# Domain-spanning Maintainability Analysis for Software-intensive Systems

Kiana Rostami
Institute for Program Structures and Data Organization
Karlsruhe Institute of Technology (KIT)
rostami@kit.edu

## Abstract

Long-living software systems undergo changes during their life cycle, due to modifications of the systems themself or in their environment. Software-intensive systems such as manufacturing systems consist of heterogeneous components involving electrical, mechanical and software artifacts. A Change to one of these components may result in changes in another component, which may not be modeled at all. Thus, change propagation analysis of such heterogeneous systems is inaccurate and high cost- and time-consuming due to change impact on unknown artifacts. We address this issue in design phase by considering the system design as the key artifact for change impact analysis. Therefore, we propose to tailor existing meta-models to reflect the heterogeneity of such systems. Specifying perspectives for heterogeneous components helps developer to identify unknown artifacts, which increases the prediction accuracy.

## 1 Challenges

Longevity is one of the key quality criteria for software-intensive systems such as information and manufacturing systems. These systems continuously undergo changes during their life cycle, due to modifications in requirements, technology, usage profile, architecture, and environment. Software-intensive systems comprise a set of heterogonous components consisting of electrical, mechanical, or software artifacts. Thus, artifacts may be neglected due to heterogeneity, when analyzing the change impact. This results in inaccurate effort estimation. Furthermore, there are mutual dependencies between software-intensive systems and their technical processes (automated by manufacturing systems) and business processes (automated by software systems). In addition to heterogonous components, these related processes have to be considered in order to identify the affected artifacts by a change request. For example, adding a new service to a software system may affect the business process, as the new service may be applied to automate certain activities of the process. Thus, focusing on software artifacts only is not sufficient when analyzing the change propagation and effort estimation during the evolution of software-intensive systems.

Currently, only few meta-models exist, which show dependencies between heterogonous components in software-intensive systems. Thus, the heterogeneity of such systems and the lack of meta-models make the analysis of change propagation through these systems difficult. Considering the dependencies between processes and systems, this issue becomes even harder.

Existing approaches to change propagation analysis are limited to software systems. These approaches analyze the maintainability from organizational (e.g., resource and personal management) and/or technical perspective (e.g., design or implementation) in order to identify affected artifacts. However, applying the existing approaches to manufacturing systems face the challenge of identifying appropriate perspectives for heterogeneous systems involving mechanical or electrical components. As a consequence, both maintainability prediction and change propagation analysis of such heterogeneous systems are often inaccurate and high cost- and time-consuming due to change impact on unknown artifacts.

## 2 Related Work

Related work to change impact propagation and effort estimation can be divided into four categories:

(i) **Task-based project planning approaches** involve either a hierarchically decomposition of tasks into sub-tasks (e.g., Hierarchical Task Analysis (HTA) [KA03]) and/or the use of different cost estimating approaches (e.g., Comprehensive Cost Model (COCOMO) II [BABea00]). However, using coarse-grained system design models leads to poor predictive capability.

(ii) **Architecture-based project planning approaches** (e.g., Carbon [CFR+12] or Architecture-Centred Software Project Planning (ACSPP) [PB01]) combine software architecture with the corresponding project plan. However, these approaches do not offer support for change effort estimation based on a given system design, nor automated change impact analysis, nor derivation of task lists required.

(iii) **Approaches to architecture-based software evolution** (e.g., Naab's approach [Naa12] or the approach of Garlan et al. [GBSC09]) analyze software evolution by considering software architecture. However, these approaches do not support effort estimation, nor automated change impact analysis.

(iv) **Approaches to scenario-based architecture analysis** (e.g., Software Architecture Analysis Method (SAAM) [CKK02], Architecture-Level Modifiability Analysis (ALMA) [BLBvV04], Architecture-Centric Project Management (ACPM) [PB01], or Karlsruhe Architectural Maintainability Prediction (KAMP) [SR09]) assess the quality of software architectures using scenarios. KAMP is a tool-supported approach, which derives task lists from the organizational and technical perspective. KAMP can be applied to derive task lists to address performance issues [HH14] and to plan software project [HH13].

However, the approaches associated with the latter three categories only support software architecture. Thus, they do not consider components in other domains, such as electronic components.

## 3 Solution Idea

Software architecture is the main artifact in a software system, which reflects important design decisions on structural, technological and organizational aspects in all stages of the software life cycle. There are already various approaches to meta-model the software architecture (e.g., PCM [BKR09]). System design in manufacturing systems can be seen as an analogy for software architecture. Both reflect the composition of the system by components and interfaces. As there are similar questions about maintainability estimation in the domain of software systems and manufacturing systems at design-time, we propose to adapt existing approaches to software change propagation (e.g., [SR09]) to the domain of manufacturing systems. To this end, we build upon the approaches to change propagation, which consider both the organizational and technical perspective based on the software architecture model. Currently, existing approaches model the control software in manufacturing systems. Building upon Systems Modeling Language (SysML) [OMG12], the next step is to meta-model the entire system design of manufacturing systems involving electrical and mechanical components. To this purpose, we have to identify the appropriate abstraction layer of system elements. Identifying the further perspective similar to those identified for software systems is the key to estimating change propagation in heterogeneous systems. Knowing the perspectives will improve our knowledge of affected artifacts by a change.

In order to include associated processes in the change impact analysis, we aim to meta-model the business processes and technical processes. KAMP is the only approach, which uses a formal architecture meta-model. It builds upon the following scenario: (i) **Preparation phase**: The user manually models the software system and annotates the architecture model to show the additional information (e.g., test cases, build configuration, or deployment context). To this end, the user applies an extended PCM. (ii) **Change request phase**: After the user alters the architecture model to show the change request, KAMP automatically calculates a list of tasks required to implement the change request. However, according to related work KAMP focuses only on software systems. In order to analyze the change propagation in manufacturing systems, we aim to extend KAMP's idea to include the meta-models and processes mentioned above.

After determining the meta-models of system design and related processes, we propose to extend the above scenario to a roadmap of change requests, which occur with a certain probability. Roadmaps are already used in the industry to prioritize future change scenarios. Thus, each change request of the roadmap leads to a maintenance task list. The purpose of the resulting task list is to assess the maintainability of the software architecture or the system design. Evolving different variants of the initial architecture over time results in a maintainability simulator. The input to the maintainability simulator is also an initial architecture and a certain change roadmap. The resulting task lists would support developers to estimate the change effort. Furthermore, comparing various resulting architectures helps developer to identify the most maintainable one and to estimate thus the maintenance of the initial architecture at design time.

Future work includes (i) developing and extending meta-models to reflect software systems and manufacturing systems, associated processes and their interrelations, and (ii) specifying perspectives of heterogeneous system. Based on the meta-models and perspectives we extend KAMP to consider heterogeneous system components and to implement the aforementioned maintainability simulator.

## References


[BABea00] Barry W. Boehm, Chris Abts, A. Winsor Brown, and et al. *Software Cost Estimation with Cocomo II with Cdrom*. Prentice Hall, 2000.

[BKR09] Steffen Becker, Heiko Koziolek, and Ralf Reussner. The palladio component model for model-driven performance prediction. *J. Syst. Softw.*, 82(1):3–22, January 2009.

[BLBvV04] PerOlof Bengtsson, Nico Lassing, Jan Bosch, and Hans van Vliet. Architecture-level modifiability analysis (ALMA). *J. Syst. Softw.*, 69(1-2):129–147, January 2004.

[CFR+12] Ralf Carbon, Kaiserslautern Fraunhofer, Dieter Rombach, Peter Liggesmeyer, and Frank Bomarius. *Architecture-Centric Software Producibility Analysis*. Fraunhofer IRB Verlag, 2012.

[CKK02] P. Clements, R. Kazman, and M. Klein. *Evaluating Software Architectures: Methods and Case Studies*. SEI series in software engineering. Addison-Wesley, 2002.

[GBSC09] David Garlan, Jeffrey M. Barnes, Bradley R. Schmerl, and Orieta Celiku. Evolution styles: Foundations and tool support for software architecture evolution. In *WICSA/ECSA*, pages 131–140. IEEE, 2009.

[HH13] Oliver Hummel and Robert Heinrich. Towards automated software project planning - extending palladio for the simulation of software processes. In *KPDAYS*, pages 20–29, 2013.

[HH14] Christoph Heger and Robert Heinrich. Deriving work plans for solving performance and scalability problems. In *EPEW*, pages 104–118, 2014.

[KA03] B. Kirwan and L.K. Ainsworth. *A Guide To Task Analysis: The Task Analysis Working Group*. Taylor & Francis, 2003.

[Naa12] Matthias Naab. *Enhancing architecture design methods for improved flexibility in long-living information systems*. PhD thesis, Fraunhofer IESE, 2012.

[OMG12] OMG Object Management Group. OMG Systems Modeling Language (OMG SysML), Version 1.3, 2012.

[PB01] Daniel J. Paulish and Len Bass. *Architecture-Centric Software Project Management: A Practical Guide*. Addison-Wesley Longman Publishing Co., Inc., 2001.

[SR09] Johannes Stammel and Ralf Reussner. Kamp: Karlsruhe architectural maintainability prediction. In *Proc. of 1st. L2S2 Workshop*, pages 87–98, 2009.