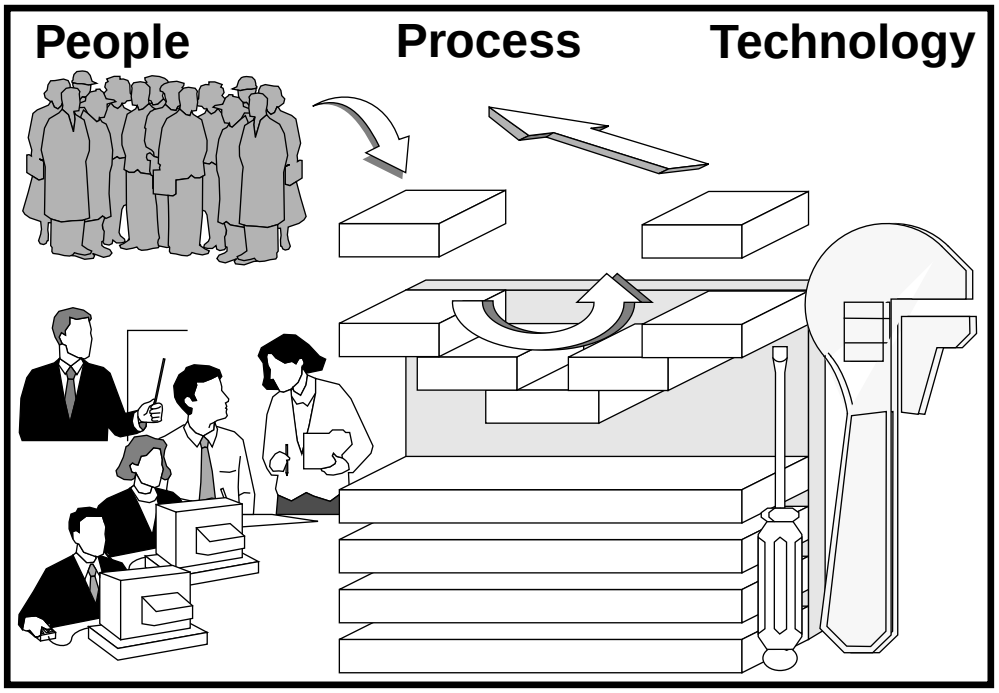
SOFTWARE PROCESS AND QUALITY

Arash Khodabandeh
Information and Programming Technology Group
CERN, Geneva, Switzerland

The production of software is a labor intensive activity, especially in the field of High Energy Physics, where the complexity of current and future experiments require large software projects.

For most of the scientists and engineers involved in software production, the business is science or engineering, not computing. As software scope continues to grow so does the feeling that its development and maintenance are out of control. The situation is made even worse by a lack of software engineers and from an uneven software culture. Better organization and control of software production are clearly needed in order to face the challenge of developing new software for the LHC and of maintaining software for previous experiments.



1 SOFTWARE PROCESS AND QUALITY

To be able to control the production of software it is essential to improve (a) the knowledge of the PEOPLE involved, (b) the organization and improvement of the software development PROCESS and (c) the TECHNOLOGY used in the various aspects of this activity. The goals are better systems at lower cost, and of better quality.

The process is the set of orderly actions to be performed to produce the software throughout the life cycle. The quality of a process can be measured in terms of maturity against a recognized framework. The reference framework is the Capability Maturity Model (CMM) proposed by the Software Engineering Institute (SEI). This model consists of five levels and the most difficult step is to move from level 1 to level 2 because of all the management procedures and activities that have to be put in place. As an organization moves up the maturity levels, management visibility on the software process improves, estimates become more accurate, schedules are met more precisely and the time required to produce a software system shortens.

Except for level 1, each maturity level is decomposed into several key process areas (KPA) that indicate the areas an organization should focus on to improve its software process. One of the KPAs of level 2 is Quality Assurance whose goal is to verify process and product compliance and to address non compliance. This verification can only take place if effective measurements are performed to quantify compliance.

2 SOFTWARE METRICS

Measurement can affect all aspects of the software process including the software itself. Among various aspects of software metrics we will mainly concentrate on metrics derived directly from the source code.

The correlation of software metrics and source code quality have been verified by experimental studies conducted for example by T. McCabe and M. Halstead. Closer to us, E. Lancon studied the evolution of the ALEPH reconstruction code called JULIA from 1990 to 1995. It showed that the routines with the poorest quality in terms of metrics score are also the one modified most for bug correction, and after 5 years of such a correction the quality of the code automatically improved.

Early measurement helps achieving quality by monitoring complexity, quantifying test effort, forecasting maintenance cost, identifying risks or planing preventive maintenance. Once the problem to be solved has been clearly defined, a quality model can be build combining the right metrics as basic components for measurement. The quality model defines the acceptable range of complexity. It needs to be adapted both to the development team expertise level and to the type of project.

Software tools can help collecting the metrics and assess the software compared to the defined quality model. One approach is to use source code metrics patterns to cope with the amount of collected data.

3 SOFTWARE METRICS LAB WORK

In the hands-on sessions, students practiced with Logiscope, a tool for software metrics. Logiscope is a toolbox for improving programming quality and test coverage. It can analyze more than 80 language variations including C, C++ and Fortran. Logiscope features include:

- Code quality with support for software metrics computation to assess maintainability, testability and component re-usability;
- Test coverage with support for coverage rates on source code branches, procedure calls, instruction blocks etc.;
- Code standards with support for verification of the program against programming rules and customization of rules to check;
- Graphical reverse engineering.

The lab works started with a walk through demo on a sample C++ code with step by step instructions through a possible quality analysis session with introductions to the most common features of the tool. For the remaining part of the session, students could analyze their own code (C, C++ or Fortran) using provided list of hints and road map or patterns presented during the lecture. Some of them undertook corrective actions to improve the quality of the code or took results back home to share with colleagues.

The lab work were followed by a wrap up session where 6 groups presented very interesting results and their conclusions about the approach.

4 REFERENCES

Complete information, the slides of the lectures, articles and related WWW pointers are at the following URL:

`http://www.cern.ch/IPT/CSC/1997/`

4.1 METRICS

[1] B.W.Boehm, ‘Characteristics of Software Quality”, TRW, 1975

[2] B.W.Boehm, “Software Engineering Economics”, Prentice Hall

[3] R.B.Grady, “Successfully Applying Software Metrics”, *IEEE Transaction Software Engineering*, vol. 27, n. 9, Sep. 1994

[4] M.H.Halstead, “Elements of Software Science”, Elsevier, 1977

[5] M.A.Hennell, D.Hedley, M.R.Woodward, “Experience with Path Analysis and Testing Programs”, University of Liverpool Publication

[6] M.A.Hennell, D.Hedley, M.R.Woodward, “On Program Analysis”, University of Liverpool Publication

[7] ISO/IEC 12119:1994, 'Information Technology - Software Packages - Quality Requirements and Testing”

[8] T.McCabe, “A complexity Measure”, *IEEE Transaction on Software Engineering* - vol. SE-2, n. 4, pp. 308-320, Dec. 1976

[9] T.McCabe, C.W.Butler, “Design Complexity Measurement and Testing”, *CACN*, vol. 32, n. 12, pp. 1415-1425, Dec. 1989

[10] J.A.McCall, “Factor in Software Quality”, *General Electric* n. 77C1502, June 1977

[11] B.A.Neejmeh, “NPATH: A Measure of Execution Path Complexity and its Applications”, *Communication of the ACM*, vol. 31, n. 2, 1988

[12] S.N.Mohanty, “Models and Measurements for Quality Assessment of Software Computing Survey”, vol. 11, n. 3, Sep. 1979

[13] R.S.Pressman, “Software Engineering, a Practitioner’s Approach”, McGraw-Hill

[14] D.Schutt, “On a Hypergraph Oriented Measure for Applied Computer Science”, *Proc COMPCON*, pp. 295-296, 1977

[15] M.R.Woodward, “An Investigation into Program Paths and their Representation”, *Technique et Science Informatique*, vol. 8, n. 4, 1984

[16] SEI, “C4 Software Technology Reference Guide”, CMU/SI-97-HB-001, http://www.sei.cmu.edu/technology/str/, Jan. 97

[17] E.Lancon, “Software metrics can improve HEP software quality”, *CHEP’95*, http://www.hep.net/conferences/chep95/html/abstract/abs_45.htm

4.2 PEOPLE, PROCESS, TECHNOLOGY

[18] ESA, “Software Engineering Standards”, Prentice-Hall, ISBN: 0-13-106568-8

[19] ESA, “Software Engineering Guides”, Prentice-Hall, ISBN: 0-13-449281-1

[20] W.S.Humphrey, “Managing the Software Process”, Addison-Wesley, 1990, ISBN: 0-201-18095-2

[21] W.S.Humphrey, “A Discipline for Software Engineering”, Addison-Wesley, 1995, ISBN: 0-201-54610-8

[22] W.S.Humphrey, “Introduction to the Personal Software Process”, Addison-Wesley, 1997, ISBN: 0-201-54809-7

[23] ATLAS, “ATLAS Software Development Environment, User Requirement Document”, Issue 1.0, ATLAS internal note SOFT-NO-034, Dec 1996, http://atlasinfo.cern.ch/Atlas/documentation/notes/SOFTWARE/notes.html

[24] W.S.Humphrey, “CASE planning and Software Process”, CMU-SEI-89-TR-26, May 1989, http://www.sei.cmu.edu/products/publications/89.reports/89.tr.026.html

[25] V.Stenning, “On the Role of an Environment”, *9th International Conference on Software Engineering*, Montery, California, March 30, 1987

[26] W.E.Deming, “Quality, productivity and Competitive Position”, MIT 1982

[27] SEI, “Capability Maturity Model for Software”, V1.1. CMU/SEI-93-TR-24, 1993

[28] SEI, “Key Practices of the Capability Maturity Model”, V1.1. CMU/SEI-93-TR-25, 1993

[29] J.Herbsleb, A.Carleton, J.Rozum J.Siegel, D.Zubrow, “Benefits of CMM-Based Software Process Improvement: Initial results”, CMU/SEI-94-TR-13, 1994

[30] SEI, “Software Quality Assurance”, SW-CMM v2 Draft A, Oct 96, http://www.sei.cmu.edu/technology/cmm/draft-a/a25qa.html

[31] B.J.Brown, “Assurance of Software Quality”, SEI-CM-7-1.1, Jul 87, http://www.sei.cmu.edu/products/publications/cms/cm.007.html