Key Requirements for Component Configuration Management (CCM)

Imran Ali Qureshi, Asif Iqbal Paracha, Saqib Afzal, and Shahzad Rafiq

***Abstract*—Component Configuration Management became very important and significant in Software Configuration Management discipline. By the passage of time we have seen that a lot of work was done for the improvement. During the Component Configuration Management there are certain problems faced by stakeholder such as version management, conflict of new and old components, change impact analysis, historical view and ripple impact identification**

In this paper our main focus remains on identification of requirements for Components Configuration Management, we thoroughly reviewed the literature and after analyzing the problem, we have identified and defined three key requirements.

***Index Terms*—Component, configuration, management.**

I. Introduction

Configuration Management is not only critical and important step in the engineering process of a system; it has same importance as well in the life of a system. The ultimate objective of configuration management (CM) is to establish and maintain integrity.[1]

To ensure integrity, components should be properly defined and documented; components should have enough information that not only integrate it with other components, but can also provide traceability, which can help us to identify how components are functioning in a system.[1]

According to the Pressman, Software Configuration Management (SCM) is "Set of activities designed to control change by identifying the work products that are likely to change, establishing relationships among them, defining mechanisms for managing different versions of these work products, controlling the changes imposed, and auditing and reporting on the changes made"[2].

Main discipline of SCM are Version Management, Configuration Management and Change Management.[2]

The discipline of configuration management (CM) is concern with the management of artifacts, software which are integrated in a software system. Artifacts are evaluated and then suggested changed configurations are either incorporated or discarded.[2]

Mainly SCM is used to eradicate the puzzlement and uncertainty among different components/ versions of artifacts / software. In real life it is possible that in a project when different people are working together and creating different versions of the same artifact then at the time of configuration we may have wrong version of an artifact/ component. So to avoid such mistakes there should be a proper mechanism to define a component.

There are several tools/ techniques available for SCM during Software Development Life Cycle (SDLC), but a little work is performed actually in the area of Component Configuration Management (CCM).

The objective of the said paper is to highlight the significance and importance of Requirement Engineering (RE) for CCM as well as the dimensions for RE that are concern with CCM

Rest of the paper consists of following Sections. Section 2, is about literature review, Section 3 highlights and defines requirements for CCM, and Section 4 describes conclusion and future work.

II. Literature Review

After thoroughly reviewing the literature it was found that although the importance and significance of the CCM was realized and different authors have highlighted different issues which should be cater as a part of CCM but no one has provided any fundamental methodology to cater the highlighted problems.

Pilatti L. *et al.* [3] addressed the different challenges and problems of SCM in distributed environment. Authors recommended that inter-dependency between globally distributed teams should be minimum. To overcome synchronization and change management issues, all teams must define the main concepts with clarity and detail, and placed all the configuration items on a centralized & single instanced SCM environment. All SCM activities must be well planned, documented and prioritized as project evolved. Their work showed that the configuration of items is an important issue [3].

Bendix L. *et. al.* [4] explained and elaborated the vital role and importance of SCM activity during requirement engineering and suggested a return of investment (ROI) model and metrics. Authors summarized the key SCM activities as Configuration Identification to record and identify configuration items, Configuration control: for changes traceability and management, Accounting and Audit for reporting and verification. It is recommended by the authors that SCM method and principles must be applied during all phases, however, no explicit formula or metrics has been given for ROI calculation [4].

Volzer H. *et al.* [5] introduced a hierarchical Model for software configuration and change management on software development artifacts and built a prototype called subCM.

Manuscript received August 6, 2012; revised October 7, 2012.

The authors are with the Mohammad Ali Jinnah University, (MAJU), Islamabad, Pakistan (e-mail: i4imran@mail.com, asifparacha786@gmail.com, saqibafzal@gmail.com, shahzad@jinnah.edu.pk).

The method is capable to define and store subsystem configuration specification (SCS) or a short textual summary e.g. what artifacts were produced in what subsystem, the change description by describing change item, its type, baseline reference and version comparison by comparing two/more parallel or adjacent versions. A case study of interfacing subCM prototype with telelogic and synergy toolset CM, its results and experienced were also discussed [5].

Carnduff T. W. *et al.* [6] proposed a model by the name of Configuration Management Model through which configuration changes are cater as versions. The proposed system facilitates to manage and overcome the difficulties faced during the configuration [6], but still no concrete measurement are proposed for component configuration management.

Mao M. and Jiang Y. [7] proposed the 3C Model which is an outcome of the research of CMMI and IEEE criterion. 3C Model is actually based on component-based and layered architecture. Authors' proposed model is divided into three layers, at the lowest level is Control layer, then Configuration layer and on the highest level is Component layer that is the reason it was named as 3C Model. Authors claimed that among the available other configuration models their proposed model is most suitable for component-based configuration because their model is structurally layered and follows component-based CM scheme while other uses file-based CM scheme. This paper is about 3C Model and its realization, although the authors have very nicely elaborated and described their proposed model but they have not provided any pre-requisites for components rather this paper aims to provide a good framework but it is revealed that component configuration management is very important and needs special attention at every level.

Estublier J. *et al.* [9] highlighted the significance of SCM and point out different issues and treated them as challenges for SCM, such as software versioning, configuration control, consistency, data modeling, versioning in component repository, etc.. authors has just given a road map for the better SCM by identifying certain challenges that should be overcome [9].

Larsson M. *et al.* [10], [11], [12], [14] and Crnkovic I. *et al.* [13] have highlighted the importance of CCM and point out it as a new challenge. Authors' contribution is significant in this manner that first time they suggested that at run time there should be component configuration and also said that this should be treated as a new discipline in SCM [8], [10], [14]. The major issue which was highlighted by the authors is managing component dependencies [12].

Crnkovic I. and Larsson M. have done a lot in identifying the new challenges and issues which are specific to the CCM [8], [12], [14], [15].

We have identified three key requirements that are necessary for CCM, which we have explained in Section 3.

III. Requirements for CCM

CCM's objective is to curtail the problems such as: version management, conflict of new and old components, change impact analysis, historical view and ripple impact identification, which are related between components and the rest of the system. This objective is very well highlighted by Larsson M. and Crnkovic I. [12] as shown in Fig. 1.

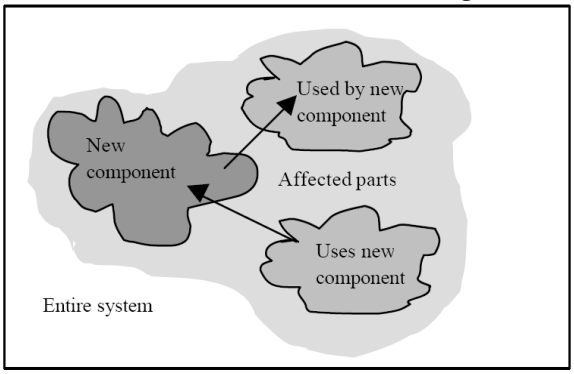

Fig. 1. Components relationship [9]

For CCM first of all we should know what are the necessary requirements for the CCM, for this purpose after reviewing the existing literature as mentioned in Section 2, and prevailing practices being adopted by the developers and designers we have identified, three key Functional Requirements (FR), as shown in Fig. 2. These FR act as catalyst for the CCM.

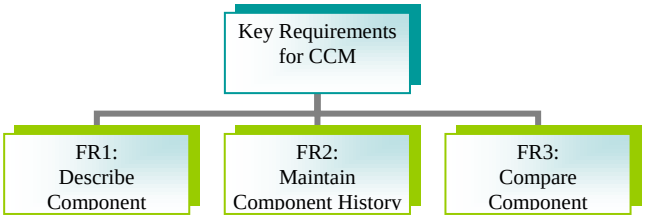

Fig. 2. Key requirements of CCM.

A. FR1: Describe Component

There should be a systematic and organize way to describe a component. A lot of method for describing anything can be adopted. The simplest way could be xml but one can describe according to their suited environment.

The description will contain the complete information about a component like, the type of component, component interfaces, input/ output recursively, component environment, etc.. It will be editable and can be written at any time in any phase like at the time of designing phase, developing phase or even at the time of just before component configuration.

B. FR2: Maintain Component History

This requirement is directly concern with versioning of a component. It will maintain the history of a component on the basis of its versions. The history of a component will clearly identify the change in a component since its first version.

Component history will highlight every change either it is a minor or it is a major.

Component history will identify elements like component version number, author of a component, the description of a specific version of a component, etc..

C. FR3: Compare Component

This requirement will compare different versions of a component. The comparison of different versions of a component will highlight different aspects, as we shown with the help of Venn diagram in Fig. 3(a), Fig. 3(b) and Fig. 3(c).

A = earlier version of a component
$\acute{A}$ = modified version of a component

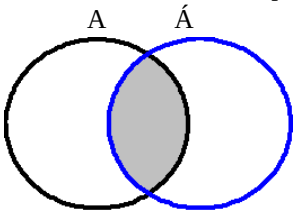

Fig. 3(a).

$A \cap \acute{A}$ = Common features which are available in both the earlier version and in the modified version of a component.

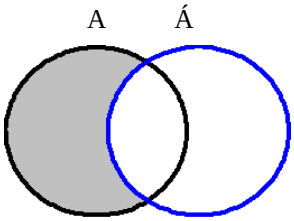

Fig. 3(b).

$A - \acute{A}$ = Features which were available in earlier version but not-available in the modified version of a component.

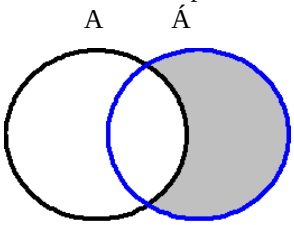

Fig. 3(c).

$\acute{A} - A$ = Features which are available in modified version but not-available in the earlier version of a component.

IV. Conclusion and Future Work

We have thoroughly reviewed the literature and after analyzing the problem, we have identified and defined three key requirements for component configuration management, i.e., component description, maintain history, and ability to compare two components.

The role of a unified system for component configuration management is vital and our contribution, to identify the key requirements, is significant for such system.

This System is useful for software designers, developers, configuration managers and third party component providers.

In future work we shall develop a prototype to implement our proposed work.

References

[1] *Freeway management and operations handbook*, FHWA Report No.: FHWA-OP-04-003, EDL No.: 13875, Final Report, September 2003

[2] M. Larsson and I. Crnkovic, "New challenges for configuration management," in *Proceedings of 9th Symposium on System Configuration Management*, *Lecture Notes in Computer Science*, Springer Verlag, no. 1675, 1999.

[3] L. Pilatti *et al.*, "Software configuration management over a global software development environment: Lessons learned from a case study," in *Proceedings of International Conference on Software Engineering (ICSE'06),* Shanghai, China, May 2006, pp. 45 - 50.

[4] L. Bendix and L. Borracci, "Towards a suite of software configuration management metrics," in *Proceedings of the 12th international workshop on Software configuration management*, 2005, pp. 75 - 82.

[5] H. Volzer, A. MacDonald *et al.*, "Sub CM: A tool for improved visibility of software change in an industrial setting," *Software Engineering, IEEE Transaction*, vol. 30, no. 10, pp. 675 - 693, October 2004.

[6] T. W. Carnduff and J. S. Goonetillake, "Configuration management in evolutionary engineering design using versioning and integrity constraints," *Advances in Engineering Software*, vol. 35, no. 3-4, pp. 161- 177, March-April 2004.

[7] M. Mao and Y. Jiang, "A New Component-Based Configuration Management 3C Model and its Realization," *ISISE, International Symposium on Information Science and Engineering*, vol. 1, pp. 258-262, 2008

[8] I. Crnkovic, "Component-based software engineering – new challenges in software development," in *Proceedings of the 25th International Conference on Information Technology Interfaces, (ITI 2003),* June 2003, pp. 9-18

[9] J. Estublier, "Software configuration management: A roadmap," in *Proceedings of 22nd International Conference on Software Engineering, The Future of Software Engineering*, ACM Press, 2000.

[10] M. Larsson, "Applying configuration management techniques to component-based systems," *Licentiate Thesis Dissertation 2000*, Department of Information Technology Uppsala University, vol. 7, 2000.

[11] M. Larsson and I. Crnkovic, "Development experiences of a component-based system," *7th IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2000).*

[12] M. Larsson and I. Crnkovic, "Component configuration management," in *Proceedings of ECOOP Conference, Workshop on Component Oriented Programming Nice*, France, June 2000.

[13] I. Crnkovicand and M.Larsson, "The different aspects of component based software engineering," in *Proceedings MIPRO (Microprocessor systems, Process control and Information Systems) Conference Opatija*, Croatia, May 2000.

[14] R. S. Pressman, *Software engineering: A practitioner's approac*h, 6th Edition, 2005.

[15] I. Crnkovic and M. Larsson, "Challenges of component-based development," *Journal of Software Systems (Elsevier Science Home)*, 2001.