Configuration Management: A Review

Ramandeep Kaur
Student, M.Tech.(CSE)
Department of Computer Science
Sai Institute of Engineering &Technology,
Amritsar, India

Aman Arora
Assistant Professor
Department of Computer Science
Sai Institute of Engineering &Technology,
Amritsar, India

Abstract—A software development process is concerned primarily with the production aspect especially the management of software development. The development of a software process passes through various phases and there is a need to manage all issues particularly configuration issues during the evolution of a software process. This paper makes an attempt to deal with various configuration issues like version management, change management, build management, release management and workspace management.

Keywords—Component, Configuration Management, Software Process, Version Management, Change Management

I. INTRODUCTION

Configuration management (CM) is the discipline of identifying the configuration of a system, at distinct points in time for the purpose of systematically controlling changes to the configuration, and maintaining the integrity and traceability of the configuration throughout the system life cycle. It is formally defined as “a discipline applying technical and administrative direction and surveillance to: identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change, processing and implementation status and verify compliance with specified requirements”. The activities associated with CM include configuration artifact identification, version management, change management, build management and release management [3][4][1]. Configuration management is concerned with the development of procedures and standard for managing an evolving software system product. In essence, it is concerned with how to control change, how to manage systems, which have been subject to change, and how to release the change systems to customers [3].

II. GOALS OF CONFIGURATION MANAGEMENT

The goals of Configuration Management (CM) can be summed as follows [2]:

- Identifying the items that are to be managed.
- Controlling changes in terms of the change process, and registering and reporting status.
- Controlling that items are complete and correct, involving registering and reporting items status, and controlling the distribution and use of changed item.
- Maintaining approved configurations of the items, implying the control of correctness, completeness and consistency.
- Controlling items to be included in a given product.

A Standard definition taken from IEEE standard 729-1983 highlights for classic operational aspects of CM:

- **Identification**: An identification scheme is needed to reflect the structure of the product. This involves identifying the structure and kinds of components, making them unique and accessible in some from by giving each component a name, version identification and configuration identification.
- **Control**: Controlling the release of a product and changes to it throughout the life cycle by having controls in place that insure consistent software via the creation of a baseline product.
- **Status Accounting**: Recording and reporting the status of components and change requests and gathering vital statistics about components in the product.
- **Audit Review**: Validating the completeness of a product and maintaining consistency among the components by ensuring that components are in an appropriate state throughout the entire project life cycle and that the product is a well defined collection of components.

III. COMPONENTS OF CONFIGURATION MANAGEMENT

This section describes about the different components of configuration Management as follows:-

A. Version Management

Version management involves drawing up an identification scheme for different version of a system, ensuring that the scheme is applied when new system version are created. A system version is an instance of a system, which differs in some way, from other instances [3]. Version management performs the identification of entities and recognizes different
versions of entities. An element of software or hardware placed under version control is designated a configuration item. Version management can be applied to the components at run time. Every component in the system should be identified by a name, version number and other version attributes such as creation date, historical information, etc. Version control covers the management of different versions of items, usually represented by a tree structure [4].

B. Change Management

Change management is a systematic approach to dealing with change, both from the perspective of an organization and on the individual level. A somewhat ambiguous term, change management has at least three different aspects, including: adapting to change, controlling change, and effecting change. A proactive approach to dealing with change is at the core of all three aspects. For an organization, change management means defining and implementing procedures and/or technologies to deal with changes in the business environment and to profit from changing opportunities. Change control procedures ensure that the changes to a system are made in controlled way so that their effect on the system can be predicted [3]. Change management becomes important when a new activity version is created. When a change is initiated, change request are created to track the change until it is resolved and closed. An example is discussed with the help of the figure. Fig.1 shows how change is organized in a system:

![Fig 1. Change Management](image)

A customer requests a change to correct an error in the system. The support organization receives the change request, taking direct action and solving the problem if possible. The configuration control board (CCB) analyses the change request and decides which action is to be taken. Revised software is re-validated to check that the changes has been properly made and has not adversely affected other parts of the system. The changed software is handed over to the CM team and is incorporated in a new version of the system.

C. Build Management

System building is the process of combining the components of a system into a program, which executes on a particular target configuration [3]. Build management supports the user by collecting source code for a particular release and then using build tools, such as Make, to create configurations [1]. Build management methods select and identify specific versions of entities and to integrate them into a new version of the composite entity. It also includes build procedures. The building procedures use information about the dependencies between the entities. If the system is built every day the integration time is reduced, since broken dependencies and faults are discovered early. The daily build process also permits rapid development and early testing of the system. Support for parallel development with version control, which resolves inconsistencies is required for daily build [1].

D. Release Management

A Software release is a version, which is distributed to customers. As an appropriate rule of thumb, each system release should offer significantly different functionality from the previous release. There may be multiple versions of the same release tailored to different configuration. Some versions may be created for internal development and may never be released. The identification and organization of are documents and supplements incorporated in a release is designated release management. It is possible with appropriate release management to create installation kits automatically to ease the task of the build manager. The build manager is responsible for providing the finished product with the correct configuration and features [1].

Release management is complicated by the fact that customers may not actually want a new release of the system. A system user may be happy with an existing system version and may consider the cost of changing to a new version unwarranted. Eventually, however as the system’s functionality is enhanced, it is likely that the customer will decide to change [3].

E. Workspace Management

Introducing CM in an organization is cumbersome without effective support from tools. Changing an existing culture requires massive education, support and not least motivation. To motivate developers to use all the tools and methods available with CM, support for integrated tools in the development environment is needed. Developers want to work independently of the configuration management, this alternative being denoted workspace management. Developers usually focus on solving particular problems and have less interest in administrative tasks. An example of integrated features is when the developer “logs-in” to a project environment in which project structures and data repositories are already prepared for the developer. The developer then enters a transparent environment in which the development with
configuration management is handled behind the scenes. This approach adopted in such major configuration management tools available on the market today as Clear Case [6] and Continuous [7].

IV. CONCLUSIONS AND FUTURE WORK

Component – based systems are becoming increasingly important in software process management. The continuous change in component-based systems, demand for an efficient configuration management. The different components of CM like version management, change management, build management, release management and workspace management are briefed. Various configuration control issues can be solved with the use of configuration management tools. Future work includes the design of an automated tool which helps in detecting the configuration issues during the installation of a new component.

REFERENCES