Homework #2 – Software Configuration Management
CSE 566: Software Project/ Process/ Quality Management
(2015 spring)

Submitted By:

Madhu Meghana Talasila

(1207740881)

Part - 1A

Lean Software Configuration Management: Applying lean principles on configuration management is called lean software configuration management. Lean CM is more adaptive to changes instead of predictive and support empirical nature of software development. Applying lean principles in configuration management i.e, test driven development, refactoring, pair programming and close customer collaboration.

- Eliminate waste: Removes Change Control Board meetings. Ownership for code is not given to single person but is in rotating basis.
- Build-In Quality: Automate the testing process and follow Test driven Development
- Create Knowledge: Allow frequent releases with efficient build process and integration between parallel activities.
- Other lean principles include defer commitment, do not commit until required strictly in irreversible commitments. Respect people, deliver fast and optimize the whole.

Lean Configuration management differs from traditional configuration management mainly because of economic perspective i.e., improving productivity. This can be done by "reducing software development effort, streamline development process and increase customer value" [1].

1. **"Reducing Software Development Effort":** This is done by reducing costs on investments for features that are not valuable to customer. This can be done by using empirical approach rather than doing complete analysis in the beginning like water fall method. For example: shortening feedback loops, manage requirements in prioritized product backlog [1]. It completely removes redundancy and complexity in all the development process by differentiating what is really required from gold-plated requirements. Lean focus on delivering what is asked with high quality [2].

2. **"Streamline Development Process":** Stream line the development process and use resources more efficiently. This can be done by using lean principles to reduce waste and use automated tools for design, development, testing and by automating the tasks [1]. In configuration management, control is illusion. Lean development process gives freedom to developers by managing short term feedbacks. Everyone in agile team is responsible to develop code, change and maintain consistent code throughout the development process rather than a single person as in agile teams [2].

3. **"Increase Customer Value":** Though Configuration management is not directly related to customers, it is always useful to involve customer into it by taking smaller feedback loops which increase the quality of production in shorter time frame. [1]

Lean configuration management also controls the flow in shorter intervals. That is either a sprint, release or a feature. When a particular sprint is completed, sign-off from the customer is taken as delayed decision might increase the complexity of the code and also might lead to rework. Studies proved that Lean configuration management improved the productivity.

References:

1. http://www.methodsandtools.com/archive/archive.php?id=62
2. http://www.cmcrossroads.com/article/agile-software-configuration-management-communications-and-documentation

<u>Part – 1B</u>

There are a number of branching and merging strategies. They are working on mainline, Branch for release and branch by feature [1]. How branch for release works with scrum. Branching and merging in scrum can be discussed based on the size of the team.

Branching and Merging in Smaller SCRUM Projects: For branching in scrum teams, process is divided into three branches: Main, Development and Release. These branches remain for all the sprints in SCRUM project. From the main branch a task is identified and a trunk is made to development branch. When the development work is done then the trunk is again merged in to the main branch. Thus Main branch and development branch are used to develop and stabilize the release. Now a version of the project is developed and then it is branched to release branch. By the end of first sprint, a version of software is released and development of new version starts for another release. This process continues till all the product backlog items have been delivered as working software. [2]

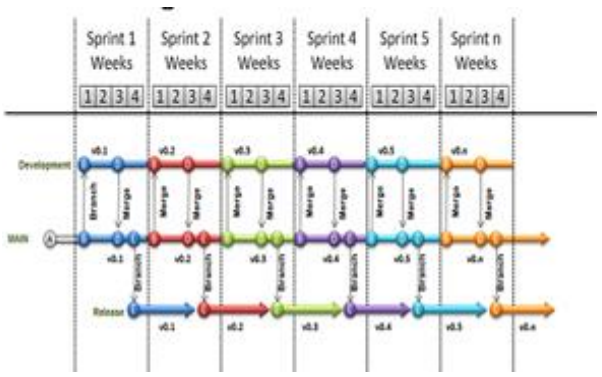

Figure 1 "branching for smaller scrum teams" [2]

Branching and Merging in Complex SCRUM Projects: "In larger SCRUM projects, product backlog with be extensive, development of product might take longer timeframe or multiple scrum teams might be working in parallel in a shorter time frame" [2]. The figure 2 shows three scrum teams contributing to the working increment of software delivered at the end of each sprint. At the end of each sprint, each of the team might participate in a joint review or each team might be having their own review and then followed by a joint review before starting of next sprint. As in figure 2, Main branch is branched into 3 branches to support parallel development for a sprint. Now after the development work is done a merge request is sent from main line to each sprint team and then code is merged back to the main line. After merging the entire the work for entire sprint 1 then main branch is branched to release. [2]

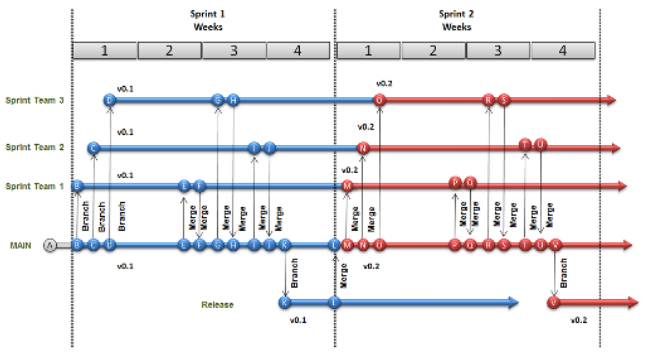

Figure 2 "branching for complex scrum teams" [2]

In branch by feature kind of branching, branching is done for every feature and merged after that feature is developed. If there is any urgent bug fix in the middle of a feature then that fix is branched to another separate branch and is done separately from development branch. So that it is also included in next release. [3]

<u>References:</u>

1. http://eugenedvorkin.com/continuous-integration-strategies-for-branching-and-merging/
2. http://blogs.msdn.com/b/billheys/archive/2011/01/18/branching-for-scrum.aspx
3. http://wiki.colar.net/software_releases_-scrum_and_subversion_branches#.VN5_8fnF_0c

Part - 2

Fossil is an open source version control system with a number of benefits [1]. The ways in which fossil supports the four functions (Identification, Control, Auditing, and Accounting) of software configuration management is by its features like Bug Tracking and Wiki, Web Interface, Autosync, simple networking, robust and reliable [1].

Configuration Identification is to specify baselines in Software Development Life Cycle (SDLC) where one can evaluate the deliverables and label all configuration items uniquely [2]. How fossil supports Configuration Identification: In fossil tool every version of file is called as an "Artifact". Each Artifact has a unique ID called "Artifact ID" which is a 40 character hexadecimal. Whenever a change is done to the existing artifact the new artifact will be saved with a new artifact ID. Thus a new version of file is created and older version also remains in the repository, an unordered collection of artifacts. Fossil can use any text editor to document the changes using Bug Tracking and Wiki. With these features Fossil can evaluate and label all configuration items uniquely. Thus fossil supports configuration identification.

Configuration Control is to keep track of the changes made to the configuration items [2]. How fossil supports Configuration Control: Fossil supports Autosync and Manual-merge feature. This feature in Fossil tool helps users to identify change request, evaluate change request, produce a change proposal, approve or disapprove change proposal and finally release of a change. Using Autosync, everything is done automatically i.e., forking and merging becomes very easy. But if user wish to control the process then one can choose Manual merge feature. [3]

Configuration Auditing is to "verify the system integrity in terms of degree to which it complies with its specifications" [2]. How fossil supports Configuration auditing: Repositories maintain manifest files and Artifacts. Communication between repositories is done via HTTP. Each remote repository is identified by a URL. One can point a web browser and get information required for auditing such as status, history and tracking information. This information helps the auditing person to verify the integrity of the system with its specifications.

Configuration Status Accounting is to understand the status of project deliverables at specific point in SDLC [2]. How fossil supports Configuration Status Accounting: In repositories all the Artifacts are stored and are immutable so user can get the status of work at the required point of time and can use the data for status accounting. Also, fossil tool supports two types of repositories. One is Remote repository and other is local repository. By using clone operation, each user can maintain a copy of their own local repository, duplicate of remote repository. [3]

Thus Fossil tool supports all the functions of the configuration management. A part from that Fossil tool is very easy to use with its features like Web Interface, self-contained, simple networking, CGI enabled [1]. The tools is robust as it uses enduring file format so all the transactions are atomic during emergency or power loss and also it can maintain years of data without data loss.

References:

1. http://fossil-scm.org/index.html/doc/trunk/www/index.wiki
2. http://www.chambers.com.au/glossary/configuration_management.php
3. http://fossil-scm.org/xfer/doc/tip/www/concepts.wiki