Integration of software measurement into the software development environment

BERNHARD DAUBNER
bernhard.daubner@uni-bayreuth.de

ANDREAS HENRICH
andreas.henrich@wiai.uni-bamberg.de

Although the benefit of software measurement is commonly accepted, in operational practice one is often afraid of accomplishing the effort for the collection of software measures, because it seems to be out of proportion to the benefit. Therefore, software measurement should be automated as far as possible, which can be achieved by the integration of software measurement into the software development environment.

This article describes two corresponding approaches on the basis of the development environments *IBM Rational Suite* and *SAP R/3*.

1 Tool support for software measurement

ISO/IEC 15939 [Int02] is the standard for software measurement published by the *International Organization for Standards* (ISO). This standard includes both a process model for software measurement and an information model. The later describes the information needs that lead to a choice of software measures and analysis techniques, whereas the information needs themselves are derived in turn from the goals of the software measurement. Thus, it is quite well known, which entities have to be measured in the context of the software development process to reveal the status of a project including possible risks.

However, we think there is a lack of appropriate tool support, that would automate the software measurement process as far as possible. Some measures, e.g. the number of open bug reports, are measurable quite easily in an automatic way. For instance within bigger projects one will certainly use a database for bug management and can therefore determine and observe very comfortably the number of bugs according to several categories.

The project progress in the sense of already implemented functionality of the product being developed can only be determined with a significant effort and only under certain circumstances. For this, it is necessary that one can trace back from the software component under development to the underlying functional requirement. From the development level of the related software components one can then deduce which functionalities are just under work, for which functions test cases exist and which software components have already passed the unit tests. So one gets a quite good picture about the implementation progress. At last, a requirement is classified *implemented*, if the corresponding software component has passed all functional tests and all integration tests.

In the following we will look at these possibilities with respect to their realization within the IBM Rational Suite and a possible integration into the SAP R/3 development environment.

2 Software measurement within the IBM Rational Suite

Within the IBM Rational Suite requirements are administered by *RequisitePro* within a relational database. Thereby the requirements can be linked with *Microsoft Word* texts or a UML diagram of the underlying use case. For the handling of test cases the Rational Suite includes the *TestManager*. This tool is able to generate test cases directly from requirements, UML models or Microsoft Excel spreadsheets.

Thus, within the IBM Rational Suite requirements are linked either directly or indirectly by means of a UML diagram to test cases. Additionally the requirements can be linked by means of a UML model with source code, which has been generated out of the UML diagrams.

IBM Rational Software has published a White Paper about the possibilities of software measurement that are provided by the IBM Rational Suite, especially if the tool *ProjectConsole* is used [GWI04]. Astonishingly, the software measures described within this paper are restricted to the mere enumeration of artefacts of the development process. For example one describes the number of requirements, changes in the number of requirements, the number of use cases with a certain status or things like that. Of course, one counts also the total number of lines of code and the number of bugs.

But there is nothing said about a possibility to directly determine the implementation progress from the software components under development. Corresponding software measures cannot be gathered until the programming activities have proceeded so far that particular use cases can successfully be tested against the developed software.

3 Software measurement with SAP R/3

3.1 Non object-oriented software development

Besides the object-oriented software projects there are also many development projects concerning legacy applications that either have to be maintained or enhanced. Here one might think for example of the huge host based accounting systems of banks and insurance companies that mainly are written in COBOL and do not have an object-oriented but at best a structured design. Besides there is a very large number of applications for SAP R/3, that are written in ABAP. Thereby ABAP has got an exceptional position, since with roots in the mainframe area it originally only supported structured programming, but has in the meanwhile been extended with object-oriented concepts.

Especially as far as SAP R/3 is concerned there are hardly any scientific reports about the application of software measures within the SAP development process. There is only some work in which the requirements management in the SAP R/3 area is analyzed [Dan03]. Thereby also measures for effort estimation und software reuse are inspected.

Therefore, it is of great interest which software measures are actually applicable within ABAP projects, how the data for the software measures can be collected, and how this can be integrated into the SAP software development environment.

Moreover there is the question, which development process is actually appropriate for such an environment. From the authors point of view one might doubt, whether a use case driven process like the *Rational Unified Process* (RUP) is perfectly adapted for the creation of SAP applications, that are less characterized by user interaction but by mere data transactions. Here possibly a more adapted process could be derived from a process framework like *OPEN* [GHSY97].

3.2 Software organization within the SAP R/3 development environment

The role of the software repository within the SAP R/3 development environment is taken by a relational database system, that belongs to the R/3 system. This database system not only contains the business data of the R/3 system but also programs, input forms, table definitions, and data types. To access this data independently of the applied database system a general and system independent interface to this dataset is provided by the *Data Dictionary* [Mat02].

Besides the Data Dictionary also the *Correction and Transport System*, which is centralized in the *Transport Organizer* tool, plays a vital role within the SAP R/3 development. Beneath the version control the Transport Organizer provides possibilities to transfer software changes to other SAP systems and especially from the test system to the production system.

The actual organization of the software developed with SAP R/3 is done with *packages*, that were called *development classes* within former SAP releases. A package is a container combining development objects that belong together logically. Additionally packages can include subpackages.

The logic of an SAP application resides either within ABAP programs and subprograms, within function modules or within methods of *ABAP Objects* classes. Thereby, the classes provide the highest and the subprograms the lowest level of data encapsulation [Kel02]. A great part of present-day SAP functionality is indeed implemented with function modules [KK01]. Therefore, we want to present the integration of software measurement by means of function modules. These imply a very strict interface logic and are organized within function groups, whereby reusability and data encapsulation are supported at least to a certain degree.

3.3 Traceability and monitoring of project progress

In contrast to the IBM Rational Suite there is no integrated tool support for requirements management, modelling, and implementation within the SAP R/3 development environment. The requirements management and modelling activities are accomplished outside of the R/3 system whereas the coding and testing is done within the SAP R/3 development environment. In so far, there is first of all no automatic traceability guaranteed between the requirements, the development model, and the implementation.

Since there are, however, appropriate possibilities for the source code organization given within the SAP development environment, we want to present an approach, how this mapping can be done manually. As we have stated in [DH03], the UML activity diagrams are also suitable to represent data flows as they are used within the structured programming and especially within the modelling of SAP applications. Thereby, data flow diagrams (DFD) feature the possibility to adequately model the functionality of programs like SAP applications. Moreover, one can derive effort estimations from the DFD already at a very early stage of the project.

We propose to map the UML packages within which the activity diagrams are organized to the packages of the SAP development environment, whereas being restricted to only one lower level within the package hierarchy. One can then either associate each activity diagram with an ABAP function group and the enclosed activities with a ABAP function module, or the packages are structured in a more object oriented way combining all functions associated with certain data into one function group. At the end it is crucial to have a mapping between the software model and the function groups and function modules. That means one must be able to derive from the name of an element of the software model the identifier of the associated function group and the function module respectively. If necessary the identifiers within SAP must be preceded with a prefix that is mandatory for customer developments.

By means of the Data Dictionary it can be checked which function groups and function modules have already been created within the SAP development environment, whereby at least a rudimental monitoring of the project progress can be realized. We want to show that this progress monitoring can be automated by determining the ABAP elements that belong to a software project via appropriate database requests. Because of their defined identifiers they can be assigned to the requirements modelled in the UML activity diagrams.

4 Progress measurement on an example application

Färber und Kirchner describe in [FK03] an example project, which deals with the implementation of an accounting application with ABAP on an SAP R/3 system. We want to use this example to explain our above ideas for progress measurement, because it is very accuratly described in [FK03] including business processes, architecture description and the resulting source code.

Within this example basically three functions have to be implemented that concern the warehouse and the accounting of a pharmaceutical company. Whenever a change in inventory within the warehouse takes place, a corresponding business transaction has to be stored in the database. These business transactions are from time to time transformed into accounting documents with respect to double entry bookkeeping. And finally an accounting list can be generated. Figure 1 gives an overview on this application.

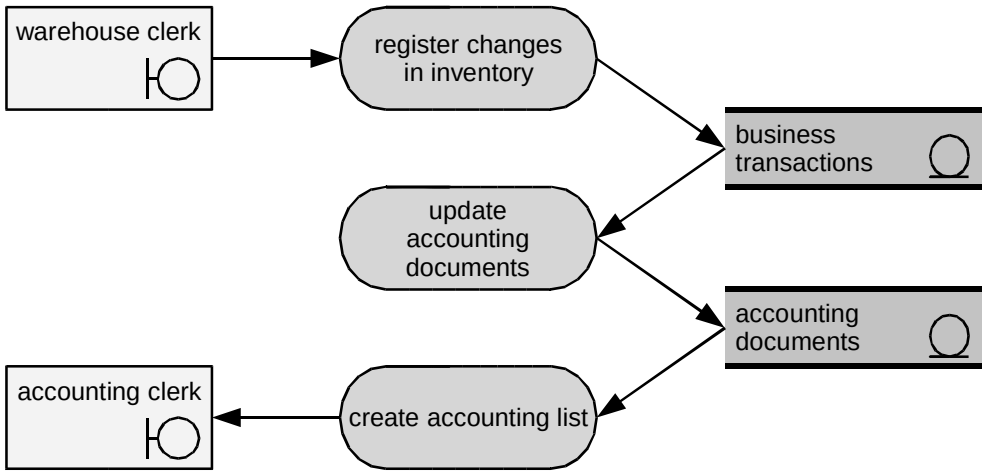

Figure 1: Accounting Application

Since the business transactions and the accounting documents are the two main elements of this application, it makes sense to pool them in two ABAP function groups with the identifiers `ZBTB00_OBJ_BTA` and `ZBTB00_OBJ_DOC`. Here `Z` is a valid prefix for customer developments within an SAP system, `BTB00` is an identifier for this application, and the authors of [FK03] use the identifier `OBJ` to characterize application logical functions in contrast to e.g. interface functions. Altogether, we get the SAP function modules listed in table 1.

Technically, a function group within the SAP system is nothing else than an ABAP program and a function module is an included subprogram. The main difference to normal ABAP programs is that function modules must not be edited with the standard editor but with the *Function Builder*. All function modules of the system are listed in the table `TFDIR` of the ABAP Data Dictionary. We now can query this table together with some other tables, that contain additional maintenance information, for all function modules belonging to our two function groups. Doing so we get the screenshot displayed in figure 2, where we can see that all above mentioned function modules at least already exist within our SAP system.

ZBTB00_OBJ_BTA	
zbtb00_obj_bta_edit	edit business transaction
zbtb00_obj_bta_save	store business transaction into DB
zbtb00_obj_bta_load	get business transaction from DB
zbtb00_obj_bta_mark_as_booked	mark BT as booked within DB
ZBTB00_OBJ_DOC	
zbtb00_obj_doc_book	create accounting document from BT
zbtb00_obj_doc_save	store accounting document into DB
zbtb00_obj_doc_load	get accounting document from DB

Table 1: Function Groups and Function Modules

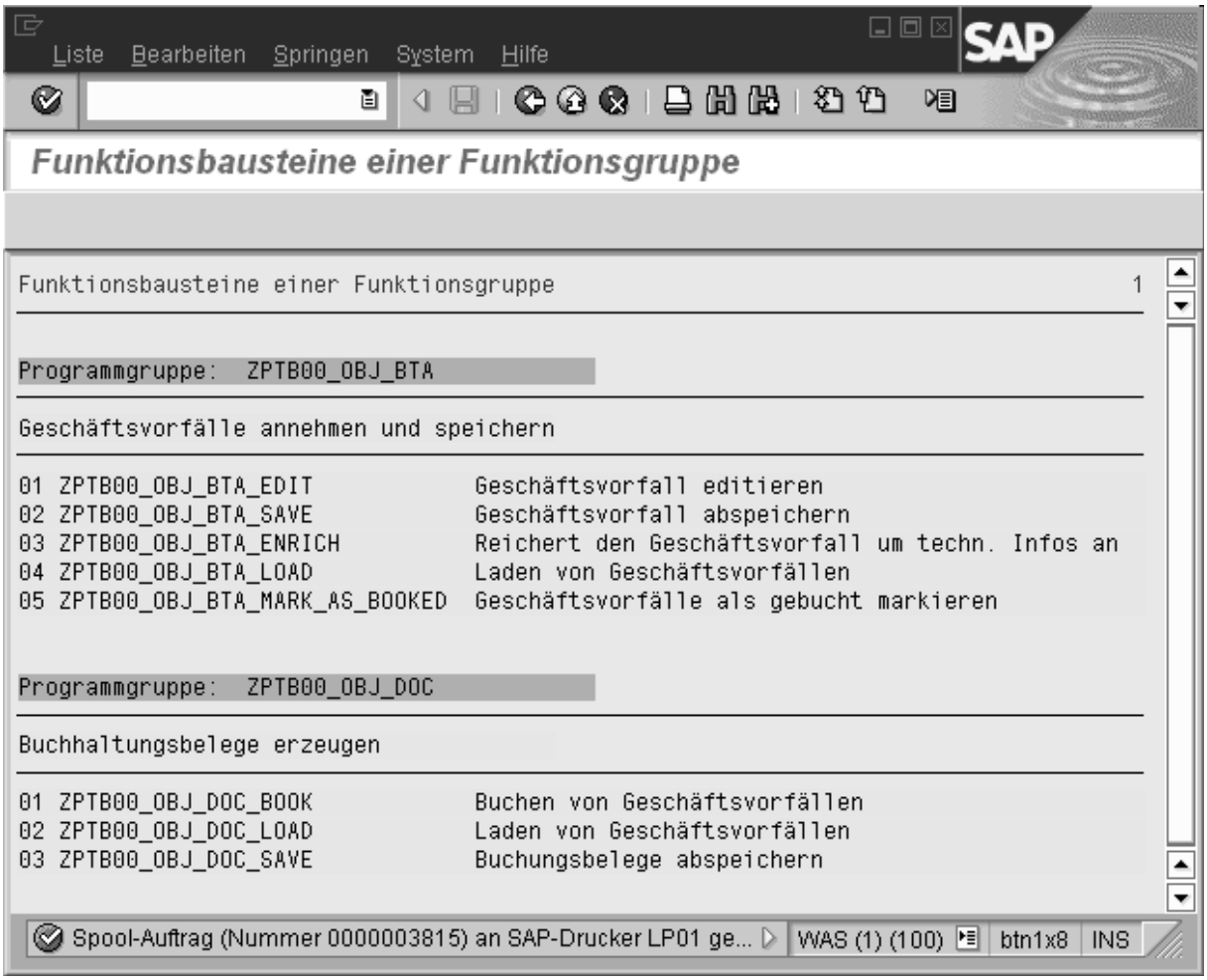

Figure 2: List of function modules as a means of progress management

What we have done hereby is to establish a link from the design model to the implementation by means of a strict naming convention. This offers a rudimental possibility to monitor the project progress, since one can check, which of the required functionality is already worked on. To provide complete traceability we also need a link from the requirements to the design model.

5 Further work to be done

So far we only know that someone has created the listed function modules in figure 2 within the SAP system. We do not know whether they work or even whether actually any programming work has been done. The next step could be to compute the sizes of code of these function modules that could possibly be compared with calculated Function Point measures.

We also have not yet considered, whether the corresponding entry masks to use this function modules have already been created. And we do not know, whether the database tables used by the function modules already exist.

The next step will be to provide a mechanism to derive these informations (definition of database tables, required forms) from the design model. Therefore, it is necessary that the modelling is done with an appropriate tool. Then this information must be read by an analysis software within the SAP system to check how far the corresponding ABAP elements have been already created. The elements can be found because of their predetermined identifiers as we have shown above.

And finally a testing tool should be integrated into this framework to assure that the required ABAP elements not only exist but also work. Here the newly developed tool *ABAP Unit* might be a promising approach.

References

[Dan03] Maya Daneva. Lessons Learnt from Five Years of Experience in ERP Requirements Engineering. In *11th IEEE International Requirements Engineering Conference (RE'03)*, page 45ff. IEEE, 2003.

[DH03] Bernhard Daubner and Andreas Henrich. Ein Plädoyer für Datenflussdiagramme aus der Sicht der Aufwandsschätzung und der agilen Softwareentwicklung. In Klaus R. Dittrich, Wolfgang König, Andreas Oberweis, Kai Rannenberg, and Wolfgang Wahlster, editors, *INFORMATIK 2003 - Innovative Informatikanwendungen, Band 1, Beiträge der 33. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 2003 in Frankfurt am Main*, volume 34 of *LNI*, pages 191–195. GI, 2003.

[FK03] Günther Färber and Julia Kirchner. *Praktischer Einstieg in ABAP Objects*. Galileo Press, Bonn, 2003.

[GHSY97] Ian Graham, Brian Henderson-Sellers, and Houman Younessi. *The OPEN Process Specification*. Addison-Wesley, Harlow (England), 1997.

[GWI04] Bill Gilbert, Dave West, and Doug T. Ishigaki. *IBM Rational Team Unifying Platform: IBM Rational ProjectConsole Sample Measures*. IBM Corporation, 2004.

[Int02] International Organization for Standardization. *ISO/IEC 15939:2002 – Software Engineering – Software Measurement Process*, 2002.

[Kel02] Rainer Kelch. *ABAP Objects – Ein Lehr- und Trainingsbuch für die klassische und objektorientierte Programmierung*. dpunkt-Verlag, Heidelberg, 2002.

[KK01] Horst Keller and Sascha Krüger. *ABAP Objects – Einführung in die SAP-Programmierung*. Galileo Press, Bonn, second edition, 2001.

[Mat02] Bernd Matzke. *ABAP – Die Programmiersprache des SAP-Systems R/3*. Addison-Wesley, München, fourth edition, 2002.