Open XML Requirements Specifications, a Xylia based application

Naeim Semsarilar
Dennis K. Peters
Theodore S. Norvell
Faculty of Engineering and Applied Science
Memorial University of Newfoundland

November 12, 2003

Abstract

System designers have one goal in common: the system must behave according to the design requirements. Designers need to succinctly specify the behavioural requirements of a system. Peters in [1] proposes a technique for documenting system requirements in which the required system behaviour is described in terms of the environmental quantities that the system is required to observe and control; these are modeled as functions of time.

This paper presents OXReq, a tool, based on Xylia [2], for authoring and processing of requirements documents. This tool employs two XMLs [3]: one for storage, analysis, and verification of requirements documents, and another for presentation to and editing by human users. The grammars for these XMLs are defined by document type definitions (DTDs) [3]. Mathematical and tabular expressions are integral parts of requirements documents. OXReq builds on OpenMath [4] for encoding complex mathematical and tabular expressions in XML.

Using Xylia, XSLT [3], and a JavaCC generated parser, this tool provides the seamless transition from storage XML to presentation XML and vice versa. The plug-in mechanism of Xylia allows complex tasks to be performed on requirements documents, such as verification of internal consistency, completeness, and other properties of the documents, and generation of oracles.

1. Introduction

A key early step in good engineering practice for the development of most systems is that a system requirements document be produced. In the case of computer based systems, this will include the *system behavioural requirements*, which give a clear, precise and unambiguous description of the required behaviour of the system in terms of the environmental (i.e., external to the system) quantities that the system must either monitor or control. [1]

Increasing system complexity, coupled with a need for a high level confidence that the system is correct, dictate that analysis and verification tasks be at least partially automated. This can only be accomplished if the requirements are written in a notation that has a precise syntax and semantics, so that it is amenable to computer processing.

XML (Extensible Mark-up Language) [3] is a growing standard for storage and communication of data. Data in XML is stored as tagged plain text, which allows for cross-platform compatibility. XML is not a language by itself. Rather, it is a basis for creating domain-specific languages. In this paper, we discuss an XML based language, OXReq, that can be used for storage and manipulation of requirements specifications. Encoding requirements documents in XML format is beneficial because it facilitates easy communication of documents, and unifies the syntax and grammar for specifying requirements.

Xylia [2] is a toolkit that can be used to produce Java-based, application-specific WYSIWYG (what you see is what you get) XML editors. It provides application developers with all the common features of an editor (clipboard operations, undo and redo, etc.), as well as a plug-in interface that can be used to extend the functionality of editors at run-time. We shall base OXReqEd (an editor for OXReq documents) on Xylia for two reasons: Firstly, many of the common editor functions can be inherited from Xylia. Secondly, the plug-in mechanism of Xylia can be used to interface OXReqEd with other tools, such as consistency verifiers and oracle generators.

2. OXReq

Open XML Requirements (OXReq) is an XML application that can be used to encode SCR style requirements documents [1] as XML. Mathematical expressions that appear in requirements documents are encoded in OpenMath notation [4]. In order to maintain compatibility with Xylia and be able to import OpenMath elements, the grammar of OXReq is defined in a document type definition (DTD) [3].

Different types of terms can be identified in requirements documents: controlled or monitored quantities, constants, etc. OXReq distinguishes between these different types using a single element type with different values for its "type" attribute. Using this method, all types of terms are represented uniformly in XML, while allowing for a different presentation style for each in OXReqEd, using CSS (cascading stylesheet) [3]. Each term encloses a textual description of that term and a value-set that can be regular text (informal) or a mathematical expression (formal).

Mathematical expressions are encoded in OpenMath notation, which is imported into the OXReq DTD. Each expression is enclosed within an OMOBJ element. In order to tie its functionality with OpenMath, OXReq introduces the "id" attribute for different elements. The idea is to create a cross-referencing mechanism between OpenMath variables (OMV elements) and OXReq elements. The value of the "name" attribute of an OMV element implies equivalency of that element with an OXReq element that has the same value for its "id" attribute. For example, if the name of an OMV is "speed", that OMV is a reference to an element in the document with the id "speed". If there is no such element, that OMV is considered to be a bound variable with no association with the rest of the document. This mechanism dictates using a unique value for the "id" attribute of an element.

As discussed in [1], a requirements document contains four required sections: Environmental Quantities, Environmental Constraints, System Behaviour, and Dictionary. In addition there are three optional sections: System Overview, Notational Conventions and Anticipated Changes. Below is a brief description of each:

Environmental Quantities: This section contains a list of terms. These terms can be of three types: monitored, controlled, or monitored/controlled.

Environmental Constraints: This section encloses a list of constraints, which in turn enclose an optional mathematical expression defining the constraint and a textual description.

System Behaviour: This section gives the required behaviour of the system by specifying the value of all controlled quantities for any possible value and history of the monitored quantities. It contains two sub-sections, as follows:

Mode Classes: This section contains a list of mode classes, each of which in turn contains a set of modes, an initial mode, an optional maximum delay, and the mode function expressed as a math object.

Controlled-variable Functions: For each controlled term in environmental quantities, there exists a controlled variable function in this section. A function is known to refer to a particular

term if the function's "for" attribute matches the ID of that term. Functions are expressed formally as mathematical expressions.

Dictionary: The dictionary consists of a list of terms, types, and function definitions. Only the terms that are not listed in environmental quantities are included in the dictionary. Types are represented as mathematical expressions or a regular textual description of a type. A function definition consists of an ID, domain and range (mathematical expressions), a list of arguments, and the function itself, as a math object.

Notational Conventions: This section is a mix of regular text, tabular text, and math. Any knowledge that is required prior to reading the rest of the document is included in this section.

System Overview: This section is regular text that gives an overview of the system being defined.

Anticipated Changes: This section is regular text that describes aspects of the system requirements that are most likely to change.

Using this grammar and conventions, one can produce complete requirements documents. OpenMath is extensible by nature through the use of content dictionaries. Hence, domain-specific mathematical expressions can be expressed in OpenMath if the right content dictionary is developed.

3. Storage and presentation OXReq

We shall call the definition presented above "storage OXReq". Documents of this type are semantic. The information is nested in a logical manner and a machine interpreter for such an XML document has no difficulty parsing and processing the document. Furthermore, mathematical expressions of storage OXReq are encoded in OpenMath, which is a widely known standard, and many tools exist that are capable of processing it.

Xylia is designed to present information in the same order and nesting that the information appears in the original XML. Also, to increase readability, Xylia hides XML attributes from the main view that is presented to the user. For these reasons, storage OXReq is not suitable for editing with Xylia. These documents make substantial use of XML attributes and the order and nesting of the elements is not necessarily the same as it would be in a format suitable for presentation. Modifying Xylia to accommodate these features of OXReq documents is an expensive task and defeats the purpose of basing OXReqEd on Xylia.

The solution is to create a second XML application for requirements documents that is specifically tailored to be presentable in Xylia. We shall call this XML application "presentation OXReq". This XML application makes little use of attributes, and the elements are nested according to how they should be laid out on the computer screen. Instead of the seven components of storage OXReq, presentation OXReq contains sections and subsections, tables with rows and cells instead of a list of terms and their definitions, and subscript, superscripts, and other presentational mathematics elements, instead of OpenMath objects. For example, figure 1 illustrates what the simple expression $a+\sqrt{b}$ looks like in both presentation and storage OXReq, as well as OXReqEd's visualization for each.

Documents in presentation OXReq format are presentable to the human user via OXReqEd and resemble the way in which they would be printed on paper. Compared to storage OXReq, XML elements in a presentation OXReq document are less deeply nested, which results in an inherently less structured document. Consequently, presentation OXReq is less suitable for machine processing, and interfacing OXReqEd with other tools becomes more difficult. For instance, 3rd party OpenMath-aware tools will not be of any use.

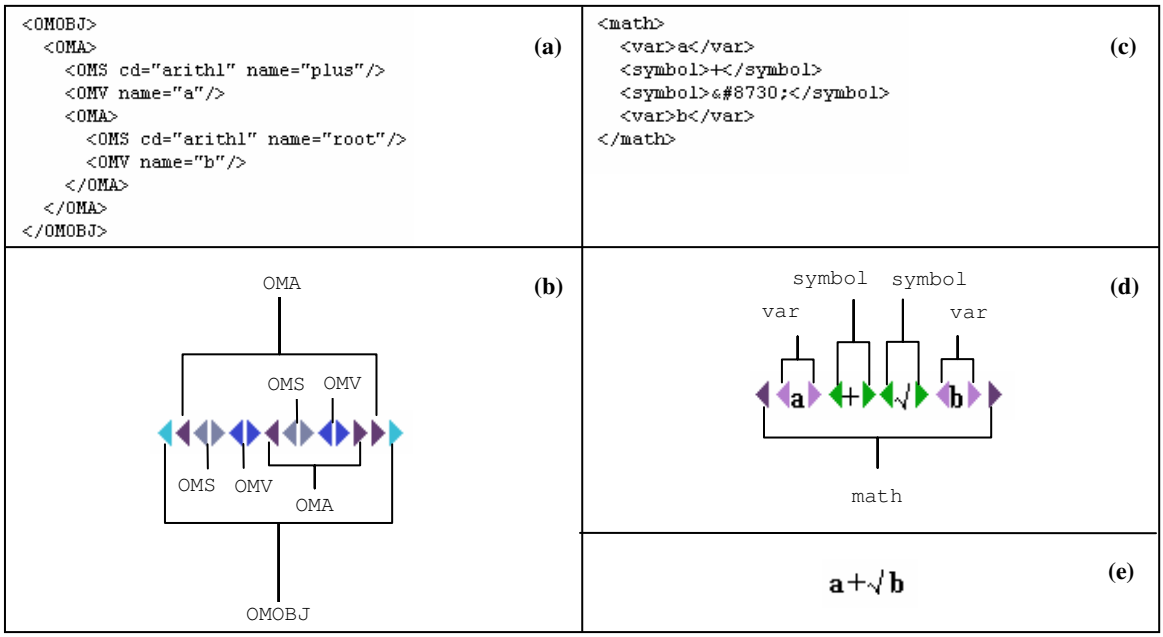

Figure 1: **(a)** The expression $a+\sqrt{b}$ in storage OXReq format. **(b)** OXReqEd's representation of (a). **(c)** The expression $a+\sqrt{b}$ in presentation OXReq format. **(d)** OXReqEd's representation of (c). **(e)** The same as (d), but with hidden sentinels (triangles). Note that (b) and (d) are annotated with sentinel labels for clarity – these labels are not shown in OXReqEd.

4. Conversion between storage and presentation OXReq

Storage and presentation OXReq have their advantages and disadvantages. Storage OXReq is suitable for machine processing and exporting to other tools, but it is not suitable for presentation to the human user. Presentation OXReq, on the other hand, is specifically designed to be presentable to the human user, but is not ideal for machine processing. In order to take advantage of both formats, we need to be able to convert between the two formats at any time.

XSLT (Extensible Stylesheet Language for Transformation) [3] is a language that is used for transforming from one XML application to another. Based on the rules and templates defined in an XSLT, the input XML is traversed systematically and the output XML is generated. Because of its structured format and deep element nesting, storage OXReq is an ideal input for XSLT. We can define templates for all top-level elements of storage OXReq documents, which will in turn use templates of lower level elements (terms, functions, etc.) to traverse the whole document systematically and produce the corresponding presentation OXReq as the output.

A template can be defined for OMOBJ elements as well, delegating the job to templates of children of OMOBJ such as OMV and OMA. Correctly fencing (surrounding with parentheses) the resulting presentational math expression is very important, because failure to do so causes ambiguity. Fencing is implied by nesting of OMA elements in storage OXReq, but must be translated into pairs of parentheses in presentation OXReq. In order to achieve correct fencing using only the features that XSLT provides, we establish two look-up tables, one for operator precedence and another for argument precedence. These tables map operator names to integers that indicate the level of precedence for that operator. Using a reverse-parsing algorithm, pairs of parentheses are added to the output where necessary.

Conversion from presentation to storage OXReq is not as easily achieved as the conversion explained above. Presentation OXReq documents are less structured than storage OXReq documents, since they are designed to be presentable, much like a plain, linear text document. As a result, they are not ideal inputs for

an XSLT which can transform them into storage OXReq. This shortcoming is especially apparent with presentational math. XSLT alone is not readily capable of transforming presentational mathematics into OpenMath notation.

A JavaCC [5] parser is employed to parse the math portions of presentation OXReq and produce the corresponding OpenMath representation. The grammar and lexical specifications for these portions are defined and fed into JavaCC to produce a recursive descent math parser. This parser operates in a top-down fashion, which is ideal for presentational math. The precedence climbing algorithm described in [6] is used to take into account the precedence of operators, in order to produce the correct OpenMath output. Through XSLT extensions, this Java-based parser can be embedded in the XSLT that transforms non-mathematical portions of presentation OXReq into storage OXReq. The resulting XSLT/Math Parser package can then be applied to presentation OXReq in order to obtain the corresponding storage OXReq. Figure 2 illustrates the transformation process.

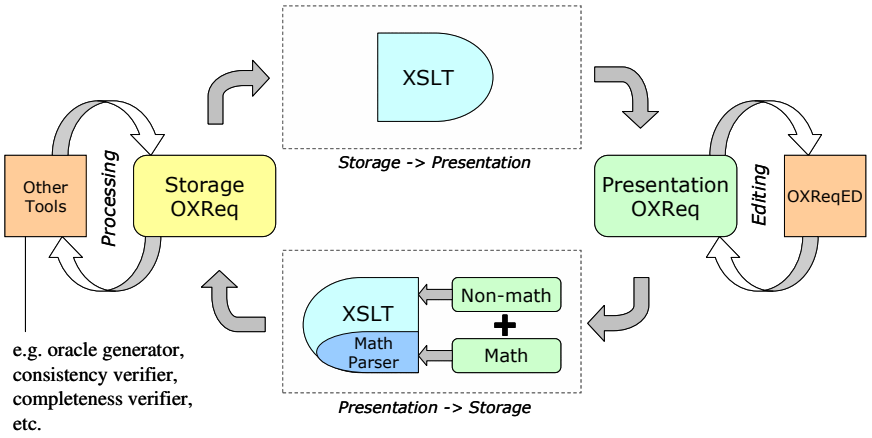

Figure 2: The transformation process from storage to presentation OXReq and vice versa

Input and output filters can be developed for OXReqEd, using the plug-in mechanism of Xylia, in order to facilitate the transformation from one format to another, at the user's will. Authoring and editing of OXReq documents will be done in presentation OXReq using OXReqEd. The results can be stored in storage OXReq and used for machine processing. Plug-ins can also be developed to enable on-the-fly conversion between the two formats for run-time interfacing of OXReqEd with other tools.

5. References

[1] D. K. Peters, "Deriving Real-Time Monitors from System Requirements Documentation," Ph.D. Thesis, McMaster University, 2000.

[2] P. Shaker, T. S. Norvell, and D. K. Peters, "Edits in Xylia: Preserving the Validity of XML Documents," Memorial University of Newfoundland, 2002.

[3] "World Wide Web Consortium," [Online Website], Available at URL: `http://www.w3c.org/`

[4] "OpenMath," [Online Website], Available at URL: `http://www.openmath.org/`

[5] "JavaCC," [Online Website], Available at URL: `https://javacc.dev.java.net/`

[6] T. S. Norvell, "Parsing Expressions by Recursive Descent," [Online Document] 1999-2001, Available at URL: `http://www.engr.mun.ca/~theo/Misc/exp_parsing.htm`