Change Measurements in an SCM process

Ivica Crnkovic, ivica@sw.seisy.abb.se, ABB Industrial Systems, Västerås, Sweden
Per Willför, dat95pwr@idt.mdh.se, Mälardalens Högskola, Västerås, Sweden

Abstract. An SCM database contains data which can be used as input for Software Metrics. Both data for Size-Oriented Metrics, and information for Process-Oriented Metrics are available from SCM systems. This paper describes measurements taken from an SCM database used at ABB Industrial Systems. The SCM tool is change-oriented and collects information about changes in Change Request (CR) documents. As CRs are under version control, the measurements taken on them give information not only about the amount and type of changes but also about the change process behavior. The measurements, generated from CRs by a tool, are used during the development process and in the final analysis of the project. The paper presents some of the measurements showing typical cases of lifecycle models.

1 Introduction

There are a number of different Software Metrics and they can be classified in relation to the relevant development process, products or resources [1]. Typical product metrics are size metrics (number of lines of code, number of documents, etc.), quality metrics, etc. Process metrics are a result of measurements related to the different phases of the development process. Process measurements help us to understand the processes concerned, to control them, improve and predict them [2].

Can SCM tools provide information for process metrics? The answer is yes for tools which integrate a Process Management with SCM, for example ClearGuide [3]. Change-oriented tools provade information about changes which can also be used as input to process measurements.

This paper describes a metrics tool and measurements performed on a change-oriented SCM tool, designated Software Development Environment (SDE), developed and used at ABB Industrial Systems [4]. Different metrics showing the behavior of development projects are presented.

2 Change Management in SDE

2.1 SDE basic characteristics

SDE is a software package intended for use in the development of large systems. SDE is based partly on RCS[5]. Using slightly modified RCS commands and certain new commands related to RCS files, SDE enables easy and fast browsing through the hierarchical system structures and versioned files. The SDE and RCS commands are encapsulated in GUI-applications.

2.2 Change Management

SDE provides support in the management of Software Development Processes. Change Management is a part of the support. Any change in SDE is under change-set control. A basic item of SDE Change Management is a Change Request (CR), an entity which describes a logical change to be made in a software system. Change Requests are created from Requirement Specifications or from defect reports. During the development process CRs collect information about physical changes made in the system: When a developer checks in a file, he/she refers to a related CR. The file name, file version and log message are registered in the CR. The final version of a CR includes both a description of a logical change and information about all modified file versions. The references between CRs and changed files are controlled by the SDE tools. No change can be introduced in the software without referring to a CR.

A Change Request passes through different states during the development process. When a CR is created it is in the state *Init*. During the work sessions it passes through other states, such as *Exp, Implemented, Tested* and reaches the *Terminated* state. The CRs integrated in a product release are in the state *Released.* Figure 1 shows different states of a CR in a development process.

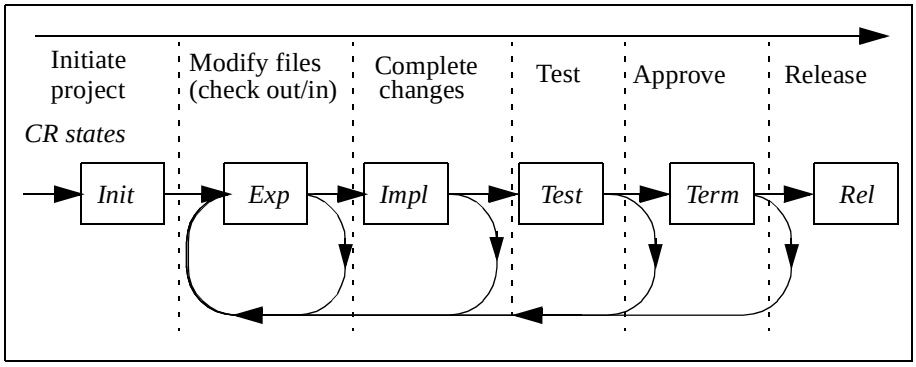

Fig. 1. Change Requests in a development process

Change Requests are under version control. Each time a CR is changed it is automatically checked out and after the modification, checked in. CRs are saved in an RCS directory located in a CR library which contains all logical changes of a software version. As a versioned file under RCS control, a CR includes not only the change description and list of changed file versions, but also attributes from RCS: a state, a responsible user (author) date of change and other RCS attributes.

Change Requests are implemented as text files which follow a specific syntax. The header part of a CR includes keywords such as Priority and CR Type, creation date and termination date. A list of files being checked in follows. The body part includes a description of the change and log messages of the files checked in.

3 CR Metrics

The main purpose in using CR metrics is to increase the predictability of the behavior of software development projects. The second purpose is to obtain an overview of the current project status. To achieve these objectives we need answers to the following questions:

- How many changes and types of changes have been made, how many changes should be made?
- What is the dynamic of the changes? How often are further changes required and how does the project respond to them?
- How are changes distributed among developers? What is the current state of change per developer? How many changes has a developer to implement?
- How did the change process look in the projects completed previously?

CRs contain data which can be used for providing the answers. The number of changes, their states, classification according to priority, type or function, number of changed files, etc. is one type of possible metrics. Since CRs are under version control, the history of every change is also available. The states of all changes are available for the whole period of the development process. The state changes data in a time period is input to another type of metrics - process metrics. There is also information about the authors of changes introduced in CRs. It is therefore possible to obtain metrics for the entire development project and also for every member.

A CR-Metrics application collects data from CR libraries, by parsing CRs. Information about every CR version is taken and saved in a spreadsheet. The measurements are displayed as embedded Excel objects [6] in the form of different graphs. All measurements are performed in a similar way. All versions of all CRs are parsed and different criteria are used for extracting data.

The CR-Metrics application presents the following measurements:

Current status	The states of the latest CR versions for each period are shown.
Accumulated CRs	Completed and incomplete CR are sorted according to date.
New CRs	The graph shows the number of new CRs in a time interval.
Latest Changes	The CRs being changed during a given period are presented.
New and completed	The number of new and the number of completed (“input” and “output”) CRs per time interval are shown.
CR life length	A distribution of CR life lengths is shown.
CR Type	CRs are classified according to CR types (Error, Improvement, etc.).
Priority and State	A classification according to the change priority for completed and incomplete CRs is shown.

4 Measurements

Some of the metrics for a development project designated WinSDE 1.1-0 are presented here. The graphs show characteristics of the development process, i.e. the dynamics of change introduction and their completion.

The WinSDE 1.1-0 project uses a development model which is a combination of the spiral model and the evolutionary prototyping model [7]: The development is performed as a number of iterations, each consisting of several phases: prototyping, evaluation of different alternatives, refining the prototype, developing and building the deliverables and a plan for the next iteration. An implication of this model is a constant growth of the number of CRs - new requirements are defined during each iteration (refinements) and at the beginning of the new iteration (new functions). In the last iteration the number of new CRs describing new functions declines, but new CRs related to the release activities are still being created. The test activities with CR termination become more intensive at the end of each iteration, but especially at the end of the project. The unsolved CRs are postponed to a succeeding release.

The current status graph (Figure 2) shows the number of CRs created during the project. CRs are classified according to their states. The graph shows how many changes have been completed and how many remain open.

The example in the graph shows that the number of new CRs grows constantly. The number of completed CRs also increases. The number of open CRs is approximately the same during the entire project except during the last phase. Some steps in completing CRs can be seen - when the time for a new deliverable approaches, many CRs are terminated.

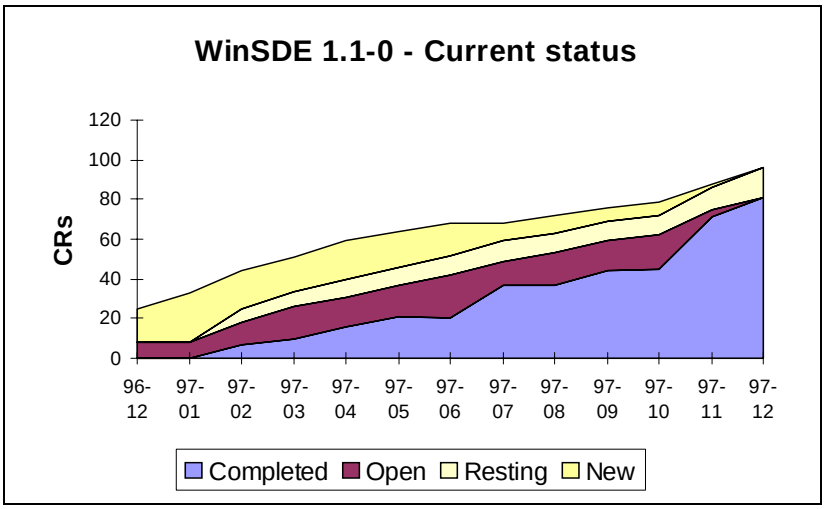

Fig. 2. Current states of CRs for a project using the Spiral model

Figure 3 shows the same type of graph for another project which followed the Waterfall model. In the project initiation phase, all the requirements have been defined and CRs have been created. The graph shows how the work has improved. The number of open CRs increases, and in the final, test phase, the CRs are being completed.

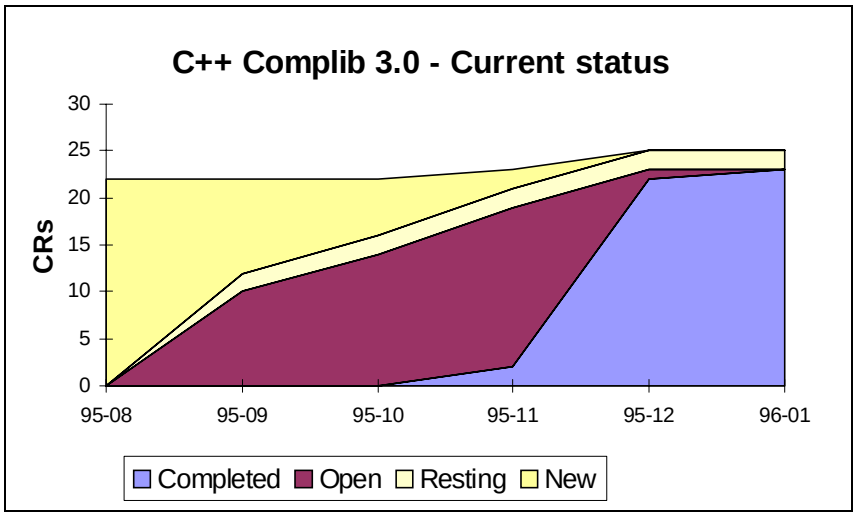

Fig. 3. Current states of CRs for a project using the Waterfall model

Figure 4 shows the distribution of the times taken in making changes. The time on the X-axis is the time interval between the reception of the CR and the completion of the change. The Y-axis shows the number of changes which are completed in the specified period. The graph indicates that most of the CRs are completed within six months, and it suggests that it is acceptable to deliver the product modified in that period without waiting further six months for making the remaining changes planned in the project..

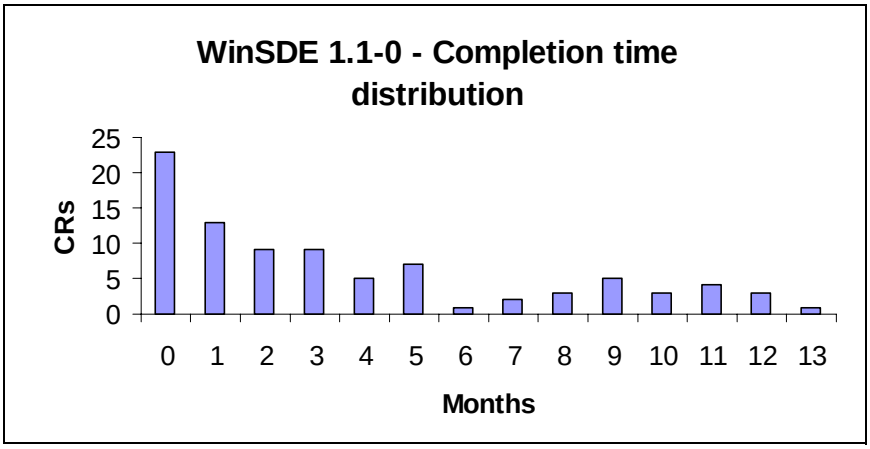

Fig. 4. Distribution of times for making changes

5 Conclusion

The measurements shown in this paper illustrate the possibility of using an SCM database not only for size metrics but also for process metrics. The process metrics should lead to a better understanding of development processes and help in making appropriate decisions in process management. Experience in the use of the CR-Metrics tool at ABB Industrial Systems remains limited because of its recent development. However, a case study has shown very similar patterns of metrics: Three projects for the development of similar products using the Spiral model have been measured, and the results from CR-metrics have shown surprisingly many similarities. The total number of CRs was three times greater than the number of CRs at the outset. The number of postponed CRs was approximately 20% of the total number of CRs (which suggests a weakness in the project planning). The number of open, i.e CRs being processed was constantly about 6 per developer during the development process in all projects. The most interesting fact is that the shapes of the curves in all projects were similar.

The idea is to use CR-metrics during execution of the project when both the project manager and project members can follow up the project status. The final measurements can be taken at the completion of the project. The metrics can be compared with those from other projects and related to other facts. One advantage of the tool is the possibility of collecting data from earlier projects, as all data concerned is under SCM control.

References

1. The Software Measurement Laboratory, University of Magdeburg, *http://irb.cs.uni-magdeburg.de/sw-eng/us/metclas/index.shtml*
2. William A Florac, Robert E. Part, Anita D. Carleton - Practical Software Measurement: Measuring for process management and Improvement, *Software Engineering Institute, Carnegie Mellon University, CMU/SEI-976-HB-003, April 1997*
3. David B. Leblang, Managing the Software Development Process with ClearGuide, *Software Configuration Management ICSE'97 Workshop, Boston, May 1997, proceedings, Springer Verlag, ISBN 3-540-63014-7*, pages 66-80
4. Ivica Crnkovic, Experience with Change-Oriented SCM Tools, *Software Configuration Management ICSE'97 Workshop, Boston, May 1997, proceedings, Springer Verlag, ISBN 3-540-63014-7*, pages 222-234
5. Walter F. Tichy, RCS - A System for Version Control, S*oftware and Practice Experience*, 15(7):635-654, 1985
6. Microsoft Visual Basic 5.0 ActiveX Controls Reference, 1997, ISBN 1-57231-508-3
7. Steve McConnell, *Rapid Development: timing wild software schedules*, Microsoft Press, 1996, ISBN 1-55615-900-5