

Schema editor as central design tool: the xVRML experience

Jeffrey Sonstein
CME Group – IT Department
Rochester Institute of Technology
Rochester NY 14623
+1 585-475-7315
jeffs@it.rit.edu

ABSTRACT

The xVRML Project grew out of an industry-based effort to develop a next-generation XML-based 3D markup language for the World Wide Web. Each major stage in the life cycle of a complex design and development project such as xVRML requires the use of a different “web of tools.” This may be thought of as: the interconnecting and cooperating set of applications used to produce a particular set of software and documentation outputs. Each particular stage of a project will tend to revolve around a different “central tool.” Understanding this aspect of the design and development process can help a team choose the right tools for each stage. The author reviews his experiences with the xVRML Project and suggests some generally applicable lessons learned.

Categories and Subject Descriptors

I.7.2 [VRML]: Language Constructs and Features – *abstract data types, polymorphism, control structures*. This is just an example, please use the correct category and subject descriptors for your submission. The ACM Computing Classification Scheme: <http://www.acm.org/class/1998/>

I.7.2 [XML]: Language Constructs and Features – *abstract data types, polymorphism, control structures*. This is just an example, please use the correct category and subject descriptors for your submission.

General Terms

Management, Documentation, Design, Reliability, Human Factors, Standardization, Languages.

Keywords

XML, VR, virtual, reality, modeling, languages, 3D.

1. PROJECT LIFE-CYCLE STAGES

Like any complex design and development project, the xVRML Project has a definable life cycle which can be broken down into stages. Each major stage may require the use of a different assortment of tools. The design and development process proceeds along a smoother path if the tools used in a particular stage can cooperate with each other.

1.1 The “web of tools”

Design and development projects vary greatly in complexity. Both simple and complex projects have a few things in common, including the use of design and development tools. These tools might include such things as an IDE (“integrated development environment”) used by project programmers and a UML (“unified modeling language”) diagram generator used by project designers.

As a project gets more complex however, the number and complexity of the tools used can increase dramatically. Consciously or unconsciously, those working on a complex project will often try to select tools that “play together well.” The need for outputs from one or more project tools to be useable as inputs to one or more other project tools and stages often grows as project complexity grows. Without at least a little integration, excessive effort may be needed to accomplish this “wiring together” of design and development tool outputs and inputs. Many of us have vivid memories of working on projects with seemingly endless meetings just to try to join together the scattered inputs and outputs of various project roles.

A thoughtfully constructed toolbox of interconnecting and cooperating applications can be used to produce needed software and documentation outputs with minimal wasted energy. If you were to create a diagram of the relations between the tools used within a particular stage of a project, the result would often look like a “web” or “network.” This web of tools may be defined as: the interconnecting and cooperating set of applications used to produce a particular set of software and documentation outputs.

As a project progresses, there is often a “flow” of team focus and energy from one part of the design and development process to another. This can be seen as a set of project life cycle stages. Each stage in the life cycle of a complex design and development project may require the use of a different web of tools. Any particular stage will probably require that outputs from one tool become inputs to another, ultimately leading to a particular set of outputs from the stage becoming the inputs to another stage.

1.2 The “central tool”

Each web of tools required by each major stage in a project may tend to revolve around a different “central tool”, producing output(s) which the rest of the tools for that stage depend upon to do their work. Understanding this aspect of the design and development process can help a team choose the right central tool and combination of tools for each stage.

On a high level of abstraction, a project may be thought of as having a design stage and a development stage. These stages cannot be rigidly delineated, for things discovered during the development stage are often used as feedback to complete the design phase. Nonetheless, this can be a useful abstraction for work-flow analysis.

The design stage is often focused on data and metadata. For this reason, the design stage of any complex project may often require a “data-centric” approach. This is probably why the central tool for the design phase of the xVRML Project has turned out to be data-centric: the Schema editor used to define the structure and constraints for the data in an instance document.

2. THE xVRML PROJECT

The xVRML Project grew out of an industry-based effort to develop a next-generation XML-based 3D markup language for the World Wide Web. A system for describing, loading, and persisting the state of 3D instance documents (“worlds”) is bound to end up being data-centric. Such a system requires software and documentation outputs and instance documents which are readable and understandable by *both* machines and humans.

Content creators need to be able to read and understand documentation for the language and sample instance documents. Machines need to pass tree-structured and fairly tightly-constrained data to other machines. XML can be a very good choice for such a situation. The primary software technologies used to define an XML-based markup language are DTD (Data Type Definition) and XML Schema.

XML Schema-based markup languages are particularly good at providing unambiguous structure and constraints to validate data from an instance document. An XML Schema-based markup language can be created which both humans and machines can read and understand. For these reasons and others, XML Schema was chosen as the data-structuring and –constraining software technology for the xVRML Project.

2.1 Background

The first widely-accepted attempt to develop a modeling language for virtual reality on the internet was VRML which later became ISO/IEC 14772 (also known as “VRML97”) [1]. As was true of most modeling and markup languages of that period, the form of VRML was fairly idiosyncratic. For a machine to read in a VRML file, you needed to create a specialized parser just for VRML. For a human to read or create a VRML file, you needed to learn the particular formats.

An industry group (the Web 3D Consortium) formed in the late 1990s, one goal of which was to develop “a powerful and extensible open file format standard for 3D visual effects, behavioral modelling and interaction” [2]. The development of this new specification was delegated to a “Contributors Group” on which the author served, and that group decided to develop an

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Conference '05, Month 1–2, 2004, City, State, Country.
Copyright 2005 XXX n-nnnnn-0/00/0005...\$n.nn.

XML-based file-format.

XML was chosen for the new file-format because of its portability, extensibility, and readability. Generic XML tools can be used to load, parse, validate, and persist any XML-based file. Properly structured XML is human-readable by anyone understanding how to read XML in general. XML Schema was judged by the majority of the “Contributors Group” to be early in its development, so the group made a decision to use DTD to define the new specification. The author wrote the first “straw man” DTD for the group, and in the process discovered some of the limitations of DTD.

2.2 DTD Limitations

Two problems with DTD became apparent almost immediately. One problem is that DTD uses a different notation than the XML it describes. This means that someone creating a new specification for an XML document type must learn both the notational system used in DTD and the notational system used in the XML described by that DTD. As you may imagine, this can add severely to the learning curve for a designer. Another problem is that DTD allows one to specify the structure of the data but not the constraints to be placed on the data. This can add to the instance document validation which needs to happen at the application level after the document has been loaded and validity-checked against the DTD. “Post-validation” overhead can become quite steep due to the limitations on data constraints in DTD.

2.3 Advantages of Schema

XML Schema has two main advantages over DTD in specifying what is legal in an instance document. For one thing, Schema uses the same notation as the XML it describes. An XML Schema is written in XML. If you have learned to read and understand XML, then you already know how to read and understand Schema.

For another thing, Schema allows the designer to specify not just the structure of, but also the constraints for, the data within a legal instance document. A Schema-based document can require significantly less post-validation work by the application as a result, and errors in the data can be caught closer to data input/creation. Using Schema helps minimize learning curves and maximize data structuring/constraining capabilities.

The Web3d Contributors Group process eventually did develop a DTD for “X3D” to replace VRML97 [3]. As was predictable, the post-validation overhead for this DTD is also quite steep and the DTD itself is quite difficult for humans to read and understand. Because of these problems, the xVRML Project was born.

xVRML was planned from the start to be based on XML Schema. This allowed the development of a more human-readable specification. This also allowed the development of a specification with fairly tight data-entry constraints which can be checked against the Schema itself when an instance document is created and validated.

Many XML editors are now “Schema-aware.” Because Schema allows tight constraining of data, the document editor itself can provide a content creator with immediate feedback when trying to insert an element or attribute in the wrong place or in the wrong way. It is a long-understood maxim of data management that: the closer content validation happens to content creation the lower the error rate will be for that content. The experience in this project

has been that a combination of Schema for defining and constraining data and a Schema-aware editor for entry of data leads to “clean” instance documents and “clear” specifications.

2.4 Project goals and objectives

One key step in any design process is identifying the goals and objectives for the project at hand. The goals of the xVRML Project are: (a) to design and develop a specification for an XML Schema-based 3D markup language for the World Wide Web, (b) to construct at least one demonstration implementation of a viewer, and (c) to develop sets of documentation aimed at both programmers and content creators.

The Schema is the central output for the first stage, and it is “more or less” finalized. The web of tools for the first stage of the Project has included a Schema-aware XML editor, a build tool, a compiler, two documentation extractor/generator tools, and a data-binding tool. The specification documentation for content creators and programmers, the data structures for the sample implementation code, the documentation for the sample implementation code, and sample instance documents all depend upon the Schema.

All the other tools depend in one way or another upon outputs from the Schema-aware XML editor, and require XML editor outputs (or outputs derived from them) as inputs at this stage of the Project. In our web of tools model, the Schema-aware XML editor may be thought of as the central tool for this stage. A good one will help you achieve your goals and objectives with less work.

2.5 Project Status

The Schema is almost complete and is currently in “beta testing” status. By beta testing I mean that it is being tested by domain experts (both content creators and computer graphics programmers) from outside the Project. The Schema itself, example Java classes to represent an instance programmatically, and documentation for the Schema and the Java classes in HTML form are available for download from SourceForge [4] and posted online at the xVRML Project site [5]. A sample viewer application for Linux and Mac OSX platforms is available for download from the Carina Project site [6].

As the design phase starts to wind down, focus is shifting more towards the demonstration implementation phase. At this point, the design of the Schema is being driven by two things: (1) what is discovered in the process of creating example instance documents, and (2) what is discovered in the process of creating a sample implementation. Shifting focus from design to development will likely require rethinking the current web of tools.

3. AUDIENCES, DEPENDENCIES, AND FEEDBACK MECHANISMS

As you can see, the two main audiences for the xVRML Project are programmers and content creators. Both groups need to be able to read and to understand project outputs. The project Schema is the central output for the design phase. Source code and documentation are dependent upon the Schema. Both

programmers and content creators need to provide feedback at numerous points during the design phase, through examining draft Schemas and example instance documents.

Programmers appear to focus on the low level details of the data being provided by the Schema-as-specification. From their comments so far in the life of this project, they need to see if the data and data-types and data-structures they expect to find to do their work are present in the Schema, and if the constraints they expect to find are in place. They compare the data structures and constraints of a new specification with what they already know in that domain. They seem to ask themselves: “does this look reasonable to me as a graphics programmer?”

Content creators appear to focus on the low level details of example instance documents. Not surprisingly, they provide better feedback when even a partially-operational viewer application is available. They compare the structure and content and constraints of instance documents with their other experiences in the domain. They seem to ask themselves: “does this look reasonable to me as a 3D artist?” They ask themselves basically the same question as the programmers, they just ask it from their own perspective. The “looks reasonable” test seems to me to be a very reasonable gauge of progress while shifting focus from design to development.

4. SUMMARY

The xVRML Project grew out of an industry-based effort to develop a next-generation XML-based 3D markup language for the World Wide Web. As a complex design and development project it has required the coordinated use of a varied set of cooperating software tools. As a data-centric project it has required a focus on data structure design and constraint handling during the early stages of the project. As a project using Schema-based XML data the central tool in the web of tools used during the design phase has been a Schema-aware XML editor. In a world of practice where XML-based data is used more and more, there seem to be some generally applicable lessons in our experiences so far with the xVRML Project.

5. REFERENCES

- [1] *The Virtual Reality Modeling Language*. (n.d.). Retrieved January 19, 2005, from <http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-IS-VRML97WithAmendment1/>
- [2] *Web 3D Consortium*. (n.d.). Retrieved January 19, 2005, from <http://www.web3d.org/>
- [3] *Extensible 3D (X3D) Document Type Definition (DTD) x3d-3.0.dtd*. (n.d.). Retrieved January 19, 2005, from <http://www.web3d.org/specifications/x3d-3.0.dtd>
- [4] *SourceForge.net: Project Info – xVRML*. (n.d.). Retrieved January 19, 2005, from <http://sourceforge.net/projects/xvrm/>
- [5] *The eXtensible Virtual Reality Modelling Language*. (n.d.). Retrieved January 19, 2005, from <http://www.xvrm.net/>
- [6] *entigo.website:carina:main*. (n.d.). Retrieved January 19, 2005, from <http://entigo.rh.rit.edu/?page=carina:main>