The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.70 e-ISSN: 2251-7545

An Approach for Configuration management in Ultra Large Scale systems

Mohammad Ali Torkamani
R&D Department
Iranian Telecommunication Manufacturing Company
Shiraz, Iran
torkamani_ali@yahoo.com

Seyyed Hossein Ahmadi
R&D Department
Iranian Telecommunication Manufacturing Company
Shiraz, Iran
Sha13267@yahoo.com

Ali Bayat
R&D Department
Iranian Telecommunication Manufacturing Company
Shiraz, Iran
alibayat23@gmail.com

Abbas Bahrami
R&D Department
Master Student of Information Technology, Shiraz University,
Shiraz, Iran
Abbas_bh1@yahoo.com

Hamid Bagheri
Information Technology
Kurdistan University
Sanandaj, Iran
bagheri.hamid@gmail.com

Mohammad Reza Khodabakhshi
R&D Department
Iranian Telecommunication Manufacturing Company
Shiraz, Iran
khodabakhshi.2008@gmail.com

Abstract— Ultra Large scale systems have some characteristics which are derived from their scale. These characteristics of ULS systems make it impossible to rely on our current approaches in software development and new approaches for development, deployment, control and management should be made. Today's central developing approach in software systems to tackling ULS challenges will not suffice. One of the most important phases in developing information systems is configuration management. In ULS Configuration management is much more complicated than current practices. The available tools which are developed for configuration management so far are operating centrally while different developers participate in developing ULS systems independently. Due to the existing dependencies between components of such systems and their characteristics, change management needs new approaches. Change in one component may result in some side effects in other relating components. In some approaches, for instance, changing in one component which is used by many developers, while they are unaware, may influence their systems. To tackle such problems, in our paper we propose and analyze new approach for configuration management. This method is implemented in R&D department of Iranian Telecommunication Manufacturing Company (ITMC).

Keywords-component; Ultra Large Scale system(ULS); Configuration Management; Component (key words)

I. INTRODUCTION

Nowadays several team and organization developers are participating in developing large scale systems. These teams may locate in different part of the globe, have different time working and may have different culture and formal languages. Each team is responsible for developing one or more sub-systems [1]. Some of these developers may develop their sub-system through the assembly of available components, developments of available codes and modules or the purchase of commercial of the Shelf (COTS) components, or outsourcing of some parts of work. One of the current problems in such large systems is applying new change and Software Configuration Management (SCM). In other words due to the dependency between components the change management is very important [2]. How to apply these changes so that all teams are informed about updates as quickly as possible? How changes should be executed in order to have minimum re working? Does every change should be done without informing other developers? In fact some sequences of changes should be managed, in other words the dependency chain in developing system should be managed and taken into account [3]. Because when a system is made up of assembled components, an error in system configuration may result in software miss-functioning.

In Ultra Large Scale systems, the problem of configuration management is more serious than traditional systems. The requirements of ULS system have basically conflict with each other and are unknown. Some requirements of ULS systems may be unknown until the time which they are used. The Components of ULS systems for example computes, users, software are basically inconsistent, heterogeneous and constantly changing. Components of software due to the different reasons such as different platforms, methodology and programming languages are heterogeneous and inconsistent. Services and components are inconsistent because they are developed by different developers. As a result the consequences of some changes are unpredictable. Scale is a main reason of such consequences [4]. Beside that components and users of such large systems are heterogeneous. It means they have different requirements and different potentials.

So for developing Ultra Large Scale systems we need tools which support different sites and contain different

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.70 e-ISSN: 2251-7545

repository [5]. Most of the available tools use single shared repository and in fact they use centralized approach [6]. Furthermore due to the variety of developers and heterogeneous and developing approaches in ULS systems, the tools which have undependable platforms are applicable. Nowadays most available tools for management configuration are platform dependent. Indeed in ULS systems there is no single developer for developing, maintaining and evolution of systems. The centralized coordinator and controller of traditional systems in available models are not applicable for ULS systems. In fact in ULS systems there are decentralized and localized control. For example in Internet nobody is responsible for developing, running and evaluation but different organizations have different responsibility toward the Internet [4]. Centralized approaches can't cope with Ultra Large Scale systems [2, 7]. In other words decentralized approaches are more appropriate than centralized approaches [2, 7]. The tools which are developed for configuration management so far are centralized. In some approaches developers must use the last version of component, which is developed by another developer, without delivering any idea about it while these changes may affect their functionality. For example in StarTeam, the component server is installed on server and developer by their clients through user name and password can use the server. In this approach there is a change management which approved file is available and it is used by developers.

In the rest of the paper, to tackle such problems the decentralized approach for configuration management will be introduced. In the second section of the paper the proposed model for configuration management will be introduced and then in third section evaluation of the proposed model and case study will be discussed. Finally we have a conclusion section.

II. DECENTRALIZED APPROACH FOR ULS SYSTEMS

In configuration management process which is introduced in [8, 5] by pressman and Somerville, customer sends change request; developer verifies it and makes changes if applicable. But ULS systems and some situations where we encounter different developers not addressed.

Suppose we want to develop a ULS system which different developers are cooperating in developing the system. Each developer is completely independent and each team has its own manager.

Each developer have their own repository which holds their components. Each developer may use some components which are made by other developers. Owner of the software is the customer but he/she is not necessarily the owner of the all classes and components. Developers are owner of components; of course this issue is depending on the contract. In current configuration management software and approaches if one developer wants to change his component, the other developers should use the last version of component and viewpoint of other developers are not considered whereas these changes due to dependency between components may have influences on other developers' job.

An idea used in our approach is very simple. Developers of ULS systems deliver their idea about the component which they have used. In fact a polling system is used. The algorithm of this approach is shown in Figure 1.

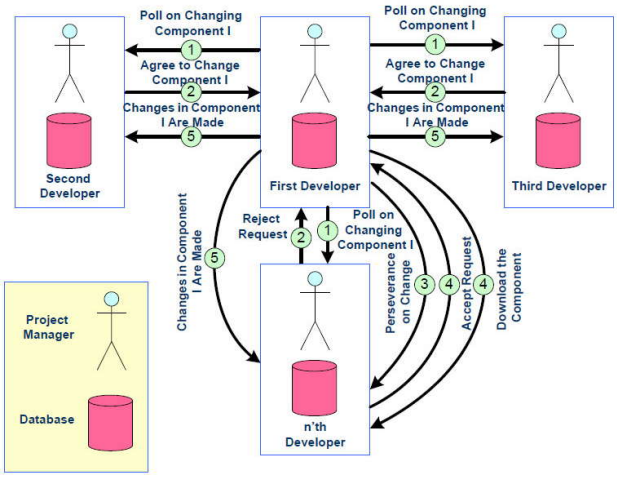

Figure I. proposed approach

Suppose a developer makes a change in his component. In proposed approach there is relationship between developers. When a developer wants to make a change in his component, he sends a message to other developers who are using his component. Each developer sends a vote (reject or accept) to the requester. In this approach each developer should be aware of other developer's responsibilities. In fact, every developer should have a table in his database which contains fields such as component name and developer's ID. This table should be updated constantly. If all developers approve the change, the requester can update the component and repository. After that the requester sends a message to show that the changes are made. Developers who need a component can download new version through the network. If a developer disagrees with the changes, gives a requester a negative vote with the reason. If this reason can convince the requester, the changes may be canceled. If the agreement does not reach, the software owner will make a decision about applying changes. In proposed model analyzing of dependencies is done immediately after each change request and before enforcing the changes. All organizations, teams and developers are taking part in applying these changes. So if one component changes the risk of consequences of these changes in other components will be minimized.

III. EVALUATION AND CASE STUDY FOR PROPOSED MODEL

The proposed approach is implemented by Visual Studio .Net in R&D department of Iranian Telecommunication Manufacturing Company (ITMC). The implemented software is able to demonstrate the dependency graph as well

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.70 e-ISSN: 2251-7545

as management and transfer the change between different developers across the globe. The software is currently passing the final test and is used for developing large scale systems in R&D department in ITMC. In table I the proposed approach is compared with some available approaches in this area.

TABLE I. COMPARE PROPOSED APPROACH WITH AVAILABLE APPROACH

No	COMPARING	Star Team	Clear Quest	CCC Harvest	Proposed approach
1	Database	Centralized	Centralized	Centralized	Distributed
2	Graphically display the dependency graph	NO	NO	NO	YES
3	User interface support Farsi	NO	NO	NO	YES
4	All prices are for the supply of IRAN	NO MORE EXPENSIVE	NO MORE EXPENSIVE	NO MORE EXPENSIVE	Appropriate
5	Supports variety of development tools	YES	YES	YES	YES
6	CMS website	NO	NO	NO	NO
7	Performance and execution speed	HIGH	HIGH	HIGH	GOOD
8	Support RUP	NO	YES	NO	YES
9	Support development of SOA approach	YES	YES	NO	YES
10	Support manages dashboards of development team and sending email	NO	NO	NO	YES
11	Comment on the changes by all teams	NO	NO	NO	YES
12	Show history of components graphically	NO	NO	NO	YES
13	Web Usability	YES	YES	YES	YES
14	Platform	Win,Solaris, Linux	Win, Solaris, Linux	Win, Solaris, Linux	Win

As clearly implies from first row of the table, proposed model as opposed to all available approaches has distributed database and this approach is applicable for ULS systems. Also as compared to other approaches, implemented software can support graphical reports as well as proper final cost. As clearly shown by row 11 in table 1, in proposed model all developing teams send request change to the manager of team developer. This is the main difference between proposed model and current models.

Table II shows the approach evaluation which resulted by experts and software developers who have at least 15 years experience. This evaluation covers 10 aspects. The maximum number is 4 for each aspect. The average number of evaluation is registered in the table. The total average number for evaluation is 3.3 out of 4.

TABLE II: EVALUATION BY EXPERTS

No	Aspects of	Average number
1	Methods and Algorithms	3
2	Safe and convenient access	4
3	User interface	3
4	Speed and efficiency	3
5	Appropriate information systems major	4
6	The system suitable for large scale	3
7	Take advantage of the various methodologies	4
8	Version Management	3
9	Last Cost	4
10	Can be used as an educational tool	2
11	Average	3.3

IV. CONCLUSION

Nowadays different developers worldwide are participating in developing ULS systems. Due to the existing dependencies between components of such systems and their characteristics, change management needs new approaches. Change in one component may result in some side effects in other related components. In some approaches, for instance, changing in one component which is used by many developers, while they are unaware, may affect their systems.

Available central developing approach in software systems to tackling ULS challenges will not suffice.

In ULS Configuration management is much more complicated than current practices. The available tools which are developed for configuration management so far are operating centrally while different developers participate in developing ULS systems independently.

In proposed model we introduce decentralized approach for configuration management in ULS systems.

In proposed model, analysis of dependencies is executed immediately after each change request and before final confirmation. All organizations, teams, developers and stakeholders are participating in changes. This approach for ULS systems which have different and unknown characteristics is more applicable. As a result of this

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.70 e-ISSN: 2251-7545

approach, the consequences of changes and side effects will be the minimum. The benefit of proposed approach derives from prevention action. It means before any changes could take place different developers can reach agreement and proper coordination will be done. This approach is compatible with characteristics of ULS systems. As mentioned before, proposed approach is implemented by Visual Studio .Net in R&D department of Iranian Telecommunication Manufacturing Company (ITMC). The implemented software is able to demonstrate the dependency graph as well as management and transfer the change between different developers across the globe. Proposed model as opposed to all available approaches has distributed database and this approach is applicable for ULS systems. The total average number of evaluation, which was contain 10 different aspects, was 3.3 out of 4 which simply shows the moderate success of the proposed approach.

V. FUTURE WORK

It can be clearly seen in table I that proposed model just supports Windows, but in ULS systems developer may use different platforms. In other words there are heterogeneous operating systems and softwares, so we are planning to extend the implemented model to support different platforms.

REFERENCES

[1] B.Al-Ani, H.Edwards, "a Comparative Empirical Study of Communication in Distributed and Collocated Development Teams", IEEE, 2008.
[2] SEI, Ultra-Large-Scale Systems, the Software Challenge of the Future, Carnegie Mellon University, 2006.
[3] Ivins, Wendy, "Managing Flow Dependencies: The Missing Link in Co-coordinating Distributed Teams in Large-Scale Development Projects", IEEE, 2008.
[4] S.Ostadzadeh, F.Shams Aliee, "The Role of MDA in ULS Integration: Challenges and solutions", 5th International Conference on Information Technology Managenent(ICTM), 2008.
[5] I. Sommerville, Software Engineering, 9th Edition, 2010.P.Cravino, D.Lawrence, A.López, O.Alonso, S.Brandt, Enterprise Software Configuration Management Solutions for Distributed and System z, IBM, 2009.
[6] M.Torkamani, S. Bashavard and N. khalili Safa, "ULS Configuration Management challenges", 2nd International Conference on Contemporary Issues in Computer and Information Science(CICIC 11), Iran, Zanjan, 2011.
[7] R.S.Pressman, Software Engineering: A Practitioner's Approach, 7th International edition, 2009.