SCORM Interactions

Creative use of interactions in SCORM®

Claude Ostyn

For the most recent version of this book, see www.ostyn.com/resources.htm

Draft 1.1 – work in progress

Abstract

This document explains how to take advantage of the interactions data model specified in SCORM to support rich interactive content as well as classic question-based assessments and simulations. It provides a number of examples of applications of the interaction types. It provides a conceptual model of the data model and its uses for tracking and resume or suspend operations for a SCORM content object. The document also shows how to combine interaction objects and SCORM interaction records to provide advanced behavior such as advanced response tracking or performance tracking, or to track the multivariate data used in confidence based markup.

Table of Content

Chapter 1 - Introduction

Who should read this book

This book is for designers and developers of interactive experiences delivered through SCORM content objects. Whether those experiences are simple questionnaires or complex simulations, they can often take advantage of the interactions data model specified for SCORM 2004 to track and manage interactive features such as learner response and scoring. To take full advantage of this book, some background in instructional design, test theory and test design is also helpful. Although this book can be useful to advanced assessment designers whose work is deeply grounded in sound theory, it is not written primarily for them and they may find it rather lightweight.

A basic understanding of data model concepts and JavaScript are useful for some of the more technical topics.

How to use this book

This book contains both conceptual overviews and technical information. If you read this book for design ideas that will be implemented by others, you can safely skip over the technical sections. The technical examples are, by their very nature, written using programming languages that are not really fit for normal human beings, but I tried to make them understandable even for people with only a passing acquaintance with JavaScript and XML.

It is highly recommended that you read at least the introductory chapters of my book *The Eye of the SCORM* to help understand the context in which the examples of this book work.

If you read the technical sections of this book because you are implementing a SCORM project and if it seems hard to understand, I recommend that you print out the ADL SCORM 2004 Runtime Environment document or otherwise keep it available for reference when reading this book. A copy of the IEEE 1484.11.3 Standard may also be helpful, since it is the source on which the SCORM 2004 "cmi" data model is based. In various places, this book will paraphrase what is in the ADL documents and in the IEEE standard. This is by design. Often, viewing the same dense information from different angles can make it easier to understand.

Some of the provided examples are simple enough to reproduce using a simple text editor such as Notepad. Others require automation or authoring tools for implementation because of the complexity of the underlying code.

Acknowledgements

Thanks to all the seekers and practitioners who share their knowledge generously.

Special thanks to Schawn Thropp for his keen editorial comments on the first draft.

...as becomes the ignorant, I must learn from the wise...

Plato, The Republic

Chapter 2 - Overview of interactions in SCORM

Conceptual overview

SCOs and interactions

The interactive experiences that fall under the scope of the SCORM specification for interactions always happen in the context of a SCO – a Shareable Content Object, which is the smallest unit of "smart" content managed and sequenced by a SCORM runtime environment. The SCORM runtime environment launches the SCO, but for the runtime environment the SCO is a black box. Once launched, the SCO communicates back with the runtime environment. This communication may involve the exchange of information. To achieve this, some script in the SCO calls functions provided by the runtime environment to get data values by name and set data values by name. The information exchange may include various kinds of information about the SCO. Some of this can be about interactions.

Kinds of interaction

There is a bit of confusion over the definition of an interaction in SCORM. In a later section we will introduce a more formal terminology so we can talk about specific aspects of interactions without introducing even more confusion. Until then, let us just say that the interaction information that can be exchanged between a SCO and a runtime environment can be used to report about all kinds of interactions between a user and the SCO. This could be test items in a quiz, responses to survey questions, simple or complex simulations, the performance of a task, and so on.

Interaction types

SCORM defines a number of interaction types with specific names:

- **True-False**
- **Multiple Choice**, including single Response and Multiple Response Multiple Choice variations
- **Fill-In**, sometimes called "short answer", including the Cloze variation in which several short answers must be given.
- **Long Fill-In**, which allows for the input of longer text as a form of response.
- **Numeric**, in which the response is a numeric value
- **Likert**, in which the learner chooses a value on a specified scale
- **Matching**, in which the learner identifies pairing matches between two sets of data
- **Sequence**, in which the learner response consists of an ordered sequence
- **Performance**, in which the learner response consists of a series of steps
- **Other**, which allows content developers to specify and build other kinds of interactions that still work with SCORM conformant systems.

Interaction types vs. interaction devices

You may have noticed that the interaction types don't include things like "*hot spot*" or "*slider*." This is because the SCORM interaction data model is concerned with the data resulting from interactions, not with the interactive device themselves. For example, the common "hot spot" question device is really a true-false or multiple choice question – either you click the right spot, or you click a wrong spot. Similarly, a slider device is really a numeric question – the response is a number within a given, fixed range. Many simulations or problem solving questions can be reported as performance interactions. SCORM does allow you to include a description of the device along with other data.

SCORM interactions vs. QTI

A standard way to specify the actual interactive devices is the main difference between a

specification like the IMS Global Consortium Question and Test Interoperability (QTI) Specification and the SCORM specification for interactions. In QTI, one specifies the devices and all their features as well as the evaluation mechanisms and the resulting data. In theory, given a QTI document, you can construct a test, present it to a user, collect the responses, evaluate and digest them, based solely on the content of the QTI document. SCORM is much simpler. It does not specify how to build and present the devices or how to evaluate the responses. It allows you to communicate information about the learner response, how long it took to respond, and what the result is in the form of a score or a correct/incorrect judgment. It also allows you to pass data about correct responses, if that is relevant, as well as a simple textual description of the interaction. However, unlike QTI, the SCORM specification assumes nothing about how the interactive devices are constructed and behave or how responses are evaluated.

It is of course possible for a SCORM content object to use its own QTI interpreter engine. A test engine generator can construct SCORM content automatically, using a QTI document. This content can then report a relevant subset of summary results as SCORM conformant information. However this is out scope for SCORM. SCORM conformance does not in any way require supporting any aspect of the QTI specification.

Evaluating learner responses

Controlled by the designer

The designer of the interaction objects decides how to evaluate the learner responses and what the results of this evaluation will be. SCORM does not specify how to do this. It just provides a way to record and communicate some information about the evaluation assumptions, such as predetermined correct responses and weights. Of course it also provides a way to record what the learner actually responded.

There is no requirement in SCORM that the responses be evaluated at run time. However, there is currently no standard process defined to collect responses through a SCORM runtime environment for automated or manual evaluation in another environment. This would be nice, but it will have to wait for champions to sponsor such standards.

No direct effect on SCO score and status

SCORM does not specify any connection between interaction response evaluation and the overall score for the SCO that provides the interactions. Obviously, a content developer might create such a connection. For example, the summary score for a quiz contained in a SCO can be calculated from the scores for the interaction objects in the SCO. But whether and how to do this is up to the content developer.

No direct effect on objectives

Although it is possible to specify objectives that are relevant to an interaction, SCORM does not specify any way for interactions to automatically affect the status of such objectives, or to be affected by the status of such objectives. A content developer can of course devise ways to do this in a particular way. For example, it is possible to use scripting and objective mapping to modify the status of shared global objectives in a SCORM package

Introducing the SCORM data model for interactions

Data models

The information that can be exchanged between a SCO and the runtime environment is specified in a standard *data model*. A data model is a document that specifies how to represent information using a well-defined set of data elements. For example, if the SCO needs to find out the name of the learner, it can ask for it by asking the runtime environment for the value of the data model element named "learner_name". This data element is specified in the data model named "cmi", which is itself described by SCORM. The SCORM data model is in turn based on an IEEE standard. In a data model, data elements are often contained in data structures. A data structure is basically a small data model within a larger data model. For example, a series of steps to solve a problem correctly can be represented in a data structure that is a collection of information about each step.

The interaction data model

A special data model is specified in SCORM to communicate information about interactions. This data model is a part of the larger "cmi" data model. We will look at the *interactions* data model in more detail later in this document. The *interactions* data model is specified as a collection of *interaction* records. A record is a data structure that contains various data elements. For example, each *interaction* record in the *interactions* collection is an instance of a data structure that contains a data element named "identifier", along with other data elements that contain additional information about the interaction.

Recording interactions: Interaction status vs. journaling

A controversial topic is whether interactions records in SCORM describe interaction events or interaction object state. In software, an object typically is an instance of a data model that also has some behaviors and some state. So, for example, you could say that a question in a test is an interaction object. The learner can interact with it, and the state of the question changes as a result of the learner interacting. If a learner changes her response, the state changes again. Some hold that a new interaction record should be created for each change of state. This is the journaling approach. Others hold that for reporting purposes it is good enough to report the last state of each interaction object. This is the object state approach. A paper test is a typical example of the state approach – when the test is submitted, the only data available for evaluation, reporting and analysis are the final state of each question on the paper test.

Choosing between status and journaling

In theory it would be nice to record each event to analyze what actually happened, whether the learner changed her answer, and to what, and so on, using a journaling approach. However the journaling approach could easily result in excessive amount of data that are never used in practice. The SCORM data model allows both approaches. There are some fairly tight limits on the number of events that can be recorded using the journaling approach. It is also possible to report a series of events even if you use the status approach. Since both approaches are supported by the data model, you can choose the one most appropriate to your situation. My preference is to use the status approach because it is more consistent with object oriented practices and it results in more compact, easier to use summary results.

A matter of terminology

What are we talking about?

By now your head may already be spinning, because the same words are used in the SCORM documents to mean different things. Sometime SCORM uses terms with a meaning that is different from what the rest of the world thinks it means. Even within the field of learning technology, different communities often use the same terms to mean different things. Therefore we need to agree on some common terminology before we go on.

The terms below are somewhat arbitrary. Different people have different names for the same concepts, or ascribe different meanings to the terms used here. You may need to translate mentally if you are one of those people.

Interaction object vs. interaction record

The SCORM data model does not specify what an interaction actually is. Is it something that happens, is it an interactive device in a SCO, is it a data record?

In this document, we will use the following terms to describe very different things:

- **User interface device**
- **Interaction object**
- **Interaction record**

Interaction device

A **user interface device** is what a learner interacts with directly. It is typically rendered on a screen and relies on keyboard or mouse, or both, for input from a user. It may contain various kinds of controls and implement various behaviors. For example, a set of radio buttons is a user interface device. These days, user interface devices also often support assistive technologies that can afford interaction through various kinds of rendering and input devices for people with special needs.

Interaction object

An **interaction object** is what a learner interacts with. It is a conceptual software object that provides some interactive functionality for a particular purpose. It is typically composed of one or more user interface devices coupled with data objects and scripts that control its behavior. SCORM does not define an interaction object. I just made it up because it is convenient for conversation. An interaction object could be a test item in a quiz, for example.

Interaction record

An **interaction record** is the set of data specified by SCORM to represents information about a specific interaction object. As we saw above, in this book we will use the "status approach", and therefore an interaction record is always used to communicate or hold the available information about the current status of an interaction object. This information may be incomplete. Sometimes the information is not available yet because the learner has not responded yet. Sometimes some data elements are not available at all, because the developer of the interaction object chose not to implement them. SCORM does specify what an interaction record may contain.

Note that it is not a requirement that every interaction object in a SCO report to the runtime environment through an interaction record. For example, you could design a SCO so that an interaction record is created only for interaction objects for which there is a response from the learner. You might also have interaction objects, such as practice objects, that never report any tracking information because they don't contribute to an overall score for the SCO.

Advanced interaction objects

Conceptually, it is useful to think of advanced interaction objects that may exist only in connection with other interaction objects.

- **Interaction monitor object**
- **Complementary interaction object**

Interaction monitor object

An interaction monitor object is an interaction object that watches what is going on with one or more other interaction objects and records its

observations. Such an object usually does not have a user interface and the learner is not aware of the existence of that object. For example, an invisible interaction object of type `performance` could monitor an interaction object of type `multiple_choice`. This interaction monitor object can then report, using a separate interaction record, in which order the learner made choices.

Complementary interaction object

A complementary interaction object is an interaction object that does have a user interface, but functions only in association with another interaction object. For example, in Confidence Based Marking (also called Certainty Based Marking), a learner must specify how confident she is that her answer to a question is correct, and this confidence rating affects the score given to the response to the question. This can be achieved and tracked in SCORM by using two complementary interaction objects, each of which will provide its own interaction record. The first interaction object is the main question, which can be any interaction type. The second interaction object is the confidence question. This would typically be a Likert interaction in which the learner must choose a value on a scale.

For Confidence Based Marking, the main question object scoring script could first interrogate the confidence question when it needs to evaluate its own response. If the confidence question has not been answered, the question scoring script could then take some action – either give up, or if the questions are still being displayed, ask the user to answer the confidence question. The response to the confidence question can then be used to adjust the score of the main question. Each of the associated questions can provide its own SCORM interaction record to allow analysis by a reviewer, or only the main question might provide a SCORM interaction record.

Chapter 3 - Tracking SCORM Interactions

The information in an interaction record

Overview

For each interaction that is tracked in SCORM, there is an interaction record.

As we saw above, in design approach we will assume that by interaction we mean an interaction object, and that when something changes in the interaction object this is reflected in an interaction record that is used to communicate information about the interaction object.

The record contains information about the interaction object itself and may contain other information useful for analysis, such as how a user responded to an interaction object.

Information about the interaction object

Mandatory information about the interaction

SCORM requires the following information about in every information record:

- **Identifier** – This is a label that uniquely identifies this particular interaction record. It is like a license plate on a car – no two interaction records will have the same identifier in the same SCO. If the interaction object is an item from a question bank, this could be the item catalog number in the question bank.
- **Interaction type** – This is a label that specifies the type of interaction. It must be one of the labels defined by the IEEE Standard 1484.11.1 for interaction types, such as `true_false` or `performance`.

Optional information about the interaction

- **Description** – This is an optional description of the interaction, intended for human readers. Since the Identifier is often meaningless or hard to read for humans who review the interaction data, this allows you to provide a more meaningful description. It can also be used to report the question stem for some type of test items.
- **Objectives** – This is an optional list of identifiers of objectives that are relevant to the interaction. SCORM does not specify any standard automation process that uses the information in this list. It is just informative.
- **Correct responses** – This is an optional data structure that contains information about correct responses. This is mostly intended for analysis of tracking logs. SCORM does not specify how to use the correct response information, if it is included. However, SCORM does specify different data models for the correct response information, depending on the interaction type. For more complex interactions, it is possible to specify multiple patterns of correct response. For example, there may be more than one correct way to perform the steps in a task; each way can be represented by a different pattern.
- **Weight** – This is an optional value that represents the weight of this particular interaction relative to the other interactions for the same SCO. SCORM does not specify how to use the weight. It is just informative

Information about the learner responses

Everything about responses is optional

Since there may not have been a response when an interaction record is communicated, nothing about the learner response or the result of evaluating a learner response is mandatory.

- **Learner response** – This represent the user's response to the interaction device. SCORM specifies different data models for the learner response information, depending on the interaction type.
- **Time Stamp** – This is a time stamp to record the date and time at which the learner responded. If a learner responds in multiple steps, this would typically be the time of the last response.
- **Latency** – This is the time that elapsed between when the learner was first presented with the interaction device, and when the learner started responding to it.

Information about evaluation results

Result information is optional

SCORM does not require that evaluation take place in a SCO, and it does no specify how the evaluation takes place.

- **Result** – This is a single data value that results from evaluating the learner's response. The possible values for the result are constrained by SCORM to a choice between a predefined result label and a numeric score.

Security considerations

Test security

SCORM has no specific provisions to provide for test security. How to secure content and sessions is out of scope for SCORM. How to ensure that users are authenticated is also out of scope. So is ensuring that users cannot tamper with the software on their computer while experiencing SCORM content. Obviously, the higher the stakes in a test, the more incentive there is for some learners to cheat.

Cheating

Some people have expressed great concern that the interaction data and evaluation code can be exposed in a SCO and potentially analyzed by a cheater to beat the test. In reality, however, if the stakes are high enough, I would ask whether more security questions don't have to be addressed first, such as how you know that the person taking a test is really the person you are trying to test. In a proctored environment that ensures this kind of security, it is also highly doubtful that test takers would have the time and tools to analyze the JavaScript to determine what the correct answers are. In any case, for such a test the code can be obfuscated, or SCOs can be designed to only collect responses, with the data collected through the SCORM user interface analyzed offline.

Appropriate uses

Problems arise only when naïve expectations are associated with online delivery of tests and exams. I, for one, would never trust a test administered online, using SCORM or any other method, unless I could also control the conditions in which the test is being taken by the end user. On the other hand, real time evaluation of interaction information can be wonderful in supporting adaptive content to provide learning experiences tailored to the learner's need. .

Chapter 4 - The SCORM interaction data model

Caution. If you are not technically inclined, you may want to skip this chapter

Data model organization

Hierarchical data model

The data model for interactions is hierarchical. It is defined in IEEE Standard 1484.11.1. Since SCORM is a conformant implementation of that standard, it uses the same hierarchical structure.

There is an ordered collection of interaction records. The order may or may not be meaningful for a SCO, depending on the design of your SCO, but the standard makes no assumption in this regard. The order is important for people who are using interaction records for journaling, i.e. to record interaction events as they occur, using separate interaction records. The order is much less important for people who are using interaction records to represent the state of interaction objects.

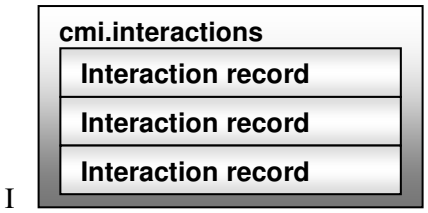

Figure 1 – The cmi.interactions element

Each interaction record consists of a number of elements that can be referenced individually in the GetValue and SetValue functions of the SCORM API.

In the SCORM cmi data model implementation of the IEEE Standard 1484.11.1, each record is assigned an index in an array of interaction records. This index value is not part of the standard data model and is only an artifact of the "dot notation" binding used in SCORM to reference elements in the data model. When using the SCORM API, this array index value is required to address individual interaction records. The Identifier element cannot be used directly to locate an interaction record in SCORM.

Internal organization of the interaction record

The interaction record is itself a hierarchy. Some elements are atomic, like `id`. Others are collections, like `objectives`. The most complex element is `correct_responses`, which can support multiple patterns for some of the interaction type.

The main implication of this for implementers is that storing the interaction record data elements as simple name-value pairs does not work well.

```
SetValue("cmi.interactions.0.id","Q_123435467-foo")
SetValue("cmi.interactions.0.type","multiple_choice")
SetValue("cmi.interactions.0.objectives.0.id","iobjid1")
SetValue("cmi.interactions.0.timestamp","2007-01-18T22:10:35.84Z")
SetValue("cmi.interactions.0.correct_responses.0.pattern","blue[,]white][,]red")
SetValue("cmi.interactions.0.weighting","2")
SetValue("cmi.interactions.0.learner_response","blue[,]red")
SetValue("cmi.interactions.0.result","incorrect")
SetValue("cmi.interactions.0.latency","PT2M54S")
SetValue("cmi.interactions.0.description","What are the colors in the French flag?")
```

Figure 2 - Setting interaction record values, using the SCORM dot notation

The SCORM binding

The dot notation

For historical reasons, SCORM API allows the getting and setting of values for atomic data elements only. It uses an odd "dot notation" to specify the name of each data element within a data structure. For example, the name "cmi.interactions._count" means "the count of interaction record instances of this instance of the cmi data model", where each "of" is represented by a dot and the order of the parts in the name represents the hierarchy of elements within the data model.

Beyond the dot notation for interaction data

However, transmitting data element by element is just too much a strain on performance for some of the complex data used in interaction records. For this reason, the ADL team designed a special way to represent a number of data values within the same string, using separators symbols that are highly unlikely to occur in the real world. This notation may seem a little strange, but it is very effective in allowing some of the more complex data for correct response or learner response information to be transmitted through a single transaction. See the SCORM runtime Environment specification for details.

This binding that is specific to SCORM is not mandated by IEEE Standard 1484.11.1 which defines the data model that SCORM calls "cmi" data model. In fact, IEEE Standard 1484.11.3 specifies an XML schema for the same data model. However, at the time the SCORM 2004 specification was being written, that XML standard did not exist. In fact, there was no reliable way to deal with XML in JavaScript and so using XML was not practical for the JavaScript based SCORM API. Even today, working with XML in JavaScript can be difficult for many people. In many cases, the ADL binding with its weird syntax for complex response data turns out to be just as easy to use.

However, beyond the API, translating the SCORM data to use the XML binding defined in IEEE 1484.11.3 makes a lot of sense, since there are a lot of tools and processes that can use XML data. For example, generating pretty reports from SCORM interaction data formatted in XML is quite easy with standard tools like XSLT. .

Mandatory vs. Optional elements

In a SCORM interaction record, everything is optional except the identifier and the data type. The identifier must be set first because it will be the key to associate the record with a particular interaction. The data type is required because the data models for learner responses and correct responses are different depending on the type of interaction.

For example, there is no requirement that an interaction record contain information about correct responses. However, if it does contain information about correct responses, the information must be organized exactly as specified in the SCORM data model for that sub-element and the type of interaction.

Chapter 5 - Basic interaction types

True-false

Overview

True-false interactions only allow two choices that can be represented as true or false.

User interface devices

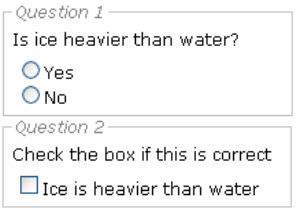

Figure 3 - Two devices for a true-false question

Because there are only two choices, the user interface device can be a familiar checkbox, with checked for true and unchecked for false, but there are many ways to render a True-False question and also many ways to label the choices. For example, the labels might be “Yes” and “No”, corresponding to the respective standard response identifiers “true” and “false”.

Instructional design considerations

In spite of their popularity because they are so easy to set up and score, True-False questions are often considered a poor assessment tool by many people because lend themselves too easily to random responses.

It may be important to be able to report as a response the label for the True-False choices, rather than just true or false. In that case, you might consider using a Single Response Multiple Choice interaction with two choices instead of a True-False interaction, because that allows you to use reflect the labels in the identifiers for the choices.

Interaction object

A True-False interaction is basically a simplified version of the Single Response Multiple Choice interaction object. It is simpler because the identifiers for its choices are predefined: true or false.

Interaction record

The learner_response in an interaction record of type `true_false` is either the token value `true` or the token value `false`.

Multiple Choice

Overview

Although in SCORM there is only one Multiple Choice interaction type, in practice there are two main flavors of multiple choice, which we'll call "Single Response Multiple Choice" and "Multiple Response Multiple Choice". Different authors have different names for these.

In a Single Response Multiple Choice, only one of the presented choices is correct. The choices are exclusive. The learner can only make one choice at a time. In "Multiple Response Multiple Choice", more than one of the presented choices may be correct, In fact, none of the choices might be correct. Because Single Response and Multiple Response are quite different behaviors for Multiple Choice, we will describe them in two subsections.

Single Response Multiple Choice

User interface devices

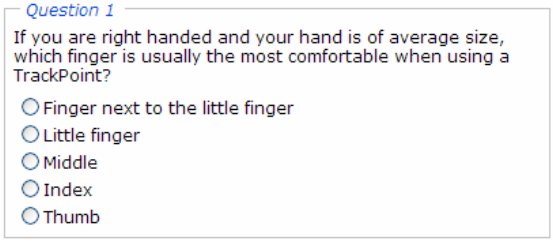

Figure 4 – Radio buttons allow a single choice

All kinds of user interface devices can be used to implement a Single Response Multiple Choice interaction. It could be a set of radio buttons, with the text or image for each choice next to each button in the set. Clicking a radio button registers a choice made by the learner. Clicking another one changes the choice. Sometimes the response can be registered by clicking anywhere on the choice text or image. Sometimes, a keyboard shortcut is associated with each of the choices. This facilitates input for users who are not proficient with a mouse or who have to use an alternative input device. Depending on the taste of the designer, a choice can be "locked in" so that changing the choice is not allowed, or the user may be free to make another choice until the results are submitted and finalized.

A "hot spot" is another typical example of a multiple choice interaction. Clicking on a region of the screen constitutes a choice. Clicking another region changes the choice. A well designed user interface design should provide some visual confirmation that the choice was made.

Other devices include drag and drop devices, drop-down choice interface widgets, voice activated menus, and so on.

A less frequently found device for multiple choice asks the user to type in a choice in a box. Because there is a high risk that the user will type in something that is not a recognized choice, this may be better suited for a Fill-in interaction.

Instructional design considerations

In spite of their popularity because they are so easy to set up and score, Single Response Multiple Choice questions are often considered a poor assessment tool because they tend to assess recognition rather than knowledge. Little cognitive processing is required to answer, unless the distracters are very carefully crafted to avoid pure recognition. A Multiple Response Multiple Choice is more intellectually challenging and just as easy to score in an automated environment.

Interaction object

The interaction object underlying a multiple choice question basically updates its state, including its current result, every time the user makes a choice. Initially, there is no response to report. It is up to the designer to decide what kind of result to report, if any, if there was no response. It is perfectly legitimate not to report a result if none is available.

Interaction record

The value of `learner_response` in an interaction record of type `multiple_choice` for a Single Response Multiple Choice interaction object is the identifier of the choice.

Multiple Response Multiple Choice

Overview

In a Multiple Response Multiple Choice interaction, zero or more among the choices presented constitute a correct answer. For example, a question might present a list of countries, only two of which are in Europe, and the correct response requires choosing both of those countries and none of the others.

User interface devices

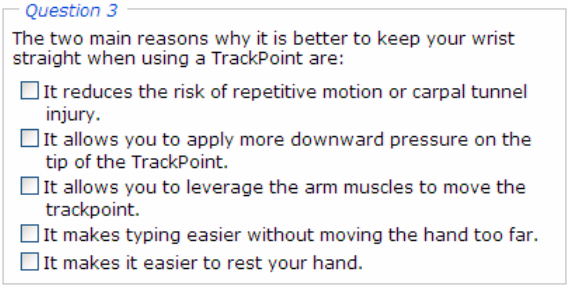

Figure 5 - Checkboxes allow several choices

The most common user interface device for a Multiple Response Multiple Choice interaction object is also familiar to any computer user these days: It is a group of check boxes, with the text or image for each choice next to check box in the set. Clicking a check box toggles whether or not it is registered as a choice made by the learner. Checked means “chosen” and empty means “not chosen”. Sometimes the response can be toggled by clicking anywhere on the choice text or image. The other user interface considerations are the same as for a Single Response Multiple Choice question.

A “hot spot” question can also be used as interface for the Multiple Response Multiple Choice interaction object, by allowing the user to select or unselect more than one of the choices.

Drag and drop devices are also useful interface devices. For example, a learner might be asked to drag screws to the holes in a panel that are designed to accept screws.

A rarely found version of multiple choice asks the user to type in a list of choices in a box. This can be frustrating for all concerned.

Instructional design considerations

Unlike Single Response Multiple Choice questions, Multiple Response Multiple Choice questions typically require a lot more thought from the learner, especially since wrong choice count against the good choices. Random choosing is therefore not a good strategy here because it is much more likely to result in a bad score.

Multiple Response Multiple Choice interaction objects are also useful outside of assessments, or when the assessment is purely functional as part of a task. For example, a learner might be asked to select three items to carry on a quest as part of a game, or to decide which tools to use for a task.

Interaction object

The interaction object underlying a multiple choice question basically updates its state, including its current result, every time the user makes a choice. Incorrect choices may be penalized to discourage random responses. Initially, there is no response to report. It is up to the designer to decide what kind of result to report, if any, if there was no response. It is perfectly legitimate not to report a result is none is available. The difficulty with evaluating this type of interaction object is in deciding when the user’s response is “complete.” A simple way is to just associate a “submit” button or similar device to click after completion. Another way is to consider the interaction complete automatically if the learner moves to another page or starts answering another question.

Interaction record

The value of `learner_response` in an interaction record of type `multiple_choice` for a Multiple Response Multiple Choice interaction object is a collection of zero or more identifiers, one for each of the chosen choices.

Fill-In

Overview

The response in a Fill-In interaction is typically a string of characters typed in by the user. The SCORM Fill-In interaction type allows for a response that consists of more than one string. This is useful when, for example, the learner must provide the missing words in a sentence.

User interface devices

Fill-In user interface devices typically consist of text fields in which the learner is asked to type a short string.

For example, a typical Cloze test is a text with blanks that the learner is expected to fill by typing in the missing words or fragments of sentence.

Other devices that work with this type of interaction object include crossword puzzles and any other device in which the user is expected to provide a constructed answer that consists of one or more short chunks of text.

Depending on the target learner capabilities and the purpose of the interaction, it is often necessary to ignore capitalization, extraneous spaces, and sometimes even the order of words or the spelling when the actual device prepares the learner response for evaluation.

An alternative interface device that does not require typing might ask the user to select words out of a text. The user might even be invited to drag the words that seem relevant into the "blanks" that contain the response elements. This interaction type may be more appropriate for this kind of interactive device because, unlike a Multiple Choice interaction type, it reports as learner response the actual chunks of text rather than identifiers of the chosen distracters.

Instructional design considerations

Constructed answers are typically considered more valuable than choices because they require more elaboration on the part of the learner. However they can sometimes be more difficult to evaluate. For example, a user might be using a different but unexpected term with the correct meaning.

Another problem is with poor spellers. Unless the test is also a spelling test, users who misspell the expected strings will be penalized. Working around this involves some complicated ways to compensate for poor spelling, such as trying to recognize what was typed and asking the learner for confirmation if a match seems likely.

Interaction record

The value of `learner_response` in an interaction record of type `fill_in` is an ordered collection of zero or more character strings.

Long Fill-In

Overview

A Long Fill-In allows a longer response than the normal Fill-In. Unlike the Fill-In interaction type, it provides for only a single response in the form of a string of characters.

User interface devices

The typical interface device for a Long Fill-In is a text field.

Another device might be a selection tool that allows the learner to select a relevant passage from a long text.

Instructional design considerations

The Long Fill-In interaction type may not be appropriate for fully automated online use, since evaluation of a long textual answer such as a short essay is difficult. Automated essay evaluation tools do exist, but they tend to be server-dependent and thus not suitable for inclusion in a SCO. So, this kind of interaction is probably best suited to collect text that will be evaluated later by someone who reviews the tracking information collected by SCORM.

However, there are other potential uses that lend themselves to simpler forms of automated evaluation, such as programming exercises or short text translations.

Interaction object

The interaction object for a Long Fill-In interaction usually process the text input by the learner according to various rules. This may include capitalization, suppressing extra white space, spelling check or various forms or smart matching with expected responses, and so on. Some of those rules can be reported in Correct Response information in the interaction record.

Interaction record

The value of `learner_response` in an interaction record of type `long_fill_in` is a character string.

Numeric

Overview

A Numeric interaction type produces a numeric response in the form a single numeric value.

User interface devices

Typical interface devices for a numeric response include a text field, a slider or a dial.

Numerous other devices can also produce a numeric response. For example, they might include the tension applied to a simulated spring, a "click" counter actuated by the learner, a counter that counts music beats and that stops when the user clicks a button, the single result of an equation that the learner must construct, the output from a process in a simulation, and so on.

Instructional design considerations

If the user interface device uses direct manipulation of an analog scale and the range of valid responses range is not continuous, but consists of a set of specific values, some "stops" or "notches" might be provided in the user interface, similar to the way the tab icons stop on invisible "notches" in the ruler in the Microsoft Word user interface.

Interaction object

The interaction object for a numeric interaction might do some processing and support more or less explicit feedback. For example, depending on whether the numeric value is constrained to a range, or the value is expected to be a real number or an integer, the interaction object may perform filtering or data shaping.

Interaction record

The value of `learner_response` in an interaction record of type `numeric` is a real numeric value represented as a string. The precision of such a value is constrained – see the SCORM documents.

Likert

Overview

A Likert interaction produces a response in the form a single value in a specified range or scale. The range may be discontinuous and allow for only specific values, for example integer numbers in the range 1 to 5, or values ranging from “strongly disagree” to “strongly agree”.

User interface devices

The same user interface devices used for an interaction of type multiple choice or numeric can be used for Likert. However, traditionally the user interface device will consist of a number of discrete choices that correspond to specific values within a predefined range. Sometimes a slider is used, to allow finer values, or to make the user interface more interesting.

Instructional design considerations

Although a Likert question is typically used to measure opinions and is thus not judged, SCORM does allow you to treat it as a marked item and to associate a correct response with it. This can be useful if you use a Likert question to represent a learner’s judgment of probability before triggering an experiment, for example.

Interaction object

Each discrete value is represented by an identifier. If the interaction device uses a continuous scale as an input device, the interaction object typically rounds up the input value into an allowable value for which there is a specific identifier.

Interaction record

The value of `learner_response` in an interaction record of type `likert` is a single identifier.

Matching

Overview

In a matching interaction, a learner must match one or more pairs of something. The pairs are not necessarily exclusive. For example, given the elements A, B and C on one side and the elements 1, 2 and 3 on the other side, SCORM does not preclude combinations of pairs such as A1 and B1 and C1. Of course, the designer of the interaction device and interaction object can make those exclusive. For example, the designer might disallow B1 if A1 is already paired.

User interface devices

Many possible user interface devices can be used for matching. One of the simplest one presents a series of labeled visual elements, and asks the user to type in the labels in text fields that are visually associated with other visual elements.

Other user interface devices may use drag and drop, asking the user to drag a line from the first element in a pair to the corresponding elements.

Another form of drag and drop makes the user drag a visual element to a particular position. For example, a user might be asked to drag values into placeholders in the graphic representation of an equation.

Instructional design considerations

Interesting uses for matching are not limited to discrete, independent elements. For example, a task simulation might ask a learner to select from a group screws in two different sizes, and drag the screws to the holes in an instrument panel, making sure that a long screw is put where a long screw is expected and a short screw is put where a short screw is expected. There may be more than one long screw or more than one short screw, and it does not matter which one is chosen as long as it is the correct size. In that case, the same identifier may be used for all the long screws and a different identifier for all the short screws, while each hole has its own identifier.

Interaction record

The value of `learner_response` in an interaction record of type `matching` is a collection of pairs of identifiers.

Sequence

Overview

A Sequence interaction captures a sequencing operation by a learner; in other words, it captures the result of a task in which a learner arranges things into a predefined order.

User interface devices

User interface devices for Sequence interactions may be as simple as a text field in which the user is asked to type labels in a particular order. At another extreme, it might be as complex as a game or simulation in which the user must do some tasks in a particular order. If the tasks have associated tracking information, a Performance interaction may be more appropriate. But if all you need is to record the order the Sequence is more appropriate.

A more typical user interface device for the Sequence interaction involves drag and drop. The learner is asked to arrange elements by dragging them into the proper position in some kind of graphic alignment device. For example, a user might be asked to arrange the words in a jumbled sentence, or to arrange historical events on a timeline.

Instructional design considerations

Sequence interaction can be a very powerful learning and testing device. The sequencing dimension may be time, importance or any other arbitrary order. For example, task planning often involves arranging the tasks into a suitable sequence before undertaking the work.

Interaction object

The interaction object for a Sequencing operation might be part of a simulation.

Interaction record

The value of `learner_response` in an interaction record of type `sequence` is an ordered list of identifiers. The order represents the order specified by the learner.

Performance

Overview

The Performance interaction is the most flexible and rich of the standard interaction types in SCORM. It allows the capture of a number of arbitrary steps performed by a learner, along with information about every step.

User interface devices

The typical interface device for a Performance interaction is usually not visible to a learner as a question or test item. Instead, it may be, for example, a simulated software user interface, a game, or even a flight simulator.

The user interface device for a Performance interaction might also look just like the user interface device for any other interaction types described above. However, instead of just recording a simple response, the performance interaction might record the steps leading to the response or every time the learner changed a response.

Instructional design considerations

Imagination is the main limit in the use of the Performance interaction type. It is particularly useful in the paradigm of "learning by doing" because it supports observation of how the user performs a task, with a standardized way to report on how the task was performed regardless of the type of task.

If this type of interaction is graded, it is possible to specify multiple performance patterns as part of the standard correct responses data provided by the designer.

Interaction object

The interaction object for a Performance interaction typically manages or observes the performance of a task or workflow. For each significant event, it adds a step record to the interaction record

Interaction record

The value of `learner_response` in an interaction record of type `performance` is an ordered collection of up to 125 performance steps, with additional data items associated with each step. For each step, you can specify one or two of the following: An identifier for the step and either a short string of text or a numeric value. For example, you might be able to report how a learner went through a series of adjustment of various controls in a simulated control panel; for each step, the identifier of the control and the setting can be reported. Or you can use a Performance interaction record to report the steps taken by a user in solving an equation.

Other

Overview

The Other interaction type is provided as a placeholder to allow a designer to specify any kind of interaction that does not fit the standard types, and to provide data about such interactions using the standard data model for interactions.

User interface devices

User interface devices for an interaction of type other can be anything.

Instructional design considerations

An Other interaction can implement just about any design, only limited by the imagination and the size allowed in an interaction record for the correct response or response information, if those are reported through the SCORM interface.

Because no standard governs the representation of the data for an Other interaction, I would strongly recommend to first try to fit what you need using one of the predefined interaction types. In many cases, it is also possible to combine multiple complementary interaction objects to provide richer views of a particular interaction, test or learning process through groups of interaction records.

Interaction record

The value of `learner_response` in an interaction record of type `other` is a character string. I recommend that application developers encode that character string to be XML safe and script evaluation safe. This can prevent accidental data loss or unwanted side effects when the string is being passed around through the SCORM interface, in a LMS, or in reports.

Chapter 6 - Special application scenarios

Interaction monitor object

Overview

An interaction monitor object monitors some other interaction invisibly and creates its own interaction record to report on what it observed. This is best illustrated through an example.

Example

Imagine the matching interaction example cited above, where the user has to match the correct size screws with the holes for an instrument panel. Let us say that this is implemented as a semi-realistic simulation, and the learner drags the screws from a bin to the holes in the panel, and that the interaction device is also used as a learning device, by rejecting any screw that is the wrong size. In the final evaluation, the designer wants to be able to tell not only whether all the matches are correct when the learner finishes the exercise, but also whether the user tried some incorrect matches.

The main interaction object implements the matching functionality and reports the matches and the result of the evaluation of the matches through an interaction record of type `matching`. The interaction monitor object, which the learner never sees, records each attempt at a match as a step in the response value of an interaction record of type `performance`. For each attempt at a match, a new step is added to the response, and a short textual description describing the attempted match and its result is added to the step.

Depending on how the results of the exercise are evaluated, one might be interested only in the final result for the matching question, which is contained in the primary interaction record: Does each hole have a screw of the correct length? What is the result? If an evaluator wants to check on what actually happened, the information can be found in the interaction record created by the interaction monitor object: Which steps did the learner actually perform, and what was the result of each step?

The questions can be related to each other for reviewers by using the description element. For example, in the interaction record for the monitoring object the description might say something like "`Steps taken by learner in question xxx`" where `xxx` is a human readable identifier for the main question. The interaction identifiers in the interaction records could also contain a common fragment, so that for example the identifier for the main interaction record might be "`Q123345-Status`" and the identifier for the complementary interaction record might be "`Q123345-Steps`"

Complementary interaction object

Overview

A complementary interaction object works with another interaction object to provide richer functionality. This is also best illustrated through an example.

Example

As mentioned above, Confidence Based Marking requires some indication from the learner of how confident the learner is that a response to a question is correct. This confidence value is then used in the calculation of a score for the question. Setting this up in an application is easy. Let us look at a practical example in more detail.

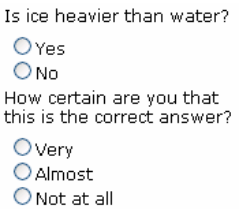

Figure 6 - Interaction device for CBM

In this example, the main question is a True-False question, for which the correct answer is true. When the question is presented to the learner, the learner is also prompted to indicate how confident he or she is that the response is correct.

This may result in two interaction records. One interaction record is of type `true_false`, and the other is of type `likert`. The `true_false` record includes the learner response (true or false) and a result, which is a score calculated by taking into account the response chosen in the complementary interaction object. The second record is of type `likert` and includes the learner response. Here, the value of `learner_response` is an identifier for a specific confidence level. For example, the identifier can represent a step on a scale ranging from `confidence_0` to `confidence_3`. However the second interaction record does not include a result because no judgment or marking is done on the confidence question.

In practice, the second interaction record is not even necessary if all one cares about is the result of the marking of the True-False question; however, it may be useful to someone who is interested in analyzing confidence levels expressed by the learner for various questions.

The interaction records can be related to each other for reviewers by using the description element. For example, in the interaction record for the confidence level interaction the description might say something like "`Confidence level for question xxx`" where `xxx` is a human readable identifier for the main question. The interaction identifiers in the records could also contain a common fragment, so that for example the identifier for the main interaction record might be "`Q123345`" and the identifier for the complementary interaction record might be "`Q123345-Confidence`"

Chapter 7 - Suspend and resume

Suspending and resuming SCOs with interactions

Overview

SCORM uses the concept of learner sessions and learner attempts. A learner attempt may involve one or more learner sessions. When a session ends without ending the attempt, the SCO used for the session is unloaded until the attempt can be resumed later, in another session. When it finishes a session, a SCO can ask the runtime environment to store some suspend data that will be available later if the same SCO is launched in a resumed session. This will be arbitrary data that only the SCO itself can interpret. But the runtime environment will also automatically suspend most of the other data values in the SCORM cmi data model. This includes interaction records. Note that this feature was not available in SCORM 1.2.

Of course, a designer is not obligated to enable suspend and resume in a SCO. For example, many assessments are designed so that they must be completed in a single session.

Suspending and resuming interaction records

To indicate that it wants to store suspend data, a SCO use SetValue in the SCORM API to set the data element `cmi.exit` to "`suspend`" at any time before terminating the communication session with the SCORM runtime environment. Any interaction record that was been created by using the appropriate SetValue statements will also be suspended. When a SCO is launched, it can get the value of the data element `cmi.entry`. If this value is "`resume`", then the SCO can assume that any previously suspended data elements are available, and that includes the interaction records. The SCO developer doesn't need to do anything special for this to happen, other than setting `cmi.exit` to "`suspend`" and getting the value of `cmi.entry` to check whether the SCO is resuming from a previous session.

Suspending and resuming interaction state

An interaction record often does not contain all the information required to restore the state of an interaction device. For example, the interaction device for a multiple choice question might use randomized distracters. There are no data elements in the SCORM interaction data model to represent the distracters. This is where the data element `cmi.suspend_data` is useful. A SCO developer can store all kinds of information into the value for that data element.

Suspending and resuming random tests

Many tests are designed to draw test item randomly from a test item bank. If one wants to suspend and resume this, there are two straightforward options.

One is to create an interaction record for each test item, even before the learner has a chance to respond, and then update the records as the learner responds. When resuming, the SCO can walk down the list of interaction records and use the interaction ID values to look up the test items to use. The other option is to simply store a list of the test items in the cmi.suspend_data element. When resuming, the SCO can walk down that list and know which test items to use.

If when the SCO is launched it determines that it is not resuming from a previous session, it simply does a random drawing from the test item bank. Of course, both options assume that the selection of test items does not take place until after the SCO has had a chance to detect whether it is resuming from a previous session or not.

Appendix 1 - Resources

Designing online tests and interactions

There are many resources for online test and interaction designers. Those resources reflect a wide variety of philosophies and priorities. A web search on the terms “designing online tests” and “designing online interactions” will return a number of impressive articles along with the usual chaff of vendor pitches. Some older articles that predate SCORM are still very valid from a design perspective.

The plans for future editions of this eBook include the addition of links to functional examples illustrating the concepts introduced in the text.

SCORM specification documents

This book is based on the SCORM 2004 3rd Edition specification documents as of November 16, 2006. Get the most current version of the SCORM document set at http://www.adlnet.gov/scorm/

Third party SCORM primers and documentation for SCORM 2004

Ostyn, Claude. *The Eye of the SCORM* (2006). Retrieved 22 January 2007 from http://www.ostyn.com/resscormtech.htm

Rustici, Mike. *SCORM 2004 Reference Poster* (2006). Retrieved 22 January 2007 from http://www.scorm.com/pages/resources.aspx

Sample SCORM related scripts

Ostyn, Claude. *ostyn2004sco.js -- Generic reusable script for SCORM 2004 conformant SCOs* (2006). Retrieved 22 January 2007 from http://www.ostyn.com/standards/scorm/samples/ostyn2004sco.js
This sample script provides a library of functions and default behaviors for SCOs. It is documented in an appendix of *The Eye of the SCORM*.

Standards and specifications reference

IEEE 1484.11.1-2004 IEEE Standard for Learning Technology — Data Model for Content to Learning Management System Communication (2004). Piscataway, NJ: IEEE.

IEEE 1484.11.3-2005 IEEE Standard for Learning Technology - Extensible Markup Language (XML) Schema Binding for Data Model for Content Object Communication (2005). Piscataway, NJ: IEEE.

RFC 2396 - Uniform Resource Identifiers (URI): Generic Syntax (1998). Retrieved 1 November 2006 from http://www.ietf.org/rfc/rfc2396.txt

DHTML scripts to help JavaScript experts build rich interactions without plug-in dependencies

Zorn, Walter. *JavaScript: DHTML API, Drag & Drop for Images and Layers* (2004) Retrieved 1 November 2006 from http://www.walterzorn.com/dragdrop/dragdrop_e.htm

Content developer resources

LSAL SCORM Best Practices Guide for Content Developers (2005.) Retrieved 1 November 2006 from http://lsal.org/lsal/expertise/projects/developersguide/

The Reload Project. *Reload Editor 2.5* (2006). Retrieved 22 January 2007 from http://www.reload.ac.uk/editor.html
This is a Java based open source editor for SCORM packages.