A COMMON SOFTWARE CONFIGURATION MANAGEMENT SYSTEM FOR CERN SPS AND LEP ACCELERATORS AND TECHNICAL SERVICES

E. Hatziangeli, R. Bartolome, A. Bragg, P. Ninin, J. Patino, H. Sobczak, CERN, Geneva, Switzerland

Abstract

Software configuration management activities are crucial to assure the integrity of current operational and the quality of new software either being developed at CERN or outsourced. The functionality of the present management system became insufficient with large maintenance overheads. In order to improve our situation, a new software configuration management system has been set up. It is based on Razor ®, a commercial tool, which supports the management of file versions and operational software releases, along with integrated problem reporting capabilities. In addition to the basic tool functionality, automated procedures were custom made, for the installation and distribution of operational software. Policies were developed and applied over the software development life cycle to provide visibility and control. The system ensures that, at all times, the status and location of all deliverable versions are known, the state of shared objects is carefully controlled and unauthorised changes prevented. It provides a managed environment for software development, in various domains of the SPS and LEP CERN accelerators, and the technical services, automating code and lifecycle management. This paper outlines the reasons for selecting the chosen tool, the implementation of the system, the problems solved and the final goals achieved.

1 BACKGROUND

The present software management system for the control software of LEP and SPS and Technical services was developed in CERN in the late '80s. Its capabilities were limited in a basic version management of only the head and the previous version of all software items in the repository. With the move to HP-UX, as the main development platform, and the use of PCs and Power PCs running LynxOS, the limitations of the present system were reached.

The maintenance of the system itself became very taxing to the software administrators, and extensions of its functionality were difficult to implement. This, in addition to the lack of proper procedures to allow the introduction of externally developed software, had led to the creation of the project for the implementation of a common Software Configuration Management (SCM) System.

2 THE PROJECT

The project started with the process of evaluation, in order to determine the best possible SCM system for our needs. Therefore, this first phase of the project, was focused in:

- Identifying and capturing all user requirements [R1].
- Conducting an in depth evaluation of CERN, commercial and public domain SCM solutions.
- Identified and evaluated the impact of the possible solutions on our present software development.
- Produce a technical proposal with the evaluation of the recommended SCM systems [R2].

Once a solution was found, the final phase of the project was concentrated in the implementation and deployment of a complete SCM system capable of supporting, amongst others:

- Present and future in-house or subcontracted software development.
- Management of Software Problems and Change Requests, and being able to relate changes to the issues that drive them.
- Common software repository, which will facilitate the exchange of software and documentation between various CERN divisions and groups.
- Software management practices, and proper development procedures and policies.
- Properly supported SCM service.

3 THE REQUIREMENTS

The project had to provide an integrated solution to the following requirements:

- Improve the complete software development and maintenance cycle.
- Make the testing of software easier.
- Provide identification and tracebilily of software components that are related together.
- Manage consistent product releases in an error-free manner.
- Automate CM procedures, removing error-prone steps.
- Ability to understand and evaluate correctly the real impact of the changes in the Operational software.

Overall, a group of weighted requirements [R1] were collected from our users, administrators, project and functional managers, which were addressing usability, performance and scalability criteria. Once the available public domain and commercial CM tools were identified, they were compared against our selection requirements and the best two products [R2] were chosen for a detailed technical evaluation.

4 THE SOLUTION

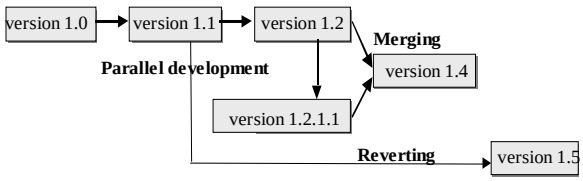
In an effort to provide the best solution, a proper SCM system should integrate together and automate all CM processes. The CM solution implemented in this project provides:

- A set of CM procedures and policies to support our software development process.
- A Change Management process, by which software problems, bugs and enhancement requests are reported to the project responsible team and are followed through the software lifecycle.
- A commercial tool, named Razor ® [R3], which supports the management of file versions and operational software releases, along with integrated problem reporting capabilities.
- A CM administration service and support, for the users developing new projects and/or maintaining existing software.

5 THE OVERALL SYSTEM

5.1 Version Management

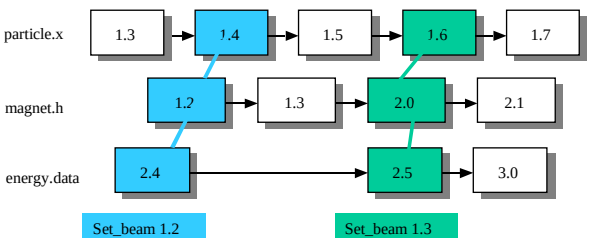
The checkout/checkin paradigm is used, which provides versioning of individual components and concurrency control on components through locking as well as through



branching and merging.

5.2 Release management

A release is a collection of specific versions of software items (configuration items), that are grouped together for a reason, i.e., building, testing, prototyping. Once a configuration of a new product release is ready, the corresponding versions of the configuration items are gathered together, tested and released into the operational area. The configuration and the evolution of our product releases are kept in the repository. The access of a complete release is via the checkout/checkin model.



5.3 Change Management

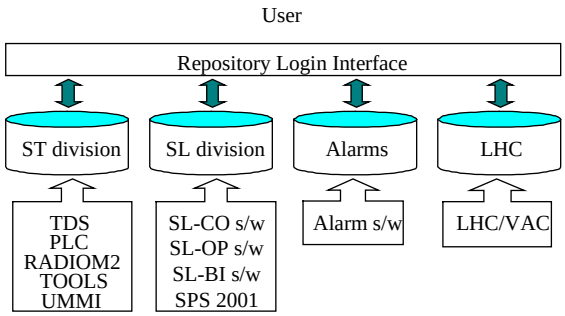
All software problems and enhancement requests are managed through our software lifecycle.

Software changes are tracked by relating versions of files or products releases with one or more change requests. Every request follows a specific lifecycle, acquiring a *signature* by the promoter. The promotion of a request to the next phase is done by the project leader, or the developer assigned to implement the request. The lifecycle is tailored to suit the needs of the specific software development process. All configuration items, which are checked out, are automatically stamped with the related change requests, which have achieved the proper *signature* level.

5.4 The repository

Our software is organised into several individual, and physically separate, repositories, in order to achieve maximum customisation (archive engine, lifecycle, etc.).

Each repository contains software projects related by a common theme. It is set up using standard default policies, which are finely tailored to suit the local development process of each CERN division/group.



5.5 The SCM tool

Razor ® [R3] was chosen because it demonstrated a good all round functionality, including an intuitive GUI and command line interface, a simple set up and installation, straight forward migration of our existing software, minimal learning curve for our users and administrators, and easy customisation, with a good level of technical support.

In addition, it demonstrated the ability to scale up, in order to accommodate larger number of users and data for the future, as our needs evolve.

5.6 Roles and Responsibilities

The following set of roles was created, and different responsibilities assigned to each role, according to the corresponding activities.

The **Software Engineers** (Developers) are responsible for developing and maintaining their software products and creating their own releases. In addition, they respond to change requests from their clients.

The **Librarian** is in charge of the installation and distribution of new versions of public software (libraries) and operational product releases.

The **Configuration Manager** is in charge of the user training, the development and tuning of the CM procedures and policies, the repositories and the CM tool. The configuration manager also ensures that the procedures for creating, changing, testing and releasing code are followed properly.

5.7 Access Control

Using the tool's intrinsic functionality, additional procedures were developed to regulate access control to each project in the repository. It is carried out transparent to the user without the need of passwords.

The underlying principle behind the access control is that only the software owner and its project team should have write access to a configuration item, being, individual files and projects as a whole.

In addition, the developer's CERN Division/Group is given access to the project, while the project team is notified by email, specifying who and what software has been accessed. This is necessary, in order to allow emergency bug fixes to take place in the absence of the project team.

5.8 The Build and Release Management Process

Build management is the process of combining configuration items, which belong to a baseline, together into composites. It is done in order to construct all or part of the product deliverables from its components, for the purpose of prototyping, testing new functionality, or before releasing a new version of a product into production. The Build process is an automated process, which builds the product for every platform it will be delivered for. Electronic logs are kept to insure tracebility and completeness of the Build, as well as being able to compare against previous Builds or reproduce the exact same Builds in the future.

Release management is the process of releasing a built and tested system, into the Operational distribution area. The automated Release process installs and distributes all deliverables on their corresponding platforms to be used in operation. All previous operational versions of a product are available, and automatic version revert capabilities are offered to the Operations crew.

6 BENEFITS

6.1 Operational Software

- Delivery of consistent operational software.
- Being able to trace and identify any component of an operational system, as well as the exact version of the software running in the control room.
- Minimisation of uncontrolled changes of the operational software.
- Able to access any released version of the operational software or reverting back to previous working version, in case of unforeseen problems.

6.2 Development Environment

- Development of thorough policies concerning the construction, integration and installation of operational software and the introduction of software modifications.
- Public software packages, which are commonly used by the development teams to produce operational software, change in a controlled manner.
- Easy to debug, since the exact version of the software concerned can be reproduced.
- Transparent access of software and documentation.
- Software written by external contractors can be properly synchronised with software written in-house, since the introduction of new software and of software changes follows a well-documented procedure.

7 CONCLUSIONS

An effective SCM solution involves more than buying the latest and most functional CM tool available in the market. The most important factor is identifying and putting the proper SCM processes in place.

In order to define correctly the SCM process that fits the organisation, one should identify and understand the existing processes in place, so they can be refined or even re-implemented, if they are inadequate.

Moreover, in order to introduce SCM activities successfully in an organisation, there are two important prerequisites.

Firstly, the management should perceive SCM functions as key issues in software development and it should commit to the process.

Secondly, one should identify the hidden cultural constraints in the target group, as they might have a strong bearing on the final SCM solution and the success of the system.

REFERENCES

[1] R. Bartolome, A. Bland, E. Hatziangeli, I. Last, P. Ninin, "Software Configuration and Management System User Requirements Document", CERN SL Note (CO) 97-59.

[2] R. Bartolome, A. Bland, E. Hatziangeli, I. Last, P. Ninin, H. Sobczak, "Software Configuration and Management System Project Evaluation Report", CERN SL Note (CO) 98-22.

[3] Tower Concepts, Inc., "Razor ® Release Management, File Version Control, Problem Tracking", Oct 1998.