ECAI 2009 - International Conference – 3rd Edition
Electronics, Computers and Artificial Intelligence
3-5 July, 2009, Piteşti, ROMÂNIA

Transaction over versioned objects in hierarchical workspace environment

VLADIMIR Jotov
PhD Student, Veliko Tarnovo University "St. Cyril and St. Methodius"
2 Teodosij Tarnovski str., Bulgaria
vjotov@acm.org

Keywords: version control, versioned object, transactions, hierarchical workspaces

Abstract. *This article is an attempt to present and calcify transactions performed over versioned objects. The examined SCM environment model is based on hierarchical composition of workspaces. The presented classification is elaborated on principles of versioned objects visibility from parental workspaces to its children workspaces.*

INTRODUCTION

In [4, p.406] says that *"…a modern workspace is created "behind the science" to perform a particular user-selected task…"* In this article we will try to elaborate the workspaces as a major infrastructure element of SCM systems. In section 1 are presented different aspects of workspace element in SCM systems. The presented transaction classification is based on two principles of versioned object visibility among workspaces – section 2. Transactions classification building versioned-object lifecycle are presented in section 3 and section 4.

1. RELATED WORKS

In software systems development the workspace is associated with an isolated place for development. In [4] is defined a workspace as a system element that provides three essential functions:

• Sandbox where users can free edit their files.

• Building of selected software version and/or configuration.

• Isolation of changes, testing, debug without interfering with other users.

In [4, 1, 8] authors understand as workspace work isolation using file directory.

On other hand in [2] workspace is defined as place for project collaboration. The challenge in front of that interpretation of workspaces is to ensure concurrent access policy to the versioned objects. The most common technique for assuring isolation is the check-in/check-out method using lock-modify-unlock or copy-modify-merge approaches [3].

As a consequence of above we can conclude that two approaches of workspace use exist:

- Isolation in order to avoid interfering during parallel work.
- Collaboration place where teams gain synchrony and integration.

Hierarchical workspaces

Trying to gain the balance between the collaboration and isolation of workspaces it has been naturally evolved to use hierarchies of workspaces. For example in [7] authors attempt to resolve the concurrency access to workspace meta-data, land lock mechanism the authors propose simplification of access policy by applying a two-level hierarchy of workspaces compound of group workspace and private workspaces. As effect of that authors predict decreasing of contention only during simultaneous check-in or check-out in group

workspace [5]. In [8] it is proposed multi level hierarchy of workspaces with file based mechanism for check-in/check-out from parental workspace. In [4, 6] are presented virtual workspaces or views as a manner allowing write access to selection of versioned objects and read-only access for other versioned objects. As a reason for appearing virtual workspaces it is pointed out increase of system speed by reducing required system resources.

In current article we are abstracting from inherited file based meaning of versioned object. We will focus an object abstraction of software artifact. It could be constructed not only of source code element but of related binary code element.

2. PRINCIPLES OF VERSIONED OBJECTS VISIBILITY WITHIN HIERARCHY OF WORKSPACES

The further presented transactions are based on two principles for visibility of versioned object within a hierarchy of workspaces.

The main principle we can define as: *Workspace local version of a versioned object is visible within that workspace in despite of existing versions in parental workspaces.*

The second principle could be defined as: *One version of versioned object from certain workspace is visible in all children workspaces unless there are no local versions* (e.g. main principle). The defined visibility model is similar to the model of "view" of ClearCase [6]. Additionally this principle is recursively applicable for the whole tree of workspaces.

Coming both principles we could enunciate the following consequence: *In certain workspace none local versioned object is presented by its version situated in closed in the hierarchy parental workspace*.

3. TRANSACTION OVER VERSIONED OBJECTS WITHIN WORKSPACE

In this section we will try to present transactions over versioned objects within one workspace. Below we delineate the following transactions: creation of versioned object, update of not-local versioned object, state-mark creation of local versioned object, "delete object" state-mark, and rollback of a "state-mark".

Creation of versioned object is the birth transaction of each versioned object. The object is created with its initial form and version.

State-mark creation of versioned object – Here under the term "state-mark" we will understand a single immutable version of certain versioned object. We can regard on this transaction as a mechanism for making save-point, it related with roll-back transaction described below.

Update of not-local versioned object is a compositional transaction over a versioned object that is not situated in local workspace but in one of parental workspaces. The components of this transaction are: a check-out of the versioned object from parental workspace, creation of new object state-mark in local workspace.

"Delete Object" state-mark is a method for excluding selected versioned object from current workspace in despite of its existence in parental workspaces. This state-mark makes the versioned object "invisible" for further manipulations in local workspace.

As we mentioned before the purpose of introducing state-marks is caused by *rollback* functionality that shall be supported by versioning system. Using this transaction we can always destroy last state-mark and to get back to previous one.

4. TRANSACTIONS OVER VERSIONED OBJECTS AMONG WORKSPACES

In this section we will present transactions over a versioned object among parent and child workspace.

Publication transaction is the essential transaction in presented research. Publication could be compared with check-in transaction. The publications may be divided in four groups: simple publication, update publication, merge publication and publication of "delete object" state-mark. *Simple publication* is performed when we deal with new versioned object in local workspace – The transaction includes a move the versioned object to parent workspace and as it is a new object there are

no versioning conflicts. After movement the versioned object became visible from parental workspace (fig.1)

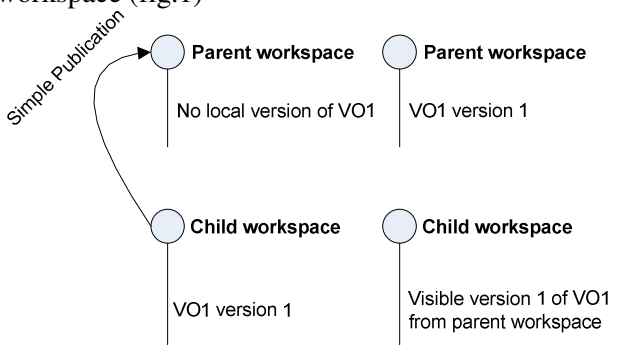

Fig.1 Simple publication

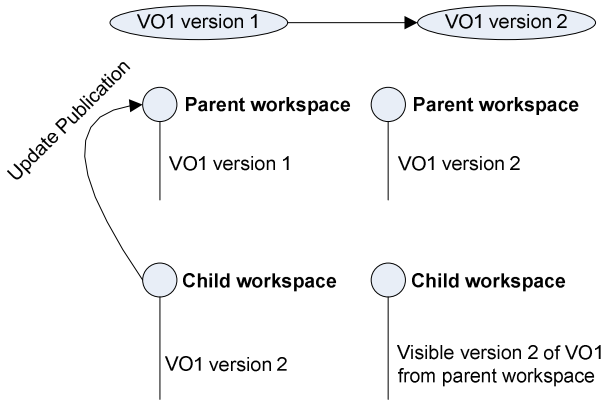

Fig. 2 Update publication

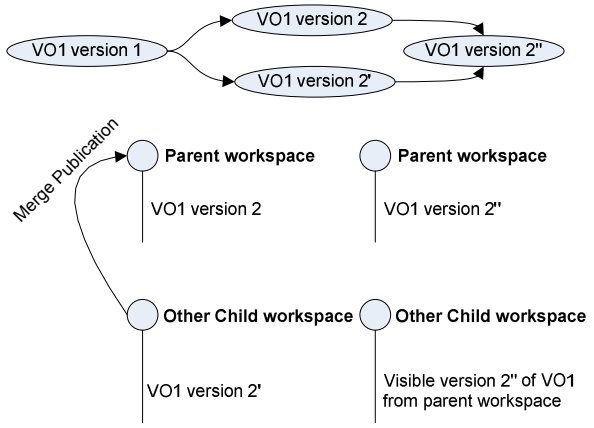

Fig.3 Merge publication

Update publication is performed when version of versioned object in child workspace is a straight inheritor of version in parental workspace. Another precondition for update publication is when we have a straight chain of state-marks in child workspace. Performing this transaction version is set in parental workspace as a state-mark or is moved as chain of versions from child workspace to parent one (fig.2).

Merge publication is executed when version in parent workspace and child workspace are historical siblings or branches of one version. The situation is the same and for chain of versions. On this transaction we perform merge of two states. In [4] are described three possible approaches for merging: line-base, syntactic-oriented and semantic-oriented. After this transaction the merged version of versioned object in form of new state-mark is visible in all child workspaces of parent one (fig. 3).

Publication of "Delete object" state-mark is an update transaction. Using it versioned object's deletion or invisibility is published in form of state-mark. As a consequence versioned object becomes invisible for all child workspaces where local version of it is missing.

Put-back transaction described in [8] is a method that allows performing undo of versioned object's changes. This transaction is similar to rollback transaction described in previous section. The difference here is the fact that Put-back transaction in performed on workspace level (fig. 4). The versioned object local version is removed from workspace. Follow-up principles described in section 2 in local workspace will be visible the version of versioned object from parental workspace.

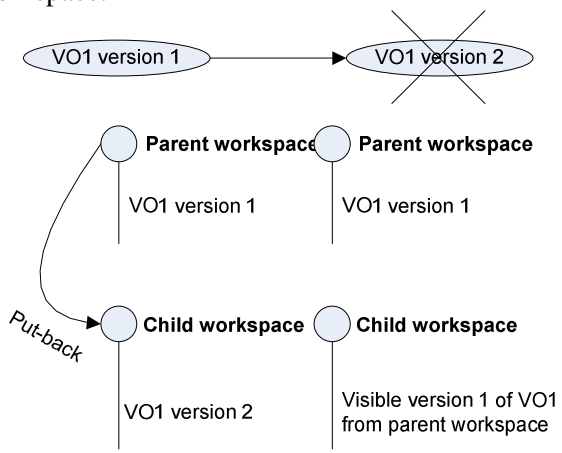

Fig.4 Put-back transaction

CONCLUSIONS

Current research allows be used as a background research in area of workspace in SCM systems. Using an appropriate workspace configuration policy it could be gained better software development, support and maintenance via presented transactions. Another aim of the paper is to provide fundamental on the versioned object lifecycle researches.

REFERENCES

[1]. Cederqvist, P., CVS—concurrent versions system.http://ximbiot.com/cvs/manual/cvs-1.11.22/cvs.html, 2006, (accessed 20.03.2009).

[2]. Clemm, G., Amsden, J., Ellison, T., Kaler, C., and Whitehead, J., Versioning Extensions to WebDAV (Web Distributed Authoring and Versioning). RFC3253. RFC Editor, 2002.

[3]. Collins-Sussman Ben, Brian W. Fitzpatrick, C. Michael Pilato, Version Control with Subversion, Draft Revision 9837, http://bebas.vlsm.org/v06/tools/svn-book.html, 2004 (accessed 20.03.2009).

[4]. Estublier, J., Leblang, D., Hoek, A., Conradi, R., Clemm, G., Tichy, W., and Wiborg-Weber, D., Impact of software engineering research on the practice of software configuration management, ACM Trans. Softw. Eng. Methodol. 14, 4 (Oct. 2005), pp 383-430, 2005.

[5]. Katz, R. H., A database approach for managing VLSI design data, In Proceedings of the 19th Conference on Design Automation Annual ACM IEEE Design Automation Conference. IEEE Press, Piscataway, NJ, pp. 274-282, 1982.

[6]. Posner, J., Block, J., CASEVision™/ClearCase Concepts Guide, 1994, http://techpubs.sgi.com/library/dynaweb_docs/0620/SGI_Developer/books/ClrC_CG/sgi_html/index.html (accessed 20.03.2009)

[7]. Silva, M., Gedye, D., Katz, R., and Newton, R., Protection and versioning for OCT, In Proceedings of the 26th ACM/IEEE Conference on Design Automation, Las Vegas, Nevada, United States, June 25 - 28, 1989, DAC '89, ACM, New York, NY, 264-269, 1989.

[8]. Sun WorkShop TeamWare User's Guide, http://docs.sun.com/source/806-3573/TeamWareTOC.html (accessed 20.03.2009).