A Software Development Project-Adapted Architecture of the Extensible Version Control System

Ryo HAYASAKA (010089)

School of Information Science,
Japan Advanced Institute of Science and Technology

February 14, 2003

Keywords: version control system, CVS, layered architecture, object-oriented application framework, process support.

Since there is no process support function in a version control system, the development process of a project which is using a version control system is performed by the project administrator. When the scale of a project becomes big and a number of artifacts in a version control system increases, the project administrator's load becomes large. Since a development project has an original development process in general, required process support functions also differ. Extendibility is required in order to add such process support functions to version control systems.

On the other hand, Software Configuration Management (SCM) Systems have advanced functions, such as process support functions, process support functions, change management functions, distributed development support functions, etc. besides version management functions. In general, SCM systems take an approach which makes development processes customizable rather than makes them extensible. The development processes are predefined by the SCM vendors. However, if a development project of which the development process is out of the target processes of SCM systems tries to use the process support functions, SCM systems cannot realize such process support functions. There is also a problem that SCM systems cannot be used simply. So, this is one of a reasons that simple version control systems are still widely used.

In this paper, a way of construction for a extensible version control system which can be easily added extended functions various development projects require is proposed. As opposed to SCM systems, it is a meanings to the construction for a extensible version control system which can be easily added extended functions various development projects require.

To realize extensibility, object-oriented (OO) frameworks are used. They have hot spots which are not implemented. Extensibility is realized by extending functions in hot spots. There are some basic class structures in extensible hot spots. High extensibility is realized by using design patterns and metapatterns based on the basic class structures. An OO programming language used to implement OO frameworks provides subclassing and object composition techniques and incremental programming can be used. By using this, the programming cost to extend functions is low.

layered architecture using OO framework is adopted for extensibility. According to the structure clearly separated in upper OO framework layers and the lower version control layer, extensibility is realized by reusing the implementation of a version control system and using OO frameworks. The explanation of each layer is as follows:

- Extended Function Framework Layer

  This is a OO framework specialized for a domain of a extended function required by development projects. A OO framework is divided into the Basic System Framework layer and this layer so that the implementation cost and the complexity of the system decrease. This layer is programmed with low cost by using incremental programming for hot spots in OO frameworks.

- Basic System Framework Layer

  This layer implements the fundamental functions which the system adopting this architecture has. By making it the extensible structures which can extend a function, this layer provides extensibility to the upper layer. The upper layer can add new functions and customize the existing functions. The following four basic functions are designed as hot spots: (1) managing the flow of control function, (2) event definition function, (3) task manager function, and (4) event handler function.

- Adapter Layer

  This is used as a wrapper. The role of this layer is conversion of interface between a version control system and an OO framework. Using this layer, the upper layer does not depend on the version control system directly. So developers of OO framework layers can concentrate.

- Version Control System Layer

  The implementation of a version control system is reused. This layer provides version control functionalities. The version control system does not be changed. So, the cost of implementing this system is reducible.

The design and implementation of a prototype system system based on this architecture were done and the feasibility was showed. CVS (Concurrent Versions System) was used as the version control system layer. Using the client / server function of CVS, the implementation architecture which adopted a proxy placed between a client and a server was used. CVS adapter which analyzed CVS protocol was implemented and called a function of the basic system framework as necessary. Because of this, an extensible version control system was realized without modification to CVS.

I designed and implemented highly extensible OO frameworks including the four hot spots as described above, applying design patterns and metapatterns. Especially, by using "managing the flow of control function," it is possible to customize the timing calling for extended functions.

As an extended function, I picked up an access control function and showed the design and implementation of ACL (Access Control List) and RBAC (Role Based Access Control). ACL is a way to have a pair of a user and permissions to available resources. RBAC uses roles defined according to the structure of the organization and permissions which defined the role can issue permitted operations. These are based on completely different access control models. A user of this system can select one of the functions and customize some hot spots easily. The programming costs drastically decreased by using incremental programming to the hot spots of the OO frameworks.