Position Paper

Experience of Using a Simple SCM Tool in a Complex Development Environment

Ivica Crnkovic
ABB Industrial Systems AB
721 67 Västerås
ivica@sw.seisy.abb.se

1.0 SCM Tool used in ABB Industrial Systems

1.1 Background

Five years ago our organization started the development of a new generation of real-time systems for process control. The new process supervision systems were based on Unix and Motif. The new development process has required a new, modern and efficient Software Configuration Management (SCM) tool. After the investigation of SCM tools on the market, and some unsuccessful tests of different commercial products, it was decided to start with a simple tool, well integrated in Unix. RCS (Revision Control System) was chosen.

As the number of programmers grew and development projects became larger and more complex, the need for a support beyond pure RCS became apparent. At that time we did not find a CM tool that fulfilled our requirements, so we have started building new functions based on RCS. This development resulted in a product called **SDE** (Software Development Environment) that is used today by several hundreds of programmers.

This paper gives a brief description of the tool and summarizes the experience of its usage.

1.2 SDE basic characteristics

SDE is a software package based on RCS and adjusted for the development of large systems. It includes functions that define working environments for projects and structures of repositories where versioned files are saved. Hierarchical structures of repositories are divided into configurations which makes it possible to develop different versions of software in parallel.

Using slightly modified RCS commands and some new commands related to RCS files, SDE enables easier and faster browsing through the versioned files. In a project environment, users always have a view to a specific configuration of file versions.

SDE includes support for Change Management. Every logical change that has to be done in a development/maintenance session, is described in a Change Request (CR). When a modified file is checked in, it is registered in a CR. A CR is implemented as a versioned file, so it automatically gets RCS file attributes, like state, author, and change date.

SDE uses **make** for building systems. Makefiles that are automatically generated, use definitions of configurations of all used components, both internal and external.

The Product Management part includes support for the generation of products. A release specification, which is automatically generated, describes changes made in the product.

All SDE functions are available as shell commands. On top of them there are several applications with graphical user interface.

2.0 Experience of the SDE usage

2.1 Positive experience

The four-year experience of SDE usage shows the importance of having a CM tool that is simple, flexible and efficient. RCS is both simple and flexible and it offers wide possibilities to develop methods that can support different development processes.

SDE functionality has grown as the development projects have grown so SDE has lived up to the new requirements. The SDE basic functions, related to project management and check-in/check-out procedures, are introduced in the organization smoothly without big problems. The new functions of SDE, dealing with formal methods for managing development processes, were added after the basic part was widely used. These more complex parts were fast accepted by some projects, while it took more time for other projects. Some projects have simply stayed with the basic functionality. The experience has shown that it is good to have the possibility of using different levels of complexity.

Change Management is a facility that helps both programmers and project managers to organize and follow their activities. CRs are used as a “to do” list and as formal descriptions of changes made in a product version.

Two kinds of SDE users exist today: Those, experienced Unix users, which mostly use SDE shell commands, and those, with a MS-Windows background or less experienced Unix users, which are using only GUI-applications. It seems that it is important to offer both types of user interfaces - at least on Unix platforms.

2.2 Negative experience

Problems which SDE users meet, are in the area of security and performance. When a number of programmers are involved in the same project, the Unix security, which is the base for the SDE security, is not sufficient. There is always a risk that a programmer removes, not only his/her working files, but also the versioned files.

The performance problem becomes serious when creating or merging very large software configurations where a large number of files are checked out and in.

SDE also uses RCS files for saving non-ascii files. The format of versioned files is not optimized for binary files - they occupy a large space, and the check-in/check-out procedures are slow.

Makefiles, automatically generated, look very complicated and manual modifications require a lot of efforts. Files created by make do not have a version identity, so a lot of unnecessary building is done.

SDE is missing support for propagating of a change done in one version of software to other versions.

2.3 Conclusion

Using a base tool like RCS is not sufficient for large software systems, but such a tool is very good for building more complex tools on top of it.

The experience has shown that it is very important to have an SCM tool that is a general enough and flexible enough for adjusting to development processes.