Object-Oriented Software Configuration Management

Tien N. Nguyen
Electrical and Computer Engineering Department
Iowa State University

Abstract

The ability to manage the evolution of a software system is critical to a successful development process. In a typical software development process, software engineers think and reason in terms of logical abstractions, their compositions and interrelations. However, existing version control and software configuration management (SCM) systems treat a software system as a set of files in a conventional file system. This creates an impedance mismatch between the design and implementation domain (semantic level) and the configuration management domain (file level). File-oriented SCM systems, whose concepts are heavily based on the storage structure, can become burdensome for developers partly because design/implementation methods and SCM infrastructures require different mental models.

This dissertation describes Molhado [2, 3, 6], an extensible and adaptable SCM framework and infrastructure that helps developers quickly create the core of an object-oriented SCM system for any application domain independent of the concrete file structure. SCM systems based on Molhado can be built to operate entirely at the logical level, eliminating that impedance mismatch.

1 Problem Statement

A variety of software development methodologies and frameworks have been introduced to provide developers with systematic approaches for producing high-quality software. These development methods often introduce *abstractions* and impose different *logical structures* among them, permitting one to concentrate on problems at some level of generalization without regard to irrelevant low-level details. For example, in object-oriented frameworks, abstractions typically are classes and objects, and examples of logical structures include class inheritance hierarchy, control hierarchy, etc. In the relational database framework, tables are related via relations, forming an entity relationship diagram.

Software development is a dynamic process where software engineers constantly modify and refine their systems. As a consequence, everything evolves. Unlike source code, for which the use of a software configuration management system (SCM) is the predominant approach to capturing evolution, approaches to managing evolution of high-level abstractions span a wide range of disconnected alternatives. Many *file-oriented* SCM systems treat a software system as a *set of files* in directories on a file system, and stable configurations are defined implicitly as sets of file versions with a certain tag. Following a development method, developers usually think and reason in terms of high-level abstractions, compositions and their interrelations. This creates an impedance mismatch between the design and implementation domain (semantic level) and the SCM domain (file level). This mismatch causes several structural and logical problems [2].

To bridge that gap between software development methods and SCM, it is necessary to have SCM systems that allow users to manage the evolution of a software system in terms of logical abstractions and relationships without worrying about the concrete level of actual file storing and versioning. In this paper, these systems are referred to as *object-oriented* SCM systems. However, despite many successes, most SCM systems are either specifically designed and too restrictive to a certain development framework or domain, or too generic and do not provide sufficient SCM supports at the logical level. For example, while architectural SCM systems support a *fixed set* of architectural objects, some text file-oriented SCM systems depend heavily on a line-oriented model of internal changes that totally disregards logical structures of contents. Unfortunately, building object-oriented SCM systems from scratch for a new domain or framework is a difficult endeavor that regularly requires a lot of time and efforts from developers.

2 Molhado Framework

Molhado [2, 3, 6] is an extensible and adaptable software configuration management framework and infrastructure that helps developers quickly create the core of an object-oriented SCM system for any application domain independent of the concrete file structure.

22nd IEEE International Conference on Software Maintenance (ICSM'06)
0-7695-2354-4/06 $20.00

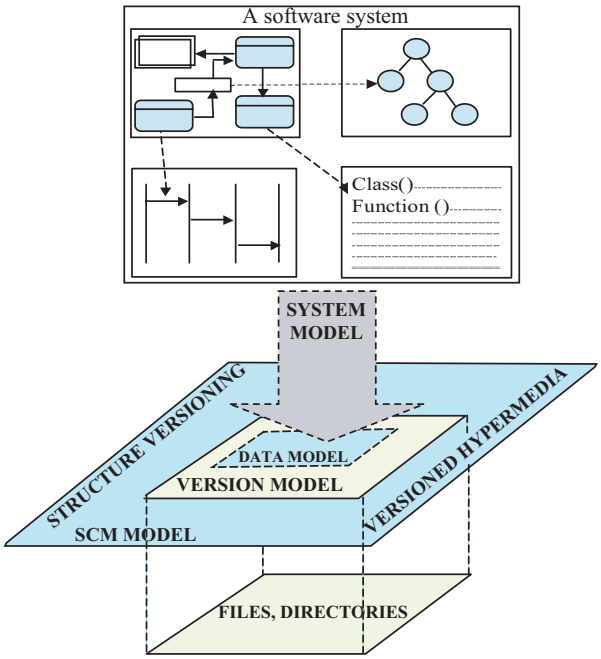

Figure 1. Molhado Approach

The ultimate goal of Molhado is to be able to manage a software system in any domain and its system objects at the logical level. That is, the developers work with the logical objects created during a software life cycle, compositions and interrelations among objects, and whenever they are ready, they must be able to commit the changes to their software artifacts without worrying about concrete file/directory representation level. When checking out their logical objects, they must be able to see changes at the logical level without using extensive analyses of file contents.

The general approach to achieve this goal is summarized in Figure 1. First of all, the data representation for software artifacts must preserve the structural and logical contents of the objects in a software system. That is the principle of our data model. The data model can be regarded as an intermediate representation between software artifacts and physical files and directories. A framework is needed to map a software system and its logical objects, structures, and relations into that data model. That framework, called *system model*, allows developers to model a software system in terms of logical objects and their relations at different levels of granularity according to any particular software development paradigm. The system model is extensible in order to support new types of software artifacts. It works at the logical level and does not depend on the physical file level. It is also able to handle different types of structures in a software artifact. To build their customized SCM systems using Molhado framework, developers must model their application domains using this system model.

The reason that text file-oriented SCM systems failed to provide SCM services at the logical level is their disregard of the logical representation of an artifact. In those systems, a version model is built on top of a (textual) data model and the version model has no knowledge about the textual representations for artifacts. In contrast to file-oriented SCM approaches, in Molhado methodology, the version model understands the basic entities of the data model. This is called the *combination approach* between the data and version models. In the combined model, a piece of data is defined either versioned or non-versioned. This combination approach facilitates version control for logical objects since their internal properties and logical structures are visible to the version model. To free developers from worrying about the concrete level of file storage and versioning, a persistent storage mechanism is needed to save/load the objects represented via the data model.

To support versioning for structured objects, it is also necessary to have a structure versioning engine. The engine represents and provides version control for any kind of structural objects that might occur within a software system. As in any SCM system, configuration management services, operation services, and transaction supports must also be provided by an SCM model. This SCM model usually takes advantage of the structure versioning engine to manage consistent configurations for complex composite objects, structures, and entire software project. While both the structure versioning engine and the SCM model must handle structure versioning, structure versioning engine works at the data model, but the SCM model works at the object and entire software system levels (i.e. at the logical level). To address the management of relationships among objects over time, Molhado suggests the enhancement of an SCM system with a versioned hypermedia model [5]. That model manages the evolution of all types of logical relationships among objects at any level of granularity.

3 Molhado Infrastructure

This section summarizes Molhado reusable SCM infrastructure that helps developers to quickly build the core of an object-oriented SCM system for any editing environment in any application domain. Molhado's modules have a layered architecture (Figure 2), in which the upper layers are constructed using services provided by lower layers. To use Molhado to build a new SCM system, developers need to implement new types of objects in terms of Java classes, and to reuse the Molhado infrastructure and its versioning engine. Those Java classes must be inherited from the base classes in Molhado's system model. They can extend or use those basic classes to model different abstractions or structures in their application domains. Logical structures are modeled as attributed, typed, nested, directed graphs.

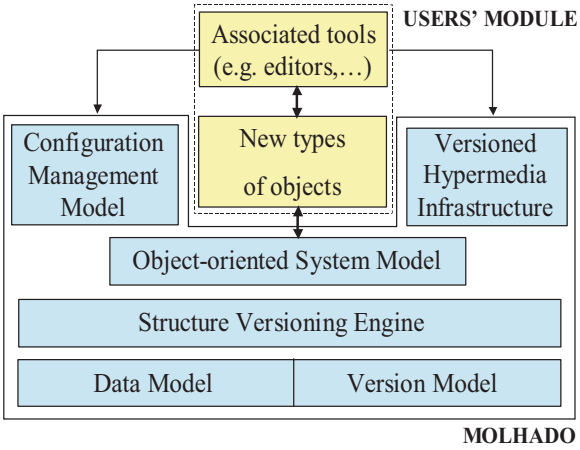

Figure 2. Molhado Architecture

One of the characteristics of the Molhado infrastructure and framework is the tight interoperation between SCM modules and specialized tools for logical objects such as object editors. The result of this integration is an *SCM-centered* development or editing environment, in which the SCM system is the heart of the environment and all changes to logical objects and structures during the development process are recorded. These specialized tools (e.g. editors) call provided library functions to manipulate object properties or structures, and to use configuration management and transaction services. When developers compile their user-defined modules (new object types and associated tools) with the reusable Molhado SCM modules, they get a custom, object-oriented SCM system for their particular environment. To work with an existing environment, a converter must be built to connect the environment's native representation and Molhado's representation for objects. The converter calls our provided functions to manipulate objects or to use SCM services. We have built such a converter as an Eclipse plugin to work with Eclipse's Java program editor.

4 Experiments and Evaluations

Molhado SCM infrastructure and tools are available both as an Eclipse plugin and a separate infrastructure. The extensible Eclipse architecture [1] is used in supporting the semi-automatic generation of SCM code for a new type of objects. Generated code from Molhado is also displayed in the Eclipse editor. From the newly generated code and Molhado infrastructure, the novel SCM system can be built and then used in the corresponding editing/design environment.

The goals of the evaluation of Molhado SCM framework/infrastructure are to examine its usability in different domains with various object types, to investigate advantages that Molhado brought to resulting SCM systems, and to demonstrate how it promotes reuse in building object-oriented SCM systems. Six prototypes of *object-oriented* SCM systems and associated SCM-centered development environments for different domains have been built using Molhado SCM framework and infrastructure. Those prototypes of object-oriented SCM systems are

1. *MolhadoArch* [4], an architectural SCM system in which consistent configurations are maintained not only among source code but also with the high-level software architecture. In current practice, a system architecture described in some graphical notation or in some architectural description language (ADL) is often versioned as text files, whose logical contents are *irrelevant* to SCM systems. With MolhadoArch, architectural entities and the traceability relationships between architectural entities and the source code that realizes them are also managed.

2. *WebSCM* [8], an SCM system for Web engineering. Key limitations of existing SCM systems for Web engineering include their inadequacy to represent Web object semantics and their inability to manage changes to the important structures among objects making up a Web site. In contrast, WebSCM is able to manage the evolution of a Web-based application in terms of Web contents, as well as crucial structures such as *navigational*, *compositional*, *internal*, and *logical* structures [8].

3. An SCM system for UML-based object-oriented software development [7]: We have built a converter that makes the internal representation of Thorn UML editor [9] compatible with our object-oriented versioning framework. In this case, logical objects in UML specifications and diagrams are modeled as Molhado components. Molhado's product versioning helps to maintain *version consistency* between UML diagrams and source code.

4. An SCM system for relational database applications [6]: Existing database management systems (DBMSs) have versioning tools only for data. There are no support for developers in managing the history or changes to their designs over time. To address this, it is very natural to take advantage of the ability of Molhado to manage structures at the logical level. This prototype showed its unique ability to help not only to manage versions of ER diagrams and tables, but also to control fine-grained and structural changes between versions of both designs and data.

5. A fine-grained version control system [7]: Applying Molhado approach for software documents, we easily produced a multi-level, fine-grained version control system for Java programs and structured documents. The main idea is to implement a program or a structured document as a Molhado object by using our versioned tree data structure. The system supports versioning of any fine-grained units at any structural level in a software document.

6. *MolhadoRef*: is a refactoring-aware SCM tool that is capable of capturing and versioning of the semantics of Java

program entities and refactoring operations that were performed on those entities. It helps track the history of refactored, fine-grained program entities, and eliminates many conflicts when merging refactored versions in multi-user environments. It uses the operation-based SCM approach to represent and record refactoring operations as first-class entities in the repository.

With respect to version and configuration management, those object-oriented SCM systems have shown clear improvements over existing ones in the same domains. Their distinguished SCM characteristics are due to our object-oriented SCM approach. Also, they support artifacts that have different nature than source code and live in contexts that are traditionally not suitable for existing SCM systems.

5 Key Contributions

In brief, Molhado is the first fully object-oriented SCM infrastructure and is based on an advanced model of version control that allows it to achieve a high level of reuse. SCM systems that use Molhado have the ability to manage the evolution of logical objects, compositions, structures, and the logical connections among them in different models at various levels of abstraction and granularity. Furthermore, with Molhado's product versioning model, consistent configurations that correctly relate software artifacts from all phases of a software process are easily maintained.

SCM systems based on Molhado gain several other advantages. Molhado's structure versioning framework facilitates the construction of structure-oriented difference tools for various types of software artifacts. There is no overhead from extensive analyses of file contents to recover structural changes, as would be required when using file-oriented SCM systems. Molhado's flexible logical system model is able to accommodate complex software development paradigms in many application domains. A wide variety of prototypes of SCM systems have been built using Molhado, showing its ability to support software artifacts that are quite different in nature from source code such as hypertexts, Web applications, design diagrams, software system architecture, and relational database applications.

The Molhado versioned hypermedia model [5] provides supports for managing the evolution of fine-grained logical relationships among objects and the versions of complex hypermedia structures in a software system. Hyperlinks are explicitly represented as first-class entities with variable arity and are managed separately from document content, thus facilitating systematic analysis, information retrieval, browsing, navigation, and visualization of document relationship networks. Molhado not only supports SCM for hypermedia structures in a fine-grained manner, but also provides version control for individual hyperlinks. Molhado is the first versioned hypermedia model that is based on product versioning. It overcomes several serious issues identified by earlier research on versioned hypermedia. The dissertation research also addresses a parallel problem to logical relationship management, which is keeping software artifacts consistent or conformant to each other during the evolutionary process of a software system [2].

6 Conclusions

This dissertation presents *Molhado*, a methodology and its extensible, adaptable SCM infrastructure that can help developers quickly create the core of an object-oriented SCM system for any application domain independent of the concrete file structure. SCM systems that use Molhado have the ability to manage the evolution of logical objects, compositions, structures, and the logical connections among them in different models at various levels of abstraction.

As with any course of research, future work remains to be done. Our current implementation requires the addition of routines into a user-defined class. An semi-automatic tool via macro expansion is currently being developed to generate all of those routines in user-defined classes.

References

[1] Eclipse. http://www.eclipse.org/.

[2] T. N. Nguyen. *Object-oriented Software Configuration Management*. PhD thesis, University of Wisconsin, Milwaukee, USA, 2005.

[3] T. N. Nguyen, E. V. Munson, and J. T. Boyland. Object-Oriented, Structural Software Configuration Management. In *Proceedings of 19th ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2004), Formal Demonstration*. ACM Press, 2004.

[4] T. N. Nguyen, E. V. Munson, J. T. Boyland, and C. Thao. Architectural Software Configuration Management in Molhado. In *Proceedings of 20th International Conference on Software Maintenance (ICSM 2004)*. IEEE Computer Society, 2004.

[5] T. N. Nguyen, E. V. Munson, J. T. Boyland, and C. Thao. The Molhado Hypertext Versioning System. In *Proceedings of the Fifteenth Conference on Hypertext and Hypermedia (Hypertext 2004)*, pages 185–194. ACM Press, 2004.

[6] T. N. Nguyen, E. V. Munson, J. T. Boyland, and C. Thao. An Infrastructure for Development of Object-Oriented, Multi-level Configuration Management Services. In *Proceedings of 27th International Conference on Software Engineering (ICSE 2005)*, pages 215–224. ACM Press, 2005.

[7] T. N. Nguyen, E. V. Munson, J. T. Boyland, and C. Thao. Multi-level Configuration Management with Fine-grained, Logical Units. In *Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2005)*, pages 248–255. IEEE Computer Society, 2005.

[8] T. N. Nguyen, E. V. Munson, and C. Thao. Structured Software Configuration Management for Web projects. In *Proceedings of the Thirteen International World Wide Web Conference (WWW 2004)*, pages 433–442. ACM Press, 2004.

[9] Thorn UML editor. http://thorn.sphereuslabs.com/.