Kontroll- und Datenstrukturen etablieren. Lediglich die Konventionen der Schnittstellen, für die Einbindung, also auch die Verarbeitung der Beschreibung muss eingehalten werden. Das bedeutet im simpelsten Falle, dass das neue Aggregat eine Schnittstelle implementieren muss, die Funktionen zu seiner Initialisierung, seiner Zerstörung und zur Übermittlung seiner Dimensionen an die Gesamtanwendung umfasst. So ist es etwa möglich, vollständige in sich geschlossene, komplex verschachtelte Macromedia-Flash-Anwendungen einzubinden, ohne jedoch a priori gezwungen zu sein, sich mit den Mechanismen der Schachtelung so genannter *Flash-Moviclips* auf technischer Ebene auseinanderzusetzen. Diese Aufgabe ergibt sich im Wesentlichen nur noch während der initialen Ergänzung des *Funktionalen Rahmen* um neue Interaktionselemente bei der Adaption des Frameworks für einen neuen Projektkontext. Ein Ziel der evolutionären Weiterentwicklung des Frameworks soll es sein, den Fundus an Interaktionselementen so weit wachsen zu lassen, dass vollständige Neuentwicklungen auch dabei nicht mehr nötig werden. Wie die verbleibenden zumindest im Hintergrund technisch motivierten Schritte der Verzahnung, also der Auswahl, Ausprägung und der Einbindung von Elementen im angestrebten Produktionsprozess, aus Sicht der nicht technologisch orientierten Entwickler aussieht wird im Abschnitt zur adaptierbaren Entwicklungsmetapher besprochen.

Der angebotene Vorrat an Komponenten an sich ist jedoch bereits so angelegt, dass er zumindest diejenigen Funktionen enthält, die man von gängigen interaktiven Systemen und insbesondere Systemen zur Präsentation multimedialer Inhalte und der Vermittlung von Faktenwissen erwarten würde. Dazu gehören alle Einrichtungen zur Wiedergabe elementarer Medien, wie Abspielkonsolen für digitales Filmmaterial oder Audiosequenzen oder die schlichte Anzeige von Grafiken. So kann die Arbeit am Arrangement und der Inszenierung digitaler Medien die Grundlage für einen sehr schnellen Einstieg der Fachexperten in die partizipative Entwicklungsarbeit bilden und perspektivisch helfen, die initiale Phase der technischen Adaption des Frameworks für neue Anwendungskontexte weiter zu minimieren.

Auf Grund der Tatsache, dass die Mittel der Interaktion beim Umgang mit den elementaren digitalen Medien Audio und Video weitgehend standardisiert sind – die Gestaltung von Bedienelementen und der Einsatz von Symbolen weltweit gültigen Konventionen folgen sollten und Benutzer entsprechende Erwartungen hegen – ist es nahe liegend vom in Abbildung 3.10 schematisierten grundsätzlichen Aufbau auszugehen. Varianten ergeben sich sicherlich bezüglich der grafischen Umsetzung und ggf. der Gliederung. Das Framework hält entsprechende Komponenten und damit Aggregate an Interaktionselementen für den Einsatz innerhalb der Panels bereit. Das heißt auch diese Player-Komponenten

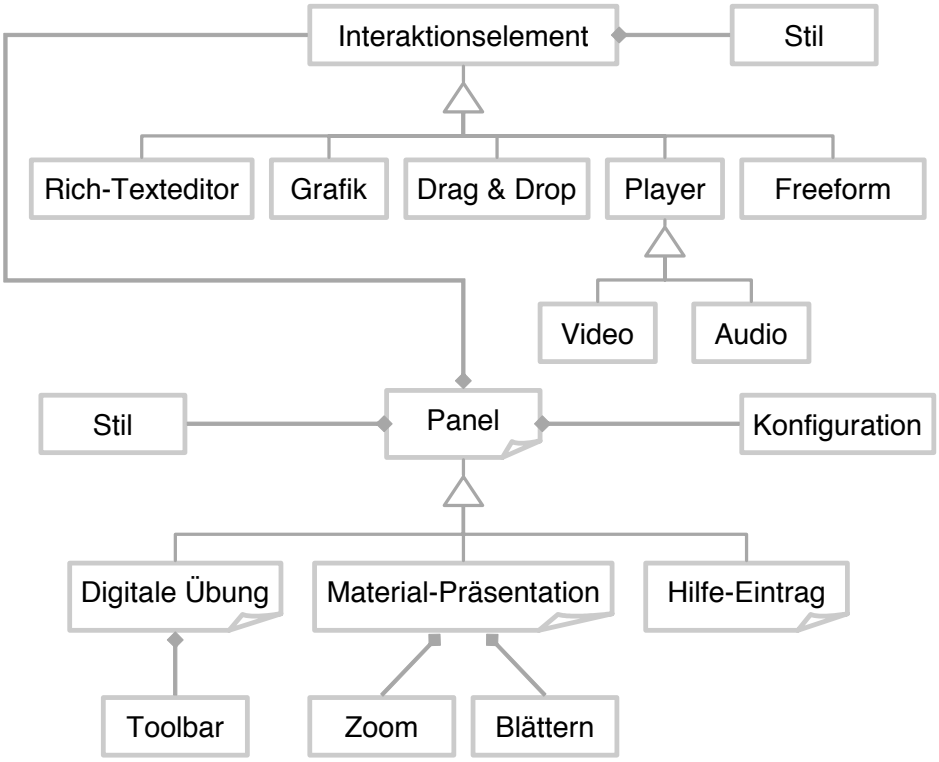


Abbildung 3.9.: Organisation von Elementen der Interaktion

„erben“ alle Eigenschaften der oben beschriebenen allgemeinen Interaktionselemente – mit ihnen kann dementsprechend frei disponiert werden.

Die Interaktionen der Anwendung und somit die Nutzung der Funktionalitäten entsprechen den intuitiven Mustern: Der im oberen Bereich der Abbildung wiedergegebene Fortschrittsbalken gibt dem Nutzer den aus gängigen Anwendungen gewohnten Überblick über die bereits verstrichene Spielzeit des aktuell abgespielten Audio- bzw. Videoabschnitts (Tracks). Diese Angabe wird präzisiert durch die Angabe der Track-Nummer im Verhältnis zur Anzahl der verfügbaren Tracks (01/03) sowie der Anzeige der verstrichen Zeit. Die Position innerhalb eines Tracks kann der Nutzer durch das horizontale Verschieben des Bedienelements (mit gehaltener Maustaste) innerhalb des Fortschrittsbalkens frei ansteuern. Einen ähnlichen Effekt hat der Druck auf die Tasten des schnellen Vor- bzw. Rücklaufs. In der Grafik dargestellt durch Doppelpfeilsymbole. Das Springen zu benachbarten Tracks bzw. zur Start- und Endposition eines Tracks erfolgt über die den Klick auf die Tasten mit den durch vertikale Balken begrenzten Pfeilen. Die Nutzung der länglichen Abspieltaste zum Start der Wiedergabe und die Stopptaste (mit dem Rechtecksymbol) zum Anhalten und Zurücksetzen der abgespielten Sequenz funktionieren in gleicher Weise intuitiv und scheinbar trivial wie die Pausentaste zum Stoppen der Tonsequenz an einer bestimmten Stelle. Ein Film würde entsprechend mit einem Standbild stoppen. Auch der Schieberegler zur Regulierung der Lautstärke erscheint

dem technisch versierten Anwender oder gar Entwickler von Informationstechnologie als nachgerade simpel. Die Einführung dieses kontextbezogenen Steuerelements, über die globale Steuerung der Lautstärke hinaus, ist jedoch didaktisch bzw. ergonomisch intendiert und für die methodische, medienbezogene Arbeit von Bedeutung.

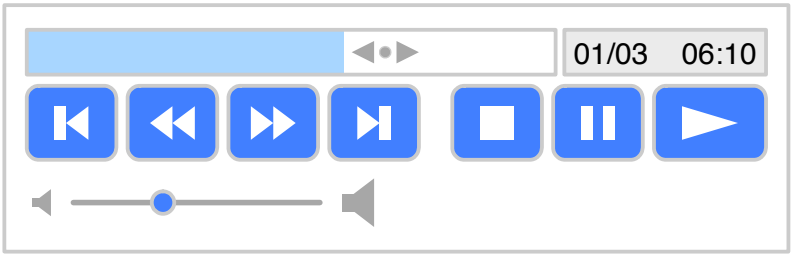


Abbildung 3.10.: Schema eines typischen Bedien-Panels von Audio- und Videoplayer-Komponenten

In den meisten der hier betrachteten Anwendungskontexte reicht es nicht, Text zu präsentieren. Für viele Fachbereiche ist es wichtig, dass der Endnutzer aktiv mit angebotenen oder auch während der Nutzung einzubringenden Texten arbeiten kann. Auch dabei unterstützt das Framework zunächst die elementarsten erwartungskonformen Funktionalitäten – wiederum als Ausgangspunkt für aufgabenbezogene Verfeinerungen:

Rich-Text-Formatierung: Es ist möglich, das Erscheinungsbild eines mit der Maus ausgewählten Texts oder eines Teiles davon farblich, bezüglich der Schriftfamilie, des Stils sowie in Bezug auf die Schriftgröße zu ändern. Die Auswahl der Farbe geschieht dabei in der Voreinstellung des Frameworks durch die Auswahl aus einer Palette. Schriftfamilie, -stil sowie -größe werden dem Nutzer entsprechend dem Quasistandard in Auswahllisten angeboten, die sich über kontextsensitive Werkzeugleisten – beim setzen des Eventfokus' auf Textelemente – aktivieren lassen.

Layout: Durch die Möglichkeit, Textbausteine auf mehrere Text-Elemente aufzuteilen (Interaktionselemente im Sinne des Frameworks – siehe Abbildung 3.9 auf Seite 149), erhält man frei disponierbare und in ihren Dimensionen veränderbare Textboxen, auf deren Basis es beispielsweise Möglich ist, einfache Tabellen zu setzen oder Beschriftungen für grafische Schemata zu erstellen.

Präsentation: Ebenso elementar wie entscheidend ist die Möglichkeit, jede dieser Textboxen im Falle eines „Überlaufs", also bei einem im Verhältnis zur Textlänge und -breite zu kleinen Rahmen, automatisiert mit Scroll-Balken auszustatten. Dieser augenscheinlich recht simple Mechanismus birgt für traditionell bezüglich des Medieneinsatzes eher an Texten orientierte Domänen einiges Potential bei der Gestaltung neuer Formen der Präsentation und damit neuer Sichten auf Inhalte und deren Zusammenhänge. So können etwa umfangreiche Textpassagen, die in gedruckter Form lediglich durch Umblättern von Papierseiten zu lesen wären, auf kleinstem Raum und in Beziehung gesetzt rezipiert werden.

Die in Abbildung 3.9 als `Freeforms` bezeichnete Elementklasse steht als abstrakte Klasse für eine Sammlung von grafischen Primitiven, wie Kreisen, Rechtecken, Blockpfeilen, Verbindungslinien sowie einer Einrichtung zur Eingabe eines freien Polygonzugs. Aus technischer Sichte entsprechen diese Primitive Konkretisierungen einer Oberklasse `Freeform`. Eine Erweiterung der Sammlung, etwa um Fachbezogene Symbole – ist also durch Ableitung weiterer Unterklassen im Rahmen der initialen Ausprägung des Frameworks einfach möglich. Diese Elemente können einerseits innerhalb der Panels als fixierte rein grafische Gestaltungsmittel eingesetzt. Andererseits können sie als zentraler Bestandteil der Nutzerinteraktion genutzt werden. Damit ist es zum Beispiel möglich, dem Nutzer einfache Zeichen- oder sogar Modellierungswerkzeuge zur Verfügung zu stellen.

Über diese vektor-orientierte Form der Grafikeinbindung hinaus, können Rastergrafiken in Form von JPEG-Dateien in Panels eingebunden und als frei disponierbare Elemente verwaltet werden. Auch für die Unterstützung der Rezeption elementarer Grafiken stehen spezielle Interaktionskomponenten zur Verfügung, die abhängig von der Intention der Präsentation bzw. der Didaktik für jede eingebundene Grafik konfiguriert werden können. So kann Grafik durch einfachste Schritte der Konfiguration, letztlich das Setzen einer Auswahlbox, mit einer Steuerleiste ausgestattet werden, die es dem Nutzer ermöglicht, die Abbildung zu vergrößern bzw. zu verkleinern (also zu zoomen). Überschreitet dabei die Größe der gesamten Grafik den darstellbaren Ausschnitt, eröffnet sich dem Nutzer automatisch eine Einrichtung zum verschieben des Bildausschnittes durch Ziehen des Bildes mit dem Mauszeiger. So können etwa digitale Faksimile historischer Buchseiten als Grundlage historischer Quellenarbeit inszeniert werden. Darüber hinaus kann jede Grafik, wiederum durch einfache konfigurierende Schritte in eine Folge von Bildern eingegliedert werden. Entsprechende Schaltflächen ermöglichen eine Navigation in dieser Folge von Grafiken, die den Eindruck des Blätterns erzeugt. Dies ist etwa wieder bei der digitalen Repräsentation von Druckwerken sinnvoll. Dient aber etwa auch zur Verdeutlichung von Unterschieden zwischen mehreren Darstellungen, ein und desselben Bildinhalts.

Einen Spezialfall unter den vorgefertigten disponiblen Elementen stellen so genannte `DropZones` dar. Diese fungieren als Zielanker für die Positionierung beliebiger Interaktionselemente. Neben dem schlichten Einrastenlassen von Elementen, um dem Nutzer eine gewisse Gestaltungsvorgabe etwa bei der Bearbeitung von Puzzles machen zu können, ist es ferner möglich, inhaltliche Zuordnungen zwischen Elementen und Zielzonen zu machen. Auf diese Weise wird es möglich, die Zuordnungsentscheidung des Nutzers zu bewerten. So kann beispielsweise die korrekte Vervollständigung eines Puzzles bzw. die richtige Platzierung eines Elements (Puzzleteils) zur Laufzeit geprüft und dem Anwender eine entsprechende Rückmeldung gegeben werden. Diese als *Drag & Drop* bezeichnet Interaktionsform ist als vorgefertigtes Modul Bestandteil der meisten Autorenwerkzeuge und schafft somit bezüglich des Frameworks einerseits Erwartungskonformität. Auf der anderen Seite eröffnet die Möglichkeit, beliebige Medien also auch Filme und beinahe beliebige Software-Module sehr einfach zum Teil von Drag & Drop-Arrangements zu machen ein neues Spektrum an Gestaltungsmöglichkeiten. Die einfache Beherrschbarkeit leitet sich dabei von den Prozessen der Co-Adaption ab, aber auch von der Ergonomie

der Anlage des Frameworks und insbesondere der entsprechenden Entwicklerschnittstellen, die von den technischen Aspekten der Umsetzung weitgehend abstrahieren.

Der Grund für die ausführliche Darstellung der sehr nahe liegend erscheinenden Interaktionselemente ist die Verdeutlichung der Möglichkeiten, die sich für einen Fachexperten ohne jegliche Technologieaffinität ergeben, dadurch dass er in die Lage versetzt wird, Mediale Inhalte seiner Fachdomäne mit derartigen interaktiven Mitteln gestalterisch aufzubereiten. Durch die Co-Adaptivität des Ansatzes wird dies im Gegensatz zu den meisten technologisch abstrahierenden Werkzeugzentrierten Vorgehensweisen möglich, ohne Kompromisse bei der Flexibilität machen zu müssen.

Gerade das Anliegen, ein Vorgehen mit evolutionärem Charakter zu induzieren, was die nicht technisch orientierten Beteiligten ganz behutsam an den Einsatz neuer Medien zur Wiedergabe ihres Wissens heranführen soll, wird durch die Vorgabe vertrauter Interaktions-Paradigmen unterstützt. Dabei ist auch diese Vorgabe auf Grund der beschriebenen Architektur der Interaktionselemente im Allgemeinen flexibel genug, um auf die Nuancen der zwischen den Fachdomänen variierenden Anforderungen eingehen zu können. Das betrifft in erster Linie die Anpassung des Erscheinungsbildes durch den vom Screen-Design-Rahmen bereitgestellten Skinning-Mechanismus. Aber auch funktionale Anpassungen sind auf Grund des transparenten Aufbaus einfach möglich: im Beispiel der in Kapitel 4 vorgestellte Umsetzung eines Systems u.a. zur Vermittlung von Methoden der Filmanalyse war es nötig, den Standard-Movieplayer um eine Zoomfunktion zu ergänzen. Eine derartige Erweiterung ist ohne Änderungen am Code möglich, einfach durch die einmalige Anpassung der Vorlage für die Konfiguration - die Metabeschreibung der Architektur zur Konfiguration des Renderers. Man nutzt an dieser Stelle die so genannte Black-Box-Schnittstelle zur Einbindung neuer Funktionen. Eine weitere für den Beispielkontext nötige Anpassung war die Einführung von Start- und Stopmarken entlang der Zeitachse des Films für die Steuerung des Filmablaufs – als simple Möglichkeit, den Filmschnitt zu simulieren. Diese stellen eine echte Erweiterung der `Movieplayer`-Komponente, die an den White-Box-Schnittstellen des Framework-Codes ergänzt werden mussten.

Beispiele für die spontane kreative Ausnutzung der durch die elementaren Mittel gewonnen Gestaltungsspielräume etwa durch Historiker sind Übungen zur historischen Topographie, wobei mit den beschriebenen Mitteln eine Aufgabe zur Beschriftung einer „Stummen Karte“ erstellt wurde. Interaktive Übungsblätter (Panel) zur Vermittlung von tatsächlicher Handlungskompetenz bei der Analyse historischer Texte fordern von den Rezipienten die Markierung von Schlüsselwörtern (mit den beschriebenen Mitteln der Textmanipulation) entsprechend bestimmter thematischer Schwerpunktsetzungen oder die systematische Extraktion und Kommentierung von Passagen. Das erfolgt etwa durch das Kopieren aus einer Text-Box und das Einfügen in ein grafisch gliederndes Schema von Textboxen. Dabei ist die Fokussierung der entsprechend benötigten Interaktionsmöglichkeiten auf diese Tätigkeit von didaktischer Bedeutung. Die kontextbezogene Reduktion der Funktionsvielfalt – etwa die Beschränkung der Einträge des Kontextmenüs auf die Befehle *„Kopieren“* und *„Einfügen“* – kann jedoch mit den erläuterten

Mitteln so gesteuert werden, dass der Eindruck der Arbeit mit Standardbedienparadigmen im Sinne der Vermittlung von Medienkompetenz nicht verwischt wird. Eine detailliertere Beschreibung zu den Erfahrungen, die während dieser Entwicklung gemacht wurden, folgt im Kapitel zur Evaluation.

3.1.3. Das Anwendungs-Framework

Wie zu Beginn des Kapitels (Seiten 125) beschrieben, ist das Framework bezüglich zweier Abstraktionsebenen differenziert zu betrachten. Nach den eben beschriebnen Aspekten, die dem Framework seinen domänenspezifischen Charakter verleihen, wird nun auf die Teile des Softwarerahmens eingegangen, die isoliert betrachtet einem allgemeingültigen Anwendungs-Framework entsprechen und theoretisch auch als solches genutzt werden können. Im Rahmen der hier präsentierten Lösung unterstützen sie den entscheidenden Aspekt der Fokussierung der Entwicklungsarbeit auf Fachlichkeit dadurch, dass sie Infrastrukturaspekte und querschnittliche domänenunabhängige querschnittliche Funktionalitäten wie die Fehlerbehandlung, die Kommunikationsschichten oder Ein- und Ausgabemodi wegkapseln und durch offene Schnittstellen interoperabel halten.

Für den fachlich orientierten Entwickler sollen etwa auch Details der Integration und Handhabung von Funktionen zum Drucken oder Speichern von Inhalten verborgen bleiben. Diese Aspekte sollen implizit und konform zu dem was der Domänenfachmann im Kontext bestimmter Interaktionselemente und Teilanwendungen intuitiv erwartet, zur Verfügung stehen. Selbst elementare physische Aspekte, die sich etwa hinter dem Begriff der Datei verbergen, sollen dem Nutzer potentiell verborgen bleiben. Er speichert Inhalte und Daten innerhalb seines rein fachlichen Kontexts – also beispielsweise innerhalb einer fachsystematischen Struktur – ohne sich mit den informationstechnologischen Abstraktionen von Dateien und Verzeichnissen auseinandersetzen zu müssen. Ein weiteres Beispiel ist die Einrichtung einer Druckmöglichkeit für jedes Interaktionselement, für das dies intuitiv wichtig und sinnvoll erscheint. Aus der Sicht des nicht technisch orientierten Autoren ist es entscheidend, dass sich dies nach der initialen Ausprägung des Frameworks entsprechend der dort getroffenen Festlegungen einfach im Zuge seiner disponierenden arrangierenden Tätigkeit implizit ergibt, ohne dass er sich mit der Konfiguration oder gar Implementierung von Funktionen auseinandersetzen muss.

Das heißt, dass die im Folgenden beschriebenen Instrumente hauptsächlich die Arbeit der technischen Entwicklung in den frühen Phasen der avisierten Projekte vereinfachen sollen. Ihre Anwendung und Anpassung erfordert die Kenntnis der wichtigsten Konzepte des Frameworks und technischen Sachverstand für den Umgang mit den verwendeten Technologien. Sauber definierte Programmierschnittstellen und die Verwendung von Standardtechnologie sollen jedoch einen Betrag dazu leisten, den technischen Entwicklungsaufwand insgesamt so gering wie möglich zu halten um der Programmatik des Ansatzes gemäß, die fachliche Entwicklung in den Mittelpunkt stellen zu können.

Die Mikro-Server-Architektur

Das tragende software-technologische Element des Frameworks ist die so genannte *Mikro-Server-Architektur*. Sie bildet die Basis für die Dekomposition und damit auch für die Konfiguration verteilter Anwendungen auf der Grundlage des hier vorgestellten Entwicklungsansatzes. Den Ausgangspunkt für die entsprechende Entwurfsentscheidung markierte die Schwierigkeit, lokal zu installierende Flash-Anwendungen modular zu betreiben und mit flash-fremden Komponenten zusammenarbeiten zu lassen. So wurde eine Infrastruktur nötig, um elementare Funktionen der lokalen Betriebssystemschicht in Anspruch nehmen zu können sowie Interoperabilität bzw. Interprozesskommunikation zu realisieren und Entwicklern zur Verfügung zu stellen. Das ursprünglich für die Erstellung so genannter reichhaltiger Web-Anwendungen[8] entwickelte Flash bietet lediglich die Möglichkeit auf der Grundlage des HTTP-Protokolls mit der Außenwelt zu kommunizieren. Diese Eigenschaft kollidiert beispielsweise mit Anforderungen, wie der Index-Basierten-Suche über einem lokalen Datenbestand einfach dadurch, dass die benötigte Funktionalität sinnvollerweise durch eine eigene, etwa in Java implementierte Anwendungsschicht bereitgestellt werden muss. Stellt man die Verbindung auf die vorgesehene Weise her, ist die Einrichtung eines entsprechenden lokalen web-basierten Dienstes unumgänglich. Beim Einsatz der marktverfügbaren Stacks zur Etablierung der beispielsweise benötigten Servlet-Kommunikation ergeben sich dabei insbesondere Schwierigkeiten mit dem Ressourcenbedarf sowie der Komplexität der Konfiguration. Das wiederum induziert Risikopotentiale bei der Sicherheit, der Handhabbarkeit und der Flexibilität während der Entwicklung sowie bei der Distribution.

Die Abbildung 3.11 verdeutlicht die grundsätzliche Anlage der Architektur und gibt einen Ausblick auf die Varianten, die die Zielarchitekturen, der auf dieser Basis abgeleiteten Anwendungen haben können. Die Angedeuteten HTTP-Verbindungen veranschaulichen dabei jeweils denkbare Schnittstellen zwischen der Präsentationsschicht (in der Grafik als Rich-Client bezeichnet) und der potentiell beliebig lokalisierten Anwendungs- bzw. Datenzugriffsschicht (jeweils durch den Block „Servlet“ symbolisiert). Die Komponenten, die das jeweilige Servlet aufnehmen, entsprechen der Middleware, die den Austausch physisch realisiert. Die gestrichelt hergestellte Verbindung zwischen der Präsentationsschicht und der Datengrundlage zeigt die vorgesehene Möglichkeit eingeschränkt auch den direkten Daten-Zugriff zu realisieren, um völlig statische Zielkonfigurationen zu erlauben.

Das als Teil des Flame-Frameworks entwickelte Anwendungs-Framework zur Ausprägung der *Mikro-Server-Architektur* nutzt einerseits die Flexibilität der Web-Architektur und insbesondere die Webservice-Techniken, um es den Entwicklern auf sehr einfache Weise zu ermöglichen, verteilte Anwendungen anzulegen. Durch die Einschränkung auf die wichtigsten Funktionen eines solchen Web-Stacks wird es jedoch andererseits möglich, die nötigen Komponenten so kompakt zu halten, dass diese Architektur selbst bei geringem Hardware-Einsatz innerhalb eines Adressraumes genutzt werden kann. Das heißt den Entwicklern steht für das Anlegen der grundlegenden Architektur des Zielsystems

[8] Die übliche englische Bezeichnung ist *Rich Web Client*.

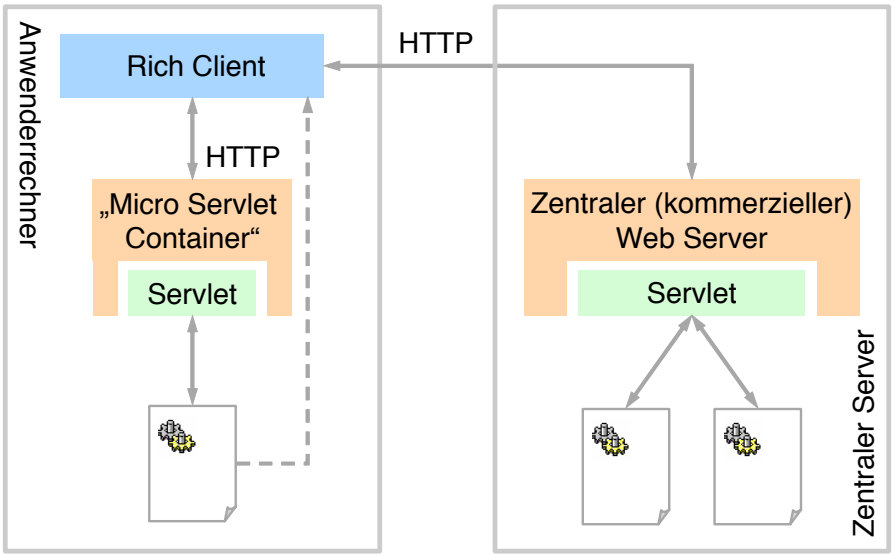


Abbildung 3.11.: Übersicht über den technologischen Aufbau der Infrastruktur

während der Phase der initialen Ausprägung der Prototyping-Umgebung ein leichtgewichtiger Container für Java-Servlets und darauf aufbauend SOAP-basierten Webservices zur Verfügung. Dessen elementarste Möglichkeit des Einsatzes in der Ausnutzung der als Bestandteil des Frameworks mitgelieferten Basisfunktionen besteht. Dazu gehören beispielsweise:

Personalisierung: Das Framework bietet Dienste, die es der Client-Anwendung ermöglichen, Nutzerkonten anzulegen und zu verwalten. Es stellt dazu die für die Registrierung, Identifikation, Authentifikation sowie die Autorisierung von Nutzern nötigen Funktionen bereit. Auf diese Weise lässt sich sowohl der Zugriff auf die Anwendung als auch auf Daten und Medien steuern. Damit ist es beispielsweise auch möglich, persönliche Bereiche innerhalb der Anwendung voreinander abzuschotten.

Die im Framework bereits realisierte Schnittstelle sendet dabei etwa als Reaktion auf einen `Login`-Request mit dem Nutzernahmen und einem Passwort als Parameter einfach einen Hinweis auf die Authentizität der Angaben und eine Kennung, die alle weiteren Anfragen innerhalb der Session als authentisch kennzeichnet. Weitere Parameter erlauben es ferner, den Request in das Anlegen eines Nutzerkontos münden zu lassen. In diesem Falle legt das System in der Voreinstellung ein Verzeichnis für die persönlichen Daten des Nutzers an. Die identifizierenden Einträge werden dem Verzeichnis verschlüsselt zugeordnet.

Datenzugriff: Das Lesen und das Schreiben persistent zu verwaltender Daten – in der aktuellen Ausbaustufe des Frameworks vorzugsweise in Dateien – ist ebenfalls eine Aufgabe, die die Infrastrukturschicht übernehmen wird. Neben der Anbindung an

die rein physische Verwaltung der Daten im Dateisystem oder zukünftig auch in Datenbanken umfasst das auch das logische Aufbereiten der Daten für den Transport und die Ablage. Das heißt also beispielsweise, dass XML-Dateien auszuwerten (parsen) und umgekehrt zu generieren sind.

Suche: Die Wirkungsweise der Suche aus Sicht des Nutzers wurde im Wesentlichen bereits im Abschnitt zum Navigationsrahmen (Seite 142) beschrieben. Die Querschnittsfunktionalität der Suche auf der Ebene des Anwendungs-Frameworks wird vom Client aus über einen so genannten SOAP-Aufruf angesprochen. Die Auswahl bzw. Ausprägung und Verknüpfung der Interaktionskomponenten zur Gestaltung der dazugehörigen Nutzerschnittstelle erfolgt in der initialen Phase der Entwicklung.

Zur technischen Umsetzung sei hier lediglich angedeutet, dass mit den Mitteln der Suchmaschinenbibliothek *Lucene*[9] entsprechend inverse Indexe erzeugt und auch dynamisch gepflegt werden können. Dabei wird jedes mit einer Teilanwendung assoziierte „Dokument“[10] mit den in ihm enthaltenen und im Index organisierten Schlagworten verknüpft. Die Möglichkeit, diese Schlagworte frei definierbaren *Feldern* [Luc06] zuzuordnen werden genutzt, um die attributierte Suche zu realisieren. Um Erweiterungen und Anpassungen an den genutzten Attributmengen vorzunehmen, müssen die vorhandenen Services lediglich fachlich angepasst oder in Analogie zu bestehenden ergänzt werden. Die für die Formatierung sowie Übertragung von Aufruf und Response benötigte Infrastruktur ist im Framework fixiert.

Wie sich eine Erweiterung bzw. grundsätzlich die Etablierung eines Flame-Dienstes im Zuge der technischen Ausprägung des Frameworks gestaltet, wird in Abschnitt 3.1.5 zu ausgewählten technischen Aspekten der Realisierung erörtert. Mit der Nutzung der vorgegebenen Infrastruktur und unter Einhaltung der Maßgaben für die Codierung neuer Dienste – also entsprechend des impliziten quasi Programmiermodells – ist darüber hinaus von vorn herein sichergestellt, dass die Server-Komponente für den Betrieb im Mehrbenutzerzugriff auch bezüglich schreibender Requests bereit ist. Das ist die Voraussetzung für die große Bandbreite an Zielarchitekturen, die auf der Grundlage des Frameworks realisiert werden können.

Neben der Ausnutzung der im Lieferumfang des Frameworks enthaltenen Basisdienste, bietet die Idee der Mikro-Server-Architektur den Software-Entwicklern vor allem die Möglichkeit die Zielarchitektur flexibel zu Gestalten. Die Entscheidung für den Aufbau der Gesamtanwendung wird dabei durch einen festen Satz gängiger Architektur-Gerüste determiniert. Der dadurch vermeintlich eingeengte Spielraum und daraus resultierende Risiken sind dadurch argumentativ zu relativeren, dass die zu erreichende Spannbreite die wichtigsten der Basisanwendungsfälle bezüglich der Verteilung und Knotenlokalisierung von Funktionalitäten und Daten abdeckt. Die Festlegung auf eine sinnvolle Auswahl an im evolutionären Framework-Entwicklungsprozess gut getesteten Grundaufbauten

[9] Siehe dazu http://lucene.apache.org/java/docs/ (zuletzt besucht am 02. März 2006)
[10] Im Sinne des Konzepts der Darstellungsunabhängigen Datenhaltung – beschrieben in Abschnitt 3.1.5

leistete darüber hinaus wiederum einen entscheidenden Beitrag bei der Vereinfachung der Technologie-Entwicklung zu Gunsten der Fokussierung auf die fachlichen Belange der Entstehungsprozesse. Ferner sind Bedenken bezüglich der Flexibilität schließlich dadurch zu entkräften, dass der Mikro-Server vollständig in frei veränderbarem Quellcode zur Verfügung steht. So dass es, wenn auch im Widerspruch zum Anliegen des Ansatzes, jederzeit denkbar ist, beliebige neue Architekturmodelle einzuführen.

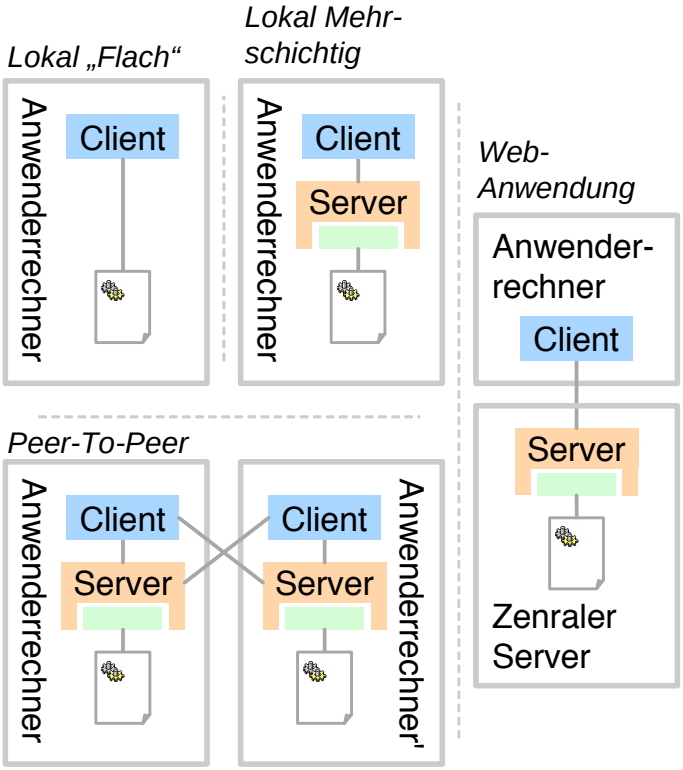


Abbildung 3.12.: Die Varianten möglicher Ziel-Architekturen

Die möglichen Architekturvarianten des Anwendungs-Frameworks und die damit zu unterstützenden exemplarischen Anwendungsfälle sind:

Lokal „flach“: Wie in der Abbildung 3.11 angedeutet, ist es im aller einfachsten Falle möglich, auf die eine „aktive“ Anwendungsschicht vollständig zu verzichten. Die Client-Anwendung bietet die Möglichkeit elementarste Präsentations-Aufgaben durch den direkten Zugriff auf die Datenbasis zu übernehmen. Sie besitzt an sich die Fähigkeit, sich im Sinne der noch zu besprechenden Meta-Architektur, auf Grund der konfigurierenden Teile der Datengrundlage selbst zu „entfalten“. Die Interaktionsfähigkeit beschränkt sich dann auf die beschriebenen Interaktionselemente der Oberfläche. Für die Navigation stehen lediglich statische Funktionen zur Verfügung. Einrichtungen wie die Suche oder die dynamische Nutzerverwaltung sowie pauschal alle schreibenden Aktionen können nicht genutzt werden. Diese

Variante adressiert bewusst klassische statische Medienproduktionen. Noch immer wird sehr oft Präsentation-Software entwickelt und vertrieben, die ausschließlich gebunden an das physikalische sekundäre Distributionsmedium (CD bzw. DVD) ablauffähig ist und eine Installation auf einem Rechner nicht erforderlich macht.

Lokal mehrschichtig: Unter Einsatz der sehr kompakten Server-Komponente (Mikro-Server), die integraler Bestandteil des Anwendungs-Frameworks ist, kann Anwendungslogik, und Funktionalität für den Zugriff auf persistent gehaltene Daten in eine separate Schicht der Architektur ausgelagert werden. Die ab einer bestimmten Größe des Systems und ab einer bestimmten Komplexität der abzubildenden Anwendungslogik notwendige Trennung zwischen den Aspekten der Nutzerinteraktion und Inhaltepräsentation von den Aspekten der Datenverwaltung und -verarbeitung lässt sich durch einfache Schritte der Konfiguration bzw. Installation bewerkstelligen. Die bereits angesprochenen Mechanismen zur Verwaltung personalisierter Datenbestände und die damit verbundenen Aufgaben der Verschlüsselung von Daten und der Koordination des Zugriffs auf Speichermedien sollte nicht innerhalb der auf die direkte Nutzerinteraktion spezialisierte Client-Komponente realisiert sein. Eine Anbindung entsprechender Funktionalitäten über feste Schnittstellen auf der Grundlage standardisierter Protokolle (wie SOAP) schafft neben der reinen Komplexitätsbewältigung, Datenunabhängigkeit und Transparenz auch die Chance auf die Wiederverwendbarkeit und Austauschbarkeit einzelner Komponenten. So kann die Client-Anwendung von einem Wechsel der personalisierten Datenverwaltung auf der Basis von XML-Dateien hin zu entsprechenden Funktionen, die sich eines relationalen Datenbankverwaltungssystems bedienen, unberührt bleiben.

Web-Anwendung Server: Unterstützt der im vorangegangen Punkt erläuterte Aufbau im wesentlichen Stand-Alone-Systeme, die zu einer Zeit auch nur von einem Anwender benutzt werden, ermöglicht es die Ausprägung dieser Architekturvariante, dass potentiell beliebige viele Nutzer von jeweils einer Client-Anwendung aus auf eine gemeinsame Anwendungsschicht und entsprechend eine gemeinsame Persistenzschicht zugreifen. Dabei skaliert die durchgängig auf Standart Web-Technologien aufbauende Mikro-Server-Architektur beinahe beliebig. Indem die auf das Wesentliche reduzierte Web-Service-Komponente gegen marktübliche Server, ausgetauscht werden, können die Durchsatzanforderungen üblicher Web-Anwendungen befriedigt werden. Aufgrund der Nutzung der Standards, können jederzeit für den Stand-Alone-Einsatz entwickelte Anwendungen in hochgradig verteilte Webanwendungen überführt werden – im Wesentlichen durch administrative Schritte. Dabei hängt es vom jeweiligen Projektkontext und dessen Distributionsanforderungen ab, ob etwa ein Sockel an Daten und die Client-Anwendung fest lokal beim End-Anwender vorinstalliert werden und lediglich Daten (etwa Strukturinformationen) und Medien von einem zentralen Server bezogen werden oder ob die gesamte Anwendung server-seitig vorgehalten und bei jedem Neustart der Anwendung vollständig und aktuell bezogen wird.

Peer-To-Peer: Der Einsatz eines Kleinst-Web-Servers, der potentiell auf jedem An-

wenderrechner, also auf jedem Client-Rechner installiert werden kann, erlaubt die Etablierung einer Architekturvariante, bei der jeder Client-Knoten gleichzeitig auch die Rolle des aktiven Anbieters also Servers für Inhalte übernehmen kann. Innerhalb dieser gemeinhin als Peer-To-Peer-Architektur bezeichneten Struktur kann jeder Knoten mit jedem direkt kommunizieren. Sie ist insbesondere für den Aufbau von aus Nutzersicht synchronen Verbindungen, etwa für die „Echtzeit“-kommunikation und den direkten Austausch von Daten geeignet. So können beispielsweise Arbeitsergebnisse zwischen Lernenden ausgetauscht werden. Aber auch die direkte Kommunikation mit Coaches kann etabliert werden, ohne den potentiellen Flaschenhals eines zentralen Servers belasten zu müssen. Neben der physikalischen Lastverteilung ist es so ferner einfacher machbar, einzelne Teile größerer Netze logisch voneinander zu trennen um sie etwa aus Gründen der Wahrung der Privatsphäre voreinander abzuschotten. Die Chancen die Peer-To-Peer-Lösungen bieten werden in [Ora01] ausführlich beschrieben. Eine Anwendung dieser Prinzipien zum Austausch digitaler Lehr-Medien wird in [KW02] diskutiert.

Die Abbildung 3.12 verdeutlicht schematisch die unterschiedlichen Möglichkeiten der Ausprägung von Zielarchitekturen auf der Basis des Anwendungs-Frameworks innerhalb von *Flame*.

Aus technischer Sicht, ist die Entscheidung für eine der Architekturen, wie angedeutet, ein Schritt der Konfiguration bzw. der Ausprägung der Meta-Level-Architektur, wie sie in Abschnitt 3.1.5 ausführlicher beschrieben ist. Ohne den Code des Frameworks berühren zu müssen, wird auf einer externen Ebene der Beschreibung des Systems (Meta-Level) festgelegt, welche Wege der Kommunikation zu nutzen sind.

Wie der folgende Auszug aus der XML-Repräsentation der Beschreibung der Architekturkonstellation zeigt, können Komponenten und die damit verbundenen Ressourcen sehr feingranular verteilt werden.

```
<BASE_URL type="String">.</BASE_URL>
<STREAMING_URL type="String">
     http://www.LeMOLernen.de/content
</STREAMING_URL>
<SAVE_URL type="String">
     http://127.0.0.1/SavePage</SAVE_URL>
<SHUTDOWN_URL type="String">
     http://127.0.0.1/Shutdown
</SHUTDOWN_URL>
<WEBSERVICE_URL type="String">
     http://127.0.0.1/axis/services/FlameService?wsdl
<WEBSERVICE_URL>
```

In dem wiedergegeben Beispiel verweist die Anwendungsbasis (`BASE_URL`) relativ zum Startort des Renderers auf eine lokale Installation. Das heißt, alle Informationen zur Generierung der Client-Anwendung werden ohne eine HTTP-Verbindung zu nutzen aus

dem lokalen Verzeichnis gelesen. Im Gegensatz dazu werden alle durch Streaming eingebundenen Inhalte (`STREAMING_URL`), wie etwa die Video-Materialien, von einem entfernten zentralen Web-Server bezogen. Die Anwendungslogik zum persistenten Speichern von Arbeitsergebnissen der Nutzer wird über den lokal installierten Mikro-Server angesprochen (`SAVE_URL`). Gleiches gilt für spezielle Funktionen, die beim Schließen der Anwendung ablaufen – referenziert durch die `SHUTDOWN_URL`. Auch Dienste wie die Suche und die Nutzeranmeldung, die in Webservices gebündelt werden, laufen auf derselben Maschine wie der Client und werden entsprechend über die `WEBSERVICE_URL`, der die Standardadresse `127.0.0.1` zugewiesen ist, angesprochen.

Das Beispiel entspricht also einem hybriden Aufbau, bei dem Teile der Anwendung vollkommen lokal und ohne die aktive Komponente der Anwendungsschicht angesprochen werden. Das umfasst sinnvollerweise einen Anwendungssockel mit den Basiskomponenten des Renderers, grundlegenden Strukturinformationen sowie einigen grafischen Ressourcen. Der Effekt dessen ist, dass grundlegende immer wieder kehrende Nutzerinteraktionen und Navigationsschritte sowie der Systemstart sehr flüssig laufen – unabhängig sind von Bandbreite und Server-Durchsatz. Extern gelagerte und mit dem Streaming-Verfahren eingebundene Filmsequenzen, helfen dagegen Ressourcen des Anwenderrechners zu sparen. Ferner besteht die Möglichkeit, die zentral auf einem möglicherweise redaktionell betreuten Server bereitgestellten Inhalte jeder Zeit aktuell zu halten.

Ergänzende querschnittliche Funktionalität

Auf Grund ihrer Unabhängigkeit von der Domänenspezifik sollen im Kontext der Beschreibung des Anwendungs-Frameworks noch querschnittliche Funktionalitäten besprochen werden, die erwartungskonformer Teil jeder interaktiven Software sein sollten. Durch die feste Vorgabe entsprechender Komponenten und ein Modell für den impliziten konsequenten durchgängigen Einsatz, unterstützt das Framework auch dabei die Fokussierung der Entwicklungsarbeit auf die fachlichen Aspekte. So kann beispielsweise in der initialen Phase der Ausprägung festgelegt werden, dass für einen bestimmten Typ von Anwendungsfenstern – etwa für Fenster mit interaktiven Arbeitsmaterialien – alle Nutzerinteraktionen, die zum Schließen des Fensters führen, pauschal in einem Dialog quittiert werden müssen, der durch Nachfrage sicherstellt, dass ungesicherte Veränderungen an den Inhalten persistent gespeichert werden. Während der eigentlichen fachlichen Entwicklungsarbeit impliziert man also mit der rein fachlich motivierten Entscheidung für ein interaktives Arbeitsmaterial, pauschal die Anlage aller für den korrekten Ablauf der Nutzerinteraktion notwendigen Funktionalitäten. Dieser Framework-Teil beinhaltet also alle grundlegenden Interaktionskomponenten zur Verwendung von Systemmeldungen und elementaren Systemdialogen, die beliebig oder wie beschrieben pauschal in die System-Abläufe aber auch die Navigationspfade eingebaut werden können. Dazu gehört etwa auch der Dialog zur Nutzeranmeldung, also zur Abfrage von Nutzername und Passwort zur Identifikation für den personalisierten Zugriff. Ein weiteres Beispiel ist eine Sicherheits-Nachfrage, die den Nutzer auf den Bedarf nach einer Netzverbindung oder ein potentielles Sicherheitsrisiko hinweist, wenn er einen Link nutzt der in das Internet

führt. Die Framework-Komponente von der dieser spezielle Internet-Link abgeleitet ist, besitzt die Fähigkeit, Zustände also etwa den Nutzerwunsch „Diesen Hinweis nicht mehr zeigen“ persistent zu speichern.

Um über die in den vorgefertigten Komponenten enthaltenen Funktionen hinaus projekt- bzw. kontextabhängig erforderlich Individualisierungen vorzunehmen, können wiederum nach dem White-Box-Prinzip Erweiterungen und Ergänzungen etwa durch Ableitung von bestehenden Klassen vorgenommen werden. Inhaltliche und stilistische Anpassungen – also die Konkretisierung bzgl. Des Erscheinungsbildes und der Ansprache des Nutzers – werden über konfigurierende Schritte bzw. unter Nutzung des Skinning-Mechanismus (Screen-Design-Rahmen) vorgenommen und erfordern keinen Implementierungsaufwand.

Ebenfalls im Rahmen der Querschnittsfunktionen anzusiedeln ist die zunächst trivial erscheinende Druckfunktion – in der Abbildung 3.8 repräsentiert durch das entsprechende Symbol – Deren globale Verfügbarkeit, spielt aus Sicht der Konvergenz der Mediennutzung zwischen Disziplinen eine offenbar nicht unerhebliche Rolle. Die Analyse des Problemkontexts ergab immer wieder die Forderung nach einer Möglichkeit, Bildschirminhalte zu drucken. Selbst für Ansichten interaktiver Anwendungen, deren Fixierung auf Papier vordergründig als kaum sinnvoll erscheint – Textfenster mit Scroll-Einrichtung, interaktive Zuordnungsübungen etc. – wurde in der qualitativen Analyse immer wieder gefordert, dass ein Ausdruck möglich sein müsse, um etwa in der Vorbereitung auf den Unterricht etwas „in der Hand zu haben“. Das gewohnte Medium erfüllt dabei offensichtlich eine Art Vermittlerfunktion am Übergang zwischen den klassisch angestammten Arbeitsweisen der fachlichen Domäne – diese sind für viele wissenschaftliche Domänen seit jeher stark papierorientiert – und den angepassten Arbeitsprozessen, die durch neue Medien unterstützt werden. Das Framework enthält eine entsprechende Funktion, die im Grunde für alle Ausprägungen weitgehend invariant bleibt. Eine elementare Funktionalität für die „Druckserialisierung“ langer Texte und eine entsprechende Umsetzung der Bildschirmdarstellung für gängige Seitenformate ist implementiert. Spezielle Anpassungen für die Erzeugung von Druckdarstellungen neu eingebrachter Medienelemente können vorgenommen werden. In der aktuellen Umsetzung ist eine konsequente Konfigurierbarkeit über Styles, wie sie aus dem HTML-Bereich bekannt sind jedoch noch nicht implementiert, sodass diese spezialisierten Transformationen im White-Box-Modus im Renderer ergänzt werden müssen.

3.1.4. Adaptierbare kollaborative Entwicklungsmetapher

Die Ausprägung eines Frameworks zu einer konkreten Anwendung nimmt bei dem hier vorgestellten Ansatz wie erwähnt insofern eine Sonderstellung ein, als dass vor dem Schritt der Ableitung der eigentlichen Anwendung zur Unterstützung der Aufgaben im Nutzungskontext zunächst ein Zwischenschritt der Instanziierung des Anwendungsrahmens zur domänenspezifischen Entwicklungsumgebung gegangen wird. Die Fähigkeit des Frameworks mit sehr einfachen technischen Mitteln in eine ablauffähige Prototyping-Umgebung überführt werden zu können, die ihrerseits ebenso nahtlos in ein tatsächliches

Produkt münden kann, bildet die tragende, technologisch instrumentelle Säule des im Abschnitt 3.2 vorgestellten Ansatzes.

Das Ziel ist es dabei zu vermitteln zwischen der technischen Perspektive des Einsatzes von Komponenten und der Entwickler-Schnittstellen des Frameworks einerseits und der Ausnutzung der Überschneidung von Tätigkeiten der Anwendung und der fachlichen Arbeit während der Entwicklung andererseits. Diese zunächst offenkundige Dualität soll jedoch nicht in völlig symmetrischer Weise überwunden werden. Es wird vielmehr angestrebt die Fachlichkeit des Nutzungskontexts, den Entwicklungsprozess dominieren zu lassen. Aus instrumenteller Sicht hat dies die Konsequenz, dass in untergeordneten möglichst kurzen und kostengünstigen Phasen der technisch orientierten Ausprägung und Anpassung des Frameworks die Werkzeuggrundlage dezidiert für die Bedürfnisse der rein fachlich orientierten Projektbeteiligten geschaffen werden. Rein technisch geprägte Abstraktionen als Vehikel der Gestaltung des Zielsystems – zur Beschreibung und schließlich auch zur Implementierung des Zielartefakts – werden ersetzt durch die Modelle, Begriffsbildungen und Ordnungsprinzipien der Anwendungsdomäne. Entwicklungsaufgaben mit ursprünglich technologischer Prägung werden transformiert in Aufgaben, die weitgehend durch an der Fachdomäne ausgerichteten Tätigkeiten erfüllt werden können. Die *Adaptierbarkeit*[11] der Entwicklerschnittstellen unterstützt dabei technologisch das „Entgegenkommen“ der Technologieentwicklung innerhalb des Prozesses der Verzahnung der Tätigkeiten der Disziplinen – innerhalb des Prozesses der Co-Adaption (siehe Abschnitt 3.3.2).

Die Schwierigkeit besteht darin, grundlegende Aufgaben der Entwicklung mit ursprünglich technischer Prägung, hinter Aufgaben der fachlichen Gestaltung zurücktreten zu lassen. Das Potenzial zur Erreichung dieses Ziels besteht in der Reduktion der Aufgabenvielfalt der Software-Entwicklung und insbesondere der Implementierung im Allgemeinen durch die Fokussierung auf domäneninhärente Muster und Vorgaben.

Antizipieren von Anforderungen: Die Ursprüngliche Aufgabe, Nutzeranforderungen widerspruchsfrei und vollständig zu erheben, zu spezifizieren und zu Systemanforderungen zu verfeinern, wird reduziert auf einen von den Nutzervertretern selbst getriebenen Prozess der experimentellen, diskursiven Entwicklung am evolutionären Prototypen. Der Prozesses ist im Idealfall selbstorganisierend – lediglich geführt durch die Vorgaben der Prototyping-Umgebung und moderiert entsprechend der Ideen des Vorgehensmodells (siehe Abschnitt 3.2).

Ableiten einer Architektur: Die Anzahl Architektur-Varianten und die Art ihrer technologischen Umsetzung kann ebenfalls begrenzt werden. Neben den Erfahrungen aus der Evaluation des Ansatzes spricht auch das Ergebnis der Untersuchung existierender Entwicklungssysteme dafür, dass es eine Auswahl an gängigen Formen des Aufbaus und der Distribution interaktiver Systeme im Sinne der hier verwendeten Definition gibt. So sollte die Auswahl, ob eine Standalone-Anwendung zur Installation von einem Massenspeichermedium entstehen soll oder Daten bzw.

[11] Unter Einsatz der Entwicklungsleistung der technisch fokussierten Entwickler im Gegensatz zur (automatischen) *Adaptivität* [Pre99] (Seite 62).

Funktionen in einem verteilten Umfeld genutzt werden sollen (für alle Varianten siehe Abschnitt 3.1.3 ab Seite 154) auch während der Entwicklung tatsächlich nur eine elementare Entscheidung erfordern. Details der technischen Auslegung werden in rein fachliche Gestaltungsschritte und gekapselt. Das heißt, sollte es fachlich nötig sein, dass einzelne Arbeitsplätze untereinander kommunizieren, wird die entsprechende Variante gewählt. Fragen zu Details des Protokolls stehen dann nicht zur Disposition – sind festgelegt durch das Framework.

Etablierung globaler Kontrollflüsse: Auch die Gestaltung grundsätzlicher Ablaufstrukturen des Systems, wie das Verhalten des Programms beim Start oder dem Beenden, das Zwischenschalten von Sicherheitsabfragen und Fehlerüberprüfungen und -meldungen sowie die Einrichtung von Speichermöglichkeiten etc. können reduziert werden auf simpelste Entscheidungen der Fachentwickler. Diese Entscheidungen münden in die implizite Integration und Anwendung erwartungskonformer technischer Lösungen. So kann etwa für die Entität des digitalen Übungsblattes pauschal festgelegt werden, dass beim Schließen des umgebenden Fensters, die persistente Speicherung von Änderungen durch einen entsprechenden Dialog sichergestellt wird. Der Autor bzw. Redakteur inszeniert aus seiner Sicht also lediglich ein Übungsblatt – die Mechanismen der Einbindung in das Aufrufregime der Gesamtanwendung bleiben ihm verborgen.

Setzen von Navigationspfaden: Die Aufgabe der inhaltlichen Strukturierung des Systems ist sehr stark abhängig von den Ordnungsprinzipien der Zieldomäne. Diese sind abzubilden auf die Abstraktion der Verknüpfung von Inhalten über das Setzen von Links. Aus technischer Sicht entspricht das dem Etablieren einer Nachrichtenverarbeitung, die die Nutzerinteraktion eines Mausklicks und die sich daraus ergebende Systemnachricht mit dem Laden und zur Anzeige bringen des gewünschten Zielknotens verbindet. Das eigentliche Strukturieren der Anwendung: Netze von Klassen und Messages aber auch Datenstrukturen Collections Listen, Bäume etc. werden vollständig überführt in die Begriffe und Ordnungsprinzipien der Anwendungsdomäne. Dort wird dann in der Regel von „Erlebniswelten“ und „Themenräumen“ gesprochen, die wiederum thematisch untergliedert sein können und schließlich „Lernwege“ und „Übungsserien“ enthalten. Die Gliederung ist dabei oft Fachsystematisch motiviert und hierarchisch aufgebaut. Querschnittliche Ordnungskriterien, wie die Komponente der Zeit im Fach Geschichte (also etwa als Zeitstrahl) sollen in der Regel auch abgebildet werden können.

Inszenieren von Medien: Als Verfeinerung der Etablierung der Navigation werden die Knoten der Navigationspfade konkretisiert. Einzelne Medien und die Funktionen zu deren Manipulation und Rezeption werden realisiert und in die Ablauf- und die Aufbau-Struktur eingebettet. Das heißt beispielsweise, das Pop-Up-Fenster mit sofort startendem Filmausschnitt als Reaktion auf den Klick eines grafischen Repräsentanten einer Thematik (technisch gesprochen: Button) implementiert werden müssen. Der Inhalt des präsentierten Panels (Dialogfensters) muss formatiert und mit der entsprechenden Interaktivität zur Steuerung des Films ausgestattet werden. All diese Aktivitäten gilt es wiederum hinter der fachlichen Aufgabe der

Zusammenstellung und Erstellung der Inhalte und deren Einbettung in didaktische Konzepte zurücktreten zu lassen. Einfach indem nahe liegende stereotyp benötigte Funktionalitäten und Layout-Prinzipien sich implizit durch die Auswahl der Medien ergeben. Ein einfaches und sehr praktikables Muster die übliche technische Sicht der Funktionsbibliothek zu ersetzen ist eine kanonische Assoziation von Daten- bzw. Medientypen mit passenden Funktionen bzw. Teilanwendungen – in Analogie zur Zuordnung von Dateitypen und Anwendungen bei der Nutzung grafischer Oberflächen gängiger Betriebssysteme. So wird die Präsentation eines Videoausschnitts immer die Player-Funktionalität benötigen, oder aber einem Bild immer eine textuelle Quellenangabe folgen. Varianten im Verhalten und der Struktur der Anordnung können in der Regel durch Schritte der Konfiguration oder gar WYSIWYG-Korrektur erstellt werden. Festlegungen und Spielräume die Konfigurationsmöglichkeiten eröffnen sollen, werden schrittweise im Rahmen der Phasen der initialen technischen Ausprägung des Frameworks getroffen und in entsprechenden Komponenten der domänen- oder gar aufgabenspezifischen Entwickler-Clients materialisiert.

Arrangieren von Funktionalitäten: Das Arrangieren von Funktionalitäten – als Teilaufgabe der Inszenierung der Medien eben besprochen – subsumiert hier eine Mischung aus Auswahl, Konfiguration, Spezifikation und einfachsten Implementierungs-Aufgaben von Interaktionskomponenten, die dem Nutzer die Rezeption und die Manipulation von Inhalten erlauben aber auch einfach der Unterhaltung oder dem Training von Handlungskompetenzen dienen können. Auch hier muss das Verständnis der Funktionsbibliothek möglichst durch eine Metaphorik belegt werden, die der Domäne Begrifflich leichter zugänglich ist. Dieses Verständnis für einen „Methodenbaukasten“ muss ggf. wiederum abhängig vom Entwicklungskontext individuell zum Projektstart etabliert werden.

Zur Illustration des Prinzips folgen nun im Vorgriff auf die im Rahmen der Evaluation beschriebenen Fallstudie einige der dort kommentierten Lösungen. Im Blickpunkt steht dabei die Möglichkeit, die Entwicklungsumgebung zur Unterstützung der ursprünglich technischen Teile der Arbeit so anzupassen, dass die fachlich inhaltliche Gestaltung oder gar die Weiterentwicklung im Kontext der Anwendung in den Vordergrund rücken können.

Die Abbildung 3.13 zeigt Auszüge eines der Werkzeuge, die im Zuge der Ausprägung einer domänenspezifischen Entwicklerschnittstelle dezidiert auf die Bedürfnisse der nichttechnischen Projektbeteiligten zugeschnitten wurden. Die hier erkennbaren Begriffsbildungen und Gliederungsprinzipien entsprechen weitgehend den Konzepten und Strukturen, mit denen sich Autoren bzw. Redakteure in ihrer Arbeit in einem mediendidaktischen Umfeld auseinandersetzen. Die Entwicklungsmetapher – also das was in gängigen Autorensystemen technisch abstrakten *Timelines* oder *Kartenstapeln* entspricht – ist hier angepasst an die Tätigkeitsmuster von Fachredakteuren, die üblicherweise im Umfeld von Printmedien arbeiten. Sie erhalten die Möglichkeit in beinahe gewohnter Weise Inhalte zusammenzutragen und zu gliedern. Die Konzepte und die damit verbundenen Begrifflichkeiten bzw. Symbole mit denen sie sich auseinandersetzen müssen, folgen

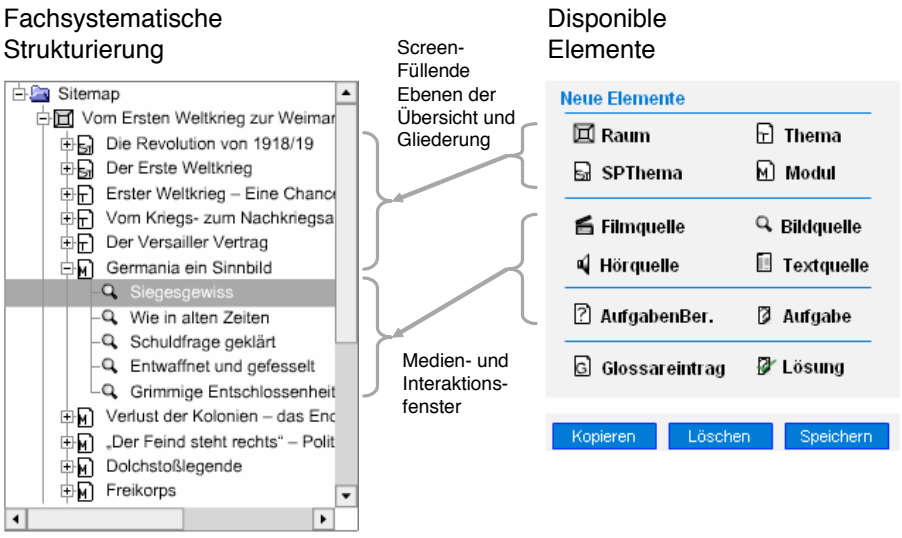


Abbildung 3.13.: Entwicklungsbausteine aus fachlicher Sicht am Beispiel LeMOLernen

vertrauten Mustern. Sie können mit *Film-* und *Hörquellen* arbeiten, die sie in *virtuelle Räume* und *Schwerpunktthemen* eingliedern, ohne sich mit Details von Videodateien und -formaten oder gar Instanziierungs-, Aufruf- und Zerstörungsregimen der Komponenten beschäftigen zu müssen.

Die Interaktionsschemata der Bildschirmtätigkeit sind dabei grundsätzlich so gewählt, dass sie intuitiv beherrschbar sind. Über einen Drag-und-Drop-Mechanismus ziehen die Autoren beispielsweise einen Baustein aus der Palette disponibler Elemente in den Baum der Gliederung. Dabei ist es möglich über Regeln festzulegen, welches Element unter welches geordnet werden kann. Auch Anzahlen können so z.B. gesteuert werden. Eine evolutionäre Anpassung der Nutzerschnittstelle sowohl bezüglich der Konzepte des Aufbaus und der Präsentation als auch bez. der Interaktivität ist jedoch Teil der Lösung, die die fachzentrierte Entwicklung möglich und effektiv machen soll. Das heißt also, dass im Rahmen anderer Projekte und insbesondere anderer Fachkontexte andere Tätigkeiten, damit Begriffsbildungen und Interaktionschemata zum Tragen kommen und entsprechend unterstützt werden müssen. Der evolutionäre Prozess der Anpassung der Entwicklungsumgebung und der Konfiguration des Renderers muss jeweils individuell angestoßen und gesteuert werden. Den Einstieg in den Prozess markiert jedoch die intensive Auseinandersetzung mit der Domäne und den dort vorherrschenden Arbeitsprozessen. Das Vorgehensmodell gibt dazu entsprechende Empfehlungen ab. Eine teilnehmende ethnographische Untersuchung der Produktions- und Anwendungskontexte der Zieldomäne erweisen sich als effektives Mittel beim Erkennen selbst von Nuancen innerhalb routinierter Tätigkeiten. Damit ergeben sich auch die Gestaltungskriterien der potentiellen Mittel zur Unterstützung der fachlichen Arbeit.

Die Ausnutzung denkbarer Überschneidungen von Anforderungen zwischen verschiedenen Zieldomänen erfordert die Erfahrung vieler Instanziierungen des Frameworks und

des Vorgehens. Eine solche fachübergreifende Aufgabe wird in jedem Falle die Feinjustierung des Arrangements der Medien und der Funktionalitäten durch die Redaktion und das Lektorat sein. Der dazu bereits etablierte WYSIWYG-Modus zur Bearbeitung der Software durch die direkte Manipulation der Oberflächen der Ziel-Anwendung wird weiter untern in diesem Abschnitt beschrieben.

Das Ergebnis der Entwicklungsleistung mit den Mitteln des Beispiels aus Abbildung 3.13 ist die Anlage der zentralen Abfolge von Bildschirminhalten, Dialogfenstern und entsprechender Menüstrukturen zu deren Aufruf. Mit dem Einfügen einer Bildquelle unterhalb eines Themensymbols wird entsprechend der Vorgaben der initialen Framework-Ausprägung ein Verweiselement (z.B. ein Button) im Panel des bildschirmfüllenden Themenüberblicks angelegt. Der Klick des Nutzers auf dieses Verweiselement, löst das Öffnen eines Fensters zur Präsentation der Bildquelle aus. Wiederum entsprechend der adaptiven Templates des Frameworks impliziert dies automatisch, dass das eigentliche Medium mit Funktionalitäten zur Unterstützung der Rezeption und Manipulation oder aber mit einem Feld für eine textuelle Quellenangabe versehen und im Sinne des Layout fertig arrangiert ist. Elemente der globalen Steuerung der Anwendung, wie etwa die *Sitemap* werden ebenfalls für den Entwickler transparent und implizit, automatisiert angepasst.

Die alternativ erwogene strukturierte Erfassung und Transformation der mit Hilfe dieses Werkzeugs gewonnenen Informationen durch die Extraktion aus strukturierten Manuskripten, stellte sich in Praxis als ungleich komplizierter heraus. Technische Abstraktionen, wie die Anwendung von Formatvorlagen, derer es bedarf, um Dokumente so zu strukturieren, dass sie tatsächlich maschinell auswertbar sind, binden bereits mehr Aufwand, als die eigentlich inhaltliche Leistung des Autors erfordert.

Eine direkte („straight-forward“) Transformation der Ergebnisse der Systemgestaltung, die über die weitgehend fachlich orientierten Entwicklungsschnittstellen zum Artefakt gewonnen wurden, in das Zielsystem ist im Umfeld industrieller Software-Produktionen nur begrenzt praktikabel. Die Vielfalt bzw. Vielgestaltigkeit der einzubindenden und umzusetzenden Inhalte und Funktionalitäten ist zu groß, als dass sich Transformationsregeln festlegen ließen, die einerseits generisch genug sind um beliebig komplexe interaktive Systeme zu konfigurieren andererseits aber auch präzise genug um ggf. pixelgenaue Präsentationsschichten ableiten zu können. Darin besteht unter anderem der Unterschied zu gängigen Content-Management-Systemen, die es erlauben eine statische Beschreibung der Transformation von Inhalten sowie Präsentations- und Konfigurationsinformationen in die eigentliche Darstellung zu fixieren.

Das durch die Fachleute über ihre individualisierten Schnittstellen erzeugte, im Einzelnen noch sehr an Schablonen orientierte System, kann aus diesem Grunde innerhalb des bereits erwähnen WYSIWYG-Modus direkt überarbeitet werden. Das erlaubt eine unmittelbare visuelle Bearbeitung der Nutzerschnittstellen, Inhalte und Funktionen des Zielsystems. So können Texte überarbeitet und Medien ausgetauscht und ergänzt werden. Es ist möglich das Layout zu überarbeitet und das Erscheinungsbild (Farben, Symbole, etc.) im Detail zu verändern. Auch Interaktionselemente und damit eigentliche Funktionalitäten können hier eingefügt und arrangiert werden. Dies erfolgt wieder

primär gekoppelt an Medien. Also das Einfügen eines Films in diesem Modus induziert wieder pauschal die Anlage einer entsprechenden Player-Einheit. Für einige der Interaktionsbausteine ist es das praktikabelste, wenn sie in diesem Modus in Panels des Systems aufgenommen werden. Etwa die Zielanker (*Dropzone*) der Drag-and-Drop-Übungen und freie Vektorgrafikelemente werden am einfachsten hier angelegt.

Wie die Abbildung 3.14 verdeutlicht, wird auch dabei versucht, technische Aspekte der Umsetzung weitgehend hinter Oberflächen- und Interaktionskonzepten zu verbergen, wie sie etwa aus Standard-Büroanwendungen bekannt sind. Das heißt also, dass der Nutzer dieses Modus' zumindest mit elementaren Bedienschemata üblicher grafischer Editoren vertraut sein muss. Die verwendeten Metaphern, Icons und Beschriftungen im Einzelnen sind jedoch auch hier durch Skinning-Mechanismen des Screen-Design-Rahmen bezüglich des Erscheinungsbildes flexibel und so bestmöglich an das Verständnis der Entwickler anpassbar.

Die inhaltliche Adaptierbarkeit des Modus beschränkt sich im Wesentlichen auf die Liste der disponiblen Elemente. So ist in den technischen Phasen der Ausprägung des Frameworks sukzessive festzulegen, welche Elemente zur Verfügung stehen und welche ggf. kontextabhängigen Ausprägungen entsprechender Software-Komponenten bei der Auswahl durch den Autoren tatsächlich eingebunden werden. Die Linkkarte z.B. (als Verweis „Link erstellen“ unten in der Liste der disponiblen Elemente in Abbildung 3.14, als eigentliches Element im Hintergrund der Bildmitte) ist mit Sicherheit nicht nur ein domänenspezifisches Element sondern sogar ausschließlich auf das Produkt bezogen verwendbar. Es stellt ein Aggregat aus einem miniaturisierten Medienelement (in der Regel ein Thumbnail-Bild), welches pauschal mit einem Verweis auf ein Materialfenster und optional mit einem Link zu einem Block interaktiver Übungen versehen ist. Auch die grafische Komposition des Aggregats als immer wieder, stereotyp genutztes Element des Panels ist vorgefertigt. Aus Sicht des Autors sind also lediglich die Karte auszuwählen und die entsprechenden Medien und Verweisziele anzugeben. Aspekte der Adressierung und des Aufrufs von Funktionen sowie der Integration von Komponenten bleiben ihm auch hier verborgen.

Durch diese explorative Arbeitsweise, die es dem technikfremden Entwickler erlaubt in sehr kurzen Zyklen zwischen konzeptioneller fachlicher Gestaltung mit den Mitteln der Zielanwendung, der Unterstützung der Umsetzung und der Anwendung der entstehenden Ergebnisse zu wechseln, verschwimmen Phasen der Entwicklung, des Tests und des Einsatzes des Softwareartefakts.

Die Bedeutung des entstehenden Software-Artefakts als gemeinsames Material im Sinne des Kooperationsmanagements – also als Dreh- und Angelpunkt des diskursiven Prozesses der Entwicklung – wird in Abschnitt 3.3.2 eingehend erörtert. Eine technologische Grundlage dafür legt das Framework und seine Anpassbarkeit bzgl. der Architektur. Die oben (Abschnitt 3.1.3) beschriebene Fähigkeit, die Anwendung in einem über ein Netzwerk verteilten Umfeld zu betreiben und die Tatsache, dass Nutzungskontext und Entwicklungskontext bez. der Werkzeuge weitgehend ineinander übergehen können, erlauben eine direkte Unterstützung geografisch verteilt arbeitender Teams. Aus technischer Sicht erfolgt die zentrale, inhaltliche Entwicklungsleistung über das so genannte

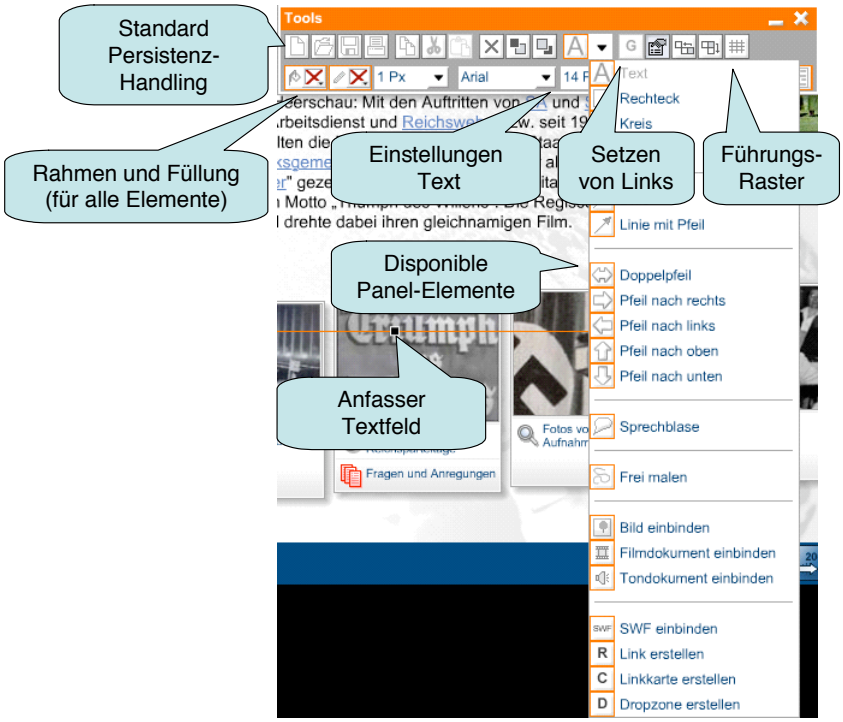


Abbildung 3.14.: WYSIWYG-Nutzerschnittstelle für Autoren und Redakteure

Meta-Object-Protocol (MOP)[12]. Dieses technische Konzept implementiert, den bisher allgemein als *Vorlage* oder *adaptives Template* bezeichneten Mechanismus. Die Clientsoftware zur Manipulation der das System beschreibenden und damit bestimmenden Objekte kann genau wie der Client des Endnutzers über eine HTTP-Verbindung mit dem „Backend“, also dem Software-Kern, verbunden werden. So werden die in Abbildung 3.15 schematisierten Teamkonstellationen möglich.

Die Meta-Level-Objekte der Konfiguration werden über das definierte Meta-Object-Protokoll modifiziert. Das MOP etabliert die jeweilige individuelle Entwicklungssicht auf die zentrale Systemausprägung. Das heißt zur Entwicklungszeit existiert potentiell für jeden Entwicklungsbeteiligten eine dezidiert für den Entwicklungskontext erzeugte Perspektive auf das Artefakt – er sieht und bearbeitet das Zielsystem ausschließlich über die Nutzerschnittstellen des MOP. Das gilt auch für den WYSIWYG-Modus der Bearbeitung. So kann jeder der Entwicklungs-Clients, zugeschnitten auf die jeweilige Aufgabe, die beschriebene spezialisierte Ausprägung haben. Die so ermöglichte Verteilung der Aufgaben auf Vertreter unterschiedlicher Disziplinen ist in der Abbildung durch unterschiedliche Symbole für Akteure dargestellt. Das Framework ist also auch bezüglich der „Projekttopologie“ adaptierbar.

[12] [BMR+98], Seite 195 – die Anwendung dessen ist in den Abschnitten 3.1.5 zu technischen Aspekten der Umsetzung näher beschrieben

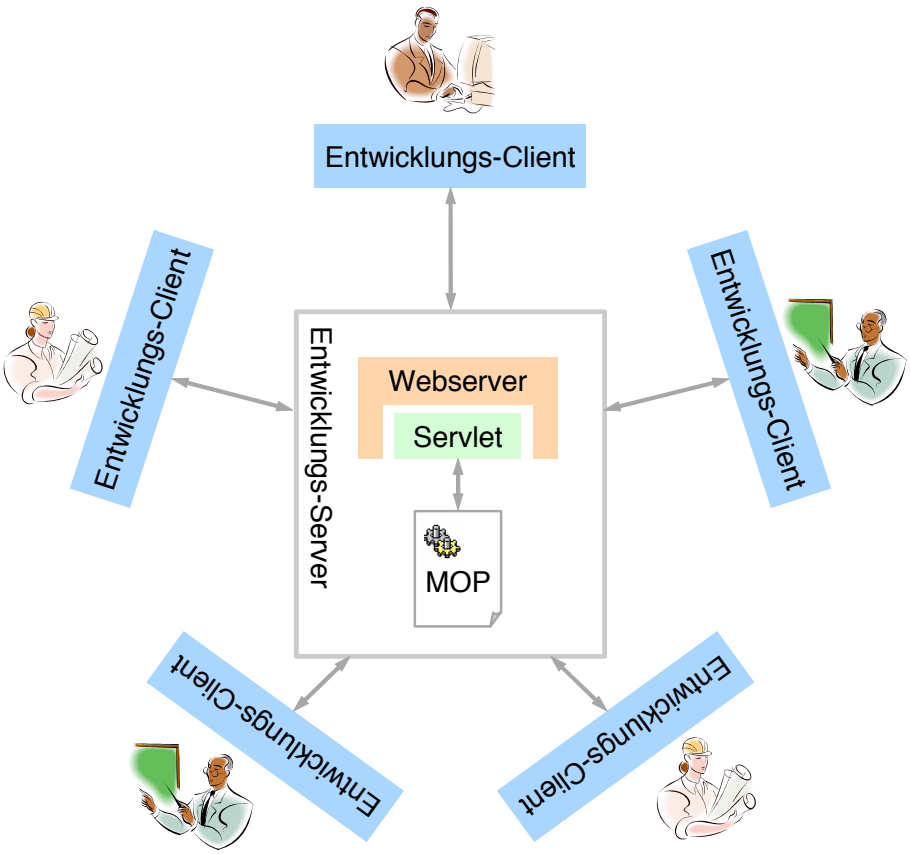


Abbildung 3.15.: Infrastruktur für die kooperative Entwicklung am singulären Artefakt

3.1.5. Aspekte der technologischen Umsetzung

Auf die bei der Entwicklung des Frameworks konkret eingesetzten Technologien wird hier nur schlaglichtartig eingegangen. Es werden lediglich diejenigen Aspekte näher beleuchtet, die die zentralen Beiträge zur Flexibilisierung des Frameworks leisten, so dass dieses die technologische Grundlage für den co-adaptiven Gesamtprozess bilden kann.

Die Meta-Level-Architektur

Die eigentliche technologische Grundlage der Flexibilität des Frameworks schafft die Anwendung des so genannten *Reflection-Architekturmusters*. Diese Art des Aufbaus von Systemen wird auch als *Meta-Level-Architektur* bezeichnet. Sie ist explizit zur Gestaltung dynamischer, also auch zur Laufzeit adaptierbarer Software konzipiert ([BMR+98] Seiten 193 ff.). Wesentliches Charakteristikum dieses Musters ist es, dass die so genannte *Basisebene* und die dort etablierte fixierbare Anwendungslogik bestimmt werden durch beschreibende veränderliche Informationen einer *Metaebene*. Die Basisebene kreiert und

konkretisiert die eigentliche Anwendungslogik mit Hilfe der Meta-Informationen über veränderliche Systeminterna.

Übersetzt in die schon mehrfach innerhalb der hier vorliegenden Arbeit verwendeten Begrifflichkeiten bedeutet das: der Renderer bildet die Basisebene, die ausgestattet mit einer Art „Keimzellen"-Funktionalität in der Lage ist, das eigentliche Ziel-System mit seinen Datenstrukturen und Kontrollflüssen aus der Konfiguration der Metaebene dynamisch zur Laufzeit abzuleiten. Im Falle des Flame-Frameworks basiert die Metaebene auf einer transparenten XML-Serialisierung, die mit beliebigen Mitteln manipuliert werden kann.

Die übliche Form der Beschreibung von Mustern durch die Konkretisierung der Kategorien *Kontext, Problem* und *Lösung* (gemäß [BMR+98] oder auch [GHJV05]) wird hier direkt übertragen auf die Anwendung des Musters im Rahmen der gefundenen Lösung:

Kontext: Die einfache und bereits aktiv durch das Instrumentarium unterstützte Anpassbarkeit sowohl der entstehenden Zielsysteme, wie auch der Zwischenschritte in Form der interaktiven Prototypen ist entscheidende Voraussetzung für den co-adaptiven Prozess des Entwicklungsvorgehens.

Problem: Der angestrebte Entwicklungsprozess soll von den frühesten Phasen an bis hinein in die Phase der Nutzung direkt am Artefakt „Zielsystem" ausgerichtet verlaufen und dabei potentielle Nutzer und Fachexperten einbeziehen. Um den damit verbundenen kontinuierlichen Weiterntwicklungszyklus von experimentellen Prototypen über ein Produkt bis hin zur Nutzerindividualisierung im Einsatzkontext etablieren zu können, ist es nötig eine Architektur zu nutzen, die für Veränderungen offen ist. Insbesondere den sich permanent verändernden Anforderungen durch sich entwickelnde Sichtweisen der Beteiligten innerhalb des wechselseitigen „Lernprozesses" muss möglichst direkt Rechnung getragen werden können. Die *Kräfte*[13] die das Problem für den hier adressierten Kontext im speziellen ausmachen sind:

- Software im herkömmlichen Sinne zu ändern ist für End-User im Grunde kaum möglich. Gerade spontane Anpassungen, deren Notwenigkeit sich aus Erkenntnissen im Experiment mit dem Prototypen oder gar im produktiven Einsatz ergeben, sind nicht vereinbar mit der üblicherweise notwendigen technischen Expertise und den bei großen Systemen anfallenden Integrations- bzw. Build-Prozessen.
- Adaptierbare System haben in besonderem Maße die Eigenschaft intern komplex strukturiert zu sein. Man kann von einem Overhead an Modulen und Schnittstellen sprechen, deren einziger Zweck die Sicherstellung der Anpassbarkeit ist. Es bedarf einer Möglichkeit, um die eigentliche Kompliziertheit, der Software vor denjenigen zu verbergen, die Änderungen hauptsächlich aus fachlicher Sicht vornehmen möchten.

[13] In Anlehnung an die Erörterung der *Kräfte*, die die Problemstellung im Allgemeinen aufspannen nach [BMR+98] Seite 194.

- Die üblichen Mechanismen zur Öffnung von Software für Anpassungen, wie die Verwendung von Mix-Ins, oder die Ableitung von Unterklassen sind auf bestimmten Ebenen zwar im Sinne der Flexibilität notwendig. Die sich ergebende Heterogenität der Mechanismen und auch das Niveau der technischen Abstraktion sind nicht geeignet, um technikfernen Fachleuten eine einheitliche, ihnen einfach zugängliche Einflussnahmemöglichkeit zu bieten.
- Die Art der Veränderung, denen das System unterworfen werden soll, kann bezüglich der Granularität sehr unterschiedlich sein. Es können sehr umfangreiche und grundlegende Änderungen benötigt werden, etwa an der gesamten Ablaufstruktur. Es kann aber auch lediglich gefordert sein, das Symbol einer Schaltfläche zu tauschen.

Lösung: Indem man die Software sich ihrer selbst bewusst macht ([BMR+98] Seite 194), also veränderliche Teile in eine Ebene der konfigurierenden Beschreibung aus der eigentlichen Programmstruktur herauslöst, schafft man die nötige Flexibilität, um die für das interaktive Prototyping nötige unmittelbare Adaptierbarkeit der Software zu gewährleisten. Veränderungen an einem solchen System erfolgen dann im wesentlich durch die Manipulation dieser Beschreibung über definierte Schnittstellen. Diese werden als *Meta-Object-Protocol* (*MOP*) bezeichnet und stellen sicher, dass die sich ergebende Konfiguration konsistent ist und durch die Basisebene „verstanden“ werden kann. Im hier präsentierten Ansatz gewährleisten sie ferner die Methodenunabhängigkeit der Entwicklerschnittstelle. Dabei erlaubt es die Flexibilität der Nutzerschnittstelle des MOP beliebige Sichten auf das bearbeitet Artefakt zu etablieren und damit beliebige Tätigkeitsprofile der Entwicklungsbeitragenden zu unterstützen. So kann von technischen Belangen der Programmierumgebung abstrahiert werden: fachlich fokussierte Entwickler müssen sich so nicht mit Konstrukten wie Klassen und Kontrollstrukturen auseinandersetzen, sondern in einer ihnen angemessenen Weise Inhalte und Funktionalitäten inszenieren etc. (Siehe dazu auch den Abschnitt zur adaptierbaren Entwicklungsmetapher).

Es ist überdies grundsätzlich einfacher und zuverlässiger, ein lauffähiges getestetes System unter Nutzung vieler wiederverwendbarer Komponenten über einen zentralen, durch alle Projektbeteiligten nutzbaren Mechanismus zu konfigurieren, als die häufigen in verschiedener Granularität benötigten Änderungen jeweils vollständig zu implementieren bzw. zu integrieren. Dieser Mechanismus macht es möglich auch Feinheiten an der Software zu manipulieren und den sich ergebenden Effekt sofort sichtbar zu machen, ohne lange und komplizierte Zyklen der Compilation bzw. des Release-Baus anstoßen zu müssen.

Struktur: Die Metaebene des Flame-Frameworks besteht aus einer Reihe von Metaobjekten, die jeweils eine der dynamisch veränderbaren Komponenten repräsentieren. So bündelt etwa ein Metaobjekt alle Informationen, die zur Initialisierung der Gesamtanwendung benötigt werden. Ein weiteres konstituiert den Aufbau und den Ablauf der Anwendung. Ferner bestimmt jeweils ein Metaobjekt den Aufbau und die Interaktivität der Anwendungsknoten – also die Dialoge und Bildschirme der Anwendung. Das Zusammenspiel mit dem Renderer, der im Wesentlichen die Basisebene bildet wird in

Abbildung 3.16 in Anlehnung [BMR+98] Seite 199 zusammengefasst. Die Klassen des *Meta-Object-Protocol* sind dabei zugunsten der hier interessanten Konkretisierung ausgespart. Die entsprechenden Komponenten gelten als Teil der Meta-Ebene und sind mit den Meta-Objekten jeweils so assoziiert, dass das MOP die Meta-Objekte „modifiziert“. Die Verbindungen zwischen der Meta-Ebene und der Basisebene stellen jeweils „Nutzt“-Beziehungen dar: die Basisobjekte „nutzen“ also die korrespondierenden Metaobjekte.

Eine Besonderheit des Ansatzes ist, dass Teile der durch die Adaption der Metaebene entstehenden Software quasi rekursiv wieder eingesetzt werden können als Bestandteil des Metaobjektprotokolls. Insbesondere die in den initialen Phasen vorgenommenen Anpassungen an Metaobjekten führen zu Ausprägungen der Gesamtsoftware, die von den nichttechnisch orientierten Entwicklern genutzt werden können, um die Objekte der Metaebene und damit wiederum die aktuell genutzte Werkzeugumgebung evolutionär anzupassen. So ergibt sich die Möglichkeit, direkt im Rahmen der aktuell ausgeführten Tätigkeit am Zielartefakt auch die Werkzeuggrundlage an entstehende Bedürfnisse anzupassen. In der aktuellen Umsetzung ist eine solche Arbeitsweise lediglich ansatzweise so unterstützt, dass Anpassungen an den für die Entwicklung relevanten Teilen tatsächlich selbstständig durch etwa Fachautoren vorgenommen werden können.

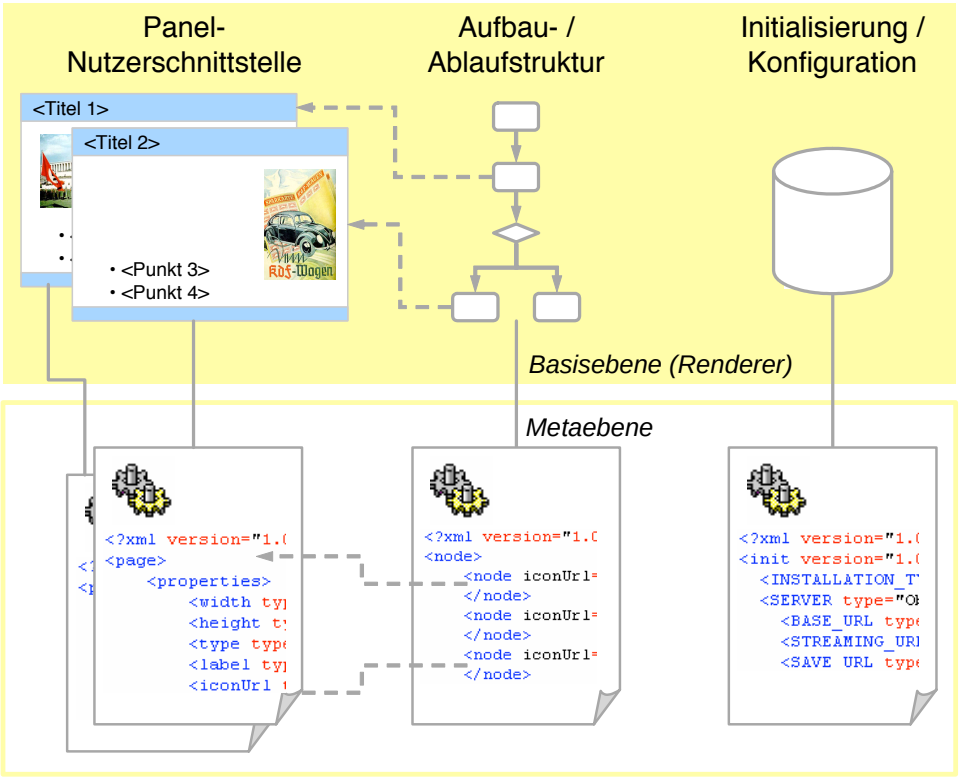


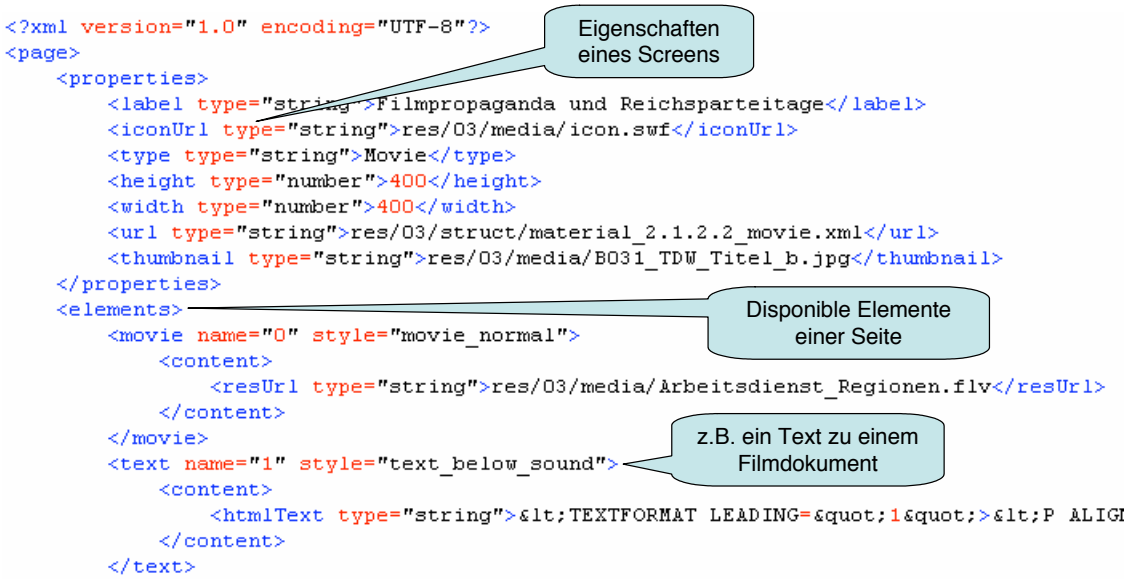
Abbildung 3.16.: Struktur des Systems entsprechend des Reflection-Architekturmusters

Umsetzung: Das in Anlehnung an die Dokumentenübersicht von Web-Sites als `sitemap`

bezeichnete Meta-Level-Objekt bestimmt das „Rückrat“ der gesamten Aufbau- und Ablaufstruktur des Zielsystems. Die fachliche Gliederung der Interaktions-Elemente ist hier zusammengeführt mit Ebenen der Navigation. Das heißt die fachliche Ordnung auf den Inhalten ist angereichert um Informationen, die die Abfolge und die visuelle Gestaltung der mit den Inhalten verbundenen Präsentations- und Interaktionselemente (Panels, Fenster, Medien-Player, etc.) steuern. Diese Beschreibung ist ferner die Grundlage für die Generierung der Meta-Navigation, also den Inhalteüberblick und die Möglichkeit der Angabe von Kontextinformationen, wie etwa den Pfad zur aktuell genutzten Teilanwendung. In Abbildung A.6 im Anhang (Seite 284) ist ein Auszug aus der für den Austausch und die persistente Speicherung genutzte XML-Repräsentation des Meta-Level-Objektes wiedergegeben. Im angegebenen Beispiel wird dem Nutzer auf oberster Ebene eine Auswahl an fünf gliedernden Panels (in der Abbildung Top-Level-Panels) präsentiert. Das heißt von einem anwendungsspezifischen Startbildschirm aus verzweigt die Anwendung auf fünf bildschirmfüllende Panels, die einen Themenschwerpunkt – im konkreten Fall eine historische Epoche – repräsentieren, orientierende Informationen und Fakten bieten sowie auf untergeordneten Detailthemen verweisen. Wie die Struktur bzw. die Informationen zu Ressourcen über die interne Ablaufsteuerung hinaus durch den Renderer schließlich tatsächlich in Elemente des Nutzerinterfaces umgesetzt werden, hängt von der initialen Ausprägung des Renderers auf der White-Box-Ebene ab. Standardmäßig kann jedoch etwa die Übersichtsanzeige in den Rahmen der Anwendung (siehe Seite 137) eingefügt werden oder das durch die `icon_url` referenzierte Symbol als Orientierungshilfe im gesamten Kontext des Themenschwerpunktes, also auch bei der Anzeige von untergeordneten Themen und Materialien präsent sein.

Das zentrale, von der Metaebene aus steuerbare Elemente ist das `panel` – in der XML-Repräsentation historisch bedingt auch als `page` bezeichnet. Jede Instanz eines Panels besitzt ein dediziertes Metaobjekt. Die darin enthaltenden Informationen steuern die Struktur eines Knotens innerhalb der Navigationsstruktur, also einen Dialog, einen vollständigen Screen bzw. den Inhalt eines Fensters bzw. Frames. Einen Eindruck zum Aufbau der XML-Repräsentation vermittelt die Abbildung 3.17: im Abschnitt `properties` sind grundlegende Eigenschaften des Panels konfiguriert. Diese Informationen nutzt der Renderer zur Einbindung des Panels in den Anzeigekontext. So wird etwa abhängig vom Typ (`type`) des Panels entschieden, ob es im Kontext eines Fensters zur Anzeige kommt, oder beispielsweise den Bildschirm füllen soll.

Ein detailliert kommentiertes Beispiel eines Panel-Kopfes findet sich in Grafik A.1 im Anhang (Seite 279). Die Angabe der Dimensionen wird im Falle der Ausgabe in einem Fenster genutzt, um die Rahmengröße des Fensters statisch vorgeben zu können – im Gegensatz zur dynamischen Ermittlung der Größe in Abhängigkeit von den Inhalten. Informationen wie der Titel (`label`) zur Generierung einer Fensterüberschrift oder der Hinweis auf eine Symbolgrafik (`iconUrl`) zur Verwendung als Platzhalter durch das aufrufende Panel bzw. Interaktionselement sind Angaben, die unter anderem für die durchgängige fachlich orientierte Präsentation aller disponiblen Elemente benötigt werden. Der Fachexperte disponiert also im Beispiel mit Videomaterial des Titels „Stalingrad“. In Übersichtsdarstellungen – etwa zur Gliederung – ist diese dann reduziert




```
<?xml version="1.0" encoding="UTF-8"?>
<page>
    <properties>
        <label type="string">Filmpropaganda und Reichsparteitage</label>
        <iconUrl type="string">res/03/media/icon.swf</iconUrl>
        <type type="string">Movie</type>
        <height type="number">400</height>
        <width type="number">400</width>
        <url type="string">res/03/struct/material_2.1.2.2_movie.xml</url>
        <thumbnail type="string">res/03/media/B031_TDW_Titel_b.jpg</thumbnail>
    </properties>
    <elements>
        <movie name="0" style="movie_normal">
            <content>
                <resUrl type="string">res/03/media/Arbeitsdienst_Regionen.flv</resUrl>
            </content>
        </movie>
        <text name="1" style="text_below_sound">
            <content>
                <htmlText type="string">&lt;TEXTFORMAT LEADING=&quot;1&quot;&gt;&lt;P ALIGI
            </content>
        </text>
```

Abbildung 3.17.: Auszug aus einer Page-XML-Instanz

auf ein Symbol oder auch die Miniaturansicht (`thumbnail`)[14] dargestellt. Diese elementare Auswahl an Attributen, benötigt das Framework in seiner aktuellen Ausprägung mindestens, um über den Reflection-Mechanismus alle für den Ablauf des Zielsystems nötigen Objekte und Objektstrukturen erzeugen zu können. Eine Erweiterung der Attributmenge und damit der durch das Panel unterstützten adaptierbaren Eigenschaften und Funktionen in Abhängigkeit vom Einsatzkontext ist möglich. Diese Erweiterung mündet jedoch in Änderungen auf der White-Box-Ebene des Frameworks. Das heißt, auch die Basis-Ebene (der Renderer) muss angepasst werden.

Die einzelnen im Panel arrangierten Inhalte und Interaktionselemente sind unter `elements` zusammengefasst. Ein detailliertes Beispiel für die Datenstruktur eines entsprechenden Meta-Level-Objektes ist in Abbildung A.2 auf Seite 280 wiedergegeben. Das Panel im angegebenen Beispiel beinhaltet ein Element des Typs (`movie`). Streng genommen ist dieses Element im Sinne der Objektorientierung Teil des Aggregates Panel. Es ist seinerseits wiederum ein Meta-Level-Objekt welches die Ausprägung der Basis-Level-Komponente zum Abspielen von Videosequenzen bestimmt. Entsprechend dem Reflection-Prinzip erzeugt der Renderer für jedes Panel-Element ein korrespondierendes Basis-Level-Objekt innerhalb der Projektionsfläche des Panels. Im Falle des Elements (`movie`) wird also im Panel unter anderem ein Objekt der Klasse Movie-Player erzeugt und mit den Informationen des Meta-Objektes initialisiert und konfiguriert. Primär bedeutet das die direkte Assoziation mit dem konkreten Videomaterial über den Verweis auf die Ressource (`resUrl`). Über so genannte *Styles* können Aspekte der Ablaufsteue-

[14] Eine statische, explizit festgelegte Miniaturansicht macht sich aus technischen Gründen bei Filmen erforderlich. Auch aus redaktioneller Sicht ist es oft notwendig, das passende Schlüsselbild eines Films oder einen repräsentativen Ausschnitt eines Bildes bewusst wählen zu können.

rung beeinflusst werden. Die Angabe `autoPlay` mit dem Wert `true` lässt den Film etwa sofort beim Öffnen des umgebenden Panels ablaufen. Das Beispiel illustriert dabei ferner, wie die globale Stilvorgabe des Filmelements

```
<movie name="movie" style="movie_norm">
```

lokal im konkreten Panel überlagert werden kann durch das einfügen eines `style`-Knotens.

Das Element vom Typ `text` im Beispiel realisiert aus inhaltlicher Sicht eine Bildunterschrift zum Filmmaterial. Die entsprechende Zuordnung im Sinne einer Schablone realisiert der globale Stil `text_below_sound`. Der sorgt dafür, dass sich das Textfeld mit Hilfe Layout-Mechanismen zur relativen Positionierung in immer gleicher Weise unter das Player-Element fügt. Die Definition solcher Styles ermöglicht es also Aggregate von Anzeige- bzw. Interaktionselementen zu bilden. Aus Sicht des Fachautors ist dies dann etwa stereotyp ein Filmmaterial mit Bildunterschrift zur Angabe von Quellen etc. Technische Interna des Layouts und der funktionalen Verbindung spielen aus dieser Perspektive keine Rolle. Die nötige Abstraktion wird durch den Einsatz der Stildefinitionen und der sich daraus ergebenden initial festzulegenden Interaktionsmuster (*Templates*) geschaffen.

Auf diese Weise ist auch die Zoomfunktionalität der Player-Komponente ergänzt – ausschließlich mit den Mitteln der Meta-Level-Beschreibung, ohne den Quellcode des Renderers zu berühren. Dazu ist in das Panel, grafisch innerhalb der Player-Bedienleiste ein Grafikelement eingefügt.

```
<image name="button_2x" style="button_movie_zoom">
```

Der Link, mit dem dieses Element verbunden ist, löst das Öffnen eines neuen Panels doppelter Größe aus. Das `image`-elemente agiert also aus Nutzersicht als Button. Dieses ist symmetrisch zum aufrufenden Panel aufgebaut. Es zeigt das zweifach gezoomte Filmmaterial und einen Button, der zurückverweist auf das Panel einfacher Größe. Eine Stildefinition stellt wiederum sicher, dass das Bedienelement im Verhältnis zur Player-Komponente innerhalb des Panels immer richtig positioniert ist.

Das Metaobjekt der zentralen Konfiguration (`init`) enthält alle Informationen, die für die Initialisierung der Gesamtanwendung benötigt werden. Dazu gehört der Wurzel-Verweis auf das erste Meta-Objekt der durch den Renderer zu entfaltenden Inhaltestruktur und Parameter zur Steuerung des Ablaufs. So wird etwa konfiguriert, ob und welche Clients (bzw. Domänen) sich mit dem Mikroserver verbinden dürfen oder nach wie vielen vergeblichen Versuchen, der Client es aufgibt, eine Verbindung mit Server-Komponenten aufzunehmen. Auch generelle Aspekte der Ausprägung der Architektur gibt diese Konfiguration vor. Eine ausführlichere Beschreibung zur Nutzung der Architekturkonfiguration befindet sich in Abschnitt 3.1.3 auf Seite 154. Ferner werden globale Stilrichtlinien zur Steuerung eines durchgängigen Erscheinungsbildes festgehalten sowie

elementare Ressourcen, wie Textlabels und Symbolgrafiken spezifiziert. Zur Illustration ist dazu auf Seite 282 ein Ausschnitt aus der entsprechenden XML-Repräsentation abgebildet und kommentiert.

Auswirkungen: *Vorteile:* Es werden keine Änderungen am Quelltext nötig. Es entfallen langwierige Zyklen der Integration und der Kompilierung. Die Manipulationen, die Nutzer und technikferne Projektbeteiligte am Prototypen bzw. am Zielsystem vornehmen, münden in Anpassungen der Metaobjekte. Die Auswirkungen auf das Verhalten und die Struktur des Systems werden unmittelbar deutlich. Dieses Feedback erlaubt eine direkte Verschränkung von operativer Arbeit auf der Grundlage des Systems mit der sukzessiven Anpassung der Instrumentengrundlage, die das System etabliert.

Das *Meta-Object-Protocol* unterstützt die Entkopplung der durch die Nutzerschnittstelle bedienten fachlich orientierten Tätigkeitsmuster von den Tätigkeiten eigentlicher Programmierung. Einrichtungen zur Veränderung des Systems können beliebig einfach gestaltet werden, unter entsprechender Einschränkung der Aspekte, die verändert werden können. Die dazu benötigte Offenheit für die Art und die Granularität der möglichen Anpassungen ist ein weiterer der Vorzüge des Architekturmusters. So kann eine sehr einfache Nutzerschnittstelle des MOP darauf spezialisiert sein, lediglich eine grobe Erfassung und Gliederung der Inhalte zu ermöglichen. Das arrangieren von Funktionalitäten im Detail bedarf dann komplexerer Interaktionsmechanismen des Entwickler-Interfaces. Diese Eigenschaft ist die Grundlage für die Gestaltung aufgabenbezogener, disziplinangepasster Werkzeuglandschaften.

Über definierte interne Schnittstellen und flexible Entwicklungssichten darauf kann die Fähigkeit zur Adaptierbarkeit abhängig vom Nutzungskontext kanalisiert werden. Die technische Konsistenz der Veränderungen kann durch das *Meta-Object-Protocol* weitgehend sichergestellt werden. Das heißt, das System bleibt in jedem Falle grundsätzlich ablauffähig – in der Regel auch nach umfangreichen Änderungen an der Struktur.

Nachteile: Die volle Flexibilität der Entwicklungsumgebung zur Ausnutzung aller Möglichkeiten der Interaktion steht nur zur Verfügung, wenn Anpassungen durch technisch versierte Entwickler vorgenommen werden können. Die Beschränkung auf initial festgelegte Interaktionsschemata kann sich als Hemmnis herausstellen, das aufwändige White-Box-Anpassungen nach sich zieht.

Der Aufwand für die Anpassung des MOP für einen Domänenkontext muss in der Regel durch wiederholten Einsatz gerechtfertigt werden. Insbesondere die Notwendigkeit, die Komponenten des MOP so robust zu gestalten, dass Inkonsistenzen in den Meta-Level-Objekten vermieden werden, macht Anpassungen am MOP teuer. Selbst eine leicht fehlerhafte Struktur der Metadaten, würde dazu führen, dass die Basis-Ebene die Informationen nicht interpretieren könnte. Die Erzeugung der Basis-Level-Objekte würde unmöglich und das System nicht ablauffähig. Inkonsistenzen, die sich bei der Erstellung der Meta-Level-Objekte über ein fehlerhaftes MOP ergeben haben, wären kaum zu korrigieren. Es bliebe in diesem Falle lediglich die Möglichkeit, das Protokoll anzupassen und die Objekte neu zu erzeugen bzw. anzupassen.

Darstellungsunabhängige Datenhaltung

Die Bezeichnung der hier beschriebenen Infrastruktur als „Flash Markup Engine (FLAME)“ ergibt sich neben dem Hinweis auf den Einsatz von Macromedia Flash als Präsentationsschicht aus der Tatsache, dass die zugrunde liegende Datenhaltungskomponente auf einer Auszeichnungssprache[15] ähnlich HTML beruht. Diese dient der Repräsentation sowohl der Inhalte, als auch aller strukturellen Aspekte der entstehenden Anwendung. Sie ist die Basis der persistenten Speicherung und des Austauschs der bestimmenden Konfiguration für den Aufbau, den Ablauf und das Erscheinungsbild der entstehenden Anwendung.

Ein Grundsatz dabei ist die strikte Trennung funktionaler Aspekte, von denen der inhaltlichen Aufbaustruktur und des Inhalts an sich. Die Inhaltliche Gliederung ist dabei direkt Verquickt mit der Struktur der Navigation und damit dem zentralen Ablauf des Systems als Abfolge von Interaktionsknoten – also der Panels, die Dialoge und Screen-Inhalte realisieren. Im System manifestiert ist diese Struktur in den beschriebenen Meta-Level-Objekten. Die Abbildung A.6 und A.2 im Anhang zeigen XML-Serialisierungen exemplarischer Objekte. Diese sind von der eigentlichen Präsentation – also der grafischen Umsetzung im Nutzerinterface noch weitgehend unabhängig. Erst die Bindung der globalen Stildefinitionen innerhalb einzelner Elemente eines Panels und die Ausprägung des beschriebenen Screen-Design-Rahmens bestimmen das Erscheinungsbild. Die Abbildung A.5 auf Seite 283 zeigt einen Auszug aus der XML-Repräsentation des zentralen Meta-Level-Objektes der Initialisierung (`init`) mit Beispielen der dort gebündelten Stildefinitionen. Diese werden, wie bereits angedeutet über das Attribut (`style`) durch die jeweiligen Elemente referenziert und können individuell überschrieben werden.

Die zunächst von der Art ihrer Präsentation unabhängigen Datenstrukturen und Medienelemente sind so klar zu trennen von Informationen, die etwa das Layout des Panels durch die Positionierung der Elemente bzw. deren Farbgebung bestimmen. Diese *Styles* sind vergleichbar mit den CSS zur Formatierung von HTML. Durch dieses Prinzip ist es möglich, rein fachlich strukturierte Inhalte auf unterschiedliche Art zu präsentieren, einfach, indem der genutzte Style – beispielsweise in Abhängigkeit vom Ausgabemedium – ausgetauscht wird. Mit der Trennung der inhaltlichen Aspekte von denen der Darstellung ergibt sich auch die Möglichkeit für eine Entkopplung der Arbeitgebiete. So können die Screendesigner an den Details der Präsentation arbeiten, ohne die Ergebnisse der Fachautoren berühren zu müssen. Der entscheidende Unterschied zu CSS besteht in der Erweiterung der Stilinformationen um domänenspezifische Aspekte der Steuerung von Interaktionselementen und deren Einbettung in globale Kontrollflüsse. So werden beispielsweise auch Zugriffsrechte auf einzelne Elemente und Aspekte des Verhaltens (z.B. der Autostart von Videos) durch die Informationen der Styles gesteuert.

Die innerhalb der Knoten präsentierten „atomaren“ Inhalte sind zunächst bewusst auf eine Auswahl gängiger Standardformate beschränkt, um die Portabilität des inhaltlichen Gerüsts zu maximieren und damit die Verknüpfung des Entwicklungsprozesses mit beliebige Schritten der Preproduktion zu ermöglichen. So besteht für jedes der unterstützten

[15] engl. Markup Language

Formate jeweils die Möglichkeit, es unter anderem mit einer entsprechenden Consumer-Standard-Software oder gar freien Werkzeugen zu erzeugen bzw. zu bearbeiten. Dies lässt einerseits den Produktionsprozess Skalieren von Anforderrungen, die pixelgenaues Arbeiten auf der Grundlage von professioneller Software erfordern bis hin zu Rapid-E-Learning-Produktionen mit Entwicklungszeiten kleiner als drei Wochen (vgl. [DV04]).

Andererseits wird es damit sehr einfach möglich den Individualisierungs- bzw. Weiterentwicklungsprozess des Systems für den Nutzungskontext zu öffnen. Jeder Anwender ist damit potentiell in der Lage, das System um selbst erstellte Medien und entsprechend implizit die Funktionen zu deren Manipulation zu ergänzen. Umgekehrt kann es im Sinne der Vermittlung medialer Handlungskompetenz wichtig sein, Medien aus dem vorgegebenen methodischen Kontext des Systems herauszulösen, um sich mit alternativen Mitteln und Vorgehensweisen zu deren Rezeption und Bearbeitung auseinanderzusetzen. So können etwa Bilder mit externen Grafikprogrammen bearbeitet werden. Innerhalb des Evaluationsprojektes beispielsweise kommt diesem Konzept insofern Bedeutung zu, als dass es Schülern über die Mittel der Bildmanipulation des Systems hinaus ermöglicht wird, unter den realen Arbeitbedingungen die eine externe Bildbearbeitungssoftware bietet, auch die „echte“ Handlungskompetenz im Umgang mit Informationstechnologie und damit die Konfrontation mit der Vielfalt an Handlungsoptionen zu trainieren.

Die Einfachheit, der gewählten Formate und die Loslösung von der Art ihrer Präsentation bzw. Einbettung in interaktive Elemente verfolgt ferner das Ziel, einer möglichst großen inhaltlichen Interoperabilität des Frameworks. Mit Hilfe der beschriebenen Architekturansätze wird es möglich, auf die Datenbestände verteilter Inhalteanbieter zuzugreifen. Auf der Ebene der elementaren Medien bedeutet das, dass einzelne Elemente über Web-Verbindungen zu den jeweiligen Inhalteanbietern eingebunden sein können. Um Konvertierungsproblemen bei der Nutzung der heterogenen Datenbestände zu entgehen und aufwändige Prozesse des statischen Einpflegens zu vermeiden, ist die unmittelbare dynamische Adressierung des „kleinsten gemeinsamen Nenners“ bezüglich der Medien eine adäquate Lösung.

Die aktuell durch das Framework unterstützten Formate und die damit assoziierbaren Funktionalitäten sind zusammengefasst:

Bilder (jpeg, png): In der Voreinstellung werden Rastergrafiken dieser Formate ohne jede Funktion in Panels eingebunden und entsprechend der Stilvorgaben formatiert und positioniert – ggf. mit einem Schatten oder einem Rahmen versehen. Die Angabe der folgenden Stildefinition bettet die angezeigte Grafik in einen „Projektor“ ein, der es dem Nutzer erlaubt durch Druck auf entsprechende Tasten (in der Regel mit Lupen-Symbolik) die Grafik zu zoomen.

```
<depth type="number">1</depth>
<zoomable type="boolean">true</zoomable>
<maxZoomSteps type="number">2</maxZoomSteps>
```

Die Anzahl der `maxZoomSteps` begrenzt dabei die mögliche Vergrößerung. Die Tiefe `depth` gibt an, welcher der initiale Vergrößerungsfaktor beim Öffnen der Grafik ist. Die Definition

```
<browsable type="boolean">1</browsable>
```

innerhalb des Stils einer Grafik ergänzt der Projektor ferner um Funktionen, die das Blättern in einer Folge von Bildern erlaubt.

Videosequenzen (flv): Werden Videosequenzen eingebunden, erhalten diese pauschal die Player-Komponente zur Steuerung des Ablaufs. Optional können so genannte Schnittmarken konfiguriert werden, um den Ablauf des Filmes auf einen vorgewählten Abschnitt beschränken zu können. Das Format ist zwar an die Produkte der Flash-Linie vom Macromedia gebunden aber dennoch losgelöst von einer Flash-Anwendung und insbesondere Außerhalb der mit dem Framework erstellten Systeme mit einer Vielzahl von externen Projektoren nutzbar. Der entscheidende Vorteil, besteht in der Fähigkeit, die Video-Inhalte über eine HTTP-Verbindung zu streamen. Auf diese Weise ist es möglich die Architekturvariante der Web-Anwendung so zu gestalten, dass der Client mit einem Mindestmaß an Ressourcen auskommt – keine kompletten Video-Dateien für die Anzeige übertragen und zwischengespeichert werden müssen.

Audio (mp3): Auch das Einbinden von Audioabschnitten (Element `sound`) in ein Panel induziert immer zunächst auch die Anlage des entsprechenden Player-Moduls, mit dessen Hilfe der Anwender das Abspielen von Audiosequenzen in erwartungskonformer Weise beeinflussen kann. MP3 ist das momentan am weitesten verbreitet Format zur digitalen, komprimierten Speicherung von Audiodaten. Das bedeutet unter anderem, dass Inhalte, so diese nicht durch explizite technische Einrichtungen intentional gesichert sind, etwa auch auf externe handelsübliche Abspielgeräte übertragen werden können.

Text (html): Textelemente werden durch die aktuelle Ausbaustufe des Frameworks aus rein pragmatischen Gründen noch innerhalb der Meta-Level-Beschreibung des umgebenden Panels (quasi *inline*) verwaltet. Eine Extraktion in eine externe abgeschlossene Form der Repräsentation – etwa eine HTML-Datei – wäre relativ einfach möglich. Der Text-Content nutzt jedoch im Wesentlichen nur die Teile von HTML, die den Text im Sinne von Rich-Text formatieren – keine vollständigen HTML-Dokumente. So werden die entsprechenden Auszeichnungen durch den Renderer beispielsweise in die gewünschte Schriftfamilie, den Schriftstil oder die Textfarbe umgesetzt.

(Interaktive) Animationen (swf): Der Wahl von Macromedia Flash als Plattform für die Generierung der Nutzerschnittstelle liegt die Intention zu Grunde, die dezidierte Ausrichtung dieser Plattform auf die Entwicklung interaktiver Systeme auszunutzen. Jenseits der methodischen und instrumentellen Einbettung dieser Technologie und der resultierenden end-user-zentrierten Entwicklungsszenarien, ist es so weiterhin möglich, das volle Spektrum der Leistungsfähigkeit von Flash zu nutzen. So können im klassischen Sinne entwickelte abgeschlossene Flash-Anwendungen beliebiger Komplexität im Shock-Wave-Format (swf) als externe Ressource in das System eingebettet werden. Die so eingebundenen interaktiven Animationen

müssen lediglich eine simple Schnittstelle implementieren, um vom Framework angesteuert (aufgerufen, skaliert, etc.) werden zu können. Die Schnittstelle ermöglicht es auch, dass die eingebundenen Teil-Anwendungen auf Ressourcen und Funktionalitäten der umgebenden Gesamtanwendung zugreifen, etwa um Systemmeldungen auszulösen.

Eine Sonderrolle nehmen Grafiken ein, die mit Hilfe der grafischen Primitive (*Linie*, *Rechteck*, *Ellipse*, *Freihandzeichnung*, etc.) im WYSIWYG-Modus des Frameworks erstellt wurden. Diese werden als Elemente direkt in die Meta-Objektstruktur des Panels eingebettet und entsprechen durch ihre Repräsentation einem proprietären Vektorformat. Dies ist zunächst der Pragmatik des derzeitigen Entwicklungsstandes des Frameworks geschuldet. Die Nutzung eines Standardformats wie **S***calable* **V***ector* **G***raphics* im Rahmen der Weiterentwicklung des Frameworks ist sinnvoll und wünschenswert. Eine Transformation zwischen beiden Formaten auf der Ebene der XML-Repräsentationen ist dazu denkbar.

Die Entscheidung für dieses sehr einfache aber damit flexible Konzept der Verwaltung von Inhalten leistet ebenfalls einen Beitrag zur Etablierung einer Entwicklungssystematik, die versucht Kompetenzgrenzen zwischen den Disziplinen aufzuheben. Die hohe Flexibilität des Konzepts ermöglicht es einerseits den Technikern, in den frühen Phasen der Ausprägung des Frameworks auf ein breites Spektrum an Anforderungen einzugehen. Die Möglichkeiten der Ausprägung von Interaktionsprinzipien auf der Grundlage der elementaren Medien sind beinahe beliebig vielfältig – das Spektrum reicht von der simplen Präsentation (mit der Funktion der Vermittlung aber auch der reinen Gestaltung) bis hin zu Werkzeugen der produktiven Ver- bzw. Bearbeitung der Daten.

Auf der anderen Seite sind die Medien so beschaffen, dass der Umgang mit ihnen auch auf der Ebene ihrer technischen Repräsentation (Dateien und Verweise zu ihrer Lokalisierung bzw. Identifizierung[16]) kaum noch wirkliche informationstechnologische Expertise erfordert.

Entkopplung durch Webservices

Ein weiteres Mittel zur Öffnung des Frameworks für möglichst weit reichende Adaptierbarkeit durch den Einsatz fachlicher, von technischen Interna abstrahierender Komponenten ist die Verwendung eines transparenten Schnittstellenstandards für die Ein- bzw. Anbindung der Komponenten und Systemteile. Auch die beschriebene Flexibilität der Architektur erfordert die Anwendung einer Schnittstellentechnologie, die die Fachlichkeit der zu integrierenden Funktionalität in den Mittelpunkt stellt. Infrastrukturelle Aspekte, wie die Etablierung von Netzverbindungen und die Interoperabilität im Allgemeinen, sollten dagegen aus Sicht des Entwicklers in den Hintergrund treten. Der ohnehin benötigte Einsatz des HTTP-Protokolls legte nahe, eine Middleware-Technologie

[16] URL bzw. URI im Sinne der durch das W3C-Konsortium referenzierten Spezifikation [BLFM05]

auf der Basis von Webservices für diesen Zweck zu nutzen. Das so genannte SOAP[17] Protokoll ermöglicht es, Funktionsaufrufe und die entsprechende Rückgabe über eine HTTP-Verbindung zu übermitteln und entsprechend zu koordinieren. Dabei sendet der Client (im SOAP-Kontext auch Requestor genannt) den Funktionsaufruf in Form eines HTTP-Requests an den Server. Es existieren für die meisten Entwicklungsplattformen Bibliotheken, die die Transformation des nativen Aufrufs – etwa in der Programmiersprache *Macromedia Flash Actionscript* – einschließlich der Serialisierung von Parameter-Objekten in die XML-formatierte SOAP-Message vornehmen. Der Client-Entwickler kümmert sich also lediglich um die fachlichen Aspekte der verwendeten Komponenten und der in Anspruch genommenen Dienste in Form von Proxy-Objekten und deren Methoden.

Der Server beantwortet die Anfrage indem er den HTTP- bzw. XML-Funktionsaufruf und seine Parameter deserialisiert und auf den Zugriff auf das entsprechende Serverobjekt abbildet. Das Ergebnis des serverseitigen nativen Funktionsaufrufs (z.B. eine Java-Funktion) wird wieder gemäß dem SOAP-Protokoll verpackt und als HTTP-Response an den Client zurückgesendet. Auch die Registrierung einer Funktion als Webservice wird für viele Entwicklungsplattformen bereits durch Toolkits unterstützt. So wird ein Service als herkömmliche Funktionalität der Serverplattform verfasst. Eine einfache Auszeichnung als Webservice über die Konfiguration des Webservers und des Toolkits veröffentlichen den Service. Das Toolkit sorgt dann für die Transformations-Kette der Requests und Responses und auch die Steuerung der Aufrufe – etwa das Dispatchen konkurrierender Zugriffe beispielsweise unter „Ausgabelung“ verschiedener Serverprozesse. Ob ein Server einen Dienst zur Verfügung stellt und wie genau er in Anspruch zu nehmen ist, also welche Aufrufe mit welchen Parameterlisten möglich sind, kann der Client aus einer vom Server gesondert zur Verfügung gestellte Dienstbeschreibung (mit Hilfe der WSDL[18]) extrahieren.

Wie konkret die Web-Service-Technologie innerhalb des Flame-Frameworks genutzt werden können und wie der entsprechende technische Aufbau aussieht, illustriert Abbildung 3.18 am Beispiel der Volltext-Suchfunktion. Die entsprechende implementierende Java-Klasse durchsucht mit Hilfe einer durch einen Index unterstützten Bibliothek alle Textdokumente der Metaebene auf den Anfrage-String (`Query`) hin. Das Ergebnis ist ein Array von `Result`-Objekten mit Attribute zur Typisierung und Referenzierung der Treffer-Dokumente. Auf der linken Seite der Abbildung ist wiedergegeben, wie die Java-Konstrukte der rechten Seite in der Flash-Entwicklungsumgebung repräsentiert werden. Dabei wird deutlich, dass bereits aus dieser Perspektive, die dem technisch fokussierten Flash-Entwickler vorbehalten ist, ausschließlich fachliche Aspekte der Schnittstelle präsent sind. Probleme der Adressierung, der Übertragung oder des Mappings von Datentypen spielen keine Rolle bei der Integration von Komponenten.

Auf dieser Grundlage ist es sehr einfach, im Rahmen der initialen Phase der Ausprägung

[17] **S**imple **O**bject **A**ccess **P**rotocol – die Standardisierung wird vom W3C-Konsortium vorangetrieben. Entsprechende Dokumente finden sich unter `http://www.w3.org/2000/xp/Group`.

[18] **W**eb **S**ervices **D**escription **L**anguage – eine XML-basierte Sprache zur maschinenauswertbaren Beschreibung von Webservices.

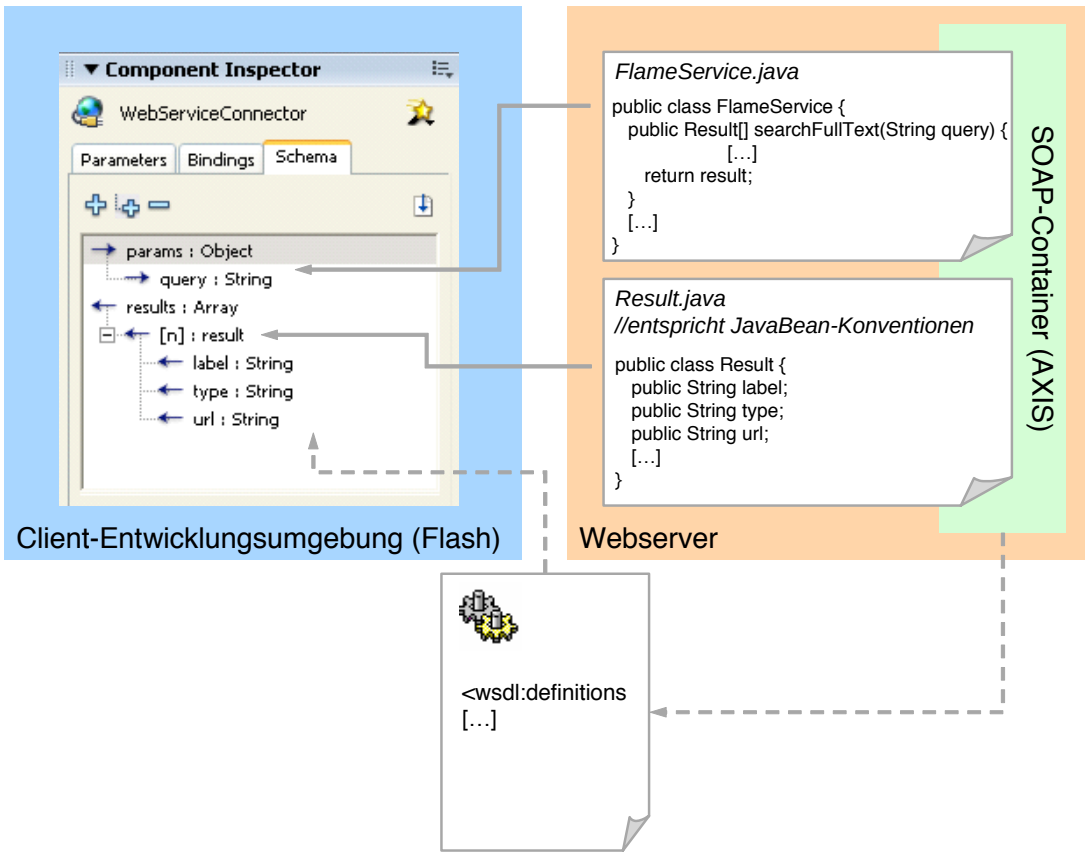


Abbildung 3.18.: Webservices – zentrale Schnittstellentechnik in Flame

des Frameworks, in Vorbereitung auf die Entwicklungsarbeit der Fachexperten, Komponenten abzuleiten, die keiner technischen Abstraktionen mit Begriffen wie `type` oder `param` mehr bedürfen. Der Fachentwickler erhält eine auf den Kontext zugeschnittene Suchkomponente, mit der er entsprechend der für den Einsatz definierten Gestaltungsspielräume disponieren kann. Im elementarsten Fall, kann er die Verfügbarkeit bestimmen. Es kann aber auch ein Teil der fachlichen Arbeit sein, die inhaltliche Struktur der Ergebnispräsentation festzulegen – etwa durch die Vorauswahl von Zielkategorien. So dass beispielsweise in einem fachlichen Kontext, der die Aufmerksamkeit des Nutzers auf Textmedien lenken soll, die Suchergebnisse ausschließlich aus Textquellen bestehen – Inhalte entsprechend gefiltert werden können.

3.2. Das Vorgehensmodell

In den folgenden Abschnitten wird nun beschrieben, welches Vorgehen einerseits nötig ist, um die Fähigkeiten des Frameworks ausnutzen zu können. Es ist aber auch ande-

rerseits das Vorgehen und die Unterstützung durch das Framework, was die Probleme der adressierten Entwicklungskontexte löst. Also nur das Zusammenspiel aus dem Framework und dessen gezielte Anwendung erlauben eine Effektivierung des Entwicklungsgeschehens in transdisziplinären Projekten.

Den Ausgangspunkt der Beschreibung bildet eine Charakterisierung idealtypischer Projekte nach dem Schema des Extreme Programming entlang von Werten, die während der Arbeit stets zu wahren sind, Prinzipien und Verfahren, die die Aufrecherhaltung des Wertesystems determinieren. Nachdem die Struktur des Prozesses zusammengefasst wurde, werden die beiden zentralen Phasen des Vorgehens eingehend erörtert. Insbesondere die Auswirkungen auf das Projektmanagement werden bezüglich relevanter Aspekte untersucht.

3.2.1. Werte, Prinzipien, Verfahren – ein Überblick über die Methodik

Das konzeptionelle Fundament der Methodik soll in Anlehnung an die in XP [Bec00] bzw. bei Agile Modelling [Amb02] etablierten und in Abschnitt 2.3.1 ab Seite 61 umrissenen Kategorien *Werte, Prinzipien, Verfahren* gelegt werden. In Analogie zur Definition einer Methodik zur agilen Wissenserzeugung [Aue06] – also einer vergleichbaren Problemdomäne – sollen dabei die jeweiligen Konzepte für die Spezifik des Problemkontexts ausgewählt, adaptiert oder ergänzt werden.

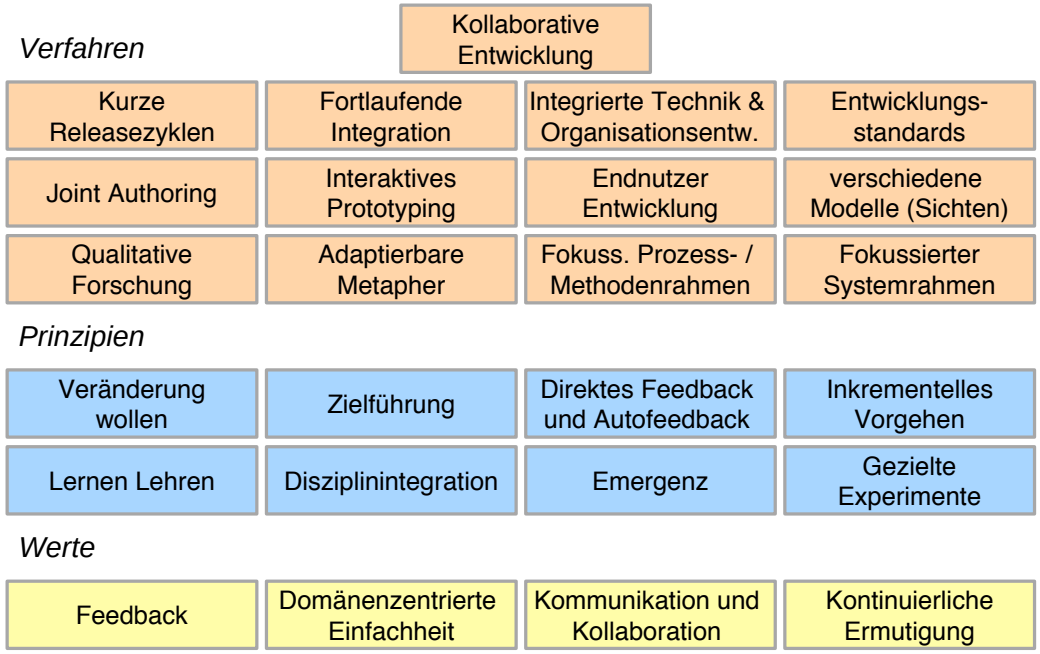


Abbildung 3.19.: Der Aufbau der Methodik in Anlehnung an die Definition von XP und die Darstellung von [Aue06]

Die Abbildung 3.19 gibt zunächst einen Überblick über die Gesamtkonzeption der Methodik. Die *Werte* bilden das Fundament der Vorgehensweise, indem sie einen Rahmen mit Kriterien abstecken, innerhalb dessen Projekte gesteuert werden sollen. Die darauf aufbauenden *Prinzipien* machen dann konkretere Maßgaben, die das Handeln der einzelnen Projektbeteiligten bestimmen sollten, um das Projekt innerhalb der Parameter der Wertvorstellungen halten zu können. Dieser noch immer sehr allgemeine „Kodex" der Prinzipien wird schließlich mit Vorschlägen adäquater *Verfahren* so weit konkretisiert, dass das Gesamtsystem der Methode im Zusammenspiel mit dem Framework relativ direkt auf die Bedürfnisse individueller Projektsituationen übertragen werden kann.

Werte

Die im Folgenden beschriebenen Werte sind im Wesentlichen Adaptionen der Werte, die die „Mentalität" von XP-Projekten [Bec00] prägen sollten. Dementsprechend „erben" diese Werte Maßgaben und Konsequenzen von XP weitgehend. Erläutert sind lediglich die Spezifika der Adaption.

Kommunikation und Kollaboration: Der Wunsch nach systematischem Austausch der Projektbeteiligten zur Vermeidung von Missverständnissen und zur Schaffung eines gemeinsamen Verständnisses für Motive, Vorgehensweisen und Zielstellungen auch über Disziplingrenzen hinweg soll hier verschärft werden zur Forderung nach direkter Zusammenarbeit an gemeinsamen Artefakten. Durch echte kollaborative Entwicklung lässt sich der intensivste Austausch zwischen den Beteiligten etablieren – eine nachhaltige Konvergenz der Disziplinen erreichen. Die permanente Präsenz des Diskursgegenstands bildet die Basis des angestrebten diskursiven Prozesses, in dessen Rahmen Lösungen emergent werden können.

Dieser Ansatz erlaubt es den Technikern über die Grenzen der verbalen Kommunikation hinaus, das benötigte Verständnis gerade auch für habitualisierte, aus langer Erfahrung heraus unbewusst betriebene, sehr komplexe Abläufe der Zieldomäne zu entwickeln. Umgekehrt erfahren Domänenexperten so sehr direkt, welche technischen Möglichkeiten und Beschränkungen es gibt. Das geschieht durch das machen direkter gemeinsamer Handlungserfahrungen und die wechselseitige Beobachtung.

Feedback Rückkopplung ist die treibende Kraft des Entwicklungsansatzes auf mehreren Ebenen. Zunächst wirkt sie im Sinne der Idee von XP und ist die Grundlage der Projektsteuerung. Durch die kontinuierliche Arbeit am Zielartefakt und die so fortlaufend entstehenden Prototypen, ist der Zustand des entstehenden Zielsystems jeder Zeit am System selbst ablesbar. Risiken können so einfach erkannt und die Wirkung entsprechender Interventionen unmittelbar bewertet werden.

Durch die direkte Einbeziehung von Nutzern und Domänenexperten ist das Feedback zwischen den fachlichen Anforderungen und den technischen Notwendigkeiten sichergestellt. Der permanente Diskurs zum Gegenstand des gemeinsam entwickelten Systems

– im Wechselspiel der Sichtweisen und Ideen zwischen den Disziplinen und in der kontinuierlichen Rückkopplung zwischen der Entwicklungstätigkeit und den unmittelbar sichtbaren Ergebnissen – ist die Grundlage der Emergenz von Lösungen.

Bezogen auf den Entwicklungsprozess bedeutet das, dass eine Konvergenz der unterschiedlichen Vorgehensweisen möglich wird. Durch eine permanente Orientierung an den Tätigkeiten der Anwendungsdomäne lässt sich zunächst der traditionell technologisch geprägte Entstehungsprozess, an die Produktionsprozesse der Zielgruppe anpassen. Im zweiten Schritt lässt sich der Prozess, unterstützt durch ein entsprechendes Instrumentarium (domänenspezifische Entwicklungsumgebung) auch auf den eigentlichen Anwendungskontext ausweiten. So wird es dem Endnutzer möglich in Rückkopplung zwischen den Eigenschaften des Systems und seinen individuellen Bedürfnissen, Optimierungen an der Software vorzunehmen.

Domänenzentrierte Einfachheit: Erfahrungen aus typischen Projekten lassen die Hypothese zu, dass die Kluft zwischen dem technisch Machbaren und den Anforderungen der Zieldomäne in der Praxis durch drei Bedingungen vertieft wird:

- Die technologischen Möglichkeiten sind aus der Sicht der IT-Fachleute sehr Umfangreich und die Ausnutzung neuester Technologien erscheint pauschal erstrebenswert.
- Dem entgegen stehen Anwendungsfelder mit ebenso tradierten wie durch die lange Tradition und Entwicklung komplexen Aufbaustrukturen und Abläufen.
- Die Grenzen der Technologie werden aber gerade auch in elementaren Bereichen erreicht – wie etwa der Sprachverarbeitung. Die Vermittlung dieser für den Techniker unumstößlichen Beschränkungen an technische Laien, der die technologische Abbildbarkeit einer aus seiner Sicht simplen Anforderung voraussetzt ist sehr konfliktträchtig.

Zur Überwindung der Kluft ist es also wichtig, zunächst immer lediglich nach der einfachsten Lösung zu suchen, die die Bedürfnisse der Anwender befriedigt. Wobei permanent hinterfragt werden muss, für welche Aspekte der Zieldomäne der Technologieeinsatz tatsächlich eine Effektivierung bringen kann. Die Auswahl der Aspekte soll ausschließlich getrieben werden von fachlichen Erwägungen – lediglich unter Berücksichtigung der technischen Machbarkeit. Die fachbezogene Einfachheit der umzusetzenden Features leistet auch einen entscheidenden Beitrag zum Abbau von Berührungsängsten seitens der einzubeziehenden Fachentwickler und zur Fokussierung auf die Fachlichkeit. Der Framework-Einsatz hilft beim Abstrahieren von „abschreckenden“ technisch komplexen Details und zeitaufwendige Routineaufgaben der Software-Entwicklung. Die Aufrechterhaltung dieses Wertes wird zum entscheidenden Kriterium dafür, ob der sich weitgehend selbst organisierende Prozess der Entwicklung, für den es kaum möglich ist a priori Ziele zu definieren, zu einem für alle befriedigenden Ergebnis führt.

Kontinuierliche Ermutigung: Mut, insbesondere im Umgang mit Veränderungen des Arbeitskontextes und bei der Aufgabe von Besitzständen, lässt sich nur durch einen behutsam forcierten Prozess der Gestaltung der Projektkultur etablieren. Die

Berührungsängste oder zumindest Vorbehalte der Vertreter nichttechnischer Disziplinen mit dem Einsatz von Informationstechnologie können nur überwunden werden, indem sie kontinuierlich ermutigt werden, die Technologie als Gestaltungsmittel zu entdecken und zu nutzen. Den Ansatzpunkt dazu muss eine systematische Auseinandersetzung mit den Belangen der Fachdomäne bilden, um die initiale technologische Arbeitsgrundlage möglichst gut an die Bedürfnisse der Tätigkeiten von Autoren und Redakteuren anzupassen. Damit müssen etwa die grundlegende Hemmschwelle gesenkt und Anreize für das Experimentieren geschaffen werden. Die tatsächlichen Potentiale des Technologieeinsatzes müssen darüber hinaus sowohl im permanenten Diskurs der kollaborativen Arbeit als auch ggf. in gezielten Schulungen verdeutlicht werden. Gerade in Disziplinen mit einer gewissen methodischen Tradition wird es aber eine Herausforderung sein, den Wert des Veränderungswillens plausibel zu machen.

Der Erfolg eines Projekts wird nun davon abhängen, inwieweit man in der Lage ist durch die Einhaltung der im folgenden Abschnitt beschriebenen Prinzipien und die Anwendung der Verfahren das Projekt so zu steuern, dass diesen Werten möglichst optimal entsprochen werden kann.

Prinzipien

Die Prinzipien konkretisieren die Richtschnur der Handlungsoptionen der Projektbeteiligten. Sie geben also vor, wie man grundsätzlich zu agieren hat, um den eben beschriebenen Werten genügen zu können. Auch hier ist das Gerüst der Konzepte angelehnt an XP nach [Bec00], Agile Modeling nach [Amb02] sowie RapidOWL nach [Aue06].

Lernen Lehren: Initial muss zunächst die Basis geschaffen werden, um den selbstreflexiven Prozess des interaktiven Prototyping, getrieben durch technikferne Akteure, anzustoßen. So soll es den Entwicklern durch die für sie individualisierte und adaptierbare Entwicklungsumgebung ermöglicht werden, in ständiger Rückkopplung zwischen Arbeitsanforderungen, erkannten eignen Kompetenzdefiziten sowie Verbesserungspotentialen am Instrumentarium und entsprechender selbständiger Intervention, einen informellen Lernprozess voranzutreiben. Dieser Selbstlernprozess sollte zentral auf intrinsischer Motivation beruhen – quasi auf systematisch geschürter Entdeckungslust, bei der aus Erfahrungen der aktuellen Arbeit direkt Konsequenzen für den Weiterentwicklungsprozess gezogen werden können. Neben der initialen Etablierung der passenden instrumentellen Rahmenbedingungen muss durch kontinuierliches Coaching sichergestellt werden, dass der Arbeitsprozess als wechselseitiger informeller Lernprozess fungieren kann. Dazu muss den Fachleuten mit Rücksicht auf individuelle Bedürfnisse (also den Lernertyp) vermittelt werden, wie man sich in die Nutzung technischer Hilfsmittel einarbeitet, welche Informationsressourcen genutzt werden können. Es muss vorrangig durch intensive gemeinsame Arbeit am Zielartefakt und die Kommunikation dazu verdeutlicht werden, wie man sich Gestaltungs- und Optimierungspotentiale erschließt und diese ausnutzt. Der tätigkeitsorientierte Prozess wird zur bewussten wechselseitigen Lernerfahrung zwischen den Disziplinen. Es gilt Frustration zu vermeiden und die Möglichkeit realer Erfolgserlebnisse zu schaffen.

Disziplinintegration: Die problematischsten Reibungsverluste während der Entwicklung innerhalb von bezüglich der Disziplin heterogener Teams ergeben sich aus den unterschiedlichen Begriffswelten, Methoden, Werkzeugen und Lösungsstrategien. Insbesondere die Unvereinbarkeit der fachlichen Bedarfe bzw. Bedürfnisse der Zieldomäne mit den für Erarbeitung einer medialen oder informationstechnologischen Unterstützung nötigen Abstraktionen muss als Problem bewusst gemacht und überwunden werden. Die Technologie tritt dabei zwar idealerweise als Vehikel der fachlichen Gestaltungsleistung in den Hintergrund. Die direkte und vor allem aktive Auseinandersetzung der technikfernen Domänenexperten mit den technischen Aspekten der Gestaltung und die Einbeziehung in die entsprechenden Entwicklungsaktivitäten ist jedoch eine unvermeidbare Anforderung an den Gestaltungsprozess des Entwicklungsumfeldes.

Direktes Feedback und Autofeedback: Eine möglichst direkte Rückkopplung zwischen den erzielten Entwicklungsergebnissen und den Anforderungen an das Zielsystem muss im Bereich interaktiver Anwendungen primär auf der Grundlage subjektiver Kriterien etabliert werden. Die Tragfähigkeit einer didaktischen oder inhaltlichen Konzeption ist beispielsweise nicht formal zu spezifizieren und lässt sich damit kaum über automatisierte Tests prüfen. Mit der Einbindung von Nutzervertreten in den Prozess der Umsetzung und der Orientierung der Entwicklungsprozesse an den Tätigkeiten und Begriffen der Zieldomäne und damit der unmittelbaren Verzahnung von Entwicklung und Nutzung ist quasi im „Kurzschluss“ sichergestellt, dass die Ergebnisse den Vorstellungen der Nutzer entsprechen. Durch die permanente moderierte Reflektion der nichttechnischen Entwicklungsbeteiligten über ihre aktuell ausgeübten Tätigkeiten und deren direkt sichtbare Konsequenzen für das Zielsystem entstehen sehr kurze Autofeedback-Zyklen. In deren Rahmen entwickeln sich der Erfahrungshorizont der Entwicklungsbeteiligten und in der Konsequenz dessen Anforderungen an die Software und schließlich auch Ausprägungen von Werkzeugen und Zielsystem aufeinander zu.

Gezielte Experimente: Die unmittelbare Überprüfung konzeptioneller Überlegungen durch prototypische Implementierungen ist bei den bezüglich des Technologieeinsatzes weitgehend unerschlossenen Fachgebieten unabdingbar. Nur so ist es möglich, die Potentiale der Technologie im Verhältnis zu den Notwendigkeiten der Domäne transparent zu machen. Interaktives Prototyping ermöglicht die unmittelbare Umsetzung und Evaluierung von Ideen. Die gewonnen Erkenntnisse können direkt in einen Zyklus der Korrektur münden. Über die auf diese Weise betriebene Reduzierung technischer Risiken hinaus können so auch technikferne Domänensachverständige ab einem gewissen Punkt selbstständig die konzeptionelle Tragfähigkeit Ihrer Ideen Überprüfen und sich vor allem explorativ Gestaltungsspielräume erschließen.

Inkrementelles Vorgehen: Es liegt in der Natur der hier betrachteten Projekte, dass die vollständige Planung und a priori Konzeption des Ergebnisses einerseits und der Projektkonstellation andererseits kaum möglich ist. Veränderungen der Weiterentwicklung des Zielartefaktes selbst, aber auch die dazu nötigen Änderungen des Arbeitskontexts, etwa des Produktionsprozesses oder der Kompetenz- bzw. Tätigkeitsprofile der Teammitglieder, können nur in kleinen Schritten vollzogen und beherrscht werden. Nur so ist es möglich, systematisch die Grenzen des Machbaren auszuloten – sowohl bezogen auf

die Möglichkeit der Einbindung der Nichttechniker, als auch bezogen auf die technisch medialen Gestaltungsspielräume. Insbesondere der evolutionäre Charakter der Entwicklung am Zielsystem bedingt das im Anschluss besprochene Prinzip der Emergenz. Dieses Prinzip ist wiederum die Voraussetzung für die Erarbeitung von Lösungen, die sich im Sinne der Evolution innerhalb eines Diskurses zwischen den Disziplinen durchsetzen. Lösungen beziehen sich dabei wiederum auf das eigentliche Artefakt und das Arbeitsumfeld der Entwicklung – die eingesetzten Vorgehensweisen und den Instrumenteneinsatz.

Neben dem angestrebten qualitativen Reifungsprozess des Systems und des Entwicklungskontexts, ist der Prozess in der Regel auch inkrementell evolvierend zu gestalten. Durch entsprechende Mittel der Dekomposition müssen überschaubare Teilaspekte des Systems möglichst isoliert bearbeitet werden können, um die eigentliche Komplexität zu bewältigen und die Behutsamkeit zu etablieren, der es in der Regel bedarf, um Berührungsängste im Umgang mit der Technologie zu überwinden.

Emergenz: Die Nichtvorhersehbarkeit der Ergebnisse bzw. die Nichtlinearität der Entwicklungsprozesse sind unabänderliche risikobehaftete Charakteristika der hier betrachteten Projekte. Es ist also sinnvoll, die damit verbundenen Potentiale auszunutzen und zu instrumentalisieren, um die nahe liegenden Risiken zumindest zu kompensieren. Dazu sind die Rahmenbedingungen der Entwicklung so zu gestalten, dass die Erstellung der Software weitgehend ohne bewusste Steuerung, eigendynamisch (etwa im Sinne von [Ste04] Seite 65) ablaufen kann. Die von [Ste04] beschriebene Übertragung des Begriffs des *Emergent Design* auf die Software-Entwicklung vertraut dabei auf den induktiven Charakter agiler Entwicklungsprozesse. Diese schaffen es, ausgehend von elementaren Schritten, während der jeweils kleine Inkremente entstehen, das Gesamtsystem zu entwickeln, ohne einen umfassenden Plan zugrunde legen zu müssen. Erst das Zusammenspiel vieler kleiner Ideen und Aktivitäten führt zur Formierung des Gesamtsystems. Das Spannungsfeld zwischen den Disziplinen, welches durch die direkte Kooperation entsteht, bringt Lösungen hervor, die isoliert arbeitende Disziplinen nicht in der Lage wären zu erzielen. Diese intuitive Einsicht muss im Hinblick auf die *Brandmauer der Informatik* explizit durch entsprechende Verfahren als Teil der Entwicklungskultur etabliert werden. Das gemeinsame Artefakt des Prototypen dient dabei der Bündelung und Fixierung des emergent entstehenden fachlichen und technischen Projektwissens.

Zielführung: Allein das Vertrauen in das eben beschriebene Prinzip der Emergenz reicht für die Arbeit im produktiven, industriellen Maßstab nicht aus. Ein gewisser Determinismus bezüglich der Konvergenz des emergenten Prozesses in Richtung der globalen Projektziele muss durch gezielte Interventionen der Moderation, der Personal- und der Organisationsentwicklung und damit die dynamische Korrektur der Rahmenbedingungen sichergestellt werden. Das Framework mit seinen definierten Schnittstellen zu seiner Instanziierung und der inhärente Sockel einer Lösung tragen entscheidend dazu bei, die Entwicklungsleistungen zu kanalisieren. Ein Abdriften in einen konzeptionellen Wildwuchs der Ideen, verbunden mit entsprechenden technischen Risiken ist damit beispielsweise a priori erschwert. Die Domänenfixierung des Frameworks auf interaktive

direkt reaktive Systeme und die Vorgabe von Architektur und Funktionalitäten leistet umgekehrt einen Beitrag zur technologischen Fokussierung auf Projektziele.

Veränderung wollen: Gerade in etablierten wissenschaftlichen Disziplinen mit tradierten Arbeits- und Präsentationsformen, deren Effizienz sich über Jahrhunderte ausgebildet hat, ist es schwierig, neue Perspektiven und Potentiale aufzuzeigen oder gar Defizite plausibel zu machen. Die Erkenntnis, dass positive Erfahrungen aus Erfolgserlebnissen im Arbeitsalltag eine Öffnung für Veränderungen bewirken können wird umgesetzt in die behutsame direkte Konfrontation mit der Technologie. Durch die Anregung zu permanenter Reflektion über die eigene Tätigkeit und in der Rückkopplung der Evaluation der erzielten Ergebnisse schließlich die Optimierung der Tätigkeit und der unterstützenden Mittel lässt sich ein kontinuierlicher Fluss des Umdenkens und Überarbeitens in Gang halten. Wichtig ist dabei, dass Veränderungen sowohl am Artefakt als auch im Arbeitsumfeld kontrollierbare Erfolge zeitigen und nicht vordergründig Risiken und Kosten induzieren. Das heißt etwa, dass eine Veränderung am System jeder Zeit mit geringem Aufwand umsetzbar und der entsprechende Effekt überprüfbar sein muss. So kann etwa eine über das Meta-Object-Protocol (MOP) durchgeführte Veränderung direkt am laufenden integrierten System getestet werden. Das MOP stellt technologisch sicher, dass Inkonsistenzen, die möglicherweise das gesamte System gefährden nicht entstehen können.

Verfahren

Um nun schließlich gemäß der aufgestellten Prinzipien handeln zu können und damit den Werten zu genügen, bedarf es konkreter Praktiken bzw. Verfahren, deren Anwendung die Projektarbeit mehr oder minder zwangsläufig entsprechend steuern.

Qualitative Forschung: Um die Basis für eine echte Zusammenarbeit zu schaffen, muss zumindest ein fundamentales wechselseitiges Verständnis für die Belange der jeweils kooperierenden Disziplinen geschaffen werden. Da es dabei zu großen Teilen auch um kaum explizierbare Nuancen in den Sicht- und Handlungsweisen geht, sind herkömmliche Methoden der Analyse und der Spezifikation nur begrenzt geeignet um insbesondere den Entwicklern der Technologie die komplexen Zusammenhänge der zu repräsentierenden Domäne zu vermitteln. Die möglichst genaue Erfassung der Tätigkeitsmuster der Fachexperten ist eine entscheidende Voraussetzung für den Erfolg des ersten Aufschlags der Ausprägung von Entwicklerwerkzeugen als Grundlage der anschließenden evolutionären interaktiven Verfeinerung. Das Gelingen dessen bestimmt über die Motivation der potentiellen technikfernen Entwickler, sich mit neuer Technologie in ihrem Arbeitsalltag auseinanderzusetzen. Um das zu erreichen ist es sinnvoll, sich der Methoden der empirischen Feldforschung zu bedienen.

Aus der teilnehmenden Beobachtung heraus wird es möglich, auch feinste habitualisierte Routinetätigkeiten der Akteure der Zieldomäne zu erkennen. Dazu versucht sich der Analytiker – also in der Regel der Entwickler der Technologie, der für die Anwendung des Frameworks verantwortlich ist – in die bestehenden Arbeitsabläufe der Zieldomäne

einbinden zu lassen und die dort anstehenden Aufgaben zu erfüllen. Das sich ergebende umfassende Bild wird unmittelbar reflektiert am Verständnis für die technologischen Potentiale. Über einfache erste Prototypen (Mock-Ups) wird ebenfalls direkt im laufenden Produktionsprozess evaluiert, ob diese ersten Ideen tatsächlich tragfähig sein können. Diese Praxis und die Erfahrung damit während der Evaluation des Ansatzes werden später ausführlicher erläutert.

Adaptierbare Metapher: Aus technologischer Sicht wurde diese Herangehensweise in Abschnitt 3.1.4 erläutert. Die Etablierung einer Metapher als gemeinsamen Nenner des Verständnisses innerhalb des Projekts wird damit in den individualisierten Nutzer-Oberflächen der Entwicklungsumgebung durch entsprechende Adaptionen des Frameworks materialisiert. Neben einer Metapher, die den Austausch über das Zielartefakt selbst unterstützt, ist es sinnvoll auch den Entwicklungsprozess und die damit verbundenen Organisationsstrukturen begrifflich an ein gemeinsames Verständnis anzupassen. Sind in der Zieldomäne etwa üblicherweise Redaktionsaufgaben, wie das Korrekturlesen von Druckfahnen zu erledigen, sind die äquivalenten Schritte der Arbeit an der Software sinnvollerweise entsprechend zu bezeichnen und durch entsprechende Rollen zu verantworten. Für das evaluierte Projekt bedeutet das, dass man davon spricht, dass der für eine bestimmte Epoche zuständige Redakteur Themen und Schwerpunktthemen korrigiert. Im Gegensatz dazu dass, ein Testverantwortlicher Blackboxtests an Modulen vornimmt – wenngleich Intention und Tätigkeit die grundsätzlich gleichen sind: das visuelle analysieren der Bildschirminhalte und deren Überarbeiten im WYSIWYG-Modus.

Fokussierter Prozess- und Methodenrahmen: Die Konzentration auf das Wesentliche schafft der hier vorgestellte Ansatz durch die permanente Orientierung an den innerhalb der Zieldomäne zu unterstützenden und etablierten Tätigkeiten. Dazu wird der zunächst allgemein als interaktives Prototyping bezeichnete methodische Ansatz möglichst früh so adaptiert, dass die Interaktion während der Arbeit ausgerichtet ist an den Abläufen der Zieldomäne. Dazu werden einerseits Tätigkeiten und die entsprechende Werkzeugunterstützung angepasst (siehe dazu auch das nächste Verfahren). Andererseits werden auch die Abläufe der Arbeit insgesamt so gestaltet, dass sie den innerhalb der Domäne etablierten möglichst nahe kommen. Das heißt auch übergeordnete Prozesse und Rollenverteilungen von Akteuren werden bei der Etablierung des Entwicklungsvorgehens aufgegriffen. So werden beispielsweise Redaktionsprozesse mit den Aufgaben der Software-Entwicklung hinterlegt – die Zusammenstellung und Gliederung der Inhalte als klassische Redaktionsaufgabe impliziert etwa die Erstellung von Navigationsmustern.

Fokussierter Systemrahmen: Um den Technologieeinsatz so einfach wie möglich gestalten zu können, also die Komplexität von vorn herein möglichst zu begrenzen und die Beherrschbarkeit auch aus der Sicht der Domänenexperten hoch zu halten, werden im wesentlichen zwei Ansätze verfolgt. Das erfolgt zum einen, durch die Festlegung von innerhalb der Domäne funktional invarianter Softwarekomponenten (fixiert im Framework) und deren Wiederverwendung in verschiedenen Kontexten. Die Komplexität der Komponenten ist damit reduziert auf ihre Schnittstellen. Aufwände zu ihrer Entwicklung und Konsolidierung haben nach einer begrenzten Anzahl wiederholter Einsätze kaum noch Einfluss auf die Projektressourcen.

Zum anderen wird die Anwendung der Entwicklungswerkzeuge im Verhältnis zu den Kompetenzen der Projektbeteiligten dadurch vereinfacht, dass sie dezidiert an die damit zu erfüllenden Aufgaben angepasst werden. Das Agile-Modeling-Verfahren *Nutze einfachste Werkzeuge* [Amb02] wird also erweitert zur Praktik *Nutze einfachste, aufgabenspezifische Werkzeuge*. Dazu gehört auch, dass in den frühen Phasen des initialen Auslotens der Domänenspezifika zur Unterstützung der Antizipation der fachspezifischen Entwicklerschnittstellen einfache Mock-Ups zum Einsatz kommen. Mit Hilfe des Frameworks und der Idee der adaptierbaren Entwicklungsmetapher wird es schließlich möglich, die eigentlichen Werkzeuge der Umsetzung nicht nur so einfach wie möglich zu gestalten, sondern auch direkt auf die anstehenden Aufgaben auszurichten.

Kollaborative Entwicklung: Die von Beck geforderte *gemeinsame Verantwortlichkeit* für den Quelltext, verbunden mit der Forderung nach Aufgabe von Besitzständen und weit reichender Transparenz im Aufbau des Codes, so dass er tatsächlich allen zugänglich ist, wird hier verallgemeinert. Einerseits soll sich die gemeinsame Verantwortlichkeit nicht mehr nur auf die Ebene des Quellcodes beziehen, sondern ausgeweitet werden auf das Zielartefakt im Ganzen. Die Verantwortlichkeit soll andererseits nicht allein zwischen den Technologieentwicklern geteilt werden, sondern zu großen Teilen auch auf die Domänenexperten ohne Technikhintergrund und Endanwender übergehen. Möglich wird dies durch den Einsatz von Entwicklerschnittstellen und damit quasi Entwicklungsstandards (MOP), die durch alle Beteiligten beherrscht werden können. Der aktuelle Stand der Entwicklung wird allen zudem über die permanente Integration eines lauffähigen Zielsystems transparent gemacht. So können zum Beispiel auch die nach der Phase der inhaltlichen Konzeption ansonsten weitgehend isoliert arbeitenden Redakteure eines beauftragenden Fachverlags nicht nur jederzeit den Projektfortschritt überblicken, sondern ihre Entwicklungsbeiträge in direkter Verzahnung mit allen Disziplinen am „lebenden System“ leisten und auch in die Ergebnisse anderer, etwa der Autoren, eingreifen. Die direkte Einbeziehung potenzieller Nutzer in den Prozess der Systemausgestaltung im Sinne von *End-User-Development* ersetzt die Forderung nach der Anwesenheit eines End-Anwenders (XP-Verfahren: „Kunde vor Ort“).

Interaktives Prototyping: Die augenscheinliche Schwierigkeit bei der direkten Einbeziehung aller Beteiligter in den Systementwicklungsprozess, die sich aus den Differenzen in den Sicht- und Arbeitsweisen zwischen den Disziplinen ergeben, wird durch eine sukzessive Annäherung auf der ebene der Tätigkeiten adressiert. In einem experimentellen Prozess erkunden die Entwickler der Technologie einerseits die fachlichen Bedarfe der Domänenexperten. Umgekehrt erschließen sich die ursprünglich nicht technisch orientierten Fachleute die Mittel und Potentiale der Technologieabbildung, der ihnen vertrauten Strukturen und Abläufe. Die Basis für den Erfolg dieses Prozesses besteht einerseits in der intensiven Kommunikation innerhalb echter kooperativer Arbeitsprozesse. Andererseits ist es wichtig die Grenzen des Einsatzes technischer Abstraktionen als Vehikel der Systemgestaltung auszuloten.

Angestrebt wird eine visuelle Repräsentation der Entwicklungsmetapher, die sich an der Zieldomäne orientiert, und die damit bezüglich der Interaktion während der Gestaltungstätigkeit durch den Domänenexperten intuitiv beherrschbar ist – sich idealerweise

in angestammte Arbeitsprozesse einbetten lässt. Eine möglichst präzise Angleichung wird möglich durch die permanente Rückkopplung zwischen dem Einsatz der Mittel und ihrer Überarbeitung im Sinne einer stetig optimierten Effektivität für den Nutzer. Die Anpassung und Weiterentwicklung der Werkzeuge und parallel dazu des Zielartefakts selbst erfolgt im Rahmen der kollaborativen Entwicklung zunächst primär unterstützt bzw. angeleitet durch technologisch Sachverständige. Angestrebt wird jedoch eine weitgehend selbständige Handlungsfähigkeit der rein fachlich inhaltlich getriebenen Entwicklung. Fachleute sollen also nach und nach in die Lage versetzt werden, Defizite sowohl am Instrumentatrium (den Entwicklungswerkzeugen) als auch an der Art ihrer Verwendung innerhalb des Prototyping-Prozesses erkennen und korrigieren können. Auch hier ist die weitgehende fachliche Kongruenz der Interaktionsformen der Werkzeuge der fachlichen Systemausgestaltung mit denen der Nutzung hilfreich. Also ähnelt beispielsweise die Art wie Historiker im Entwicklungskontext eine Stoffsammlung erstellen in der Regel den Arbeitsabläufen, die im Fach Geschichte vermittelt werden müssen. Die unterstützten Tätigkeiten der Zieldomäne werden so systematisch, explorativ durch die Interaktion während der Entwicklung in das System abgebildet.

Endnutzer Entwicklung: In direkter Anknüpfung an die Praktik des interaktiven Prototyping (sowohl zeitlich als auch konzeptionell) fungiert tatsächliches End-User-Development als Instrument zur Feinabstimmung der Software auf individuelle Nutzerbedürfnisse. Dabei handelt es sich um die Ausweitung der Beteiligung nicht technisch fokussierter Fachleute an der Entwicklung auf die selbständige Weiterentwicklung für die partikulären Bedarfe der Nutzung durch die Endanwender selbst. Ermöglicht wird dies durch die angesprochene weitgehende Konformität von Einsatzkontext und einsatznahem Entstehungskontext. Die Mittel der Interaktion, die auf die fachlich orientierte Entwicklungsarbeit der Domänenexperten evolutionär abgestimmt ist, sind Anwendern mit dem gleichen fachlichen Hintergrund in gleicher Weise zugänglich. Der fließende Übergang zwischen den Mitteln der eigentlichen Anwendung und denen der Weiterentwicklung kann so genutzt werden, um den Prozess der Wartung und Pflege der Software ein Stück weit in die Verantwortung des Anwenders zu übergeben. Je nach Produktstrategie können dem Endanwender alle Möglichkeiten zur Veränderung der Software – das Meta-Objekt-Protokoll in seiner domänenspezifischen Ausprägung – zur Verfügung gestellt werden.

Entwickle verschiedene Modelle (Sichten) gleichzeitig: Um auf die unterschiedlichen Sichtweisen eingehen zu können, die sich aus der direkten Beteiligung unterschiedlicher Disziplinen ergeben ist es im Sinne der Konvergenz wichtig, die daraus resultierenden unterschiedlichen Aufgaben und Tätigkeiten auch durch unterschiedliche Sichten auf das Artefakt zu unterstützen. Bezogen auf die frühen konzeptionellen bzw. sondierenden Phasen der initialen Ausprägung des Frameworks bedeutet dies etwa, dass Mock-Ups so gestaltet werden, dass individuelle Begriffsbildungen und Abstraktionen der beteiligten Disziplin genutzt werden.

In der Fortführung für die Ausprägung der Nutzerschnittstellen für das interaktive Prototyping bedeutet das, dass den unterschiedlichen Aufgabenbereichen, jeweils die passenden Sichten auf das Entwicklungsartefakt mit jeweils aufgabenangemessener Funktio-

nalität – Interaktivität und Metaphorik der Nutzerschnittselle – zur Verfügung gestellt werden. Das MOP koordiniert dabei die eigentlichen Modifikationen am Zielsystem. So können gleichzeitig unter einer fachsystematischen Sicht Inhalte eingepflegt und strukturiert werden; während unter einer anderen, etwa der WYSIWYG-Sicht, das Layout bearbeitet werden kann. Dabei erlaubt das Framework jeder Zeit unmittelbare Wechsel zwischen den Sichten – etwa von der inhaltlich konzeptionellen Anlage der Gesamtanwendung in den WYSIWYG-Modus – da alle Werkzeuge über das Meta-Objektprotokoll das singuläre Artefakt manipulieren. So können die Auswirkungen der Arbeit unter allen nötigen Perspektiven gleichzeitig evaluiert werden. Screendesigner können beispielsweise sofort sehen, wie sich ein vom Fachautoren eingepflegtes Medienelement in das Erscheinungsbild eines Bedien-Panels einfügt. Umgekehrt können Entscheidungen, die das Screendesign trifft direkt Einfluss auf die Arbeit der Fachautoren nehmen. Über das MOP können etwa Regeln abgebildet werden, die eine bestimmte Gliederungsstruktur vorgeben.

Kurze Release-Zyklen: Um die Tragfähigkeit der Software-Lösung und insbesondere die Erfüllung der Nutzeranforderungen möglichst früh im Projekt sicherstellen zu können, ist es laut XP wichtig, das System so in Inkremente zu dekomponieren, dass Teile innerhalb kürzester Zeit fertig gestellt, veröffentlicht und in der Praxis eingesetzt werden können. Durch den Einsatz des Frameworks steht nach der initialen Ausprägung und der Wahl der Basisarchitektur bereits ein grundsätzlich lauffähiges System zur Verfügung. Die inkrementelle Verfeinerung im Rahmen des interaktiven Prototyping führt jeweils zu Versionen, die direkt in die Nutzung überführt werden. Es entspricht dem Wesen des interaktiven Prototyping, dass Nutzungs- und Gestaltungskontext weitgehend verschmelzen. Allein die Tatsache, dass Nutzervertreter und in der letzten Konsequenz im echten End-User-Development auch Endnutzer selbst die Zyklen der Gestaltung und des evaluierenden Anwendens mit tragen, garantiert von Beginn an Release-Stände, die Anforderungskonform sind.

Fortlaufende Integration: Die technische Voraussetzung für das eben beschriebene Verfahren ist die permanente Verfügbarkeit einer im Ganzen ablauffähigen Systemversion. Im Gegensatz zur isolierten Entwicklung von Einzelmodulen, deren Einbindung in die Gesamtarchitektur immer mit Risiken behaftet ist, wird jeder Entwicklungsschritt direkt am zentralen, framework-basierten Artefakt ausgeführt. Die nötige Modularität und die dementsprechende Entkopplung disjunkter Systemteile schafft das MOP implizit durch die Generierung der Meta-Level-Objekt-Struktur. So können das System und die damit verbunden Entwicklungsaufgaben zwar zerlegt und verteilt werden, aber jeder Entwicklungsschritt besteht aus der Konkretisierung des Frameworks und impliziert damit die Einbindung in den Gesamtrahmen der Anwendung. Das MOP erlaubt dabei auch die physische Verteilung der Entwicklung über die Verteilung der Entwickler-Clients sowie die Anlage unterschiedlicher Sichten und damit auch eine virtuelle Dekomposition und Verteilung des entwickelten Systems – im Sinne lokaler Kopien einzelner zu bearbeitender Module bei der klassischen Programmierung. Die Ergebnisse münden unmittelbar in das singuläre Artefakt – sind nur in diesem Kontext lauffähig bzw. wirksam. Die technische Konsistenz aller Modifikationen wird vom MOP sichergestellt.

Integrierte Technik und Organisationsentwicklung: Die angestrebte Konvergenz zwischen technikfernen Disziplinen und den Vertretern der Systementwicklung lässt sich nicht durch ein einseitiges uneingeschränktes Entgegenkommen durch die Seite der Technologie erreichen. Der Einsatz neuer Techniken erfordert immer auch Umgestaltungsprozesse im gesamten Arbeitskontext der Zieldomäne. Angestammte Strukturen, Begriffswelten, Abstraktionen und vor allem Tätigkeitsmuster der Akteure innerhalb der von der Technikunterstützung adressierten Prozesse werden sich naturgemäß mit der Einführung der Technologie ändern müssen. Der Prozess des interaktiven Prototyping verzahnt a priori die Weiterentwicklung der Technologie mit dem Bestreben, die unterstützten Prozesse neu zu gestalten. Im explorativen Umgang mit der Technologie ergeben sich dabei Optimierungspotentiale sowohl für den fachbezogenen Entstehungskontext als auch für den bezogen auf die Abläufe ähnlichen Anwendungskontext. Der kleinschrittige zyklische Prozess der Überarbeitung der eingesetzten Mittel und die damit verbundene Anpassung der umgebenden Prozesse sowie die jeweils direkte Rückkopplung zu den im Einsatz gemachten Erfahrungen, erlaubt eine behutsame Weiterentwicklung des Arbeitskontexts von innen heraus. Entscheidend dabei ist, dass alle Veränderungen durch die Betroffenen selbst intrinsisch motiviert vorangetrieben werden. Die damit einhergehende, jeden Zyklus begleitende Organisationsentwicklung muss in der Regel moderiert, in einigen Fällen sicher auch gezielt gesteuert werden. So können neben den Neudefinitionen von Organisationsstrukturen (Zuständigkeiten) auch punktuell etwa Maßnahmen der Qualifikation von Beteiligten notwendig werden, um den Prozess tatsächlich in Gang zu setzen.

Entwicklungsstandards: Die Schnittstellen des beschriebenen Meta-Objekt-Protokolls des Frameworks machen klare Vorgaben für die Entwicklung. Die sich ergebenden eigentlichen Artefakte, die Meta-Level-Objekte und deren Serialisierung (etwa in XML) werden durch das MOP erzeugt und sind damit per se projektweit konsistent. Visualität und Interaktivität der Entwicklungsschnittstellen eröffnen nicht die gleichen, etwa stilistischen Freiheitsgrade, wie das Formulieren in gängigen Programmiersprachen. Die am Kern der Anwendungen selbst möglichen Anpassungen über White-Box-Schnittstellen sind weitgehend geregelt durch die Vorgaben der Schnittstellen selbst und die entsprechenden Aufrufregime des globalen Kontrollflusses innerhalb des Frameworks. Das Framework impliziert also auch dabei ein Programmiermodell, dessen Einhaltung allerdings eine gewisse Disziplin und die Kenntnis des Frameworks erfordert. Die Wiederverwendung von in vielen Anwendungen kontinuierlich weiterentwickelten und getesteten Komponenten erleichtert die Etablierung der für die Fokussierung auf die Fachlichkeit nötigen Routine im Umgang mit den Entwicklerschnittstellen und -werkzeugen. Auch der für den diskursiven, kollaborativen Entwicklungsprozess fundamentale, einfache Austausch von Entwicklungsergebnissen wird dadurch vereinfacht.

Joint Authoring: Die im Grundlagenkapitel besprochene Methodik des Joint Application Development (JAD) (nach [WS95]) liefert die Idee für die methodische Komponente des Vorgehens, die dazu dient, den sich weitgehend selbst organisierenden Prozess im Rahmen der globalen Zielvorgaben zu steuern. Diese Praktik strukturiert das Zusammenfinden und -arbeiten der Entwicklungsbeteiligten im Sinne einer gesamtheitlichen, aktiven Einbeziehung aller Disziplinen. Die von JAD formulierten „Gebote“ – etwa der

Gleichwertigkeit und Gleichberechtigung aller Beteiligten oder der fachlichen Fokussierung der Projektsprache – werden genutzt, um die Kultur der hier adressierten Projekte zu prägen. In strukturierten Meetings werden dabei Aufgaben bewältigt, in Analogie zu denen, die sich JAD stellt:

- Es muss Verbindlichkeit und Verantwortungsbewusstsein bezüglich des gemeinsam erarbeiteten Zielsystems, manifestiert und gebündelt durch den kollaborativ entwickelten evolutionären Prototypen, geschaffen werden. Eine Grundlage für das Erkennen und Wahrnehmen der gemeinsamen Verantwortung ist unter anderem auch das Aufheben von Besitzständen bezogen auf Ergebnisse. Das ist zu verstehen im Sinne von XP, was fordert, dass alle Teile des Quelltexts allen Beteiligten für Änderungen und konstruktive Kritik zugänglich sein müssen.
- Die Formierung des sozialen Gebildes des Projektteams und die Identifikation mit gemeinsamen Zielen bedarf kontinuierlicher unterschwelliger „Pflege“. Der Sinn des von den Einbezogenen ggf. als unkonventionell betrachteten Vorgehens muss transparent gemacht werden. Die Beteiligten und ihre Aufgabenbereiche werden ausgewählt und festgelegt. Die in Abhängigkeit vom Projektverlauf wahrscheinliche Veränderung von Entwicklungszielen, auch in Bezug auf die Kompetenzen und damit Aufgaben einzelner Beteiligter, wird entsprechend dynamisch in der zyklischen Anpassung der Planungen berücksichtigt.
- Eine konsequente „Mikroplanung“ der gemeinsamen Arbeit und insbesondere der konzeptionellen Teile sowie der Diskussion soll die Zielorientierung aller Aktivitäten sicherstellen. Ausufernde Diskussionen und im Speziellen verhärtetet Fronten zwischen Interessengruppen müssen durch die gezielte Zusammenstellung der Diskussionsrunden aber auch der Prototyping-Teams und die Vorgabe von Themen und Zielstellungen vermieden werden.

Das Joint Authoring bildet den strukturgebenden Überbau – den „roten Faden“ – der Kollaboration. Es hilft dabei, Ergebnisse zu sichern und den diskursiven Gesamtprozess in Richtung der Zielstellung zu fokussieren. Es fügt einzelne Verfahren, wie die qualitative Erforschung des Fachkontexts und die Etablierung einer projektspezifischen Metapher zusammen, indem es dazu anhält Festlegungen für den Ablauf der jeweiligen Aktivitäten im einzelnen zu treffen, z.B. sich auf eine Agenda für gemeinsame Sitzungen zu einigen. Das Verfahren nimmt entsprechend kontinuierlich und über den Gesamtablauf verteilt Einfluss auf den Entwicklungsprozess.

Der bewusste und auch domänenorientierte Einsatz von Mitteln der Präsentation und der Moderation sowie die geschickte Inszenierung des Umfeldes der gemeinsamen Arbeit können in Analogie zum JAD helfen, die subtileren Reibungspunkte bei der Konvergenz der Disziplinen auszuräumen.

Ein Modell für den Entwicklungsverlauf

Nachdem nun die prinzipielle Wirkungsweise des Vorgehens und die dafür grundlegenden Verfahren vorgestellt wurden, beschreiben die folgenden Abschnitte den idealtypischen Ablauf eines Projektes entsprechend der intendierten Methodik. Dabei werden die erörterten Praktiken konkretisiert und in einen modellhaften Ablauf eingebettet.

Das grundsätzliche Muster des Vorgehens bei der Anwendung des Frameworks wurde bereits in Abschnitt 3.1.1 beschrieben und insbesondere in der Abbildung 3.3 auf Seite 129 verdeutlicht. Wesentlich dabei ist das zyklische Wechselspiel zwischen der Ausprägung des Frameworks und des Methodenrahmens für den konkreten Entwicklungskontext einerseits und der Anwendung dessen im eigentlichen produktiven Prozess der Systemerstellung andererseits. Die Erfahrungen der Anwendung fließen jeweils als Feedback zurück in die Überarbeitung des Entwicklungsrahmens. Die Adaption wirkt aber auch auf den Einsatz ein – Erfahrungshorizonte und Zielvorstellungen der Beteiligten und entsprechende Arbeitsabläufe ändern sich. Sie gleichen sich ihrerseits in einem Prozess der Co-Adaption an die neuen Möglichkeiten an. Eine Maßgabe dabei ist jedoch, dass die Adaption des Frameworks und der Methodik und die entsprechende technologische Prägung der Aktivitäten, bezüglich der Ressourcennutzung und auch der konzeptionellen Präsenz (in der Wahrnehmung der Beteiligten) im Projekt eine untergeordnete Rolle spielt. Die fachlichen Aktivitäten der Systemausgestaltung, die von den Domänenexperten getrieben werden, sollen das Projektgeschehen bestimmen können.

Dieser Zyklus wird auf zwei unterschiedlichen Niveaus gestaltet. Um die angestrebte Selbstorganisation des Prozesses überhaupt in Gang zu setzen, Hemmschwellen zu überwinden und Kristallisationspunkte für die explorative Arbeit zu setzen, wird die Konvergenz der Disziplinen bewusst betrieben. Mit Methoden der empirischen Feldforschung und Kreativtechniken werden Potentiale des Technologieeinsatzes ergründet. Die eigentliche Arbeit des Prototyping der Fachexperten gleicht auf dieser Ebene noch „Trockenübungen“. Ist schließlich die Basis für das kooperative Prototyping geschaffen, sind also alle Entwicklungsinterfaces des Frameworks initial ausgeprägt, wird der iterative Zyklus auf die produktive Ebene der Systemausgestaltung gehoben. Dabei wird dann die Prototyping-Umgebung an die sich systematisch weiterentwickelnden Kompetenzen und Bedürfnisse der fachlichen Entwickler angepasst. Diese arbeiten auf der Grundlage dieser sukzessive mitwachsenden Entwicklungsumgebung am eigentlichen Zielartefakt.

Die Abbildung 3.20 fasst die typische Ablaufstruktur für die Entwicklung entlang der hier vorgeschlagenen Methodik in einem Modell zusammen. Die vier wesentlichen Aktivitätsblöcke, die nicht notwendigerweise zeitlich disjunkten Phasen entsprechen müssen, werden in den folgenden Abschnitten in ihren tragenden Schritten beschrieben.

3.2.2. Phase der Konvergenz der Disziplinen

Um eine konstruktive Zusammenarbeit zwischen den zu beteiligenden Disziplinen in einem sich selbst tragenden tätigkeitsorientierten Prozess organisieren zu können, bedarf

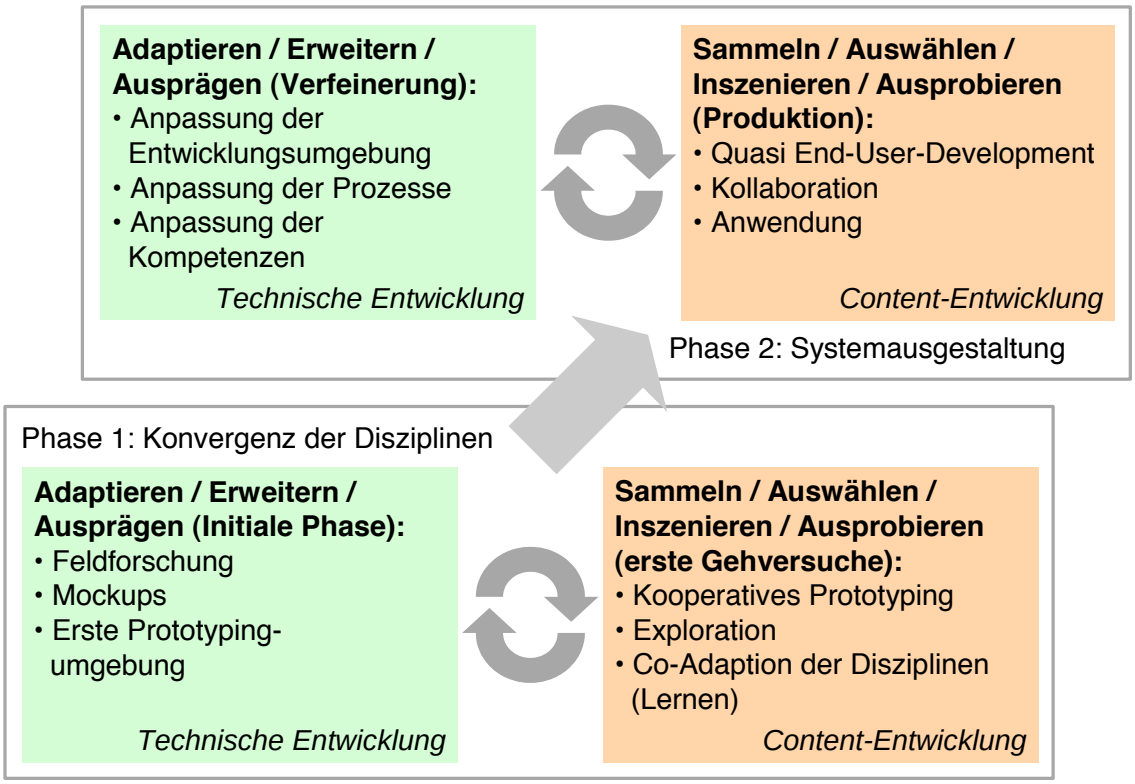


Abbildung 3.20.: Modell für den Ablauf der Iterationen auf unterschiedlichen Niveaus.

es gezielter Interventionen insbesondere beim Start des Projekts. Die besondere Qualität der angestrebten Kollaboration besteht dabei im Unmittelbaren des Austauschs bei gemeinsam zu bearbeitenden Aufgaben. Es gibt dabei weder zeitliche noch physische Puffer in Form von Meilensteinen und Dokumentation. Auf Grund der Tatsache, dass dieses Herangehen im Widerspruch zu einem traditionellen Projektverständnis mit klar trennbare Aufgabenbereichen und entsprechend spezialisierten Akteuren sowie Schnittsstellen steht, muss mit den im Folgenden dargestellten Mitteln gleich von Beginn an die Kultur des Projekts beeinflusst werden.

Es erweist sich als wichtig, den Beteiligten dabei jederzeit auch diese übergeordneten Zielstellungen der möglicherweise als unorthodox bewerteten Aktivitäten transparent zu machen. Nur so ist es möglich, die oft fundamentalen konzeptionellen Schwellen zwischen den technischen und den nichttechnischen Disziplinen zu überwinden, Verständnis- und Arbeitswelten anzugleichen. Die Betonung der produktiven Konzepte der Zieldomäne innerhalb des traditionell durch Technologen getriebenen Software-Entwicklungsprozesses setzt dabei besonderes Einfühlungsvermögen der Systementwickler voraus. Sie sind in der Regel diejenigen die den Prozess gemäß dieser methodischen Vorgaben moderieren, denn zumindest im Grundsatz (quasi hinter den fachlichen „Kulissen“) bleibt ein ingenieurmäßiges Vorgehen erfolgsbestimmend. So werden die in Anlehnung an das *Joint*

Application Development entwickelten Prinzipien des *Joint Authoring* in Verbindung mit der Nutzung des Frameworks alle Aktivitäten im Sinne der Projektziele reproduzierbar kanalisieren.

Adaptieren / Erweitern / Ausprägen (initiale Phase)

Das Vorgehen während der Projektinitialisierung ist geprägt von den Schritten des Joint Authoring – der Ablauf wird gemäß der folgenden Phasen strukturiert und gewissermaßen inszeniert:

Definition: Nicht anders als in den meisten anderen Software-Entwicklungsprojekten, müssen vor Beginn der Arbeiten der Diskursbereich abgesteckt, globale Ziele und Verantwortlichkeiten identifiziert werden. In intensiven Gesprächen mit Entscheidern aller beteiligten Parteien werden die Rahmendingungen, Voraussetzungen und Annahmen sowie der geplante Ressourceneinsatz definiert und ggf. vertraglich fixiert. Das Auswählen und Erkennen der zu Beteiligenden, und der Stakeholder spielt dabei insofern eine sensible Rolle, als dass versucht werden muss, die für die kollaborative Arbeit geeigneten zu finden. In vielen Fachbereichen etwa mit wissenschaftlicher Prägung aber auch unter hochgradig spezialisierten Technologieentwicklern ist es durchaus nicht selbstverständlich, dass die vornehmlich sozialen Kompetenzen, die für die Zusammenarbeit in unmittelbarer Kooperation benötigt werden, ausreichend ausgeprägt sind. Es sollte also ggf. unter zu Hilfenahme von Mitteln der Personalentwicklung versucht werden, designierte Akteure für den Prozess des interaktiven Prototyping so zu wählen bzw. zu schulen, dass es möglich wird, ein Team zu formieren, welches zumindest offen ist für Veränderung und Experimente sowohl bezüglich der Arbeitsweise als auch der Gegenstände der Arbeit.

Noch als Teil der Phase der Definition des Projektes sind alle Beteiligten zum geplanten Vorgehen zu schulen. Es wird von Beginn an Transparenz geschaffen im Bezug auf die Ziele, insbesondere auch auf die übergeordneten. Das Team ist quasi „einzuschwören“ auf die Prinzipien der Kollaboration und den angestrebten Effekt der Emergenz von Lösungen aus dem Diskurs heraus. Dazu werden die zu Grunde liegenden Werte bzw. Prinzipien vermittelt. Das Vorgehen und die Wirkungsweise des Frameworks werden so weit dies zu dem Zeitpunkt möglich ist bereits mit Bezug zum individuellen Projektkontext skizziert. Das heißt, nahe liegende Möglichkeiten der Ausprägung von Werkzeugen, Makroabläufen und Zuordnungen entsprechender Akteure werden zumindest konzeptionell – vorrangig zu Illustrationszwecken – erläutert.

Ethnographische Erforschung: Die eigentliche Konkretisierung des Fachkontexts als Grundlage der Ausprägung der Entwicklungsumgebung bzw. des Produktionsprozesses erfolgt in einer eigenen Phase der Erforschung des Zielkontexts – wiederum im Sinne der Ideen des JAD (Research–Phase). In der Weiterentwicklung der dort vorgeschlagenen intensiven Interviews mit den Domänenexperten und den entsprechenden Verantwortlichen, soll hier quasi ein Prozess der „Selbsterkenntnis“ mit Methoden der ethnografischen Feldforschung betrieben werden. Erst die Anwendung von qualitativ,

interpretierenden Methoden bei der Erforschung der Anwendungsdomäne in Anlehnung an die Ideen von [BGMSW93] gewähren den Disziplinen einen wechselseitigen Einblick, der es erlaubt, nuanciert genug aufeinander eingehen zu können.

Aus einer teilnehmenden Untersuchung heraus sollen zunächst primär die für die Ausprägung des Frameworks zuständigen Technologienentwickler einerseits die Produktionsprozesse der Zieldomäne erkunden. Andererseits sollen sie damit auch die für den Nutzungskontext des Zielsystems entscheidenden Aspekte verinnerlichen – konzeptionelle Strukturen (Begriffe und Metaphern) verstehen, Tätigkeitsmuster aber auch organisatorische oder gar politische Zusammenhänge erfassen. Die Forschenden werden dazu in die adressierten realen Prozesse der Zieldomäne eingebunden. Idealerweise nehmen sie dabei die Rollen aller potentiell betroffenen Akteure ein, um ein möglichst vollständiges Bild der Strukturen innerhalb einer alltäglichen Arbeitssituation zu erhalten. So wird es möglich auch habitualisierte, durch die Domänenexperten nicht mehr bewusst gemachte Anteile der Tätigkeit sowie der Kommunikation zu erfassen. Darüber hinaus erlaubt diese Untersuchungsmethode die Entwicklung eines gesamtheitlichen Verständnisses. Durch die Betrachtung einzelner Entitäten, Beziehungen und Abläufe sowie insbesondere von Handlungen bzw. Äußerungen von Protagonisten eingebettet in den Gesamtkontext, werden auch Dinge, die isoliert betrachtet unsinnig erscheinen, als sinnvoll oder gar wichtig erkannt.

Der Prozess des Einarbeitens und der Unterweisung des Untersuchenden, schafft die Interaktion zwischen den Beteiligten, die einerseits die soziale Basis sein wird für den gesamten kollaborativen Ablauf der Entwicklung. Andererseits wird in dieser ersten Zusammenarbeit der angestrebte weitgehend informelle, wechselseitige Lernprozess angestoßen, der die notwendige Co-Adaption zwischen den Disziplinen, ihren Arbeits- und Sichtweisen schafft. Es zeigt sich, dass trotz der zunächst asymmetrisch angelegten Situation der Untersuchung der Zieldomäne die Interaktion aus der direkten Einbindung in den Arbeitsalltag auch die benötigte Rückwirkung in den Domänenkontext erzielt. So gelingt es die Ideen und Sichtweisen der Technologieentwicklung zu lancieren, für Potentiale und Risiken zu sensibilisieren.

Vorbereitung der Mock-Up-Sessions: Das Ergebnis der Untersuchung sollte dennoch zunächst lediglich eine wechselseitige deskriptive Charakterisierung der Disziplinen sein – mit der kalkulierten Asymmetrie einer Betonung der Beschreibung der Fachdomäne. Es sollte zu diesem Zeitpunkt noch vermieden werden, Konzepte der Umgestaltung (präskriptiv) vorwegzunehmen. Zum einen gebietet es die Sensibilität gegenüber der jeweils anderen Partei. Damit sollen von vornherein das Risiko des „Überfahrens“ und die damit verbundenen atmosphärischen Verwerfungen, die ein konstruktives Zusammenarbeiten verhindern würden vermieden werden. Zum anderen ist ein zunächst unverfälschtes Bild entscheidend für einen systematischen zielorientierten Projektablauf. Spontane und damit in der Regel auch zunächst nicht vollständig reflektierte Lösungsansätze für punktuelle Probleme neigen dazu die Sicht auf den Gesamtzusammenhang zu verzerren und das Finden einer ganzheitlichen Lösung zu erschweren. Es ist also zu diesem Zeitpunkt darauf zu achten, dass die Neigung zu einer Denkweise und Äußerungen wie zum Beispiel „Man könnte mit dem Einsatz einer zentralen Datenbank die

Suche nach Medien viel effizienter gestalten“ zunächst unterdrückt werden.

Diese Ideen sind vielmehr zu kanalisieren in die Systematik der Mock-Up-Entwicklung. Diese sollte als eigenständige Phase im Anschluss an den Abschnitt der Erforschung und deskriptiven Charakterisierung bewusst gemacht und gestaltet werden. Auf der Grundlage eines möglichst vollständigen Bildes der Zieldomäne werden Vorschläge für den Einsatz von Medien und Informationstechnologie und deren Einbettung in Prozesse der Produktion aber auch der Präsentation und Anwendung im Zielkontext abgeleitet. Der Einsatz der Mittel bei der skizzenhaften Präsentation der Ideen ist abhängig von der Projektsituation zu gestalten. In vielen Fällen wird es ausreichen, einen ersten Eindruck möglicher Nutzerschnittstellen mit Hilfe einfacher Grafiken oder gar von Handzeichnungen auf Papier zu vermitteln. Ideal sind Präsentationsmittel, die bereits ein Minimum an Interaktivität zulassen um im Ausblick die Handhabbarkeit wirklich „haptisch“ erfahrbar zu machen.

Diese Mock-Ups bilden die Vorstufe des eigentlichen Prototypen, der auf der Basis des Frameworks abgeleitet werden soll. Der entscheidende Unterschied besteht darin, dass sie wesentlich flexibler sind. Auch tief greifende konzeptionelle Änderungen müssen ad hoc und ohne Aufwand möglich sein. So ist es möglich mit einem maximalen Maß an Sensibilität auf die Bedürfnisse der Domänenexperten einzugehen. Hemmschwellen hin zum selbstständigen Umgang mit der Technologie können durch den vermittelnden Einsatz vertrauter Präsentationsmittel (z.B. zunächst Papier und Bleistift) behutsam überbrückt werden. Genau wie später der Prototyp werden Mock-Ups nicht bloß präsentiert und diskutiert, sondern auch versuchsweise in simulierte Produktions- bzw. Anwendungsprozesse eingebettet. Dementsprechend fließen kann sich der Übergang zwischen der Phase der Erforschung und der Mock-Up-Entwicklung bezüglich der Arbeitsweise gestalten. Die den Entwicklungsprozess moderierenden Systementwickler arbeiten weiterhin direkt aktiv eingebettet in den Arbeitskontext der Zieldomäne. Sie lancieren dabei prototypische Anpassungen sukzessive. Die Produktionsprozesse, die nun bezüglich einiger Aktivitäten versuchsweise Mock-Ups einsetzen, können nun jedoch sicherlich keine wirklich operativen, im Rahmen laufender Projekte sein. Der simulierende experimentelle Charakter zielt nun auf die Ausgestaltung des Produktionsprozesses der eigentlichen Endanwendung.

Die Analogie zum Vorgehen des JAD besteht darin, dass im Anschluss an die Phase der Erforschung, hier ausgeprägt durch die ethnographische Feldforschung, auch die eigentliche Session der partizipativen Systemkonzeption strukturiert vorbereitet wird. Während dabei das wichtigste Instrument des JAD das *Working Document* darstellt, bilden im hier vorgestellten Ansatz die Oberflächenprototypen und die simulierte Einbindung in die Anwendungsprozesse die Grundlage für die Diskussion erster Konzepte der Systemausgestaltung und auch die Umgestaltung der Abläufe der Nutzung. Das entspricht ferner dem Prinzip der Fixierung einer fachlichen Projektmetapher (im Sinne von XP) im gemeinsamen Diskursgegenstand.

Die präzise Planung und Inszenierung der JAD-Session findet ihre Entsprechung in der Vorbereitung der auf der Grundlage der Mock-Ups simulierten Arbeitsabläufe. Neben der vom JAD geforderten Schaffung einer Atmosphäre, die die Konzentration und die

Kreativität während der Session fördert, wird es hier entscheidend sein, möglichst reale Bedingungen zu schaffen. Die müssen es erlauben einerseits in den diskursiven Prozess der Systemausgestaltung einzutreten und dies andererseits mit den Anforderungen der realen Abläufen und Strukturen des Arbeitsumfeldes der Zieldomäne zu verbinden. Das Umfeld sollte dabei dem entsprechen, in dem schließlich die zunächst simulierten Arbeitssituationen und -abläufe dann tatsächlich auftreten. Im Idealfall bleiben, wie bereits angedeutet, die Rahmendingungen zunächst die tatsächlich realen – der Konstellation während der Erforschung entsprechend. Die während der Erforschung bearbeiteten Projekte werden abgelöst von einem idealtypischen Referenzprojekt, in dessen Rahmen dann mit den Mock-Ups und darauf aufbauend angepassten Arbeitsschritten, Verantwortlichkeiten etc. experimentiert wird. Die Planung und Inszenierung dessen muss ein Teil der eigenen bewusst durchgeführten Phase im Vorfeld der eigentlichen Durchführung der Mock-Up-Sessions sein. Insbesondere die Auswahl und die Unterweisung der zu Beteiligenden und die organisatorische Einbindung in bestehende Produktionsprozesse und Unternehmensstrukturen muss sorgfältig geplant und durchgeführt sowie mit den Verantwortlichen abgestimmt werden. Die Erfahrung der Fallstudie zeigt, dass die nötigen Eingriffe in die in sehr langer erfolgreicher Tradition gewachsenen Arbeitsabläufe etwa des Verlagswesens im geisteswissenschaftlichen Bereich in dieser Phase auch ein hohes Maß an politischem Fingerspitzengefühl voraussetzen.

Durchführung der eigentlichen Mock-Up-Sessions: den Ausgangspunkt bildet der angestammte Prozess der Zieldomäne und die Arbeit entlang der gewohnten Tätigkeiten. Die während der Phase der Erforschung als Kandidaten für eine IT-Unterstützung identifizierten Tätigkeiten oder die für eine sinnvolle mediale Repräsentation in Frage kommenden Entitäten und Strukturen werden versucht unter Nutzung der entwickelten Mock-Ups verändert in den Gesamtablauf einzubinden. Die Schnittstellen zum umgebenden Prozess werden entsprechend angepasst.

Um beispielsweise die traditionelle Autorenarbeit des Sammelns und Strukturierens medialer Inhalte auf der Basis von Schnellheftern, Archivanfragen, „Kopiererkollagen“ und handschriftlicher Texterfassung durch ein Software-System unterstützen zu lassen ist es sinnvoll die Möglichkeiten der Mensch-Maschinen-Interaktion mit Hilfe einfacher Screenskizzen zu erörtern – bezüglich des Mitteleinsatzes also zunächst noch sehr nahe am Ursprungsmedium der Domäne zu bleiben. Dabei kann eine freie Arbeitsfläche – etwa eine Schreibtischplatte – noch die Rolle der später virtualisierten Arbeitsfläche des Hauptschirms innerhalb Anwendung übernehmen. Notizzettel in verschieden Größen, Farben und Formen werden genutzt um Bedienelemente und Inhalte zu symbolisieren und zu simulieren. So können typische Handgriffe, wie das Anfassen und Bewegen (drag & drop) von Inhalten eingebettet in ein Layout auf ihre Effektivität untersucht werden, wobei zukünftige Nutzer in der direkten Hands-On-Erfahrung [EK91] antizipieren können, ob eine solche Arbeitsumgebung für sie einen Mehrwert verspricht und akzeptabel sein kann. Insbesondere vermeintlich gängige Bedienparadigmen und Oberflächen-Metaphern, wie Fenster oder der Einsatz von Symbolen (Icons) als Platzhalter für Entitäten der Anwendungsdomäne können so auf Verständlichkeit und Akzeptanz geprüft werden. Aber auch die Gesamtdramaturgie globaler Screenfolgen kann durch einfache Skizzen von Storyboards im Überblick erfasst und plastisch gemacht werden.

So kann verdeutlicht werden, wie sich Schritte der Navigation auswirken und wie effektiv oder umgekehrt umständlich Informationen zu erreichen sind.

Entscheidend ist jedoch, dass die Erkenntnis von Problemen, ohne Aufwand und interaktiv, aus der Diskussion heraus in einen verbesserten Prototypen umgesetzt werden kann. Der Aufwand und damit die Kosten für die in dieser Phase möglicherweise noch sehr tief greifenden Änderungen sind sehr gering (vgl. [EK91] Seite 173). An gleicher Stelle wird auf einen weiteren Vorzug dieses Vorgehens hingewiesen: es sollte Spaß machen, mit dieser speziellen Form einer Kreativtechnik zu arbeiten. Auch wenn dies zu großen Teilen von der Moderation des Prozesses und der Bereitschaft der Beteiligten Phantasie und Kreativität aufzubringen abhängen wird, ist dies ein wichtiger Beitrag zur Motivation für echte partizipative Arbeit. Die Schwierigkeit bei der Moderation besteht lediglich darin, die Gradwanderung unter Kontrolle zu halten – zwischen der Wahrung eines maximalen Gestaltungsspielraums für eine möglichst individuelle Anpassung der Gestaltungsgegenstände an die Bedürfnisse der Domäne einerseits und den Möglichkeiten einer Umsetzung in tatsächliche technische Konstrukte andererseits.

Die während der Mock-Up-Session eingesetzten Mittel können jedoch durchaus auch tatsächliche Interaktive Prototypen sein. Im Rahmen des Projekts, welches zur Evaluation der Konzepte dieser Arbeit herangezogen wurde, bildet eine bestehende Entwicklungsumgebung den Ausgangspunkt für die diskursive Arbeit an einer Schnittstelle für Autoren. Entlang der, während der Versuchsweise durchgeführten Arbeiten aufgedeckten Defizite wurde schrittweise eine individualisierte Entwicklerschnittstelle für die beteiligten Autoren und Redakteure abgeleitet. Die Abbildung 3.21 zeigt das im Bereich des E-Learning bekannte Werkzeug LRN-Editor der Firma Microsoft. Ein Werkzeug, was hauptsächlich der Strukturierung und Sammlung von Inhalten dient. Das entspricht den Aufgaben, auf die sich Autoren konzentrieren sollen – zunächst ohne sich um Aspekte der Darstellung kümmern zu müssen. Es lässt jedoch intuitiv erahnen, dass Abstraktionen zur Gliederung und Adressierung von Medien mit der Arbeitsweise von beispielsweise Historikern kaum vereinbar sind. Der Einsatz des Werkzeugs als Mock-Up stellte sich dennoch als sinnvoller Ansatzpunkt zur Verdeutlichung technischer Möglichkeiten heraus und fungierte als Kristallisationspunkt für die Ableitung fachgerechter Schnittstellen. Die Illustration der möglichen Anpassungen erfolgte dabei hauptsächlich über rein grafische Vorschläge.

Die Screenfolgen der Zielanwendung wurden ebenfalls zunächst anhand einfachster Skizzen diskutiert. Ein zumindest einem der Redakteure vertrautes Werkzeug zur Unterstützung dessen bildete Microsoft Powerpoint. Grobe Gliederungen der Oberflächen und auch Abläufe von Bildschirminhalten können damit sehr einfach simuliert werden. Es kann aber als glücklicher Umstand, innerhalb der individuellen Projektsituation gewertet werden, dass einer der Fachredakteure sehr routiniert mit diesem Werkzeug umgehen und damit in interaktiven Sitzungen aber auch selbständig Mock-Ups aktiv mitgestalten konnte. Die parallel betriebene Entwicklung der Screendesigns konnte mit den Aktivitäten dieser Ablauf- bzw. Layout-Gestaltung sehr einfach verzahnt werden.

Im Gegensatz zur weitgehend monolithischen Organisation der JAD-Session bildet die Mock-Up-Session hier lediglich den Auftakt zum iterativen Prozess des interaktiven

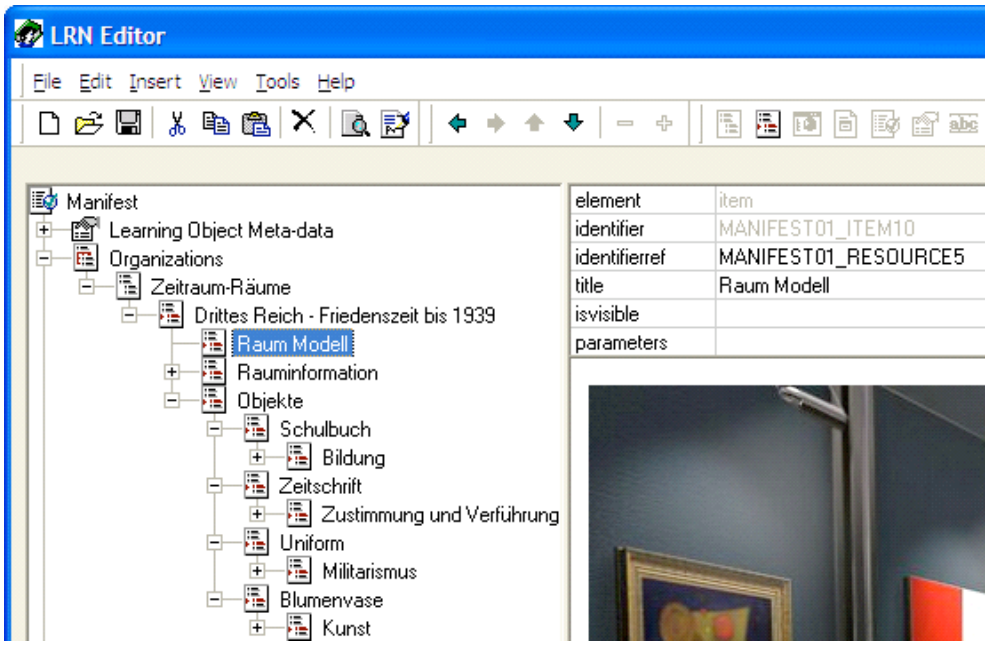


Abbildung 3.21.: Der Microsoft LRN-Editor als interaktive „Vorstudie“ mit versuchsweise von Fachexperten eingepflegten Inhalten

Prototyping. Die Analogie ist jedoch insofern weiterhin aufrechtzuerhalten, als dass die Strukturiertheit der Organisation und der Aktivitäten der Ergebnissicherung übertragen werden sollen. So ist etwa das Ergebnis der Mock-Up-Session ein weitgehend ausgereifter und idealerweise dokumentiertes Mock-Up-System, das als Vorlage für die Ausarbeitung der eigentlichen Prototyping-Umgebung trägfähig ist. Vergleichbar ist dieses Ergebnis somit mit dem Designdokument, das die Ausgabe der zusammenfassenden Dokumentationsphase im Anschluss an die JAD-Session bildet. Mit entsprechender Sorgfalt sollte der Abschluss der Mock-Up-Phase organisiert werden. Die Ergebnisse sollten einen von allen Beteiligten akzeptierten und verstandenen initialen Konsens widerspiegeln. Die Art und der Aufbau der Dokumentation sind auch dabei an das Verständnis der Akteure anzupassen. Im Falle des Evaluationsprojektes ist dazu eine so genannte Werkbeschreibung entstanden, die bezüglich ihrer Struktur und ihrer Aussage an ein in diesem konkreten Unternehmenskontext (Schulbuchbranche) übliches Dokument angelehnt ist. Flankiert wird dies in der Regel sehr pragmatisch durch Protokolle und die letzten Arbeitsstände der Mock-Ups in diesem Falle der kommentierten Powerpoint-Dateien.

Die Ableitung der initialen Prototyping-Umgebung auf der Grundlage es Frameworks stellt also die strukturierte Zusammenfassung der Session-Ergebnisse dar. Die in den Mock-Ups zunächst Skizzierten Funktionalitäten werden, mit den Mitteln des Frameworks als erster tatsächlich ablauffähiger Prototyping materialisiert. Erkenntnisse über Anforderungen zur Einbindung in einen operativen Zielanwendungsprozess und

entsprechende Verteilungskonzepte werden herangezogen, um die Architekturvariante zu wählen – gemäß der auf den Seiten 157 ff. beschriebenen Möglichkeiten.

Die Ausprägung der einzelnen Sichten, also der Schnittstellen für die spezifischen Nutzerzugänge aber auch die Abbildungen in die endgültige Präsentation der Zielanwendung (inklusive der WYSIWYG-Komponenten), erfolgt mit den Mitteln des Frameworks. Dies wird zumindest bei der Erstellung des ersten Aufschlags einer Prototyping-Umgebung fast ausschließlich technische Expertise, ganz konkret die eingehende Kenntnis des Frameworks erfordern. Abhängig vom Grad der Abweichung der Anforderungen von bereits angelegten Komponenten müssen zunächst auch Anpassungen an White-Box-Schnittstellen, also „tief “ am Framework-Kern vorgenommen werden.

Die Aktivitäten umfassen im Einzelnen:

- Die Festlegung von Regeln für den Aufbau der Navigationsstruktur. Die während der Mock-Up-Session eruierten sinnvollen Abläufe und Gliederungsstrukturen fließen dabei zum einen in eine Schablone für das Meta-Level-Object des Systemaufbaus. Das entspricht einem Schema für dessen XML-Repräsentation auf der Grundlage von im Framework bereits angelegten Mustern. Diese wird schließlich im Wesentlichen materialisiert in Ablaufregeln für die Werkzeuge, die für die Gliederung genutzt werden. Diese müssen schließlich sicherstellen, dass nur diejenigen Verknüpfungen zwischen Navigationsknoten hergestellt werden, die geplant sind – ein „Thema“ also beispielsweise immer unter ein „Schwerpunktthema“ gegliedert wird. Diese Regeln müssen auf der Grundlage der aktuell verfügbaren Implementierung des Frameworks noch fest im Code der Werkzeuge angepasst werden. Eine generische Ableitung der Steuerfunktionalität aus dem XML-Schema ist jedoch angedacht. Außerdem, wird die Navigationsfunktionalität des Anwendungsrahmens gewählt, konfiguriert (in der XML-Repräsenation des zentralen Meta-Level-Objektes der Anwendung) und gegliedert.
- Der initiale Vorrat an disponiblen Interaktionselementen wird ausgewählt und in den Werkzeugleisten der jeweiligen Entwicklersicht verfügbar gemacht. Es werden ferner erste Aggregate aus elementaren Bedienelementen zusammengesetzt. Dies geschieht grundsätzlich über die Erstellung von Schablonen für XML-Repräsentationen von Meta-Level-Beschreibungen. So entstehen etwa neue „Klassen“ von Fenstern oder auch neue Ausprägungen von Player-Komponenten. Es werden jedoch auch Neuentwicklungen und Verfeinerungen am Framework-Code vorgenommen. Die Grundlage für die Ausprägung des Interaktionsrahmens bilden die im Framework angelegten Interationselemente (siehe 3.9 auf Seite 149).
- Es erfolgt die Entwicklung und Anpassung der Entwicklerwerkzeuge entsprechend der zu unterstützenden Abläufe und mentalen Modelle. Den fachlichen Entwicklungsschritten werden dabei die implizierten Transformationsschritte der Disposition von Metalevel-Objekten, also auch der neuen Aggregate zugeordnet. Damit ist der fachlichen Gliederung eines „Themas“ unter ein „Schwerpunktthema“ die implizite Implementierung der Verknüpfung (Link) entsprechender Meta-Level-Objekte und wiederum deren Einordnung in das Gesamtgerüst zugeordnet.

- Schließlich werden erste Ideen zur grafischen Gestaltung umgesetzt. Bedienelemente, sowohl die im Rahmen der Entwicklung disponiblen, als auch diejenigen der Entwicklungswerkzeuge, werden mit Grafiken bzw. Icons zur Widerspiegelung der innerhalb der Anwendungsdomäne vertrauten Bildsprache versehen. Stildefinitionen (XML-repräsentierte Erweiterungen der Meta-Objekt-Beschreibungen) werden erstellt, um erste Entwürfe des Layouts und der Farbeschemata von Navigationsknoten bzw. Aggregaten von Bedienelementen sowie der Gesamtanwendung sichtbar zu machen.

Das Ergebnis sollte ein erster Prototyp sein, der die Keimzelle des evolutionären Wachstums der Zielanwendung bildet. Im Sinne von XP ist es dabei nicht entscheidend alle eventuell aufkommenden Anforderungen zu antizipieren und quasi prophylaktisch umzusetzen. Viel wichtiger ist es schnell ein in den wesentlichen, während der Mock-Up-Sessions diskutierten Eigenschaften funktionales System zu erstellen, um die Kontinuität des Dialogs zwischen den Disziplinen möglichst nicht zu unterbrechen. In vielen Fällen, wird es dann zur Herausforderung der Moderation des Prozesses zu vermitteln, dass die schrittweise Vervollkommnung des Systems eine gemeinsame Aufgabe und im gemeinsamen Interesse zu erledigen ist und dass ein monolithischer, isolierter, ausschließlich von den Technologen betriebener Entwicklungsabschnitt auch im Sinne der kollektiven Kontrolle über das entstehende Ergebnis, kein gangbarer Weg ist.

Sammeln / Auswählen / Ausprobieren (erste Gehversuche)

Auf der Grundlage des initial zu einer Entwicklungsumgebung bzw. zu einem ersten Prototypen ausgeprägten Frameworks beginnt nun die eigentliche Arbeit des interaktiven kooperativen Prototypings als eigene fachlich bzw. inhaltlich zentrierte Phase der Entwicklung. Im Gegensatz zu der im folgenden Abschnitt erläuterten tatsächlich zu leistenden produktiven Implementierungsarbeit im Rahmen des Grundsätzlich gleichen Entwicklungsschritts, wird der Charakter der Tätigkeit auf diesem Niveau (siehe *Phase 1* in Abbildung 3.20) noch geprägt sein von intensiven Lernerfahrungen bei den Beteiligten. Auch die Integration in die an den Entwicklungsprozess angrenzenden Abläufe wird noch immer zu großen Teilen lediglich eine provisorische Basis haben. Diese wird erst im Zuge der gezielten Korrektur der integrierten Organisations- und Technikentwicklung auf der Grundlage eines weitgehend ausgereiften produktiven Prototyps konsolidiert werden können.

Um die Parallele zum Vorgehen des Joint Application Development weiterhin zur Orientierung nutzen zu können, ist eine Erweiterung des eigentlich als zeitlich linear zu charakterisierenden Ablaufs nötig. Mit der Erstellung des initialen Prototyps wird eine neue Iteration der gemeinschaftlichen Entwicklung eingeleitet. Die Instanziierung des Frameworks entspricht der Vorbereitung einer zweiten, vertieften JAD-Session auf der Grundlage eines erweiterten wechselseitigen Verständnisses und vor allem auf der technologischen Basis des gemeinsam entwickelten ersten Projektergebnisses. Teil dieser Phase der Vorbereitung ist es nun auch den Entwicklungskontext von dem unter Umständen während der Mock-Up-Phase noch notwendigen Laborcharakter zu befreien und so zu

gestalten, dass er den Anforderungen des produktiven Alltags genügen kann. So muss es nun auch möglich sein, geographisch verteilt arbeitende Teams durch den Prozess unterstützen zu lassen. Die Entwicklungsinfrastruktur mit einem zentralen Repository für das Artefakt und die Zugänge über verteilte Entwicklerschnittstellen müssen in der nahezu endgültigen Form etabliert sein. Die Flexibilität für Anpassungen, die im Zuge des diskursiven Weiterentwicklungsprozesses erkennbar und notwendig werden, sollte erhalten bleiben. Bei verteiltem Arbeiten sollte ergänzend eine Infrastruktur zur Unterstützung des wechselseitigen Coachings, der Prozessmoderation und des Wissensaustauschs etabliert werden. Dabei ist es wichtig, möglichst synchrone und unmittelbare Plattformen auch für den informellen Austausch zu schaffen, um die Interaktivität des Entwicklungsprozesses am gemeinsamen Artefakt zu gewährleisten (siehe dazu auch die Abschnitte zu den Grundlagen 2.3.6 sowie 3.3 zur bewussten Wissensarbeit).

Die eigentliche Prototyping-Session im Verhältnis zur rein konzeptionellen JAD-Session ermöglicht nun eine erste intensive kollaborative Auseinandersetzung mit den Möglichkeiten der Technologie. Dabei müssen von Beginn an auch die Möglichkeiten des Vorgehens transparent gemacht werden. Nur die Einsicht, dass durch die direkte Adaptivität der Technologiegrundlage für aktuelle, individuell im Arbeitsprozess entstehende Bedürfnisse und Bedarfe, keine Risiken und Einschränkungen bezüglich des fachlichen Gestaltungsspielraums entstehen, verleiht dem Projekt den angestrebten diskursiven und explorativen Charakter. Nur so kann der Entwicklungs- bzw. Erstellungsprozess auch als wechselseitiger moderierter Lern- und Anpassungsprozesses fungieren, den alle Beteiligten intrinsisch motiviert tragen.

Das vorsichtige Zusammenfinden der Disziplinen durch gegenseitiges Lernen – betrieben in Anlehnung an die Methoden, der skandinavischen Schule 2.3.3 – mündet nahtlos in die produktive Arbeit der Entwicklung.

Das zentrale interaktive und kooperative Prototyping basiert auf der Idee in [BG91]. Die dort erstmals vorgeschlagene aktive Beteiligung von potentiellen Nutzern am Designprozess an Oberflächenprototypen wird im hier vorgestellten Ansatz weitergeführt auf der Grundlage eines evolutionären Prototypen unter zu Hilfenahme eines dezidiert dafür konzipierten Frameworks. Analog zum Einsatz der von [BG91] vorbereiteten Prototypen, werden die designierten Fachentwickler mit dem initial ausgeprägten also auch lediglich mit exemplarischen Inhalten ausgefüllten System konfrontiert. Ihre Aufgabe ist es nun zum Beispiel, zwar unter Anleitung aber auch möglichst selbstständig Inhalte und Medien einzufügen, zu strukturieren, zu inszenieren – im Zusammenhang mit der jeweils dafür passenden Funktionalität der Interaktivität sowie Funktionalitäten im einzelnen zu arrangieren (zu den Aufgaben im Detail siehe den Abschnitt 3.1.4 zu den Entwicklungsaufgaben ab Seite 161). Auf auftretende Schwierigkeiten in der Handhabung aber auch Fehler in der technischen Umsetzung und vor allem zu ergänzende Funktionen, die neuen Ideen und Erkenntnissen entspringen wird soweit möglich sofort eingegangen.

Anders als beim von [BG91] vorgeschlagenen kooperativen Prototyping steht dabei allerdings nicht der jeder Zeit einzunehmende Blickwinkel des Endnutzers allein im Mittelpunkt, sondern zunächst auch die Perspektive des Vermittelnden bzw. Lehrenden der

Fachinhalte. Angestrebt wird eine Verzahnung der Prozesse der Nutzung und der Erstellung – zumindest bezüglich der Metaphorik, der Interaktivität sowie der Muster des Aufbaus (Gliederung) und des Ablaufs der genutzten bzw. der gestalteten Infrastruktur. Das an den Endnutzer zu vermittelnde fachliche Verständnis, die Kompetenzen und das Wissen, welches in die Software fließt, soll weitgehend dem entsprechen, was zur Gestaltung nötig ist. Durch die fachliche Prägung der Entwicklung gelingt die Aufhebung der Dualität zwischen Entwicklung und Nutzung – entsprechend der Idee der adaptierbaren Entwicklungsmetapher.

Die fachlich orientierten Entwicklungsbeteiligten versuchen so auf der Grundlage einer Entwicklungsumgebung, die den ihnen angestammten Tätigkeiten und Begriffswelten möglichst gut entgegenkommt, das System auszugestalten. Dabei können sie die Auswirkungen eines jeden ihrer Schritte auf das Zielartefakt jeder Zeit überprüfen – im Rahmen der WYSIWYG-Bearbeitung sogar unmittelbar.

Die Kernidee des interaktiven kooperativen Prototyping besteht nun darin, dass neben der sukzessiven kollaborativen Überarbeitung des Zielartefakts auch die dazu verwendeten Mittel kontinuierlich an die sich ändernden Anforderungen des Produktionskontexts – etwa durch den veränderten Erfahrungshorizont – angepasst und verfeinert werden. So erschließen sich die Fachleute in Personalunion mit ihrer Rolle als Nutzervertreter nach und nach die Gestaltungsspielräume, die sich aus dem Technologieeinsatz ergeben. Zunächst werden die nötigen Anpassungen, insbesondere an den aufgabenspezifischen Nutzerschnittstellen noch immer in eignen, möglichst kurzen Phasen der technologischen Entwicklung vorgenommen werden müssen. An dieser Stelle kommt der kooperative Charakter des Ansatzes zum tragen. Nur in engster Zusammenarbeit zwischen den beteiligten Disziplinen kann ein kontinuierlicher Gestaltungskreislauf, orientiert an den Tätigkeiten der Anwendungsdomäne, lediglich unterstützt durch die Technologie in Gang gehalten werden, der in vervollkommneten Produktionsmitteln und dem damit erstellten Produkt konvergiert. Im Diskurs der gemeinsamen Arbeit erkennt man Defizite sowohl an der Technologie, als auch an den Kompetenzen der Beteiligten im Umgang mit der Entwicklungsumgebung sowie den umgebenden Organisationsstrukturen. Diese Defizite werden unmittelbar ausgeglichen und deren Auswirkungen können sofort evaluiert werden. Die diesen Prozess begleitenden technologisch Sachverständigen werden ihrerseits auch das Verständnis von der Domäne und den dort vorherrschenden Sichtweisen und Bedürfnissen systematisch vertiefen und so immer präziser in der Lage sein, mit ihrer Unterstützungsleistung auf die Prototyping-Arbeit der Fachexperten einzugehen. Angestrebt wird jedoch auch, dass insbesondere die „Feinkalibrierung“ der Werkzeuge durch das nichttechnische Entwicklungspersonal selbständig vorgenommen werden kann – etwa die Überarbeitung globaler Styles und damit die veränderte Wirkung von Layoutwerkzeugen. Die Übertragung der tätigkeitstheoretischen Konzepte hinter dieser Rückkopplung auf die Belange der Software-Entwicklung und den bewusst betriebenen sozio-technologischen Prozess der Co-Adaption zwischen den beteiligten Disziplinen ist in Abschnitt 2.3.6 auf den Seiten 110 ff. ausgeführt. Der co-adaptive Prozess zwischen den Disziplinen wirkt vernetzt zwischen den Methoden und Werkzeugen der Disziplinen sowie auch zwischen den Kompetenzen der jeweiligen Vertreter.

Von den Erkenntnissen bezüglich der Spielräume, die der Technologieeinsatz bei der Repräsentation fachlicher Inhalte und Zusammenhänge eröffnet, profitiert der Anwendungskontext direkt. Das was sich während der Erstellung aus fachlicher Sicht als sinnvoll und praktikabel herausstellt und in der Entwicklungsumgebung verankert wird, findet in der Regel auch als Konzept in der Anwendung seinen Niederschlag. Als einfaches Beispiel hierfür kann die inhaltliche Gliederung des Systems betrachtet werden: wenn sich ein Netz von „Themenräumen“ und „Themenschwerpunkten“ als eine in der Entwicklung tragfähige, fachlich akzeptierte Metapher für die Strukturierung der Inhalte erweist und jeweils konsensfähige grafische und interaktive Repräsentationen dafür gefunden und umgesetzt sind, dann fließt die entsprechend ausgeprägt Komponente in die Prototyping-Umgebung, also in den Satz disponibler Elemente ein. Damit ist deren Einsatz in der Regel auch im Rahmen der Präsentation gegenüber dem Nutzer sinnvoll.

3.2.3. Phase der Systemausgestaltung

Am Übergang von der experimentellen Phase der Konstituierung der projekt-, fach- und organisationsspezifischen Entwicklungsumgebung zur eigentlichen produktiven Arbeit der evolutionären, mehr und mehr eher evolvierenden Systemausgestaltung, werden auch die Aktivitäten zur Korrektur und Anpassung des Entwicklungskontexts fokussierter auf produktive Bedürfnisse ausgerichtet. Aspekte des systematischen Konfigurations- und Änderungsmanagements, der Qualitätssicherung oder auch der Nachhaltigkeitssicherung – etwa zur Übertragung der Konzepte in Unternehmensstrategien und die Personalentwicklung – werden nun berücksichtigt.

Die anstehenden Aufgaben im Überblick

Wie in der Abbildung 3.20 auf Seite 197 angedeutet, sind im weiteren Projektverlauf die grundsätzlich gleichen Aufgaben zu erledigen wie in der initialen Phase. Lediglich die Ansprüche an das Vorgehen und die Ergebnisse werden nun vom weitgehend erkundenden Niveau auf ein mehr produktives gehoben (vgl. [KFK05]). Im Folgenden werden – um den Bezug zum bisherigen Vorgehen zu wahren – die Aufgaben entlang der durch das Schema in Abbildung 3.20 vorgegebenen Struktur im Überblick erläutert. Die in der Praxis benötigte prozessuale Stringenz wird dann durch die Einbettung der Aufgaben in den anschließend dargestellten Referenzprozess etabliert.

Die Phasen des **Adaptieren / Erweitern / Ausprägen**, nun mit dem Zusatz **(Iterationen)** versehen, bietet im Wesentlichen nur noch Unterstützungsleistungen für die eigentliche Ausgestaltung des Systems. Dabei werden Coaches, Prozess- und Framework-Sachverständige dafür sorgen, dass die fachlichen Entwickler produktiv am Artefakt arbeiten können. Es liegt in der Verantwortung dieser Moderatoren dafür zu sorgen, dass die wechselseitige Anpassung zwischen der Anwendungsdomäne und der entstehenden Technologie so verläuft, dass der Prozess an den Zielen des Gesamtprojektes ausgerichtet bleibt und in diese Richtung konvergiert. Aus der Sicht der Fachentwickler

sollten die Gestaltungsspielräume entsprechend der sich entwickelnden Sichtweisen und Kompetenzen mit wachsen. Eine wichtige Leistung beider Seiten ist die Etablierung und Aufrechterhaltung des sozialen Umfeldes, innerhalb dessen, eine echte Partizipation und Kollaboration zustande kommt. Das ist nötig, um die fachliche Individualisierung des Entwicklungsgeschehens mit der entsprechenden wechselseitigen Sensibilität schaffen zu können.

Die entscheidenden in diesem Sinne zu bewältigenden Aufgaben sind:

- Die fortgesetzte kontinuierliche Anpassung der Prototyping-Umgebung entsprechend der individuellen Erfordernisse der fachlich orientierten Entwicklung. Die Änderungen, die sehr tief am Framework über White-box-Schnittstellen vorgenommen werden müssen, werden wie erwähnt mit dem Projektverlauf immer geringer im Umfang. Ziel sollte es sein, die nötigen Korrekturen ad hoc, also nahezu interaktiv vornehmen zu können. Im Idealfall, erreicht man einen Punkt, an dem Fachentwickler selbständig oder lediglich unter der Anleitung der Framework-Experten ihre Produktionsumgebung an ihre eigenen aktuellen Bedürfnisse anpassen können.

  Die Erfordernisse werden sich über die Iterationen mit dem Verlauf der Evolution des Zielsystems ändern. Sowohl der Erkenntnisgewinn über die Gestaltungsmöglichkeiten der Technologie auf Seiten der Nichttechniker als auch die damit verbundene Lernkurve im Umgang mit den Werkzeugen, werden diese Entwicklung forcieren. Die Erfahrung zeigt, dass dies mit dem Risiko verbunden ist, nicht konvergierende Entwicklungen an Anforderungen auszulösen. Es ist die Aufgabe des organisatorischen Umfeldes in Verbindung mit den Moderatoren bzw. Coaches, die Grenzen der globalen Zielstellung mit dem Erreichten abzugleichen.

- Eine weitere Aufgabe ist die permanent nachgeführte Anpassung der Produktionsprozesse und der arbeitsorganisatorischen Rahmenbedingungen im Allgemeinen. Letzteres bezieht sich sowohl auf den Entwicklungsprozess, als auch auf den Kontext der Nutzung durch die Endanwender. Die auf Grund der Einführung der Technologie veränderten Rollenverständnisse und Verantwortlichkeiten müssen dabei Berücksichtigung finden. Das kann für den Produktionskontext beispielsweise bedeuten, dass sich das Selbstverständnis von ursprünglich ausschließlich für den Bereich der Printmedien arbeitenden Autoren wandelt – die Wege der Medienrecherche und -bestellung oder des Austauschs untereinander sich grundlegend ändern. War es etwa ursprünglich Usus, nach einer langen Phase der isolierten Arbeit, ein mehrere hundert Seiten umfassendes Papiermanuskript bei der Redaktion eines Verlages vorzulegen, was nach der Diskussion im Rahmen von Redaktionskonferenzen verabschiedet wurde, kann der diskursive Prozess nun ggf. in viel kleineren Zyklen ablaufen. Leichter beherrschbare kleinere Zwischenergebnisse können Schritt für Schritt von der Redaktion oder den Autoren selbst zusammengeführt und jeder Zeit im Kontext des angestrebten Gesamtergebnisses auf seine Qualität geprüft werden. Die engere Zusammenarbeit erfordert ein angepasstes Management des Gesamtprojektes. Was sich damit potentiell auch ändern muss sind die Modelle der Honorierung. Bezogen auf die Anwendung des Ergebnisses können

die Auswirkungen so weit reichen, dass etwa die Art des Unterrichts im adressierten Fach vollkommen revolutioniert wird. Die Organisation der adressierten Unterrichtssituation muss dann dementsprechend neu gestaltet werden. Eine Abstimmung mit der für diesen Unterricht nötige Infrastruktur muss vorgenommen werden.

- Um den Akteuren innerhalb der neu gestalteten Prozesse ein effektives Arbeiten zu ermöglichen, müssen mit der Weiterentwicklung der Rollen und den daraus resultierenden Anforderungen auch die entsprechend benötigten Kompetenzen angepasst werden. Neben dem angestrebten informellen und behutsamen „Hineinwachsen“ der am diskursiven Prozess des interaktiven Prototyping Beteiligten in ihre neuen Rollen, muss der Prozess des Erwerbs des benötigten Wissens und der Handlungskompetenz gezielt gesteuert und betrieben werden. Auftretende Defizite, insbesondere solche, die den Fortgang des sich weitgehend selbst organisierenden Entwicklungsablaufs behindern, müssen ggf. aktiv, etwa durch Schulungsmaßnahmen ausgeräumt werden. Dabei ist es zentral die Aufgabe der Moderation, dafür Sorge zu tragen, dass diese Defizite erkannt werden können. Der Moderator ist jedoch nicht für die Analyse der Situation selbst zuständig, sondern vielmehr gehalten, für kontinuierliche Selbstreflektion der beteiligten zu sorgen.

Der Entwicklungsabschnitt des **Sammeln / Auswählen / Ausprobieren** auf der Grundlage der eben zusammengefassten Anpassungen des Entwicklungskontexts erreicht nun tatsächlich produktives Niveau. Die fachlichen Entwickler sind in der Lage, selbständig ihre fachlich inhaltlichen Konzepte medial umzusetzen und in das interaktive Gesamtsystem zu integrieren. Diese Phase entspricht nun im Grunde tatsächlichem End-User-Development. Dabei wird zunächst aus dem Blickwinkel der Anwendungsdomäne die domänenspezifische Entwicklungssoftware angewendet und schrittweise an individuelle Bedürfnisse angepasst. Im Zuge dessen entsteht dann das eigentliche Zielsystem, wobei Abläufe, Terminologie, Metaphorik und Interaktivität der Entwicklungsumgebung und der Zielsoftware ineinander übergehen. Diese Entwicklung ist also geprägt von der permanenten Anwendung zum Zwecke der Evaluation im Wechselspiel mit der Anwendung zum Zwecke der fachlichen Weiterntwicklung.

Das entstehende System ist damit charakterisiert durch seine flexible Anpassbarkeit über Interaktionsformen, die den Bedürfnissen und Tätigkeiten der Anwendungsdomäne entsprechen und mit einer Metaphorik die auch dem End-Nutzer geläufig ist. So wird schließlich auch ein End-User-Development im klassischen Sinne möglich. Abhängig von der Produktstrategie und einer entsprechenden Öffnung des Produkts für Änderungen kann der Nutzer so selbst weitere Individualisierungen vornehmen – die Phasen der Erstellung, der Anwendung und der Wartung fließen direkt ineinander. Wichtige Einsatzszenarien dafür sind beispielsweise die Anpassungen, die etwa Lehrer an einem System für den Einsatz im Unterricht vornehmen möchten. Aber auch die Individualisierung oder gar Weiterentwicklung durch Rezipienten selbst ist in vielen Szenarien wünschenswert.

Der Produktionsprozess systematisch

Um einen belastbaren idealtypischen Prozess zu erhalten nutzt der hier entwickelte Ansatz eine Interpretation bzw. Adaption etablierter Vorgehensmodelle: sowohl STEPS als auch die *Integrierte Technologie- und Organisationsentwicklung (OTE)* wurden in den grundlegenden Abschnitten (2.3.4 auf den Seiten 90 ff.) eingeführt und der Brückenschlag zur hier vorgenommenen Konkretisierung für den speziellen Kontext transdisziplinärer Projekte bzw. unter der Bedingung des Einsatzes framework-basierten interaktiven Prototypings vorgestellt.

Wesentlicher Impuls – auch in Abgrenzung zu etablierten Vorgehensmodellen – ist die bewusste Einbeziehung von Phasen der Nutzung in den Erstellungsprozess. Diese Phasen dienen der gezielten Evaluation der entstehenden Technologie und in der Erweiterung durch OTE auch der Anpassung der umgebenden Arbeitsorganisation an die sich im und durch den Entstehungsprozess ändernden Bedingungen. Der innerhalb von STEPS noch weitgehend diskrete Prozess, der offensichtlich organisatorisch und bezüglich der Zeit noch klar trennt zwischen den Schritten der Konzeption, der Umsetzung, der Nutzung und der Wartung soll im hier propagierten Ansatz durch einen kontinuierlich ablaufenden Zyklus ausgeprägt werden. Dabei entstehen keine einzelnen, im eigentlichen Sinne des Begriffs „Version“ disjunkten Systemvarianten, die im Anschluss an ihre Entwicklung und eine entsprechend Freigabe durch die Software-Entwickler in die ihrerseits „monolithische“ Phase der Nutzung überführt werden. Auch die Änderungsanforderungen, die sich aus der Nutzung ergeben münden nicht in eine fixierte Spezifikation als Grundlage einer weiteren isolierten Phase der Technologienetwicklung.

Nach der Konstituierung des Projekts und der Infrastruktur – entsprechend der Beschreibung im vorigen Abschnitt – wird hier vielmehr angestrebt, die Konzeption, die Umsetzung und die evaluierende bzw. zum Teil auch die produktive Nutzung direkter zu verzahnen. Das wird einerseits dadurch möglich, dass die Partizipation potentieller Endnutzer bzw. der Fachexperten ausgeweitet wird auf Aktivitäten der Implementierung und andererseits durch die permanente Verfügbarkeit eines ablauffähigen integrierten Prototypen. So können die Fachexperten in Kooperation mit den Technologieentwicklern oder im besten Fall lediglich unter deren Anleitung das System ausgestalten und jeder Zeit die Rolle des Nutzers einnehmen. Das erfolgt auf zwei Ebenen: zum einen evaluieren sie aus der Sicht und mit dem Verständnis des Endnutzers der Zielsoftware das Ergebnis ihrer eigenen Entwicklungsleistung in direkter Rückkopplung. Zum anderen fließen die Ergebnisse ihrer Arbeit (ggf. unterstützt durch die Technologieentwicklung) auch in die kontinuierliche Verfeinerung der von ihnen genutzten Entwicklungsschnittstellen selbst. Diese nutzen und evaluieren sie dann im tatsächlichen produktiven Zusammenhang. Diese Zyklen des Rollenwechsels können beliebig (quasi unendlich) kurz sein – unter Nutzung des WYSIWYG-Modus ist ein Übergang zwischen Produktion und Nutzung nahezu aufgehoben. Durch diese Direktheit des Wechselspiels zwischen der Tätigkeit der Entwicklung und der Anwendung, also durch diese Interaktivität der Entwicklung wird dem interaktiven Charakter der Nutzung Rechnung getragen.

Das in [FRS89] präsentierte Modell von STEPS (wiedergegeben in Abbildung 2.14 auf

Seite 91 und in diesem Kontext diskutiert) wird also für die hier adressierte Aufgabenstellung modifiziert. Abbildung 3.22 zeigt das veränderte Modell. Die entscheidende Anpassung ist durch den breiten Ringförmigen Pfeil wiedergegeben. Er symbolisiert den kontinuierlichen Ablauf des Wechselspiels zwischen Konzeption bzw. Entwurf (System-Design), Umsetzung (Software-Realisierung) und Nutzung des entsprechend kontinuierlich wachsenden Artefakts. Diese Art der Darstellung löst die Phasenorientierung und insbesondere die klare Zweiteilung in Erstellung und Nutzung innerhalb *eines* Revisionszyklusses in STEPS ab. Das kontinuierliche Wachstum des Zielsystems auf der Grundlage des Frameworks – im Gegensatz zu den diskreten Schritten der Versionserstellung in STEPS – ist durch die dreidimensionale Darstellung des zentralen Artefakts repräsentiert. Die Legende erinnert an die disjunkte Unterteilung in fortschreitende *Zyklen* der Reife durch unterschiedliche Stärken der Rahmen. Bezüglich der Aufgaben und des sich entwickelnden Anspruchs ist dies ebenfalls als Kontinuum und relativ zum Start eines Zyklusses zu sehen. Mit jedem Durchlauf – die Anzahl der Iterationen und die Granularität der Dekomposition in Inkremente sind im Grunde beliebig – entwickeln sich die Ansprüche an die Entwicklung aber auch die Fähigkeiten der Entwickler sowie die Tragfähigkeit und Reproduzierbarkeit des Gesamtprozesses weiter. Dies ist das Erfolgskriterium für die Konvergenz des Vorgehens. Jeder Durchlauf vollendet ein inhaltlich und technisch projektabhängig definiertes Inkrement – einen Teil des kontinuierlich wachsenden Systems.

Der Neustart eines Zyklus' ist als Meilenstein und Synchronisationspunkt zu sehen, an dem fertig gestellte Ergebnisse im Sinne von Code-Freezes fixiert werden. Die Erreichung von Inkrementen globaler Ziele unter Einhaltung der geplanten Zeit und der veranschlagten Kosten wird vom Management auf Basis einer aggregierten Darstellung der Ergebnisse einer Ist-Analyse (siehe weiter unten im Text) zusammenfassend beurteilt als Grundlage der Planung des sich anschließenden Durchlaufs. Im Sinne neuer Aufgaben, die sich aus einem neuen zu erstellenden Inkrement des Systems ergeben, ist der Beginn eines neuen Zyklus' tatsächlich als der Start eines neuen Teilprojekts zu sehen – die Reife der damit verbunden Aktivitäten und deren Ergebnisse im Einzelnen nimmt bezogen auf die neuen Herausforderungen wieder zu (symbolisiert durch wieder beginnend zunehmende Rahmendicke in der Abbildung 3.22). Wie sich die Aufwände im Idealfall über dem Projektverlauf verteilen und schließlich in die Fertigstellung des Produktes münden ist in der Abbildung 3.24 dargestellt.

Die Kategorie der Aufgaben, die ausschließlich durch *EntwicklerInnen* als Technologen erledigt werden, haben lediglich noch flankierenden Charakter – „im Hintergrund" der Aktivitäten des kooperativen Prototyping. Dabei werden Anpassungen etwa an White-Box-Schnittstellen des Frameworks vorgenommen. Im Idealfall kann das so zeitnah zum Aufkommen des entsprechenden Bedarfs geschehen, dass dies für den eigentlich Entwicklungsablauf transparent bleiben kann – sich das Framework aus der Sicht der fachlichen Entwicklung quasi proaktiv adaptiv verhält. Mit dem Projektfortschritt und der zunehmenden Reife sowohl des Systems als auch der Befähigung der Fachentwickler zur Arbeit mit den Werkzeugen nimmt die Intensität des dezidiert technologischen Entwicklungsanteils ab. In der Abbildung 3.22 ist dies wiedergegeben durch die (umgekehrt proportional zu den Kernaufgaben) abnehmende Rahmendicke der hellgrauen Boxen,

die die rein technologischen Arbeitspakete symbolisieren. Im Wesentlichen liegen damit auch die Phasen der Software-Realisierung nunmehr in der gemeinsamen Verantwortung aller beteiligten; es gibt keine von den IT-Experten wirklich isoliert bearbeiteten Aufgaben mehr innerhalb des Schemas. Die Nutzervertreter und die Kollaboration zwischen den Disziplinen dominieren das Geschehen.

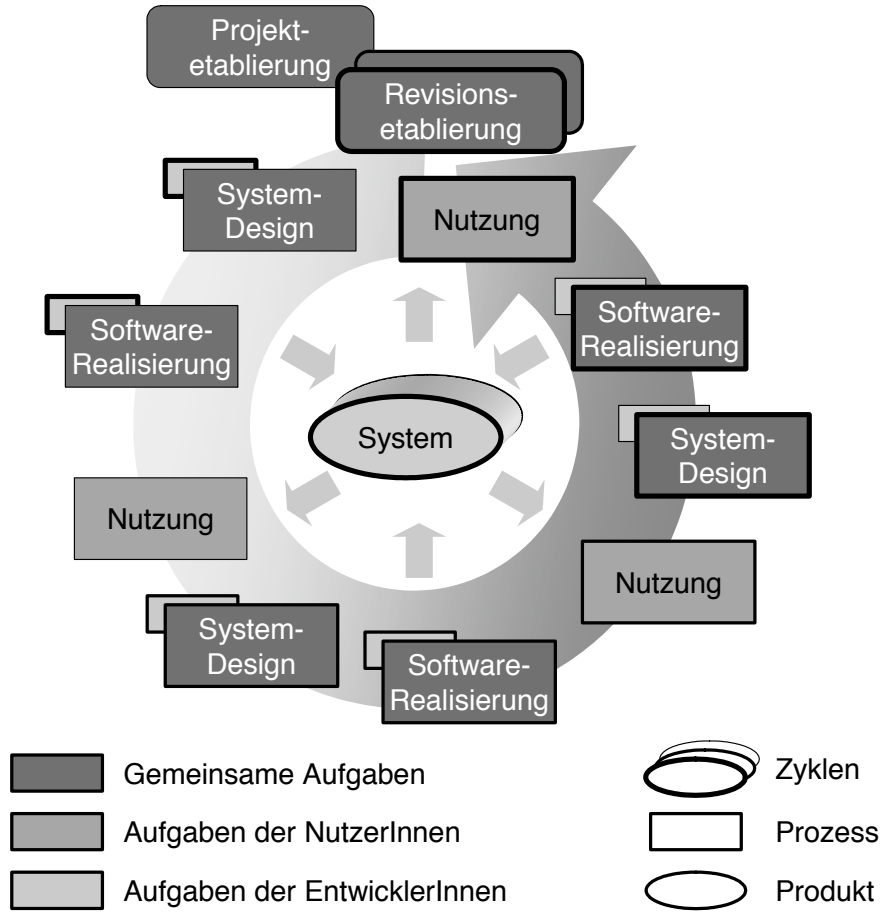


Abbildung 3.22.: Die Adaption des Vorgehensmodells nach STEPS [FRS89] für das interaktive Prototyping auf der Basis des FLAME-Framworks

Um das zu realisieren muss die Technologieentwicklung wie angesprochen flankiert werden von der systematischen Weiterentwicklung des umgebenden Arbeitskontexts, also der Aus- und Umgestaltung von Organisationsstrukturen der beteiligten Unternehmen aber auch der gezielten Personalentwicklung. Um die dafür notwendigen Aktivitäten koordiniert betreiben zu können, orientiert sich das hier beschriebene Vorgehen wie angedeutet am Prinzip der *Integrierten Technologie- und Organisationsentwicklung (OTE)*. Dabei wird insbesondere der Schritt der evaluierenden Anwendung von Zwischenergebnissen der Entwicklung dahin gehend systematisiert, dass einerseits die Nutzung der

neuen Technik immer auch in ihren Wechselwirkungen mit dem Gesamtprozess der Produktion, also beispielsweise mit den definierten redaktionellen Prozessen eines Verlages, untersucht und gestaltet wird. Andererseits werden aber auch die Auswirkungen des Umgestaltungsprozesses auf die persönlichen Fähigkeiten und Bedürfnisse der vom Technologieeinsatz betroffenen Akteure innerhalb jeder Iteration aufs Neue überprüft. Das so genannte Arbeitssystem, welches im Wesentlichen Charakterisiert ist durch die Verflechtung arbeitsorganisatorischer, technologischer und personeller Parameter wird in einem eigenen iterativen Prozess so optimiert, dass das Zusammenspiel möglichst Reibungslos verläuft, und damit die Konvergenz und Zielorientierung des gesamten Entwicklungsprozesses sichergestellt werden kann.

Den Auftakt bildet die Etablierung dieses Gestaltungsprozesses (siehe Abbildung 3.23). Das umfasst im Wesentlichen die Schaffung eines entsprechenden Bewusstseins bei allen beteiligten, aber auch die Einbindung in die administrativen Strukturen der beteiligten Organisationen – etwa die Koordination von Maßnahmen der Personalentwicklung mit den in den Unternehmen etablierten dafür zuständigen Institutionen (Personalabteilung). Die Aktivitäten der Prozess-Etablierung werden bereits in der Phase der Projektetablierung angestoßen. Ihre Intensität wird mit der fortschreitenden Evolution des Arbeitssystems abnehmen – zumindest zwischen den Iterationen variieren. So ist es z.B. denkbar, dass erst spät im Projekt ein Feature eingeführt wird, welches eine explizite Schulung der Beteiligten oder aber die Veränderung der Teamzusammensetzung voraussetzt. So dass auch erst zu diesem späteren Zeitpunkt die initialen Schritte unter Einbeziehung neuer Institutionen und Organisationsstrukturen gegangen werden müssen.

Die sich anschließende *Ist-Analyse* ist auch in der im vorangegangenen Abschnitt beschriebenen Auftaktphase des Projekts am intensivsten Ausgeprägt. Eine wirklich methodisch systematische Untersuchung des Arbeitsumfeldes mit ethnografischen Methoden wird nur dort stattfinden. In der Phase der produktiven Ausgestaltung des Systems erfolgt diese Beobachtung im Rahmen des diskursiven Prozesses der kollaborativen Entwicklungsarbeit. Idealerweise sind die Beteiligten oder zumindest das Team im Ganzen in der Lage einen entsprechenden Prozess der Selbstreflektion zu betreiben. Die strukturgebenden Kriterien leiten sich dabei aus OTE ab. In die Beobachtung einzubeziehen sind:

Organisation Dieser Aspekt erfährt in der Anwendung hier insofern eine Spezialisierung, als dass sowohl die Organisation der vom eigentlichen Zielsystem unterstützten Anwendungsdomäne als auch der Aufbau und die Abläufe während der fachlich orientierten (anwendungsnahen) Entwicklung ins Kalkül gezogen werden müssen. Es ist also einerseits in dem Rahmen zu antizipieren, wie sich der Nutzungskontext unter Einsatz des Systems neu gestalten wird.

Es ist beispielsweise zu fragen, welchen Einfluss das System auf die Zusammensetzung der Zielgruppe hat – wen man möglicherweise als Rezipienten von Inhalten gewinnen oder aber verlieren wird. Weiter Fragen sind zum Beispiel: Wie ändert sich der Ablauf des Unterrichts, in dem das Zielsystem eingesetzt werden soll?

Müssen die didaktischen und inhaltlichen Konzepte, die innerhalb der letzten Iteration eingeführt wurden, in Einklang gebracht werden mit Rahmenlehrplänen?

Qualifikation In gleicher Weise ist die Analyse der Qualifikation der potentiellen Nutzer abzustufen: auf der einen Seite ist sicherzustellen, dass die ursprünglich rein fachlich orientierten Entwicklungsbeteiligten jeder Zeit in der Lage sind, die Produktionsmittel zu beherrschen. Die weitgehende Angleichung der Werkzeuge an die individuellen fachlichen Tätigkeiten und damit Bedürfnisse wird im Allgemeinen an Grenzen stoßen. Nicht alle der angestammten Handgriffe sind so in mediale Mittel zu übersetzen, dass die Handhabung völlig reibungslos und intuitiv übertragen werden kann. Einige der Eigenarten des IT-Einsatzes bleiben definierender Bestandteil der technischen Anforderungen. Das Prinzip der Co-Adaption erfordert also eine, wenn auch behutsame, Angleichung der Fähigkeiten der Fachleute. Die entsprechenden Differenzen und damit Risikopotentiale für einen konstruktiven Verlauf des diskursiven und explorativen Prozesses sind in jedem Zyklus zu ermitteln. Dazu gehört neben den Fähigkeiten der Nutzung der technischen Mittel auch die sozialen und kommunikativen Kompetenzen, die nötig sind um kollaborativ an einem singulären Produkt arbeiten zu können. Ein schwieriger und kaum steuerbarer Aspekt dabei ist die Aufrechterhaltung der Neugier und Offenheit für sich auftuende Gestaltungsspielräume.

Auf der anderen Seite ist hier wieder die Perspektive der Nutzung im Zielkontext zu wahren. Es besteht das Risiko, dass durch die mit dem Entwicklungsprozess einhergehenden, zum Teil unterschwelligen und informellen Lernprozesse, selbst die Domänenexperten, die eigentlich die Endanwender vertreten sollten, das Gefühl für die Fähigkeiten als Voraussetzung für die Nutzung der Zielsoftware verlieren. Bei der evaluierenden Anwendung muss also immer wieder bewusst die Perspektive der eigentlichen Zielgruppe eingenommen werden.

Technik Die Analyse der entstehenden technischen Lösung im Rahmen der Nutzungsphasen ist das zentrale Element der Qualitätssicherung. Durch die kontinuierliche Verfügbarkeit eines integrierten, ablauffähigen Artefaktes und die permanente Nutzung der Entwicklungsergebnisse können sowohl Fehler auf der Ebene der aktuell vorgenommenen Änderungen (in gewissem Sinne Modultests) erkannt werden, als auch die Übereinstimmung der Lösung mit den Nutzeranforderungen sichergestellt werden.

Im nächsten Schritt der *Kreation von Gestaltungsoptionen*, werden die Möglichkeiten geprüft, die zur Verfügung stehen, um erkannte Defizite auf den drei Ebenen auszugleichen. Diese Funktion erfüllt der experimentelle Charakter des Vorgehens: zum einen durch das kooperativen Mock-Up-Design in den frühen Phasen der Entwicklung und zum anderen durch das sich anschließende explorative Prototyping am Zielartefakt.

Aus dem Spektrum der Gestaltungsoptionen werden im nächsten Schritt tatsächliche Ziele für den kommenden Überarbeitungszyklus abgeleitet. Das angestrebte *Soll-Konzept* beschreibt wieder unterteilt in die Bereiche Organisation, Qualifikation und Technik, welche der Gestaltungsmöglichkeiten tatsächlich realisiert werden müssen bzw.

können. Eine Fixierung der technischen Gestaltungsoptionen besteht jeweils im Ausprägungsschritt der Prototyping-Umgebung, als vorbereitende Phase (grün hinterlegt in Abbildung 3.20 auf Seite 197 die eigentlich Entwicklung. Dabei sind die resultierenden Wechselwirkungen der Ebenen untereinander zu beachten. Die Ziele der Weiterentwicklung des der technologischen Aspekte des Zielsystems, die sich in veränderten Entwicklerschnittstellen am Framework niederschlagen, werden primär an den Zielen der Kompetenzentwicklung der designierten Nutzer ausgerichtet. Einfach formuliert bedeutet das, dass man nur die dezidiert technischen Weiterentwicklungen vornimmt, also die neuen Features ausprägt, deren Handhabung sich die Fachexperten zutrauen bzw. bei denen sie bereit sind das Erlernen der Handhabung als Ziel zu akzeptieren und zu verfolgen.

Die Art, wie sich Abläufe und Zuständigkeiten verändern bestimmt die Zielsetzungen für die Umgestaltung des arbeitsorganisatorischen Rahmens. Auf Grund der sehr engmaschigen Iterationen in der Phase der Systemausgestaltung, wird die Zieldefinition in der Praxis eher informell vollzogen werden. Entscheidend werden also weniger fest vorgegebene Dokumentationsstrukturen und Terminregime sein, als vielmehr die prinzipielle Durchsetzung dieser Schritte. Es ist die Aufgabe der Projektmoderation, bei allen Beteiligten das Bewusstsein für die Notwendigkeit einer kontinuierlichen Zieldefinition bzw. -justierung zu schaffen. Dabei sollte verdeutlicht werden, das es eine Struktur an Zielsetzungen gibt, die es zulässt, globale Ziele hierarchisch in Teilziele zu untergliedern bis hin zu Zuständen, die am Ende eines Tages erreicht sein sollen. Auch die Sensibilität für den nötigen permanenten Abgleich der Teilziele und ihrer Konsistenz im Verhältnis zu globalen Zielen muss gezielt geschaffen und geschult werden. Dementsprechend interveniert ggf. der querschnittliche Prozess der *Qualifikation zur Partizipation* (der vertikal verlaufende Prozess rechts in der Abbildung 3.23). Gradmesser für den Erfolg dieses Schritts wird eine in dem Sinne gelungene Umsetzung der Ziele sein, dass die nächste Iteration der evaluierenden Nutzung des Arbeitssystems eine größere Übereinstimmung mit übergeordneten Zielen nachweist.

Die Umsetzung der Ziele in ein neues Arbeitssystem ist der nächste Schritt. Bezüglich der Software-Realisierung ergibt sich dabei eine Modifikation des in [KRW96] vorgestellten und angewendeten Konzepts des OTE. Die dort vorgesehenen einfachen, in der Regel konfigurierenden Anpassungen im Zuge des Schritts der iterativen Ausgestaltung eines Arbeitssystems für die evaluierende Nutzung unterscheidet sich vom Schritt der Software-Realisierung im Rahmen des STEPS-Prozesses, an den OTE anknüpft (siehe Schema [KRW96] Seite 102). Diese beiden Schritte gehen im hier vorgeschlagenen Vorgehen ineinander auf. Der Schritt Realisierung der technischen Teile des Soll-Konzepts umfasst also die eigentliche Entwicklungsarbeit (Schritte des Entwurfs und der Umsetzung), die auf das zentrale Artefakt des Systems wirkt (das *„System“* im Zentrum der Abbildung 3.22). Der Einsatz des Frameworks und der domänenspezifische Entwicklungsumgebung machen es möglich die Grenze zwischen konfigurierenden Feinabstimmungen an fertigen Systemen und Schritten der technologischen Entwicklung aufzuheben. Das interaktive Prototyping sollte nun auf produktivem Niveau im Wesentlichen getrieben werden von den Fachentwicklern. Das weiterentwickelte technische System

als Ergebnis des Anpassungsschritts wird Teil des überarbeiteten Arbeitssystems als Grundlage der nächsten Iteration der evaluierenden Nutzung.

Parallel dazu werden die Ziele der Organisationsentwicklung sowie der Personalentwicklung bzw. Qualifizierung umgesetzt, indem die entsprechenden Prozesse implementiert werden. Sollte es sich beispielsweise als sinnvoll und nötig erweisen, dass jede Aktivität eines Autors zum Einpflegen und Strukturieren von Inhalten zwangsläufig einen Schritt der redaktionellen und didaktischen Aufbereitung eines Lektors nach sich zieht, ist dies zu einem Standard-Ablauf des nächsten Entwicklungsschritts zu machen. Dies ist ggf. sogar technisch zu unterstützen. Die Einführung einer Komponente zur Steuerung derartiger *Workflows* in das ohnehin verteilte personalisierte System der Entwicklungsumgebung ist als zukünftige Aufgabe denkbar. Bestehen Konflikte zwischen technisch notwendigen Aspekten der Systementwicklung und den Fähigkeiten der einbezogenen Fachleute bezüglich ihrer Mitwirkung, sind diese gemäß des Soll-Konzepts beispielsweise durch Schritte der Schulung oder der informellen Anleitung (coaching) aufzulösen. Der querschnittliche Prozess, der den Beteiligten immer wieder ins Gedächtnis ruft, dass das Projekt von seinem kollaborativen, tätigkeitsorientierten Charakter lebt (*Qualifikation zur Partizipation → Moderation und Schulung*), ist hierbei angehalten eine kontinuierliche wechselseitige Weiterentwicklung der Kompetenzen zwischen den Disziplinen zu forcieren.

Bezogen auf die Nutzung im Sinne der Endanwendung (im Gegensatz zur Anwendung während der fachlichen Entwicklung) bedeutet das zum Beispiel, dass die während der Analyse möglicherweise erkannten Diskrepanzen zwischen den Einsatzmöglichkeiten des Systems im Unterricht und den Vorgaben von Rahmenlehrplänen durch ein Soll-Konzept adressiert werden müssen, welches die resultierenden Risiken ausräumt. Das kann im einfachsten Falle bedeuten, dass ergänzende Handreichungen vorgaben machen, wie das System in eine konkrete Unterrichtssituation einzubinden ist. Das Beispiel von Schulen als Kontext für die Endanwendung illustriert ferner die Schwierigkeit Risiken zu antizipieren und zu behandeln, die etwa die mediendidaktische Kompetenz der Lehrerschaft oder die vorauszusetzende technische Ausstattung der Schulen betrifft. Auch dabei sollte die weitgehende Kongruenz zwischen der fachlichen anwendungsorientierten Systemausgestaltung bzw. der Nähe der direkt beteiligten Fachexperten zum Zielkontext einerseits und der eigentlichen Anwendung helfen, den Bezug kontinuierlich zu wahren. Auch dabei muss ein gezielt geführter Diskurs durch den querschnittlichen Prozess der Moderation in Gang gehalten werden.

Der Einfluss des Framework-Einsatzes auf die integrierte Technologie- und Organisationsentwicklung wirkt in allen Schritten. Die Artefakt-Zentrierung auf der Basis des Prototypen (Framework) ist das Regulativ für den Prozess. Die permanente Präsenz eines greifbaren materiellen Diskursgegenstandes gibt den Schritten jeweils ihre Richtung vor. Das Framework macht dabei Vorgaben, die das Finden eines kleinsten gemeinsamen Nenners im Diskurs zwischen den Disziplinen zwangsläufig machen. So wird man bei der Erforschung des Zielkontexts immer das Kriterium anlegen, ob und wie sich das Framework und der Ansatz der adaptierbaren Entwicklungsumgebung einsetzen lassen werden. Man wird etwa primär nach Anhaltspunkten suchen, die die Bereitschaft

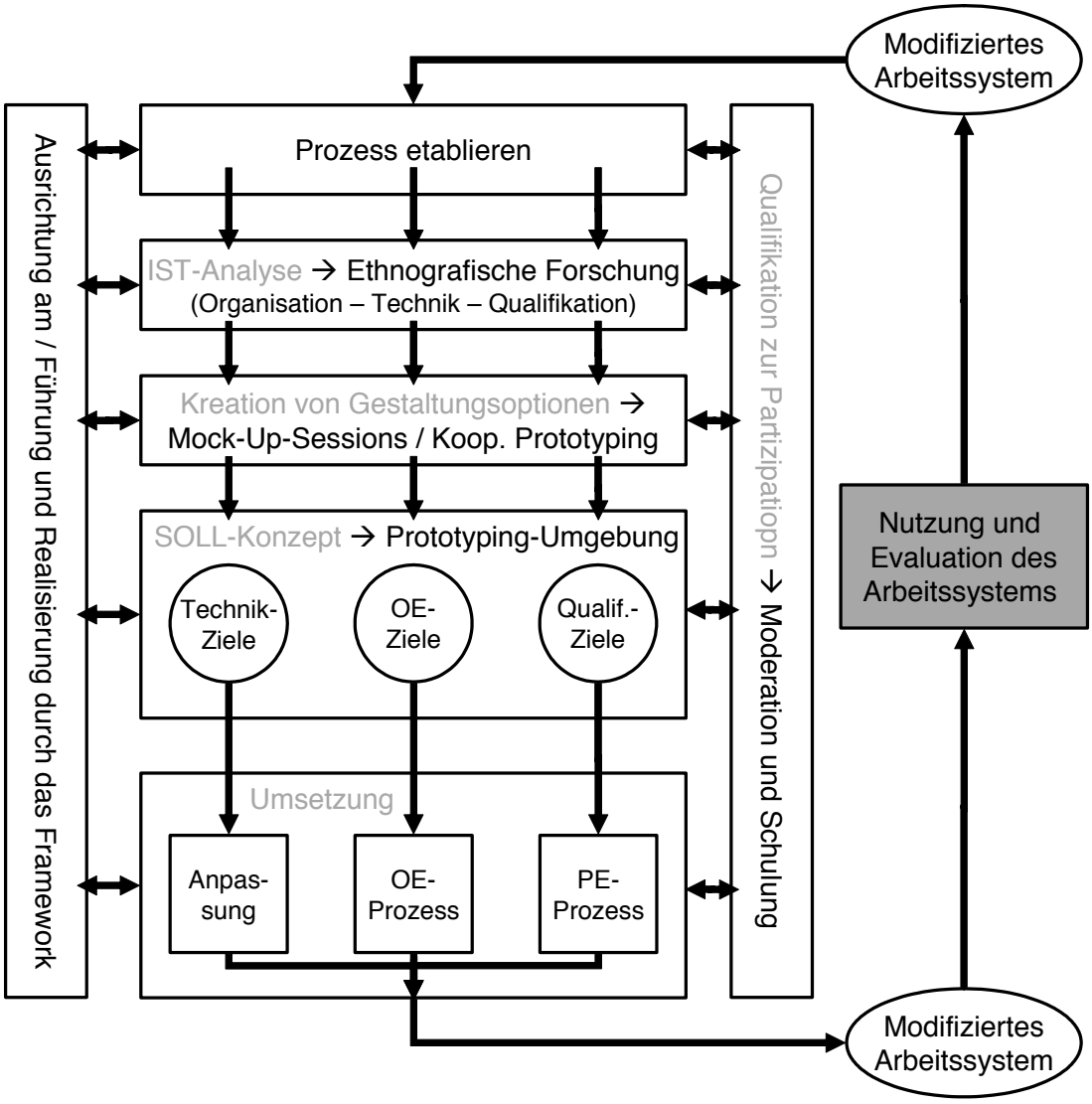


Abbildung 3.23.: Die Interpretation von OTE (vgl. etwa [KRW96]) für das interaktive Prototyping auf der Basis des FLAME-Frameworks

potentieller Fachentwickler erkennen lassen, sich mit einer neuen Form der Arbeit auseinanderzusetzen.

Vorhandene, wieder verwendbare Software-Teile bzw. die framework-inhärenten Architekturmuster machen Vorgaben bezüglich der technischen Gestaltungsspielräume. Das wiederum bildet einen gewissen Rahmen für den technologischen Teil des Soll-Konzepts. Die Umsetzung selbst ist ganz zentral beeinflusst vom Framework und den Möglichkeiten der fachlich individualisierten Entwicklung.

Nach einer fortgeschrittenen Anpassung des Frameworks ist die Entwicklung fast ausschließlich fokussiert auf die Möglichkeiten des abgeleiteten interaktiven Prototypen und den durch Co-Adaption daran ausgerichteten Fähigkeiten der Entwickler bzw. auf die Abläufe der umgebenden Organisation. Die vermeintliche wachsende Beschränkung von Gestaltungsspielräumen durch die Spezialisierung auf den Einsatzzweck ist schließlich das entscheidende Kriterium für die Konvergenz des Projekts im Sinne der globalen Zielstellung.

3.2.4. Implikationen für das Projektmanagement

Die Verteilung der Aufwände und damit das Wechselspiel der Aktivitäten innerhalb der hier anvisierten Projektverläufe sind in Abbildung 3.24 größenordnungsmäßig wiedergegeben. Die Darstellung ist angelehnt an die des in 2.12 auf Seite 60 als Grundlage vorgestellten *Rational Unified Process* [JBR99]. Dabei wird zunächst deutlich, dass die ursprüngliche Trennung zwischen der Ableitung technischer Anforderungen aus den Nutzeranforderungen und deren anschließende Überführung in technologische Konzepte (design) sowie schließlich in die Implementierung aufgehoben wird. Diese in schneller Folge iterierten und zum Teil lediglich implizit ausgeführten Aktivitäten sind zusammengefasst in den weitgehend an der Tätigkeit des interaktiven Prototyping orientierten Aufgabensträngen der *technischen Entwicklung* und der *Content-Entwicklung*. Die unterstützende Leistung der technischen Entwicklung wird jeweils zum Auftakt (Kollaboration, Co-Adaption) einer Iteration der eigentlichen produktiven, fachinhaltlichen Umsetzung neu erkannter Anforderungen benötigt. Anforderungen ergeben sich jeweils aus der evaluierenden Nutzung bzw. zum Projektauftakt aus der Erforschung und der Arbeit an den Mock-Ups.

Die Abbildung 3.24 verdeutlicht, dass die Phase der Systemausgestaltung ausgedehnt werden soll in die Transition und den Nutzungskontext.

Die Kardinalitäten der Iterationen sind lediglich exemplarisch zu verstehen. Je nach Projekt, kann sich beispielsweise der Projektauftakt auch über mehr als eine Iteration erstrecken. Das Ende einer Iteration markiert idealerweise ein strukturiert durchgeführtes Meilenstein-Review. Während der Phase der Ausführung (construction) bietet sich dafür der Abschluss der Phase der evaluierenden Nutzung nach einem durchgeführten Schritt der integrierten Technologie- und Organisationsentwicklung an, die eine neue *Revision* zusammenfasst.

Risikobewertung und -beschränkung

Das in den Grundlagen diskutierte Manko der häufig riskanten Entkopplung von Phasen der technologischen Entwicklung von denen der inhaltlich fachlichen Konzeption in klassischen Projekten der Produktion medialer, interaktiver Systeme, räumt der beschriebene Ansatz systematisch aus. Auf Grund der Tatsache, dass das gesamte Entwicklungsgeschehen vom kooperativen und partizipativen Charakter lebt und durch den

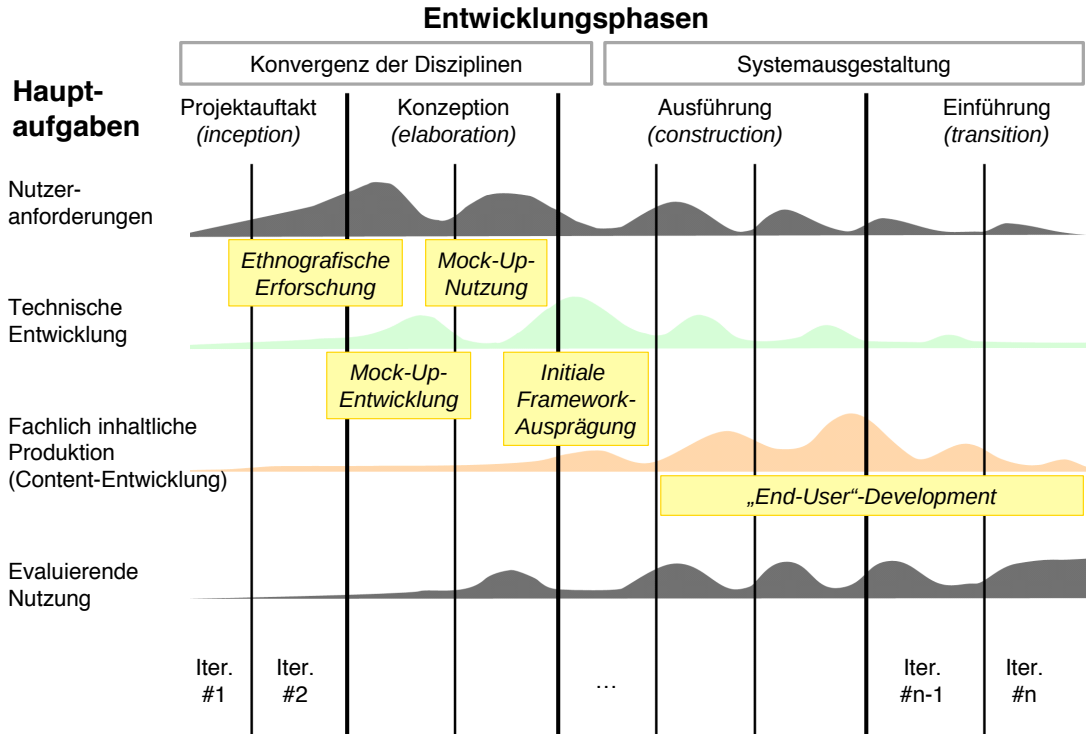


Abbildung 3.24.: Verteilung der Aufwände über Iterationen in Anlehnung an RUP

Diskurs zwischen allen maßgeblich Beteiligten getragen ist, wird die Situation vermieden, dass Auftraggeber und Projekt-Controlling über längere Zeit den Status des Projektes nicht erfassen können. Am permanent verfügbaren Artefakt und an den auch für die nicht technisch versierten, fachlichen Projektkontroller leicht verständlichen Entwicklerschnittstellen, kann der Projektfortschritt jeder Zeit abgelesen werden.

Die zwangsläufige regelmäßige Rückkopplung zwischen Produktion und Anwendung in sehr kurzen Zyklen und die kollektive Bewertung von Gefährdungspotentialen bzw. Eintrittswahrscheinlichkeiten von Problemen ermöglicht es dem Management jeder Zeit die Gesamtsituation zu beurteilen.

Um von den Vorteilen bewährter Methoden der Risiko-Analyse bzw. der Aufwandsschätzung profitieren zu können, ist es denkbar den kollektiven Prozess der Analyse und der Ableitung des Soll-Konzepts beispielsweise mit den Mitteln der Delphi-Methode (vgl. [See79]) zu strukturieren. So könnte man etwa die Projektbeteiligten zunächst unabhängig voneinander einschätzen lassen, was an Aufgaben noch zu erledigen ist, und welche Features noch umzusetzen sind als Zwischenschritte auf dem Weg zur Erreichung globaler Ziele. Gibt es dabei vom Konsens stark abweichende Ansichten, kann dies in kommentierter Form zurückgegeben werden in die Runde, um in der Selbstreflektion ergründen zu lassen, ob etwa die Liste an vermeintlich noch umzusetzenden Systemteilen und Inhalten, nicht doch den Rahmen des tatsächlich benötigten sprengt. Es ist die

Aufgabe der Moderation, diesen möglicherweise in sich wiederum iterativen Prozess in einen Konsens münden zu lassen.

Es bleibt jedoch anzumerken, dass gerade bei der Konvergenz der Entwicklungsbemühung in Richtung der Erreichung globaler Ziele Konflikte mit dem evolutionären Charakter des Vorgehens auftreten können. Die Erfahrungen aus dem Evaluationsprojekt etwa zeigen, dass fachlich fokussierte Redakteure mit der Entdeckung der Möglichkeiten, die der Technologieeinsatz eröffnet auch die Ansprüche systematisch steigern. Gerade auch die beliebige inhaltliche Skalierbarkeit des Systems kann in Paarung mit dem Drang nach wissenschaftlicher Vollständigkeit zu einer Sprengung von Zeit- und Kostenrahmen führen. Die ansonsten gewünschte Eigendynamik, die die intrinsische Motivation in den Entwicklungsprozess trägt, muss an einem solchen Punkt durch die Projektleitung bzw. die -moderation kontrolliert werden.

Wiederverwendung

Die Wiederverwendung von Entwicklungsergebnissen nimmt einerseits durch den Framework-Einsatz und andererseits durch die integrierte Entwicklung von Technologie und Arbeitsorganisation eine Sonderstellung ein. Die Investition in die Entwicklung des Frameworks aber auch in die der Ausprägung einer Entwicklungsumgebung für einen Kontext lohnt sich grundsätzlich erst durch den wiederholten Einsatz. Der ohnehin anfallende zusätzliche Aufwand für die Etablierung einer allgemeingültigen und anpassbaren Infrastruktur wird dabei gesteigert durch die Kosten für die Adaption und Einführung der Prozesse sowie die Einarbeitung der Beteiligten.

Ideal ist also eine strategische Einführung des gesamten Arbeitssystems (im Sinne von OTE [KRW96]). Für eine langfristige Nutzung in immer wieder ähnlichen Entwicklungsprojekten. Nur so kann das im Iterativen Prozess der Exploration akkumulierte und im Framework manifestierte Domänenwissen effektiv nachgenutzt werden. Damit würde die Wiederverwendung den Ausgangspunkt markieren für eine fortgesetzte Adaption und Verfeinerung sowohl des instrumentellen Teils als auch der arbeitsorganisatorischen und personalentwicklerischen Rahmenbedingungen.

Beispiele für sinnvolle Einsatzszenarien sind etwa Fachbereiche von Schulbuchverlagen, die immer wieder ähnliche Software-Produkte in immer gleichen personellen und organisatorischen Konstellationen erstellen. Variationen ergeben sich im Wesentlichen in der Präsentation und der Strukturierung und Auswahl der eingebundenen Inhalte und Medien. Die Umsetzung der beinahe invarianten Interaktionsformen, Navigationsmechanismen sowie Aufbau- und Ablaufstrukturen der Software können aus einem auf den fachlichen Kontext spezialisierten Fundus an fertigen Komponenten schöpfen. Für die individualisierten Entwicklerschnittstellen gilt das gleiche. Die Art und Weise, wie Redakteure oder Autoren mit Inhalten und Funktionalitäten disponieren können und wollen ist über viele Projekte die gleiche. Auch die beteiligten Personen und die Abläufe

der Zuarbeit untereinander ähneln sich immer wieder. Zumindest das fachliche Verständnis und die medialen Kompetenzen der Akteure sind immer vergleichbar, sodass es sich lohnt eine entsprechende Entwicklungsumgebung anzupassen.

Qualitätssicherung

Die Qualität der hier entwickelten Software-Produkte wird zentral durch den interaktiven kollaborativen endnutzer-orientierten Entwicklungsprozess sichergestellt. Die Übereinstimmung des Ergebnisses mit den Anforderungen wird über die Evolution des Systems und den kontinuierlichen Abgleich mit den sich während der Entwicklung ihrerseits anpassenden Zielen erreicht. Die Ausrichtung des Entwicklungsgeschehens an den fachlichen Tätigkeiten und die unmittelbare kurzzyklische Adaption an die Bedürfnisse sowie die Co-Adaption der Akteure erlauben eine sehr präzise Angleichung des Systems an die Anforderungen – trotz der Tatsache, das diese grundsätzlich bewegliche Ziele sind.

Durch die Direktheit der Umsetzung von Nutzerwünschen – im Idealfall selbstständig durch die Fachentwickler (Nutzervertreter) realisiert – werden Missverständnisse a priori vermieden. Reibungsverluste auf dem Weg von der Anforderungserhebung über die Spezifikation bis hin zur Umsetzung sind beinahe ausgeschlossen. Die Wahrscheinlichkeit, dass Systemmerkmale auf Verdacht antizipiert werden – über das fachlich Notwendige bzw. technisch Machbare hinaus – ist beschränkt. Darüber hinaus stellt die enge Zusammenarbeit zwischen allen beteiligten sicher, dass die gemeinsame Verständnisgrundlage systematisch konsolidiert wird. Der diskursive Prozess lässt die Sichtweisen und Kompetenzen der Beteiligten konvergieren. So scheiden etwa auch unterschiedliche Begriffs- und Modellbildungen als Quelle für Missverständnisse weitgehend aus. Je nach Zusammensetzung des Projekts besteht sogar die Möglichkeit von einem Basar- bzw. Community-Effekt (vgl. [Ray00]) zu profitieren, vergleichbar mit dem der Opensource-Entwicklung.

Die fachliche Qualitätssicherung setzt ferner auf das Prinzip des Fokussierens auf das fachlich Notwendige in Verbindung mit dem Mut jeder Zeit zu konsolidieren und zu Refaktorisieren im Sinne von [Bec00]. Die Hemmschwelle, Änderungen an gefunden Lösungen vorzunehmen oder gar nicht mehr relevante Teile zu verwerfen, soll dabei durch die Einfachheit der Entwicklungsschnittstellen und die Steuerung des Vorgehens in Richtung eines experimentierenden Arbeitens gesenkt werden. Die Kultur der Verteilung der Verantwortung (ebenfalls in Anlehnung an XP) also auch der Aufgabe von Besitzständen bezüglich der eigenen Leistung ermöglicht es dem kollektiven Prozess der Verbesserung wirksam zu werden.

Die direkte Zusammenarbeit – in XP etwa in der Forderung nach der Programmierung in Paaren manifest – fördert ebenfalls die Qualität. Auch im hier verallgemeinerten diskursiven, kollaborativen Prozess des interaktiven Prototyping wirkt eine direkte Wechselseitige Kontrolle zwischen den Entwicklern. So werden bereits direkt im Rahmen der aktuell verrichteten Tätigkeit gegenseitig Fehler erkannt und korrigiert. Der transdisziplinäre Charakter der Teamzusammensetzung und die aus der Fachlichkeit heraus jeweils

eingenommenen unterschiedlichen Blickwinkel ermöglichen es dabei, Fehlerquellen, die aus der Routine heraus entstehen (Betriebsblindheit), zu erkennen.

Das beinahe nahtlose Wechselspiel zwischen Anwendung und Entwicklung wird ermöglicht durch die kontinuierliche Verfügbarkeit eines integrierten ablauffähigen Systems. Lediglich abhängig von der korrekten Funktionsweise des beschriebenen Meta-Objektprotokolls führt jeder darüber ausgeführte Entwicklungsschritt zu einer technisch konsistenten und stabilen Systemversion. Ein Erfolgskriterium bei der Erreichung hoher technischer Qualität ist wiederum die möglichst häufige Wiederverwendung und damit nachhaltige Evaluation aller Komponenten bzw. ihrer fachlichen Konkretisierungen. Das gilt in besonderem Maße auch für das MOP und den Renderer der Basisebene der Architektur. Letzterer ist als Teil des Frameworks Bestandteil aller damit entwickelten Anwendungen und verantwortlich für die zuverlässige sowie effiziente Umsetzung der Meta-Objekte in die eigentliche Präsentation. Das heißt die Stabilität und die Ausführungseffizienz ist ausschließlich von dieser Komponente abhängig.

Ein bisher unbehandeltes Manko des Ansatzes stellt die fehlende *Tracebility* zwischen Anforderungen bzw. Änderungswünschen und entstehenden Software-Lösungen dar. Insbesondere die Möglichkeit sehr einfach und schnell Änderungen vorzunehmen und der kleinschrittige evolutionäre Charakter des Systemwachstums erschweren eine unter Umständen (im Falle von Revisionsbedarf) nötige Nachvollziehbarkeit zwischen dem Auftreten eines Bedarfs der Übertragung in eine entsprechende Lösung.

Systempflege und -änderung

Die Meta-Level-Architektur, die eine externalisierbare und serialisierbare Repräsentation von Systemkomponenten umsetzt in die eigentliche ablaufende Software, erlaubt Korrekturen zu jedem Zeitpunkt des Lebenszyklus. Das ermöglicht es zum Beispiel sowohl inhaltliche als auch funktionale Aktualisierungen des Systems sogar zur Laufzeit vorzunehmen. Dabei werden die veränderten XML-serialisierten Beschreibungen über eine HTTP-Verbindung zugänglich gemacht. Das Framework setzt diese unmittelbar und ohne Kompiliervorgang in veränderte Funktionalitäten um. Die Änderungen sind lediglich abhängig von der Frameworkversion des Clients. Wechselwirkungen mit der Betriebssystemumgebung des Nutzers sind jedoch ausgeschlossen.

Nach der Übertragung in den eigentlichen Nutzungskontext profitiert man weiterhin von den evolutionär an das Domänenverständnis angepassten Entwicklungsschnittstellen. Der eigentliche breite Nutzerkreis teilt naheliegenderweise exakt dieses Domänenverständnis und ist nun seinerseits in der Lage die Werkzeuge zu anzuwenden. Je nach Produktstrategie kann dem Nutzer eingeräumt werden, diese Schnittstellen zu nutzen, um Anpassungen an die eigenen Bedürfnisse vorzunehmen. So können etwa Individualisierung lokaler Installationen vorgenommen werden – beispielsweise Anpassungen, die ein Lehrer für einen bestimmen Einsatzzweck im Unterricht vornehmen möchte.

Es sind aber auch Modelle denkbar, die Ergebnisse solcher Weiterentwicklungen aus dem Kontext der Nutzung heraus der Gemeinschaft der Nutzer zugänglich machen. Dabei

erlaubt etwa die Ausprägung der Architekturkonfiguration mit einem zentralen Server Kollaborationsmodelle, die denen von Wikis entsprechen. Um eine größere Kontrolle über die Veröffentlichung gemeinschaftlich weiterentwickelter Versionen zu erhalten, ist es nötig mehrstufige Freigabeprozesse in der Art von Open-Source-Projekten zwischenzuschalten.

3.3. Unterstützung kooperativer transdisziplinärer Wissensarbeit

Als ein zentrales vom hier vorgestellten Ansatz adressiertes Problem wurde die Schwierigkeit der Explizierung fachlicher Zusammenhänge als Grundlage ihrer Übertragung in technische Repräsentationen angeführt. Was die Tatsache einschließt, dass innerhalb der hier adressierten Kontexte, die Antizipierung und Vermittlung der technologischen Potentiale für die Unterstützung der Domäne als problematisch gelten können. In Verbindung mit der zum Teil tatsächlich herrschenden Unbestimmbarkeit konkreter Ergebnisse auf Grund des Forschungscharakters der Bemühungen wird deutlich, dass sich ein klassischer dokumentenorientierter Entwicklungsprozess kaum etablieren lassen wird. Die bei der Entwicklung interaktiver, direkt reaktiver Systeme auch de facto kaum verbreitet Arbeit mit Dokumenten nach festen am Prozess orientierten reproduzierbaren Strukturen, die durch alle betroffenen gleichermaßen verstanden werden, führt neben dem Fehlen einer belastbaren Arbeitsbasis auch zu einer schlechten Steuerbarkeit und Reproduzierbarkeit typischer Projekte.

Um dem zu begegnen implementiert der beschriebene Entwicklungsprozess (inhärent) Mechanismen, die die Schaffung einer gemeinsamen Verständnisbasis als Ausgangspunkt nimmt für einen gezielten Austausch von Wissen und Kompetenzen. In gewisser Weise tragen diese Mechanismen auch zur „Gewinnung“ von Wissen bei. Der Ansatz bedient sich dabei so genannter personifizierender oder auch sozialisierender Strategien des Wissensmanagements, die den Transfer des Wissens primär direkt von Person zu Person und durch die Dynamik innerhalb einer Gruppe anstrebt (siehe dazu die entsprechenden Grundlagenabschnitte 2.3.6 auf den Seiten 2.3.6 ff.).

3.3.1. Wissensmanagement als Teilprozess

Der beschriebene, forcierte Prozess der kollaborativen Arbeit an einem zentralen Artefakt, der getrieben ist vom Diskurs und dem engen Wechselspiel zwischen der fachlichen Systemausgestaltung und der Anpassung bzw. Weiterentwicklung der technologischen Grundlage dafür, impliziert auf welche Weise der Dialog zwischen den Disziplinen zustande kommen soll.

Der Zusammenhang zwischen der produktiven Arbeit und der Vermittlung von Wissen und Kompetenzen kann Analog zu den Ergebnissen der Untersuchung von [BR06]

verstanden werden. Die Autoren untersuchen, welche informellen Lernprozesse die konsequente Anwendung von Prinzipien des Extreme Programming induzieren sollten. Sie stellen dazu den Werten und Prinzipien, die XP zu seiner Definition nutzt Prinzipien gegenüber, die den Arbeitsprozess *lernförderlich* gestalten. Die in [BR06] zusammengestellten Prinzipien sind:

- Ganzheitliche Handlung
- Handlungsspielraum
- Problem- und Komplexitätserfahrung
- Soziale Unterstützung
- Individuelle Entwicklung
- Feedback

Die Korrespondenzen zwischen einigen dieser Kriterien und den Charakteristika des hier beschriebenen Vorgehens lassen sich auf Grund der Anlehnung an XP im Analogieschluss von den Erkenntnissen in [BR06] übertragen.

Etwa der Wert der **Kommunikation und der Kollaboration** in seiner Ausprägung als direkt partizipative Zusammenarbeit an einem gemeinsamen Artfakt schafft lernhaltige Situationen im Sinne von [BR06]. Neben der verbalen Kommunikation in der direkten Zusammenarbeit, also eine eher externalisierende Vermittlung expliziten Wissens, wird der Austausch insbesondere auch dadurch gefördert, dass Sachverhalte direkt am Objekt demonstriert werden können – Handlungsweisen vorgeführt bzw. nachgeahmt und trainiert. Der Wert folgt damit nicht nur dem Prinzip der *Sozialen Unterstützung* sondern ermöglicht durch den partizipativen und transdisziplinären Charakter über den gesamten Entwicklungszyklus auch *das ganzheitliche Handeln.*

Die unmittelbare Korrespondenz ist beim Wert **Feedback** offensichtlich. Die Autoren von [BR06] identifizieren eine schnelle Rückmeldung zu einer ausgeführten Aktion als entscheidend für den Lernerfolg. Genau das ist wiederum ein entscheidender Wert, der das Wesen des beschriebenen Vorgehens bestimmt. Einerseits gibt es permanente direkte Rückkopplungszyklen zwischen den Beteiligten des diskursiven Prozesses. Beginnend mit der ethnographischen Untersuchung, also der *teilnehmenden* Untersuchung des Domänen- bzw. des designierten Projektkontexts über das initiale explorative, kooperative Prototyping bis hin zur eigentlichen gemeinschaftlichen Produktion lebt der Gesamtprozess von der wechselseitigen Reflektion zu Tätigkeiten und Ergebnissen. Andererseits ermöglicht das interaktive Prototyping auf der Grundlage des Frameworks ein permanentes Feedback zwischen der eigenen Arbeit und deren Auswirkungen auf das Artefakt aber auch auf den Umgebenden Prozess.

Auch, wenn der hier vorgestellte Ansatz das Ziel verfolgt, technologische Aspekte eines Software-Entwicklungsprozesses hinter die fachlich inhaltlichen Gestaltungsprozesse

weitgehend zurücktreten zu lassen, birgt das Vorgehen noch immer Bedarf nach umfangreichen Veränderungen: in der Arbeitsweise, in der Nutzung der eingesetzten Mittel, etc. Schließlich werden Bleistift und Paper beispielsweise – wenn auch behutsam – doch abgelöst. Es ist die Aufgabe des Coachings, die daraus resultierende **Problem- und Komplexitätserfahrung** so zu kanalisieren, dass sie die Sogwirkung erzeugt, die die Beteiligten dazu bewegt intrinsisch motiviert, neue Möglichkeiten zu erforschen. Der Erfolg eines Projektes hängt also davon ab, ob die **Problem- und Komplexitätserfahrung** einen Beitrag zur Lernförderlichkeit leistet oder durch Überforderung zum Hemmnis wird.

Das Konzept der systematischen Co-Adaption ist die primäre Grundlage des für die Wissensvermittlung und in der Fortsetzung für die emergente Gewinnung neuer Erkenntnisse notwendigen Wertes **Handlungsspielraum**. Die gezielte Angleichung von Werkzeugen und Produktionsverhältnissen an individuelle Bedürfnisse und Sichtweisen und die bewusste Heranführung auch nicht technischer Fachexperten an die Systemausgestaltung eröffnet ein Maximum an Freiheitsgraden zur aktiven Partizipation sowohl bei der Ausgestaltung des eigentlichen Zielartefaktes als auch bei der Gestaltung der Arbeitsabläufe. Möglich wird dies durch die technologische Flexibilität des Frameworks und auch durch die Tätigkeitsorientierung des Vorgehens.

Vordergründig ist das den Prozess leitende Prinzip **Lernen Lehren** ein Instrument mit dem die Projektleitung bzw. aus operativer Sicht das Coaching sicherstellen kann, dass Wissensflüsse tatsächlich in Gang gesetzt werden. Durch die Betonung der Bedeutung des Wissenstransfers und die kontinuierliche Anleitung zur Erschließung und Vermittlung von Wissen aber auch durch die organisierte Bereitstellung von Hilfsmitteln muss eine Kultur etabliert werden, die mit Informationsdefiziten (bzw. Wissenslücken) weitgehend von sich aus konstruktiv umgehen kann.

Der Wissensmanagementprozess verläuft also eng verflochten mit Entwicklungsprozess. Innerhalb der wie beschrieben etablierten lernförderlichen Rahmenbedingungen müssen Freiräume entstehen, die das Kontinuum aus der permanenten Auseinandersetzung mit dem Diskursgegenstand, der Einarbeitung in neue Sachverhalte und Tätigkeiten und der Selbstreflektion zur Bewusstmachung der Arbeit als Lern- bzw. Lehrprozess entstehen lassen.

3.3.2. Co-adaptive Entwicklungsprozesse

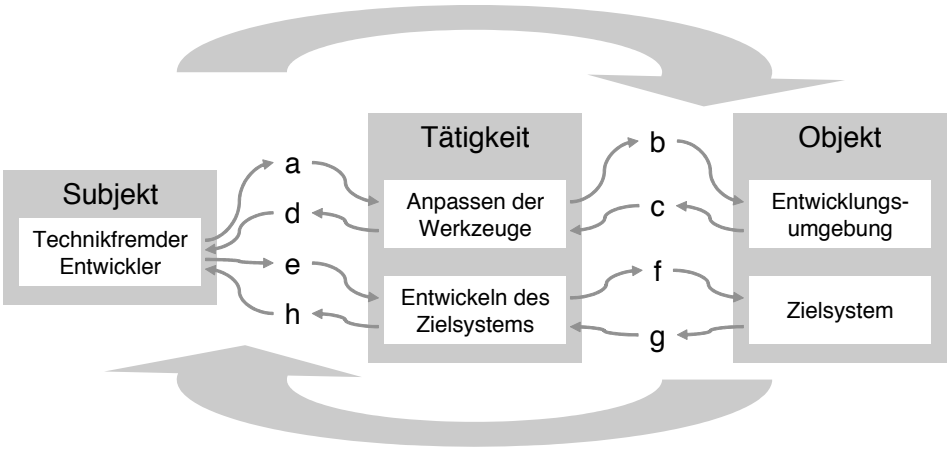
Eine wesentliche Inspiration für die Instrumentalisierung des Prinzips der systematischen wechselseitigen Annäherung zwischen den Subjekten und den Objekten innerhalb des Gestaltungsprozesses Software-Entwicklung bezieht der vorgestellte Ansatz aus den Konzepten des *interaktiven Designs*. So wie klassische Wanderschuhe zunächst in einem zum Teil schmerzhaften interaktiven Prozess eingetragen werden müssen [Wys97], um dann derart individualisiert umso bequemer zu sein, soll der Prozess der Software-Entwicklung so gesteuert werden, dass sich Bedürfnisse, Gewohnheiten und Fähigkeiten

der Nutzer schrittweise an die Potentiale der Software angleichen und umgekehrt. Idealerweise lässt sich der Prozess so gestalten, dass „Schmerzen“ weitgehend vermieden werden, etwa indem das gestaltet Material und die dazu genutzten Werkzeuge eine genügend hohe Flexibilität aufweisen, so dass die Rückwirkung auf den Gestaltenden möglichst gering im Umfang bzw. auf positive Effekte beschränkt bleiben.

Dass das verwendete Instrumentarium, das FLAME-Frameworks flexibel genug ist, um den Bedürfnissen und Eigenheiten der Nutzer so weit wie möglich entgegenzukommen, ist durch die Beschreibung der technologischen Grundlage weitgehend dokumentiert.

Orientierung an der gemeinsamen Tätigkeit

Das im Abschnitt 2.3.6 auf Seite 110 grundlegend für den Anwendungskontext adaptierte Schema 2.18 des Kontinuums der Wechselwirkung zwischen Subjekt und Objekt getrieben durch die Tätigkeit wird nun in Abbildung 3.25 weiter konkretisiert für den vorgestellten Entwicklungsprozess.



a) Auswählen, Konfigurieren, Arrangieren nach Anleitung, entsprechend des Bedarfs
b) Entstehen der neuen Entwicklungsumgebung, damit neuer Möglichkeiten und Bedienanforderungen
c) Rückwirkung der veränderten Entwicklungsmöglichkeiten auf die Arbeitsweise
d) Veränderte Arbeitsweise wird zu Erfahrung und in der Konsequenz zu neuen Planungen und Strategien bei der Entwicklung der Werkzeuge
e) Eigentliche inhaltliche Arbeit der Entwicklung auf der Grundlage der angepassten Entwicklungsumgebung
f) Entstehen des Zielsystems und Gewinnen von Erkenntnissen über die Wirkung der Werkzeuge
g) Rückwirkung der erzielten Effekte auf die Entwicklungsprozesse
h) Veränderte Arbeitsweise wird zu Erfahrung und in der Konsequenz zu neuen Planungen und Strategien bei der Anwendung und Weiterentwicklung der Werkzeuge

Abbildung 3.25.: Co-adaption durch Tätigkeit nach [FS99].

Die wechselseitige Annährung von Technologie und Anwendungsdomäne im Sinne von [KFK05] basiert auf der Ausrichtung der Tätigkeit der Entwicklung an den Tätigkeiten

der durch das Zielsystem zu unterstützenden Domäne. Auch wenn sich die grundlegenden Anwendungsfälle der Nutzung und der Ausgestaltung unterscheiden können, so ist doch evident, dass Begrifflichkeiten, Strukturen und Metaphern und die dadurch unterstützten mentalen Modelle und schließlich auch Tätigkeiten im Einzelnen gleich sind. Zumindest durch die Möglichkeit der unmittelbaren Wechsel zwischen der anwendungsnahen Gestaltung des Zielsystems und der evaluierenden Nutzung in der direkt partizipativen Entwicklung ist die ursprüngliche Dualität größtenteils aufgehoben. Durch diese Interaktivität wird die direkte Wechselwirkung als Grundlage eines evolutionären, sich weitgehend selbst tragenden bzw. treibenden Prozesses etabliert.

Den Ausgangspunkt bildet die initiale für den Kontext individualisierte Entwicklungsumgebung und damit auch ein erster Aufschlag für die Ausprägung des Frameworks zur Zielanwendung. Die Individualisierung besteht im Wesentlichen in der Ausrichtung des *Objekts* an den angestammten Tätigkeiten und Zielvorstellungen der Anwendungsdomäne – ermittelt in einer Phase der empirischen Erforschung. Im Grunde ist bereits der zwischengeschaltet Schritt der Arbeit mit Platzhaltern für die eigentlichen Objekte, die Arbeit mit Mock-Ups, im Sinne des Modells Tätigkeitsorient, insofern als dass in der simulierten Anwendung und Korrektur dieser Prototypen die Tätigkeit der Domäne die Entwicklungsarbeit bestimmt. Mit der Umsetzung der Ergebnisse in den evolutionären Prototypen auf der Grundlage des Frameworks und der Aufnahme der kooperativen interaktiven Entwicklungsarbeit beginnt der Ablauf der kontinuierlichen Anpassung einerseits der Werkzeuge und damit andererseits der Zielsoftware an die aus der Tätigkeit erwachsenden Bedarfe. Was wiederum mündet in veränderte technologische, organisatorische aber auch personale Rahmendingungen für die Tätigkeit, was schließlich die Akteure beeinflusst (deren Wissen um Gestaltungsspielräume, Kompetenzen und Bedürfnisse bez. der Werkzeuge, etc.).

Um einen selbst organisierten von der intrinsischen Motivation der Beteiligten getragenen Anpassungsprozess zu etablieren ist es entscheidend, dass die Tragweite der Wechselwirkungen begrenzt wird. Das heißt, die im nächsten Schritt nötigen Veränderungen an der Technologie sollten von einem Umfang sein, der eine möglichst unmittelbare (interaktive) Umsetzung zulässt. Umgekehrt sollte die Rückwirkung auf die Tätigkeit und den Akteur selbst immer einen überschaubaren konstruktiven Charakter wahren, der Effekte wie Überforderung ausschließt.

Ein Beispiel für einen solchen Zyklus ist etwa die Einführung und evolutionäre Überarbeitung einer Zoomfunktion für digitale Bilder. Es zeigt sich etwa während der Sichtung des Bildmaterials im Rahmen der Entwicklung bzw. der evaluierenden Anwendung, dass bestimmtes Bildmaterial (z.B. Faksimiles von Zeitungen) sowohl im Überblick als auch im Detail darstellbar sein muss. Das heißt bereits die Tätigkeit der Sichtung wurde beeinträchtigt. Es bedarf also der Anpassung des Werkzeugrahmens und der Korrespondierenden Funktionalität der Zielanwendung, um das Problem zu lösen. Die zunächst trivial erscheinende Zoomfunktion und die intuitiv damit verbundenen Bedienmuster und -elemente ist nicht zwangsläufig Teil des von den Fachexperten antizipierten Gestaltungsspielraums. Eine entsprechende Funktionalität ist als Teil des Frameworks

grundsätzlich angelegt und muss mit den betreffenden Inhalten lediglich assoziiert und konfiguriert werden.

So ist zum Beispiel denkbar, dass Autoren zunächst eine sehr einfache Möglichkeit erhalten, eine zweistufige Zoomfunktion mit fester Vergrößerung mit einem gewünschten Bild zu verbinden – etwa durch eine Auswahlbox innerhalb des Werkzeugs zum Zusammenstellen der Inhalte. Bilder, für die diese Option gewählt ist, werden dann in der Sicht der Endanwendung etwa mit einem Button ausgestattet (z.B. mit dem Icon einer Lupe versehen) die bei Betätigung je nach Zustand, das Bild vergrößert oder verkleinert darstellt. Erkennt der Autor das Potential dieser Funktion wird er die Auswahl seiner Materialien anpassen – mehr von denen Einbinden, die eine ähnliche Funktion benötigen. Er wird während dieser Tätigkeit möglicherweise dadurch an seine Grenzen stoßen, dass die feste Zoomweite und die Begrenzung der Anzahl der Schritte nicht für jeden der neu entdeckten Anwendungsfälle ausreichen – weil er etwa eine Anwendung des Zooms auf geografisches Kartenmaterial versucht hat. Er braucht nun die Möglichkeit den Zoom detaillierter, evtl. in Abhängigkeit von der Auflösung des Materials, zu definieren. In der Präsentation für die Nutzung werden nun Buttons jeweils für die Vergrößerung und die Verkleinerung benötigt. Die Anpassungen am Framework können ad hoc von technischen Entwicklern vorgenommen werden – bzw. es bedarf lediglich eines Hinweises durch einen der Coaches, um den Fachentwicklern die entsprechenden Schnittstellen zu den Metaobjekten zugänglich zu machen. Nötige Anpassungen an den Oberflächen der Werkzeuge werden entsprechend der zukünftig veränderten Tätigkeit des Inszenierens von Bildern so in enger Kooperation umgesetzt, dass der bis dahin etablierte und vertraute Arbeitsfluss der Autoren möglichst kaum verändert wird.

Die Erfahrung des Evaluationsprojekts zeigt, dass durch die Nutzung eines zunächst simplen Mechanismus die technischen Notwendigkeiten tatsächlich einfacher erfahrbar und einsichtig werden. So deutet sich während der Arbeit mit dem einfachen Zoom bereits an, welche grundsätzlichen Konsequenzen die Entscheidung für eine Zoom-Charakteristik hat; etwa dass ein eingebundenes Bild so vergrößert wird, dass es im Ganzen nicht mehr in den sichtbaren Bereich eine Panels passt. So entwickeln die nicht technisch orientierten Entwickler nach und nach ein Gefühl dafür, was auf sie zukommt, wenn sie einen Schritt weiter gehen und den Zoom mit mehreren Schritten oder gar stufenlos umsetzen möchten. So entwickeln Sie Verständnis für die Bedeutung aus ihrer Sicht neuer Interaktionsformen. Sie erschließen sich beispielsweise die Möglichkeit bzw. auch die Notwendigkeit Inhalte zu Scrollen, dies zu einem Teil der ihnen vertrauten Tätigkeiten zu machen.

Es ist die Aufgabe der Moderation des Entwicklungsprozesses bzw. der gegenseitigen Kontrolle im Rahmen der kollaborativen Entwicklung sicherzustellen, dass die fortschreitende Anpassung des Objekts ein sich nach und nach konsolidierendes Bezugssubjekt erhält. Der evolutionäre Prozess der wechselseitigen Anpassung terminiert also nur, wenn die fachlichen Entwickler in der Lage sind, die Weiterentwicklung individueller Vorstellung, die sich aus der alltäglichen Tätigkeit ableiten innerhalb des Bezugssystems globaler Ziele zu steuern. Das heißt die Entdeckungslust darf die Grenzen der vorher festzulegenden Ziele nicht überschreiten.

Hintergrund für den Erfolg des tätigkeitsorientierten Herangehens ist die Fokussierung auf die Arbeit mit bzw. an implizitem Wissen und echter Handlungskompetenz aller Beteiligten. Entsprechend der im Schema 2.17 auf Seite 106 wiedergegeben Wissensflüsse, wird durch die nach und nach routinisierte Arbeit auch mit technischen Konstrukten und Termini systematisch verinnerlicht. Durch das Nachahmen und Üben in der kooperativen Projektpraxis lassen sich auch Handgriffe (etwa zu Bedienmustern) vermitteln, die kaum verbalisierbar sind. Das gleiche gilt für den Kompetenztransfer aus der Fachdomäne in Richtung der technologischen Entwicklung. Für den hier adressierten Entwicklungsprozess mit einem hohen Maß an Unbestimmbarkeit bezüglich der benötigten Ergebnisse und infolge dessen des z.T. experimentellen, interaktiv, visuellen Charakters der Arbeit ist es gerade dieses prozedurale Wissen [Ker01] (Seite 161), welches benötigt wird.

Software als „gemeinsames Material“

Was [Pre99] (auf Seite 5) als *„Gestaltungserfahrung“* bezeichnet und über die reine Fachkenntnis der domänensachverständigen Zielanwender hinaus als tatsächliche Kompetenz während der Entwicklung notwendig ist, ergibt sich erst im effektiven Zusammenspiel aller Disziplinen. Das heißt, nicht die Einsicht allein, dass der Einsatz eines professionellen Screen-Designers zwingend notwendig ist, führt zur erfolgreichen Entwicklung interaktiver Systeme, sondern die systematische Harmonisierung der unterschiedlichen Dimensionen des Gestaltungsraums. Neben der Orientierung an der gemeinschaftlichen Tätigkeit bedarf es also eines „kleinsten gemeinsamen Nenners“, der das Verständnis der gemeinsamen Arbeit zusammenführt und vereinheitlicht. Diese gemeinsame Arbeitsgrundlage muss den durch die Intersubjektivität entstehenden Konsens ermöglichen und materialisieren. Sie entspricht als Kristallisationspunkt dem *Objekt* im oben beschriebenen Zyklus der Tätigkeitsorientierung.

Dazu wird das in Abschnitt 2.3.6 auf Seite 112 eingeführte, um das so genannte „gemeinsame Material“ erweiterte Kommunikationsmodell konkretisiert. Unter dem im Bereich der klassischen CSCW-Forschung in der Regel mit gemeinsamen bearbeiteten Dokumenten assoziierte Material, soll nun das zentrale Software-Artefakt verstanden werden.

Das Framework bietet durch seine Anlage als Verteiltes webbasiertes System die Möglichkeit, das Konzept des gemeinsamen Materials auch in geografisch verteilt arbeitenden Entwicklerteams zu etablieren. Die aufgabenspezifischen Clients – Sichten auf das gemeinsame Material – können jeweils über die beschriebene Web-Infrastruktur simultan am Artefakt Arbeiten. Die Einrichtung einer entsprechenden technischen Unterstützung des Kommunikationskanals ist eine separate Aufgabe, die sich am direkten, personalen Charakter des nötigen Wissensaustauschs orientieren sollte.

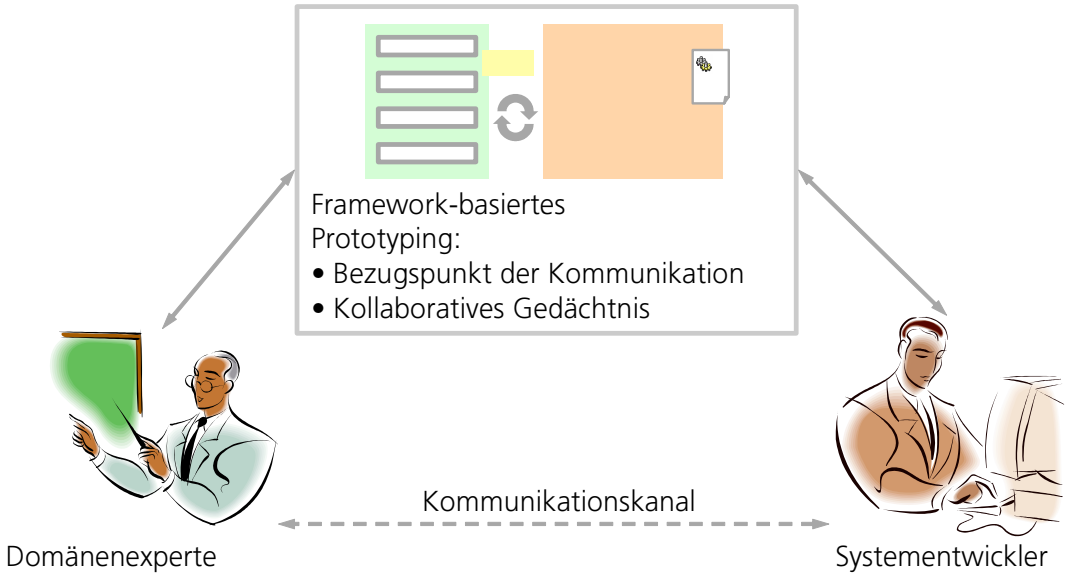


Abbildung 3.26.: Kommunikationsmodell der kooperativen Software-Entwicklung adaptiert nach [Sch90] und [Sch01].

Verwaltung des inhaltlich fachlichen Wissens

Neben der vorrangig angestrebten personenbezogenen Strategie des Wissensmanagements ist eine Sicherung konzeptioneller inhaltlicher Ergebnisse auch über externalisierende Mechanismen unabdingbar. Schließlich ist das prozedurale Wissen Ergebnis der Verknüpfung von explizitem Wissen (vgl. [Ker01] (Seite 161)).

Die Aufgabe der Externalisierung oder auch Kodifizierung erfüllt letztlich auch die Software in ihrer Rolle als gemeinsames Material. Vergleichbar mit der Forderung des XP, nach der allein gut lesbarer, kollaborativ verfasster Code und automatisierte Tests die Dokumentation ersetzen sollen, bildet das stetig wachsende Repositorium von das System definierenden Metaobjekten die externalisierte Repräsentation des eingeflossenen fachlichen Wissens. Die Übersichtlichkeit und Verständlichkeit wird dabei weniger durch Disziplin und unbedingten Mut zur Refaktorisierung erreicht, als vielmehr durch die fachlich individualisierten visuellen Entwicklungsschnittstellen und den Festelegungen die das Metaobjektprotokoll bezüglich der Ablage und Transformation der Eingaben macht. Das Framework etabliert so beispielsweise eine elementare Richtlinie zur darstellungsunabhängigen Strukturierung der Inhalte, die jeweils fachbezogen ausgeprägt werden – abstrakte Schnittstellen werden beispielsweise bezüglich der inhaltlichen Struktur und des Vokabulars wie beschrieben konkretisiert. Die Sichten auf das Artefakt haben jeweils deskriptiven Charakter. Die primären Ordnungsprinzipien zur Gliederung entsprechen weitgehend denen die die jeweilige Disziplin bevorzugt.

Die zur Fixierung von Systemeigenschaften von XP genutzten automatischen Tests fin-

den ihre Entsprechung in der direkten Rückkopplung der Entwicklungsbemühungen mit der Anwendung im (ggf. simulierten) Zielkontext. Das heißt, dass das Domänenwissen durch das permanente Einbetten der entwickelten unterstützenden Technologie permanent bewusst gemacht, also in gewisser Weise auch expliziert werden muss.

Ein wichtiges Vehikel zur Externalisierung von benötigtem explizitem Wissen entsprechend dem in 2.17 auf Seite 106 dargestellten „Wissensfluss“ ist die im Prozess als Wert institutionalisierte Kommunikation und Kollaboration. Der in der Regel informelle mündliche Austausch zu einzelnen Konzepten und insbesondere neuen Ideen (quasi als Einstieg in den Lebenszyklus des Wissens) kommt der in Abschnitt 2.3.6 auf Seite 108 analysierten Charakteristik der hier adressierten Projekte am besten entgegen.

4. Evaluation: Fallstudie „Geschichte und Geschehen IV“

Den Ausgangspunkt für die vorliegende Arbeit markiert das vom BmfBF geförderte Forschungsprojekt „LeMOLernen“. Dessen Ziel war es zum einen Vorgehensweisen und Instrumentarien zur Unterstützung der Verwertung hochwertiger medialer Inhalte im Bereich der schulischen Bildung zu entwickeln. Zum anderen sollten die entwickelten Konzepte in ihrer unmittelbaren Anwendung im Rahmen eines konkreten, in das Forschungsprojekt eingebetteten Projekts zur Erstellung einer interaktiven Software für das Fach Geschichte[1] erprobt werden. Es sei darauf hingewiesen, dass die damit verbundene angestrebte Entwicklung eines tatsächlichen Produktes für den Einsatz im Geschichtsunterricht keineswegs als sekundäres Ziel zu sehen war. Auch damit verknüpften sich – wie für derartige Projekte üblich – Forschungsaufgaben etwa im Bereich der Medien- und der Geschichtsdidaktik.

Die Tragfähigkeit der in dieser Arbeit vorgestellten Lösung soll also zentral anhand der Ergebnisse der das Forschungsprojekt begleitenden qualitativen Untersuchung eines exemplarischen Entwicklungsverlaufs belegt werden. Dazu beschreiben die folgenden Abschnitte in Form einer Fallstudie die in der teilnehmenden Betrachtung gewonnenen Erkenntnisse über die Einsatzfähigkeit des Frameworks und die Praktikabilität der Methodik zu dessen Anwendung im Software-Entwicklungsprojekt „Geschichte und Geschehen IV“. Dabei wird lediglich schlaglichtartig auf relevante Episoden der Entwicklung eingegangen. Die Projektergebnisse sind im dem Projektträger vorgelegten Abschlussbericht [CKV05] umfassend dokumentiert.

4.1. Die Spezifik des Projekts

In Abgleich mit der Charakteristik interaktiver, direkt reaktiver Systeme wird der Charakter des Projekts wiedergegeben und die Notwendigkeit spezifischer Lösungsansätze motiviert.

[1] „Geschichte und Geschehen IV“ als Produkt und Teil einer entsprechenden Reihe beim Ernst-Klett-Schulbuchverlag GmbH Leipzig

4.1.1. Historischer Hintergrund

Die Idee zum Projekt „LeMOLernen“ basiert auf dem Ergebnis einer vergleichbaren Entwicklungsbemühung. In Zusammenarbeit mit dem Haus der Geschichte in Bonn und dem Deutschen Historischen Museum in Berlin hatte das Fraunhofer ISST bereits eine Web-Anwendung entwickelt, die Teile der Sammlungen der jeweiligen Häuser in einem Virtuellen Museum präsentieren sollten. Es war eine der wichtigsten Sammlungen digitalisierten historischen Materials zur Deutschen Geschichte des zwanzigsten Jahrhunderts im Internet entstanden. Bild-, Ton- und Filmmaterial sind dort im Wesentlichen entlang der Zeitachse bzw. über drei Ebenen hierarchisch fachsystematisch gegliedert. Das Material ist historisch präzise, in für den interessierten Laien verständliche Beschreibungen und Kommentare eingebettet. Die aus technischer Sicht reine HTML-Lösung nutzt entsprechende Hyperlinks, um Querbezüge zwischen den Materialien zu schaffen.

Abbildung 4.1.: Ein Dokument der Lemo-Web-Seite

Wie die Abbildung 4.1 verdeutlicht, ist die Präsentation sehr Sachlich gehalten. Sie adressiert ein möglichst breites Publikum und dient im Wesentlichen der Information und Recherche. Eine Festlegung auf eine Zielgruppe wurde nicht dezidiert getroffen. Die Interaktivität beschränkt sich auf Funktionen der Navigation im Sinne von HTML.

Der Erfolg dieser Seite insbesondere auch als Fundus für die Ausgestaltung des Geschichtsunterrichts war schließlich der Anlass für die Überlegung, das System für den methodischen Einsatz im Unterricht fortzuschreiben. Neben der Anpassung der Ansprache der Texte und der Gestaltung für die nun sehr konkrete Zielgruppe von Schülern und Lehrern sollte insbesondere auch der Aspekte der historischen Arbeit an den Quellen in den Mittelpunkt der Unterstützung durch ein Software-Produkt rücken.

Es ergab sich also eine Kluft zwischen einem bestehenden System und der Idee für das Zielsystem. Die Inhalte waren durchgängig ungeeignet für den direkten Einsatz in einem kommerziellen Produkt für Schüler. Die Auflösungen von Filmen und Bildern waren zum Teil zu gering, die Texte bedurften einer komplett neuen Formulierung. Die Gliederungsstrukturen passten nicht zu den fachdidaktischen Konzepten und den Vorgaben durch Rahmenlehrpläne. Die Interaktivität zur Abbildung und Unterstützung historischer Methodenarbeit war nicht vereinbar mit der bestehenden Technologie. Als Konsequenz dessen ergab sich schließlich die Notwendigkeit, das angestrebte System komplett neu zu entwickeln. Lediglich Teile des Medienfundus fanden direkt Eingang in die Entwicklung. Das ursprüngliche LeMO-System war wiederum Fundus für die Recherche der Autoren. Es ist schließlich auch als exemplarischer Fundus, als Ausgangspunkt der Vermittlung der für die Recherche notwenigen historischen Methodenkompetenz, im System prominent angebunden.

4.1.2. Die Projektziele

Ziel des Projekts war die Konzeption, die Entwicklung und die Erprobung einer Software-Infrastruktur zur Unterstützung des Unterrichts im Fach Geschichte an allgemein bildenden Schulen. Das auf interessierte Laien zugeschnittene Vorgängersystem und die dementsprechend aufbereiteten Inhalte – allgemein in der Ansprache, mit nur geringen didaktischen Ansprüchen – sollte übertragen werden auf den neuen Einsatzkontext.

Damit sollte das erste Produkt, maßgeschneidert für den methodischen Einsatz im Geschichtsunterricht in 10. Klassen entstehen. Orientiert an den entsprechenden Rahmenlehrplänen sollte die umfangreiche Vermittlung historischer Fakten ergänzt werden um Lern- und Übungseinheiten zur Erarbeitung von Methoden der Erschließung historischen Quellenmaterials. Die entsprechenden Werkzeuge zur Unterstützung etwa der Filmanalyse, der Textinterpretation, der Bildanalyse etc. sollten dabei im Verhalten und im Erscheinungsbild möglichst den marktüblichen Anwendungen etwa zur Textbearbeitung oder zum Videoschnitt entsprechen.

Für gleiche Medientypen stehen immer die gleichen „standardisierten“ Mittel der Bearbeitung zur Verfügung. Lediglich im Sinne der Reduktion der Komplexität und der Wahrung der Übersichtlichkeit für eine fachlich fokussierte Lehre werden Adaptionen der gängigen Bedienmuster vorgenommen.

Auf diese Weise sollte es auch den Lehrerinnen und Lehrern erleichtert werden Anknüpfungspunkte für den Einsatz der Software bei ihrer individuellen Unterrichtskonzeption zu finden. Die Flexibilität der Architektur sollte nicht nur helfen, die Lehrer

dort abzuholen wo sie sich bzgl. ihrer eigenen Medienkompetenz befinden, sondern ihnen ferner die Möglichkeit eröffnen, die Software einfach an Ihre Bedürfnisse und die jeweilige Unterrichtssituation anzupassen.

Spielerisch explorative Elemente für die Motivation bzw. Aktivierung der Schüler zur Auseinandersetzung mit den fachlichen Inhalten sollten konzipiert und implementiert werden. Aus inhaltlicher Sicht wurde über die Ausgestaltung historischer Räume der Anschluss an das Vorsystem (LeMO) gesucht. Technologisch und optisch musste hierbei mit einem Innovationsschub auf die aktuellen Entwicklungen der Spieleindustrie und die veränderten Sehgewohnheiten der Konsumenten reagiert werden. Weitgehend realistische Inszenierungen der virtuellen Räume sollten einen Eindruck vom jeweiligen Zeitkolorit vermitteln und zum Erkunden historischer Objekte einladen.

In eben dieser Unschärfe wurden zunächst die Anforderungen an das System definiert. Zusammenhänge etwa zwischen der fachsystematischen Gliederung und der Struktur der Navigation oder der Tätigkeit der Quellenanalyse und adäquaten Steuerelementen der Interaktion sind schließlich Ergebnisse des diskursiven Prozesses der kooperativen Arbeit.

4.1.3. Die Projektstruktur

Die Zusammensetzung und die Zielstellung des Projekts sind sicherlich typisch für diese Art von Entwicklungsvorhaben. Als Besonderheit darf lediglich der explizite Forschungscharakter der gesamten Projektbemühung betrachtet werden. Der Einfluss dessen auf die Zielvorstellungen und Motive der einzelnen Beteiligten und damit auf den Ablauf des Projekts ist nur insofern von Bedeutung, als dass es den intellektuellen Stellenwert der von den einzelnen Beteiligten zu erbringenden Leistung charakterisiert. Der Anspruch des Projekts ist also in dieser Hinsicht für die hier betrachtete Klasse von Projekten relevant.

Über die benötigte Innovation hinaus, war es dennoch Ziel, ein tatsächliches Produkt zu erarbeiten. Die Anforderungen an die Qualität und die Effektivität der Entwicklung bildeten also quasi orthogonale Dimensionen zu den fachlichen, inhaltlichen sowie technologischen Ansprüchen.

Die Produktstruktur

Die Struktur des Produktes war von Beginn an eine große Unbekannte unter dem Einfluss der verschiedensten Argumentationslinien. Ausgehend von den Ambitionen der angestrebten informationstechnologischen Forschungsbemühung wurde zunächst ein System geplant, welches konstruktivistische und vor allem kollaborative Arbeitsformen im Unterricht unterstützen sollte. Ein telemedial zentrierter Ansatz sollte Lernende, Lehrende sowie Inhalteanbieter untereinander direkt von Punkt zu Punkt verbinden.

Darüber hinaus sollte eine am Fraunhofer ISST entwickelte Technologie zur halbautomatischen Systematisierung von Inhalten und Medien (entlang inhaltlicher, organisatorischer und didaktischer Struktur-Templates) zum Einsatz kommen (siehe dazu beispielsweise [KW02] bzw. [Cau00]). Damit sollte es möglich werden, den von den beteiligten Museen bereitgestellten enormen Fundus an Inhalten aufzubereiten und für die Verwendung im Unterreicht oder für das Lernen in der Freizeit zugänglich zu machen. Auf diese Weise sollten die fachlich inhaltlich sehr präzise ausgearbeiteten Texte und Mediensammlungen so mit Navigationsstrukturen und Interaktionselementen Verknüpft werden, dass ein didaktisch zielgerichtetes Arbeiten möglich wird – insbesondere um die Präsentation historischer Fakten zu bereichern um die Vermittlung von Methodenkompetenz der Arbeit an historischem Quellenmaterial.

Sehr bald wurde deutlich, dass diese Ideen sowohl an den Realitäten der üblichen Geschäftsmodelle und der Marktlage als auch an den Anforderungen des tatsächlichen Einsatzkontextes vorbei zielten. So stellte es sich als unmöglich heraus, ein reines Online-Produkt (etwa ein Web-Portal) zu entwickeln. Allein auf Grund der Tatsache, dass entsprechende Geschäftsmodelle für die Kommerzialisierung von schulischen Lehrinhalten noch kaum existieren, wurde es notwendig, sich zunächst an etablierten Distributionsformen zu orientieren. Eine DVD bzw. CD-ROM mit installierbarer Software würde sich beispielsweise auch in Ergänzung zum klassischen Lehrwerk (Buch) auf den vorhandenen Wegen der Schulbuchverlage vertreiben lassen. Über die Möglichkeit der Installation der Software und damit einer Übertragung von Programmteilen und Inhalten auf eine lokale Festplatte und der damit verbundenen kürzeren Zugriffszeiten hinaus, sollte es möglich sein, zumindest die Teile der Anwendung, die keiner Personalisierung und keiner persistenten Datenspeicherung bedürfen, ohne eine Installation, direkt vom Medium aus ablaufen zu lassen. Das entspricht zum einen dem klassischen Einsatzverständnis von Lehrmedien auf CD-ROM und erlaubt den Einsatz auch in Umgebungen, die eine Installation auf Grund administrativer Restriktionen oder wegen enger Ressourcengrenzen unmöglich machen.

Um nun zumindest beide Interessenströmungen zu vereinbaren, war es nahe liegend ein hybrides Produkt zu entwerfen. Das sollte einerseits die Möglichkeit bieten, eine lokal installierbare und auf gängigen Arbeitsplatzrechnern ablauffähige Software auf einem klassischen Medium auszuliefern. Andererseits sollte es die Distribution erlauben, die absehbare Entwicklung der Ansprüche bezüglich der Aktualisierbarkeit (über das Internet bzw. durch individuelle Inhalte der Anwender) oder in Bezug auf die Vernetzung der Nutzer in der Perspektive bereits durch die Anlage der technologischen Basis zu unterstützen. Nicht zuletzt war es von Begin an ein Ziel, eine nachhaltige, strategisch nutzbare Technologie zu entwickeln, die sich auf andere fachliche Kontexte übertragen lassen sollte und durch ihre grundsätzlich Webfähigkeit unabhängig von der Betriebssystemplattform des Anwenderrechners.

So entstand auf Grund der zunächst nicht endgültig fixierbaren Anforderungen bez. der Produktstruktur und der so bereits innerhalb des Projektes benötigten Flexibilität die offene Architektur des Systems, die die Grundlage des Frameworks bildet.

Die Stakeholder

Die Multidisziplinarität des Entwicklungskontexts macht die Zusammensetzung des Entwicklungsteams deutlich. Vordergründig sind dabei die funktionale und fachliche Heterogenität sowie die zahlenmäßige Dominanz der Gruppe der nicht technologisch orientierten Entwicklungsbeteiligten zu betrachten. So waren am Projekt allein neun Autoren beteiligt, die jeweils zuständig waren für einen inhaltlich abgeschlossenen Teil des Systems – also etwa für den Teil der sich mit einer bestimmten Epoche der Geschichte befasst. Schwerpunktmäßig haben diese Autoren einen Hintergrund als Historiker. In ihrer hauptberuflichen Tätigkeit arbeiten sie etwa als Hochschullehrer mit einer entsprechenden Neigung zur wissenschaftlichen Präzision und Abstraktheit. Bei anderen wiederum – etwa Museumspädagogen – konnte ein größerer Praxisbezug bezüglich des Medien-Einsatzes bei der Präsentation von Fachinhalten erwartet werden. Die Aufgabe der Autoren war es, das System fachlich und didaktisch zu konzipieren sowie bezüglich der inhaltlichen Ausgestaltung auch umzusetzen. Sehr deutlich war von Beginn an die Schwierigkeit, die Autoren dazu zu bewegen so strukturiert zu arbeiten, dass ein reproduzierbarer technologischer Prozess im klassischen Sinne die Ergebnisse hätte nahtlos übernehmen können. So scheiterte ein früher Versuch, Manuskripte zumindest in dem Sinne maschinenlesbar zu machen, dass so genannte in gängigen Textverarbeitungssystemen übliche Formatvorlagen die inhaltliche Struktur der Dokumente qualifiziert auszeichnen.

Ebenfalls nicht genuin technologisch ausgebildet – eher durch die Erstellung klassischer Printmedien vorgeprägt – sind die Vertreter des Verlages. Deren Aufgabe war es, die Arbeitsergebnisse der Autoren zusammenzufassen, die Qualität sicherzustellen und ggf. Ergänzungen vorzunehmen. Die beiden Redakteure zeichneten verantwortlich für die Entwicklung der Software als Produkt, passend zur Produktpalette und Strategie des Verlags. Außerdem übernahmen die Vertreter des Verlags auch die organisatorische und wirtschaftliche Administration des Projekts – steuerten den Ressourceneinsatz und die Beschaffung (etwa von Rechten an Medien und Fremdleistungen der Medienaufbereitung etc.). Die Erfahrung des Projekts zeigt, dass die Kompetenzen im Umgang mit der Informationstechnologie auf Anwendungsniveau im Vergleich zu denen der Autoren stark ausgeprägt sind. Gerade Redakteure haben genaue Vorstellungen von den Möglichkeiten gängiger Autorensysteme ohne diese jedoch selbst konstruktiv einsetzen zu können.

Das Bild aller Beteiligten ohne software-technologischen Hintergrund von Interaktiven Systemen ist geprägt von marktgängigen Produkten, die sich Hauptsächlich an der reinen Präsentation von Inhalten im Rahmen sehr linearer Gliederungs- und Navigationsstrukturen orientieren bzw. reinen zum Teil Aufwändig produzierten Animationen am anderen Ende des Spektrums. Echte Interaktivität, eine Abbildung der domäneninhärenten Tätigkeiten auf unterstützende Bedienmuster visueller Mensch-Maschine-Schnittstellen, ist noch kaum entwickelt und dementsprechend nicht bekannt. Das heißt die angestrebte fachbezogene Medienkompetenz, die es mit dem Zielsystem zu vermitteln gilt – die damit verbundenen Gestaltungs- und Effektivierungspotentiale – sind Forschungsgegenstände der Fachentwickler.

Auf Seiten der Technologie-Entwicklung waren je nach Grad der Auslastung bis zu sechs Informatiker beteiligt. Die Sondersituation, die sich aus der parallelen Entwicklung des Frameworks an sich ergab begründet diese Zahl im Wesentlichen. Am eigentlichen Prozess des interaktiven kooperativen Prototyping im Sinne der vorgestellten Lösung waren maximal drei Informatiker beteiligt.

Eine Sonderolle spielte der Entwurf der rein grafischen Gestaltung der Oberflächen. Diese wurde von extern beauftragten Firmen ausgeführt und so nur indirekt mit den Mitteln des Frameworks unterstützt. Die entworfenen grafischen Elmente wurden dabei zwar im Rahmen der mock-up-gestützten Diskussion kollaborativ weiterentwickelt, die Einführung in das Zielsystem über die entsprechenden Frameworkschnittstellen wurde aber schließlich, bis auf Feinjustierungen, von den System-Entwicklern vorgenommen.

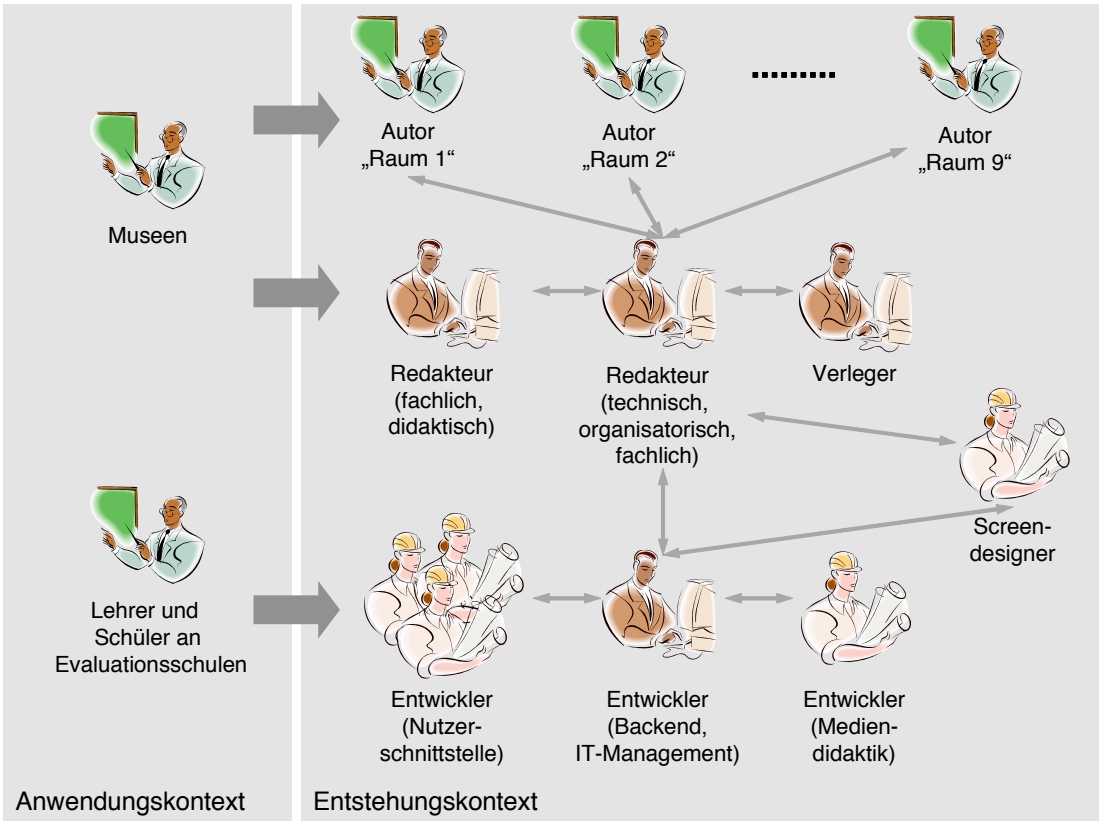


Abbildung 4.2.: Instanz des Schemas aus Abbildung 2.3 (Seite 30) für das Projekt LeMOLernen

Der transdisziplinäre Charakter des Projekts ergab sich schließlich aus der intensiven Beteiligung potentieller Anwender und durch den Einfluss des Nutzungskontexts über die Marktkenntnis der Verlage sowie die politischen Rahmenbedingungen in denen sich die Nutzung bewegen kann (Rahmenlehrpläne, technische Ausstattung der Schulen, etc.). Zwei der Redakteure füllten die Rolle der Nutzervertreter innerhalb des kooperativen

Prototyping vor ihrem Hintergrund als ausgebildete Lehrer für das Fach Geschichte aus. Ein Zyklus der evaluierenden Nutzung (nach dem Verständnis der Vorgehensweise) wurde darüber hinaus an vier Schulen direkt im Einsatzkontext des Geschichtsunterrichts durchlaufen. Sowohl Lehrer als auch Schüler nahmen mit ihrem in Fragebögen strukturierten Feedback direkt Einfluss auf den Entwicklungsprozess.

Zunächst lediglich antizipiert ist der Bedarf nach der direkten Einflussnahmemöglichkeit auf die Software durch die Nutzer selbst. In Diskussionen auf Messen und Tagungen wurde deutlich, dass Anpassungen sowohl der Inhalte als auch der Funktionalitäten zur Manipulation und zur Navigation für spezifische Unterrichtssituationen als äußerst wünschenswert – oder gar notwendig betrachtet werden. Reaktionen auf die Demonstration der dafür angelegten Schnittstellen lassen erkennen, dass die für den Fachkontext individualisierten Schnittstellen über den Entwicklungskontext hinaus Akzeptanz finden. Die dem fachlichen Nutzer fehlende Handlungskompetenz zur routinierten Nutzung der Werkzeuge, die sich die an der Entwicklung Beteiligten im Rahmen des co-adaptiven Prozesses angeeignet haben, wird dabei nicht als Hemmnis erkennbar.

4.2. Anwendung und Evaluation der Lösungsansätze

4.2.1. Einsatz des Frameworks

Welche Rolle das Framework als technologisches Instrument bei der Umsetzung der Software „Geschichte und Geschehen IV“ gespielt hat, wird in den folgenden Abschnitten beschrieben. Es wird erläutert, wie sich die Ausprägung des zunächst abstrakten Software-Rahmens zu einer kontextspezifischen Entwicklungsgrundlage für die Arbeit der fachlich orientierten Systemausgestaltung konkret gestaltet hat. Dazu wird exemplarisch gezeigt, welche Komponenten und Konzepte auf welche Weise zu domänen- und aufgabenspezifischen Bausteinen für den Einsatz bei der Entwicklung der Geschichtssoftware wurden und diese schließlich in das eigentliche Zielsystem mündeten. Die Schnittsellen, die die aufgaben- und fachorientierte Mitwirkung an der Entwicklung unterstützten werden vorgestellt.

Die Auswahl und Auslegung der Architektur

Die Produktstruktur erfordert den Einsatz zweier der möglichen Architekturvarianten (siehe dazu die Abbildung 3.12 auf Seite 157). Um personalisierte Daten der Nutzer persistent speichern und die in der Anwendungsschicht realisierten Komponenten nutzen zu können, bedarf es eines *lokalen, mehrschichtigen* Aufbaus. Nur durch den Einsatz der Micro-Servlet-Architektur wird es möglich das XML-basierte Backend der Anwendung im schreibenden und im lesenden Zugriff zu nutzen. Auch wenn sich die dafür notwendigen Komponenten theoretisch direkt vom sekundären Datenträger (DVD) laden ließen und lediglich die abzulegenden Nutzerdaten auf eine lokalen beschreibbaren Festplatte

abgelegt werden müssten, erfolgt aus Performance-Gründen eine Installation aller Programmteile. Lediglich die Option umfangreiche Medienelemente (z.B. das Filmmaterial) auf der DVD zu belassen und bei jedem Zugriff von dort zu laden, wird während des Installationsvorgangs angeboten.

Die Mehrschichtige Architektur ist für den Nutzer transparent. Der Programmstart erfolgt aus seiner Sicht z.B. in der für Windows-Programme üblichen Weise, durch den Klick auf das Programmsymbol im Startmenü. Das führt zum Start der aktiven Komponente des Micro-Servers und anschließend zum Ausführung der Client-Anwendung sowie der Kontaktaufnahme zwischen beiden Komponenten. Das geschieht innerhalb des Adressraumes der Nutzer-Session des Betriebssystems. Diese Umsetzung findet hier Erwähnung, da sie auch einer einsatzspezifischen Ausprägung entspricht. So sind mit Hilfe des Frameworks beispielsweise auch Szenarien zu unterstützen, die den Serverprozess als Dämon (oder Dienst) mit dem System starten und permanent und für verschiedene Client-Anwendungen zur Verfügung stellen.

Eine zweite Variante des Systemaufbaus wird gewählt, wenn der Nutzer sich für eine Nutzung der Software, ohne jegliche Installation von Systemteilen, also ohne jeden Schreibzugriff durch die Software auf eine lokale Festplatte entscheidet. Entsprechend der Zusammenstellung der möglichen Architekturvarianten in Abbildung 3.12 kommt in diesem Fall der Aufbau *Lokal, „Flach“* zum tragen. Dabei liest und interpretiert der Client die Meta-Objektbeschreibungen im direkten Zugriff auf Dateien mit den XML-Serialisierungen. Das heißt auch, dass der Funktionsumfang des Zielsystems auf die Möglichkeiten der Anzeige der Inhalte und die elementarsten Interaktionselemente der grafischen Nutzeroberfläche reduziert ist. So ist es etwa auch nicht mehr möglich die Volltextsuche zu nutzen oder eine Authentifizierung des Nutzers vorzunehmen. In vollem Umfang möglich bleiben die elementaren Mechanismen der Navigation.

Die Konfiguration und damit die endgültige Ausprägung des Frameworks bezüglich der Architektur des Zielsystems nimmt der Nutzer quasi dynamisch zur Installationszeit vor. Die Installationsroutine – im konkreten Falle des hier betrachteten Projekts ein plattformabhängiger, skriptgesteuerter Installer – macht die entsprechenden Eintragungen in den Meta-Objektbeschreibungen (also der Konfiguration des Gesamtsystems). Der in Abbildung A.4 im Anhang angegebene Auszug aus einer solchen Konfiguration beschreibt den konkreten Aufbau des Zielsystems von „Geschichte und Geschehen IV“ als lokale mehrschichtige Installation. Die Verweise auf den lokalen Web-Service (`http://127.0.0.1 ...`) wären in der Ausprägung für die Variante *Lokal, „Flach“* nicht vorhanden bzw. fest durch relative Pfade innerhalb der Verzeichnisstruktur des Mediums ersetzt.

Der interaktive Prototyp wurde initial als die mehrschichtige Architektur ausgeprägt, die nun auch dem Nutzer (in der ersten Variante) zur Verfügung steht. Das heißt, die Entwickler-Clients und insbesondere die Redaktionsschnittstelle, die das visuelle Arbeiten (WYSIWYG) ermöglicht, wurden in der Gleichen Konstellation betrieben, wie schließlich der Nutzer-Client. Das bedeutet, dass auch die Zielarchitektur während der Nutzung in der Produktion kontinuierlich evaluiert und konsolidiert wurde. Das machte

unter anderem frühzeitig deutlich, dass in dem konkreten Fall eine physische Kollaboration auf einem singulären Online-Repository mit allen Meta-Level-Objekten aufgrund der Teamkonstellation und der Ausstattung mit Netzzugängen kaum praktikabel sein würde. Es wurde also in der Ausprägung der lokalen mehrschichtige Architektur die Lösung gefunden, die es den Fachentwicklern erlaubte offline und zumindest logisch kollaborativ am gemeinsamen Artefakt zu arbeiten, indem die unterschiedlichen Repositories aus Entwicklersicht transparent synchronisiert wurden. Die Schwierigkeiten mit einem echten Netzbetrieb des Systems, die sich in der Antizipation der Einsatzmöglichkeiten beim Nutzer bereits angedeutet hatten, wurden also schon während der Entwicklung manifest. Das verdeutlicht in gewisser Weise die Kongruenz zwischen dem Anwendungskontext und dem fachbezogenen Entwicklungskontext.

Die Basisinteraktionselemente

Zur Unterstützung der Rezeption historischer Quellen und fachlicher Fakten dominiert natürlich der Einsatz der üblichen Werkzeuge. Konsolen zur Steuerung des Ablaufs von Filmen und Audio-Elementen beispielsweise müssen dazu primär entsprechende Erwartungen der Nutzer erfüllen. Dementsprechend konnte das Projekt bezüglicher solcher Funktionalitäten weitgehend aus dem Repertoire des Frameworks schöpfen. So konnte bei der initialen Ausprägung der Prototyping-Umgebung etwa zunächst eine feste Zuordnung zwischen Filmmaterial und den Standard-Player-Komponenten vorgenommen werden. So dass das Einbinden von Medien durch die Autoren immer direkt und ohne zusätzlichen Aufwand die Einbettung in die Software-Komponente implizierte. Lediglich kleinere Anpassungen bezüglich des Layouts und des Oberflächen-Designs im Sinne der Ausprägung des Screen-Design-Rahmens wurden nötig. Das Ergebnis der Konkretisierung der allgemeinen Komponente ist in Abbildung 4.3 am Beispiel eines Video-Materialfensters gezeigt. Mit dem Bildschirmauszug B.2 im Anhang ist zum Vergleich eine Variante des Players zur Wiedergabe von Audioinhalten abgebildet.

Der im Produktionsprozess ursprünglich vorgesehene Ablauf erforderte von den Autoren lediglich die Zusammenstellung der Inhalte. Also das Einfügen eine Audiodokuments in die Gliederung mit Hilfe des entsprechenden Werkzeugs (weiter unten beschrieben). Dies erzeugt entsprechend der während der initialen Ausprägung getroffenen Festlegungen des Meta-Objektprotokolls dynamisch zur Laufzeit des Zielsystems konkrete Basis-Objekte zur Darstellung des Players in einem dezidierten Materialfenster. Ebenfalls eine projektspezifische Konkretisierung ist dabei das Layout, bei dem sich ein erläuternder Text direkt unter den Player gliedert und das umgebende Fenster sich immer an das Format und die Dimensionen des Wiedergegebenen anpasst. Eine entsprechende Standard-Generierung existiert für alle verwendeten Basis-Medientypen. In Bild B.2 ist ein z.B. Textmaterialfenster verwendet, um das Transkript des im Tondokument gesprochenen Textes modal anzeigen zu können. Ein Fenster zur Präsentation von Bildmaterialien zeigt die Abbildung B.1.

Im Beispiel B.2 ist dieses feste Schema jedoch exemplarisch durchbrochen. Dabei wird deutlich, wie Autoren mit diesem projektspezifischen Aggregat eines Interaktionsbau-

Abbildung 4.3.: Der Video Player in „Geschichte und Geschehen IV“

steins disponieren können. Es kann als Ergebnis des wechselseitigen Anpassungsprozesses – insbesondere auch in Bezug auf die Kompetenzen der Historiker in Bezug auf die technische Entwicklung – betrachtet werden, dass einer der Autoren in der Lage war mit den einfachen, ihm zur Verfügung stehenden Mitteln, eine neue Variante des Standard-Audio-Fensters zu entwickeln. Diese simple Lösung erlaubt es Inhalte zu aggregieren und im direkten Vergleich zu präsentieren.

Tatsächliche funktionale Erweiterungen der Framework-Komponenten für den konkreten Einsatz sind beispielsweise die optionale Verbindung der Player-Komponente mit Medienelementen, die zusätzliche Informationen enthalten, wie etwa die bereits angedeuteten Transkipte von gesprochenem Text oder die Ergänzung der Funktion („2×“-Button), die ein Video auf das Doppelte vergrößert anzeigt. Diese nutzen ausschließlich vorhandene Framework-Basis-Funktionalität, indem sie erneut ein Panel in einem Fenster zu Anzeige bringen. Streng genommen entspricht dies Schritten der Navigation, wie sie im nächsten Abschnitt erläutert werden. Aus Nutzersicht handelt es sich dabei um Interaktionen, die die Rezeption des Materials flankieren und unterstützen. Dazu wurden bei der initialen Framework-Ausprägung lediglich Templates und Stilvorlagen für Metaobjekte angelegt bzw. während der Befüllung mit Inhalten und der damit verbundenen evaluierenden Nutzung evolutionär verfeinert. So war etwa die verwendete Symbolik und die Positionierung der Schaltflächen oder auch das Verhalten des sich öffnenden Fensters Gegenstand umfangreicher Diskussionen und Experimente.

Eine typische, projektabhängige Erweiterung der Player-Funktion ist auch die Ergänzung der Start- und Stopp-Marken, die der Nutzer setzen kann, um den Start- und Endzeitpunkt für das Abspielen eines Mediums festzulegen. Diese, insbesondere im Kontext der

so genannten interaktiven Arbeitsblätter eingesetzte Systematik, soll es dem Rezipienten beispielsweise erlauben, die Fokussierung auf aufgabenbezogene Teile des Materials technisch unterstützen zu lassen. So kann etwa auf ganz simple Weise bereits verdeutlich werden, wie man technische Mittel nutzen kann, um Wichtiges von Unwichtigem zu trennen und neue Perspektiven zu erzeugen bzw. wie es umgekehrt möglich ist, diese Mittel zu nutzen, um Medien zu manipulieren und zu verfälschen. Eine in der Erfahrung mit der produktiven respektive evaluierenden Arbeit für die Zukunft antizipierte wünschenswerte Erweiterung dessen ist die Einführung beliebig vieler solcher Schaltmarken. Dies würde einer simplen Einrichtung für den Schnitt von Ton- und Videomaterial entsprechen, die nur aus der evolutionären Entwicklung heraus, über die zunächst einfache technische Realisierung als Gestaltungsoption von den nichttechnischen Entwicklern antizipiert werden konnte und nun auch bezüglich der Nutzung beherrschbar scheint. Im Unterschied zur Ergänzung der Buttons, wie im Absatz oben beschrieben, müssen diese Anpassungen an den entsprechenden Schnittstellen der Framework-Komponenten auf Whitebox-Ebene durchgeführt werden.

Beide Varianten stehen den Autoren jedoch unter einer für sie adaptierten Sicht zur Manipulation zur Verfügung und sind durch entsprechende Konstrukte der Meta-Level-Beschreibung repräsentiert: das Zoomfenster etwa wie beschrieben durch den Einsatz eines eigenen frei disponiblen Elementes als fester Bestandteil der Stilvorlage für Film-Materialien, die aus Sicht des Autors mit einem einfachen „binären“ Schalter gewählt werden kann. Die Start- und Stopp-Marken werden dagegen als echte Erweiterung des Player-Elements auch über die erweiterte Konfiguration von dessen Metalevel-Repräsentation gesteuert. Im Anhang (Abschnitt B.3 auf Seite 287) ist eine Möglichkeit illustriert und kommentiert, wie sich dieses Vorgehen aus Sicht des Redakteurs oder Autors gestalten kann.

Eine weitere dezidiert für den Einsatz im Fach Geschichte vorgenommene Ausprägungen von Interaktionselementen betrifft die Arbeit mit Standbildauszügen von Filmen. Ein entsprechendes Beispiel wurde bereits zur Illustration der Konzeption diskutiert (siehe dazu Abbildung 3.5 auf Seite 133). Der Druck auf das mit dem Kamera-Symbol gekennzeichnete Schaltelement in der Player-Bedienleiste führt dabei zur Erzeugung eines neuen Bildschirmelements mit dem Standbild des Filmes, aufgenommen zum Zeitpunkt des Klicks. Dabei handelt es sich wiederum um ein disponibles Element im Sinne des Frameworks. Es kann also in Kombination mit beliebigen anderen Interaktionskomponenten genutzt werden, um verschiedene Arbeitsabläufe zu unterstützen. Im Beispiel etwa ist das extrahierte Bild für den Nutzer innerhalb des Panels verschiebbar (*dragable*) und besitzt die Fähigkeit innerhalb eines Zielbereichs im Sinne von *Drag & Drop* einzurasten. Im Beispiel wird dies für eine Interpretationsübung genutzt. So kann der Schüler das historische Filmmaterial methodisch systematisch analysieren, indem er drei Schlüsselbilder extrahiert und diese beschreibt, damit ein kleines Storyboard zum Plot des Filmes erzeugt. Neben der wirklich inhaltlichen Auseinandersetzung mit dem historischen Stoff erarbeitet er sich damit auch die benötigte Medien- bzw. Methodenkompetenz. Die Interaktionen die diesen Ablauf tatsächlich unterstützen sind ebenfalls das Ergebnis von Experimenten, die die Autoren zunächst mit einfachen prototypischen Mitteln durchführen konnten. Die Umsetzung in Elemente, die für die Autoren einfach

zu handhaben waren wiederum, eröffnete die Gestaltungsspielräume, die schließlich eine ganze Kategorie mediendidaktischer Entwürfe hervorbrachte.

Charakteristisch für die Beispiele für projektbezogene Konkretisierungen von Interaktionselementen des Frameworks ist, dass sie zwar zunächst speziell für den Einsatz bei „Geschichte und Geschehen“ entwickelt wurden, aber schließlich doch in den Fundus an Komponenten eingehen werden, die auch übertragen auf ähnliche Fachkontexte eine erneute Verwendung finden können.

Weitgehend unverändert wurden die im Framework angelegten Funktionalitäten zur Arbeit an Textmaterial genutzt. Die Textanalyse und -interpretation als eines der wichtigsten Mittel Historischer Arbeit (die meisten Quellen bestehen aus Texten) wird unterstützt durch typische Funktionen es Rich-Text-Editors. Damit es etwa auf der eine Seite den Autoren einfach möglich den Text mit den üblichen Effekten visuell zu inszenieren bzw. den Redakteuren, ihn zu überarbeiten (siehe Abbildung 4.12 auf Seite 260). Auf der anderen Seite kann auch die eigentliche methodische Arbeit der Schüler im Rahmen der Übungen damit unterstützt werden. Durch das ändern der Farbe der Schriftart beispielsweise können Auszeichnungen am Text vorgenommen werden, um etwa Schlüsselbegriffe zu markieren. In Kombination mit der Arbeit in mehreren frei disponiblen Textboxen und freien Formen können eigene Text verfasst, gestaltet und einfache Schemata entwickelt und präsentiert werden. So können auch Beschriftungen an beliebiges Material angebracht werden. Die Nutzung gängiger Paradigmen bei der Arbeit mit digitalem Text erfüllt darüber hinaus bewusst das Kriterium, auch Standard-Interaktionsparadigmen in die fachliche Arbeit einzuführen.

Insbesondere die Kombination aller elementaren Bedienelemente wird im Rahmen der Erstellung der interaktiven Übungen intensiv genutzt um die Tätigkeiten der Arbeit an historischen Quellen zu vermitteln und zu unterstützen oder aber Fakten ggf. auch spielerisch zu vermitteln. Der Grad an Vorgaben, die durch die Autoren gemacht werden und damit der methodischen Führung, die die Interaktionen der Nutzer schließlich bestimmen, ist durch das Rechtesystem technisch steuerbar. Die dadurch ermöglichte Fokussierung auf bestimmte Tätigkeiten, ist das entscheidende Vehikel zur Etablierung der Aufgabenbezogenheit der technischen Unterstützung sowohl bei der Arbeit der Nutzer, als auch bereits während der Erstellung durch die Autoren.

Abbildung 4.4 gibt ein Beispiel für ein Entwicklungsergebnis eines der Autoren, das ausschließlich im WYSIWYG-Modus der Bearbeitung des Panels eines digitalen Arbeitsblattes entstanden ist. Die Präsentation des Fensters, die Toolbar und der Bereich für die Anzeige der Aufgabenstellung sind in einem Template fixiert und bilden den Rahmen eines jeden Arbeitsblattes. Dieses Konzept des Arbeitsblattes mit dieser Struktur und seiner Einbindung in einen Aufgabenkomplex (die „Karteireiter“ am oberen linken Bildrand führen durch eine Folge von bis zu fünf didaktisch aufeinander aufbauenden Aufgaben) entspricht einer diskursiv, evolutionär erarbeiteten Festlegung und einer projektspezifischen Konkretisierung des Funktionsrahmens. Die grundsätzlich Anlage eines Arbeitsblattkomplexes als Aggregat innerhalb der Navigationsstruktur, ist damit auch bereits während der konzeptionellen Arbeit der Autoren mit dem entsprechenden

aufgabenbezogenen Werkzeug möglich (siehe dazu 4.10 auf Seite 256 – die Elemente „AufgabenBer.“ bzw. „Aufgabe“).

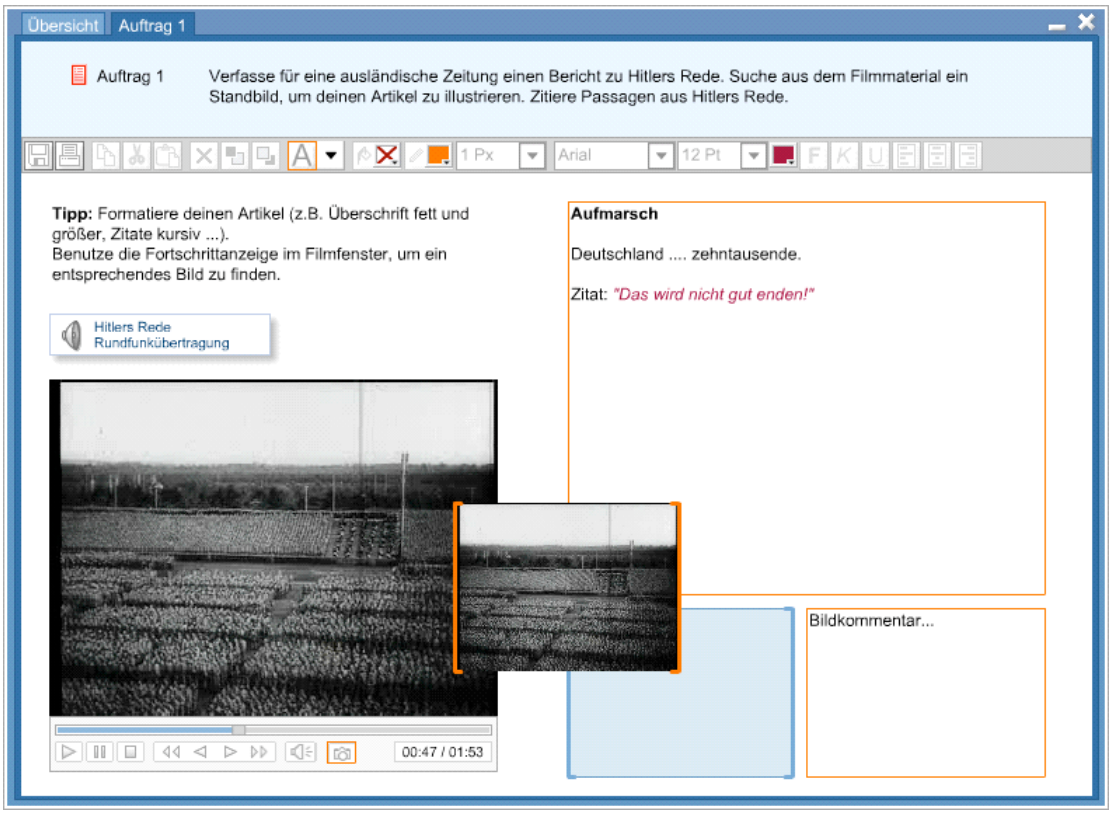


Abbildung 4.4.: Komposition interaktiver Elemente zum virtuellen Arbeitsblatt

Wie die im Framework angelegten Interaktionelemente (siehe dazu 3.9 auf Seite 149) im Rahmen einer solchen Teilanwendung – als die ein digitales Arbeitsblatt betrachtet werden kann – konkretisiert und genutzt werden, wird im folgenden anhand des Beispiels der Abbildung erläutert.

Das Einfügen eines Arbeitsblattes mit Hilfe des Gliederungseditors impliziert wie angedeutet die Anlage der kompletten Meta-Level-Objektstruktur, die zur Generierung der eigentlichen Objekte des Fensters, der Toolbar und des zunächst leeren Panels des Arbeitsblattes führt. Stil und Konfiguration folgen der Vorgabe des Templates und steuern etwa die Dimensionen des umgebenden Fensters und das Verhalten beim Schließen. Letzteres nimmt im Vergleich zu anderen modalen Fenstern der Anwendung eine Sonderrolle ein. Da ausschließlich im Bereich des Arbeitsblattes Eingaben des Nutzers personalisiert gespeichert werden können, ist der Funktions-Rahmen hier pauschal so ausgeprägt, dass das Schließen eines Arbeitsblattes mit ungesicherten Veränderungen durch den Nutzer in einem Dialog quittiert werden muss.

Die Rich-Text-Komponente wird im Beispiel der Abbildung 4.4 in zwei unterschiedlichen

Modi, über den Stil gesteuert, genutzt: der angegeben „Tipp“ ist ein fixierter Text, den der End-Nutzer (im Rezeptionsmodus) weder verschieben noch inhaltlich oder bezüglich der Formatierung verändern kann. Lediglich im WYSIWYG-Modus der Entwicklungsumgebung stehen etwa dem Redakteur die vollen Funktionalitäten der Bearbeitung zur Verfügung. Ihm wäre es beispielsweise neben der Anpassung des Textes möglich, über Anpassungen des Stils, die Farbe und Dicke des Rahmens sowie die Hintergrundfarbe des Textfeldes zu bearbeiten und die entsprechenden Rechte an den Nutzer zu übertragen. Dies wird in der zweiten Verwendung eines Textfeldes deutlich (den Orange gerahmten Eingabefeldern): die gleiche Komponente wie im ersten Fall wird hier so gesteuert, dass der Nutzer, den Text inhaltlich und bezüglich der Form bearbeiten kann. Die entsprechenden Auswahltasten für Werkzeuge in der Toolbar werden ebenfalls orange hervorgehoben, wenn eines dieser Elemente in den Fokus des Nutzers rückt (Auswahl mit der Maus). Dem Nutzer ist jedoch explizit das Recht entzogen, die Position und die Gestaltung der Textbox zu verändern und zu speichern.

Ein Grafik-Element kommt hier zum Einsatz, um ergänzendes Quellenmaterial – über den primär zu betrachtenden Filmausschnitt hinaus – zugänglich zu machen. Dazu wurde eine Basisfunktionalität des Frameworks ausgenutzt, die es erlaubt jedes Element mit einem Verweis auf den Aufruf einer Funktion des funktionalen Rahmens zu versehen, oder ein Meta-Level-Objekt zu adressieren. Letzteres wird hier genutzt um, ein Materialfenster im kanonischen Sinne der Anwendung zu öffnen. Über die Meta-Objektbeschreibung dieses Grafik-Elements ist etwa sein Schatten realisiert. Dieses Element ist für den Nutzer nicht disponibel. Für ihn erfüllt es lediglich den Zweck einer Schaltfläche zur Aktivierung einer Funktion.

Die Video-Komponente ist in der beschriebenen Weise in Kombination mit den Drag&-Drop-Mechanismen genutzt. Anwendungsweit wirksame Konkretisierungen der entsprechenden Framework-Komponenten realisieren dabei etwa die Markierungen der Elemente durch die Gestaltung der Rahmen. So ist über zum Teil in Templates materialisierte Konventionen umgesetzt, dass alle durch den Schüler bzw. Nutzer aktiv zu bearbeitenden Elmente orange umrahmt werden. Die dickere Darstellung der Seiten von an Drag&Drop-Interaktionen beteiligten Elementen ist eine Festlegung für alle Interaktionselemente dieser Kategorie. Diese Anpassung ist jedoch über White-Box-Schnittstellen zur Compile-Zeit der Entwicklungsumgebung vorgenommen.

Ergänzend ist im Anhang in Abbildung B.4 ein Arbeitsblatt gezeigt, was einen eher spielerischen Zugang zur Wissensvermittlung und Vertiefung wählt. In diesem Beispiel wird auch ein weiteres Feature der Drag&Drop-Komponente in der Anwendung im Projekt anschaulich: es ist möglich, die Zuordnung zwischen einem disponiblen Objekt und der Zielzone inhaltlich vorab festzulegen. Sodass neben dem Effekt des Einrastens und damit der Determinierung des Layouts auch eine automatische inhaltliche Bewertungen der Zuordnung (richtig, falsch) vorgenommen werden kann, was zumindest ein simples Feedback erlaubt.

Die so genannten Freeform-Komponenten schließlich kommen im System beispielsweise zum Einsatz, um mit Hilfe von Sprechblasen andere Medienelemente, wie Schaubilder

zu kommentieren bzw. zu annotieren. Mit Hilfe von Blockpfeilen, Ellipsen etc. in Kombination mit Beschriftungen, Linien und Pfeilverbindungen lassen sich ferner einfache Schemata zeichnen. So kann beispielsweise der Aufbau von Verfassungen illustriert oder zum Gegenstand von Aufgaben gemacht werden

Das wesentliche kreative Potential bei der fachzentrierten Entwicklung von Funktionalitäten der Interaktion besteht in der den Fachleuten eingeräumten Fähigkeit zur Komposition und kollaborativen Konkretisierung der disponiblen Elemente. Wobei gerade das Einräumen dieser Fähigkeiten und das senken von Hemmschwellen systematisch betrieben werden muss. Die innerhalb des Projekts dazu eingesetzten spezialisierten Entwicklerschnittsellen bzw. die gezielte Anwendung des methodischen Rahmens wird weiter unten in diesem Kapitel noch beleuchtet.

Die Navigation

Die Ergebnisse der Ausprägung des Navigationsrahmens (siehe Abschnitt 3.1.2 auf Seite 137) für die Geschichtssoftware werden im Folgenden schlaglichtartig präsentiert. Der in Abbildung 3.7 auf Seite 139 schematisch dargestellte Rahmen der Hauptanwendung ist für „Geschichte und Geschehen IV“ ausgeprägt wie in Abbildung 4.5 exemplarisch gezeigt.

Es sind grundsätzlich vier unterschiedliche Typen von Knoten innerhalb der Navigationsstruktur ausgeprägt, die die fachliche Entwicklung zur Gliederung des Systems verwenden kann. Diese Knoten-Typen ähneln sich im Aufbau und der Funktion, spiegeln aber unterschiedliche Gliederungsebenen in der hierarchisch aufgebauten, fachsystematischen, Struktur der Inhalte wider. Auf der Obersten Ebene bietet ein so genannter virtueller Raum einen spielerischen Zugang zu einer Epoche. Das interaktive virtuell „begehbare“ Panorama gibt das Zeitkolorit des entsprechenden historischen Abschnitts wieder und fasst untergeordnete Themenebenen repräsentiert durch Einrichtungsgegenstände zusammen. Ein Mausklick auf diese Objekte innerhalb des Raumes entspricht einem kontextabhängigen Navigationsschritt (in Abbildung 3.7 auf Seite 139 als Kontextnavigation bezeichnet).

Die nächste untergeordnete Gliederungs- und damit auch Navigationsebene bildet das *Schwerpunktthema*. Dieses setzt wie der Name sagt einen inhaltlichen Schwerpunkt auf einen historischen Aspekt, der so komplex ist, dass ein direkter Einstieg in die detaillierte inhaltliche Auseinandersetzung nicht möglich ist. Der entsprechende Screen hat damit auch lediglich orientierenden Charakter, der die Bedeutung der Thematik betont und auf die eigentlichen Ebenen der detaillierten Präsentation interaktiver Inhalte verweist und diese entsprechend der didaktischen Intention ordnet und gliedert – etwa im Sinne eines „Lernwegs“.

Die Präsentation tatsächlicher primärer Inhalte (im Gegensatz zu sekundären orientierenden bzw. unterhaltenden) beschränkt sich also auf die Ebenen so genannter *Themen* und *Module*. Diese sind jeweils auch bildfüllende Panele – Knoten der Navigation.

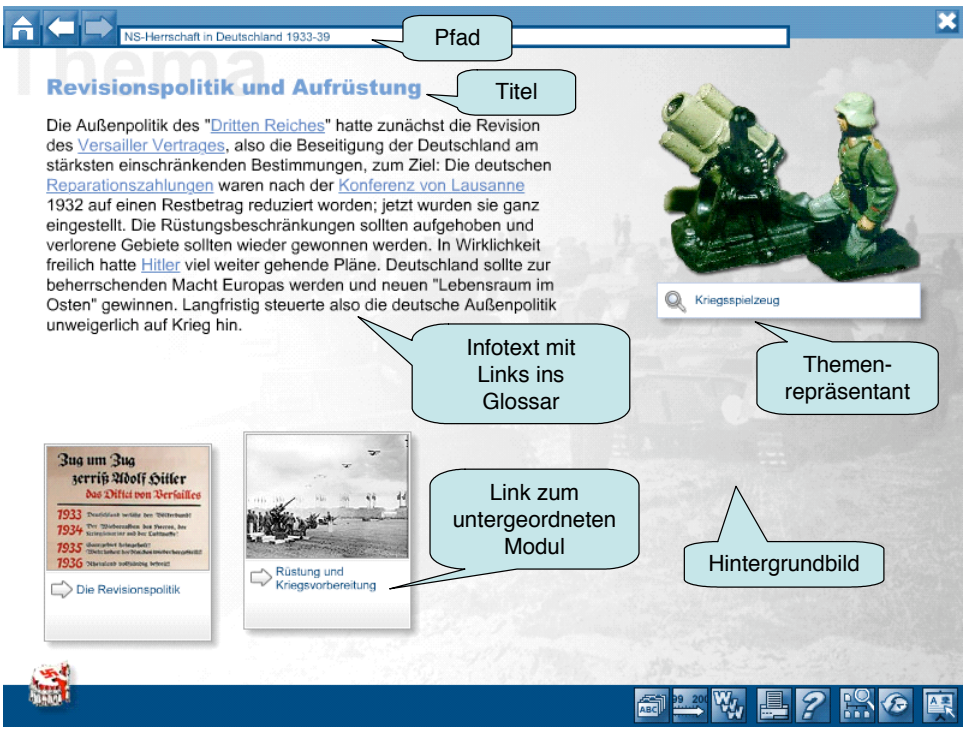


Abbildung 4.5.: Zentrale Ansicht einer typischen Anwendung

Sie betten jedoch die Verweise zu den Medien und Interaktionsbausteinen, die schließlich quasi die Blätter der Navigationsstruktur bilden, bereits in die Präsentation zu vermittelnder Inhalte ein – im wesentlichen handelt es sich dabei um Texte mit Verweisen zu Worterklärungen (Glossareinträge). Die Abbildung 4.5 zeigt einen solchen Themen-Screen. Die interaktiven Elemente der Navigation sind jeweils beschriftet. Die Kontextnavigation, also wiederum die Navigation zu Elementen die in einem direkten inhaltlichen Zusammenhang mit dem aktuell Präsentierten stehen, ist hier gekoppelt an die stilisierten Karten im linken unteren Bereich des Panels. Dabei ist die gesamte Fläche für die Mauseingabe sensitiv.

Es ist eine Vorgabe, der Framework-Instanz, dass alle Module, oder Materialien, die ein Autor einem Thema unterordnet durch ein solches Verweiselement Repräsentiert werden. Eine Sonderstellung nimmt das als Repräsentant ausgezeichnete Material ein. Dies wird entsprechend der voreingestellten Konfiguration des Meta-Objektprotokolls wie in der Abbildung dargestellt umgesetzt. Das ansonsten frei disponible Element kann im WYSIWYG-Modus jedoch frei skaliert und bewegt werden. Das gleiche gilt für alle in der Darstellung präsentierten Inhalte des Haupt-Panels des Themen-Screens. Textfarben, Schattenwürfe aber auch Textinhalte können ebenfalls in diesem Modus der visuellen Bearbeitung – also über eine Black-Box-Schnittstelle am Framework angepasst werden. Lediglich die Rahmen der Verweiskarten oder der Zoom-Button unterhalb des Repräsentanten sind auf Grund ihrer Projektbezogenen Individualität über Whitebox-

Schnittstellen im Sinne des Skinning-Mechanismus in den Screen-Design-Rahmen eingebracht.

Die Funktionalität zur Bearbeitung der Mausinteraktion und die Ausführung der entsprechenden Reaktion – in dem Falle der Navigation zu weiteren Navigationsknoten Medien-Panels, sind mit Hilfe der Standard-Link-Komponente des Frameworks realisiert. Das Metaobjekt-Protokoll ist dabei wiederum dafür verantwortlich, dass bei jedem Einfügen eine Moduls oder Medienelements in ein übergeordnetes Thema durch den Autor bzw. Redakteur die Metaobjekte entsprechend korrekt angelegt werden. Die Umsetzung der Metaobjekte in tatsächliche Objekte und damit ablauffähige Funktionalität, die dann etwa die Anzeige des gewünschten Ziel-Knotens bewirkt, ist die Aufgabe der entsprechenden Framework-Komponente.

Das in der Konzeption vorgestellte Gerüst der permanent, unabhängig von der aktuell sichtbaren Navigationsebene verfügbaren Rahmennavigation (siehe Abbildung 3.8 auf Seite 140), wird in der Geschichtssoftware ausgeprägt wie in den Abbildungen 4.6 bzw. 4.7 dargestellt. Die Gestaltung des oberen Teils des Navigationsrahmens macht bewusst Anleihen bei gängigen Werkzeugleisten von Webbrowsern. Damit wird der Anforderung Rechnung getragen, dass neben der Unterstützung fachlicher Arbeit auch allgemeine Medienkompetenz in Bezug auf die Anwendung von Standardnutzungsmustern gefördert werden soll. An dieser Stelle sei jedoch wiederholt erwähnt, dass das Framework bezüglich des Aufbaus und der Anordnung der Elemente der Rahmennavigation keine Vorgaben macht. Lediglich die grundsätzliche Möglichkeit, den in Panels disponiblen Inhalt durch die übergeordnete Funktionalität der Rahmennavigation zu überlagern und einige immer wieder benötigte Funktionalitäten sind im Framework angelegt.

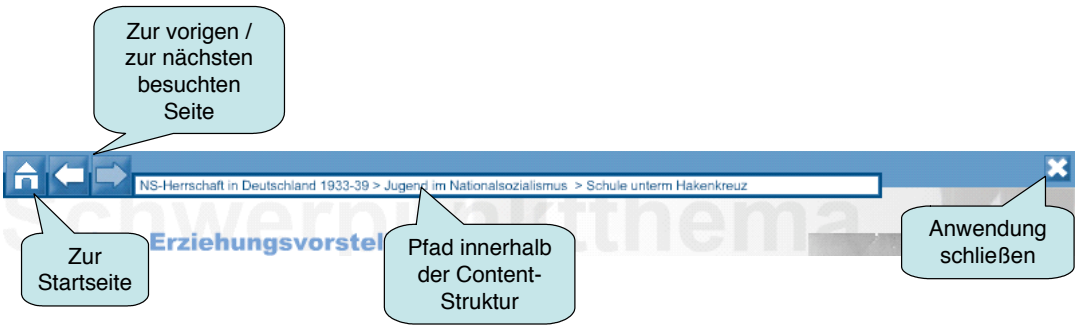


Abbildung 4.6.: Obere Steuerleiste in „Geschichte und Geschehen IV“

Die mit dem Häuschen-Symbol versehene Schalfläche führt den Nutzer auf eine Indexseite, die zum Programmstart gezeigt wird und die Verweise auf die oberste Gliederungsebene – hier konkret auf alle virtuellen Räume – besitzt. Die Vorwärts und Rückwärtspfeile bewirken die intuitiv erwartete Navigation innerhalb der Historie der bereits besuchten Inhalte um jeweils einen Schritt. Der Pfad fasst den aktuellen inhaltlichen Kontext zusammen. In der Abbildung wiedergegeben ist der Pfad zu einem Modul-Screen. Der letzte Eintrag „Schule unterm Hakenkreuz“ repräsentiert das dem Modul

übergeordnete Thema. Die einzelnen einträge des Pfades sind Sensitiv für Mauseingaben und führen den Nutzer bei einem Klick darauf zum jeweiligen Navigationsknoten. Das Drücken des mit dem Kreuz versehenen Buttons auf der rechten Seite schließlich führt zur Beendigung des Programms nach der üblichen Rückfrage, ob diese Aktion tatsächlich beabsichtigt ist.

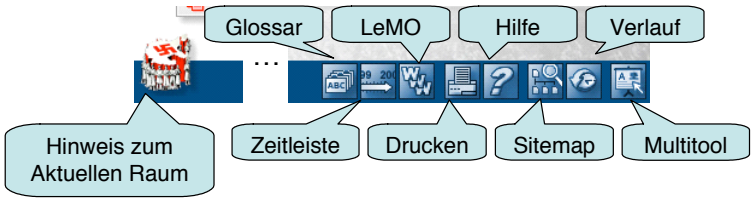


Abbildung 4.7.: Unterer Steuerleiste in „Geschichte und Geschehen IV“

Der in Anlehnung an die bekannten Betriebssystem-Pradigmen[2] unten horizontal angelegte Bereich der Steuerung nutzt ebenfalls vom Framework zur Verfügung gestellte Basisfunktionalitäten. Die für die meisten interaktiven inhaltezentrierten Anwendungen kanonischen Funktionen sind von hier aus erreichbar. Das Glossar beschreibt die Schlüsselbegriffe der Anwendung. Die Liste der alphabetisch geordneten Begriffe kann durch die Eingabe eines Suchbegriffs auf eine entsprechende Auswahl reduziert werden. Die Auswahl eines der Begriffe durch den Nutzer (Mausklick) bringt die zugehörigen Erläuterungen in einem separaten Panel des gleichen Fensters zur Anzeige. Dies ist ebenso spezifisch für diese konkrete Ausprägung wie der Umstand, dass die präsentierten Glossareinträge bezüglich des Layouts und des inhaltlichen Aufbaus einem festen Schema folgen, das über ein sehr einfaches Autoren-Back-End implementiert werden kann. Grundsätzlich wäre es auch hier denkbar, beliebige Medien und Funktionen zu adressieren.

Weitere Features, die als Erwartungskonform betrachtet werden können sind die Inhaltsübersicht (Sitemap) und die Suchfunktion. Beide Funktionalitäten dienen der gezielten inhalteorientierten Recherche und Navigation (weniger der Exploration) und sind deshalb funktional zusammengefasst. Beim Klick auf das in Abbildung 4.7 mit „Sitemap“ bezeichneten Symbol erscheint das in Bild 4.8 wiedergegebene Fenster, das Suche und Inhalteüberblick zusammenfasst und über die Dateireiter-Metapher wählbar macht. Die Sitemap zeigt dabei im Panel links die Struktur des Inhaltes in Form des aus vielen Anwendungen vertrauten interaktiven Baumes, bei dem Knoten durch den Klick auf das vorangestellte Plus- bzw. Minuszeichen in der bekannten Art entfaltet oder „kollabiert“ dargestellte werden können. Die Knoten selbst repräsentieren jeweils Knoten innerhalb des Navigationsraumes. Werden diese durch einen Klick gewählt, wird zunächst ein Überblick zu den zu erwartenden Inhalten im linken Panel gezeigt. Erst die explizite Aufforderung (in der Grafik 4.8 „Thema Öffnen“) lädt das entsprechende Element.

Ganz ähnlich aufgebaut ist die Suche. Der entscheidende Unterschied besteht im Aufbau

[2] *Taskleiste* bei Microsoft Windows oder das *Dock* bei Mac OS X

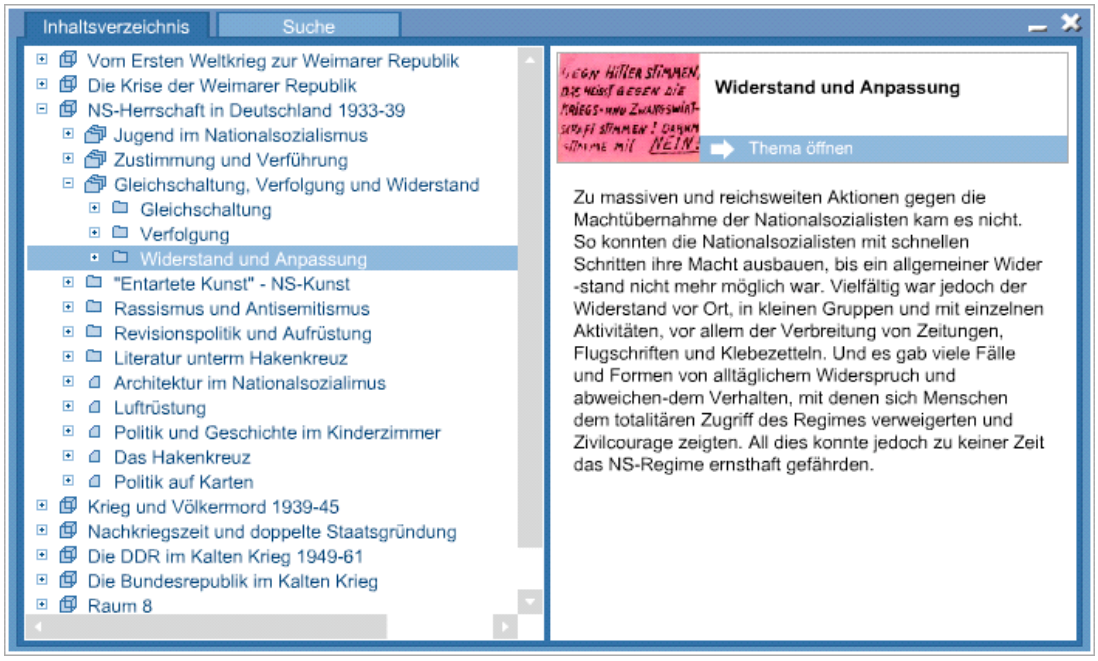


Abbildung 4.8.: Inhaltsübersicht (Sitemap)

der Linken Seite des Fensters. Das nach dem Aufruf leere Panel wird gefüllt mit dem Ergebnis der Suche. Die zum eingegebenen Suchbegriff passenden Knoten werden nach dem Absenden der Suche aufgelistet und bezüglich ihres Typs gruppiert (siehe Abbildung 4.9). Die Interaktionen, die zum Aufruf des jeweiligen Knotens führen, entsprechen denen der Nutzung der Sitemap. Die in diesem Anwendungskontext für sinnvoll erachtet Kongruenz der Interaktionsmuster lässt die Entwicklung optimal vom Wiederverwendungseffekt des Frameworks profitieren – die gleichen Anlagen können direkt eingesetzt werden.

Die verwendete Bildsprache ist hier klassisch, orientiert an Ordnern und enthaltenen Dokumenten. Wobei lediglich der Typ der Medien durch entsprechende Icons unterscheidbar wird. Dies ergab sich aus einer Design-Entscheidung im Sinne der Anforderung, sich möglichst nahe an der „Wirklichkeit“ der Metaphorik aktueller IT-Systeme zu bewegen. Das entsprechende Skinning des Screen-Design-Rahmens an dieser Stelle kann natürlich grundsätzlich mit beliebigen Grafiken umgesetzt werden. Deutlich wird dies am Beispiel des Icons, was dezidiert auf das systemindividuelle Konzept des virtuellen Raumes hinweist.

Eine weitere aus Nutzersicht vertraute Funktionalität zur Unterstützung einer komfortablen Navigation ist die „History“ – also die Chronik der während der Nutzungssession bereits besuchten Knoten innerhalb der Navigationsstruktur. Der Aufruf dieser Funktion listet die besuchten Knoten jeweils mit Bezeichnung und einem Icon analog zum globalen Style-Guide (also wie in den anderen Ansichten) auf. Der Klick auf einen Eintrag

Abbildung 4.9.: Suche

führt direkt zum Öffnen des betreffenden Bildschirmelements.

Die globale verfügbare Druckfunktion ist insofern individuell für den Anwendungsbereich angepasst, als dass der so genannte Fokus-Handler für die Auswahl des zu druckenden Bereichs der Anwendung so konfiguriert wurde, dass bei einem geöffneten modalen Fenster der Inhaltepräsentation (also beispielsweise einem offenen Fenster zur Bildbetrachtung) immer ausschließlich der Content dieses Fenster gedruckt wird. Dabei wird eine drucktaugliche Sicht auf die darstellungsunabhängigen Teile der Meta-Objektbeschreibung erzeugt. Texte etwa, die in der Bildschirmdarstellung gescrollt werden müssen, werden vollständig sequenzialisiert ggf. auf mehrere Seiten verteilt gedruckt. Kopf- und Fußzeile sind ebenfalls angepasst entsprechend der Produktanforderungen.

Eine vollkommen auf diesen Anwendungskontext zugeschnittene, aus dem Framework abgeleitete Funktionalität der Rahmennavigation ist eine zusätzliche, zeitlich gliedernde Sicht auf historische Inhalte. In Ergänzung zur fachsystematischen Gliederung des Systems werden hier historische Artefakte und Ereignisse über einer Achse von Jahreszahlen geordnet. Dem Nutzer wird dazu eine in sich geschlossene bildschirmfüllende Anwendung präsentiert. Über einen Zeit-Index und die üblichen Scrollmechanismen kann er jeweils die Darstellung eines Zeitraums von fünf Jahren in seinen Fokus rücken. Die über dem Zeitstrahl abgebildeten Titel und Grafiken sind mit sehr einfachen Erläuterungen zum Eintrag verknüpft, die auf Mausklick in einem eigenständigen Fenster präsentiert werden. Diese historische Zeitleiste markiert auch die zentrale Schnittstelle zum ursprünglichen LeMO-System. Die Inhalte sind so gewählt, dass sie jeweils einen kor-

respondierenden Eintrag im Web-Portal besitzen. Dieser ist über einen Verweis (eine Schaltfläche) in Ergänzung zum Inhalt des Fensters direkt aufrufbar – eine bestehende Internetverbindung vorausgesetzt. Auch dieses Fenster besteht im Kern aus einem Panel des Frameworks, das mit Hilfe einer entsprechenden Stilvorlage einheitlich gestaltet wird. Ein dezidiertes Werkzeug unterstütze dabei den sehr spezifischen, ansatzweise durch stereotype Tätigkeiten geprägten Prozess der Strukturierung der benötigten Inhalte und deren Umsetzung in die Teilanwendung (siehe Abbildung 4.11 auf Seite 258).

Der mit „WWW“ gekennzeichnete Button hat die einfache Aufgabe, die Startseite der ursprünglichen Web-Anwendung „LeMO“ in einem externen Browserfenster zu laden – damit ganz allgemein eine logische Verbindung zum Ursprung der Idee herzustellen und dem Nutzer darüber hinaus eine zusätzliche Möglichkeit der ergänzenden Recherche zu bieten. Dafür ist ebenfalls eine Internetverbindung erforderlich.

Hinter dem Button mit dem Symbol eines Schaubilds (ganz rechts in Abbildung 4.7) verbirgt sich ein Editor zum freien Gestalten und Verwalten einfacher Bildschirmpräsentationen durch den Nutzer. Dazu werden Funktionalitäten des WYSIWYG-Editors des Frameworks zugänglich gemacht. Diese kann der Anwender im Zusammenhang mit einzelnen Seiten (Panels im Sinne des Frameworks) seiner Präsentation uneingeschränkt nutzen. Das heißt er kann in der Implementierung von „Geschichte und Geschehen IV“ alle frei disponiblen Bedienelemente zur Gestaltung eigener kleiner interaktiver Anwendungen nutzen. So erhält er etwa die Möglichkeit beliebiges Audio-, Video, und Bildmaterial einzubinden. Die entsprechenden zur Anzeige benötigten Funktionen und Bedienelemente sind automatisch mit den Inhalten verknüpft. Grundsätzlich sind auch eine detailliertere Vorauswahl der verfügbaren Elemente und eine feinere Steuerung der Rechte der Nutzung möglich. Die Seiten kann der Nutzer jeweils logisch in einem persönlichen Ordner zur späteren Verwendung speichern. Physisch wird damit wiederum ein Meta-Objekt zur Interpretation durch das Framework gespeichert.

Die Hilfe-Funktion schließlich wird über den Button mit dem Fragezeichensymbol aufgerufen. In einer ähnlichen Form der Präsentation wie sie bei der Sitemap zur Anwendung kommt, werden hier die Funktionalitäten des Systems systematisch gegliedert Präsentiert und die Art ihrer Verwendung vermittelt. Die Gliederung wird in einer entsprechenden interaktiven Baumstruktur in einem separaten Panel auf der linken Seite eines Fensters angezeigt. Nach der Auswahl eines der Gliederungspunkte erscheint im rechten Panel die dazugehörige Erläuterung. Diese vermittelt die Informationen zunächst kompakt und im Überblick – zunächst durch textuelle Beschreibungen oder Grafiken. Weiterführende bzw. vertiefende Informationen sind über Verweise auf entsprechende Inhalte gemacht. Diese können dann allerdings wiederum vollwertige, unter der vollen Nutzung des Frameworks entwickelte, Teilanwendungen in eigenständigen Fenstern sein. Im Projekt kommen dazu in aller Regel kleine Video-Mitschnitte typischer Abläufe der Nutzung der Oberfläche (Screen Captures) zum Einsatz. Diese nutzen Anzeige und Steuerung die Systemweit einheitlich ausgeprägten Player-Funktionen.

Die mögliche Dynamik auch in der Instanziierung der Rahmennavigation verdeutlicht eine Variante der unteren Werkzeugleiste. In ihrer Ausprägung für die Umgebung des

virtuellen Raumes etwa erhält sie zwei zusätzliche Schaltflächen: eine die den Nutzer zu zusätzlichen Erläuterungen über den Raum führt. Die zweite öffnet eine kontextabhängige Auswahlliste, mit einem Überblick über alle interaktiven Objekte innerhalb des Raumes. Die Auswahl der in der Liste miniaturisiert dargestellten Elemente bzw. ihrer Bezeichnungen unterstützen das Auffinden des Gegenstandes im Raum und damit die Navigation zu den hinterlegten Inhalten.

Diese Funktionalität stellt ein gutes Beispiel für ein Ergebnis des diskursiven Prozess des kollaborativen Prototyping dar: so wurde während der Arbeit am Arrangement bzw. der Auswahl der Inhalte und der damit verbundenen permanenten Navigation durch die virtuellen Räume deutlich, dass innerhalb der Nutzung damit zu rechnen sein würde, dass der Reiz des spielerischen Zugangs erschöpft und die Phase der Exploration innerhalb der Räume ab einem bestimmten Punkt abgeschlossen sein wird. Es war nach der intensiven Nutzung innerhalb der Entwicklung klar, dass diese Suche sogar einen strukturierten Arbeitsfluss, der sich auf die historische Arbeit im fortgeschrittenen Unterricht konzentrieren sollte, auf die Dauer eher stören würde. Mit der spontan eingeführten Funktionalität wurde es nun möglich interessierende Systemteile (etwa gemäß der Unterrichtsplanung) ohne Umwege aufzusuchen; ohne jedoch, wie bei der Nutzung des Inhaltsverzeichnisses, auf die Einstimmung auf den historischen Hintergrund durch das Raumpanorama gänzlich verzichten zu müssen.

Neben der Gestaltung und der Anordnung der Schaltflächen besteht die Instanziierung des Frameworks für die Umsetzung der Rahmennavigation aus technischer Sicht in der Konfiguration der Aufrufe der jeweiligen Funktionen auf der Ebene der Metaobjekte. Die Abbildung A.4 im Anhang zeigt unter der Notiz „Verweise auf komplexe Teilanwendungen“ die Adressierung der bei einem Klick jeweils zu initialisierenden Metaobjekte (z.B. der Sitemap über `res/sitemap.xml`). Die aufgerufenen Unteranwendungen sind ebenfalls auf der Grundlage von Framework-Komponenten generiert. Sie sind ihrerseits aus der adressierten XML-serialisierten Meta-Objektbeschreibung über das beschriebene Reflection-Muster in ablauffähige Software umgesetzt. Neben den festen *Tags*, die das Framework zur Einbindung zur Einbindung von Funktionalitäten in den Navigationsrahmen interpretieren kann (wie etwa ⟨`SITEMAP_URL ...`⟩), werden beispielsweise neue Buttons zum Aufruf von Teilanwendungen über den generischen Link-Typ eingebunden. Dazu muss über eine White-Box-Schnittstelle die Zuordnung zwischen einem generischen Button-Bedienelement und dem „Einsprungpunkt“ für die aufzurufende Teilanwendung über die Registrierung eines identifizierenden Bezeichners vorgenommen werden.

Disziplingerechte Entwicklerschnittstellen

Als interessantester Teil der Konkretisierung aufgabengerechter Entwicklerschnittstellen kann die Ausprägung des Werkzeugs für die Fachautoren betrachtet werden. Wie bereits angedeutet mündete dieser Prozess zunächst in die Sackgasse der Transformation und des Imports semistrukturierter Manuskripte. Erst die Arbeit mit einem Mock-Up führte zu einer für alle akzeptable Lösung. Die dazu verwendete Hilfskonstruktion auf der Grundlage des *Microsoft LRN-Editor* wurde zur Illustration der Mock-Up-Methodik

bereits erörtert (Siehe Grafik 3.21 auf Seite 203). Das Ergebnis des Diskussionsprozesses und der intensiven experimentellen kollaborativen Arbeit an dieser speziellen Form eines Oberflächenprototypen führte schließlich zu einem Konsens, der in den Grundzügen des Aufbaus und der Bedienung denen des LRN-Editors entspricht. Die in der gemeinsamen Arbeit des experimentellen Einpflegens, Strukturierens und Inszenieren von Inhalten erkannten Defizite wurden systematisch ausgeräumt. Das Endergebnis der sich anschließenden evolutionären Entwicklung einer für den Aufgabenbereich individualisierten Entwicklungsumgebung auf der Grundlage des Content-Grabber-Rahmens ist in Abbildung 4.10 dargestellt.

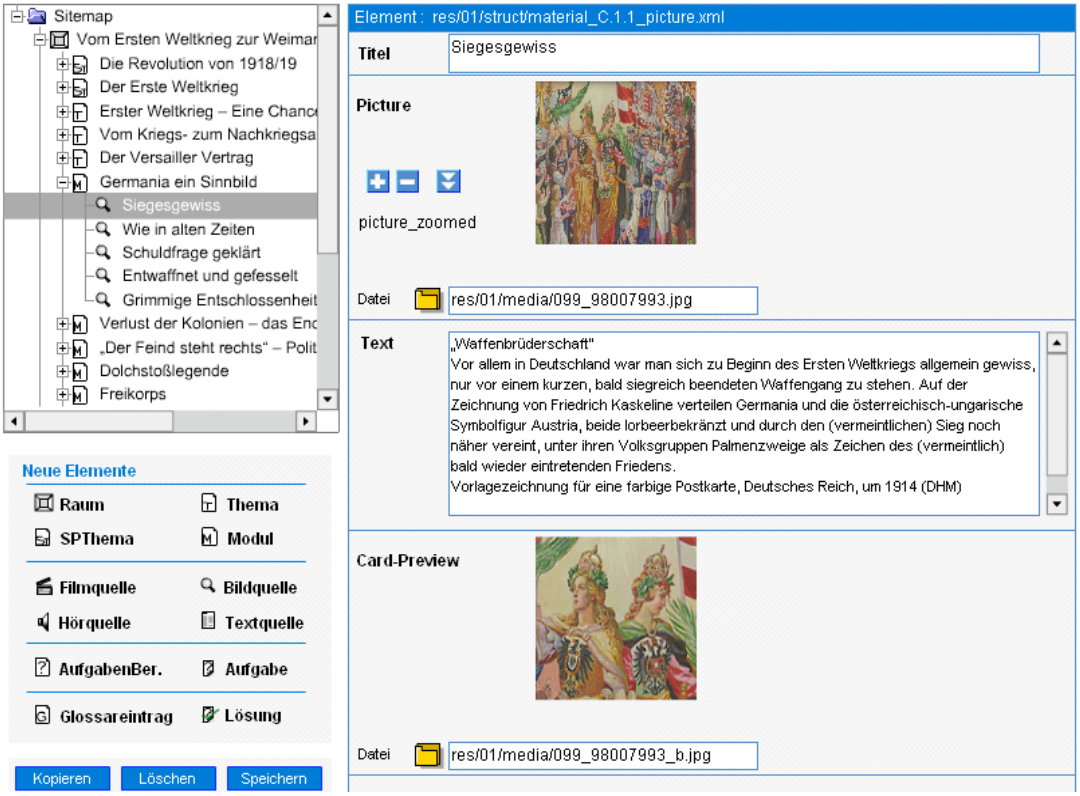


Abbildung 4.10.: Aufgabenangepasste Autorenschnittstelle

Mit dem im Projekt als Wizzard bezeichneten Werkzeug strukturieren Autoren das System inhaltlich, indem Sie etwa Gliederungsebenen entsprechend der Kategorien des gemeinsam entwickelten Verständnisses einfügen und inhaltlich ausarbeiten. Dazu ziehen sie eine der gewünschten Ebenen, repräsentiert durch entsprechende Icons, aus der Rubrik „Neue Elemente“ an die gewünschte Stelle der Gliederungsansicht. Die Inhalte des in der Gliederungsansicht gewählten Elements werden dann auf der rechten Seite zur Bearbeitung geöffnet. Entsprechend der vorher festgelegten aber auch dynamisch veränderbaren Vorlagenstruktur werden hier Inhalte für tatsächliche Anzeige- bzw. Interaktionselemente der mit der Gliederungsebene assoziierten Anwendungskom-

ponente angelegt bzw. manipuliert. Implizit erzeugen die Autoren mit der Bearbeitung Meta-Level-Objektbeschreibungen einzelner Panels oder auch ganzer Aggregate von Teilanwendungen sowie der gesamten Navigationsstruktur gemäß dem Meta-Level-Objektprotokoll.

Das Anlegen eines Aufgabenblockes und der untergeordneten Aufgaben erfolgt beispielsweise über das Ziehen des Entsprechenden Symbols („AufgabenBer." bzw. „Aufgabe") in die Gesamtgliederung. Eine Regel stellt dabei sicher, dass Aufgaben immer in den Container eines Aufgabenbereichs gegliedert werden und dieser wiederum immer mit einem Material innerhalb eines Moduls verknüpft wird. Auch das entspricht einer strukturellen Vorgabe der Anwendung, die im Zwischenschritt der Ausprägung des Navigationsrahmens implementiert wurde. Die Reihenfolge der Anlage der Aufgaben innerhalb des Komplexes bestimmt schließlich auch den Lernweg, also die Abfolge der Präsentation der Aufgaben. Es liegt hier in der Verantwortung des Autors, die Aufgaben in ihrer didaktischen Folgerichtigkeit konzeptionell zu planen und anzulegen – also etwa Aufgaben zu Entwerfen, die aufeinander aufbauen, der Lernertrag der einen die Voraussetzung für das Verständnis der nächsten Aufgabe ist.

Autoren verfassen mit dem Wizzard etwa auch Texte, aus ihrer Sicht zum Beispiel „Themen-Infotexte". Als sinnvoll erwies es sich, dies en gros und fokussiert auf eben diese ganz konkrete Aufgabe des Formulierens zu tun. Mit der projekt- bzw. produktspezifischen Textkategorie verbinden sich ein vorgegebener Umfang und ein festes inhaltliches Abstraktionsniveau sowie ein bestimmter Stil der Ansprache der Rezipienten. Das Anlegen und Bearbeiten der Texte und die klare Navigation zwischen ihnen, lediglich bezogen auf die übersichtliche fachsystematische Gliederung sollten effektiv, losgelöst von jeglichem Aspekt der Präsentation unterstützt werden. Das heißt, von Formatierungen, die Einbettung ins Layout und dem Satz sowie die Einbindung interaktiver Elemente wie den Texteditor sollte zunächst abstrahiert werden. Die Interaktionen zur Bearbeitung des Textes wurden auf das notwenige Minimum reduziert. Selbst die Ablage der Arbeitsergebnisse ist frei von technischen Abstraktionen des Betriebssystems. Der Autor Speichert den Text innerhalb der fachsystematischen Strukturen, auf der Grundlage der abgestimmten Kategorien: also beispielsweise den „Themen-Infotext" als integralen Bestandteil eines Themas, was wiederum einem „Schwerpunkt" Schwerpunkt untergeordnet sein kann. Es werden keine Laufwerke und Ordnerstrukturen sichtbar – auch die Bildsprache der Übersichtsdarstellung ist entsprechend konkretisiert.

Der pragmatische Eindruck den die Abbildung 4.10 von der Lösung vermittelt ist der Tatsache geschuldet, dass die evolutionäre Weiterentwicklung dieser Schnittstelle bis zum Zeitpunkt dieser Veröffentlichung nicht weiter vorangetrieben wurde. Das ebenfalls auf Macromedia Flash basierende Client Interface erfüllte seinen Zweck jedoch vollkommen. Es zeigt, was Autoren und Redakteure zum Vorstrukturieren und Stoffsammeln im Ergebnis eines evolutionären Prozesses der Anpassung zwischen Werkzeug und Tätigkeit benötigen. Dieses Interface materialisiert den Wert der *domänenzentrierten Einfachheit*, der als eine entscheidende Grundlage des Vorgehens eingeführt wurde. Eine schnelle Umsetzung des Werkzeugs mit unmittelbaren Anpassungsmöglichkeiten an die

Bedürfnisse der Autoren war die Voraussetzung für die direkte diskursive Zusammenarbeit, bereits während der Phase der inhaltlich konzeptionellen Arbeit. Nur durch die Möglichkeit, Features ad hoc einbauen zu können – direkt aus der Diskussion heraus – war es möglich, die produktive Arbeit der inhaltlichen Konzeption den Gesamtentwicklungsprozess jederzeit dominieren und technische Aspekte in den Hintergrund treten zu lassen.

Der Einsatz dieses Werkzeugs im Projekt zeigt dabei, dass die dafür bereitgestellten Interaktionsmuster zur Unterstützung dezidiert dieser Aufgabe, tatsächlich die Konzentration auf diese konzeptionellen, planenden Schritte unterstützen konnte. Die Separation der Ausgestaltung der Aufgaben im Detail auf der Ebene der WYSIWYG-Bearbeitung fördert die Strukturierung der Arbeit. So wird ein inhaltlicher Rahmen geschaffen, der schließlich ausgefüllt werden kann. Ein Verlieren im Detail der Umsetzung einzelner Bausteine wird eingedämmt. Damit wird auch das Risiko minimiert, dass die Arbeit am Detail den Blick für übergeordnete didaktische Konzepte verstellt – viele gute Ideen für Arbeitsblätter und deren aufwändige Umsetzung führen nicht zwangsläufig zu einem schlüssigen Konzept für einen gesamtheitlichen Lernweg.

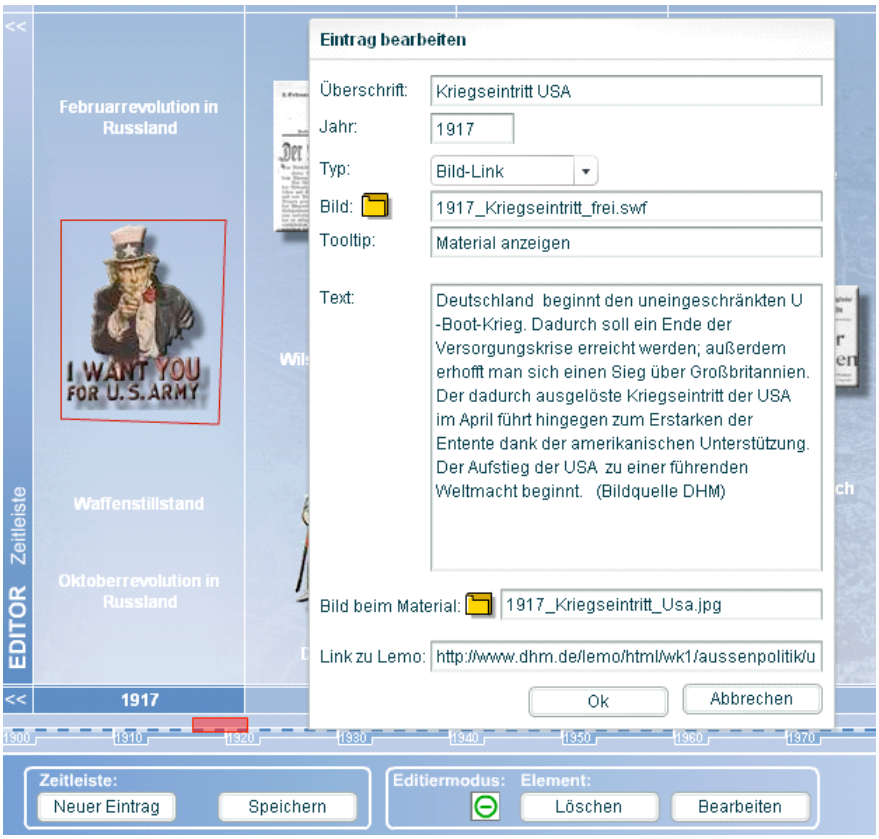


Abbildung 4.11.: Aufgabenangepasste Autorenschnittstelle zur Implementierung der Zeitleiste

Eine Spezialisierung dieser Entwicklerschnittstelle, die das fachliche gliedernde Arbeiten explizit unterstützt ist in Abbildung 4.11 gezeigt. Dieser dezidierte Editor wird verwendet, um den Zeitstrahl zur zeitlichen Einordnung und Visualisierung historischer Ereignisse auszugestalten. Auch er ist mit einfachsten Mitteln auf der Grundlage des Frameworks erstellt und systematisch an die Bedürfnisse der sehr umfangreichen, zum Teil mit Routinetätigkeiten verbundenen aber dennoch historisch, konzeptionell anspruchsvolle Arbeit angepasst.

Die Konkretisierung disponibler Elemente und damit die Verfeinerung der Interaktivität sowie der Gestaltung der Nutzerschnittstelle waren an die Lernkurve der Fachentwickler gekoppelt. Zunächst setzte ein technisch sachverständiger die im Experiment und in der Diskussion getroffenen Entscheidungen in direkter Zusammenarbeit um. Aufgrund der Einfachheit des bereits bestehenden Interfaces war es im Projekt sehr schnell möglich, dass zumindest Redakteure in die Lage versetzt waren, diese Aufgabe zu übernehmen und die Parameter der Metaobjektbeschreibung selbstständig zu bestimmen. Im Ergebnis der engen Zusammenarbeit und häufiger Konsultationen war es Redakteuren schließlich möglich die technologienäheren Aufgaben der beliebigen Verfeinerungen der Interaktionselemente über deren Meta-Level-Konfiguration mit Hilfe des so genannten *Inspektors* selbstständig vorzunehmen (siehe auch die Erläuterungen zu Bild B.3 auf Seite 287).

So konnte ein Redakteur beispielsweise entscheiden, dass alle Verweise auf externe Materialien mit immer der gleichen Schaltfläche aus einem Aufgabenfenster heraus aufgerufen werden. Wie in der Abbildung B.4 zu sehen, folgt die Präsentation der Schaltfläche konsequent dem festen Schema der Materialverweise der Navigationspanels. Dementsprechend ist das Element auch in den Vorlagenkatalog (Metalevelobjekt) für die Erstellung digitaler Arbeitsblätter aufgenommen worden. Aus der Sicht der Autoren, wird nun lediglich noch eine Materialszuordnung vorgenommen (z.B. mit dem Wizzard). Die entsprechende Schaltfläche ergibt sich dann beim generieren der Basisobjekte zur Laufzeit implizit. Sie entspricht wiederum einem disponiblen Element, welches mit Hilfe der WYSIWYG-Schnittstelle weiter konkretisiert werden kann – im Layout arrangiert, mit einem Schlagschatten versehen, etc. Auch diese von den Redakteuren getroffenen und über den *Inspektor* direkt implementierten Entscheidungen können in Form von Stilen und Vorlagen für die erneute bzw. pauschale Verwendung materialisiert werden.

Ansonsten gestaltet sich die Arbeit im mehrfach angesprochenen WYSIWYG-Modus wie intuitiv zu erwarten. Dabei entsprechen die Bedienmuster im Wesentlichen denen der interaktiven Arbeitsblätter, wobei hier alle Freiheitsgrade der Bearbeitung a priori zur Verfügung stehen. Zusätzlich besteht die Möglichkeit Zugriffsrechte einzelner Elemente für die Interaktion im End-Nutzer-Modus einzustellen. Das Anlegen von Medienelementen und das Bearbeiten von Texten – für das Lektorat der von den Autoren eingestellten Inhalte unabdingbar – nutzt dieselben Software-Komponenten, wie die Editoren der beschriebenen Nutzersicht.

Zusammenfassend ist festzustellen, dass sich auf evolutionärem Wege ein fester Arbeitsablauf der Produktion einstellte, der die Arbeit ausgehend von der inhaltlichen

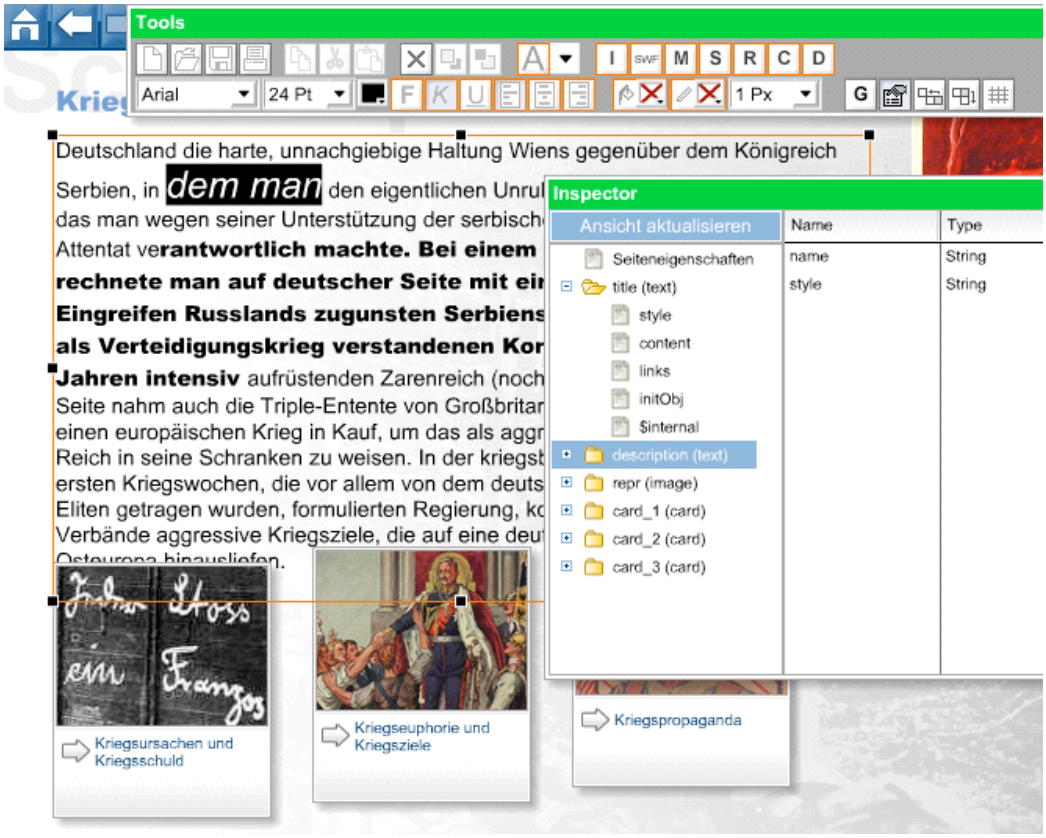


Abbildung 4.12.: WYSIWYG bei der redaktionellen Arbeit

Konzeption und der Stoffsammlung mit Hilfe des Wizzards oder auch des Zeitleisten-Editors über die schrittweise Ausgestaltung der interaktiven Inhalte, bis hin schließlich zur redaktionellen Verfeinerung und Konsolidierung des Gesamtsystems gliederte. Damit realisiert sich der Anspruch, bereits auf inhaltlicher Ebene deduktiv, ausgehend von konzeptionellen Entwicklungsphasen verfeinernd und verflochten mit der technologischen Entwicklung des Produkts zu Arbeiten. Es konnte gezeigt werden, dass der übliche Bruch zwischen einer geforderten vollständigen, fachlich inhaltlichen Ausarbeitung und einer davon vollkommen entkoppelten direkten Implementierung aufgehoben werden kann.

4.2.2. Zusammenführen der Disziplinen

Die Konvergenz der Arbeitswelten von Entwicklern, Redakteuren, Autoren und Grafikern in den frühen Phasen des Projekts – während der Ziel- und Anforderungsdefinition sowie der Konzeption wird im Spannungsfeld zwischen den Realitäten der Fallstudie und den in dieser Arbeit vorgestellten idealtypischen Lösung diskutiert. Die gemachten Beobachten bestätigen im Wesentlichen die Tragfähigkeit der Konzeption. Dabei gibt es

Punkte, an denen die im vorangegangenen Kapitel präsentierten Lösungsansätze über die im Projekt tatsächlich durchgeführten Aktivitäten hinausgehen. Die angegebenen Konzepte haben dabei versucht Lösungen für die in der Evaluation erkannten Defizite im Detail zu antizipieren.

Ethnographische Untersuchung und Mock-Ups

Die ethnografische Untersuchung der Arbeitsweise der Zieldomäne und insbesondere der Produktionsumstände bei der Bearbeitung von Projekten, wie dem adressierten erfolgte in ausführlichen Workshops – so häufig wie möglich vor Ort beim Verlag. Dabei hatten die Entwickler der technologischen Teile des Zielsystems die Gelegenheit, sowohl globale Abläufe der Zusammenarbeit zwischen Redaktion, Autorenschaft und Grafikdesign zu beobachten, als auch die tatsächlichen Handgriffe von Redakteuren und Autoren bei der Arbeit an der inhaltlichen Ausgestaltung solcher Systeme nachzuvollziehen.

Die Hauptarbeit bestand dabei zunächst im Zusammentragen und Strukturieren des historischen Quellenmaterials. Dabei wurden im wesentlichen Verweise auf die Fundorte der Materialien in Form von Manuskripten zusammengefasst, gegliedert kommentiert und mit Regieanweisungen für die Implementierung versehen. Deutlich wurden dabei zwei entscheidende Dinge: zum einen machte der Verlag von Beginn an Vorgaben für den Aufbau und die Gliederung der Manuskripte. Eine so genannte Schreibanweisung sollte sicherstellen, dass die vermeintliche Heterogenität, die sich aus verschiedenen Stilen und Arbeitsweisen unterschiedlicher Autoren ergeben würde, unterdrückt wird. Es stellte sich (offenbar auch für die Redakteure erwartungsgemäß) als schwierig heraus, die damit verbundene Forderung nach Einheitlichkeit durchzusetzen. Instrumentell war dies allein dadurch schwierig zu unterstützen, dass die Ergebnisse der Arbeit ausschließlich auf Papier und in Aktenordern materialisiert wurden. Versuche die „Digitalisierung“ der Arbeit durch den Mock-Up eines ein einheitlichen Dokumentenformates voranzutreiben scheiterten wie beschrieben an der geringen Praktikabilität des Ansatzes bzw. der fehlenden Stringenz der Führung durch die Technologie.

Zum anderen offenbarten die teilnehmende Beobachtung und insbesondere die unmittelbare Auseinandersetzung mit der gemeinsamen Interpretation im direkten Diskurs der Entstehung, dass die Regieanweisungen unter keinen Umständen eine Grundlage für eine Umsetzung im Sinne einer Spezifikation bilden konnte. Viele Aspekte der Interaktivität und auch der grafischen Gestaltung blieben implizit und unklar, hauptsächlich auf Grund der Tatsache, dass sich einerseits die Fachentwickler der technischen Restriktionen und Potentiale nicht bewusst waren, und andererseits, die Technologieentwickler, die Feinheiten didaktischer Konzeptionen nicht verstanden.

Neben den häufigen intensiven Treffen zwischen Technologieentwicklern und Domänenexperten unter Workshop-Bedingungen und über verschiedene Standorte hinweg, bestand die Möglichkeit eine Historikerin mit fachdidaktischem Hintergrund direkt vor Ort in das Team der Technologieentwicklung zu integrieren. Auch Sie war mit Aufgaben der fachlich inhaltlichen Ausgestaltung betraut. Ihre Arbeitsweise konnte entsprechend in

unmittelbarer täglicher Kooperation studiert und zum Teil in tatsächlich teilnehmender Betrachtung durch die Technologieentwicklung nachvollzogen und adaptiert werden.

So konnte zumindest im Ansatz, als Startpunkt für die evolutionäre Ausarbeitung, die typische Art und Weise ermittelt werden, wie:

- Wie der Inhalt insgesamt gegliedert, aufbereitet und zu einem geplanten Produkt strukturiert wird: dabei wird im Wesentlichen zunächst Fachsystematisch gegliedert. Texte und Ideen zu weiteren Inhalten werden zu Papier gebracht und in Manuskripten zusammengefasst. Es wurden jedoch auch Bemühungen erkennbar, die Struktur der Dokumente zu vereinheitlichen – mit dem Ziel der Zusammenführung der Arbeitsergebnisse der verteilt arbeitenden Autoren und im Hinblick auf die Abbildung der Ergebnisse in ein in sich konsistent strukturiertes System. Der Mangel des Medienbruches zwischen den Ergebnissen auf Papier und der nötigen Digitalisierung sowie die fehlende instrumentelle Unterstützung der vereinheitlichten Strukturierung war allen beteiligten bewusst.
- Medien recherchiert, gegliedert und eingebunden werden: Entsprechend der Erfordernisse der Arbeit an der Gesamtkonzeption, wurden mediale Inhalte lediglich als „hard copies“ oder nur mit Verweisen auf mögliche Quellen in die Manuskripte eingebunden. Auch dabei war Konsens, dass die Wirkung des tatsächlich zu verwendeten Materials bei der Rezeption auf diese Weise kaum zu antizipieren ist – etwa auch die satztechnische Einbettung und die Inszenierung im Zusammenspiel mit interaktiven Elementen zur Manipulation bereits während des Prozesses der Konzeption besser erfahrbar und evaluierbar sein sollte.
- Wie Konzepte der Interaktion (Tätigkeiten der Domäne) übertragen werden in mediendidaktische Konzepte zur Vermittlung von Handlungskompetenz und welche Gestaltungsspielräume dazu antizipiert werden: Die systematische Rezeption, Interpretation und Bewertung historischer Quellen sowie die Präsentation der Ergebnisse wurden als zentrale Tätigkeiten der historischen Arbeit und damit der Arbeit bei der konzeptionellen Ausgestaltung des Systems erkannt. Die Kompetenzen zur Beherrschung dieser Tätigkeiten entsprechen denen, die das System vermitteln bzw. deren Anwendung es unterstützen soll. Die in der Konzeption ausgeschöpften Möglichkeiten beschränken sich allerdings zunächst meist auf die einfachen Muster gängiger Produktionen. Selbst Paradigmen wie modale Fenster wurden zunächst eher als zu kompliziert erkannt – weniger als innovatives Mittel, etwa für das Inbeziehungsetzen von Inhalten und damit das Schaffen neuer Perspektiven.
- Die Zusammenarbeit zwischen den beteiligten üblicherweise organisiert ist: über die in der Beschreibung der Teamzusammensetzung zusammengefassten Erkenntnisse hinaus (siehe die Seiten 238), ff.) wurde deutlich, welche Strukturen invariant bleiben mussten und für einen Prozess im Sinne der Co-adaption zwischen den Disziplinen zur Disposition stehen würden. Die grundsätzliche Bereitschaft von Autoren bzw. Redakteuren ihr Rollenverständnis dahingehend zu ändern, dass sie auch Aufgaben der technischen Umsetzung übernehmen zum Beispiel war erkennbar.

Auch die damit verbunden veränderten Auftraggeber-Auftragnehmerverhältnisse schienen akzeptabel.

Das Ergebnis dieses Prozesses war schließlich der erste Versuch der Ableitung und Einführung einer instrumentellen Unterstützung durch eine automatisiert auswertbare Vorlage für die Erstellung von Manuskripten. Hauptsächlich auf Grund der schlechten Praktikabilität mündet dieser Mock-Up-Versuch schließlich in die Vorstufe der als *Wizzard* beschriebenen Entwicklerschnittstelle. In ersten kollaborativen Gehversuche wurden mit Hilfe des simplen Werkzeugs Inhalte erfasst und strukturiert. In moderierten Mock-Up-Sessions erarbeiteten sich Fachautoren und Redakteure unter Hilfestellung durch die Technologieentwickler die Anwendung der Technologie. Umgekehrt konnten die Systementwickler die Arbeitsweise der Fachexperten unmittelbar nachvollziehen – nicht nur in der Beobachtung sondern auch in eigenen Versuchen der fachlichen Gestaltung.

Auf der Grundlage der während dieser Sitzungen gewonnen Erkenntnisse wurde zum Teil bereits zeitlich parallel die schließlich tragfähige Entwicklungsschnittstelle – der Wizzard – ausgeprägt. Er instanziiert gleichermaßen den Content-Grabber-Rahmen, wie auch Teile des funktionalen Rahmens – dem Entwickler in seiner Rolle als Nutzer des Frameworks bleibt diese Unterscheidung der Bausteine des Frameworks verborgen. Es erfolgte also eine erste Konkretisierung des Meta-Objekt-Protokolls, indem festegelegt wurde, welche der Bedienvorgänge am Wizzard, welche Auswirkungen auf die Metaobjekte haben und wie sich dies schließlich auswirkt auf konkrete Objekte der Anwendung sowie auf deren Präsentation. Dieses Werkzeug bildet damit die initiale Prototyping-Umgebung, die im Zuge der kollaborativen Entwicklung verwoben mit dem eigentlichen Artefakt kontinuierlich, evolutionär ausgebaut wird.

Interaktives Prototyping und integrierte Organisationsentwicklung

In Ergänzung zur Beschreibung der Anwendung des Frameworks im Rahmen des Projekts, wird nun nochmals der Fokus auf den Beitrag des vorgestellten Ansatzes zur Überwindung der Brüche zischen den Disziplinen (bzw. den diese vertretenden Organisationsstrukturen) und schließlich der Anwendung gesetzt.

Mit der ersten Ausprägung des Frameworks und damit der Ableitung einer ersten Version der Zielsoftware sowie insbesondere mit der Erstellung des Wizzards, als erster Entwicklerschnittstelle begann das eigentlich kollaborative Prototyping. Während tatsächlicher produktiver Arbeit mit dem Wizzard wurden sowohl dessen Funktionalitäten selbst, als auch die Framework-Instanz im Ganzen ständig an die kontinuierlich, parallel zur beschleunigten Entdeckung der Gestaltungsspielräume wachsenden Ansprüche angepasst.

Mit dem wachsenden Grad der Verfeinerung, sowohl des wechselseitigen Verständnisses und damit der Kompetenzen im Umgang mit dem interaktiven Prototypen als auch des entstehenden Systems selbst, stellte sich für die Redakteure der Wizzard zunehmend als zu unflexibel heraus. Die feste Abbildung zwischen den Ergebnissen der rein fachsystematisch gliedernden Arbeit am Wizzard und der eigentlichen Präsentation in

der Nutzersicht machte immer wieder Eingriffe dezidiert durch die Technikentwicklung nötig, die sich die Autoren bzw. Redakteure auf der Grundlage der gewonnenen Kompetenzen selbst zutrauten – entsprechende Einrichtungen zur Ausführung dieser Arbeiten vorausgesetzt. So wurde schrittweise der WYSIWYG-Modus als Autorenschnittstelle verfügbar gemacht. Zunächst wurde den Redakteuren eingeräumt, Texte auch direkt innerhalb der Sicht zu editieren, die schließlich dem Nutzer präsentiert werden würde. So wurde es erstmalig auf sehr einfache Weise möglich inhaltliche Änderungen unmittelbar während der evaluierenden Nutzung vorzunehmen – auch ohne die permanente Kollaboration zwischen Technikern und Fachautoren. Der Umgang mit dieser Schnittstelle wurde jedoch sukzessive, kollaborativ erarbeitet und die Interaktionsschemata zum Teil im Detail an die Bedürfnisse individueller Entwicklungssituationen angepasst und weiterentwickelt. Im nächsten Schritt sollte es etwa möglich sein, einfache Layout- und Grafik-Gestaltungsmöglichkeiten zu nutzen. Wobei der Ausgangspunkt dieser Arbeit wiederum die fest, entlang des initialen Meta-Object-Protokolls generierten Anwendungsteile und Bildschirme sein sollten – ausgestattet mit Inhalten und Funktionalitäten, die sich aus der Arbeit mit dem Wizzard ableiteten.

So wurden einzelne Bildschirmelemente mit der Maus auswählbar und z.B. in der Dimension und der Position durch Mausinteraktionen veränderbar. Auch wenn sich die meisten dieser Interaktionsmuster schließlich in die bekannten Schemata von gängigen Grafik und Layoutprogrammen fügten, sind doch alle entstandenen Bedienfunktionalitäten das Ergebnis eines evolutionären Prozesses, bei dem sich das Verständnis der Anwender in positiver Rückkopplung mit ihren Ansprüchen auf der einen Seite und den unmittelbar realisierten technologischen Möglichkeiten auf der anderen Seite aus der operativen Tätigkeit heraus weiterentwickelte. Wichtig dabei war, die Aufrechterhaltung der Konsistenz zwischen den Sichten und den weiterhin nötigen Abstraktionsniveaus. Das Konzept des singulären Artefakts gewährleistet dabei beispielsweise, dass Änderungen, die im WYSIWYG-Modus vorgenommen wurden – quasi im Sinne vom Round-Trip-Engineering – auch unter der Sicht des Wizzard sichtbar werden, soweit sie für diese Form der Präsentation relevant sind. Von der möglicherweise veränderten farblichen Gestaltung von Elementen, wir dort z.B. weiterhin abstrahiert. Im WYSIWYG-Modus veränderte Medien und Gliederungsstrukturen erscheinen auch im Wizzard entsprechend angepasst.

In letzter Konsequenz war das Ergebnis dieses Prozesses, dass zumindest die engagiertesten der Fachentwickler tatsächlich bereit und in der Lage waren, auch Details der Parametrisierung von Basisobjekten zu nutzen, um Konzepte und Konstrukte der Bedienung und Navigation selbstständig zu ergänzen oder grundlegend zu verändern. Durch den Transfer und die Rekombination der elementaren Funktionalitäten Basisobjekte ihrer über die Parametrisierung möglichen Ausprägungen waren sie in der Lage neue Interaktionsformen zu schaffen und neue Gestaltungsspielräume für mediendidaktische Konzepte ergründen. Das zentrale Ergebnis dessen ist Konstituierung der interaktiven Arbeitsblätter zur freien Präsentation von eigenen Arbeitsergebnissen als Medium der Vermittlung von Handlungskompetenz im Umgang mit historischen Medien und den Ergebnissen eigener Bewertungen und Interpretationen im Diskurs mit anderen. Dies ist zu großen Teilen das Ergebnis des Transfers der Tätigkeitsschemata

der Ausgestaltung auf die der Nutzung. Der Schritt folgt aus der Einsicht der Redakteure, dass ihre Arbeit am System zumindest im Ansatz die Art historischer Arbeit darstellt, zu der Schüler befähigt werden sollen. So lag es nahe, die Interaktionschemata den Nutzern, also Schülern zu öffnen. Redakteure gestalteten selbständig inhaltliche Vorgaben, die den Schüler dabei entsprechend lenken sollen. So erstellten sie die Konzeption, nach der alle vorgegebenen Quellen blau zu markieren sind (z.B. durch einen Rahmen) und alles, was Schüler interaktiv manipuliere können orange. Nur noch unter einfachen Hinweisen der Technologieentwickler konnte dies unmittelbar umgesetzt, evaluiert und verabschiedet werden. Durch die Manifestation dessen in Stiles und Vorlagen für Meta-Objekt-Beschreibungen – auch weitgehend selbstständig durch die Redakteure umgesetzt – wurde diese Konzeption zum festen Bestandteil des nun genutzten Werkzeugrahmens.

Aber auch die Struktur des Systems und damit die Navigation wurden ständig und zum Teil durch Redakteure selbstständig angepasst und weiterentwickelt. So wurden zum Beispiel Zusatzinformation zu jedem historischen Raum oder auch das Glossar als spezielle Ausprägungen von Panels eingeführt. Dabei ist es die Entwicklungsleistung der Fachentwickler, die den Transfer angestoßen und schließlich realisiert hat – beispielsweise durch die Implementierung der Idee, jedem Raum noch einen Navigationsknoten in der Art eines Themenscreens zuzuordnen. Lediglich kleinere Anpassungen am Content-Grabber-Rahmen und dem Wizzard musste die Technologieentwicklung noch vornehmen.

Der kollaborative Prozess musste dabei nicht aufwändig moderiert werden. Die Eigendynamik des Diskurses führte zum benötigten permanenten Fluss:

- Der kollaborativen Analyse des Ist-Zustandes (ein Entwickler kommt mit der Handhabung eines Wizzards nicht zurecht etc.).
- Davon ausgehend der Ableitung eines Soll-Konzepts wiederum gemeinsam mit den betroffenen Akteuren (Erweiterungen am Werkzeug oder einfach Konsultation oder Übung benötigt)
- und schließlich der gemeinschaftlichen Implementierung dessen.

Dieser Prozess lief integriert in den Entwicklungskontext, also im Einsatzkontext der entstehenden Werkzeuge aber auch in ständiger Rückkopplung zum Einsatz der eigentlichen Zielanwendung im Rahmen der evaluierenden Nutzung. Also alle Entwicklungen waren unmittelbar vom Bedarf der realen Anwendung motiviert und wurden auch in der realen Anwendung direkt evaluiert. Das betrifft die entstehende Technologie in gleichem Maße, wie die Kompetenzen der Nutzer und die umgebende Organisation.

Trotz kleinerer Friktionen, die dem Forschungscharakter des Gesamtprojekts geschuldet waren, bleibt festzustellen, dass Autoren bzw. Redakteure begannen, die für sie konkretisierte Entwicklungsumgebung als alltägliche Arbeitsgrundlage aufzufassen und den Prozess des Erwerbs der Kompetenzen zu deren Beherrschung selbstständig voranzutreiben. Das veränderte Tätigkeits- und Rollenverständnis ging einher mit veränderten Notwenigkeiten innerhalb der Organisation der Arbeit. Es wurde offenkundig, dass sich

Schnittstellen zwischen Aufgabenbereichen und damit auch Organisationseinheiten bzw. innerhalb der Abläufe der Arbeit verändern. Wobei diese Änderungen wichtigerweise mit den strategischen Interessen des Managements der beteiligten Unternehmen zu harmonisieren sind. Im Falle des vorgestellten Projekts etwa bedurfte es schließlich einiger Diskussion, um alle Beteiligten davon zu überzeugen, dass die vermeintlich bequeme Kapselung der Aufgaben – insbesondere der technischen – hinter juristisch belastbare Spezifikationen in vielen Fällen nicht hinreichend ist um inhaltlich belastbare Ergebnisse zu erhalten. Schließlich war aber einleuchtend, dass es erhebliche Vorzüge hat, wenn lange Phasen der Entwicklung vermieden werden, in denen verantwortliche Teile des Teams keinen Einblick in die Aktivitäten erhalten. Zu dem Preis, dass die ursprünglich ausgelagerte Verantwortung zum Teil zurückkehrte in den für das Projekt verantwortlichen Verlag und alle Beteiligten bereit sein mussten, sich den co-adaptiven Prozess zu unterwerfen. Redakteure mussten sich darauf einlassen, dass eine (wenn auch weitgehend auf sie ausgerichtete) informationstechnologische Entwicklungsumgebung ihre Arbeit bestimmen würde – sie gewohnte Tätigkeiten ändern und Kompetenzen erwerben mussten. Umgekehrt war es eine große Herausforderung für die originären Entwickler der Technologie, sich auf die Bedürfnisse und Bedarfe der Fachexperten so einzulassen, dass sie in der Lage waren, mit der Ausprägung der initialen Prototyping-Umgebung eine Grundlage zu schaffen, mit der die Fachexperten bereit sein würden zu arbeiten. Auffällig dabei waren die Schwierigkeiten der Technologieentwickler, sich den zum Teil sehr elementaren Wünschen der Anwender zu beugen – nicht um jeden Preis das technologisch Machbare durchzusetzen.

Insbesondere die Art der Kommunikation, der unmittelbare bewusst geführte Diskurs verlangte neben großer Disziplin auch ein grundsätzlich neues Verständnis für die Projektkommunikation. Es wurde ganz gezielt eine gemeinsame Sprache entwickelt. Der „Raum“, das „Schwerpunktthema“ oder auch der „Wizzard“ (häufig „Wizz“ genannt) waren Konzepte auf die man sich fest einigen konnte, Konzepte die eine Vielzahl technischer Dinge und inhaltlicher Zusammenhänge implizieren und dabei dennoch jedem klar war, was drunter zu verstehen sei.

Im Sinne der Integration der Entwicklung der Technologie und der Organisation ist es so gelungen, einen kontinuierlichen, im Sinne der Ziele konvergenten Prozess der gemeinschaftlichen Entwicklung an einem System unter ständiger unmittelbarer Rückkopplung zu dessen Anwendung im produktiven Einsatz in Gang zu halten.

5. Zusammenfassung, offene Fragen und Ausblick

In den nun folgenden Abschnitten werden die Ergebnisse der Arbeit zusammengefasst. Die gefundenen Lösungen werden im Zusammenhang mit den Beobachtungen während der Evaluation und schließlich auch bezüglich der Perspektiven der Weiterentwicklung und der Verwertung diskutiert. Neben dem Grad der Innovation und dem daraus abzuleitenden wirtschaftlichen wie wissenschaftlichen Potential werden auch die Defizite des Ansatzes und offen gebliebene, zukünftig zu klärende Forschungsfragen erörtert.

5.1. Die Ergebnisse

Es existiert eine Vielzahl an Vorgehensmodellen, Methoden und Werkzeugumgebungen zur Unterstützung der Entwicklung interaktiver, direkt reaktiver Software-Systeme. Die zu Beginn der vorliegenden Arbeit präsentierte Analyse macht jedoch sowohl die mangelnde Durchgängigkeit von Vorgehensweisen, Modellbildungen und Werkzeugen deutlich, als auch die häufig unterentwickelte Verzahnung von Systementwicklung, fachlich inhaltlicher Expertise und Nutzung. Dies muss jedoch als entscheidende Voraussetzung für die erfolgreiche Entwicklung nutzer- bzw. domänengerechter interaktiver Software betrachtet werden.

Der vorgelegte Ansatz macht sich zur Lösung dieser Problematik, die konzeptionell weidlich diskutierte, in der Praxis der hier betrachteten Entwicklungsprojekte kaum beachtete Idee der durchgängigen Nutzerpartizipation zu Nutze. Als Innovation kann dabei gesehen werden, dass designierte Anwender und domänensachverständige Nutzervertreter (in der Regel Fachautoren) nicht allein in der Rolle analysierter Objekte in den Entwicklungsprozess einbezogen werden, sondern aktiv gestaltend als Subjekte in allen Phasen der Entwicklung mitwirken. Ermöglicht wird dies durch die gezielt forcierte Konvergenz der Arbeitsweisen der einzelnen Disziplinen und die wechselseitige Anpassung – die Co-Adaption – zwischen den Werkzeugen und Tätigkeiten bzw. den Kompetenzen im Umgang damit. Das Ergebnis sind kontext- und aufgabengerechte Abläufe und Werkzeuge, die die daraus erwachsenden individuellen Anforderungen dezidiert unterstützen. Die angestrebten Prozesse der wechselseitigen Anpassung implizieren die kontinuierliche Umgestaltung des organisatorischen Umfeldes des Einsatzes. Ein bewusster Prozess der Organisations- und der Personalentwicklung verläuft dabei

integriert in die Abläufe der Ausgestaltung des Zielsystems und der begleitenden (evaluiernden) Anwendung von Zwischenergebnissen in realen Produktivszenarien der Fachdomäne. Um das zu ermöglichen, lässt sich die Kongruenz zwischen den Tätigkeiten der fachlichen Systemausgestaltung und denen der angestrebten Anwendung ausnutzen.

Um den evolutionären, sich weitgehend selbst organisierenden Gesamtprozess in Richtung globaler Ziele steuern und effektiv gestalten zu können, bedarf es einerseits einer instrumentellen Grundlage – eines technologischen Rahmens, der die Artefaktzentrierung sicherstellt. Andererseits wird eine Methodik vorgeschlagen, die die Anwendung des Werkzeug- bzw. Infrastrukturrahmens anleitet und den diskursiven Prozess der Konvergenz der Disziplinen und der kollaborativen Erarbeitung emergenter Lösungen moderiert.

5.1.1. Die instrumentelle Basis: Das Framework

Als ein Ergebnis der Arbeit entstand ein produktiv einsetzbares Software-Framework[1]. Dieses zunächst abstrakte Software-Gerüst realisiert die vollständige Software-Infrastruktur einer interaktiven Anwendung mit allen zentralen, für den Ablauf notwendigen Kontrollstrukturen. Für die verschiedenen Aspekte des Systems gibt es jeweils Rahmen mit Funktionalitäten und Schnittstellen als Ansatzpunkte für die Instanziierung des Systems. Innerhalb dieser Rahmen sind beispielsweise Aspekte der Infrastruktur – also etwa der Vernetzung und der Persistenz – für die weitere bedarfsabhängige Ausprägung angelegt. Es gibt aber auch vorgefertigte, anpassbare Komponenten, mit deren Hilfe immer wiederkehrende Anforderungen innerhalb der Domäne der interaktiven, direkt reaktiven Anwendungen schnell bedient werden können. Die einzelnen Rahmen werden im Zuge eines iterativen, evolutionären Entwicklungsprozesses schrittweise zur Zielanwendung konkretisiert. Das Potential des Frameworks für den evolutionären Ausbau bildet dabei die Grundlage für die Vorgehensweise, die den iterativen und evolutionären Charakter als Träger für einen kontinuierlichen diskursiven Prozess der Annäherung nutzt – der Annäherung zwischen den Disziplinen und insbesondere zwischen der Anwendungsdomäne und der potentiellen Technologieunterstützung.

Die Besonderheit dabei ist, dass das von jeglicher Anwendung zunächst unabhängige Framework (als „Metawerkzeug mit Metabausteinen“) über einen Zwischenschritt zu einem domänenspezifischen bzw. aufgabenspezifischen Entwicklungs-Framework konkretisiert wird. Diese, bezüglich der Architektur, der Interaktionskomponenten, der Navigationsstrukturen und wichtigerweise der Entwicklerschnittstellen, domänenspezifische Prototyping-Umgebung, bildet schließlich die Grundlage für eine fachlich zentrierte Gestaltungsarbeit. Die in der Evaluation nachgewiesene Perspektive auf die Möglichkeit der Fachexperten bzw. sogar Nutzer, ihre Zielsoftware individuell, mit ihnen vertrauten, praxisgerechten Hilfsmitteln zu gestalten, stellt sich dabei als eine der zentralen Innovationen dar. Als entscheidend erweißt sich insbesondere, dass auch technologisch abstrahierende, universelle Modellbildungen zu Gunsten von Konzeptualisierungen der

[1] Die Implementierung erfolgte im Rahmen des Forschungsprojekts *LeMOLernen* am Fraunhofer ISST, dankenswerterweise unterstützt durch ein Team von Software-Entwicklern.

Fachdomäne abgelöst werden. Also auch die Beschreibung von Aufbau- und Ablaufstrukturen, die in den Phasen der Planung und der Konzeption bearbeitet werden, orientieren sich am Verständnis (der Metaphorik und den Ordnungsprinzipien, etc.) der Zieldomäne. Die konzeptionelle Arbeit kann durch dezidierte Schnittstellen unterstützt werden. Während der Evaluation etwa entstand ein Werkzeug, welches zunächst nur die Gliederung und damit die grobe Anlage der Navigationsstruktur in einer Weise unterstützte, die es auch Historikern und Fachredakteuren ohne technischen Hintergrund erlaubte diese Entwicklungsaufgaben zu erledigen – die entsprechenden Potentiale für eine Effektivierung ihrer angestammten Tätigkeiten zu erkennen und zu nutzen.

Der Preis für die Entstehung praxisgerechter Software unter engster Mitwirkung von Nutzern und Fachleuten bzw. sogar die Unabhängigkeit der Fachentwickler ist die diskursive inkrementelle Adaption der einzelnen Rahmen des Frameworks:

- Die Auswahl der Architektur (aus einem festen Spektrum an Varianten),
- Die Implementierung bildschirmgestalterischer Vorgaben,
- Die Auswahl und die Erweiterung der im Framework angelegten Interaktionsbausteine (durch die Implementierung bestehender Klassen und Schnittstellen, die Aggregation und die Konfiguration etc.),
- Die Vorgabe von Möglichkeiten der Strukturierung und der Navigation sowie schließlich
- Die Anlage der Schnittstellen der Bearbeitung.

Dieser Preis ist sicherlich nicht unter allen Umständen gerechtfertigt. Die Investition dieser Vorarbeiten, die auch immer wieder zu Anpassungsaufwand und Overhead führen können, setzt voraus, dass die entstandenen domänenspezifischen Werkzeuge wiederholt produktiv eingesetzt werden können. Das ist charakteristisch für den Einsatz von Frameworks im Allgemeinen. Kompensiert wird der Aufwand der initialen technologischen Adaption im Wesentlichen durch die gewonnene Flexibilität des abstrakten Frameworks. Durch den Einsatz des Reflection-Mechanismus werden Änderungen direkt, interaktiv am „lebenden“ Entwicklungssystem und damit innerhalb des diskursiven Prozesses des Prototyping möglich. Build-Prozesse entfallen zum großen Teil.

Die Wiederverwendung entfaltet ihre Wirkung auf mehreren Ebenen. Die wiederholte Verwertung der gleichen Entwicklungsleitung führt nach der Amortisierung der für die Wiederverwendbarkeit aufgewendeten Mehrkosten zu den nahe liegenden wirtschaftlichen Effekten. Auch die Qualität der vom Framework abgeleiteten Produkte profitiert von der wiederholten Verwendung und den damit einhergehenden Pflege- und Weiterentwicklungsmaßnahmen – sowohl an den abgeleiteten Werkzeugen und Bausteinen, als auch am Framework selbst.

Neben der Wirtschaftlichkeit durch die Realisierung von Synergien über den wiederholten Einsatz von Entwicklungsergebnissen, schafft die Wiederverwendung dabei auch das entscheiden Vehikel um technologische Aufwände nachhaltig zu kapseln und die fachliche Arbeit perspektivisch in den Vordergrund rücken zu können, diese sukzessive

unabhängig zu machen von technischen Abstraktionen. Diese Selbständigkeit erspart den Aufwand für Transferleistungen zwischen den Arbeitspraktiken und Sichtweisen innerhalb der Gestaltung zugunsten frei werdender Ressourcen für kreative, fachlich inhaltliche Arbeit.

Als Motivation zur Kollaboration, also als Ausgangspunkt für den Prozess der Co-Adaption dient die direkt manipulative, gestalterische Arbeit am gemeinsamen Material des zentralen Zielsystems. Diese unterstützt das Framework technisch durch die Möglichkeit, die Entwicklerzugänge zur Arbeit über das Web zu verteilen. Die Entwicklungsumgebung realisiert damit eine spezielle Art einer Kollaborationsplattform.

Das Framework und seine Wirkungsweise als eine Säule der vorgelegten Lösung wurde zentral im Rahmen der „Mensch und Computer“ -Konferenz der Fachgemeinde zur Diskussion gestellt [KFK05]. Darüber hinaus bestätigt die Resonanz auf Vorträge, Demonstrationen und Präsentationen auf Messen und in Workshops (etwa [Kar03] oder [Kar05]) einerseits den Bedarf vieler Fachdomänen nach stärkerer Einbeziehung in die Prozesse der Entwicklung informationstechnologischer Unterstützung oder gar den Wunsch nach Autonomie bei der Gestaltung medialer Informationssysteme. Die demonstrierte Lösung fand unabhängig vom tatsächlichen Grad der technischen Reife Interesse, sowohl bei der Zielgruppe für den industriellen Einsatz, als auch bei der Forschungsgemeinde. Bezeichnend beispielsweise war auch, dass bereits die Konkretisierung für den Kontext historischer Lernsoftware als potentielles Produkt an sich reges Interesse hervorrief.

5.1.2. Die Methodik: Interaktives kooperatives Prototyping

Das Framework entfaltet seine Wirkung jedoch nicht, ohne ein Modell, also eine Vorgabe zu seiner Anwendung und Weiterentwicklung. Die Idee der Konvergenz zwischen den Arbeitsweisen der Disziplin und insbesondere die Anpassung der Werkzeuge an die Bedürfnisse des fachlich orientierten Nutzers versucht der vorgestellte Ansatz über einen iterativen, evolutionären Prozess zu realisieren. Das wechselseitige Verständnis, für die benötigte Unterstützung auf der einen Seite und die tatsächlich gegebenen technologischen Potentiale zu deren Realisierung auf der anderen, wird in der Bearbeitung des „gemeinsamen Materials“ der Zielsoftware erarbeitet. In der interaktiven, direkt manipulativen und vor allem kollaborativen Auseinandersetzung mit dem gemeinsamen Entwicklungsgegenstand entsteht der Diskurs, der den für die Konvergenz der Disziplinen benötigten Austausch von Wissen und Kompetenzen ermöglicht. So wird es etwa durch Nachahmung möglich, einzelne, zum Teil unbewusst verrichtete Tätigkeiten zu vermitteln.

Den Ausgangspunkt für die eigentliche prototypenbasierte Entwicklungsarbeit bilden jedoch auch hier rein konzeptionelle Phasen der Erkundung – vordergründig der Anwendungsdomäne, ihrer aktuellen Situation und ihrer Zielvorstellungen. Primär durch die ethnographische Untersuchung des Einsatz- und des Produktionskontexts der Domäne wird ergründet, welche Aufgabe durch welche Tätigkeiten realisiert werden und welche Gegenstände (Inhalte, Medien, Dokumente, etc.) dabei Objekte der Bearbeitung sind.

Das Ergebnis dieser teilnehmenden Untersuchung des Zielkontexts wird nicht in die üblichen modellzentrierten Spezifikationen übertragen. Es entstehen vielmehr unmittelbar erste sehr einfache Prototypen, ggf. noch ohne direkte Materialisierung durch Informationstechnologie. Diese so genannten Mock-Ups können mit Hilfe simpler Papier-Skizzen realisiert sein. Wichtig ist lediglich, dass ad hoc und direkt manipulativ in einer Gruppe Aspekte des Aufbaus und des Ablauf der zu entwickelnden Prototyping-Umgebung erarbeiten werden können. Auch dabei ist die Einbeziehung aller Betroffen bereits entscheidend. Wichtigerweise sollen die nicht technisch orientierten Beitragenden (also auch Nutzer) die folgenden Aspekte im experimentellen Einsatz in moderierten Sitzungen konkretisieren und fixieren:

- Den grundsätzlichen Aufbau des Systems (verteilt, stand-alone, etc.) und die Struktur des Produkts (DVD, individuelle Installation, oder Web-Site),
- Vorgaben für zentrale Abläufe und Navigationsstrukturen,
- Elementare Interaktionselemente und ihre Ausprägungen (auch etwa in Aggregaten)
- Das grundsätzliche Erscheinungsbild der Nutzerschnittstellen (Skizzen statischer Strukturen des Aufbaus von Bildschirmen und Dialogen, etc.)
- Szenarien für die Organisation des Einsatzkontextes – etwa Ideen für die Einbettung in den Unterricht oder allgemein die durch das System unterstützten Arbeitsabläufe auch während der Entwicklung.

Im nächsten Schritt werden diese Konzepte mit den Mitteln des Frameworks zu einer initialen Prototyping-Umgebung umgesetzt: die Prototyping Umgebung wird durch Framework-Experten in der Zielkonfiguration ablauffähig installiert. Die Vorgaben bezüglich der Ablauf- bzw. der Navigationsstruktur werden in einem Schablonenartigen Grundgerüst der Anwendung angelegt. Diese ist dann bereits ablauffähig und besitzt alle grundsätzlichen Anlagen einer interaktiven Anwendung. Die Ausgewählten Interaktionselemente werden, wenn sie bereits in einer abstrakten Form Teil des Frameworks sind, ausgewählt, angepasst und zusammengestellt. Gegebenenfalls neue, benötigte Elemente werden entwickelt und in das Framework integriert. Das Ergebnis dessen ist der initiale individualisierte Baukasten zur fachlichen Gestaltung interaktiver Teilanwendungen. Außerdem werden nun spezifische Entwicklerschnittstellen ausgeprägt, über die schließlich die Fachexperten das Gerüst der Anwendung mit „Leben“ füllen, indem sie aus dem Baukasten schöpfen und Elemente konfigurieren, zu Aggregaten oder Anwendungen zusammenfügen.

Die eigentliche interaktive Prototyping-Arbeit besteht dann in dieser iterativen Konkretisierung und Befüllung des Prototypen hin zum Zielsystem. Der entscheidende Vorzug dieses Ansatzes besteht darin, dass die Entwicklerschnittstellen und die während der Gestaltung disponiblen Elemente weitgehend auf die Abläufe und das Verständnis der Domäne zugeschnitten sind und damit zu großen Teilen denen des Anwendungskontexts entsprechen. So können Fachautoren und Redakteure beinahe in ihren gewohnten Prozessen arbeiten und vertraute Tätigkeiten verrichten. Erst ganz sukzessive in der

kontinuierlichen „haptischen“ Auseinandersetzung mit der Technologie ergründen Sie selbstständig deren Potentiale. Dies wird möglich, da sich all ihre Aktivitäten in ihrer Wirkung direkt am Lauffähigen System überprüfen lassen. Aufgedeckte Defizite werden als direkte Reaktion auf das Auftreten von Problemen, interaktiv behoben. In den frühen Zyklen der Entwicklung wird dies immer wieder unterstützt durch Frameworkspezialisten geschehen. Mit der zunehmenden Erfahrung der Fachentwickler stellt sich aber auch bei der Anpassung der Werkzeuge und der Baussteine die Selbstständigkeit der fachlich orientierten Entwickler ein. Durch die Tätigkeitsorientierung des Ansatzes und das dadurch realisierte direkte Feedback zwischen den Gestaltungspotentialen und den Bedarfen sowie den ständigen Diskurs zwischen den Disziplinen entstehen emergent neue bzw. individuell angepasst Komponenten für Interaktionselemente, Muster der Navigation und vor allem Möglichkeiten zu deren Handhabung während der Entwicklung. Auf diese Weise gemeinschaftlich entwickelte Werkzeuge, Interaktionselemente und insbesondere Aggregate werden wiederum zu einem Teil des gesamten Instrumentatriums, was in der weiteren Entwicklungsarbeit durch eben diese Arbeit unmittelbar und unter Produktivbedingungen auf Ihre Tauglichkeit im Einsatz evaluiert wird. Die Entwicklung des Systems und seine Evaluation im produktiven Nutzungskontext können also tatsächlich integriert ablaufen.

Dieser Prozess der fachlichen Individualisierung der Entwicklungswerkzeuge bzw. der Gestaltungsmittel (disponible Komponenten) und schließlich deren Einsatz in einer produktiven Umgebung wirken zwangsläufig auf den Einsatzkontext zurück. Der wechselseitige Charakter dieser Konvergenz wird dadurch kanalisiert, dass mit der fortschreitenden Entwicklung der Technologie auch gezielt ihr Einsatz organisiert und geplant wird. Im Rahmen kontinuierlicher Organisationsentwicklung werden integriert mit der Technologieentwicklung veränderte Abläufe, Aufgabenbereiche und damit Rollenverständnisse und Kompetenzen an die durch den Technologieeinsatz veränderten Arbeitsverhältnisse angepasst. Dieser moderierte aber sich ebenfalls auf die Emergenz von Lösungen innerhalb der diskursiven Auseinandersetzung mit dem Entwicklungsgegenständen stützende Prozess wurde unter anderem in [KFK06] in der Community der Forschung zu tätigkeitsorientierten Prozessen der Wissensorganisation zur Diskussion gestellt. Insbesondere aber auch der Erfolg der als Fallstudie betrachteten Produktentwicklung (im Wesentlichen dokumentiert und publiziert in [CKV05].) verleiht der Tragfähigkeit des Ansatzes Evidenz. Nachweislich gelang es dort, technologieferne Historiker an den Prozess der unmittelbaren Ausgestaltung eines Systems heranzuführen und auch an der Gestaltung der Mittel zur Unterstützung dieser Arbeit mitzuwirken. Die Zufriedenheit mit den selbst erzielten Entwicklungsergebnissen wurde dadurch unabhängig von kommunikativen Brüchen innerhalb des Prozesses. Die Konformität mit den aus den eigenen (domäneninhärenten) Bedarfen bzw. Bedürfnissen abgeleiteten Anforderungen und der Umsetzung ist damit zwangsläufig verbessert worden. Auch die limitierenden Einschränkungen durch trotz Adaption nicht realisierbare Gestaltungspotentiale, wurden kompensiert durch die permanente unmittelbare Einsicht in die Möglichkeiten und grundlegenden Limitierungen des Instrumentariums.

5.2. Offene Probleme und Fragestellungen

Über die Plausibilität der konzeptionellen Universalität des Frameworks und der Vorgehensweise hinaus wurde bisher kein endgültiger Nachweis dazu geführt, inwieweit sich der Ansatz auf andere Entwicklungskontexte übertragen lässt. Offenkundig ist, dass der Einsatz für alle Entwicklungsbemühungen von einem der Evaluations-Software vergleichbaren Grad an Interaktivität und Inhalte-Aufkommen möglich ist. Dies lässt sich insbesondere für Fächer der Geisteswissenschaft verallgemeinern. Weichen der Medieneinsatz und etwa Arbeitstechniken stark von dem dort üblichen ab, besteht die Gefahr, dass der Anpassungsaufwand gerade an den elementaren Interaktionsbausteinen und damit auch der benötigte Beitrag der Technologieentwickler zu groß werden. Tatsächliche interaktive Simulationen komplexer technischer Zusammenhänge etwa, werden sich aus den vorhandenen Mitteln kaum kombinieren lassen. Eine Anwendung auf der Grundlage des Frameworks kann dann lediglich noch Rahmen für die Präsentation oder die Gliederung bieten. Komplexe, in sich geschlossen Flash-Anwendungen können zumindest als elementare, disponible Elemente unabhängig von der Problematik einfach integriert werden.

Einige der Annahmen, die der vorgestellte Ansatz bezüglich der Rahmenbedingungen macht, können sich in der Allgemeinheit als problematisch erweisen. Es ist möglicherweise nicht in jedem Fall davon auszugehen, dass das Management aller Unternehmen die potentiell nötigen Veränderungen in der Organisation und im ggf. sogar standardisierten Ablauf von Entwicklungsprojekten akzeptieren wird. So richtet sich der Ansatz pauschal an Unternehmen, die sich auch für den implizierten Strategiewechsel interessieren. Denn gerade übertragen auf den Verlagskontext bedeutet der Einsatz der Lösung in gewissem Maße ein Insourcing von Aufgaben und Verantwortlichkeiten. Synergien durch Wiederverwendung und Nachhaltigkeit sowie die gewonnen Einfluss- und Kontrollmöglichkeit über den gesamten Prozess müssen mögliche wirtschaftliche Nachteile aufwiegen. Auch die Akzeptanz bei jedem einzelnen Entwicklungsbeteiligten ist für den allgemeinen Fall des Einsatzes nicht endgültig zu garantieren. Die Behutsamkeit (beinahe Empathie) mit der das Vorgehensmodell auf die Eigenheiten der Anwendungsdomäne einzugehen versucht, ist dabei etwa kein Garant dafür, dass tiefer sitzende Berührungsängste im direkten Umgang mit der Technologie ausgeräumt werden. Auch menschliche Konflikte innerhalb des Teams können nie völlig ausgeschlossen werden. An dieser Stelle offenbart sich die Achillesverse des Ansatzes: ist es beispielsweise auf Grund von Verwerfungen zwischen Teamgliedern nicht möglich, den tätigkeitsorientierten Diskurs und damit die evolutionäre sowie intrinsisch motiviert explorative Entwicklung in Gang zu setzen und zu halten, wird es sehr schwierig, die eigentlichen Potentiale des Modells auszunutzen.

Der beim Extreme Programming [Bec00] häufig kritisierte Anspruch, Anwender kontinuierlich im Entwicklungskontext zur Klärung aller fachlichen Fragen zu Verfügung zu haben, bietet auch hier einen Ansatzpunkt für Kritik. Gerade die enge Zusammenarbeit und die kontinuierliche Präsenz lassen sich sicher nicht in allen Fällen wirtschaftlich rechtfertigen und organisieren.

Der Einsatz eines Frameworks und die Reduktion der Komplexität der Entwicklung durch Wiederverwendung und Spezialisierung auf Aufgaben zieht immer die Entscheidung für gewisse Vorgaben und die Aufgabe universeller Flexibilität nach sich. So ist selbst für das zunächst allgemeingültige Framework die Einschränkung bezüglich der Einsetzbarkeit auf die hier adressierte Domäne, die interaktiven direkt reaktiven Systeme zu machen. Auch dabei können sicher nicht alle Anwendungsfälle und Produktdimensionen bedient werden. So ist die Anlage der Architektur und insbesondere die Technologie zu deren Realisierung auf das Spektrum der beschriebenen Möglichkeiten beschränkt. Die Umsetzung der Client-Anwendung auf der Basis eines anderen Toolkits zur Generierung von Nutzerschnittstellen (etwa Java-Swing) anstelle des durch das Framework vorgegebenen Flash-Clients oder auch die Ablösung des HTTP- bzw. SOAP-Protokolls ist beispielsweise nicht sinnvoll möglich. Das würde einer Neuimplementierung zentraler Teile des Frameworks entsprechen.

Die Anlage und Anwendung allgemeiner, anpassbarer Systeme induziert darüber hinaus naheliegenderweise in der Regel einen entsprechenden Ressourcenmehraufwand. Die Anwendung des Reflection-Patterns zur Flexibilisierung der Software für Änderungen auch zur Laufzeit beispielsweise, verbraucht neben den Ressourcen für den eigentlichen Ablauf des Systems zusätzliche Systemressourcen für das Lesen der Metalevel-Objekt-Beschreibungen, und deren Umsetzung in die eigentlichen Basisobjekte. Die Anwendung und die Pflege des Frameworks binden über dies Mittel und Entwicklungszeit. Die theoretischen Grenzen der Skalierbarkeit bezüglich der Größe des Projekts und damit in der Regel des angestrebten Produkts sind jedoch bisher nicht überprüft. Für die *einmalige* Entwicklung eines einfachen Kiosksystems, was statische Informationen auf einer einfachen Folge von Seiten präsentieren soll, lohnt sich die Adaption einer Entwicklungsumgebung trivialerweise nicht. Umgekehrt wurde bisher nicht untersucht, ob in großen, unter Umständen sogar verteilt arbeitenden Teams die Idee mit der singulären Artefaktzentrierung bestand haben kann. Dabei wäre zu betrachten, wie sich neben den technologischen Auswirkungen dieser Skalierung die Projektorganisation in entsprechenden Dimensionen noch gestalten lässt. Der Aufwand für die Moderation und die Koordination der Kollaboration auf der Basis der Webzugänge zur Prototyping-Umgebung könnte den Rahmen des sowohl praktisch als auch des wirtschaftlich Sinnvollen sprengen. Eine technologische Unterstützung solcher Prozesse ist momentan nicht realisiert.

5.3. Ausblick

5.3.1. Technologische und wissenschaftliche Anschlussfähigkeit

Einige der Ideen zur Adaptierbarkeit der Software sind in der entstandenen Umsetzung des Frameworks zunächst nur skizzenhaft angelegt, um die Prinzipien („im vertikalen Durchstich“) zu erproben. Der Ansatz verlangt jedoch in der Perspektive durchgängig intuitivere Möglichkeiten der Individualisierung. Weiterentwicklungspotential besitzen die allgemeingültigen Framework-Schnittstellen insbesondere bei der Handhabbarkeit der Anpassung der Screen-Designrahmen.

In der vorliegenden Version des Frameworks müssen alle Medien über das File-System der Entwicklungsumgebung zugänglich gemacht werden, um während der Gestaltungsarbeit innerhalb disponibler Elemente genutzt werden zu können. Wünschenswert wäre jedoch eine Integration heterogener Medien-Beständen, wie den Datenbanken von Museen und Archiven, wie es beispielsweise in [KW02] angedacht ist. Mit der fortschreitenden Standardisierung, also der zunehmenden Verwendung von Metadaten (LOM[2] oder SCORM[3]) und auf der Grundlage der serviceorientierten Anlage der Framework-Architektur, könnten Prozesse der inhaltlichen Recherche und des fachlichen bzw. didaktischen Entwurfs besser integriert werden. Damit würden auch Ideen echter Adaptivität umsetzbar, mit deren Hilfe das System aktiv auf der Grundlage von Regeln auf die fachlich inhaltliche Bedürfnisse eingehen kann – Strukturen, Abläufe und Inhalte sich (teil-)automatisiert ableiten lassen aus etwa den Lern- und Vermittlungszielen bzw. den Voraussetzungen beim Rezipienten (wie etwa in [CHK] oder [KW02]).

Eine Aufgabe für zukünftige Forschungsaktivitäten besteht ferner in der konkreteren Untersuchung der möglichen Einbettung von Ansätzen des Wissensmanagements in die Entwicklungsabläufe. Insbesondere bedarf es der Konzeption und der Implementierung praktikabler Konzepte der technologischen Unterstützung der Prozesse der Kollaboration und des Wissensmanagements über das Bereitstellen eines zentralen Artefakts als gemeinsames Material (etwa über Webschnittstellen) hinaus. Zur Flankierung des Moderationsprozesses ist etwa der Einsatz entsprechender Mitteln des personalisierenden Wissensmanagements denkbar. Die dort propagierten Infrastrukturen, etwa zur Echtzeitkommunikation oder auch zum asynchronen dokumentenbezogenen Austausch in Foren oder Wikis müssten möglichst eng mit den eigentlichen Entwicklerschnittstellen verwoben werden, um die Bindung des kollaborativen Austauschs an die Tätigkeit auch über verteilte Arbeitsplätze hinweg aufrecht zu erhalten. Eine Idee besteht etwa darin, Infrastrukturen, die aufgabenbezogen, entlang der zu bewältigenden Arbeitsprozess die so genannte „Wissenskoproduktion“ unterstützen [FKS05], mit der Entwicklungsinfrastruktur des Frameworks zu verweben. Damit wäre es dann etwa möglich, individuell auf auftretende Probleme und Fragen eingehen zu können, ohne tatsächlich in Präsenzsitzungen gemeinsam zu arbeiten. Würde dann ein Fachentwickler einen Punkt erreichen, an dem er die aktuelle Tätigkeit der Gestaltung bzw. der Nutzung nicht optimal unterstützt bekommt, würden passend zur Art des Problems und der Tätigkeit sowie zum aktuellen Kommunikationsbedarf des Fachentwicklers Wissensressourcen, einschließlich der Möglichkeit von Live-Konsultationen mit Coaches und Moderatoren, zur Verfügung gestellt.

5.3.2. Wirtschaftliche Anschlussfähigkeit

Die veränderten Rollenverständnisse und die im Rahmen der Co-Adaption ausgeprägten neuen Kompetenzen und individuellen Produktionsmittel (die domänenangepassten Werk-

[2] **L**earing **O**bject **M**etadata Standard (`http://ieeeltsc.org/wg12LOM` gesehen am 13. November 2006)
[3] **S**harable **C**ontent **O**bject **R**eference **M**odel (`http://www.adlnet.gov/scorm/index.cfm` gesehen am 13. November 2006)

zeuge) induzieren das Potential neuer Geschäfts- und Betriebsmodelle. So erarbeiteten die Industrievertreter[4] innerhalb des Konsortiums einer Initiative zu einem Forschungsprojekt die Vision, auf der Grundlage eines Ansatzes, wie dem in dieser Arbeit präsentierten, die Geschäftsabläufe im Rahmen ihrer Auftragnehmer-Auftraggeber-Beziehung nachhaltig harmonisieren zu können.

So sahen Vertreter des Ernst Klett Verlags als potentielle Auftraggeber für Entwicklungsdienstleistungen das Potential, eine kontinuierliche, unmittelbare Einflussmöglichkeit auf das im Sinne der wirtschaftlichen und programmatischen Zielsetzungen des Verlages entstehende Produkt zu erhalten. Das mit der Isolation von Entwicklungsphasen verbundene Risiko der irreversiblen Manifestation von missverstandenen Anforderungen in einem schließlich nicht praxisgerechten Zielsystem, würde mit der Vorgehensweise überwunden werden. Die durchgängige Möglichkeit flexibler Reaktion auf Veränderungen in der Domäne durch die unmittelbare Anpassungen am System würde die Kosten für Änderungen im Sinne der Idee von Beck [Bec00] konstant halten – im Gegensatz dazu, dass sich die komplette Investition in ein umfangreiches, aber auf Missverständnissen basierendes und damit unbrauchbares System als Fehler herausstellt.

Umgekehrt erhalten die Auftragnehmer die Möglichkeit, ein sehr viel klareres Verständnis von der Anforderungen der Domäne zu entwickeln und auch deutlicher die Restriktionen sowie Potentiale des Technologieeinsatzes zu vermitteln. Aus dem klaren Verständnis für die Anforderungen lassen sich Aufgabenstellungen genauer ableiten und damit Aufwandsabschätzungen präziser vornehmen. Das Risiko unkalkulierbarer Kosten und unzufriedener Kunden wird gleichermaßen minimiert. Von der Reproduzierbarkeit des Entwicklungszyklus durch den wiederholten Einsatz des Frameworks und der Methode seiner Nutzung profitieren alle Beteiligten gleichermaßen. Synergien ergeben sich vor allem aus der Routine im Umgang mit den Entwicklungswerkzeugen und der wiederholten Verwendung von Entwicklungsergebnissen.

Ein verändertes Produktverständnis ergibt sich aus der Tatsache, dass die Mittel der Entwicklung selbst domänengerecht sind. Das eröffnet die Möglichkeit, Einrichtungen zur Weiterentwicklung und Individualisierung der Software an den fachlich orientierten Endanwender weiterzugeben. Damit wird es etwa möglich die benötigten Nutzungsmodelle zu unterstützen, nach denen Coaches oder Lehre, die Software auf partikuläre Unterrichtssituationen zuschneiden können. Informationssuchende bzw. Lernende selbst können die Software an ihre individuellen Rezeptions- und Lerngewohnheiten anpassen, ohne sich mit den Details technologischer Aspekte auseinandersetzen zu müssen. Die fachlich orientierte gestalterische Arbeit am System kann ferner zu einem zentralen Nutzungsschema werden. Die Systematisierung von Inhalten und die Ausarbeitung von (interaktiven) Präsentationen fachlicher Arbeitsergebnisse entspricht einer zentralen, immer häufiger zu vermittelnden Handlungskompetenzen auf vielen Fachgebieten.

[4] Die Konsortialpartner (Industrievertreter) bei der Skizzierung eines Forschungsvorhabens zur Etablierung von Prinzipien des End-User-Development in der industriellen Software-Entwicklung:
- Ernst Klett Verlag GmbH
- VR—Fabrik: virtual reality und multimedia GmbH
- KREAKTOR GmbH

Die gewonnene Selbstständigkeit der Fachdomäne, also der Verlage bzw. allgemein der beauftragenden Institution oder gar Endanwender bei vielen Aspekten der Produktausgestaltung, lässt zunächst vermuten, dass den Dienstleistern Wertschöpfungs- und damit Umsatzpotentiale genommen werden. Neben der Tatsache, dass innerhalb des kollaborativen Prozesses des interaktiven Prototyping Technologieentwickler permanent mitwirken müssen und die Anwendung des Frameworks von Dienstleistungen flankiert wird, ergibt sich für die software-entwickelnden Unternehmen die Chance, den Wandel vom reinen Dienstleister zum Anbieter maßgeschneiderter (fach- und Aufgabenspezifischer) Produkte zur Software-Entwicklung zu vollziehen.

A. Einblicke in Details der technischen Umsetzung

Auf den folgenden Seiten verdeutlichen jeweils kommentierte Ausrisse aus der XML-Repräsentation der Meta-Level-Beschreibung die Bedeutung ausgewählter Einträge. Auf die Seite mit der Wiedergabe der XML-Datenstruktur folgt jeweils, wenn es einen direkten Zusammenhang gibt, die vom Renderer erzeugte Funktionalität aus der Sicht des Nutzers. Die Beispiele sind dem Evaluations-System „LeMOLernen“ entnommen.

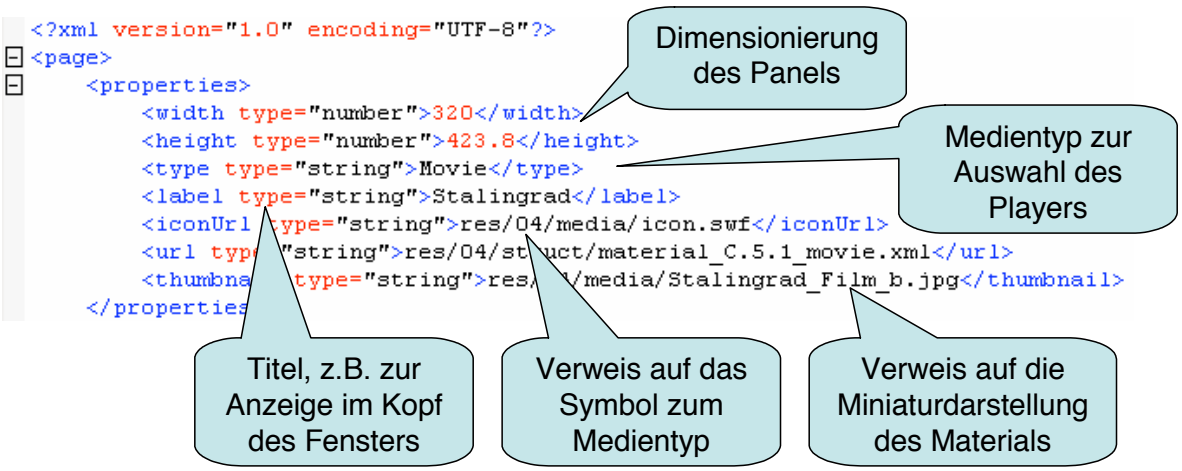
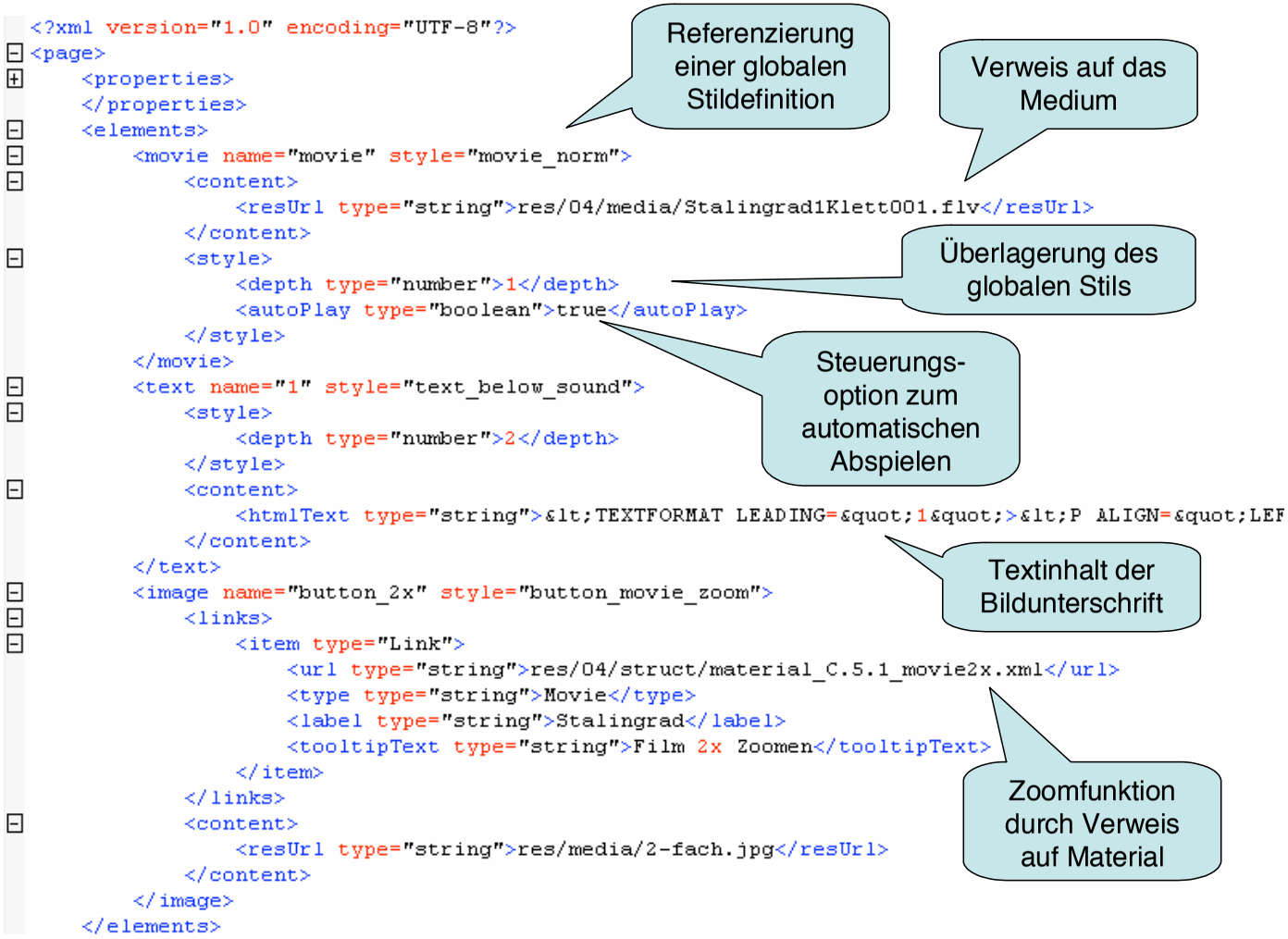


Abbildung A.1.: Kopf einer Page-XML-Instanz




```
<?xml version="1.0" encoding="UTF-8"?>
<page>
    <properties>
    </properties>
    <elements>
        <movie name="movie" style="movie_norm">
            <content>
                <resUrl type="string">res/04/media/Stalingrad1Klett001.flv</resUrl>
            </content>
            <style>
                <depth type="number">1</depth>
                <autoPlay type="boolean">true</autoPlay>
            </style>
        </movie>
        <text name="1" style="text_below_sound">
            <style>
                <depth type="number">2</depth>
            </style>
            <content>
                <htmlText type="string">&lt;TEXTFORMAT LEADING=&quot;1&quot;&gt;&lt;P ALIGN=&quot;LEF
            </content>
        </text>
        <image name="button_2x" style="button_movie_zoom">
            <links>
                <item type="Link">
                    <url type="string">res/04/struct/material_C.5.1_movie2x.xml</url>
                    <type type="string">Movie</type>
                    <label type="string">Stalingrad</label>
                    <tooltipText type="string">Film 2x Zoomen</tooltipText>
                </item>
            </links>
            <content>
                <resUrl type="string">res/media/2-fach.jpg</resUrl>
            </content>
        </image>
    </elements>
```

Abbildung A.2.: Auszug aus der Page-XML-Instanz eines Video-Panels

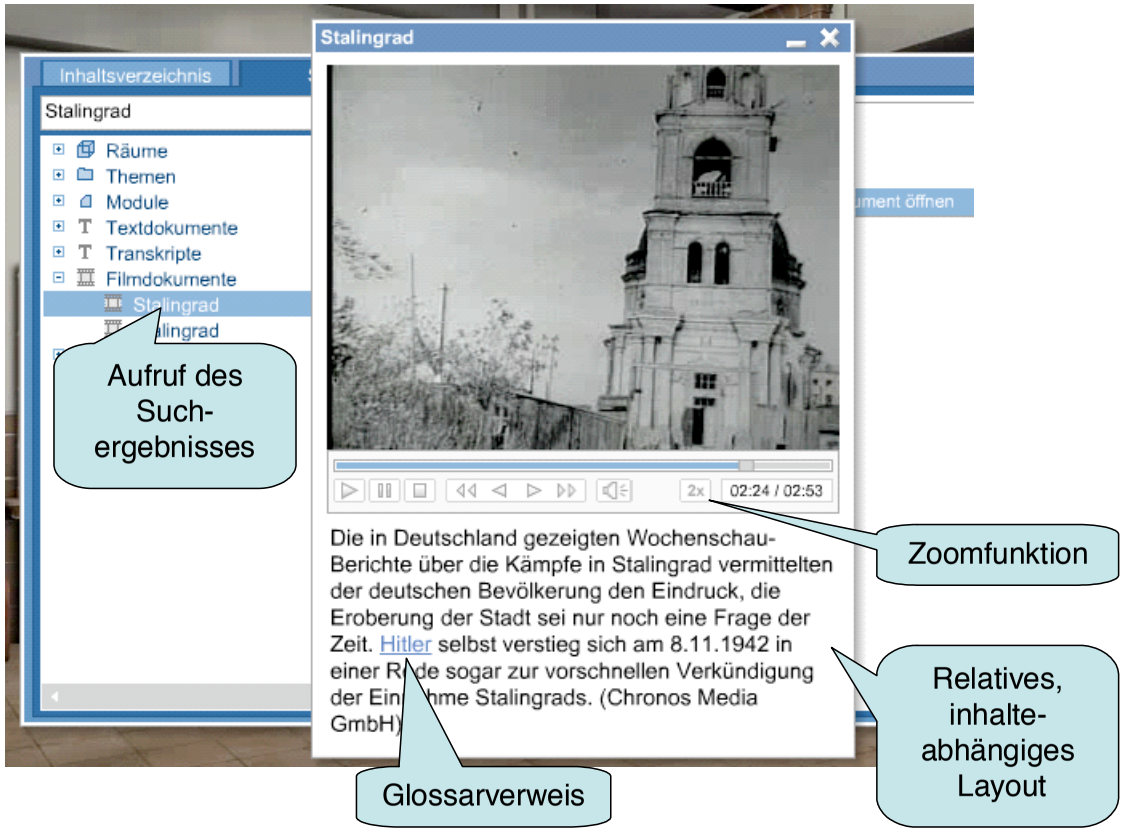


Abbildung A.3.: Zur Page-XML-Instanz in Abbildung A.2 korrespondierendes Ausgabefenster


```
<?xml version="1.0" encoding="UTF-8"?>
<init version="1.0">
  <INSTALLATION_TYPE type="String">full</INSTALLATION_TYPE>
  <SERVER type="Object">
    <BASE_URL type="String">.</BASE_URL>
    <STREAMING_URL type="String">http://www.lemolernen.de/content</STREAMING_URL>
    <SAVE_URL type="String">http://127.0.0.1/SavePage</SAVE_URL>
    <SHUTDOWN_URL type="String">http://127.0.0.1/Shutdown</SHUTDOWN_URL>
    <WEBSERVICE_URL type="String">http://127.0.0.1/axis/services/FlameService?wsdl</WEBSERVICE_UR
    <CONNECTION_RETRIES type="Number">10</CONNECTION_RETRIES>
    <CONNECTION_RETRY_INTERVAL type="Number">2000</CONNECTION_RETRY_INTERVAL>
    <WORKSHEET_URL type="String">./userdata</WORKSHEET_URL>
    <MULTITOOL_PATH type="String">multitool</MULTITOOL_PATH>
    <MULTITOOL_MEDIA_URL type="String">./res</MULTITOOL_MEDIA_URL>
    <ALLOW_DOMAINS type="Array">
      <item type="String">http://127.0.0.1</item>
    </ALLOW_DOMAINS>
  </SERVER>
  <START_LINK type="Link">
    <url type="String">res/index.xml</url>
    <type type="String">Startseite</type>
    <label type="String">Das kurze 20. Jahrhundert</label>
    <tooltip type="String">Zur Startseite</tooltip>
  </START_LINK>
  <GLOSSARY type="Object">
  </GLOSSARY>
  <HELP_URL type="String">res/help/struct/help_sitemap.xml</HELP_URL>
  <SITEMAP_URL type="String">res/sitemap.xml</SITEMAP_URL>
  <TIMELINE_LINK type="Link">
  </TIMELINE_LINK>
  <LEMO_ONLINE_LINK type="Link">
  </LEMO_ONLINE_LINK>
  <MULTITOOL_LINK type="Link">
```

Abbildung A.4.: Auszug der zentralen Framework-Konfiguration (Metalevel-Beschreibung des Anwendungsrahmens)

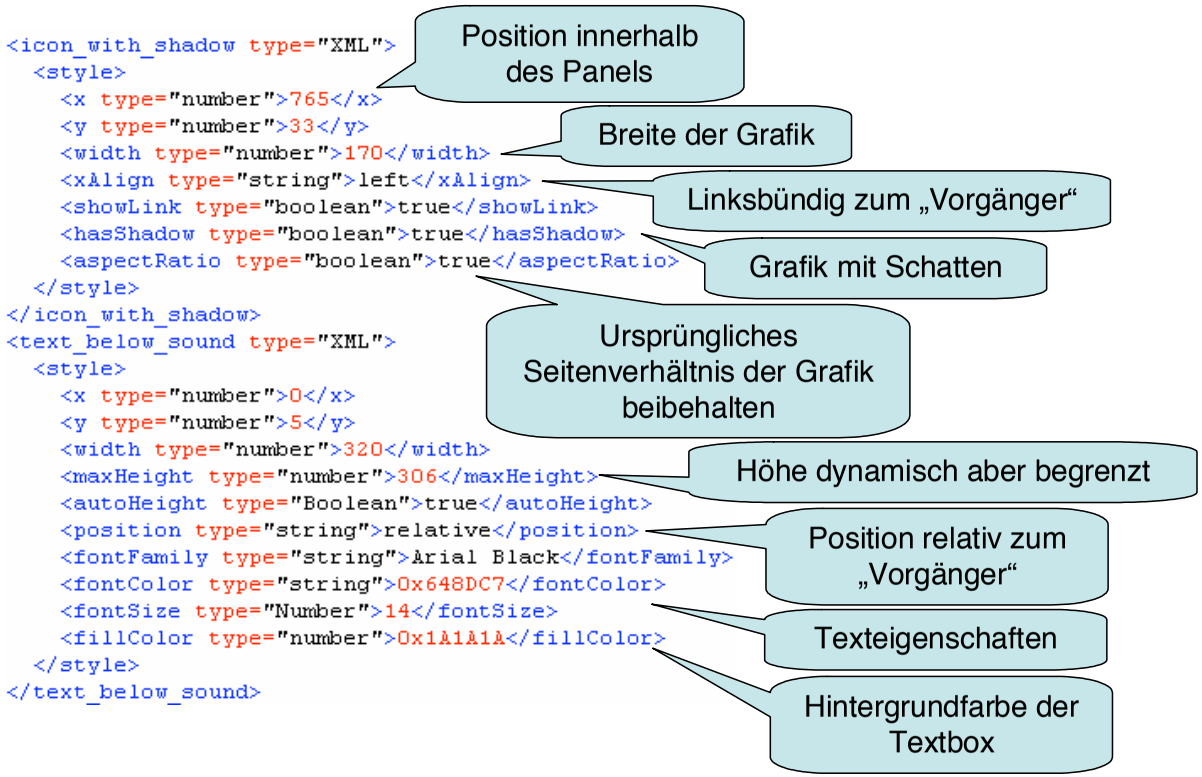
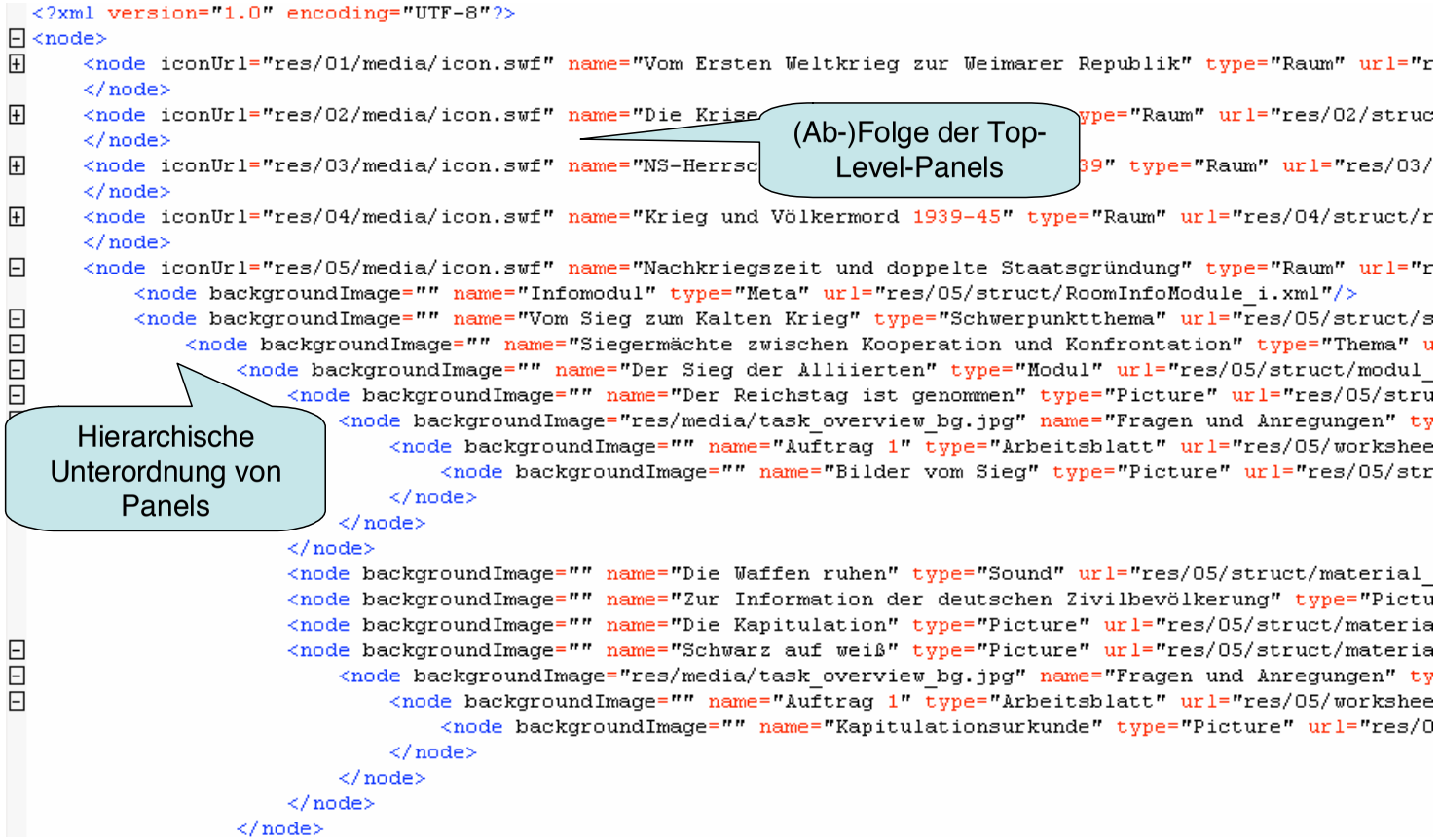


Abbildung A.5.: Beispiele globaler Stildefinitionen




```
<?xml version="1.0" encoding="UTF-8"?>
<node>
    <node iconUrl="res/01/media/icon.swf" name="Vom Ersten Weltkrieg zur Weimarer Republik" type="Raum" url="r
    </node>
    <node iconUrl="res/02/media/icon.swf" name="Die Krise                                  ype="Raum" url="res/02/struc
    </node>
    <node iconUrl="res/03/media/icon.swf" name="NS-Herrsc                                  39" type="Raum" url="res/03/
    </node>
    <node iconUrl="res/04/media/icon.swf" name="Krieg und Völkermord 1939-45" type="Raum" url="res/04/struct/r
    </node>
    <node iconUrl="res/05/media/icon.swf" name="Nachkriegszeit und doppelte Staatsgründung" type="Raum" url="r
        <node backgroundImage="" name="Infomodul" type="Meta" url="res/05/struct/RoomInfoModule_i.xml"/>
        <node backgroundImage="" name="Vom Sieg zum Kalten Krieg" type="Schwerpunktthema" url="res/05/struct/s
            <node backgroundImage="" name="Siegermächte zwischen Kooperation und Konfrontation" type="Thema" u
                <node backgroundImage="" name="Der Sieg der Alliierten" type="Modul" url="res/05/struct/modul_
                    <node backgroundImage="" name="Der Reichstag ist genommen" type="Picture" url="res/05/stru
                        <node backgroundImage="res/media/task_overview_bg.jpg" name="Fragen und Anregungen" ty
                            <node backgroundImage="" name="Auftrag 1" type="Arbeitsblatt" url="res/05/workshee
                                <node backgroundImage="" name="Bilder vom Sieg" type="Picture" url="res/05/str
                            </node>
                        </node>
                    </node>
                    <node backgroundImage="" name="Die Waffen ruhen" type="Sound" url="res/05/struct/material_
                    <node backgroundImage="" name="Zur Information der deutschen Zivilbevölkerung" type="Pictu
                    <node backgroundImage="" name="Die Kapitulation" type="Picture" url="res/05/struct/materia
                    <node backgroundImage="" name="Schwarz auf weiß" type="Picture" url="res/05/struct/materia
                        <node backgroundImage="res/media/task_overview_bg.jpg" name="Fragen und Anregungen" ty
                            <node backgroundImage="" name="Auftrag 1" type="Arbeitsblatt" url="res/05/workshee
                                <node backgroundImage="" name="Kapitulationsurkunde" type="Picture" url="res/0
                            </node>
                        </node>
                    </node>
                </node>
```

Abbildung A.6.: Auszug aus der Page-XML-Instanz eines Video-Panels

B. Frameworkausprägung in „Geschichte und Geschehen IV“

Die folgenden Auszüge aus Bildschirmansichten von „Geschichte und Geschehen IV“ illustrieren, in welcher Weise, das Framework konkretisiert werden kann – wie die zunächst abstrakten Bausteine zu disponiblen Elementen der Software-Produktion und schließlich zu konkreten Modulen der Zielanwendung umgesetzt wurden.

Abbildung B.1.: Kinderbuch mit Zoom und Blätterfunktion

So ist es für den Autor bzw. Redakteur sehr einfach zu wählen, ob eine Serie von durch ihn recherchierten und mit einem aufgabenangepassten Werkzeug zusammengestellten Bildquellen unterstützt durch Blätter- und Zoommechanismen vom Nutzer rezipiert werden soll. Die Anpassung des entsprechenden Werkzeugs und die Auswahl, die Gestaltung und Verfeinerung der Sammlung der Bedienelemente bzw. deren Verhalten ist Teil der evolutionären Angleichung zwischen den Potentialen des Entwicklungsrahmens und den Domänenerfordernissen.

Eine individuelle Lösung, die einer der Autoren selbständig aus den Möglichkeiten des Frameworks abgeleitet hat – quasi in einem Gestaltungsspielraum, den er sich selbst erschlossen – ist in Abbildung B.2 wiedergegeben. Das ursprünglich für die feste schematische Nutzung vorgesehene Audio-Player-Element sollte jeweils ein Tondokument in einem Fenster präsentieren. Die Anordnung mehrerer Player verknüpft mit jeweils einem Dokument eröffnet z.B. die Möglichkeit zum simultanen Abspielen, ohne zusätzlich Bedienelemente zur Auswahl einführen zu müssen.

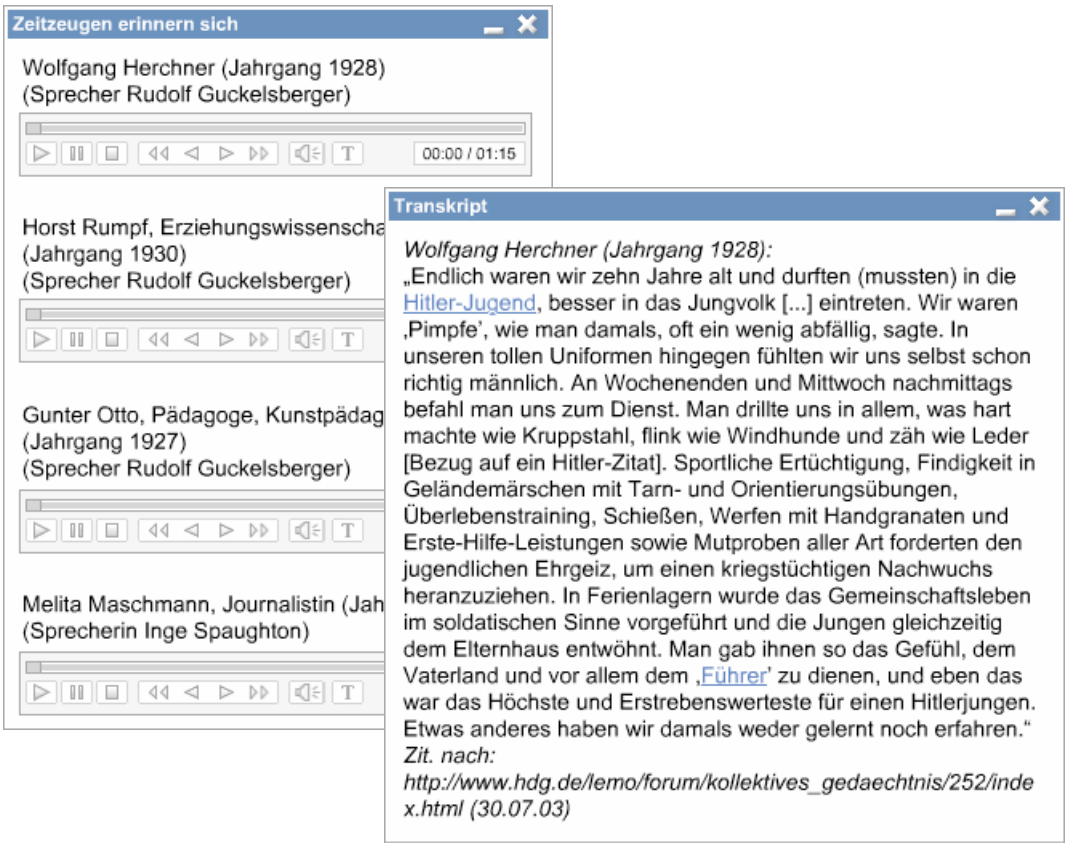


Abbildung B.2.: Eine Variante der Nutzung von Tonmaterial

Eine kontextspezifische Anpassung stellt z.B. der mit dem „T“ gekennzeichnete Button zum Aufruf des transkripierten Textes des Gesprochenen dar. Dieser ist ausschließlich mit den Mitteln der so genannten Metalevel-Objektbeschreibung eingefügt. Er ist also nicht Teil der abstrakten Player-Komponente. Im Gegensatz dazu ist etwa der Lautstärkeregler eine Erweiterung an White-Box-Schnittstellen der Framework-Komponente.

Die im Bild B.3 über dem Fortschrittsbalken der Bedienkonsole des Video-Players abgebildeten Schnittmarken sind kein originärer Bestandteil des Frameworks gewesen. Der Bedarf nach einer entsprechenden Funktionalität ist das Ergebnis des diskursiven Prozesses der Entwicklung interaktiver Aufgabenblätter, die historische Filme bzw. auch Audiomaterial zum Gegenstand von Aufgabenstellungen der Analyse und Interpretation machen.

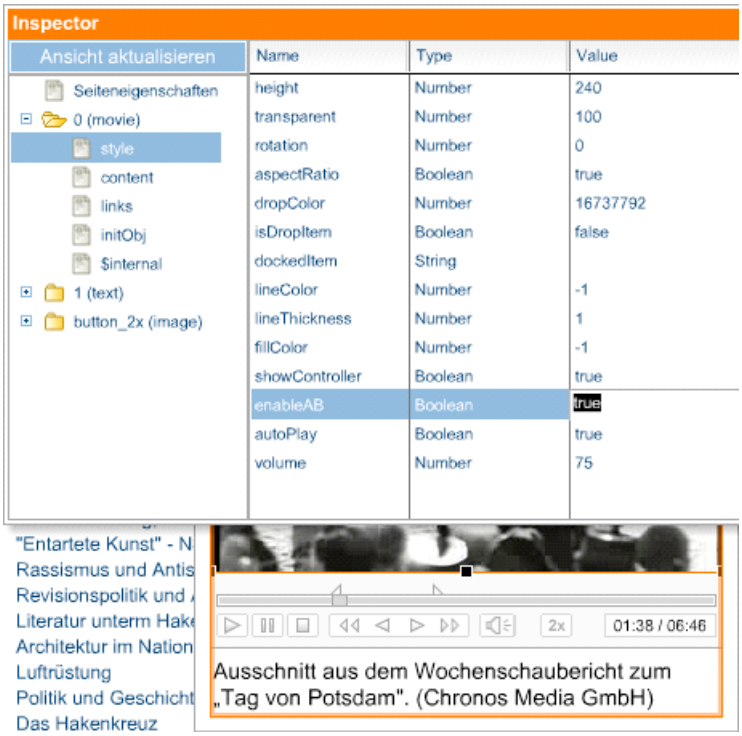


Abbildung B.3.: Schnittmarken zur Steuerung von Filmen

Das in der Abbildung angedeutet Fenster mit dem Titel „Inspector“ ist ein Bestandteil des WYSIWYG-Bearbeitungsmodus. Darin können die Eigenschaften einzelner Bildschirmelemente im Detail bearbeitet werden. Dies stellt für Redakteure eine Möglichkeit dar, die Meta-Level-Objekte der Elemente zu manipulieren. Das setzen der im Bild blau markierten Eigenschaft `enableAB` auf den Wert `true` führt etwa zu einem Eintrag in die Meta-Level-Beschreibung des Players, dessen XML-Serialisierung wie folgt aussieht:

```
<end type="number">174.766923076923</end>
<begin type="number">97.8173076923077</begin>
```

Die Anpassung und Eintragung der konkreten Zeitpunkte erfolgt bei der Bearbeitung im WYSIWYG-Modus einfach über das Verschieben der Schnittmarken.

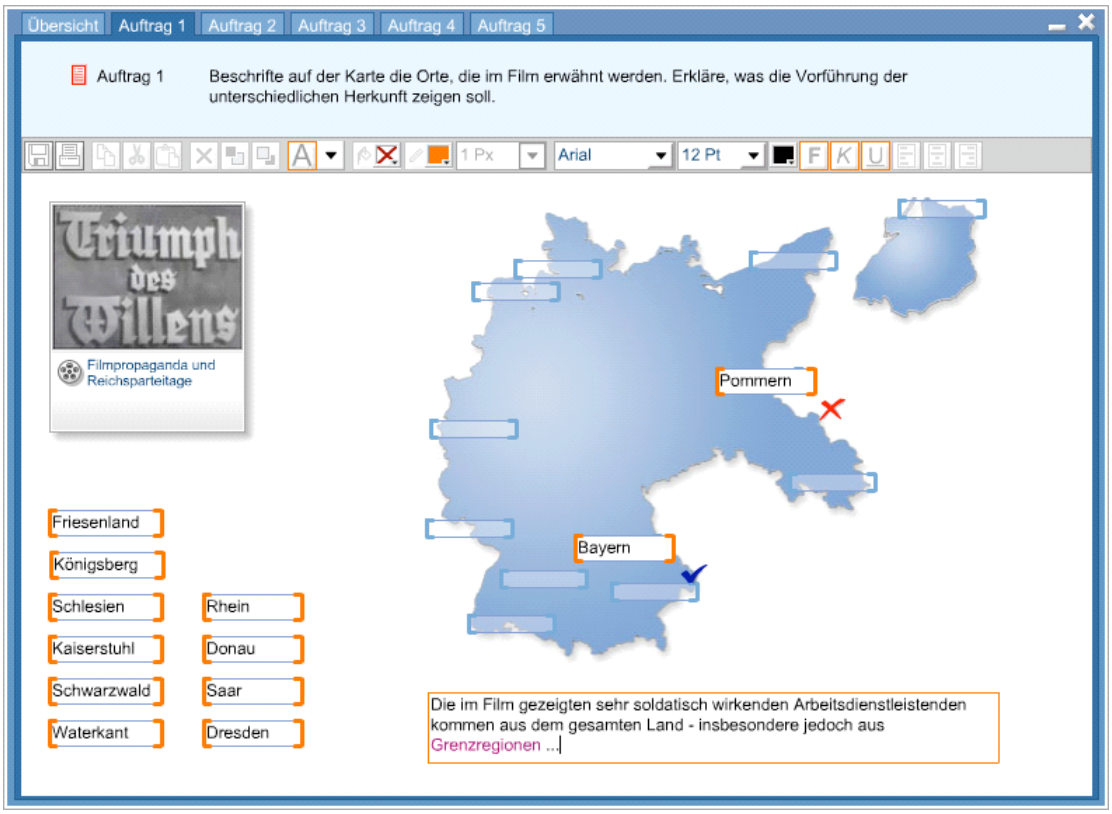


Abbildung B.4.: Arbeitsblatt zur spielerischen Vermittlung von Wissen

Literaturverzeichnis

[Amb02] S. W. Ambler. *Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process.* John Wiley & Sons, Inc., New York, 2002.

[Aue06] S. Auer. RapidOWL – an Agile Knowledge Engineering Methodology. `http://www.informatik.uni-leipzig.de/~auer/publication/RapidOWL.pdf`, 2006. Akzeptierter Beitrag zum 1st International Workshop on Semantic Technologies in Collaborative Applications (STICA 06).

[BD02] J. Bortz und N. Döring. *Forschungsmethoden und Evaluation für Human- und Sozialwissenschaftler.* Springer-Verlag, Berlin, Heidelberg, New York, 3.. Auflage, 2002.

[Bec00] K. Beck. *Exteme Programming.* Addison-Wesley, 2000.

[BG91] S. Bødker und K. Gronbæk. Design in Action: From Prototyping by Demonstartion to Cooperative Prototyping. In: J. Greenbaum und M. Kyng, (Hrsg.), *Design At Work: Cooperative Design Of Computer Systems*, Seiten 197–218. Lawrence Erlbaum Associates, Inc., Publishers, Hillsdale, New Jersey, 1991.

[BGK93] S. Bødker, K. Gronbæk und M. Kyng. Cooperative Design: Techniques and Experiences From the Scandinavian Scene. In: D. Schuler und A. Namioka, (Hrsg.), *Participatory Design: Principles and Practices*, Seiten 157–175. Lawrence Erlbaum Associates, Inc., Publishers, Hillsdale, New Jersey, 1993.

[BGMSW93] J. Blomberg, J. Giacomi, A. Mosher und P. Swenton-Wall. Ethnographic Field Methods and Their Relation to Design. In: D. Schuler und A. Namioka, (Hrsg.), *Participatory Design: Principles and Practices*, Seiten 123–155. Lawrence Erlbaum Associates, Inc., Publishers, Hillsdale, New Jersey, 1993.

[BKKZ92] R. Budde, K. Kautz, K. Kuhlenkamp und H. Züllighoven. *Prototyping: An Approach to Evolutionary System Development.* Springer-Verlag, Berlin, Heidelberg, 1992.

[BLFM05] T. Berners-Lee, R. Fielding und L. Masinter. Uniform Resource Identifier (URI): Generic Syntax. `http://www.gbiv.com/protocols/uri/rfc/rfc3986.html`, 2005.

[BMR+98] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad und M. Stal. *Pattern-orientierte Software-Architektur: Ein Pattern-System*. Addison-Wesley, München, 2. Auflage, 1998.

[Boe81] B. Boehm. *Software Engineering Economics*. Prentice Hall, 1981.

[BR06] A. Brüggemann und M. Rohs. Extreme Programming – Extreme Learning? Beitrag zum 3. Workshop der GI-Fachgruppe Didaktik der Informatik, Juni 2006.

[BvK02] C. Bunse und A. von Knethen. *Vorgehensmodelle kompakt*. Spektrum Akademischer Verlag GmbH, Heidelberg, Berlin, 2002.

[BYRN99] R. Baeza-Yates und B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press et al., New York, 1999.

[Cau00] J. Caumanns. Bottom-Up Generation of Hypermedia Documents. In: *Multimedia Tools and Applications*, volume Vol. 12, Seiten 109–128. Kluwer Academic Publishers, November 2000.

[CHK] Caumanns, J., Hollfelder, S. und Konrath, E. *Teachware on Demand: Erstellung und Nutzung konfektionierbarer Lernsoftware zur Weiterbildung in der Informations- und Telekommunikationstechnik*. Fraunhofer ISST.

[Chr03] O. Christ. *Content-Management in der Praxis : erfolgreicher Aufbau und Betrieb unternehmensweiter Portale*. Springer, Berlin, et al., 2003.

[CKV05] J. Caumanns, A. Karosseit und K. Virkus. LeMOLernen: Projekt-Abschlussbericht. Technischer bericht, Fraunhofer-Institut für Software- und Systemtechnik, Februar 2005.

[Dah95] C. Dahme. *Softwareentwicklung mit HyperCard: Benutzerfreundliche Interfacegestaltung*. Verlag Technik GmbH, Berlin, 1995.

[DEM+99] R. Depke, G. Engels, K. Mehner, S. Sauer und A. Wagner. Ein Vorgehensmodell für die Multimedia-Entwicklung mit Autorensystemen. *Informatik Forschung und Entwicklung*, 14(2):83–94, 1999.

[Dij89] E. W. Dijkstra. On the Cruelty of Really Teaching Computing Science. *Comm. of the ACM*, 12:1398–1404, 1989.

[DR97] C. Dahme und A. Raeithel. Ein tätigkeitsorientierter Ansatz zur Entwicklung von brauchbarer Software. *Informatik-Spektrum*, 20(1):5–12, 1997.

[DV04] J. De Vries. Rapid E-Learning: Groundbreaking New Research. In: LTI Newsline. `http://www.elearningmag.com/ltimagazine/article/articleDetail.jsp?id=102399`, Juni 2004.

[EK91] P. Ehn und M. Kyng. Cardboard Computers: Mocking-it-up or Hands-on the Future. In: J. Greenbaum und M. Kyng, (Hrsg.), *Design At Work: Cooperative Design Of Computer Systems*, Seiten 169–195. Lawrence Erlbaum Associates, Inc., Publishers, Hillsdale, New Jersey, 1991.

[EOOH94] E. Eberleh, H. Oberquelle und R. Oppermann (Hrsg.). *Einführung in die Software-Ergonomie*. Walter de Gruyter, Berlin, 2. Auflage, 1994.

[Ern04] N. Ernst. Macromedias Director geht in die nächste Runde. *CNET News.com*, Januar 2004.

[ES91] P. Ehn und D. Sjörgren. From System Descriptions to Scripts for Action. In: J. Greenbaum und M. Kyng, (Hrsg.), *Design At Work: Cooperative Design Of Computer Systems*, Seiten 241–268. Lawrence Erlbaum Associates, Inc., Publishers, Hillsdale, New Jersey, 1991.

[ext05] extremeprogramming.org. Extreme Rules. `http://extremeprogramming.org/rules.html`, Dezember 2005.

[Fer03] R. Ferber. *Information Retrieval: Suchmodelle und Data-Mining-Verfahren für Textsammlungen und das Web*. dpunkt.verlag, Heidelberg, 2003.

[FG04] G. Fischer und E. Giaccardi. Meta-Design: A Framework for the Future of End-User Development. In: *End User Development - Empowering People to Flexibly Employ Advanced Information and Communication Technology*. Kluwer Academic Publishers, 2004.

[FGY+04] G. Fischer, E. Giaccardi, Y. Ye, A. G. Sutcliffe und N. Mehandjiev. Meta-design: a manifesto for end-user development. *Commun. ACM*, 47(9):33–37, 2004.

[FK84] C. Floyd und R. Keil. Integrative Systementwicklung: Ein Ansatz zur Orientierung der Softwaretechnik auf die benutzergerechte Entwicklung rechnergestützter Systeme. (hrsg. v. BMFT, Forschungsbericht DV-84-003), September 1984.

[FKS05] F. Fuchs-Kittowski und P. Stahn. Kooperative Wissensarbeit in wissensintensiven Dienstleistungen – IT-Unterstützung mit der WiKo-Anwendung aus Anwendersicht. In: T. Schlegel und D. Spath, (Hrsg.), *Entwicklung innovativer Dienstleistungen*. Fraunhofer IRB Verlag, Stuttgart, 2005.

[Flo92] C. Floyd. Software Development as Reality Construction. In: C. Floyd, H. Züllighoven, R. Budde und R. Keil-Slawik, (Hrsg.), *Software Development and Reality Construction*, Seiten 86–100. Springer, Berlin, Heidelberg, 1992.

[Flo93] Christiane Floyd. STEPS – a methodical approach to PD. *Commun. ACM*, 36(6):83, 1993.

[fN03] Europäisches Komitee für Normung. *Software-Ergonomie für Multimedia-Benutzungsschnittstellen - Teil 1: Gestaltungsgrundsätze und Rahmenbedingungen (ISO 14915-1:2002)*. Beuth, Berlin, Wien, Zürich, 2003.

[FRS89] Christiane Floyd, Fanny-Michaela Reisin und Gerhard Schmidt. STEPS to Software Development with Users. In: *ESEC '89: Proceedings of the 2nd European Software Engineering Conference*, Seiten 48–64, London, UK, 1989. Springer-Verlag.

[FS99] E. Frieling und K. Sonntag. *Lehrbuch Arbeitspsychologie.* Hans Huber, Bern, 2. Auflage, 1999.

[GBL+04] F. Grossman, J. Bergin, D. Leip, S. Merritt und O. Gotel. One XP experience: introducing agile (XP) software development into a culture that is willing but not ready. In: *CASCON '04: Proceedings of the 2004 conference of the Centre for Advanced Studies on Collaborative research*, Seiten 242–254. IBM Press, 2004.

[Geu00] A. Geukes. *Design und Produktion Digitaler Interaktiver Lektionen.* Dissertation, Wirtschaftswissenschaftliche Fakultät der Universität Bielefeld, Oktober 2000.

[GHJV05] E. Gamma, R. Helm, R. Johnson und J. Vlissides. *Design patterns : elements of reusable object-oriented software.* Addison-Wesley, München et al., 2005.

[GKH91] J. Greenbaum und M. Kyng (Hrsg.). *Design At Work: Cooperative Design Of Computer Systems.* Lawrence Erlbaum Associates, Inc., Publishers, Hillsdale, New jersey, 1991.

[GP05] P. S. Grisham und D. E. Perry. Customer relationships and Extreme Programming. In: *HSSE '05: Proceedings of the 2005 workshop on Human and social factors of software engineering*, Seiten 1–6, 2005.

[GPS91] Franca Garzotto, Paolo Paolini und Daniel Schwabe. Authoring-in-the-large: software engineering techniques for hypertext application design. In: *Proceedings of the 6th international workshop on Software specification and design*, Seiten 193–201. IEEE Computer Society Press, 1991.

[GPS93] Franca Garzotto, Paolo Paolini und Daniel Schwabe. HDM – a model-based approach to hypertext application design. *ACM Trans. Inf. Syst.*, 11(1):1–26, 1993.

[Gro05] Groove Networks, Inc., San Francisco. *Users Guide: For Groove Virtual Office 3.1*, Dezember 2005. `http://www.groove.net/pdf/v3.0_Users_Guide.pdf`.

[GSV00] N. Güell, D. Schwabe und P. Vilain. Modeling Interactions and Navigation in Web Applications. In: *Lecture Notes in Computer Science 1921, Proceedings of the World Wild Web and Conceptual Modeling'00 Workshop, ER'00 Conference*, Salt Lake City, 2000. Springer. (Extended version).

[Hay04] F. Hayes. Chaos Is Back. `http://www.computerworld.com/managementtopics/management/project/story/0,10801,97283,00.html`, November 2004.

[HBM03] G. Hedin, L. Bendix und B. Magnusson. Introducing software engineering by means of Extreme Programming. In: *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*, Seiten 586–593, Washington, DC, USA, 2003. IEEE Computer Society.

[HH02] A. Hohenstein und K. Wilbers (Hrsg.). *Handbuch E-Learning*. Verlag Deutscher Wirtschaftsdienst, Köln, 2002.

[JBR99] I. Jacobson, G. Booch und J. Rumbaugh. *The Unified Software Development Process*. Adison Wesley Longman, Inc., Reading, Massachusetts, 1999.

[Kar03] A. Karosseit. LeMOLearning (Virtual Museum Online) Implementation of a Learning-Environment for School History. Beitrag zum Online Educa 2003 IBI Preconference-Workshop (`http://www.ibi.tu-berlin.de/diskurs/Veranstaltungen/onlineduca/onleduc03/karosseit.pdf`), 2003.

[Kar05] A. Karosseit. Partizipative Autorenworkflows für transdisziplinäre Content-Projekte. Tagungsbeitrag zur Learntec 2005 in Karlsruhe (`http://db.kmkg.de/cgi-bin/congress/course.pl?language=1&eve_id=41&cou_id=2806`), Februar 2005.

[Kea82] Greg Kearsley. Authoring systems in computer based education. *Commun. ACM*, 25(7):429–437, 1982.

[Ker01] M. Kerres. *Multimediale und telemediale Lernumgebungen: Konzeption und Entwicklung*. Oldenbourg Verlag, München, Wien, 2. Auflage, 2001.

[KFK05] A. Karosseit und F. Fuchs-Kittowski. Co-adaptive Entwicklungsumgebungen als Grundlage transdisziplinärer Software-Projekte. In: C. Stary, (Hrsg.), *Mensch & Computer 2005: Kunst und Wissenschaft - Grenzüberschreitungen der interaktiven ART*. Oldenbourg Verlag, München, 2005.

[KFK06] A. Karosseit und F. Fuchs-Kittowski. Co-Adaption als Grundlage der Zusammenarbeit in transdisziplinären Software-Projekten. In: *Netzwerk Organisation: Bildungstechnologie, Tätigkeitstheorie, Regulation, Lernen und Ethik*. Akademie-Verlag, 2006. (In Vorbereitung).

[KHM91] F. Kensing und K. Halskov Madsen. Generating Visions: Future Workshops and Metaphorical Design. In: J. Greenbaum und M. Kyng, (Hrsg.), *Design At Work: Cooperative Design Of Computer Systems*, Seiten 155–168. Lawrence Erlbaum Associates, Inc., Publishers, Hillsdale, New Jersey, 1991.

[Kro97] P. Kroha. *Softwaretechnologie*. Prentice Hall, München, 1997.

[KRW96] H. Kahler, M. Rohde und V. Wulf. Einführung von Groupware im Prozeß integrierter Organisations- und Technikentwicklung. In: P. Brödner, H. Paul und I. Hamburg, (Hrsg.), *Kooperative konstruktion und Entwicklung – die Zukunft von CAD-Systemen*, Seiten 95–115. Reiner Hampp Verlag, München, 1996.

[KW02] A. Karosseit und A. Wendt. Teachware on Demand im Kontext eines Peer-To-Peer-Netzwerks. In: J. von Knop und W. Haverkamp, (Hrsg.), *Zukunft der Netzte - Die Verletzbarkeit meistern*. Bonner Köllen Verlag, 2002.

[LC01] B. Leuf und W. Cunningham. *The Wiki Way: Collaboration and Sharing on the Internet*. Addison Wesley Publishing Company, April 2001.

[Luc06] The Apache Software Foundation: Lucene. Apache Lucene - Query Parser Syntax. `http://lucene.apache.org/java/docs/queryparsersyntax.html`, März 2006.

[Mac04a] Macromedia, Inc., San Francisco. *Macromedia Director® MX 2004: Director-Skriptreferenz*, Januar 2004.

[Mac04b] Macromedia, Inc., San Francisco. *Macromedia Director® MX 2004: Director Verwenden*, Januar 2004.

[Mac04c] Macromedia, Inc., San Francisco. *Macromedia Director® MX 2004: Erste Schritte mit Director*, Januar 2004.

[Mar98] Aaron Marcus. Metaphor design in user interfaces. *SIGDOC Asterisk J. Comput. Doc.*, 22(2):43–57, 1998.

[Mat01] H. R. Maturana. *Was ist erkennen?* Goldmann, München, 2001.

[Mau85] Herman A. Maurer. Authoring systems for computer assisted instruction. In: *Proceedings of the 1985 ACM annual conference on The range of computing : mid-80's perspective*, Seiten 551–561, 1985.

[MFCS92] M. J. Muller, R. F. Farrell, K. D. Cebulka und J. G. Smith. Issues in the usability of time-varying multimedia. In: Blattner M. M. und R. B. Dannenberg, (Hrsg.), *Multimedia Interface Design*, Kapitel 1, Seiten 7–52. ACM Press et al., New York, 1992.

[Mic97] Sun Microsystems. JavaBeans API specification. `http://java.sun.com/products/javabeans/docs/spec.html`, August 1997.

[NBS+99] M. Nagl, H. Balzert, H.W. Six, W. Schäfer, U. Kelter, A. Behle, B. Westfechtel, C. Weidauer, P. Pauen, J. Voss und J.P. Wadsack. Studie über Softwaretechnische Anforderungen an multimediale Lehr- und Lernsysteme. Technischer bericht, Verein Nordrhein-Westfälischer Hochschullehrer der Softwaretechnik (SofTec NRW), September 1999.

[NDH86] D. A. Norman und S. W. Draper (Hrsg.). *User Centered System Design: New Perspectives On Human-Computer Interaction.* Lawrence Erlbaum Associates, Inc., Publishers, Hillsdale, New Jersey, 1986.

[NHHM+04] H.M. Niegemann, S. Hessel, D. Hochscheid-Mauel, K. Aslanski, M. Deimann und G. Kreuzberger. *Kompendium E-Learning.* Springer, Berlin, Heidelberg, 2004.

[Nie93] J. Nielsen. *Usability Engineering.* AP Professional, Cambridge, 1993.

[NTB01] I. Nonaka, R. Toyama und P. Byosière. A Theory of Organizational Knowledge Creation: Understanding the Dynamic Process of Creating Knowledge. In: M. Dierkes, A. Berthoine Antal, J. Child und I. Nonaka, (Hrsg.), *Handbook Of Organizational Learning & Knowledge*, Seiten 491–517, Oxford, 2001.

[Ora01] A. Oram. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies.* O'Reilly UK, March 2001.

[Pre99] B. Preim. *Entwicklung interaktiver Systeme: Grundlagen, Fallbeispiele und innovative Anwendungsfelder.* Springer-Verlag, Berlin, Heidelberg, 1999.

[PS87] Nava Pliskin und Peretz Shoval. End-user prototyping: sophisticated users supporting system development. *SIGMIS Database*, 18(4):7–17, 1987.

[Ray00] E. S. Raymond. The Cathedral and the Bazaar. `http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/`, September 2000.

[RH02] M. Rohs (Hrsg.). *Arbeitsprozessintegriertes Lernen: Neue Ansätze für die berufliche Bildung.* Waxmann, Berlin, 2002.

[Roo04] K. Roost. Untersuchungen zu interdisziplinären Aspekten der Entwicklung von Multimedia Software. Diplomarbeit, Hochschule Wismar, 2004.

[Rup02] C. Rupp. *Requirements-Engineering und -Management. Professionelle iterative Anforderungsanalyse für IT-Systeme.* Carl Hanser Verlag, 2. Auflage, 2002.

[Sch90] M. Schrage. *Shared minds: the new technologies of collaboration.* Random House Inc., New York, 1990.

[Sch97] R. Schulmeister. *Grundlagen hypermedialer Lernsysteme: Theorie – Didaktik – Design.* R. Oldenbourg Verlag, München, Wien, 2. Auflage, 1997.

[Sch01] G. Schwabe. *Gemeinsames Material und Gruppengedächtnis.* Schwabe, G.; Streitz, N.; Unland, R. (Hrsg): Lehr- und Handbuch zum computerunterstützten kooperativen Arbeiten. Springer, Berlin / Heidelberg, Deutschland, 2001.

[Sch02] B. Schienmann. *Kontinuierliches Anforderungsmanagement.* Addison-Wesley, 2002.

[See79] T. Seeger. *Die Delphi-Methode, Expertenbefragungen zwischen Prognose und Gruppenmeinungsbildungsprozessen: überprüft am Beispiel von Delphi-Befragungen im Gegenstandsbereich Information und Dokumentation.* Hochschulverlag, Freiburg, 1979.

[SNH93] D. Schuler und A. Namioka (Hrsg.). *Participatory Design: Principles and Practices.* Lawrence Erlbaum Associates, Inc., Publishers, Hillsdale, New Jersey, 1993.

[SR98] Daniel Schwabe und Gustavo Rossi. An Object Oriented Approach to Web-Based Application Design. *Theory and Practice of Object Systems*, 4(4):207–225, Oktober 1998.

[SR02] R. Stolle und W. Rossak. A Second Generation Software Architecture for Internet-based, Directly-reactive Information Systems. In: *Proceedings of IDPT 2002*, Pasadena CA, June 2002. Intl. Conference on Integrated Design and Process Technology.

[SRB96] Daniel Schwabe, Gustavo Rossi und Simone D. J. Barbosa. Systematic hypermedia application design with OOHDM. In: *Proceedings of the the seventh ACM conference on Hypertext*, Seiten 116–128, 1996.

[SRK00] R. Stolle, W. Rossak und V. Kirova. A Component-Driven Architecture for Internet-Based, Directly Reactive Information Systems. In: *Proceedings of the IEEE ECBS TSC*, Seiten 129–237, Los Alamitos CA, March 2000. Intl. Conference on Systems Engineering of Computer Based Systems, IEEE Computer Society Press.

[SSZ83] H. Streich, K.-H. Sylla und H. Züllighoven. Anmerkungen zum Prototyping. In: *GI-13 Jahrestagung*, Seiten 344–356, Berlin, 1983.

[Ste91] C. Stefanski. Prototyping und Prototyping-Werkzeuge zur Unterstützung einer beteiligungsorientierten Systementwicklung. In: *Mensch und Technik; Werkstattbericht Nr. 103.* Ministerium für Arbeit, Gesundheit und Soziales des Landes Nordrhein-Westfalen, 1991.

[Ste04] S. Stein. Emergenz in der Softwareentwicklung – bereits verwirklicht oder Chance? Diplomarbeit, Hochschule für Technik und Wirtschaft Dresden (FH), Fachbereich Informatik/Mathematik, Dresden, Mai 2004.

[STMTK01] N.A. Streitz, P. Tandler, C. Müller-Tomfelde und S. Konomi. Roomware: Towards the Next Generation of Human-Computer Interaction Based on an Integrated Design of Real and Virtual Worlds. In: J.M. Carroll, (Hrsg.), *Human-Computer Interaction in the New Millenium.* Addison-Wesley, 2001.

[VSE05a] VSEK. Virtuelles Software Engineering Kompetenzzentrum: Glossareintrag – Organisationsentwicklung. `http://www.software-kompetenz.de/`, July 2005.

[VSE05b] VSEK. Virtuelles Software Engineering Kompetenzzentrum: Verfahrensbeschreibung – Integrierte Organisations- & Technikentwicklung (OTE) – Einstieg. `http://www.software-kompetenz.de/`, July 2005.

[Wei90] J. Weizenbaum. *Die Macht der Computer und die Ohnmacht der Vernunft.* Suhrkamp, Frankfurt am Main, 8. Auflage, 1990.

[Wei02] C. Weidauer. *Multimediale Lehr- und Lernsysteme: Effiziente Aufgaben- und Animationserstellung.* Spektrum Akademischer Verlag GmbH, Heidelberg, Berlin, 2002.

[Wel96] N. Welsch. *Entwicklung von Multimedia-Projekten mit Macromedia Director und Lingo.* Springer, Berlin, 1996.

[Wil01] H. Willke. *Systematisches Wissensmanagement.* Lucius & Lucius, Stuttgart, 2. Auflage, 2001.

[WR95] Volker Wulf und Markus Rohde. Towards an integrated organization and technology development. In: *DIS '95: Proceedings of the conference on Designing interactive systems*, Seiten 55–64, 1995.

[WS95] J. Wood und d. Silver. *Joint Application Development.* John Wiley & Sons, Inc., New York, 1995.

[Wys97] B. Wyss. *Die Welt als T-Shirt.* Dumont Literatur und Kunst Verlag, 1997.

Ehrenwörtliche Erklärung

Hiermit erkläre ich, dass

- mir die Promotionsordnung der Fakultät für Mathematik und Informatik der Friedrich-Schiller- Universität Jena bekannt ist,
- ich die Dissertation selbst angefertigt habe und alle von mir benutzten Hilfsmittel, persönlichen Mitteilungen und Quellen in meiner Arbeit angegeben sind,
- mich bei der Auswahl und Auswertung des Materials sowie bei der Herstellung des Manuskripts folgende Personen unterstützt haben
  - Prof. Dr. Wilhelm R. Rossak vom Institut für Informatik der FSU Jena,
  - Dirk Alban Adler als freier Mitarbeiter im Projekt LeMOLernen von der Firma KLITSCHE.DE sowie
  - Robert Erber von der Ernst Klett Verlag GmbH
- die Hilfe eines Promotionsberaters nicht in Anspruch genommen wurde und dass Dritte weder unmittelbar noch mittelbar geldwerte Leistungen von mir für Arbeiten erhalten haben, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen,
- ich die Dissertation noch nicht als Prüfungsarbeit für eine staatliche oder andere wissenschaftliche Prüfung eingereicht habe und
- dass ich die gleiche, eine in wesentlichen Teilen ähnliche oder eine andere Abhandlung nicht bei einer anderen Hochschule als Dissertation eingereicht habe.

Alexander Karosseit
Berlin, 11. Dezember 2006

Lebenslauf

Persönliche Daten:

Name:	Alexander Karosseit
Geburtsdatum:	11.09.1974
Adresse:	Erich-Boltze-Straße 15
	10407 Berlin
Familienstand:	ledig
Telefon:	+49 170 - 47 23 785
E-Mail:	alex.karosseit@gmail.com

Ausbildung:

Schulbildung:	1981-1990	Polytechnische Oberschule, Auerbach/Erzg.
	1990-1993	Gymnasium, Stollberg Abschluß: Hochschulreife (2,2)
Studium:	10/1993- 04/1999	Friedrich-Schiller-Universität, Jena Studium der Informatik Abschluß: Diplom (sehr gut)

Studienbegleitende Tätigkeiten:

08/1996- 06/1997	Studentische Hilfskraft am Institut für Informatik an der FSU Jena
10/1997- 03/1998	Werkstudent bei der IBM in Böblingen
11/1998- 04/1999	Diplomand bei der Carl Zeiss Jena GmbH

Berufstätigkeit:

05/1999- 06/2001	Software-Entwickler bei der EADS Dornier GmbH in Friedrichshafen
07/2001- 04/2006	Wissenschaftlicher Mitarbeiter am Fraunhofer-Institut für Software- und Systemtechnik in Berlin
Seit 05/2006	Wissenschaftlicher Mitarbeiter an der Humboldt-Universität zu Berlin

Weitere Tätigkeiten:

Akkreditierter Prüfer der Zertifizierungsstelle Cert-IT für die IT-Spezialistenprofile: Software-Developer, Database-Developer, Project-Coordinator und User-Interface-Developer

Mitglied im Gremium zur Neugestaltung der IT-Spezialistenprofile der „Software Developer“ im Rahmen des Projekts KIBNET – einer Initiative des BITKOM e.V.

Sonstige Interessen:

Langstreckenlauf, Snowboard, Klettern, deutsche-deutsche Literatur und Fotografie

Alexander Karosseit

Berlin, 11.Dezember 2006