

Using XML to support Information Integration

Kenneth M. Anderson
Susanne A. Sherba
University of Colorado, Boulder
{kena, sherba}@cs.colorado.edu

Abstract

Software engineers face overwhelming information management tasks, such as performing requirements traceability between the artifacts of a software development project, or ensuring that these artifacts are consistent with one another over time. These tasks are hindered because software artifacts are created with a multitude of tools and are stored in a variety of data formats, and none (or few) of these tools were designed to interoperate or make it easy to share information. We describe an approach to information integration based on open hypermedia to address these problems and deliver powerful information management tools into the hands of software engineers. Although the work is preliminary, the existing prototypes make significant use of XML which enables a degree of flexibility and extensibility not normally found in prototype software tools. We reflect on these features and discuss our future plans for use of XML within the information integration environment.

1. Introduction

Large-scale software development projects face a significant number of hurdles. Such projects are tasked with constructing a complex software system under stringent time and budget constraints. In addition, software engineers on these projects encounter both accidental and essential difficulties that are associated with the nature of software and its construction, as outlined by Brooks [7]. For instance, software teams must be trained to use (often complex) software life cycles and design techniques, ensure that communication roles and paths are formalized among team members, and that the appropriate artifacts are constructed and maintained throughout the lifetime of the project [8]. These tasks are difficult to achieve and, as a result, most

projects run over budget and deliver faulty systems to customers long after they were originally due [12]. A key contributor to the problem is the information management tasks that software engineers must perform, that range from finding information within the set of artifacts to performing requirements traceability between software artifacts. In particular, it is difficult to share and integrate information between artifacts due to the bewildering number of tools and data formats used to create and maintain them.

We are developing an information integration environment to aid software developers in performing complex information management tasks. In particular, we focus on supporting those tasks which involve creating, finding, maintaining, and evolving the relationships (both implicit and explicit) between software artifacts. Our approach is based on open hypermedia [14] and makes heavy use of the eXtensible Markup Language [6], also known as XML¹.

In this position paper, we present a brief introduction to our information integration environment and then focus on how our initial experimental prototypes make use of XML and how this use provides a level of flexibility and extensibility not normally associated with prototype software tools. After this analysis, we conclude by describing our future plans for using XML in our information integration environment and what benefits we hope to achieve as a result.

2. Information Integration Environment

A model of the information integration environment appears in Figure 1. The model consists of users,

1. We do not present a review of open hypermedia in this paper due to space limitations. See Østerbye and Wiil [14] or Anderson, et al. [5], for additional information.

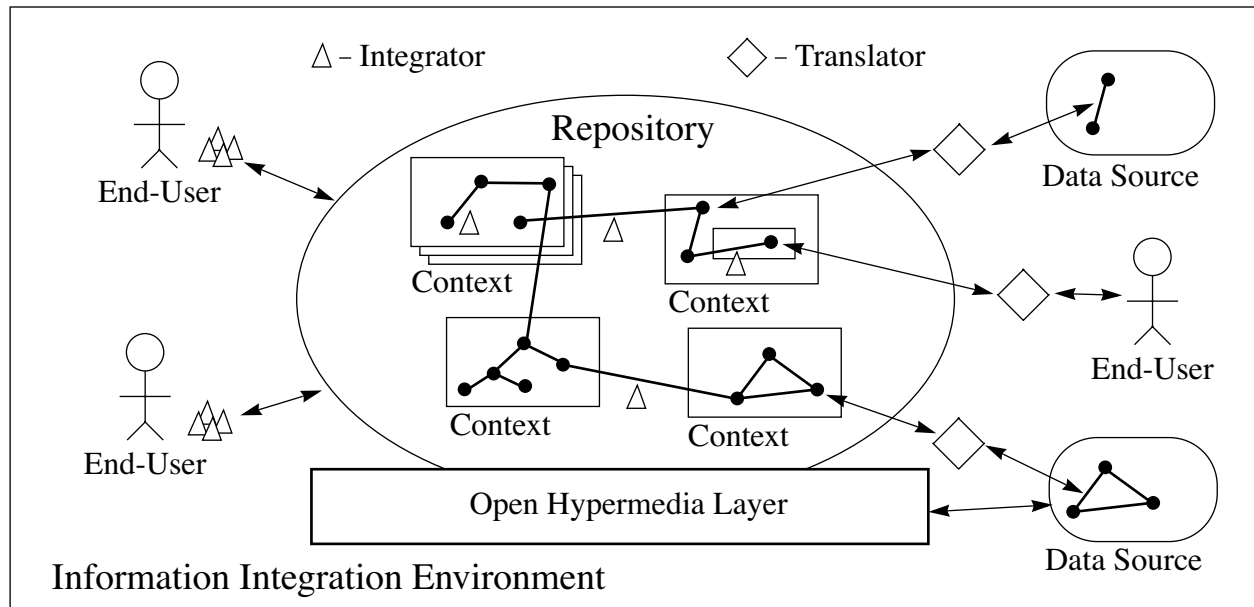


Figure 1. An initial model of the information integration environment.

data sources, integrators, translators, contexts, a repository, and an open hypermedia layer. The basic idea is that information is brought into the information integration environment by invoking a *translator*. The translator retrieves information from a *data source* and stores it in a *repository*. The repository consists of multiple *contexts*; contexts store information (in XML documents) and meta-information in the form of attribute-value pairs. Contexts can contain sub-contexts. In addition, the documents of a context can be linked together in arbitrary relationships; relationships can span multiple contexts and can have type information associated with them (e.g. a requirements traceability link, a consistency relationship, etc.). Relationships can be stored as XLinks [13] or within the open hypermedia layer. (The Chimera open hypermedia system [5] will provide open hypermedia services to our experimental prototypes.) *Integrators* aid users in finding, creating, maintaining, and evolving relationships within the repository. While our main focus is on relationship management, integrators are free to create new contexts and to store information within them while performing their tasks. For instance, an integrator that searches documents for particular keywords, may store the location of discovered keywords in a document separate from the documents being searched. It may then create a new context and include pointers to the searched documents plus the document that it created.

Finally, the relationships that are generated by integrators can be imported into the open hypermedia layer and then made available to the original data source using its native editing tools. This feature allows

software engineers to submit a set of artifacts to the repository, make use of integrators to discover relationships contained in those artifacts, and then view the generated relationships within the native editing environment of the original artifacts (provided the native editor is integrated with the open hypermedia system [10, 15]). This capability is critical to enabling the adoption of the information integration environment; while it is true that our approach requires the use of a new environment to perform information integration tasks, the results of those operations can be made available to the software engineers within their own tools. Software engineers, thus, gain access to tools and techniques to perform information management tasks, but they are not required to give up the tools they already use to gain these capabilities.

2.1. Related Work

We now briefly review two related systems. The GeoWorlds environment [16] is an information analysis environment that allows users to retrieve documents off the Web, organize them into collections, and apply a variety of analyses upon them. GeoWorlds is strictly focused on the World Wide Web and can only import information from Web-based data sources. We intend to support both remote and local information sources, with particular attention to supporting legacy, third-party, data formats. This will allow our environment to be applied to both existing and new software development projects. In addition, GeoWorlds services are focused more on information analysis while our focus

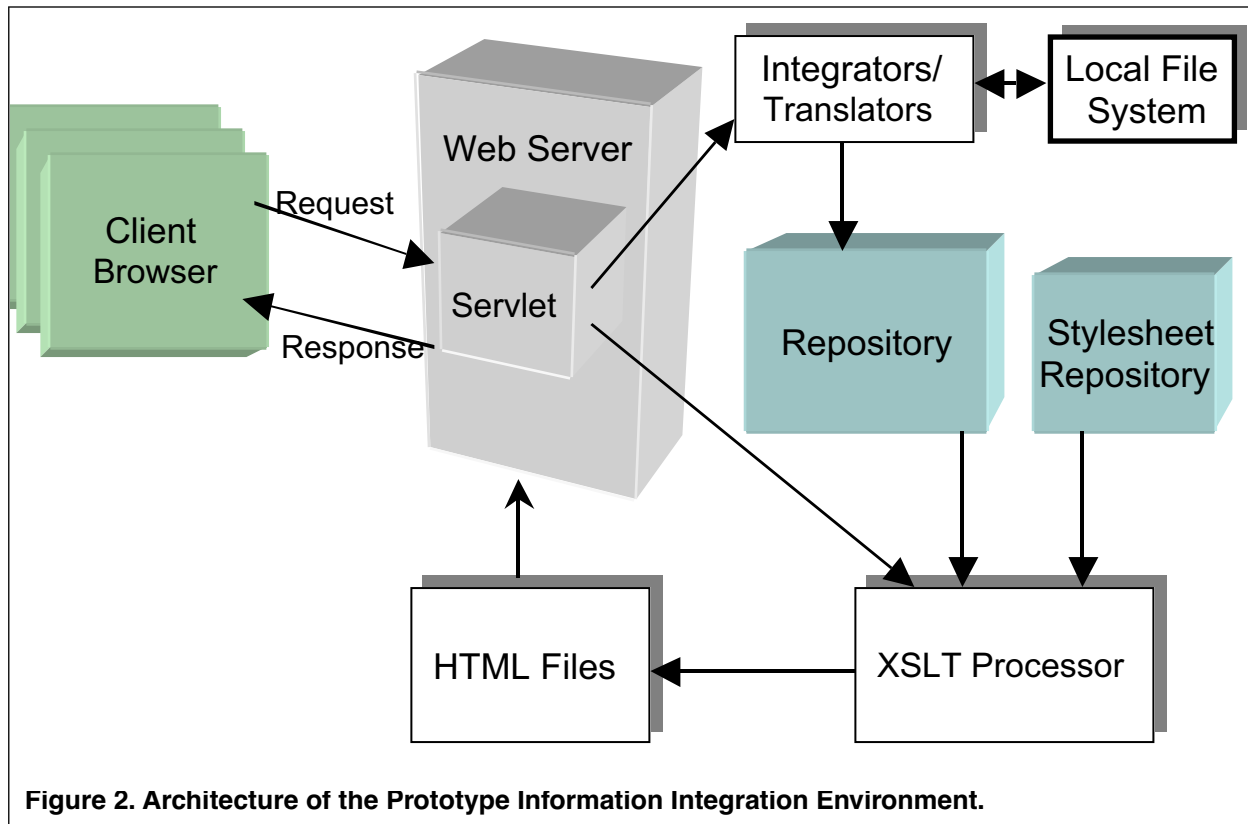


Figure 2. Architecture of the Prototype Information Integration Environment.

will be on relationship management. Our environment will thus have greater capabilities for discovering, viewing, and manipulating relationships than what is found in the GeoWorlds environment.

The second related system is located at <http://www.xlinkit.com/>. It is a link generation engine that allows consistency relationships to be specified over software artifacts. The basic idea is that a software engineer writes consistency rules for a set of documents and then submits those rules along with a set of documents. (Documents must be converted to XML before the link generation engine can process them.) The link generation engine then checks the documents to see if they follow the submitted consistency rules. As output, the engine generates a report that displays the results of the analysis: instances of the rules are highlighted and information is provided to show, for each instance, if the rule was followed or violated. Our environment can be used to check consistency relationships over software artifacts, but it is also intended to support a broader spectrum of relationship types. For instance, we intend to build integrators that can aid the process of generating requirements traceability links, similar to the results we achieved with Northrop Grumman using only the Chimera open hypermedia system [1-3]. Rather than providing a rule-based language for a single relationship type, our environment will provide

APIs to software engineers that will allow them to construct their own translators and integrators to manage the relationships relevant to their software development projects. This does not mean that rule-based languages are not helpful in automatic link generation; indeed the experience with xlinkit demonstrates the benefits of this technique. In fact, we plan to leverage the results of the xlinkit experience, along with other work in hypermedia link generation, to create a generic rule-based integrator that can support various rule sets via a plug-in mechanism. In addition, our use of open hypermedia will allow the relationships discovered in the environment to be viewable within the native editing context of the original software artifacts. Thus, while both of these systems require a translation step into XML, our approach will allow information to flow back to the original artifacts.

3. Use of XML in the Initial Prototypes

Our basic rule for the use of XML in supporting the engineering of software systems is “If a feature requires extensibility, specify its configuration information in XML.” This rule was applied successfully to our open hypermedia system [4] in which XML enables extensibility and flexibility with respect to the database used by Chimera, the information stored in

Chimera's data model, Chimera's link traversal operations, etc.

Our work on the information integration project is still in its preliminary phases, however the use of XML has enabled the development of a set of flexible and extensible prototypes upon which to base our research efforts. We have constructed a Web-based system for interacting with the repository, invoking translators and integrators, and viewing the results of integrator operations. (The integration of this prototype environment with our open hypermedia system has not yet occurred, but will involve the translation of XLinks into open hypermedia links. We have already explored issues related to the other direction, e.g. translating open hypermedia links into XLinks [11].) With the exception of the code for our prototype translators and integrators, everything contained in the environment is specified in XML. This includes the documents and contexts stored in the repository, the configuration files for specifying the availability of integrators and translators, and the configuration files for specifying the operations provided by the information environment (which includes functionality such as "Create New Context"). Our Web-based system then uses XSLT [9] to translate the XML information stored in the repository into HTML that can be viewed by a Web browser. The architecture of our Web-based system is shown in Figure 2.

A Java servlet provides access to documents in the repository. We associate different XSLT stylesheets with different types of repository documents. When a request comes in for a particular type of document, the servlet hands the requested document and its associated stylesheet to the XSLT processor (currently Apache.org's Xalan) which produces an HTML document that the servlet sends back to the client's Web browser.

4. Conclusions

We have found XML to be a useful tool for creating extensible and flexible software prototypes. Currently, we are using XML to support software engineering research into the issue of information integration and have employed a powerful Web-based architecture that directly supports experimentation with alternative designs (such as supporting multiple types of repositories, and multiple user interfaces to the environment as a whole) and can support the addition of new integrators, translators, and operations into the core environment by modifying XML configuration files. Having reached the point where we are building prototype integrators and translators, we are beginning to encounter

issues relating to the XML formats that will be supported by the repository. In particular, should we attempt to create a "canonical" format for the repository or should we attempt to support multiple types of XML documents. Put another way, do we require that all XML documents in the repository conform to a single DTD or XSchema, or do we support multiple DTDs and XSchemas? In addition, we are ready to begin experiments moving relationships stored in XLink format into the open hypermedia layer to provide access to these relationships to external clients.

References

1. Anderson, K. M. (1999). Data Scalability in Open Hypermedia Systems. In *Proceedings of Hypertext'99*, pp. 27-36.
2. Anderson, K. M. (1999). Issues of Data Scalability in Open Hypermedia Systems. *The New Review of Hypermedia and Multimedia*, 5: 151-178.
3. Anderson, K. M. (1999). Supporting Industrial Hyperwebs: Lessons in Scalability. In *Proceedings of ICSE'99*, pp. 573-582.
4. Anderson, K. M. (2001). The Extensibility Mechanisms of the Chimera Open Hypermedia System. *Journal of Network and Computer Applications*, 24(1): 1-12.
5. Anderson, K. M., Taylor, R. N., and Whitehead, E. J., Jr. (2000). Chimera: Hypermedia for Heterogeneous Software Development Environments. *ACM TOIS*, 18(3): 211-245.
6. Bray, T., Paoli, J., and Sperberg-McQueen, C. M. Extensible Markup Language (XML) 1.0, W3C Recommendation, 10-Feb-1998. <<http://www.w3.org/TR/REC-xml>>.
7. Brooks, F. (1987). No Silver Bullet: Essence and Accidents of Software Engineering. *IEEE Computer*, 20(4): 10-19.
8. Brooks, F. P., Jr. (1995). *The Mythical Man-Month, 20th Anniversary Edition*. Addison-Wesley. 322 pages.
9. Clark, J. XSL Transformations (XSLT) Version 1.0 W3C Recommendation, 16-Nov-1999. <<http://www.w3.org/TR/xslt.html>>.
10. Davis, H. C., Knight, S., and Hall, W. (1994). Light Hypermedia Link Services: A Study of Third Party Application Integration. In *Proc. of Hypertext'97*, pp. 41-50.
11. Halsey, B., and Anderson, K. M. (2000). XLink and Open Hypermedia Systems: A Preliminary Investigation. In *Proceedings of Hypertext 2000*, pp. 212-213.
12. Johnson, J. (1995). Creating Chaos. *American Programmer*, July 1995.
13. Maler, E., and DeRose, S. (1998). XML Linking Language (XLink). <<http://www.w3.org/TR/1998/WD-xlink-19980303>>.
14. Østerbye, K., and Wiil, U. K. (1996). The Flag Taxonomy of Open Hypermedia Systems. In *Proceedings of the Seventh ACM Conference on Hypertext*, pp. 129-139. Washington DC, USA. March 16-20, 1996.
15. Whitehead, E. J., Jr. (1997). An Architectural Model for Application Integration in Open Hypermedia Environments. In *Proceedings of Hypertext'97*, pp. 1-12.
16. Yao, K., Ko, I., Eleish, R., and Neches, R. (2000). Asynchronous Information Space Analysis Architecture Using Content and Structure-Based Service Brokering. In *Proceedings of the 2000 ACM Conference on Digital Libraries*.