Chapter 9
Domain-Driven Reuse of Software Design Models[1]

Audris Kalnins
IMCS University of Latvia, Latvia

Michał Śmiałek
Warsaw University of Technology, Poland

Elina Kalnina
IMCS University of Latvia, Latvia

Edgars Celms
IMCS University of Latvia, Latvia

Wiktor Nowakowski
Warsaw University of Technology, Poland

Tomasz Straszak
Warsaw University of Technology, Poland

ABSTRACT

This chapter presents an approach to software development where model driven development and software reuse facilities are combined in a natural way. The basis for all of this is a semiformal requirements language RSL. The requirements in RSL consist of use cases refined by scenarios in a simple controlled natural language and the domain vocabulary containing the domain concepts. The chapter shows how model transformations building a platform independent model (PIM) can be applied directly to the requirements specified in RSL by domain experts. Further development of the software case (PSM, code) is also supported by transformations, which in addition ensure a rich traceability within the software case. The reuse support relies on a similarity based comparison of requirements for software cases. If a similar part is found in an existing software case, a traceability link based slice of the solution can be merged into the new case. The implementation of the approach is briefly sketched.

DOI: 10.4018/978-1-61692-874-2.ch009

INTRODUCTION

Some of the most significant cornerstones for state-of-the-art software development are model driven development (MDD) and software reuse. There is a lot of success in applying them separately, but practically nothing has been done to combine them. The proposed approach provides a tight natural integration of both. The third equally important cornerstone is an adequate facility for specifying semiformal requirements to the software system being developed. All these three components together provide support for "model and requirement driven reuse". Only in this way a complete MDD life cycle can be supported, where the use of models starts from the "very beginning". Reuse can also be significantly simplified this way because requirements alone can be used to find candidates for reuse and to select system parts to be reused.

Our approach is based on a special Requirements Specification Language (RSL). This language is semiformal in the sense that it is close to a natural language and understandable to non-IT specialists, but on the other hand it has a meaning precise enough to be processed by model transformations and reuse mechanisms. Consequently, a true model driven development (MDD) is possible, where the initial version of next model in the chain is built from the previous one by model transformations. In totality, these models form a software case. Thus, there is an automatic transformation supported path from requirements to code. All these models play an important role in the reuse process.

More precisely, requirements in RSL consist of two related parts. The domain concepts to be used in the requirements are described in a domain vocabulary. This domain vocabulary serves as a semiformal easy readable equivalent of the domain class model. The meaning of domain elements can be specified by means of links to corresponding WordNet (Fellbaum, 1998) entries. The domain model serves as the basis for the other part of requirements – the required system behaviour description. This description is centred on use cases. The distinctive feature of RSL is that a use case is refined by one or more scenarios in a simple controlled language. Each noun within a scenario sentence must be defined in the domain vocabulary, thus the whole sentence gets a precise meaning. In addition to use cases, non-functional requirements to the system can be described by natural language sentences, using hyperlinks to the same vocabulary. The precise syntax of RSL is described by a metamodel. RSL will be described in more details in next section. Requirements model in RSL can be treated as a Computation Independent Model (CIM) in the classical MDA model chain (Object Management Group, 2003).

When the software case development starts, the requirements model is transformed into the initial version of Platform Independent Model (PIM) in the selected subset of UML (Object Management Group, 2009). The static structure of this model is generated from the domain vocabulary within requirements. Consequently, the whole structure of the system, especially its business logic and data access layers, depend on this domain. Thus a true domain driven design is supported. An initial version of the behaviour is obtained by transformations analyzing the use case scenarios, thus aspects of use case driven design are also present. The precise contents of the generated PIM depend on the selected architecture style for the software system to be developed. Model transformation sets supporting several architecture styles have been developed. The obtained PIM can be manually extended, then another transformation can be applied to generate the initial version of PSM. A similar step leads to initial code for the system. More details on the transformation assisted software case development will be given in section "Definition of software cases".

One more important aspect of the approach is the strong support of traceability in the form