An Integrated Approach to Software Engineering using XML

Dilip Antony Joseph
Indian Institute of Technology, Madras
dilip@peacock.iitm.ernet.in

Abstract

Software Engineering plays a very important role in the successful completion of large software projects. It revolves around the following phases [1] - Requirements Engineering, Software Architecture and Design, Implementation, Testing, Maintenance. There are a large number of approaches to manage each of the above mentioned phases. In this paper, an integrated approach to manage all the phases through the use of eXtensible Markup Language (XML) [2] is proposed. The development of an XML schema for software engineering is described. The usefulness of XML embedded source code is also explored here.

1 Introduction

The size of software projects has increased exponentially over the last few decades. Typical projects often run into millions of lines of code. The complex requirement specifications and design of the software to be developed have become increasingly difficult to manage. The mapping between code, design and requirements is often lost during the development process. This causes a big problem during later project maintenance. In this paper, an integrated approach to software engineering using eXtensible Markup Language (XML) is proposed. An XML Schema to support this process is developed in the following sections. XML embedded documents exemplify the approach taken in this paper.

2 A Brief Description of XML

XML or eXtensible Markup Language is a W3C-endorsed standard for document markup. It allows one to define tags which are used to mark up data in a simple, human-readable form. The marked-up data is completely text and can thus be processed by any application. XML has already found application in diverse fields. The use of XML in software engineering will be illustrated in this paper. A quick description of XML can be found in [4].

XML Schemas or vocabularies are defined for specific applications. These schemas define the tags used to markup the data. Various transformations can be applied to the marked-up data and it can be converted to a wide variety of formats for display. For example, the data can be converted into an application-specific display on web browsers using Cascading Style Sheets (CSS). Each XML document has a tree structure, each element of which can be referenced using the XPath language. XLinks are used for attaching unidimensional or multidimensional links to XML documents. These links can be bidirectional and can even refer to non-XML documents. The use of all these features in software engineering will be clearly described in the following sections.

3 The Software Engineering Problem

The software engineering problem can be attributed to the following simple reasons:

1. The requirements of today's software are highly complex and extremely large in size.
2. Complex requirements lead to complex and intricate software architectures.
3. A very large number of lines of code are produced. In time and cost constrained environments, the code is improperly documented and often the mapping between code, design and requirements - i.e. what code implements what design to satisfy what requirement - is lost.
4. Software maintenance becomes very time consuming and costly as a result of unwieldy code.

The field of software engineering is rich with tools aimed at alleviating these problems. These tools often use proprietary solutions and data storage formats. One system cannot interact with the other management systems. In many cases, portability between various systems is desired. This paper proposes an integrated portable solution to the software engineering problem.

4 XML in the Requirements Specification Phase

Requirements gathering and specification is the first phase in any software project. A formal requirements document is produced and this forms the basis for the agreement between the software developer and client. All requirements are explicitly documented in the Software Requirements Specification (SRS) document.

Instead of preparing the document in conventional document processing formats, the SRS should be prepared in XML marked-up text format. The XML marked-up SRS can be easily converted into a variety of formats

```
<srs author="ABC DEF" date="Feb 1, 2003"
modified="Feb 15, 2003">
<client id="C007" name="James Bond Inc"/>
<project>
<title>Complete Secret Agent Suite</title>
<keywords>secrets, crypto, weapons
</keywords>
<brief>a brief description</brief>
<long>a detailed description</long>
<aim>What is the aim of the project</aim>
<deadline type="hard">
March 1,2003</deadline>
<requirement rId="r1">
<rtitle>Weapons Management</rtitle>
<rdesc>Manage a list of weapons
currently in hand</rdesc>
<spec sId="s1">Specification 1</spec>
<spec sId="s2">Specification 2</spec>
<constraint cId="c1">
This is a constraint
</constraint>
</requirement>
    .
    .
    .
</project>
</srs>
```

Figure 1: A very simple XML encoded SRS

for publishing in the print media or in electronic form, using XML Style Sheets (XSLTs) or Cascading Style Sheets (CSS).

A very simple XML encoded SRS is shown in Figure 1. This SRS is given only for illustration. Any real XML SRS document will contain many more tags encompassing multiple levels of requirements, constraints and other relevant requirements details. A detailed schema is to be developed.

5 XML in the Design Phase

The next phase after Requirements specification is the Design. The architecture of the software is first developed. Details about the various elements of the architecture are then filled in. The efficiency and efficacy of the project depends on the strength of the design to a large extent. The design document produced is used by the persons involved in coding, documentation and maintenance. As seen in Section 4, it is proposed that the design document be XML encoded too. An appropriate XML schema can lead to a clear, unambiguous and modular design document. The XML marked-up design document will clearly relate to the XML SRS through the XLinks and XPointer features of XML.

Data Flow Design, Object-Oriented Design and Functional decomposition are some popular approaches to software design. The XML schema for describing the design details can be based on any of the above approaches to design. A hybrid schema which combines the good features of all approaches can also be attempted. An XML schema for the design phase has not been completed. However, Figure 2 gives a general picture of the same.

6 XML in the Implementation Phase

After good XML encoded Software Requirements Specification and Design Document have been prepared, implementation is the

```
<designdoc author="ABC DEF" date="Feb 5, 2003"
modified="Feb 10, 2003" xlink:type="simple"
xlink:title="Requirements Spefications"
xlink:href="/projects/secretsuite/srs.xml">
<architecture>
<module id="m1" name="moduleone">
<description> ... </description>
<input type="integer" id="i1">InOne</input>
<inputtype="weaponObj" id="i2">InTwo</input>
<output type="bool">WeaponAvl</output>
<working> ... </working>
<connected_to type="input" toId="m2" />
<connected_to type="init" toId="m4" />
 .
 .
</module>
</architecture>
</designdoc>
```

Figure 2: A sample XML encoded Design Document

next phase to be tackled. The clear and concise specifications greatly enhance the programmers' understanding of the problem and aid in higher throughput. XML can also play an explicit role in the actual coding process.

Comments are inserted in the source code to improve the readability and maintainability of the code. Tools like javadoc [3] enable automatic generation of formatted Application Programming Interface (API) documents from comments written in a specific format. There exists specialized tools for other languages too. What is proposed here is a generic format for embedding comments in source code. This format will be based on XML. An advantage of using XML is that documentation (for example, APIs) can be produced for any language by the same tool. The format of the documentation produced can be suitably adjusted using XML stylesheets. Any complex unintuitive operations performed by a particular code fragment can be described accurately in a particular format. The inputs and outputs of the various modules can be explicitly stated. Various invariants or conditions to be satisfied at particular points of code may also be documented explicitly. This can be of aid during debugging, testing and maintenance.

However, the most important advantage of this method is that it allows us to be more specific about the code written. Code fragments can be mapped to the specific requirement or design detail which led to that piece of code, using XLinks and XPointers. This will prove to be a great help to the software maintenance personnel. A sample XML commented code is shown in Figure 3.

```
<code_module id="c1121">
<input name="a" type="int"/>
<input name="b" type="int"/>
<output type="int">The gcd</output>
<algo name="Euclid" xlink:type="simple"
 xlink:title="Introduction to Algorithms"
 xlink:href="urn:isbn:81-203-1353-4">
<desc> a brief description </desc>
int gcd ( int a, int b) {
   if (b = 0)
     return a;
   else
     <tricky_code type="recursive">
     <desc> explain what happens</desc>
     return (gcd (b, a % b) );
     </tricky_code>

}
</code_module>
```

Figure 3: Illustration of source code with XML embedded comments

The question arises on how the XML tags are written in the source code files. The solution as mentioned earlier is to embed them as comments in the format described by the specific implementation language. In such an approach, the source code will have to be preprocessed before it can be consumed by an XML parser. A better approach is to directly write the XML tags into the source code file. The file can be made into a format understandable by the compiler through the application of a suitable XSLT transformation to the source file before compilation.

7 XML in the Testing Phase

The testing process is aided by the XML documents and source files produced in the earlier phases. Test strategies may be devised based on the concise and clear specifications of the program. Tricky sections of code marked in the source files may be given special attention. The invariants (for example, loop invariants) may be verified at test time. The actual test plan, detailing the inputs and outputs for the test and the methods to be followed can be written as a XML document. Various sections of this document can be linked to related sections in the SRS, Design document and source code using XLinks and XPointers. XML tagged data provide a concise and accurate method to record the results of the tests.

8 XML in the Maintenance Phase

Maintenance of software often consumes the largest amount of time and money. It involves satisfying new client requirements and features, in addition to correcting bugs that may have escaped the testing phase. Maintenance also caters to porting the system to a new environment. All these activities are aided by the availability of detailed and precise documents from the previous phases. These documents can be studied and the necessary changes are made to the software. These changes are also documented - in XML of course!

9 The Integrated Solution

The previous sections highlighted the application of XML in each phase of the Software Engineering process. All these processes are integrated into one common package. This package will consist of specialized XML schema definitions for each phase. Many of the schema elements will be common across the different phases.

A suite of tools are required for managing the XML documents produced. The generality of XML allows for easy portability across different vendors of such software tools. All the XML documents mentioned in this paper can also be generated by hand using a simple text editor. But it is certainly not an easy job. The tools belonging to the package will process the XML data and will present a complete Graphical User Interface (GUI) to manage all the aspects of the software development project. For example, a programming tool helps the programmer to insert the appropriate comments at specific locations in the code. All XML tags may be hidden from the programmer's display if he so desires.

A framework for the integrated XML based software management package is specified in Figure 4.

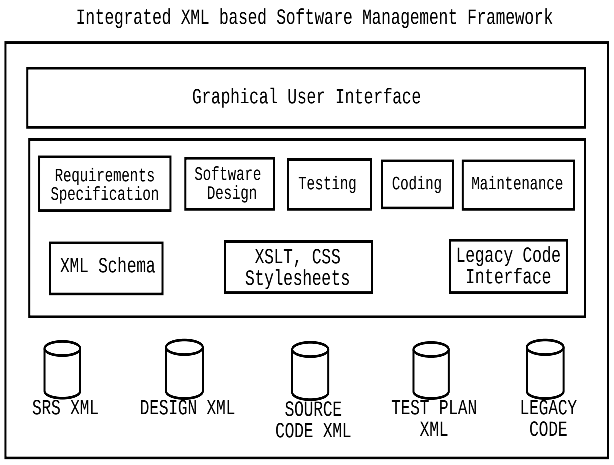

Figure 4: An Integrated XML based framework for software management

10 Conclusion and Further Work

An integrated approach to software development and management using XML was presented in this paper. The application of XML in each phase of software engineering was observed. A generic framework encompassing all the phases was proposed. Detailed and complete XML schemas for the different phases must be built. Appropriate XSLT and CSS styles sheets must be constructed. The generic architecture given here must be refined for each phase. A suitable method to incorporate legacy code without XML markup, into the new XML based system must be designed. Future work will concentrate on the above mentioned goals.

References

[1] Hans Van Vliet. Software engineering - principles and practice. Wiley. 2002

[2] World Wide Web Consortium - XML. http://www.w3.org/XML

[3] Java Home Page. http://java.sun.com

[4] Elliotte Rusty Harold, W. Scott Means, XML in a nutshell. O'Reilly, 1998