

Conceptual Architectural Framework

Scott Thorne

2/15/04

How do we communicate the relation and use of all the open specifications?

At the present time the large community of educational systems and content providers and their users are having trouble understanding the complicated specifications world. To help in the understanding of this area, we can use a general architectural pattern and functional area breakout to help organize and give a context to this information.

Producing a matrix, where we list the functional area and what specifications exist for each of the three elements is one way of doing this. This would allow someone to easily identify what specifications exist in his or her area of interest. It will also help identify areas which existing open specifications are missing. As we continue to make progress in open specifications, we would hope that we would get coverage in each of these, so that all types of interoperability were supported.

In order to create the communications device described, we first need to agree on a particular factoring of the problem domains as well as a pattern to express the general systems architecture.

Architectural Pattern

A common architectural pattern is emerging in the educational domain. It consists of a common factoring of the problem in combination with three elements.

- Data Representations
- Communications; Transport and Protocols
- Interfaces; (Contracts & Boundaries)

Open specifications can be produced for each of these three elements, each providing particular interoperability and integration benefits. Specifications in each of these areas are likely to change over time, so making these independent of each other helps both to allow different combinations to be used in different situations and allows the parts to evolve while limiting the scope of impact.

Most systems use a combination of these elements. Although this pattern is abstract and by itself doesn't provide for interoperability, a high level architectural pattern could help people understand how all the individual specifications surrounding educational systems fit together. This would provide some common context for interpreting the individual pieces and promote adoption for all the specifications.

Creating a model of a specific business process in each functional domain will provide common terms and concepts from which each of the three types of specifications can be produced. Then in any particular functional domain, such as K-12 test assembly, a particular system or implementation can use some combination of the three types of open specifications to achieve a specific interoperability goal.

Current specification organizations concentrate on individual problem areas. For example; until recently IMS produced data specifications, OKI works only on interface specification, Library organizations have produced both data specifications such as Dublin Core, as well as Protocol specifications such as Z39.50 and SRW.

Multiple specifications may exist simultaneously for one of these elements even in a single functional domain. For example, POP and IMAP are both protocols for retrieving mail. Dublin Core and METS are both data specifications of metadata for content. For systems to be flexible it is important for them to deal with these variations. Typically you would

want to decide where you could rely on a single specification, and where it might be important to support or use more than one.

Data Representations

How do you represent & interpret data?

Data representation covers; the structures, meaning, and particular vocabularies associated with data. Current examples of this are; IMS CP, SCORM, QTI, Dublin Core, Eduperson.

Open data specifications play an important role in the overall pattern; since they need to be pinned down to get real interoperability using either of the other two elements (interfaces or protocols).

Having an agreed on data specifications provides a certain amount and type of interoperability. It allows two systems to import and export data.

Communications; Protocols and Transport

How do two systems talk to one another?

Open communication specifications around define the manner in which two systems or machines communicate with each other. Current examples of this are: SOAP, Z39.50, SRW, POP, and IMAP.

Having open communications specified, enables two parties to communicate without any prior configuration or agreement. If a consumer is running an application, which supports a particular protocol, they know they will be able to communicate with another system or service provider if they also use this protocol. Agreement on an open protocol makes real-time communication possible without special integration work or configuration.

Software Interfaces

How does a system access a related system function?

Interfaces are the contract between pieces of software. An interface is the contract that defines and separates the responsibilities between the two sides. This abstraction provides application developers protection from underlying system changes. Current examples are OKI OSIDs, WSDL...

Interfaces in this context means a two-sided contract of function, which both the provider and consumer agree to; not just the calling interface of a particular piece of software. Interfaces are currently programming language dependent. They are commonly used to integrate software on a single machine, but can also provide a common mechanism to handle variations in underlying communication techniques.

Using open interface specifications allows the use and substitution of different service implementations and user facing applications. Interfaces can be used to cover up underlying changes in technology. For example, you might need to use several different protocols with similar function. Having a common interface, which could call either of these, allows the calling software to not know about this variation. Interfaces help software providers by specifying an integration point that they each can build to. This allows software to be developed and easily moved and deployed into a different environment with different technology.

Factoring and Domain Models

How can we break the problem space into commonly agreed parts?

A large part of the problem is factoring the problem space in a consistent manner. There seems already to be a fairly common factoring into functional domains. Once this factoring is agreed to, models can be used to express the common conceptual understanding of a problem domain. These models once created will make it easier to create future specifications both for each specification element as well as for particular market sectors.

In addition to the factoring of the educational area, educational software needs to interact with other non-educational systems usually in an enterprise context. Some of these element specifications will be provided from outside the

educational domain. Examples: Security, Library, HR, and Financial. It is important to factor the educational specifications so that they can take advantage and work well in combination with specifications in these other domains.