Architecture of a Software Configuration Management System for Globally Distributed Software Development Teams

Muhammad Wasim Bhatti
Engineering Management Department
CASE, Center for Advanced Studies in Engineering,
Islamabad, Pakistan

Irfan Anjum Manarvi
Engineering Management Department
CASE, Center for Advanced Studies in Engineering,
Islamabad, Pakistan

ABSTRACT
The phenomenon of global software development has changed the traditional methods of software engineering. Along with several benefits, globalization brings lot of challenges for practitioners of global software development. Among all challenges, establishment of a configuration management system for distributed teams is one of the major technical challenges. Therefore, in this study, it has been investigated that what type of configuration management system should be established and what should be its architecture for globally distributed software development teams. It has been proposed that a centralized configuration management system, designed on the principles of multi-tenancy is the appropriate architecture for configuration management system for globally distributed software development teams.

General Terms
Global Software Engineering, Configuration Management.

Keywords
Global Software Development, Configuration Management System, Software Architecture

1. INTRODUCTION
Global software development is a process of software development through globally distributed software development teams. This phenomenon was initially observed in the start of 1990, now numerous organizations have adopted it (Sahay, 2003) [1]. The teams involved in this process are dispersed across the globe. A simple scenario of team dispersion may include the possibility of teams located in USA, Europe and in Asia. Several structures were formed to exist for such teams. It could include sub-teams working on different modules and components of project, or teams based on functional roles such as designers, programmers, business analyst and quality control teams (Carmel, 1999) [2]. Global software development environment in which teams are distributed on the basis of functional roles may be shown in Figure 1. In such environment; software Team Lead may work in USA, Software Architect may work in Pakistan, while Software Developers could work in China. All such teams have different functional role and are connected through virtual private network (VPN) to achieve the common goals of software project development.

Researchers of global software development posited several reasons of adoption of global software development. Corbett (2012) [3] believed that reduced cost of software development is the main reason of dispersion of software development teams across the globe. Cheng and Atlee (2007) [4] believed that round the clock development, need of highly skilled resources and being geographically closer to the customer are main reasons of adoption of global software development. Herbsleb (2007) [5] believed that certain investment requirements, merger of large organizations and need to explore new market opportunities are main motivation factors behind adoption of global software development.

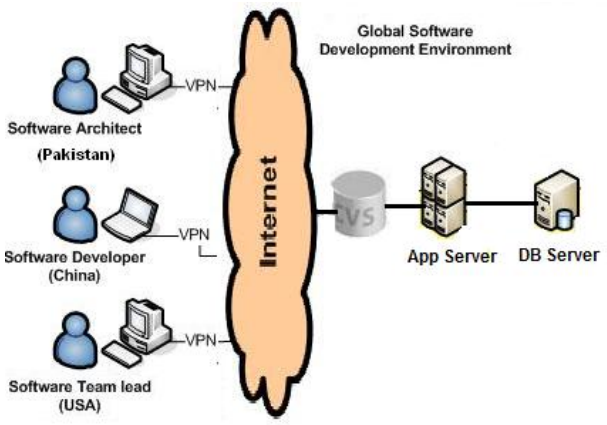

Figure 1, Globally distributed teams on the basis of functional roles

Despite the reduction of cost and other benefits of globally distributed teams, globalization brings several challenges for practitioners of global software development. Communication, coordination, trust and weak management of distributed teams were found as core challenges of globalization (Sabahat et al., 2010; Yousaf et al., 2008) [6,7]. However, besides these management and process related challenges, configuration management of globally distributed teams was reported as a major technical challenge that was faced by software engineers of globally distributed teams (Pilatti et al., 2006; Dwivedi, 2013, Komi-Sirvio & Tihinen, 2005) [8-10]. The inappropriate identification and management of configuration items (produced by distributed teams) leads to several inconsistencies and delays in overall completion of work products. Therefore, it is important to investigate the possibilities to minimize the risk of inappropriate handling of remote configuration items during development of software through globally distributed teams. Hence this study is designed to investigate the answer of following research questions;

Research Question #1: What should be the architecture of a configuration management system for globally distributed software development teams?

2. CONFIGURATION MANAGEMENT SYSTEM

2.1 Software Configuratoin Management

Software configuration management involves the development of software work products under controlled environment, where every change is traceable to its source (Berczuk & Appleton, 2002) [11]. It helps to manage the development of large software systems. Fujieda and Ochimizu (2003) [12] posited following main activities of configuration management process;

i. Establishment of configuration management system
ii. Identification of configuration items
iii. Management of configuration items under version controlled repository
iv. Management of access rights of users
v. Status tracking of various configuration items

Conradi and Westfechtel (1998) [13] believed that a configuration management system comprises of configuration items, users of repository and access rights details of repository users. According to Scott and Nisse (2001) [14], a configuration management system consists of following components;

i. Management of configuration process
ii. Identification of configuration items
iii. Software configuration control
iv. Software configuration status accounting
v. Software configuration auditing
vi. Software release management and delivery

Several techniques (e.g. branching, merging etc.) are used to manage the work products of a software project on configuration management repository. Configuration items consist upon deliverables and non-deliverables of a project. The deliverable work products are important configuration items; hence, those require extra considerations during their development. The delay in completion or any conflict in such work products leads to more risky state of project. Scott and Nisse (2001) [14] believed that all items, on which multiple practitioners are expected to work on, should be maintained under configuration control to manage the changes in controlled manners. Software project developed by collocated teams usually consists upon single configuration management server. Multiple, collocated members of software development team interact with configuration management server to accomplish their day to day tasks. Access rights are implemented on repositories of configuration management system by providing certain rights to users of configuration management system. These rights are managed by configuration engineer or configuration administrator. A simplified example of a configuration management system is shown in Figure 2.

According to Conradi and Westfechtel (1998) [13], a configuration management system serves several needs, including the support for management and development related activities. The version control mechanism of configuration management system helps the practitioners to manage and control the changes of management related work products and development related source code of projects.

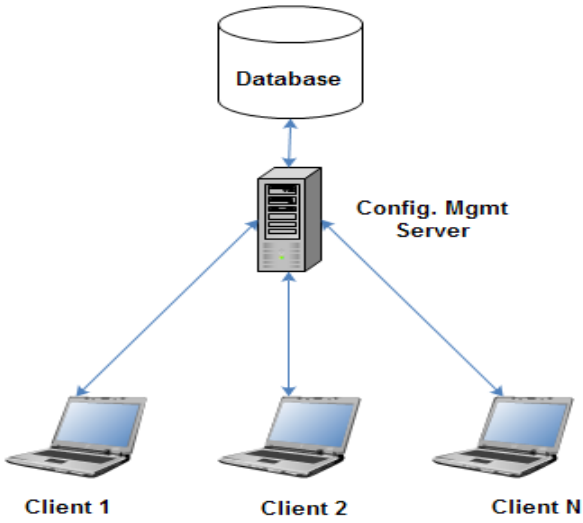

Figure 2, A typical model of a configuration management system

2.2 Configuration Management System for Globally Distributed Teams

Globally distributed teams require different processes of software development and management as compared to those of collocated teams (Carmel, 1999) [2]. Scott and Nisse (2001) [14] posited that before planning about configuration management system, there is need to understand the organizational structure and formation of software development teams. Because the configuration needs of different team structures varies from each other. These findings lead to following research question;

Research Question # 2: What type of configuration management system should be established for globally distributed software development teams?

There can be two possibilities for different types of a configuration management system;

i. Centralized configuration management system
ii. Distributed configuration management system

In global software development environment, there are several pros and cons of both types of systems. For example, it is difficult to distinguish the team specific work products in a centralized configuration management system. While integration of parts of work products becomes very difficult, when work products are integrated from distributed configuration management systems. However, existing literature helps us to investigate the answer of this research question. Pilatti et al. (2006) [8] believed that there should be single instance of a configuration management system for globally distributed teams of a software project. This model helps us to minimize the risk related to the loss of contents at the time of integration of different parts of a work product. Single and centralized repository makes sure that whole data related to configuration items is stored and maintained under single repository. But, this finding brings challenges of distinguishing of different distributed teams and their work products under single repository. This problem can be

addressed by investigating the answer of first research question of this study. Remaining section of this article is based upon the findings for architecture of a configuration management system for globally distributed software development teams.

2.3 Existing Techniques of Configuration Management System

2.3.1 Odyssey-VCS

Oliveira, Murta and Werner (2005) [15] proposed a version management system (with the name of Odyssey-VCS) for UML diagrams. They proposed that Odyssey-VCS stores complex data elements of UML diagrams and provides capabilities of versions and comparisons of UML work products. The complex data elements include actors, classes, methods, attributes, relations and different elements of UML components. Versions are managed at two levels;

(1) Version management at element level

(2) Version management at component (composition of elements) level

Provision of comparison is also available at both levels. Odyssey-VCS provides the capability of concurrent access of same component by several users. It also supports the merge facility at commit operation. The overall architecture of this system is based upon three layers of code;

(1) Client layer

(2) Transport layer

(3) Server layer.

2.3.2 Model Data Management

In Model Data Management solution, El-khoury (2005) [16] integrated the software configuration management and product development management together. In this system, work products related to software development and mock-ups of hardware can be configured into a single repository. This proposed model is very helpful for the configuration of complete system (infrastructure including hardware and software).

2.3.3 Two-way Merge Algorithm

Hayase, Matsushita and Inoue (2005) [17] proposed a two way merge algorithm to facilitate the merging of source code, when concurrent changes on same line of code will be committed by multiple programmers. They transform the source code into intermediate XML form and then merging is performed at XML level.

2.4 Proposed Architecture for Configuration Management System

In global software development, different teams work from geographically different locations. Therefore, based upon distribution of teams, configuration items can be segregated into two categories;

(1) Common configuration items for all teams

(2) Team specific configuration items

This segregation of items can be a guide for architecture of a configuration management system for globally distributed teams. Therefore, in this study, it has been proposed that, in order to develop a centralized system of configuration management for globally distributed teams, there is need to design the data architecture, which should help to separate the different information of different teams. Common configuration items should be stored separately from team specific configuration items. Technically, this separation can be achieved in many ways.

2.4.1 Separate Databases

Separate Databases can be designed for each team to maintain information of their configuration items separately. In this case, a common database will also be required to maintain information about common configuration items of distributed teams. This model is simple to design, but difficult to maintain. The number of complexities of management of such system increases with increased number of distributed teams. This approach is also expensive in terms of cost of space and resources required to manage such system.

2.4.2 Single Database (Proposed Architecture)

To answer the first research question of this study, it has been proposed that single database should be designed to maintain the configuration items of different teams and common configuration items of all teams. This model can be achieved by using the multi-tenancy architecture (Azeez et al. 2010) [18] as a reference for database design. Stating it more precisely, it can be posited that the proposed architecture is similar to the "shared database, shared schema" approach of multi-tenancy architecture. The proposed design is presented in Figure 3.

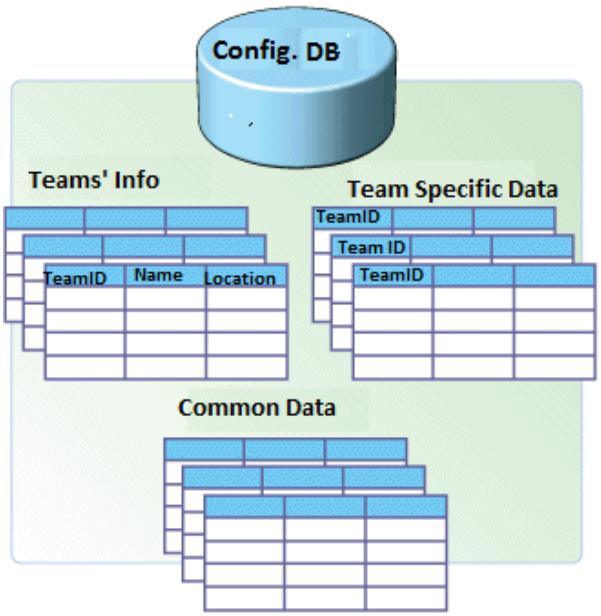

Figure 3, Proposed architecture of configuration management system for globally distributed team

In this architecture, there will be singe database that will maintain three types of tables;

2.4.2.1 Tables to maintain information about distributed teams

There will be some tables to maintain Meta data of distributed teams. These tables will maintain information about identification of teams, locations of teams, culture and functional roles of teams.

2.4.2.2 Tables to maintain team specific data

There will be some tables to maintain team specific data and team specific configuration items. Every table in this classification will maintain a reference of TeamID as a key

column. This design will help to maintain data about configuration items of all teams under single database.

2.4.2.3 Tables to maintain common data

There will be some tables to maintain common data of configuration items for all distributed teams. These tables will not contain any reference of teams as key column. Data and configuration items stored in these tables will be common across all distributed teams of a software project.

2.4.3 Configuration Model for Globally Distributed Teams

Figure 4 shows the pictorial representation of a configuration management system for globally distributed teams. Based upon the proposed architecture of this study, a single configuration management system can be established to serve the configuration needs of all distributed locations. One administrator can manage whole system from single location. This model presents simplicity for practitioners of globally distributed teams, where software engineers from all locations interact with repository in consistent ways. Configuration engineer also performs configuration related activities on single server at centralized location.

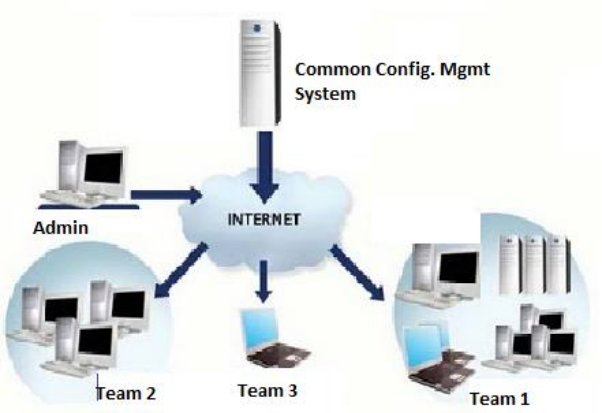

Figure 4, A model of configuration management system for globally distributed teams

2.5 Comparison with Existing Techniques

Odyssey-VCS [15] is a version control CASE tool that helps to manage UML diagrams under configuration repository. But, in SDLC, there exists several configuration items those should be stored under the umbrella of configuration management for better control and availability to all stakeholders. Therefore, being limited for UML diagrams only, Odyssey-VCS doesn't help to manage all work products of a software project developed by distributed teams or by collocated teams.

Model Data Management [16] is not designed to ensure the distinct configuration of work products of distributed teams. It also doesn't support concurrent access to the individual work products.

Two-way Merge algorithm [17] is developed for Java programming language only. It is not developed for C#, PHP and for other programming languages. It is also not workable for work products other than java code.

The architecture proposed in this paper doesn't have any of the limitations discussion in this section. Concurrent users can access single work product, they can perform changes and can commit their work back to repository. Database will ensure the atomic commit of each user. All types of work products will be handled by the proposed architecture. Single repository will be capable to store all work products of all distributed teams of a software project.

3. Conclusion

In this study, the first research question was about the investigation of architecture of configuration management system for globally distributed software development teams. Second question was about the type of configuration management system for such teams. The answer of second research question was investigated first and it was proposed that the centralized configuration management system was more appropriate configuration management system for globally distributed software development teams. To answer the first research question, the concept of multi-tenancy was linked with multiple teams of global software development and architecture of configuration management system was proposed that was similar in nature with multi-tenant based database management system. The proposed architecture was capable to store the data of all distributed software development teams under the umbrella of single configuration repository.

4. REFERENCES

[1] Sahay, S. 2003, "Global software alliances: the challenge of 'Standardization'", Scandinavian Journal of Information Systems, 15, pp. 3–21.

[2] Carmel, E. 1999, "Global Software Teams: Collaborating Across Borders and Time Zones". USA, Prentice Hall, 1999.

[3] Corbett, M., "The Strategic Outsourcing Study". http://www.corbettassociates.com [17 March 2013].

[4] Cheng, B.H.C. and Atlee, J.M. 2007, "Research Directions in Requirements Engineering". Future of Software Engineering (FOSE 07), IEEE, 2007.

[5] Herbsleb, J.D. 2007, "Global Software Engineering: The Future of Socio-technical Coordination". Future of Software Engineering, IEEE-CS Press, 2007

[6] Sabahat, N., Iqbal, F., Azam, F. and Javed, M.Y. 2010, "An Iterative Approach for Global Requirements Elicitation: A Case Study Analysis". International Conference on Electronics and Information Engineering (ICEIE 2010).

[7] Yousaf, F., Zaman, Z. and Ikram, N. 2008, "Requirements Validation Techniques in GSD: A Survey". IEEE, 2008.

[8] Pilatti, L., Audy, J.L.N. and Prikladnicki, R. 2006, "Software Configuration Management over a Global Software Development Environment: Lessons Learned from a Case Study"

[9] Dwivedi, R. 2013, "Configuration Issues and Efforts for Configuring Agile Approaches-Situational based Method Engineering", International Journal of Computer Applications, 61(17): 23-27.

[10] Komi-Sirvio, S. and Tihinen, M. 2005, "Lessons Learned by Participants of Distributed Software Development", Knowledge and Process Management, 12(2): 108-122.

[11] Berczuk, S. P., and Appleton, B. 2002. "Software Configuration Management Patterns: Effective Teamwork, Practical Integration", Addison Wesley.

[12] Fujieda, K., and Ochimizu, K. 2003. "Investigation of Repository Reprecation Models in Globally Distributed

Configuration Management". In Proc. of the Workshop on Global Software Development at ICSE.

[13] Conradi, R. and Westfechtel, B. 1998, "Version Models for Software Configuration Management", ACM Computing Surveys, 30 (2), 233-282.

[14] Scott, J.A. and Nisse, D. 2001, "Software Configuration Management", IEEE - Trial Version 1.0

[15] Oliveira, H., Murta, L., and Werner, C. (2005). Odyssey-VCS: a Flexible Version Control System for UML Model Elements. In Proceedings of the 12th International Workshop on Software Configuration Management (SCM 2005), Lisbon, Portugal.

[16] El-khoury, J. (2005). Model Data Management – Towards a common solution for PDM/SCM systems. In Proceedings of the 12th International Workshop on Software Configuration Management (SCM 2005), Lisbon, Portugal.

[17] Hayase, Y., Matsushita, M. and Inoue, K. (2005). Revision Control System Using Delta Script of Syntax Tree. In Proceedings of the 12th International Workshop on Software Configuration Management (SCM 2005), Lisbon, Portugal.

[18] Azeez, A., Perera, S., Gamage, D., Linton, R., Siriwardana, P., Leelaratne, D., Weerawarana, S. and Fremantle, P. 2010 "Multi-Tenant SOA Middleware for Cloud Computing". 2010 IEEE 3rd International Conference on Cloud Computing