Chapter 13
Extending Problem-Oriented Requirements Engineering for SPL

Abstract In the QuaDRA framework, we have provided support for developing a single system. In this chapter, we show how to enhance the problem-oriented requirements engineering for supporting a product line development. We extend the problem frames approach with a notation for modeling variability by providing a UML profile. Furthermore, we propose the structured PREVISE method, which conducts requirements engineering in software product lines taking into account quality requirements. Our method covers domain engineering as well as application engineering.

13.1 Introduction

In the QuaDRA framework, we developed a method for deriving design alternatives from quality requirements (Chapters 4 - 11). We investigated how different user preferences and needs regarding security and performance can influence the design of software. In this chapter, we perform a first effort for extending the QuaDRA framework, which supports a single-system development to a product-line development addressing quality requirements.

Software product line engineering (SPLE) represents a paradigm to develop software applications which are tailored to individual customer’s needs [194]. The benefits of applying SPLE are the reduction of development cost, enhancement of productivity, reduction of time to market, enhancement of quality, and reduction of maintenance of a software [194].

Software product lines (SPL) involve a set of common features as well as a set of variable ones. The first challenge we are facing is how to utilize and ad-
just conventional requirements engineering techniques for modeling and engineering product families. Modeling and managing variability is the central concept in SPLE. Beyond the variability which is caused by variable requirements, there exist further variabilities, which might emerge because of changes in the environment in which the software will be located. Such kind of variability should be taken into consideration when developing SPL.

In this chapter, we propose the PREVISE (PRoblEm-oriented Variability RequirementS Engineering) method, which conducts requirements engineering in software product lines considering quality requirements. Our method is composed of four phases. It covers domain engineering (Phases 1 and 2) as well as application engineering (Phases 3 and 4). While Phase 1 is concerned with exploring the variability caused due to entities in the environment of the software, Phase 2 identifies the variability in functional and quality requirements. The configuration for a concrete product is selected in Phase 3. Subsequently, deriving a requirement model for a concrete product is achieved in Phase 4.

The PREVISE method extends Phase 1 of the QuaDRA framework for developing SPL. The benefit of our proposed method is manifold.

- First, it elicits variability and adds it stepwise to the model. One can start from a description of a system-to-be containing no variability. Hence, we do not rely on (complete) knowledge about the variability of a system-to-be.
- Second, it considers all kinds of sources for variability in a structured way. It identifies the variability, which is caused by entities in the environment, the communication shared between the entities and the system-to-be, and variability due to varying behavior of the system-to-be.
- Third, the information regarding variability, environment, functional requirements, and quality requirements are kept in a single model, which facilitates consistency checking, traceability, and tool support. For modeling we rely on UML, which is widely adopted and provides diagrams for all software engineering phases.
- Fourth, our method enables the generation of feature diagrams [140] and Orthogonal Variability Modeling (OVM) [194] diagrams from the model, which enables documentation and supports the analysis of the variability (see Section 2.5 for more information regarding feature modeling and OVM).

This chapter is based on our work presented in [10]. The PREVISE method and the UML profile for variability modeling have been developed jointly with our colleague Stephan Faßbender. The application of the method to the example has been done by Stephan Faßbender. We had valuable and useful discussions with our colleague Martin Filipczyk regarding variability modeling. Michael Goedicke, Maritta Heisel, and Marco Konersmann provided valuable feedback on this work.