Chapter 17

The Application-Based Domain Modeling Approach: Principles and Evaluation

Iris Reinhartz-Berger
University of Haifa, Israel

Arnon Sturm
Ben-Gurion University of the Negev, Israel

ABSTRACT

Domain analysis provides guidelines and validation aids for specifying families of applications and capturing their terminology. Thus, domain analysis can be considered as an important type of reuse, validation, and knowledge representation. Metamodeling techniques, feature-oriented approaches, and architectural-based methods are used for analyzing domains and creating application artifacts in these domains. These works mainly focus on representing the domain knowledge and creating applications. However, they provide insufficient guidelines (if any) for creating complete application artifacts that satisfy the application requirements on one hand and the domain rules and constraints on the other hand. This chapter claims that domain artifacts may assist in creating complete and valid application artifacts and presents a general approach, called Application-based DOmain Modeling (ADOM), for this purpose. ADOM enables specifying domains and applications similarly, (re)using domain knowledge in applications, and validating applications against the relevant domain models and artifacts. The authors demonstrate the approach, which is supported by a CASE tool, on the standard modeling language, UML, and report experimental results which advocate that the availability of domain models may help achieve more complete application models without reducing the comprehension of these models.

INTRODUCTION

Domain Engineering enables identifying, modeling, constructing, cataloging, and disseminating the commonalities and differences of applications in a domain (Prieto-Diaz, 1990; Champeaux, 1993; Nakatani et al. 1999; Czarnecki & Eisenackner, 2000). A domain in this context is an area of knowledge which uses common concepts that are accepted by practitioners in that area. Similarly, software product line engineering provides aids for specifying sets of software-intensive systems that share common,
managed sets of features satisfying the specific needs of particular market segments or missions (Pohl et al., 2005; SEI-CMU, 2008). In this discipline, the term ‘software product line’ replaces ‘domain’. Domain engineering and software product line methods receive special attention from communities which deal with reuse, validation, and knowledge representation (Meekel et al., 1997; Addy, 1998; SEI-CMU, 2008). Important reasons for this tendency might be the increasing variability of information and software systems, the need to acquire expertise in different, evolving domains, and the requirements to develop “similar” artifacts taking into consideration business drivers, such as time-to-market, cost, productivity, and quality.

A core activity in domain engineering and software product line engineering is domain analysis, which identifies a domain and captures its ontology (Valerio et al., 1997). It should specify the basic elements of the domain, organize an understanding of the relationships among these elements, and represent this understanding in a useful way (Czarnecki & Eisenecker, 2000). Departing from “regular” reuse techniques, domain analysis methods are expected to provide some kind of support to specification of variability within the domain and not just to the commonality. Several methods and architectures have been developed to support domain analysis through modeling. However, these mainly focus on the specification and representation of the domain knowledge and lack in guiding and validating the reuse of domain knowledge in particular application models.

In this chapter, we present the Application-based Domain Modeling (ADOM) approach which provides aids for capturing and representing domain knowledge, creating application artifacts from them, and validating these artifacts according to the domain knowledge. ADOM’s framework consists of three layers: language, domain, and application. The language layer includes metamodels of modeling languages (or methods), such as UML. In the domain layer, the domain elements, structure, and behavior are modeled using a modeling language that is defined in the language layer. Finally, in the application layer, the designated applications are modeled using the knowledge and constraints presented in the domain layer and the modeling constructs specified in the language layer.

ADOM supports different inter-layer activities, and in particular domain layer artifacts may be used for creation and validation of application layer artifacts, while applications may be generalized into domain artifacts in a process of knowledge elicitation. Furthermore, ADOM can be used with different modeling languages for performing various modeling tasks (e.g., business modeling, requirements analysis, and design). However, when adopting ADOM to a specific modeling language, this language is used in both application and domain layers, easing the inter-layer activities. Here we use the standard modeling language, UML, for demonstrating ADOM’s principles and capabilities in both application and domain layers. This dialect of ADOM is called ADOM-UML.

The rest of the chapter is organized as follows. The next section reviews related work in the area of domain analysis. Following, ADOM-UML is presented, describing the domain layer and its provided guidelines, the application layer, and the validation mechanism between these layers. This section also includes an overview of the CASE tool used for ADOM-UML. Next, experimental results regarding the usefulness of ADOM-UML in terms of application model correctness and completeness are reported. Finally, conclusions and future research plans are outlined.

**LITERATURE REVIEW**

Domain analysis deals with identifying stakeholders and their objectives in a domain, defining selection criteria, identifying boundary conditions, examples, and counter examples, characterizing