Managing the evolution of Web-based applications with WebSCM

Tien N. Nguyen, Ethan V. Munson, and Cheng Thao
Computer Science Department
University of Wisconsin-Milwaukee
{tien,munson,chengt}@cs.uwm.edu

Abstract

In traditional software engineering tools, software configuration management (SCM) is the dominant approach to manage the evolution of a software system. However, the evolution of Web-based applications presents special challenges that have not been well addressed by existing SCM systems. They often treat hyperlinked Web documents as a set of text files in a file system and disregard crucial structures in a Web-based application such as internal structure, navigational structure, logical structure, and compositional structure among Web objects. Key limitations include their inadequacy to represent Web object semantics and their inability to manage changes to those important structures among objects making up a Web application. This paper presents a novel SCM-centered development environment for Web-based applications, named WebSCM. With WebSCM, Web developers can manage fine-grained, structural evolution of Web objects and important structures among them at different levels of abstraction (e.g. conceptual, navigational, presentation, and implementation levels). This paper also describes the motivation for this environment as well as its user interfaces, features, and implementation.

1 Introduction

Millions of Web sites have been developed in the past ten years. A lot of efforts and time have been spent to maintain Web-based applications in the daily basis. Small Web sites with less than dozen HTML pages can easily be developed and maintained. However, many organizations have failed or have struggled to avoid major failures in developing large-scaled and high-quality Web-based applications. The main reason is that many Web developers commonly use ad hoc development processes that lack rigor, systematic techniques, sound methodologies, and quality assurance and may pay little attention to issues such as requirements analysis, quality, performance evaluation, configuration management, maintainability, and scalability [19].

To address this, many techniques from traditional and modern software engineering have been applied to Web development and maintenance. Especially, software configuration management (SCM) technology has been used to manage the evolution of Web-based systems. However, the maintenance of Web-based applications do have distinct characteristics that arise from the nature of hyperlinks and the Web environment [19, 26]. The evolution of Web applications presents special challenges that have not been well addressed by existing SCM systems.

First of all, Web systems are built from a wide diversity of objects including documents in markup languages, document templates, style sheets, images, streaming media, animations, applets, scripts, etc. Unlike software engineering, where program source code is seen as the central artifact, in Web systems it is difficult to identify one type of object as most important. So, an SCM system for Web applications must be compatible with a wide variety of file types with different kinds of structure. Second, a Web document, either static or dynamic, has some internal structure. An HTML document has a tree-based syntactical structure, while a program can be regarded as an abstract syntax tree (AST) of syntactic units. Taking advantage of the internal structure of a Web page results in various benefits in processing, visualizing, searching and retrieving Web information [14, 15, 29]. As the Internet moves toward XML [21], it is likely that Web documents will gain structure with semantics of growing importance. Structural semantics is described, at least to some extent, by Document Type Definitions (DTD) or XML Schemas, which are also evolving objects. So, Web content must evolve along with its structural rules. Traditional SCM tools are not well-suited for Web applications because they often use a line-oriented model of internal changes that disregards the internal structures of Web contents.

In addition, a good navigational structure of a Web system is one of keys to success of a Web application. Navigational structure refers to the structure of Web documents with respect to hyperlinks among them. Web documents in a Web system are logically related and connected to each
other via these hyperlinks. When a user understands where he can navigate, and how he can reach a desired target point, he can benefit the most from the Web application. Navigational structure also evolves over time. Recording its evolution would facilitate the management of “dangling” links and avoid the navigation to an incorrect version of a destination document.

Configuration management for Web-based applications is much more than providing version control and change management for individual Web documents. Logical structures and relationships in a Web system are also very important. As developing large-scale Web applications, developers may apply a design methodology, which requires certain logical design models, separately from the implementation model (e.g., HTML files or scripts). For example, in OOHDM [26], the conceptual model expresses relations among data objects, while the navigation model and presentation model address navigational structure and information presentation in a Web system, respectively. Entities in each of these design models are often related to each other. The relationships among them are crucial for the success of a design process [26]. For example, between entities in the data conceptual model and those in the presentation model, there exists logical mappings such as in which pages or screens a piece of data or an object will be presented. Between presentation and navigation models, the information such as how screens or pages are connected by hyperlinks is very important. In addition, the mapping between screen design and implementation files is important, for example, in which HTML files a screen in the presentation model is realized. All of those logical mappings, structures, and relations evolve over time. Recording the evolution of those logical connections would greatly benefit for developers in their development and maintenance tasks.

This paper presents a novel SCM-centered development environment for Web-based applications, named WebSCM. WebSCM was built using Molhado [22], an object-oriented SCM infrastructure, that is able to manage versions of a software system at the logical level. With WebSCM, Web developers can manage fine-grained, structural evolution of Web objects and important structures among them at different levels of abstraction (e.g., conceptual, navigational, presentation, implementation models). Important logical connections between those models are recorded over time. WebSCM always maintains version consistency not only among implementation artifacts (e.g., source code, HTML, etc), but also among logical entities in design models. Web developers can easily relate together changes in their designs and implementation artifacts.

The next section will discuss related work on applying version control and configuration management to Web-based applications. Section 3 briefly describes the basic data and version model that is used in WebSCM. Section 4 presents our representation for designs at the conceptual data model in a Web-based application. Section 5 describes WebSCM’s supports for screen designs for Web pages. The representation for Web documents is presented in Section 6. Section 7 describes how the system manages versions and configurations of Web contents, important structures, and logical mappings. The highlights of GUI-based functionality are discussed in Section 8. Conclusions appear in the last section.

2 Related work

Researchers and vendors in the configuration management area are taking different approaches to SCM for the Web. All have added Web functions to their SCM tools by offering access to some or all SCM functionality through a browser [7]. WebSynergy [35] provides a Web front-end into all of its existing SCM capabilities as well as Web authoring tools. Similar to our approach, MKS’s WebIntegrity [34] integrates its version control facilities with an authoring tool, while in Merant’s PVCS [20], version control is the core part and is separate from authoring systems. However, both of them version control at the file level. StarTeam [30] is Web-enabled with the intention of tool integration. TrueChange [33] provides content change management along with its version control, but with less focus on configuration management. Serena’s eChangeMan [13] is focused on process management and change tracking. Rational’s ClearCase [18] provides configuration management for text files via total versioning approach.

ClearQuest [5] is a change request management tool of ClearCase, coordinating many developers in changing Web documents. Content change management in SourceSafe [28] is line-oriented. Computer Associate’s CCC/Harvest [4] pays considerable attention to supporting collaboration among distributed development teams. Perforce [25] is more lightweight than other SCM tools and it has the ability to migrate repositories from other SCM tools such as CVS and PVCS into its internal repository. Although all of these SCM tools have distinguished and valuable features, they are focused on version control of files, rather than on configuration management for objects making up a Web system. None of them supports fine-grained change management at the logical level and all content change management is line-oriented.

On the other hand, some Web application development environments (also called Web authoring environments or Web content management environments) have realized that SCM practices need to be incorporated into their tools. Tools such as FrontPage, Macromedia’s DreamWeaver [10], and ColdFusion [6] have no built-in version control support. Other tools, such as StoryServer [31] and TeamSite [32], are designed to support
many aspects of Web development, with particular strength in supporting collaborative work. TeamSite provides visual differencing tools so that two versions of the same content can be examined side by side. Inso’s DynaBase [17] is an integrated content management and publishing platform for Web applications. Similar to our approach, it is XML-based, allowing better management and reuse of data. In DynaBase, configuration management use the “tagging” technique also seen in CVS. Configurations act like a bill-of-materials for the Web site, enumerating which items are included in the site and which version of each item is in use.

ArticleBase [1] integrates content management and version control into an authoring and publishing system. Its version control support is on file basis.

Many research in versioned hypermedia community [16, 24, 37] have focused on version control for documents in the presence of hyperlinks. However, the main goals of versioned hypermedia systems often do not include supports for Web application development. Therefore, supports for program source code are very limited.

To improve the authoring and browsing features for versioned contents of Web pages, some researchers in this area followed the language-oriented approach. They have attempted to change the Uniform Resource Locator (URL) of a Web page to include a version identifier [27]. They use existing Web infrastructure such as forms, Java applets, and plug-ins to create a user interface for version control systems on the server. Bendix and Vitali proposed VTML (versioned text markup language) [3] to express change operations for HTML documents. For example, they introduced two new tags (INS and DEL) to express insertion and deletion. The WebDAV protocol [36] is an extension of the Hypertext Transfer Protocol (HTTP) to support distributed authoring and versioning. It extends HTTP to include versioning operations for Web pages. Apache is a transaction-time HTTP server that supports document versioning [12]. To construct a document version history, snapshots of the documents files are obtained over time. Transaction times are associated with each file version. The transaction time is the system time of the edit that created the version.

In general, existing SCM systems for Web applications have a large variety of useful functionality. The similarities among them include supports for Web and scripting languages, templates and stylesheets, versioning of files, rollback of complete sites via backup, audit logging, workflow support for collaborative work, commercial database interfaces, change tracking and management support. However, configuration management support is still limited to files. None of them cares about the navigational or internal structure of Web documents. Their content change management is coarse-grained, with differencing done on a line-by-line basis. None of them supports for managing logical connections among entities of different design models over time.

3 Data model

WebSCM is built based on an object-oriented SCM infrastructure, named Molhado [22]. This section describes the main concepts of the data model used in Molhado. Those concepts are node, slot, attribute, and sequence. A node is the basic unit of identity and is used to represent any abstraction. A slot is a location that can store a value in any data type, possibly a reference to a node. A slot can exist in isolation but typically slots are attached to nodes, using an attribute. An attribute is a mapping from nodes to slots. An attribute may have particular slots for some nodes and map all other nodes to a default slot. The data model can thus be regarded as an attribute table whose rows correspond to nodes and columns correspond to attributes. The cells of the attribute table are slots (see Figure 1).

There are three kinds of slots. A constant slot is immutable; such a slot can only be given a value once, when it is defined. A simple slot may be assigned even after it has been defined. The third kind of slot is the versioned slot, which may have different values in different versions. A sequence is a container with slots of the same data type. It has a unique identifier. Sequences may be fixed or variable in size and share common slots together. Slots in an attribute table can also be simple slots or constant slots. Once we add versioning, the table gets a third dimension: the version. Details of our version model are described in Section 7.

Trees and directed graphs are built using this data model. A directed graph is defined with an attribute (“children” attribute) that maps each node to a sequence holding its children. Trees additionally have the “parent” attribute, which maps each node to its parent. Each node in a tree or a graph can be associated with other slots defined by other attributes. WebSCM uses these attributed tree and directed graph data structures to represent important structures in a Web-based system. This data model is used as the basis of our representation for artifacts in design and implementation models of a Web-based application (e.g. concept, presentation, navigation, and implementation).
4 Supports for design in conceptual model

The starting point of a Web application design process is the elaboration of a model of the application domain, which determines the universe of discourse [26]. This task is often referred to as conceptual design in Web engineering. In conventional Web applications, i.e. those in which most of Web objects are static pages, any simple document model such as DOM [8] would be enough to model Web objects. However, in large-scaled Web sites where information may change dynamically with complex computations or queries, we need a behaviorally richer conceptual model. To address this, we have chosen the OOHDM conceptual model [26]. In OOHDM, a conceptual schema is built upon classes, relationships, and sub-systems. Classes are described as usual in object-oriented models, though attributes may be multi-valued representing different perspectives of the same real-world entity. A schema consists of a set of classes connected by relationships. Objects are instances of classes, and thus, when a relationship holds between classes, it abstracts corresponding object-to-object relationships.

Figure 2 shows an example of a conceptual schema for an online newspaper. There are stories, which can be essays or interviews. Every story has an author, and an interview is related to the person who grants the interview. Class and relation in a conceptual schema are defined as Molhado atomic components [22]. To represent this schema, we use an attributed directed graph. Each entity (class, relation, etc) is represented by a node except that each inheritance relation is represented by an edge (e.g. between “n1” and “n2”). Edges connect entities together to reflect the relationships in the schema. Therefore, they form a directed graph (see Figure 2b). Each node in the directed graph is associated with multiple attributes. The “component” attribute defines for each node a reference to the corresponding entities except for the “relation” nodes (e.g. “n4” and “n6” in Figure 2b). The “relation” nodes are associated with additional slots representing other properties of the relations such as arity, name, etc. Figure 2c) partially shows the attribute table for that conceptual schema. Each property of a class is represented by a versioned slot (in a data type) within the class component.

Data records for objects instantiated from a class are stored in an attributed tree with the depth of two, branching off the class node. Each of those tree nodes at the first level represents a record. Each record can have a sequence of fields, which are represented by children nodes of the record node. Each of these field nodes has additional attributes such as “name”, “type”, and “value” to represent the name, type, and value of each field. In general, the conceptual schema and data records for objects are represented as attributed directed graphs and trees at two abstract levels: the schema level and the individual data object level.

5 Web screen design

5.1 Presentation design

In current Web development practice, presentation information is often encoded within HTML files. However, it is always a good practice to separate the screen design phase from the coding phase for Web pages [26]. The separation between screen design and implementation allows Web designers to work on the design aspects of a Web system without worrying about concrete level of implementation in Web languages (e.g. HTML, scripting languages). Also, this separation enables an automatic generation of implementation files from the screen designs. This feature is particularly important in E-commerce Web sites whose most of Web pages are dynamically generated.

Supports for Web screen design in WebSCM can be classified into three categories. Firstly, Web developers can specify presentation information for a Web page in a CSS-like stylesheet. Secondly, in WebSCM, developers are able to work on the composition of Web screens. Pieces of information in the conceptual model can be presented on screens. A screen is either an atomic or composite component. A composite screen is a composition of atomic screens and/or other composite screens. A composite screen represents something like an HTML frame containing multiple HTML pages. An atomic screen, which does not contain any other screen, represents an individual Web page. In general, screens are defined according to Molhado component framework [22]. The compositional hierarchy of a composite screen is represented as a tree. For each tree node, the associated “component” slot holds a reference to a member screen contained within the composite screen.
5.2 Navigational design

The simplicity of HTML as the implementation language and the lack of a hypermedia design culture resulted in Web sites in which users experienced cognitive overhead and disorientation while navigating the Web [26]. Navigational design is a critical step in the design of a Web-based application. A navigational model is built as a view over a conceptual model, thus allowing the construction of different models according to users’ profiles. A navigational model provides a “subjective” view of the conceptual model.

To represent the design for navigational structure among screens, it is very straightforward to use a directed graph in which edges represent hyperlinks and each node in the directed graph is associated with a slot referring to the corresponding screen. In other words, the navigational structure is represented in the same way as the data conceptual schema as described in Figure 2 except that the “component” slots refer to screens and hyperlinks are represented as edges instead of nodes. Figure 4 shows our representation for a simple navigational structure among four Web pages. In a navigational schema, one representative edge between a pair of screens is kept even if there are multiple hyperlinks among them. The reason for our design choice is that the main purpose of navigational design is to sketch out coarse-grained linking structure, while detailed and fine-grained hyperlinks should be accommodated at the implementation level such as in HTML pages or scripts.

WebSCM aims not only to provide version control for designs in the conceptual and navigational models, but also to manage the evolution of the logical mappings among entities in those models. For example, to record the screens where a data record or object would be displayed, logical links are added from data elements in the conceptual data model to screens in the presentation model. Figure 5 shows the representation of those links in terms of mapping edges. If we put together the conceptual schema graph (representing for a conceptual schema), the conceptual data model tree (representing for data objects), and the presentation model graph (for navigational structure) with mapping edges, we have a complex directed graph in which edges are typed. For example, hyperlink edges are for navigational hyperlinks, and mapping edges for logical mappings from conceptual data model to presentation model. Details of how WebSCM versions this graph will be explained later.

6 Web documents

6.1 Structure-oriented representation

This section describes our representation for different types of Web contents at the implementation level such as markup files, program source code, scripts, etc. Web content can consist of data objects, code, and component libraries. Examples of data objects are data files, documents,
images, audios, and videos. Code can be active controls and scripts that can be embedded into an HTML page. Component libraries are reusable codes such as JavaBeans and Microsoft Foundation Classes. Among Web documents, hyperlinks exist that can point to any page in the Web. Templates such as stylesheets, which are written in some languages, enable separation of content and presentation style. To provide fine-grained version control and content management for Web documents, WebSCM follows the principle that textual information such as HTML, XML documents, and program source code are treated as structured instances, and program source code are treated as structured.

In general, this document tree representation is logically equivalent to the one supported by DOM [8]. A conversion facility is implemented to allow the import and export of DOM-compatible and XML-based documents into our representation. The maturity of XML technology [21] allows us to support a wide variety of information types that are based on XML, including graphics and animation written in SVG, XML-based design diagrams (e.g., UML diagrams), and HTML documents. Any XML-based source code formats for scripting or programming languages, along their respective structured editors, can be easily plugged into WebSCM. WebSCM currently supports an XML-based representation for Java programs similar to JavaML [2]. For code artifacts that are not XML-based, appropriate parsers and structured editors are required to generate our document tree representation. WebSCM has a built-in Java parser to parse textual Java programs into our AST representation.

### 6.2 Mappings from design to implementation

To help Web developers to keep track of which implementation files (HTML, scripts, etc) realize a particular screen in a design, WebSCM maintains the logical connections from entities in the design models to entities in the implementation model. Similar to the case between the conceptual data model and presentation model, the mappings are realized as typed edges between design nodes and implementation nodes in the representation graph and tree, respectively. Figure 7 illustrates edges from screen nodes in the screen design representation graph to file nodes in a directory structure representation tree.
7 Structure-oriented product versioning

This section describes how WebSCM provides version control and configuration management for Web contents and crucial structures in a Web-based application. First of all, the SCM model applied in WebSCM is the Molhado product versioning SCM model [22]. A version in this model is global across entire Web-based project and is a point in a tree-structured discrete time abstraction. That is, the third dimension in the attribute table in Figure 1 is tree-structured and versions move discretely from one point to another. All system objects are versioned in a uniform, global version space across a software project, called product versioning space. Molhado emphasizes the evolution of a software system as a whole. In this SCM model, the state of the whole software system is captured at certain discrete time points and only these captured versions can be retrieved in later sessions.

The current version is the version designating the current state of the project. When a particular version is set to be current, the state of the whole project is set back to that version. Changes made to versioned slots of the project at the current version create a temporary version, branching off the current version. That temporary version will only be recorded if a user explicitly requests that it be captured. On the other hand, no version is created if a simple slot is modified. To record the history of an individual object, the whole project is captured. Capturing the whole project is quite efficient because the version control system only records changes and works at the slot level [22]. It uses techniques derived from the work of Driscoll et al [11] to store and retrieve different data types in attribute tables.

To be able to provide fine-grained versioning for all logical entities and structures in design and implementation models, as well as logical connections among them, a structure versioning algorithm for an attributed directed graph is required. The algorithm assumes that editors for all logical structures or entities in models will call library functions for trees, graphs, or slots in order to modify their structures or values. Those library functions will accordingly update the values of slots in an attribute table including structural slots (e.g. "parent" and "children").

Figure 8 illustrates the algorithm via an example. In Figure 8, a “content” attribute is defined that holds string values for nodes. Other attributes can also be defined. Assume that there are two versions: v1 and v2. The shape of the graph at two versions is shown. Assume that after a sequence of editing operations on the graph, several changes had been made to both the structure and the “content” slots associated with nodes (see Figure 8). The values of versioned slots in the attribute table are changed to reflect modifications to the graph at these versions. For example, at the version v2, the “content” slot of node “n1” contains a new value (the string “1+”), and the “children” slot of node “n2” contains a reference to a new sequence object. That new sequence object contains only “n6” because the edge from “n2” to “n3” was removed. If there is a request for the values of slots associated with node “n5” at v2, undefined values will be returned since “n5” was deleted at version v2. Note that “n9” and “n10” were inserted in the attribute table at v2.

The versioning mechanism for directed graphs with edges having attributes (e.g. type) is similar except that each edge is now represented as a node. Each edge node is associated with two main attributes: “source node” and “sink node” attributes define for each edge node the source node and the destination node of the edge in a directed graph, respectively. Other attributes can also be defined, e.g., “typed” attribute defines for each edge node the type of the edge in the directed graph. The steps of the fine-grained structure versioning mechanism are still the same.

This structure versioning mechanism is powerful since
it can be used to version various forms of structures and logical mappings that were presented in previous sections. It is also very efficient since common structures are shared among versions and all information including structures and contents are versioned via one mechanism. Importantly, the algorithm is general for any subgraph, therefore, fine-grained version control is achieved for any abstraction represented by a node. The versioning mechanism for a tree is similar except that the “parent” attribute is added.

8 GUI-based functionality

This section presents GUIs for distinguished SCM functionality in WebSCM at both design and implementation levels as well as the logical connections among them.

8.1 Version control for conceptual schemas

At the conceptual level, WebSCM provides supports for Web application design with the OOHDM conceptual data model. A user can open an existing Web project or create a new one. The user selects a current working version, and opens or composes his conceptual design diagram via a specialized editor. Figure 9 shows an example of a conceptual schema of an online department store. The user can define different types of object-to-object relationships. The user is able to modify the properties of items and structures in the conceptual schema. All fine-grained changes to the conceptual schema are captured. At the data level, data is currently stored in an attribute table. No real back-end database is connected. A structural difference tool for conceptual schemas allows the user to see structural changes between two versions of a conceptual design diagram [22].

8.2 Logical connections among models

In addition to supports for designs at the conceptual data model, WebSCM also provides for developers a tool to work on presentation, screen, and navigational designs for a Web application. The right window in the Figure 10 shows the navigational and compositional designs for the UWM Multimedia Software Laboratory’s Web site. The top left window displays data objects (e.g. lab staffs, students, projects, publications, etc) in a hierarchical view. The directory structure of the Web project at the implementation level is presented in the bottom left window. In WebSCM, Web developers are able to know in which screens a data item is presented at a particular version. For example, when data object “Tien” is selected, the screen “Tien” is highlighted in the right window. WebSCM is also able to record the evolution of a mapping from a screen in the design model to actual files that realize the design of that screen (e.g. screen “Tien” and “tien.html”). In addition, developers can manage the composition of screens and map them into HTML frames or pages. If a screen is a composite entity, when users click on it, its member components will be presented such as the “Main” page in Figure 10.

8.3 Fine-grained versioning for Web documents

WebSCM manages the versions of Web contents at a fine granularity. Figure 11 displays the contents of an HTML document at two different versions v4 and v7. In the HTML editor, a user can select any document node and view its version history. For example, the un-ordered list of the HTML document in the left window is chosen. A new window is displayed to show the version history of the list (see Fig-
Figure 11. Versioning for HTML documents

Figure 12. HTML document comparison

Figure 13. History of an HTML fragment

Figure 14. Program comparison

The list’s content at the version $v_4$ is shown. The user is also able to see structural changes between those two versions ($v_4$ and $v_7$) using the structural difference tool for HTML documents (see Figure 12). Icons attached to entries show the nature of changes to the entries. In this snapshot, between $v_4$ and $v_7$, some of document nodes have been modified in terms of both their attributes and structures, for example, the “body” of the document (an “a” icon and a tree icon are both attached to the node icon). Meanwhile, the “table” node has only an “a” icon since only the color attribute of the table has been changed. Also, the third item of the un-ordered list was deleted from $v_4$ to $v_7$ since it has an “eraser” icon next to its entry in the left window.

This fine-grained, structural difference tool was easily built due to our structure-oriented versioning, in which every change to a document tree is recorded. Neither a tree differencing algorithm nor extensive analysis is required to compare different versions of an XML document tree since the SCM system works on the attributed tree representation, rather than on the textual representation of an XML document. Structural changes are stored as parent/children slots. The tool only needs to retrieve and display structural changes. No add-on to the SCM system is needed to derive structural changes. Structural differencing tools for programs and project directory structure were also built [23]. Figure 14 shows structural changes in a Java program. A similar set of icons is used to show the nature of changes.

The current built-in editing services in WebSCM for
Web documents at the implementation level include structured document editors for XML, XHTML, HTML and plain text documents, and a syntax-recognizing Java editor. Some of them are re-used from the previous version of WebSCM [23]. Any externally stored documents in these formats can easily be imported into WebSCM with built-in parsers. WebSCM is extensible to integrate any structured editors and parsers for new document formats. If the new document format is XML-based, the built-in XML parser can be re-used. For example, the drawSWF SVG graphic and animation editor [9], was easily integrated into WebSCM since its document format is XML-based.

9 Conclusions

This paper presents a novel SCM-centered development environment for Web applications that allows Web developers to manage fine-grained, structural evolution of Web contents and their crucial structures. The separation between design and implementation levels makes WebSCM well-suited for large-scaled Web-based applications. Each design and implementation model helps developers on a particular aspect of the development process of a Web application. The evolution of logical entities and structures in design models and logical mappings among them are integrally captured and related to each other in a cohesive manner. Version consistency is maintained not only among implementation artifacts, but also among all logical entities in different models at different levels of abstraction. This sort of versioning for model-to-model logical connections is feasible due to Molhado’s structure versioning engine [22]. Internal structure of a Web page is versioned in a fine-grained manner. Difference tools help Web developers to keep track of fine-grained changes at multiple structural levels. An experimental study is being conducted to evaluate the performance, efficiency, and usability of the system.

References