Support of Collaborative Structural Design Processes through the Integration of Peer-to-Peer and Multiagent Architectures

Sascha Alda, University of Bonn, Germany (alda@iai.uni-bonn.de)
Jochen Bilek, University of Bochum, Germany (bilek@inf.bi.rub.de)
Dietrich Hartmann, University of Bochum, Germany (hartus@inf.bi.rub.de)
Armin B. Cremers, University of Bonn, Germany (abc@iai.uni-bonn.de)

Abstract

Structural engineering projects are increasingly organized in networked cooperations due to a permanently enlarged competition pressure and a high degree of complexity while performing the concurrent design activities. Software that intends to support such collaborative structural design processes implicates enormous requirements. In the course of our common research work, we analyzed the pros and cons of the application of both the peer-to-peer (University of Bonn) and multiagent architecture style (University of Bochum) within the field of collaborative structural design. In this paper, we join the benefits of both architecture styles in an integrated conceptual approach. We demonstrate the surplus value of the integrated multiagent–peer-to-peer approach by means of an example scenario in which several structural engineers are co-operatively designing the basic structural elements of an arched bridge, applying heterogeneous CAD systems.

1 Introduction

Complexity, competition pressure and quality demands of large engineering projects are permanently increasing and, therefore, require the close collaboration of specialized building enterprises and engineering offices that, in many cases, reside in different locations (e.g. areas, cities or buildings). Consequently, temporary projects aer established based on **networked co-operation**. This co-operation exhibit highly dynamic and loosely coupled structures, because team members (structural designers, engineers, draftsmen, technicians, etc.) join and leave the co-operation depending on changing responsibilities. The collaboration within the team work is typically enhanced by software systems, such as email, chat, file sharing, or service sharing systems. Most of such systems are, today, based on the centralized client-server model having one dedicated server and potentially many clients. This model fits the static organizations being of a (relatively) fixed size and characterized by a centralized place of control (e.g. a chief office). However, for loosely coupled co-operations with permanently changing conditions, as it is typical for complex structural design projects, this model seems not appropriate. Moreover, the effective computer based coordination of the complex and dynamic structural design processes is essential if conflicts and delays in design and construction should be avoided.

In most cases, the participating companies deploy *heterogeneous* technical infrastructure, since structural designers work with a plethora of different software systems, data models and formats, and operating systems. Thus, working on a common structural product model often tends to be problematic (Bretschneider et al. 1997). Furthermore, conventional engineering standard software, created in the last two decades, like finite element programs as well as CAD-systems[1], are not capable of supporting the complex and concurrent structural design processes carried out in practice directly (Bilek and Hartmann 2003). Overall, a common concept is lacking for both modeling and implementing concurrent interlaced design processes which incorporate the characteristic organisational structures of structural design project works as well as the integration of legacy software systems.

To overcome the deficiencies above mentionend we introduce a novel formal approach that efficiently embeds the following two different concepts: 1) **peer-to-peer** and 2) **multiagent**. In the course of our common research work carried out in Bonn (peer-to-peer) and Bochum

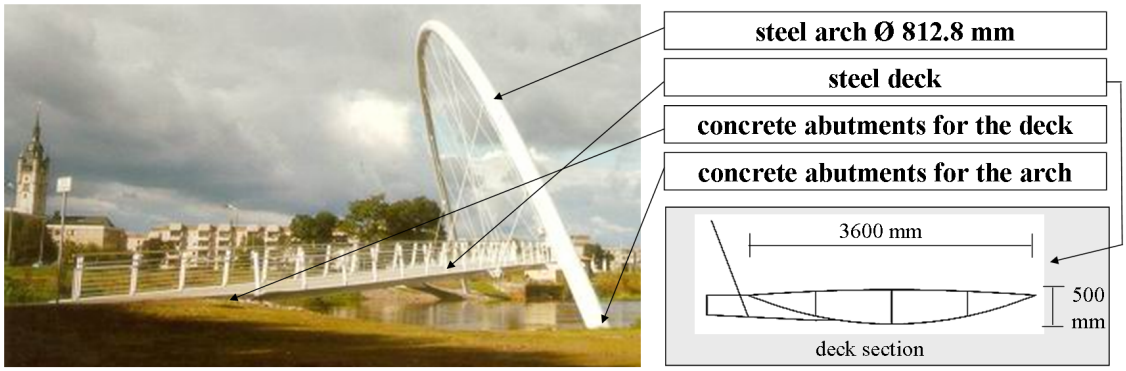


Figure 1: Reference building arched bridge in Dessau, main structural elements

(multiagent systems), respectively, we analyze pros and cons of both the peer-to-peer and multiagent architecture are analysed and the benefits of both concepts are linked in an integrated conceptual approach.

Fundamental solution concepts of this approach are demonstrated in terms of a prototype application showing the collaborative structural design process of an arched bridge, already erected, according to a reference (Fig.1). As can be seen in Figure 1, the bridge crosses the river Mulde in the city of Dessau (Germany) connecting two precincts. The bridge is a steel structure that spans 107.65 m. From the structural viewpoint, the bridge contains four main components: (i) an inclined steel arch (inclination 17°) having 15 tension rods, (ii) a bridge deck composed of several steel panels, (iii) the concrete arch and (iv) deck abutments founded on injection piles.

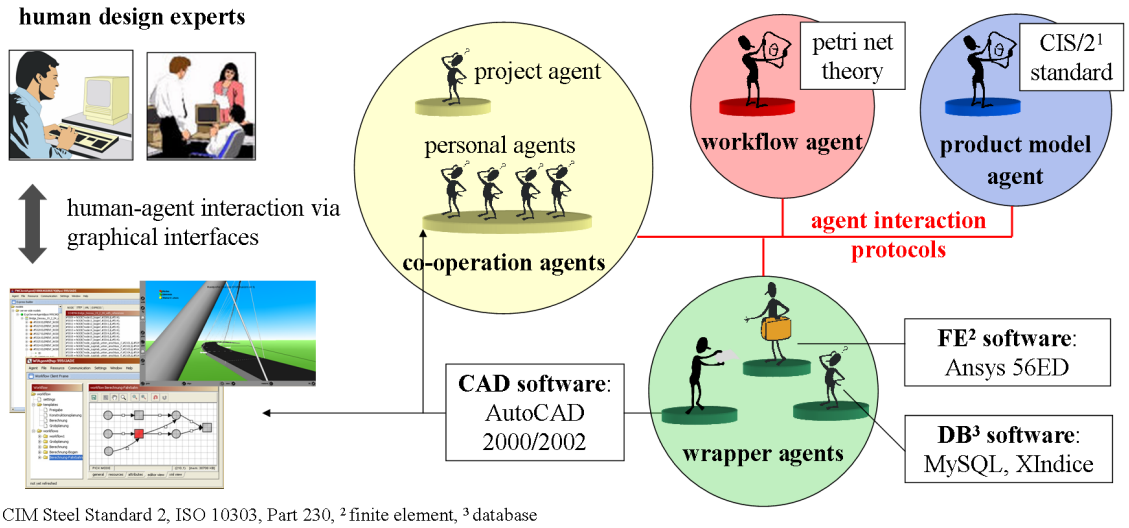
2 Background: Architectural Support for Structural Design Processes

In the course of the priority program 1103 “Networked-based co-operative planning processes in structural engineering” funded by the German Research Foundation, we have analyzed possible software architecture models for enhancing networked co-operations within structural design processes. In this context, we could identify two major models that have evidenced as practicable: the peer-to-peer and the multiagent architecture model, both evaluated by the University of Bonn and by the University of Bochum, respectively. In the following sections, we will first point out basic concepts of both models. Then, we will explain how these models have been adopted by our projects to support structural design processes. Subsequently, we will discuss the pros and cons of both approaches and how an integrated approach may benefit the collaborative work.

2.1 Multiagent Architecture

In recent years, the technology of **multiagent systems (MAS)** has become a rapidly developing area of research, emerged from distributed artificial intelligence (DAI). Thereby, a multiagent system is understood as a system consisting of several interacting autonomous problem-solving units (or software agents) that are capable of assisting their assigned design experts or other software agents. The adaptation of agent technologies to the specific needs and requirements of collaborative structural design implicitly requires the decomposition of the entire structural engineering process into adequate interacting agents. Then, these agents use specific domain expertise to assist the various engineers in their activities (Bilek et al. 2003).

During the analysis and decomposition of the entire planning process four substantial domains have been identified that must be embedded in a multiagent system for structural design leading to **agent model for collaborative structural design (ACOS)** (Bilek and Hartmann 2003). The four domains are the following: i) the specialized design experts and organizations (agent-based co-operation model) participating, ii) the specific structural design processes (agent-based



[1] CIM Steel Standard 2, ISO 10303, Part 230, [2] finite element, [3] database

Figure 2: Agent model for structural design and prototypical implemented agents within ACOS

process support model), iii) the associated (partial) product models (agent-based product model), and iv) the applied, heterogeneous engineering standard software (agent-based software integration model).

Within the **agent-based co-operation model** human experts assigned to organizations (like specialized engineering offices, authorities, other building companies, project works) are represented by means of co-operation agents. Designers are directly assisted by their own personal agents which operate as an interface between the human expert and the multiagent system. Furthermore, each project work is supported by a project-related agent. Such a project agent supervises and controls the project handling, and acts as an interface between the existing project-related resources (product model agents, wrapper agents, see below) and the human design experts.

The **agent-based product model** is based on the decomposition of the entire structural system into smaller structural subsystems. Each structural subsystem is accessible via an assigned product model agent that owns and stores knowledge about several structural elements created by the participating structural designers during the design process. The product model agents adjust their knowledge and, thus, check structural dependencies and retain product model consistency. The product model used there conforms to the CIS/2 standard (cp. Fig.2) as specified in ISO 10303 part 230. CIS/2 is a set of formal computing specifications that allows the modeling of steel structures covering the analysis model over the design model to the fabrication model.

The integration of heterogeneous engineering standard software (such as CAD-, FE-programs, databases) is provided through the implementation of wrapper agents. Within the **agent-based software integration model** wrapper agents act as an interface between the MAS and locally installed software. By that, corresponding software is made available to all other agents and/or human design experts.

The **agent-based process support model** is to control and distribute the complex workflows in the project to the appropriate designers. Therefore, a workflow agent has been implemented as a delegate to the project agent. The capabilities provided by the workflow agent are based on the petrinet theory. Based on petrinets, the workflow agent links design processes to project resources (product models, software, human experts).

In the course of the prototypical implementation the depicted four submodels have been simplified such that only the basic and necessary conceptual elements must be incorporated into

a set of structural design specific software agents (Fig.2). These agents interact with another by using standardized FIPA-ACL[2] messages defined in task specific interaction protocols. For communication purposes, several domain specific ontologies have been developed (product model ontology, workflow ontology, etc.) and concatenated with agent specific capabilities. The agent interaction implies several different abstraction layers: pure sending and receiving of ACL messages, mapping of ontology based message contents to internal objects, forwarding these objects to agent specific, internal capability modules, processing and handling the received information and, finally, accomplishing resulting activities.

2.2 Component-based Peer-to-Peer Architecture

The **peer-to-peer** architecture concept is a novel abstraction for developing software aimed to support virtual organizations. Following this style, a peer-to-peer architecture constitutes a distributed architecture that consists of uniform clients or so-called peers. Peers are capable not only of consuming, but also of providing computer resources like data, legacy applications, proprietary software solutions, simple routines, or even hardware resources. These resources are encapsulated by *peer services*. In contrast to other service-oriented architectures, peer-to-peer architectures assume an unstable and dynamic topology as an important constraint. That is because peers are solely responsible to affiliate to a network. Beyond the possibility of direct resource sharing, peer-to-peer architectures enable single peers to organize into so-called peer groups. These self-governed communities can share, collaborate, and communicate in their own private web. The purpose is to subdivide peers into groups according to common interests or knowledge independent from any given organizational or network boundaries.

In the course of the research project CoBE, we have developed DeEvolve (Alda et al. 2002), a self-adaptable peer-to-peer architecture based on top of the JXTA standard peer-to-peer framework (Sun Microsystems 2004). DeEvolve incorporates the component technology as the fundamental technology. According to the component technology, peer services are made up by the composition of single software components. A component model called FlexiBean does prescribe the valid interaction primitives for both the local interaction within a service and the remote interaction among distributed services. Peer services can be made available (published) to other peers by means of advertisements. Each service can be assigned to at least one or more group affiliations, in order to restrict the access to a service for authorized group users. Users of other remote peers are able to discover and use these services. Additionally, we provide a composition language called PeerCAT (Alda and Cremers 2004), which enables users to declare the composition of different peer services towards individual, higher-level applications. As an self-adaptable architecture, DeEvolve is capable of detecting and resolving unanticipated exceptions such as the failure or unavailability of peers. The handling of exceptions is done in strong interaction with end-users: an end-user, for instance, is able to decide which routine is to be executed for resolving an occurred exception. DeEvolve is accompanied by a couple of auxiliary tools not only for the discovery, advertisement, and composition of services, but also exception handling, as well as for the management of groups.

On top of the DeEvolve platform, we have developed the CoBE Awareness Framework (Alda et al. 2002), which realizes a decentral awareness model to coordinate the working activities within a project. This framework can be used by other applications to notify peers about changes within the application. The purpose of this model is to make the interactions between peer services explicit for end-users. End-users are thus accomplished to derive further actions based on the sole awareness of precedent activities stemming from other users who, for instance, belong to the same (project) group. Users can either be notified directly on the screen or, if they are currently unavailable, asynchronously via email.

2.3 Assessment of both Architectures

Apparently, both architectural models explained in the previous two sections exhibit not only advantages, but also limitations for the deployment in networked co-operations.

The integration of the component technology in the peer-to-peer architecture model as featured in the CoBE project exhibits a flexible way for the composition of distributed applications. Local and proprietary software can be encapsulated by components, which can then be published as peer services within the co-operation. Other users can integrate peer services with their individual services. The interfaces of services and, thus, the interaction between two services are modeled in a very simple way: event notification and data flow through a shared object are the only interaction primitives. End-users are accomplished to define compositions in a persistent way by defining connections among the service interfaces. These intuitive strategies for the creation of component-based applications are also applied for the adaptation of these architectures during runtime. These adaptations can either be carried out by end-users (e.g. due to changed personal requirements) or by the runtime environment itself to react on unexpected behavior. End-users are also involved when using a composed application through the integration of the CoBE Awareness Framework. This framework can be applied to coordinate the activities of a group of users. However, the awareness concept yields also some drawbacks: though users can perceive all activities within a group, it still depends on them, how to interpret precedent activities and whether or not subsequent activities are carried out. Consequently, there are no **autonomous concepts** that can take over this act of interpretation and execution of activities.

Autonomous computing is in fact one of the main benefits to fall back on agent-oriented architectures. Here, agents are capable of perceiving their environment and, based on the acquired information, capable of executing actions back to the environment autonomously. Compared to the rather end-user focused approach, arrogated by CoBE, the surplus value of this approach can be seen in the guaranteed and immediate execution of actions. If, for instance, an engineer has changed a part of an overall structural design model, then an agent could react instantaneously to check the consistency of this model in dependency to all other parts. Given that all other parts of the model reside on different network nodes within the co-operation, the agent could migrate to the respective nodes instead of delegating the task of consistency check to other remote agents. The migration of an agent thereby considerably decreases the amount of network traffic. However, the agent-based architecture model also exhibits some weaknesses. In contrast to components, agents are rather inappropriate for the integration and flexible composition of heterogeneous software. This due to the communication overhead that may result if several agents interact. Unlike components, which feature an intuitive and easy way for interaction, agent interaction is based on an ontology-based communication. Therefore, an ontology has to be defined for each software composition, which might be, dependent on the scale of a composition, a complex and cumbersome task. We actually recognized this circumstance during our efforts to integrate different distributed installations of the AutoCAD program by means of so-called wrapper agents (Mori and Cutkosky 1998).

What can be clearly ascertained by the comparison of both the agent-based and the component-based architecture model is that each model possesses limitations, which are in turn featured as advantages possessed by the other model. We, therefore, consider the integration of the benefits of both models towards a unified model as profitable. Our claim is to apply each model as the need arises. Primarily, components serve as ways and means for the flexible composition of distributed software artifacts. The integration of the Awareness framework offers the supplementary involvement of end-users if required. On top of this, agents can react on precedent activities and, hence, guarantee an autonomous course of action. We are convinced that an integration of both models is less complex from a technical viewpoint. What is more, it does not violate the original notions of the component-based and the agent-based architecture.

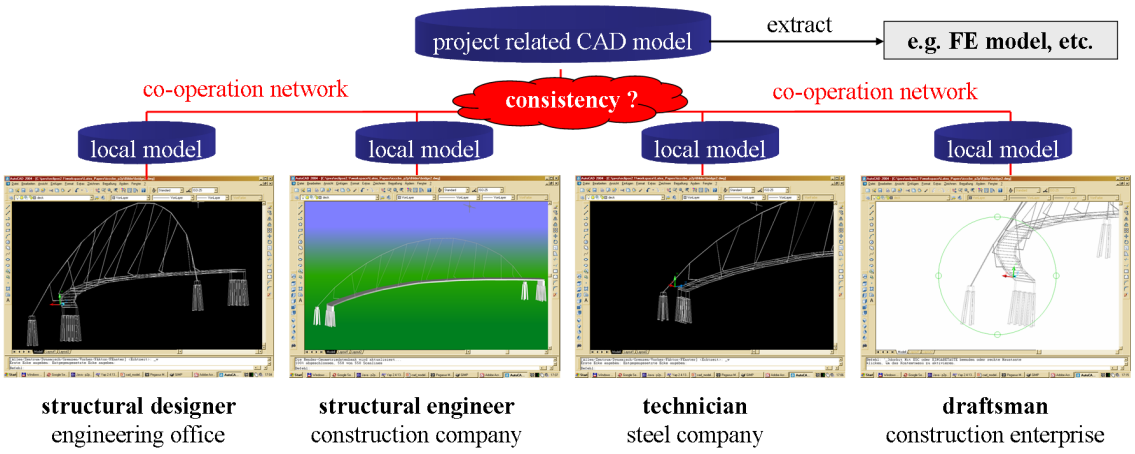


Figure 3: Collaborative preliminary design of the reference structure using heterogeneous CAD applications

So far, we represented only the key concepts for an integrated approach. We will present details of the integrated architecture design in the next chapter. Besides, we will discuss the appreciation of this approach by means of the above introduced reference structure.

3 Integrated Approach for Collaborative Structural Design

3.1 Motivation and Differentiation

In most cases, the entire structural design process can be divided into the four subprocesses 1) preliminary design, 2) structural analysis, 3) design/code verification and 4) structural detailing. Whenever the results of one subprocess are insufficient or deficient it might become necessary to reaccomplish some of the preliminary subprocesses which, then, emerges as iterative design activities. Each subprocess, in fact, leads to a partial product model (PPM), which overlaps partially with the partial product models of the other subprocesses.

An integrated MAS- and P2P-based solution concept that covers all depicted subprocesses seems extraordinary complex such that we assume some simplifications. To reduce the complexity of the iterative design process we will focus on a straightforward scenario in which we concentrate on basic CAD-based design processes in the preliminary design stage. In this scenario, several team work members design the basic structural elements of the bridge deploying heterogeneous CAD and finite element systems. The collaborative preliminary design results in a PPM in which a first qualitative digital model of the structural system is created and main parameters of geometry, topology and loads are determined. The PPM, then, is used for the following three subprocesses as a basis.

Fig.3 illustrates this situation: Several designers are modeling specific parts of the structural system of the bridge with their available CAD application concurrently depending on the role they have taken in the project work. Thereby, it might be essential for the single designers to be in the know about what their colleagues are just modeling. Thus, our scenario requires a co-operation network in which activities performed on heterogeneous CAD software systems are transmitted to all collaboratively working designers. By that, the coordination of the various designers may be better facilitated than without such a co-operation network. Moreover, the results of the individual activities are centralized in the PPM for preliminary design. The continuous analysis of the PPM could then be taken to ensure consistency. Comprising, in our scenario a kind of dispersed CAD application affiliated to a common PPM is required.

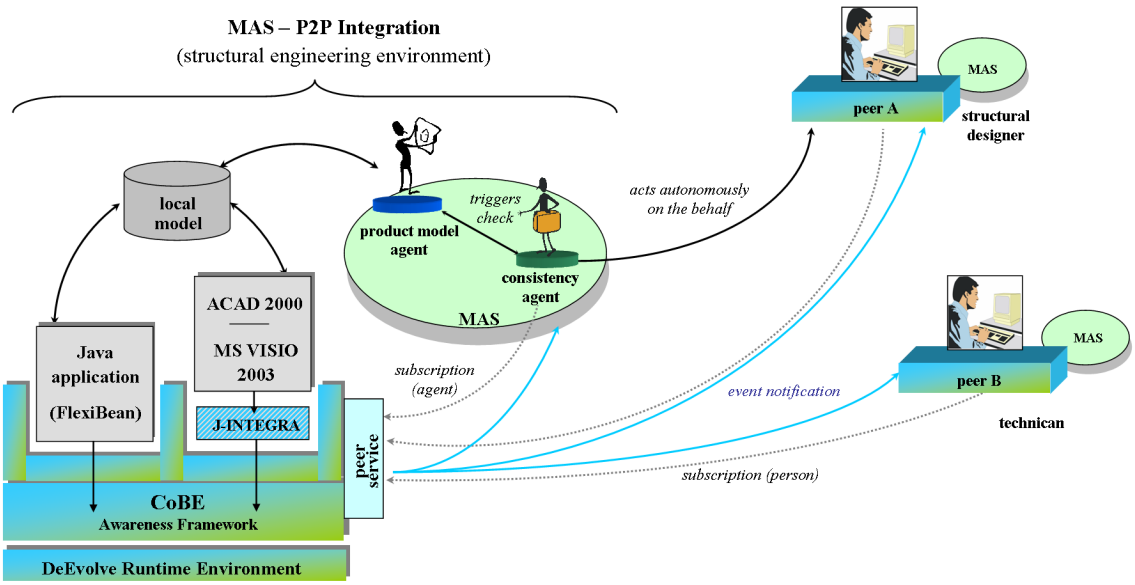


Figure 4: Overall design of the integrated MAS and P2P architecture

3.2 Design Issues

Figure 4 depicts the overall design of an integrated architecture (MAS-P2P) featuring fundamental aspects of both the peer-to-peer and the agents architecture model, respectively. What one can figure out from the first view is that each participating engineer is equipped with an individual MAS-P2P environment. The primary task of DeEvolve peer-to-peer runtime environment thereby is to enable users to deploy arbitrary software applications and tools. These applications can correspond to compositions of components, where the single components have to match the requirements of the prescribed FlexiBeans component model. We also allow to deploy components, which correspond to the widespread DCOM component model invented by Microsoft. The integration of the DCOM model in DeEvolve is handled by the DCOM-to-Java bridge J-INTEGRA. Through the deployment of DCOM components, it is possible to integrate and to use software that expose an open DCOM application programmable interface (API). Prominent tools stemming from the field of structural design, that have implemented such an open DCOM API for instance are the widely-used AutoCAD 2000 system and Microsoft's VISIO tool. Both Flexibeans compositions and integrated DCOM components can be published as a peer service within the co-operation. Through the remote interaction facilities of both component models, it is possible to access and accordingly to use applications by other remote peers. This makes sense, if engineers do not possess a local installation of the software, which is required for the collaborative design of a structural model.

Besides the possibility of sharing software among the participating members of a co-operation, we also allow to perceive modeling activities on a local model by means of the CoBE Awareness Framework. Each component can be enhanced easily by this framework so that predefined activities can made explicit to all (subscribed) members of a co-operation. The structural designer, for instance, is thus capable of identifying inconsistencies within the distributed models, and is then capable of triggering appropriate actions like the immediate notifications of the respective members. Components or applications, whose activities can be perceived by other members can also be published by peer services. Members first have to apply for membership to the publisher's peer before they can receive events (which in turn corresponds to concrete activities). The following events can be perceived: the saving of model, the change of a model, the opening of a model as well as events indicating that a member has become online or offline. A member can always request a copy of the model that has been changed by another member, for instance, to see the differences between the new model and the local model. In addition, the

editor of a model can add supplementary information, for instance, to comment or motivate the change of a model.

In the course of the MAS-P2P integration, not only (human) members, but also agents are able to subscribe to the awareness framework for perceiving activities. These agents act on the behalf of a human member by reacting immediately and autonomously on distinct events. Basically, two types of agents are provided: filter and consistency agents. Filter agents are able to block the sending of events in order to guarantee the privacy of the event publisher. The consistency agent is a special agent, which is able to perceive activities concerning the change of local model of the publisher. This agent type is responsible to check, whether or not the consistency of model has been violated. A consistency check is always carried out in strong interaction with an product model agent, whose purpose is to encapsulate the (proprietary) model. Actually, the following two options are supported: (1) a local model can be checked in interaction with the local product model agent or (2) the complete (distributed) model can be checked in collaboration with all remote product agents. For the complete check, a consistency agents is accomplished to migrate to all peers that provide MAS extension and, then, to execute the consistency check locally. By means of this extension, we can almost guarantee the consistency of a distributed model. Such a guarantee can hardly be given by the sole application of the Awareness Framework: due to the huge amount of events that may occur, it might necessarily happen that inconsistencies are simply disregarded by end-users. Given that users perceive some inconsistencies of a model, it it still up to them to resolve the inconsistent state. By the deployment of agents within the Awareness Framework, however, we reduce the degree of freedom each user has during collaborative planing process.

3.3 Implementation Issues

We have already started a prototypical implementation of the integrated MAS-P2P architecture. A first and complex task has been to implement the bridge component for deploying DCOM-enabled applications. We could use the *com2ACAD* framework developed by the University of Bochum for integrating AutoCAD 2000 as a peer service in DeEvolve. This framework is an adaptation of the general J-INTEGRA bridge explained in the previous section. An adoption of this bridge in order to deploy the VISIO tool is under current development. Both the AutoCAD and the VISIO tool provide a notification channel which can be used to receive a plethora of pre-defined event types, indicating the change of a model. These events are delegated via the com2ACAD framework to the Awareness Framework and, in turn, transmitted to all subscribed users. Recently, we provide synchronous notification with dialogs as well as asynchronous notification via Email.

Agents are deployed in the JADE agent system, which is to be executed in parallel to the DeEvolve system. We are aware of problems concerning system exhaustion due to the high resource requirements (e.g. memory) of both systems. We therefore intend to integrate a lightweighted agent system within the DeEvolve plattform as a peer service.

4 Conclusions

In this paper, we have presented our approach of an integrated architecture model that covers the benefits of the prominent peer-to-peer and agent architecture model, respectively. Apparently, the proposed model is based on experience we already made while deploying each architectural model for the enhancement of collaborative structural design. Peer-to-peer architectures, in combination with the component technology, enable users to compose and distribute software in a very flexible way. The implementation of an awareness model thereby facilitates event notification about planing processes carried out by remote users. Those event notifications can in turn be used to determine subsequent activities. However, we have concluded that this approach is not sufficient to guarantee the consistency of a (distributed) structural design model. We claim

that agent architectures fill this gap by providing a model for the immediate and autonomous reaction on events through software agents. On the other hand, agent architectures can profit by the aspect of flexible software composition and integration exposed by a component-based peer-to-peer model.

We have demonstrated the surplus value of the integrated MAS-P2P approach by means of an example scenario stemming from the field of structural design. However, the reader should note that our approach is not only restricted to this application field. We also see a huge potential to support the process of collaborative modeling of fire protection models, as these collaborations exhibit similar general conditions as supposed for structural design (Alda et al. 2003).

As a future work, we see the completion of the technical realization of the elaborated architecture. A further concept that we regard as important to take into consideration is the adoption of transaction concepts within the agent model. Thus, agents are not only able to guarantee the consistency of a model, but are also able to regulate the concurrent access to a model by a group of members.

5 Endnotes

1) CAD : Computer Aided Design
2) FIPA : Foundation for Intelligent Physical Agents, ACL : Agent Communication Language

6 References

Alda, S. and A. B. Cremers (2004, April). Towards composition management for peer-to-peer architectures. In *Proceedings of the Workshop Software Composition (SC 2004), affiliated to the* 7^{th} *European Joint Conference on Theory and Practice of Software (ETAPS)*, Barcelona.

Alda, S., U. Radetzki, A. Bergmann, and A. B. Cremers (2002, April). A component-based and adaptable platform for networked cooperations in civil and building engineering. In *Proceedings of the* 9^{th} *International Conference on Computing in Civil and Building Engineering (ICCCBE-IX)*, Taipei.

Alda, S., M. Theiss, A. B. Cremers, U. Rüppel, and U. F. Meißner (2003, June). Holistic support for the collaborative planning of fire protection concepts in building design. In K. Gürlebeck, L. Hempel, and C. Könke (Eds.), *digital Proceedings, 16. Internationales Kolloquium über Anwendung der Informatik und der Mathematik in Architektur und Bauwesen (ikm)*, Weimar. Universität Weimar.

Bilek, J. and D. Hartmann (2003, April). Development of an agent-based workbench supporting collaborative structural design. In R. Amor (Ed.), *Proceedings of the* 20^{th} *CIB W78 Conference on Information Technology in Construction*, Weiheke Island, New Zeeland, pp. 39–46. University of Auckland.

Bilek, J., D. Hartmann, I. Mittrup, and K. Smarsly (2003, August). Agent-based concepts for the holistic modelling of concurrent processes in structural engineering. In J. Cha, R. Jardim-Gonçalves, and A. Steiger-Garção (Eds.), *Proceedings of the* 10^{th} *ISPE International conference on concurrent engineering: research and applications (ISPE−CE) – Avanced design, production and management systems*, Volume 1, Madeira Island, Portugal, pp. 47–53. A.A. Balkema Publishers.

Bretschneider, D., U. Kolender, and D. Hartmann (1997, August). Modelling collaborative engineering in object-orientated design systems. In *Proceedings of the Mouchel Cetenary Conference on Innovation in Civil and Construction Engineering*, Volume D, Cambridge.

Mori, T. and M. R. Cutkosky (1998, September). Agent-based collaborative design of parts in assembly. In *Proceeding of ASME Design Engineering Technical Conference*, Number

DETC98/CIS-5697, Atlanta, Georgia.
Sun Microsystems (2004, April). Jxta v2.2.1 protocol specification. http://www.jxta.org.