

# Design Decision Support for Evolvability and Variability

## Extended Abstract

Matthias Riebisch and Stephan Bode  
Ilmenau University of Technology  
P.O. Box 10 05 65, 98684 Ilmenau, Germany  
{matthias.riebisch, stephan.bode}@tu-ilmenau.de

**Introduction** Business critical software systems have to be maintained for a long time in order to conserve their business value and for the constant provision of business services. However, frequent changes due to enhancements, business process optimization, or technological improvements have to be performed. As a consequence, evolvability and variability of software systems constitute important quality goals for business success.

Architectural design or redesign and the proper consideration of quality goals is a complex task. Therefore, there is a need for design decision support for these goals.

Recent architectural design methods and approaches, for example QASAR, Siemens' 4 Views, or Attribute Driven Design (ADD), consider quality goals such as flexibility, or changeability. However, regarding evolvability and variability a general methodical support for architectural design decisions is rare.

In our approach we provide guidance and a procedure for architectural decision-making regarding evolvability and variability. We explain how we integrate this in the architectural analysis and synthesis phases. With a goal-oriented procedure we decompose the quality goals related to change and reduce the damage to the architecture, the so-called architectural decay, due to a reduced change impact.

**Evolvability and Variability** Although methodical support for evolvability is rare and evolvability is not yet considered in a standard quality model, such as ISO 9126, there are several definitions for it. We base our definition on Breivold et al. [3] and Rowe et al. [6]: *Evolvability is the ability of a software system throughout its lifespan to accommodate to changes and enhancements in requirements and technologies, that influence the system's architectural structure, with the least possible cost while maintaining the architectural integrity.*

Since different aspects of a software system's evolution are determined by effort we use it for evaluating the system. This effort can be expressed by several subcharacteristics of evolvability. Based on the work of Breivold et al. [3] we decompose evolvability into the following subcharacteristics:

- Analyzability,
- Changeability, which aggregates extensibility, variability, and portability,

- Reusability,
- Testability,
- Traceability,
- Compliance to standards, and
- Additional process qualities.

Hence, variability is one of the subcharacteristics of evolvability. It describes *the capability of a software system or artifact to be efficiently extended, changed, customized, or configured for use in a particular context by using preconfigured variation points.*

Because of the influence on the effort of architectural design and redesign we relate the subcharacteristics to properties of good architectural design such as abstraction, modularity, encapsulation, separation of concerns, correctness, or conceptual integrity.

Further on, these subcharacteristics and design properties can be mapped to architectural solutions. This is a concept that we call the Goal Solution Scheme (GSS) [1]. Figure 1 shows a decomposition down to properties for evolvability.

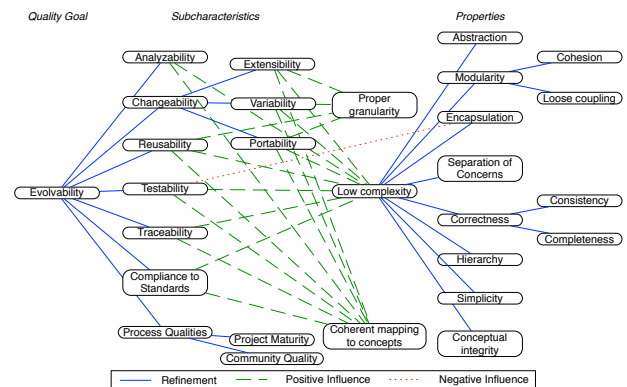


Figure 1: Decomposition of evolvability

The GSS maps quality goals to subcharacteristics, to design properties, and further to candidate solutions. In this way it represents possible decompositions of quality goals and related decisions during the design process, supports different phases as analysis and synthesis, and provides guidance for design activities. Besides, the scheme classifies candidate solutions according to their impact on quality goals, and therefore, supports architectural decision-making.

**Architectural Analysis** For design and redesign of an architecture different activities must be performed, which can be grouped to phases like archi-

tectural analysis and synthesis [4].

During analysis in reengineering we have to evaluate an existing software system regarding its evolvability. The goal is to determine weaknesses and hot spots for changes. Therefore, the analysis of the architectural dependencies is an important issue to reduce the change impact, and thus, to enhance the evolvability of the system.

For this dependency analysis a standardized set of dependency types is needed. Further, explicit dependencies should be established in models. Then, we can use rules to find further dependencies.

If all relevant model dependencies are known then the quality properties of an architecture can be evaluated regarding quality goals such as evolvability. For this a calculation of an impact factor of the properties on the goals is performed. Using the Goal Solution Scheme a calculation scheme can be derived and metrics can be developed. Such an evaluation was already partly performed in an industrial case study [2].

After a system's evaluation refactoring means or architectural solutions as patterns can be suggested in the architectural synthesis phase to support decision-making.

**Architectural Synthesis** During synthesis appropriate means have to be selected for (1) performing changes or (2) introducing new solutions to an architecture. One objective is to reduce the possible damage to the architecture due to frequent changes. Therefore, the change impact should be minimized and the most suitable solutions for evolvability should be applied.

After evaluating an existing system in the analysis phase candidate solutions must be developed and evaluated. The evaluation can be performed using a calculation derived from the Goal Solution Scheme and by predicting the quality impact after the implementation of the change. For prediction and evaluation, the results of a Ph.D. project [7] can be applied.

Knowing the calculated impact on the quality goals we can propose candidate solutions. This can be supported with a catalog of refactorings and other architectural solutions as e.g. patterns.

Figure 2 shows the procedure of the architectural decision-making for selection of candidate solutions.

**Development Process** Even if not addressed by the definition, software evolvability is to a large extent influenced by process characteristics as simplicity, project maturity, agility, or model-based or model-driven procedures [5]. Furthermore, the motivation and the skills of the developers are important. The software process has to provide adequate means for quality assessment to preserve architectural quality and to prevent architectural decay. The catalog for architectural solutions is used to accumulate architectural elements such as building blocks and tools to

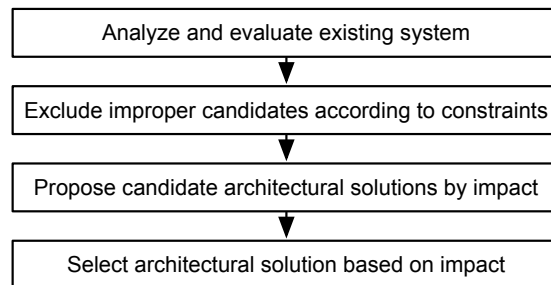


Figure 2: Procedure of decision-making

facilitate their reuse within the goal-driven decision-making procedure mentioned above.

**Conclusion** This contribution presents our concepts for the support of architectural decision-making regarding evolvability and variability. We provide guidance and a procedure with several activities such as decomposing quality goals, explicitly modeling dependencies, and evaluating a system with metrics. Moreover, we propose the establishment and comparison of candidate architectural solutions, which are evaluated according to their impact on evolvability based on a calculation scheme. The selection of candidate solutions is supported by a catalog. The whole decision-making procedure is integrated into a process including phases as architectural analysis and synthesis.

For an evaluation of the approach an industrial case study is intended to improve the scalability and practicability of the procedure during architectural design.

## References

- [1] S. Bode, A. Fischer, W. Kühnhauser, and M. Riebisch. Software architectural design meets security engineering. In *Proc. ECBS'09*, pages 109–118. IEEE, 2009.
- [2] R. Brcina, S. Bode, and M. Riebisch. Optimization process for maintaining evolvability during software evolution. In *Proc. ECBS'09*, pages 196–205. IEEE, 2009.
- [3] H. P. Breivold, I. Crnkovic, and P. Eriksson. Evaluating software evolvability. In *Proc. SERPS'07*, number 2007:02 in Software Engineering and Management, pages 96–103, IT University Göteborg, Sweden, 2007.
- [4] C. Hofmeister, P. Kruchten, R. L. Nord, H. Obbink, A. Ran, and P. America. A general model of software architecture design derived from five industrial approaches. *JSS*, 80(1):106–126, Jan 2007.
- [5] M. Riebisch and S. Bode. Software-Evolvability (in German). *Informatik-Spektrum*, 32(4):339–343, 2009.
- [6] D. Rowe, J. Leaney, and D. Lowe. Defining systems architecture evolvability - a taxonomy of change. In *Proc. ECBS'98*, pages 45–52. IEEE, 1998.
- [7] S. Wohlfarth. *A Process of Rational Decision-Making for Architectural Decisions (in German: Entwicklung eines rationalen Entscheidungsprozesses für Architekturentscheidungen)*. PhD thesis, TU Ilmenau, 2008.