ЮБИЛЕЙНА НАУЧНА СЕСИЯ – 30 години ФМИ
ПУ "Паисий Хилендарски", Пловдив, 3-4.11.2000

COMPONENT-BASED REUSE APPROACHES, NECESSARY PROCESSES AND MTRICS FOR AN ASSESSMENT

Reiner Dumke, Andreas Schmietendorf

The SW-WiVe project performed by Deutsche Telekom in collaboration with the Otto-von-Guericke University provides a detailed analysis and offers strategies for software reuse within industrial software development that can be subjected to critical evaluation. Traditional evaluation approaches, such as reuse metrics, were critically studied and necessary processes for continuous reuse were developed for this purpose. In a further step, currently available, valid reuse metrics for the software development process were classified and lacking metrics-based evaluation approaches were identified. This paper gives a survey for the aspects of reuse, the necessary processes for it and its integration into a corresponding procedure model. Furthermore, the demands on a metrics supported reuse approach are represented.

1 Introduction

In areas of industrial production such as the automotive industry, a high degree of reuse of previously manufactured intermediate products is considered to be a key for a high level of productivity, short product supply times, low costs per manufactured product, and a high level of quality. The basis for such a procedure is a process consisting of a division of labor between "suppliers" and "final assembly," as well as standards for functional and qualitative properties of the intermediate products used in order to fulfill customer requirements for the final product. Software developers also want to adopt this process, which has been successfully applied in all engineering-based industries, and are attempting in particular to use new technologies, such as object or component orientation, in order to improve what has been a mostly unsatisfactory situation up to the present.

[Biggerstaff & Perlis 1989] provide the following generic definition of reuse: "The reuse of software is the renewed use of artifacts and collected knowledge arising from the development of a software system when developing a new software system, in order to reduce the expenditure for creating and maintaining this new system."

Another definition for software reuse as a whole is provided by [Ezran 1998]: "Software reuse is the systematic practice of developing software from a stock of building blocks, so that similarities in requirements and/or architecture between applications can be exploited to achieve substantial benefits in productivity, quality and business performance."

The following conclusions can be drawn from both definitions:

- Software reuse (referred to hereinafter simply as "reuse") must always be considered in relation to software development.
- Artifacts and/or assets include a variety of reusable components, such as requirements, models, and implementation codes.
- Objectives for reuse include increased productivity, higher quality, cost reduction, lower maintenance costs, and also shorter development times.

Important for a successful reuse are unambiguous management decisions with regard to organizational modifications and from it resultantly required ones staff resources.

2 Overview of Reuse Aspects

The "Fifth International Conference on Software Reuse 1998" classified the problem of reuse according to various aspects, and presented it schematically in a so-called "reuse diamond." A short description of the equally ranked aspects contained in this reuse diamond is presented below.

- **Strategy and Management** - This aspect involves such elements as the establishment of reuse strategies and management support, as well as the need for an organizational development. (frequently described as a reuse-centered organization).
- **People** - The literature contains various role models [Jacobson 1997], [Sodhi 1998], [Coulange 1998] that interact closely with the selected organizational form.
- **Assets** - The term "assets" describes all reusable products such as components, objects, requirements, analysis and design models, codes, documentation, etc.
- **Technology** - This refers primarily to software production environments and the repositories used to store necessary information on software development, as well as appropriate component databases containing reusable assets.
- **Process** - The traditional software development process provides insufficient support for reuse. Thus necessary roles and/or personnel requirements, for example, are not defined.
- **Measurement** - The use of software metrics serves primarily to provide an element of quantification in the entire reuse process.

Because the applicability of metrics is closely associated with process quality, which is also expressed, for example, within the CMM model for evaluating the maturity of software development, the SW-WiVe project focuses on the aspects of "Process" and "Measurement."

3 Processes of software reuse

3.1 Short description and interactions of processes

Diverse and partially quite different considerations of the necessary processes exist for the organisation, management and carrying out of software reuse. The following detailed representation (Fig. 1) proved to be very suitable within the project SW-WiVe:

Some explanations:

- *Application Family Engineering*: Development of an whole architecture for different software applications of an user domain,
- *Component System Engineering*: Implementation of component,
- *Application Engineering*: Development/realisation of software applications based on reused components,
- *Domain Engineering*: consider architecture-design, requirement analysis and software development for a family of applications.
  - Characterisation of the problem area,
  - Specification of the domain requirements,
  - Derivation of a domain model through analysis of similar systems,
  - Development of an reference architecture for the user domain.

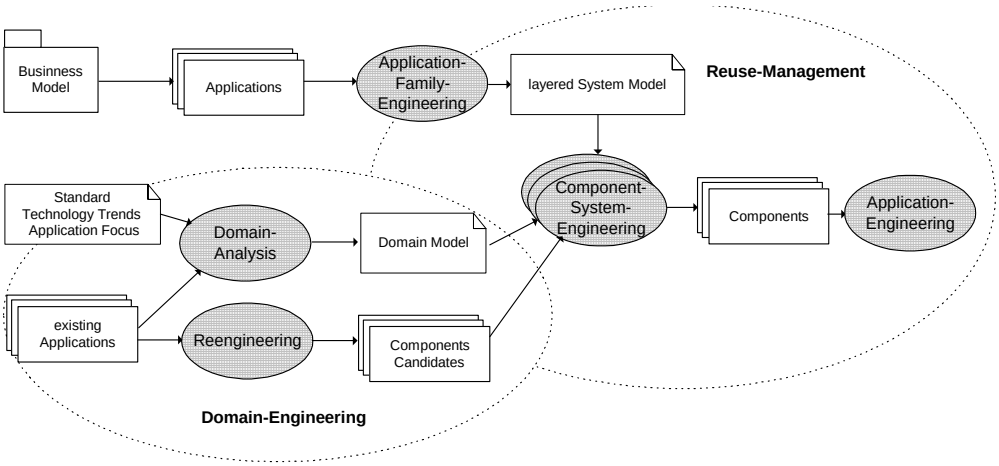

Fig. 1: Detailed representation of the reuse processes

Important aspects of reuse processes are the identification, the appropriation/administration of reusable components and an organisation to manage all these aspects. We propose the following processes for an successful reuse approach:

- **Identification support** (task within the domain engineering)
  - All developed components will be registered in a database (repository)
  - An internet-based application is employed for a simple access
- **Appropriation of reusable components** (task of application family engineering and component system engineering)
  - Preparation of suitable components for a reuse (e.g. description, property-sets)
  - Appropriation of components with a commercial software distribution system
- **Administration** (task of the reuse management)
  - Quality assurance and configurations management of reusable components
  - Procedure for inserting, modification and deletion of reusable components
- **Organisation** (task of the reuse management)
  - Processing of criteria for the classification and structuring the component catalogue
  - Processing of a type-specific description model for every reusable component

3.2 Integration of the software-development- and reuse-process

Reuse can be found in all phases/stages of the SW development. Therefore it is important to clarify:

- as the process of reuse and the process of software development are coupled with each other and
- where in the phases of software development those different artifacts reuse are be assigned.

Figure 2 represents the answers to these questions.

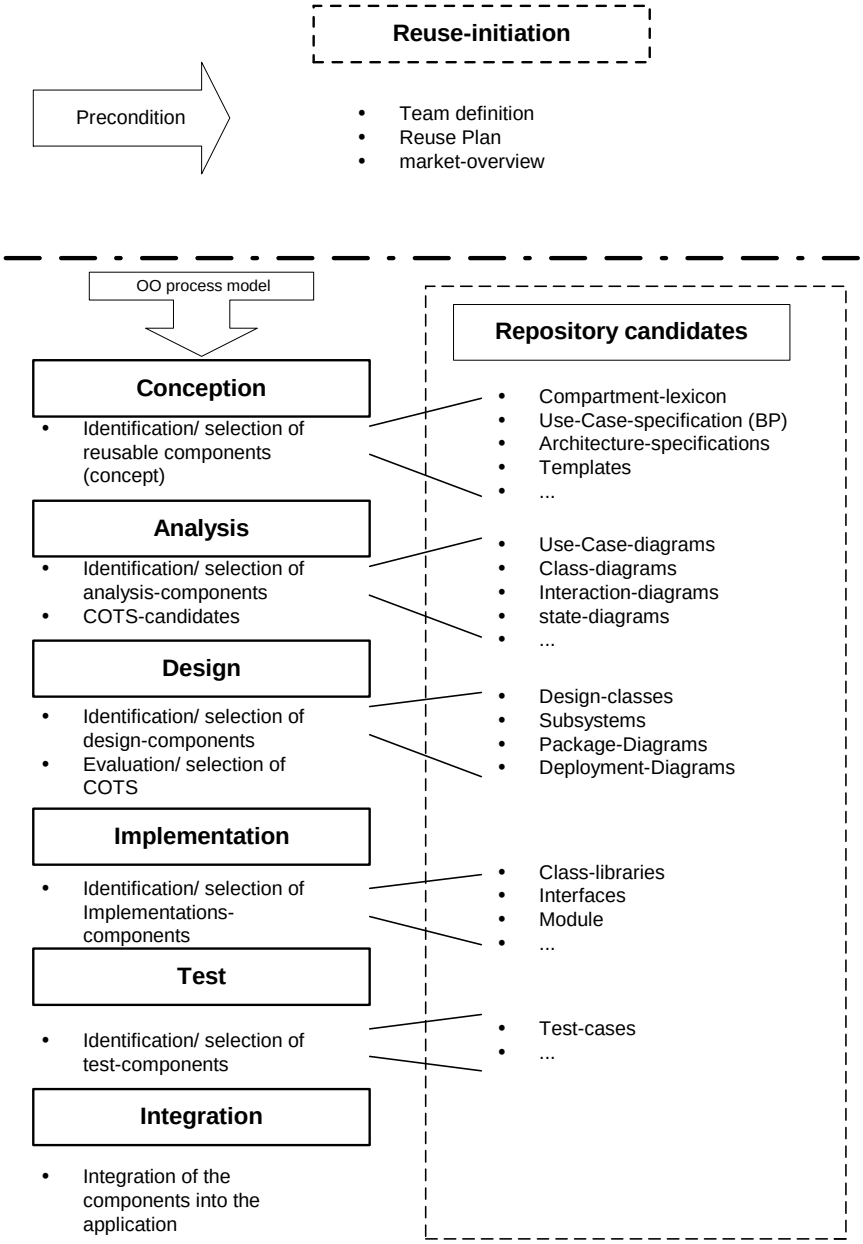

Fig. 2: Integration of the software-development- and reuse-process

The relation between the activities of reuse and the phases defined in the procedure model for object oriented software development constituted predominant itself through that so-called "concrete" reused components.

These "concrete" components are the elements resulted during the carrying out of the corresponding phase, how for example use cases, classes/objects, communications, interfaces, subsystems and so forth. However, all diagram models which can be derived from the elements mentioned above (use case diagrams, class diagrams, communication diagrams, ...) are also required for that.

A second kind of reusable artifacts are the pattern as abstracted experiece-based knowledge to the solution of specific problems. Referring to the individual phases, the following types of pattern can be used:

- **Architecture Patterns** (phase conception) they express elementary structure criteria of an software system, like e.g. 2-tier or n-tier client/server-architectures, *Model-View-Controller* – Paradigm (MVC-Pattern) and so forth.
- **Analysis Patterns** (phase analyse) used for realisation of requirements. With its help, parts of the domain are specified.
- **Design Patterns** (phase design) they describe problems recurring constantly and explain its solutions, e.g. generation patterns, structuring patterns or behaviour patterns.
- **Implementation Pattern** (phase implementation) describe as implementation specific problems are to be solved in a concrete program language.

A third group of reusable artifacts are the so-called Templates. In dependence of the phase of software development, different templates are used.

- **Conception**: Project Definition Report Templates, Project Phase Issues Template, Risk Analysis Template, Work Plan Template, Project Administration Plan Template,
- **Analysis**: Analysis Report Template, Project Summary Template, Work Plan Template, Business Requirements Test Plan Template,
- **Design**: Architecture Template, Design Report Template, Technical Design Document Template,
- **Implementation**: Installation Plan Template, Back-Out Plan Template, Implementation Report Template, Defect Summary Report.

Test cases and test scenarios represent important reusable components during the test phase.

4 Software-metrics and reuse

4.1 Strategy for an application of metrics

For the efficient application of Software Metrics [Dumke 1998] suggests a framework for the establishment of measurement programs. This framework uses the idea of [Fenton 1991] to group the measurement artefacts into product, process and resource.

The framework mentioned above consists of a CAME[1] strategy, a CAME[2] framework, and the CAME[3] tools for the recording and processing of metrics. While the CAME strategy refers to aspects such as the need for the existence of a group to promote the implementation of soft-

[1] C - Community, A - Acceptance, M - Motivation, E - Engagement
[2] C - Choice, A - Adjustment, M - Migration, E - Efficiency
[3] C - Computer, A - Assisted, M - Measurement, E - Evaluation

ware metrics and the need for management decisions, the CAME framework relates, for example, to the selection process of metrics, the analysis of scale properties of these metrics, the degree of coverage reached, and the efficiency of metrics usage by means of tools support. Because Deutsche Telekom has already made strategic decisions regarding the use of software metrics, the analysis concentrates on the selection of metrics (CAME framework) and the determination of an overall degree of coverage for available reuse metrics.

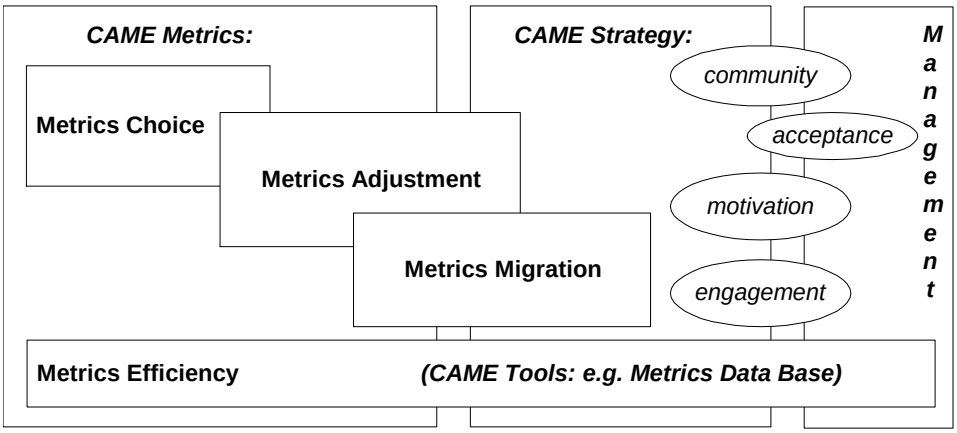

Abbildung 3: **Überblick zum CAME-Framework**

4.2 Requirements of a metrics-supported reuse approach

Based on the analyses performed and the general metrics program framework developed by the Otto-von-Guericke University at Magdeburg, the following general requirements were established for a software reuse that is quantified and thus can be evaluated:

1. In software reuse, the already existing reuse metrics can also be used, as well as the metrics valid for other paradigms, in order to make them easier to quantify and thus evaluate.
2. Selected metrics should be relatively scaled to the extent possible.
3. Above all, the selection of metrics must include every area (i.e. the products, the process, and the resources) in the evaluation.
4. The objective for the selected metrics must also be to achieve the highest possible level of automation. This also includes “traditional” measurement tools. The following table describes the current situation:
5. The measured values themselves must be integrated into metrics databases. This also applies for access statistics of individual reused components.
6. In addition to the predominately cost-oriented metrics, the aspects of quality improvement or influence must also be considered. That is valid for such questions how
7. Particular attention must be paid to the motivation for reuse. For this is to investigate

An efficient use of metrics is however of the analysis of the current software development situation and the so recognizable success-promises starting-points dependent.

4.3 Evaluating Reuse Maturity

Using the "Capability Maturity Model (CMM)" proposed by the SEI[4], which evaluates the maturity of a software development in five levels, [Sodhi 1998] proposed a corresponding "Software Reuse Maturity Model." The starting point for considerations was the insufficient evaluation of necessary details in the process for software reuse within the CMM. This evaluation model contains the following five levels. The achievement of a level is determined by means of a questionnaire.

- **Level 1 -** Informal practices
- **Level 2 -** Formal software reuse
- **Level 3 -** Implementation of formal reuse
- **Level 4 -** Management return on investment ROI[5]
- **Level 5 -** Optimization

In summarizing the evaluation possibilities studied, the SW-WiVe project proposed a variant for evaluating software reuse maturity that is adapted to Deutsche Telekom's requirements. This variant uses the following three levels:

Reuse initiatives: This level basically implies the contents of Level 2, and considers the aspects of reuse separately from one another for the most part. Typical here are the implementation of prototypical reuse databases with more technical assets, but without defining the processes necessary for effective use and without a strategic decision by management on a procedure for reuse.

Assessment-based reuse: This level basically implies the contents of Level 3. The introduction of valid metrics related to reuse means that processes, resources, and the product (assets) can be evaluated. The prototypical reuse databases that were implemented are developed further into comprehensive approaches, and are thus available to everyone involved in the software development. A reuse-centered organizational development has been established in order to develop domain-specific components, which are already recognized as standard.

Controlled reuse Process: This level basically implies the contents of Level 4, i.e. a metrics-supported reuse approach is established in order to clearly quantify the value added by reuse. In addition, the necessary organizational development has progressed to the point that centers exist for actual component creation, management of available components, and actual application development.

4.4 Metrics in the process of the software-reuse

Selection of the metrics and statement of the attributes (Choice/Adjustment) corresponding the GQM-method (Goal Question Metric). With it a definition of the goals is presupposed.

[4] Software Engineering Institute
[5] ROI Return on Investment

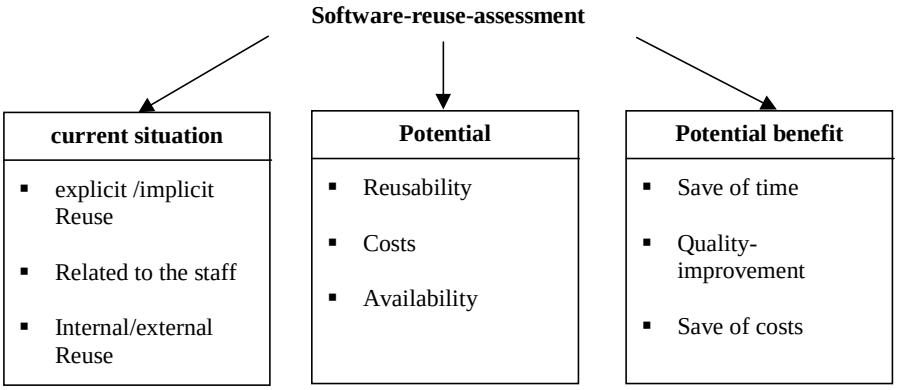

Fig. 4: Initial stage for the metrics choice

The goals already imply the questions and led immediate to the metrics. The concrete alignment of the metrics is however regarding product, processes or resources to plan. We want to restrict ourselves first of all on the **product**.

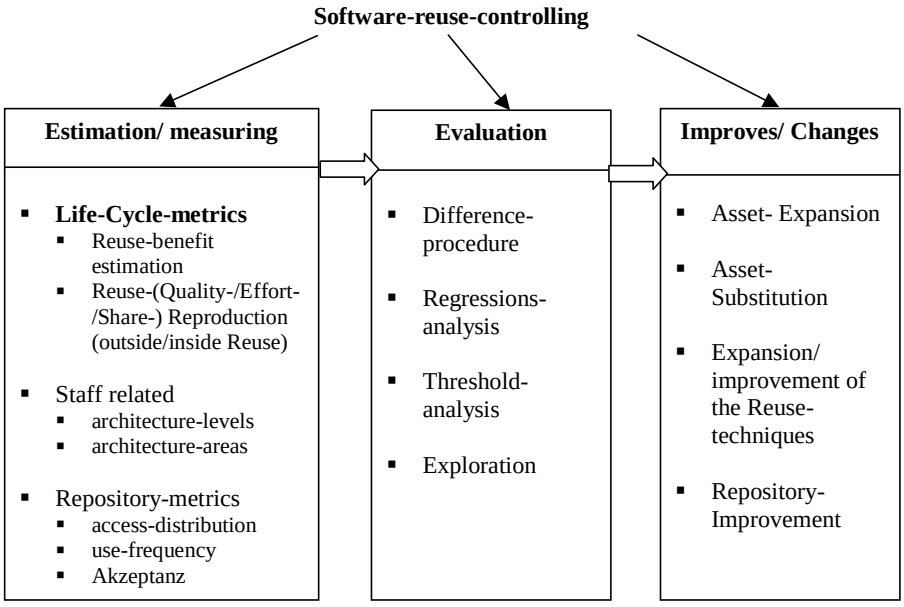

Fig. 5: Supplement of the metrics with life-cycle- and architecture-assessments

5 Conclusion and Outlook

The SW-WiVe project paid particular attention to the specified success factors for software reuse, and shows how one part of the problem can be approached and/or how a solution could be created. In addition to the studies presented here regarding the use of metrics and necessary reuse processes, the project also defined a proposal for a generic description of assets and an initial approach for formalizing this description through the use of metrics. The project also

made a practical study of reused components already employed by specific companies, and identified features that are closely associated with a successful reuse of these components.

REFERENCES

[Biggerstaff & Perlis 1989] Biggerstaff, T. and Perlis, A.: Software Reusability, Band I and II in the Frontier Series. ACM Press, 1989

[Coulange 1998] Coulange B.: Software Reuse, Springer-Verlag, London 1998

[Dumke 1998] Dumke, R., Foltin, E., Winkler, A.S.: A Framework for Object-Oriented Software Measurement and Evaluation. Proceedings of the IASTED Conference on Software Engineering, Oct. 28-31, 1998, Las Vegas, S. 129 - 132

[Ezran 1998] Ezran, M., Morisio, M., Tully, C.: Practical Software Reuse: The essential Guide. Paris: Freelife Publ., 1998

[Fenton 1991] Fenton, N.: Software Metrics – A Rigorous Approach. Chapmann & Hall Inc., London 1991

[Jacobson 1997] Jacobson, I., Griss, M., Jonsson, P.: Software Reuse (Architecture, Process and Organization for Business Success), Reading/MA: Addison-Wesley 1997

[Sodhi 1998] Sodhi, J.; Sodhi, P.: Software Reuse. Domain Analysis and Design Processes. New York ...: Mc Graw-Hill 1998

Prof. Reiner Dumke
Otto-von-Guericke-Universität Magdeburg
Fakultät Informatik, Institut für verteilte Systeme
Postfach 41 20, D-39016 Magdeburg
EMail: dumke@ivs.cs.uni-magdeburg.de

Dipl.Ing. Andreas Schmietendorf
T-Nova Deutsche Telekom Innovationsgesellschaft mbH
Entwicklungszentrum Berlin
Wittestraße 30N, D-13476 Berlin
EMail: A.Schmietendorf@telekom.de