I.J. Education and Management Engineering, **2017, 4, 13-20**
Published Online July 2017 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijeme.2017.04.02

Available online at http://www.mecs-press.net/ijeme

Version Management by Time Based Approach in Modern Era

Hafiz Suliman Munawar[a], Usama Khalid[b], Rabia Jilani[c], Adnan Maqsood[d]

[a] *Research Centre for Modeling & Simulation ,NUST Islamabad, Pakistan*
[b] *CS Department COMSATS Institute of IT, Wah Cantt, Pakistan*
[c] *Research CenterUniversity of Huddersfield, Huddersfield, England*
[d] *Research Centre for Modeling & Simulation ,NUST Islamabad, Pakistan*

Abstract

Version Management is the most crucial step in Software Configuration Management. The major phases of version management is analysis of changes made, the newer version that holds all the changes is then made available. The existing techniques involve version control by means of file locking and merging of version to control and monitor the use of files and manage the changes made in them, both these approaches have certain shortcomings including loss of changes made and unnecessary delays. Hence, a new technique has been devised, the algorithm of which has been proposed in this paper. This latest model of version control comprises of a mechanism of updates based on time. There is an integration of timer and other controls which make sure that the implementation can be made by more than one user at a time and that none of the modifications are lost. This algorithm of version control has been proposed to refine the process further and open new windows for future research in this area.

Index Terms: Checks Out, Check In, Repository, Working Copy, Configuration Item, Baseline, Release Management.

1. Introduction

Humans are constantly in search of change and betterment in all matter of life. The field of software is no different and constant changes are being implemented for removal of bugs and addition of newer features in order to enhance the usage of technology. Changes have to be accommodated in software for upgrading the performance and usability of computer software. The field of software engineering hence requires a systematic approach for handling different tasks that come in line with changes. These changes can be needed owing to newer requirements, certain errors with the older versions or the integration of newer computers that do not support the older software versions. In line with all these changes, the overall performance of the system have

* Corresponding author.
E-mail address: hafizsuliman91@gmail.com, usama_khalid@hotmail.co.uk, jilani.rabia@gmail.com, adnan@rcms.nust.edu.pk

to be enhances along with improved reliability [1]. The major problem here is that management of these changes and tracking where and when the changes are coming from. To understand these changes and all associated requirements, there is a need to have knowledge about management of software configuration. Software is developed by means of a rigorous and well defined layout that contains many phases, together known as the Software Development Life Cycle (SDLC). It is a universally accepted Life Cycle.

The cycle comprises of specific outputs for each phase, known as the software artifacts. Code nodules, SRS, Plans, Client documents, User manuals, development models and test cases are some of the artifacts. After carefully outlining these artifacts, the next stage is identification of correlations between these different entities. In case of SDLC, the output of one phase serves as the next phase's input; this makes a case of dependencies [2]. Examples include dependence of a code on an object diagram or a class. Testing of these artifacts, their inputs and outputs depends on the functional requirements. Collectively, the artifacts, their state and the interdependencies are summed and known as a configuration; the Software Configuration.

The process of configuration management is by definition, the art of identifying, subsequently organizing and gaining control to manage modifications in the software that is being made by the programming team. SCM is a discipline of engineering that is concerned with the methods and tools pertaining to system configuration management [3]. The activities involved in SCM are identification pf baselines followed by its establishment, review. The next phase is managing a control over the changes and formulating reports and audits of all modifications pertaining to the software [4]. The release of software and all activities associated with it are documented and managed by the supplier of SCM.

1.1. Version

A version is among the examples of configurations. Version refers to the particular attributes of a software and changes or evolves along with the constant evolution of software. As the software development undergoes changes, modifications and up-gradation, so does the version. All these modifications and different versions comprise of different configurations and take up complex version space. The upgraded version of a system may be having better features, properties and functions such as minimized errors and better performance [5]. There is also a chance of different versions having a mere change in the configuration of its hardware or software, without any change in the performance or functions. In case of negligible changes, the version may be referred to as a 'variant' of the given version. The release version of a system is the one that is distributed to clients. Every new release must be equipped with newer functions and better performance to support different hardware.

1.2. Version Control

Version control is a very important aspect of SCM and is also known by the terms revision control and source control. The basic idea of version control is to manage the modifications made in documents, on websites, in the computer program or any other large body of data. This control system is made up of varying technologies and techniques that are required to control the changes made to a file, particularly in documents, web contents and source codes. The modern era has seen such an importance of version control that no project is taken seriously unless you incorporate or make use of version control [6]. If nothing else, at least the source code of your project should have the source code managed by version control. Such importance of version control is attributed to its functionality in aiding the smooth run of projects and providing ease in managing changes, communication and management of version release, code stability and the authorization of changes by different developers and users. All these needs are met by having a version control system which does to provide a means of coordination for all areas [7]. The changes are merged, communicated and controlled accordingly.

2. Related Works

SCM is the software system evolution and management process. It has been defined as the process of changes in system by Somerville. Cerain laws were put forward by Lehmans and Belady which are applicable to all evolving systems [8]. SCM does to maximize the production of high quality end products and minimizes errors. Version control, change control, reporting and auditing are the activities included in configuration management. These laws are as follows:

- Continuing Change
- Continuing Growth
- Conservation of Familiarity
- Declining Quality
- Feedback system
- Increasing Complexity
- Large program
- Organizational stability

Continuing change can be program that has to be used in the practical world needs to be modified in order to remain useful. This leads to continuous growth , in order to be satisfying and useful to the users, the functionality of the system has to undergo constant changes. Familiarity is being conserved throughout the entire span of a system, the overall increase in changes remains more or less constant. If changes does not met, the quality of the system will decline. The quality of the system will decline. The constant evolution process requires a well formed feedback system which is absolutely necessary in order to track changes for maintaining the product and improve it likewise by feedback of the system. If changes are starts occurring continuously it increases the complexity of the system. The constant changes incorporated in the system make it more complex.

This required devotion of extra resources to manage the changes and keep the system simple. This result in the evolution of programs is self-regulated. The attributes of system such as the time between release and the total error reported are fixed and unchanging for each release. The rate of development of a program remains unchanged throughout the course of its development and does not in any way depend on the resources involved, in this way organizational stability can be achieved Detection of changes is only possible in cases where one has been working on that particular project from the beginning. SCM begins at the very start of a project then tracks and controls the modifications after development. The SCM has been defined in IEEE-828 standard. The applicability of this standard includes the entire span of critical and non-critical software and also the software which has already been developed [9]. There is no restriction of class or type of software in the application of the said standard. The SCM Plan outline all the activities required and the key steps that should be carried down during all those activities. It also documents the required resources, people involved in the process and when they make changes in the software. Hence the SCMP is better known as a well-defined layout including details which can easily be categorized as follows:

2.1. SCM Activities

The activities include configuration items identification, management of promotion, release management, management of change, variant management and branch management. There are two existing methodologies of version control, one of them is a centralized control and the second is a distributed control [10]. The centralized version control comprises of a common repository having a network that is client-server based. The changes are stored in this repository and then published from here. The basic idea of a centralized version control is to have a single source database where all the changes of a particular project are present and all the developers have their working copies on which they make changes and send back to the main repository. The

distributed control on the other hand has various repositories, such that all developers have their own database and all changes made can be swapped between the different repositories and the decision of assigning the role of the master repository lies with the people involved. In this case, a peer-to-peer network is needed. There are two methods involved in centralized control, these methods include

2.2. File Locking

It is a very simple model, known as 'lock modify' model. Such a model ensures that the repository sends out a file to only one person in a given time. This is done to make sure that there is no problem due to several people accessing and changing the same files. The locking function ensures that one developer gains access at a given time to make changes. While others can only view it in that time and only gain access for modifying it after the first developer returns it to the central repository [11].

The person who is working on the file at a given time has to lock it to ensure that no other person can make changes to it. Once done, they will have to unlock it in order to make it available for others. The next user requires an administrator to unlock the file for them. This might cause unwanted delays which might even lead to false serial numbers being assigned to files owing to confusion over access.

Suppose a repository has two files being worked upon by two people. One of these files is dependent on the other. If the first person locks their files, the second file will also be locked due to its dependence on the first and hence the second developer will not be able to work [12]. This shows that there are more negative consequences of file locking then positive approaches. There is a limited use of file locking and it is mainly used only when the developer is only working on small changes. A Lock-Modify-unlock model is offered by Subversion but its use is limited [13].

2.3. Version Merging

It is model that allows different people to access the files and have their working copies. They can make modifications in their personal copy of the file and then return it, all the changes from different sources are merged and a final version is added to the repository. This merger is carried out by the system in a very short span of time to ensure that none of the changes are lost. It is extremely important to make sure that there are no errors in the merger program [14]. This is followed by the Concurrent Version System, which is used to maintain a history of all changes made to a program file [15].

Automatic merging can be carried out by delta compression. This makes use of a revision control which carries out the merging of changes. This type of merging is required when different people work on the same program file simultaneously. The other case is manual merging, which is followed when the developers are granted permission to choose methods for making change on their own. This is mainly used when variants of a file are used and the user wants the changes to be maintained in both. Also, this merger can be employed when there are errors in the automatic merging [16].

2.4. Shortcomings of Existing System

Certain shortcomings are associated with version control in case of file merging. The maintained repository has to be updated of all changes. If two people work on the same file, make changes and then try to update in the repository, only one will be able to save it in the repository at a time and the other persons request is refused so, unfortunately second person has to make request again and again in order to get files from the repository. This is the main drawbacks of the systems currently in used. The differences between the two methods are given in the table 1 below:

Table 1. Comparison between File Locking and Version Merging

File Locking	Version Merging	File Locking
Access to the central repository is granted to only one person at a time.	The same document can be edited by various people at a time.	Access to the central repository is granted to only one person at a time.
There is no possibility of parallel work.	It allows parallel work to be carried out	There is no possibility of parallel work.
The file can be worked upon by one person, others can only view it.	The same file can be worked upon by different people.	The file can be worked upon by one person, others can only view it.

3. Proposed Methodology

The previous version controls have been analyzed and careful considerations enabled us to propose this model. The proposed model aims to rectify the shortcomings of all previous models and provide a newer and better model of version control. This paper presents a newer technique, which is the time-bound automatic updating model. Figure 1 outlines the three key steps of this model. The first step is the file present in the repository is sent to a temporary storage so as to enable users to obtain their working copies easily. The second step is assigning these working copies to different users. A timer is also used, after the users have processed their files, this timer does to update the files in the temporary storage. An auto-updating system ensures that changes by all users are maintained.

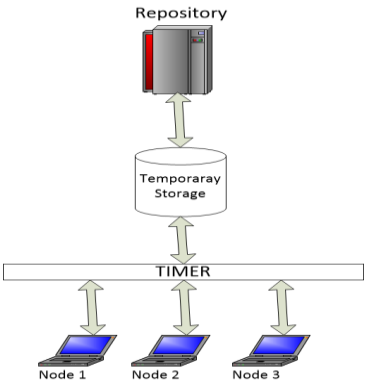

Fig.1. Time-based Model for Version Management

3.1. Timer

As the name implies, the timer is an algorithm that ensures scheduling. Once the users request for a file, the timer acts as a scheduling. After the allotted time, the timer acts as an updating mechanism, it recovers the files from the users with all changes made and updates those changes in the compiles repository file, without hindering the work of developers.

3.2. User Request Flow

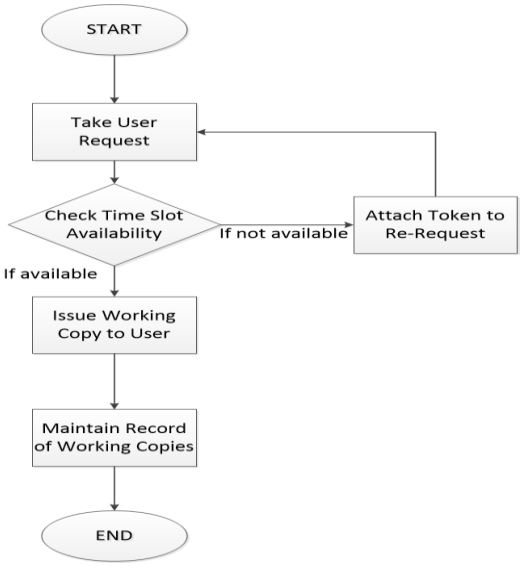

Fig.2. Flow Diagram of User Request

3.3. Auto Update Flow

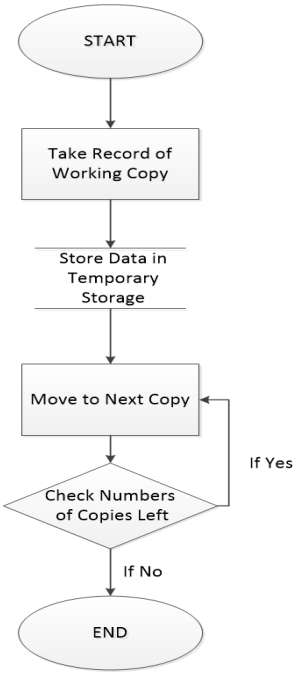

Fig.3. Flow Diagram of Auto Update

4. Conclusions

This paper has presented an algorithm for version control for SCM. Even though a number of techniques already exist, but there are certain shortcomings in those techniques. Hence, we have proposed this algorithm to overcome the previous barriers and make version control better and more refined. The integration of timer will ensure that there is no or very little chance of error, the scheduling algorithm does to make sure that different users can work simultaneously, without any changes being lost and no interruption in the work of any of the users.

The proposed model provides a great window for future research as this paper only proposed the model, the tool has not been developed yet and will be under formulations in near future. This model requires implementations, which can be achieved by means of a number of resources, particularly a client-server network and a tool. Further refinements of this model can lead to a better working system.

References

[1] Gerth, Christian, Jochen M. Küster, Markus Luckey, and Gregor Engels. "Detection and resolution of conflicting change operations in version management of process models." Software & Systems Modeling 12, no. 3 (2013): 517-535.
[2] Shekar, Raja, Armeet Deulgaonkar, Ravindra Kumer, and Anurag Palsule. "System and method for server configuration control and management." U.S. Patent 8,122,111, issued February 21, 2012.
[3] Coronel, Carlos, and Steven Morris. Database systems: design, implementation, & management. Cengage Learning, 2016.
[4] Altmanninger, Kerstin, Martina Seidl, and Manuel Wimmer. "A survey on model versioning approaches." International Journal of Web Information Systems 5.3 (2009): 271-304.
[5] Collins-Sussman, Ben, Brian Fitzpatrick, and Michael Pilato. Version control with subversion. " O'Reilly Media, Inc.", 2004.
[6] Lehman, Meir M., and Laszlo A. Belady. Program evolution: processes of software change. Academic Press Professional, Inc., 1985.
[7] Tichy, Walter F. "RCS—a system for version control." Software: Practice and Experience 15.7 (1985): 637-654.
[8] Emre Celebi, M., et al. "Border detection in dermoscopy images using statistical region merging." Skin Research and Technology 14.3 (2008): 347-353.
[9] Yongchang Ren, Tao Xing, Qiang Quan, Ying Zhao (2010) Software Configuration Management of Version Control Study Based on Baseline, 3rd International Conference on Information Management, Innovation Management and Industrial Engineering.
[10] Cederqvist, Per, and Roland Pesch. "Version management with CVS." (1992).
[11] Ian F. Sommerville (2007) Software Enginnering, 8 edn., : Addison-Wesley.
[12] Git version control, Available at: git-scm.com.
[13] Jacky Estublier, David Leblang, Andr Ë Van Der Hoek, Reidar Conradi Ntnu, Geoffrey Clemm, Walter Tichy (2005) Impact of Software Engineering Research on the Practice of Software Configuration Management, ACM Transactions on Software Engineering and Methodology.
[14] Christian Manz and Michael Stupperich (2013) Towards Integrated Variant Management in Global Software Engineering: An Experience Report, IEEE 8th International Conference.
[15] IEEE (2005) IEEE-828 Software Configuration Management Plan.
[16] Version Control Technical Structure Retrive from: http://producingoss.com/en/vc.html

Authors Profiles

Hafiz Suliman Munawar received his B.Sc Engineering degree in 2014 from UET Taxila, Pakistan. Currently he is MS research student at Research Center for Modeling and Simulation, National University of Sciences and Technology (NUST), Pakistan. With a competent interest in multidisciplinary studies.

Usama Khalid is working as a Research Assistant in Telesehat Pvt. Ltd. Pakistan and received his BS (Software Engineering) degree from COMSATS Institute of Information Technology, Pakistan in 2016. He has been working on TeleDiagnosis to optimize its performance.

Dr. Adnan Maqsood is working as Assistant Professor at Research Center for Modeling and Simulation, National University of Sciences and Technology (NUST), Pakistan, since 2012. He received his Bachelor degree in Aerospace Engineering from NUST, Pakistan in 2005 and PhD from Nanyang Technological University (NTU)

Rabia Jilani received her B.Sc Engineering (IT) degree in 2012 from UET Taxila, Pakistan and PhD in 2016 from University of Huddersfield (UK). With a competent interest in Software Engineering/AI/Pattern Recognition/Machine Learning

How to cite this paper: Hafiz Suliman Munawar, Usama Khalid, Rabia Jilani, Adnan Maqsood,"Version Management by Time Based Approach in Modern Era", International Journal of Education and Management Engineering(IJEME), Vol.7, No.4, pp.13-20, 2017.DOI: 10.5815/ijeme.2017.04.02