Evolving Delta-Oriented Software
Product Line Architectures

Arne Haber¹, Holger Rendel¹,
Bernhard Rumpe¹, and Ina Schaefer²

¹ Software Engineering, RWTH Aachen University, Germany
² Institute for Software Systems Engineering, TU Braunschweig, Germany

Abstract. Diversity is prevalent in modern software systems. Several system variants exist at the same time in order to adapt to changing user requirements. Additionally, software systems evolve over time in order to adjust to unanticipated changes in their application environment. In modern software development, software architecture modeling is an important means to deal with system complexity by architectural decomposition. This leads to the need of architectural description languages that can represent spatial and temporal variability. In this paper, we present delta modeling of software architectures as a uniform modeling formalism for architectural variability in space and in time. In order to avoid degeneration of the product line model under system evolution, we present refactoring techniques to maintain and improve the quality of the variability model. Using a running example from the automotive domain, we evaluate our approach by carrying out a case study that compares delta modeling with annotative variability modeling.

1 Introduction

Modern software systems simultaneously exist in many different variants in order to adapt to changing user requirements or application contexts. Software product line engineering [32] aims at developing a family of systems by managed reuse in order to decrease time to market and to improve quality. In addition to this variability in space, software systems are extremely long-lived and have to evolve over time in order to maintain, improve or update their functionality. This unanticipated variability in time [26] changes the system design, structure, and behavior in an unexpected manner, e.g., for adapting it to new customer requirements or technological conditions. Evolution of software systems needs to be managed, and gets particularly difficult if a family of systems is considered.

The design of the software architecture plays an essential role in software development [27,25]. The architecture allows decomposing a complex system into smaller hierarchically structured components. These can be developed independently. The change frequency of architectural descriptions is lower than the changes on the implementation level, where often bugs etc. need to be fixed. However, changes in the architecture have a wide range impact on the overall
system such that architectural changes have to be planned, modeled and analyzed to ensure that the system quality is maintained despite of the changes. This is particularly complex for software product line architectures.

Most current ADLs [25] do not support the explicit representation of architectural change. The predominantly used approaches for architectural variability modeling use annotations to assign model elements to different variants. These annotation variability modeling approaches mostly use a so called 150%-percent model of the system architecture incorporating all possible variability in which specific elements are annotated to belong to specific product variants. The monolithic 150%-percent architecture description gets easily very complex for large product families and is hard to manage in case of evolutionary changes. Introducing a new variant will most likely require changes of the whole model, as modular development and implementation of variable parts is not possible. To counter this problem, ADLs should support variability modeling by representing changes to the architecture in space and in time as explicit first-class entities. The variability description in the ADL should be modular to facilitate tracing changes to particular functions, components, or features. Furthermore, the description should be readable, easy to comprehend, to evolve, and to maintain.

In this paper, we present Δ-MontiArc, an ADL with native support for architectural variability modeling in space and in time that allows defining variants of interactive distributed and Cyber-Physical systems in a modular manner. Δ-MontiArc is based on the concept of delta modeling software product lines [6]. A product line of architectures is described by a core architecture and a set of architectural deltas that encapsulate changes to the core architecture. In order to obtain a particular product variant, a set of suitable deltas defined in a product configuration is applied to the core. As variable parts of a model, e.g. functionality for new product variants, are encapsulated in deltas, this approach overcomes the aforementioned problems of annotative variability modeling. As complexity of models is decreased and modular modeling of variability is possible, delta models are easier to comprehend and to evolve. In previous work [16,14], Δ-MontiArc was used to represent spatial variability only. In this paper, we extend it to capture temporal variability with the same linguistic means. If new products should be included in a product family, new deltas can easily be added to a delta model to generate new variants. If a product variant is no longer supported, its product configuration and redundant deltas may be removed. Modifications to certain product functionalities, e.g., for bug fixing, can be realized by replacing a particular delta by another version. In order to avoid degeneration of a delta model after some evolution steps, it can be refactored to improve its structure without changing the generated products. The evolution of architectures as considered in this paper reflects the evolution of the features contained in a software product line. However, the presented approach solely works on the level of the product line artifacts modeling solution space variability [8], in contrast to problem space variability that is typically captured with product features on the requirements level.