Empowering Software Development Environments by Automatic Software Measurement

Bernhard Daubner
Bayreuth University
Chair of Applied Computer Science I
95440 Bayreuth, Germany
bernhard.daubner@uni-bayreuth.de

Abstract

In order to facilitate the application of software measurement the gathering of software measures should be automated as far as possible by the integration of the software measurement process into the software development environment. This paper suggests to tie software measures up to the elements of a software process model. Then we present the prototypical implementation of a tool called Metrics Builder, which allows the definition of arbitrary software measures that can then be used within the software development environment.

1. Motivation

Today's software development environments (SDE) contain a variety of information about many elements of the software development process. Not only attributes of the edited source code can be analyzed, but also a lot of information about the project resources (e.g. personnel effort, development time) and project activities (e.g. analyzing, coding, testing) is provided. Various software measures could be applied on these pieces of information. Besides the software measures for the work products there exist also measures for the process roles, the activities and the whole software development process. A systematic overview on applicable measures for the different elements of a software process can be found in the books of Fenton/Pfleeger [3] or Grady [4].

But this variety is scarcely supported by current SDE. Instead of that most SDE (e.g. *Together ControlCenter*) are limited to the computation of size and complexity measures for the source code. And it is even less possible for the user to combine several of the measures provided by the SDE in order to analyse certain aspects of the current software development process or to show relationships between certain elements of the software development process.

This paper wants to outline an approach, which makes a more exhaustive use of the information provided by the SDE in order to define and analyse software measures. Especially we want to be able to define software measures for the activities of a software development process and to make individual process characteristics like productivity or progress measurable. Thereby we use a software process model in order to define the anchor points for the software measures, a project management tool to get information about the structure of the actual project and a software development environment to evaluate the software measures we have defined on the basis of the process model.

2. The *V-Modell XT* process model

The *V-Modell XT* [2] is the standard process model in Germany for managing government IT development projects and describes the software development process as a collaboration between roles that perform activities on work products. Thereby it defines process building blocks in order to combine those roles, activities and products that are crucial for a certain task. Activities and products with a strong relationship to one anothere are combined within the process building blocks to activity groups and product groups respectively. Thus the process building blocks are organized hierarchically and build the modules of the V-Modell XT.

We want to use this process model for the definition of anchor points for the software measures. To illustrate this we look at the process building block *Software Development*, which contains the activity groups *System Design*, *System Specification* and *System Elements*. Three product groups with the same name are dedi-

cated to these three activity groups. Within the acvitity group *System Design* the products *Software Architecture*, *Database Design* and *Implementation, Integration and Testing Plan* are built.

Starting from this hierarchical structure one can immediately derive corresponding size and effort measures, which can be applied on several levels of abstraction on the products and activities, the product groups and activity groups and finally on the process building block itself.

3. The Measurement Approach

Within the software development environment the software measures under consideration are to be evaluated. Therefore the evaluation software has to know for example that the effort for a entity-relationship diagram created within the SDE has to be accounted for the activity *Database Design* within the activity group *System Design* of the process building block *Software Development*. This information though can be provided by the project management tool.

When tailoring and instantiating the software process model for an actual software development project one certainly has to establish a project plan in order to assign the individual tasks to members of the project. This information normally is managed within the work breakdown structue (WBS) [6] where each work package has got a unique identification number (WBS code). Thus it is possible to assign to each work product (i.e. an entity-relationship diagram) the corresponding WBS code that can be used by the software measurement evaluation software to identify the work product within the software process model. Additionally it makes sense within a company to establish parallel to the software process model a standard work breakdown structure in order to reuse the numbering system over all development projects.

Measurement of project progress

One interesting area of software measurement is the measurement of the project progress. The *International Function Point Users Group* reports about two possibilities to do this [5]:

- Measure progress in terms of the number of percentage of completed activities.
- Measure progress in terms of the number of percentage of completed product units.

In order to apply one of these approaches one has to know either the complete set of activities to be done or the total amount of work products to complete. Since the work products are administered within the SDE we can at least count the completed work products and the work products under work. The hierarchical structure of the process building blocks of the V-Modell XT together with the work breakdown structure allow it additionally to examine the working states of the product groups and products belonging to a certain process building block by querying the project repository within the configuration management system. Additionally one can think of plausibility checks regarding the size of these elements in order to assess the degree of processing of this process building block.

Measures for process chains

The V-Modell XT defines dependencies both between the process building blocks and between individual work products. We want to call this sequence of process building blocks and activities respectively a *process chain*. The sequence of the activities *analysing – designing – coding* is such a process chain for instance.

The process chains can be utilized to compare the amount of produced work products or the expended effort between the single steps of the chains. So a disproportion between a large effort for the design activity and an exceptional small effort for the coding activity would be remarkable. Since in this case one would be in doubt about the completeness of this activity we gain an additional plausibility check by means of this software measure.

4. Vision

In order to assist the project members at the application of software measures we want to implement a tool which should allow to define and apply software measures. In contrast to the contemporary SDE the user should not be limited to the off-the-shelf measures provided by the manufacturer of the SDE. The user should instead be able to define abstract measures for certain elements of the software development process defined by the software process model. Such a measure for instance could be the *coding productivity* in terms of the average amount of source code produced at one person-day. At run time the users then can decide on which elements within the SDE's repository this measure should be applied. The Metrics Builder determines from the work breakdown structure which work products have to be taken into account and computes the software measure.

The Metrics Builder should not only consider the current project in order to assess the accomplished mea-

surements. Instead it should enable the user to compare the current measurement results with results obtained by the application of the same measures on former projects. So one can compare for instance measures like the coding productivity. Therefore we suggest to use a consistent numbering system within the work breakdown structure.

In the previous section we have introduced the application of software measures on so called process chains. The Metrics Builder should also support this. For that purpose the user should be able to define and maintain distribution ratios of effort and time spent on the activities of several projects. Such effort and time distribution ratios are reported for instance by Barry Boehm [1]. This way the Metrics Builder can check whether the effort and time measures of the current project fit to this ratio scheme or not.

5. Prototypical implementation

General considerations

Currently the Metrics Builder is being implemented as a module for the SDE *Together ControlCenter 6.1* by Borland Software Corporation. Although this product will be discontinued in the near future it has especially for our purposes some advantages against the *Eclipse IDE*: First of all Together ControlCenter already contains a metrics framework which is passably documented. For Eclipse such a metrics plugin only exists as a commercial (closed source) plugin. And secondly the Eclipse IDE itself does not support UML diagrams. For this you again have to revert to commercial plugins like the *Together Edition for Eclipse*.

Current implementation status

The first prototype of the Metrics Builder uses the metrics framework of Together ControlCenter which is part of Together's *Quality Assurance Module*. This module contains already more then 50 measures (*Lines of Code*, *Cyclomatic Complexity*, *Comment Ratio*) implemented by the producer of Together ControlCenter. These measures are only applicable to the source code that is worked on within the IDE. A short documentation is included to encourage the user to implement his own measures.

Since Together mainly maintains the source code and the corresponding UML diagrams of a development project information about activities and roles have to be acquired somewhere else. One first source of information about the people involved in the project and their coding activities is the version control system (e.g. CVS, Subversion or commercial products like *IBM Rational ClearCase*). This can be queried about who has worked on a piece of source code and, utilizing some assumptions about the programmers' working modus, one can estimate the effort in person-hours, that has been spent on this source code artifact. As a more simple first step to get this information into the Metrics Builder the effort spent on each Java file has bin recorded in the Java file itself as JavaDoc comment.

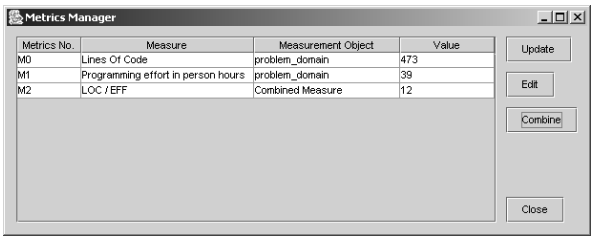

Figure 1. Productivity as combined measure of LOC and effort

A more elaborated version of the Metrics Builder will use the WBS code assigned to the software element to look up within the project management system the amount of working time that has been accounted for this work package.

A first look at the program

In Figure 1 three measures for a Java project are shown. They belong to the *Cash Sales* project that is shipped together with Together ControlCenter as a sample project. The measures *Lines of Code* and *Programming effort in person hours* have been applied on the Java package `problem_domain`. Then the combined productivity measure *LOC per person-hour* has been computed.

References

[1] B. W. Boehm. *Software Engineering Economics*. Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

[2] Bundesministerium des Innern (BMI). V-Modell XT, Version 1.01, 2004. http://www.v-modell-xt.de.

[3] N. E. Fenton and S. L. Pfleeger. *Software Metrics: A Rigorous and Practical Approach*. PWS Publishing Company, Boston, Massachusetts, second edition, 1998.

[4] R. B. Grady. *Practical Software Metrics For Project Management And Process Improvement*. Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[5] IFPUG, editor. *IT Measurement: Practical Advice from the Experts*. Addison-Wesley, Boston, Massachusetts, 2002.

[6] H. Kerzner. *Project Management — A Systems Approach to Planning, Scheduling and Controlling*. John Wiley & Sons, New York, seventh edition, 2001.