Requirements Engineering-Based Conceptual Modelling

E. Insfrán, O. Pastor and R. Wieringa

Department of Information Systems and Computation, Valencia University of Technology, Valencia, Spain; Department of Computer Science, University of Twente, Enschede, The Netherlands

The software production process involves a set of phases where a clear relationship and smooth transitions between them should be introduced. In this paper, a requirements engineering-based conceptual modelling approach is introduced as a way to improve the quality of the software production process. The aim of this approach is to provide a set of techniques and methods to capture software requirements and to provide a way to move from requirements to a conceptual schema in a traceable way. The approach combines a framework for requirements engineering (TRADE) and a graphical object-oriented method for conceptual modelling and code generation (OO-Method). The intended improvement of the software production process is accomplished by providing a precise methodological guidance to go from the user requirements (represented through the use of the appropriate TRADE techniques) to the conceptual schema that properly represents them (according to the conceptual constructs provided by the OO-Method). Additionally, as the OO-Method provides full model-based code generation features, this combination minimises the time dedicated to obtaining the final software product.

Keywords: Conceptual modelling; OO methods; Requirements engineering; Traceability; Use cases

1. Introduction

In this paper, we introduce a requirements engineering-based conceptual modelling approach as a way to improve the quality of the software production process. The approach introduces techniques to clearly specify functional requirements and a process to systematically decompose these high-level software requirements into a more detailed specification that constitutes the conceptual schema of the desired system. By improvement of the quality of the software production process we mean:

- providing predictability, in the sense of having a conceptual schema that is a precise, well-defined representation of the user requirements;
- improving productivity, because the source conceptual schema that guides the generation of the final software product is linked to the user requirements in a precise way. Having such a precise link allows tracing of changes in the user requirements in the conceptual schema and, consequently, in the final software product. This solves a classical problem in requirements engineering practice.

In order to accomplish these ideas we use the use case metaphor, which is widely extended in requirements engineering environments, but oriented to collect a specific kind of information required to generate the corresponding conceptual schema. In particular, our approach defines a Requirements Model, which captures both functional and usage aspects in a comprehensive manner. This is organised through the use of three complementary techniques: mission statement, function refinement tree and use case diagrams, which are explained in more detail in the following sections.

When dealing with complex systems, it does not seem feasible to go directly from a requirements description provided by the customer to a formal specification in one step [1]. For this reason, we also introduce a requirements analysis process to translate this structured Requirements Model into a conceptual schema specification in a traceable way.
In short, the original contribution of this paper is twofold:

- the introduction of a Requirements Model (RM), which has been created to collect all the functional information needed to face the problem of specifying a conceptual schema;
- a Requirements Analysis Process (RAP), which provides methodological guidance in the process of building such a conceptual schema based on the functional requirements identified.

To deal with the RM, a well-founded set of techniques for requirements specification is needed. To introduce a precise RAP, the target conceptual modelling constructs must be clearly defined; in consequence, a formal conceptual modelling approach should be used. This is why our approach combines some specific selected techniques from the Techniques for Requirements and Architecture Design (TRADE) [2] and a graphical object-oriented method for conceptual modelling and automated code generation (OO-Method) [3]. The blend between the TRADE and OO-Method results in a powerful environment for modelling software requirements and in a strategy for translating these requirements into a graphical conceptual schema. In the OO-Method, this conceptual schema is backed up by a formal specification language (OASIS) [4], and the final implementation is obtained in an automated way by applying the corresponding mappings between conceptual model constructs and their representation in a particular software development environment.

In our view, our approach overcomes two common weaknesses currently existing in software development methods. First, software engineers do not know whether a conceptual model represents the user’s requirements. These requirements are usually represented in a non-structured or semi-structured way with fuzzy traceability in the conceptual model constructs. Second, when the development step is reached, the value of conceptual modelling efforts is unclear, mainly because it is not possible to produce an accurate code that is functionally equivalent to the conceptual schema built earlier. In our approach, an RAP assures that each element from the RM (problem space) will have a representation in the conceptual model and, transitively, this will have its representation in the target development language (solution space) in the context of a model-based code generation (MBCG) [5].

This paper focuses only on the RM and its representation in a conceptual schema by means of the RAP. Detailed work on translating conceptual models to a specific implementation can be found elsewhere [6–8]. The paper is organised as follows: after the introduction, Section 2 presents a short description of the main features of the TRADE and the OO-Method and how we combine the best properties of each. Section 3 presents the requirements engineering-based conceptual modelling approach, where the building of the conceptual schema from the RM is explained. The last section gives the conclusions and outlines further work.

2. TRADE and OO-Method

The TRADE [2] is a set of techniques and heuristics based on an analysis of structured and object-oriented specification methods [9]. These techniques have been backed up by a formal semantics (when possible and necessary) and they are defined so that they fit together in a simple and well-defined way. The conceptual framework of TRADE distinguishes external system interactions from internal components as a structural mechanism for classifying system properties. External interactions are external functions from which we can specify their behaviour and communication. Some techniques used in TRADE for specifying external interactions and their properties are: mission statement, function refinement tree, context diagrams, use case diagrams and scenario diagrams.

The OO-Method [3,8] is an object-oriented method that provides a set of well-defined and complementary graphical techniques to build a conceptual schema of the system. These graphical UML-based techniques are backed up by a formal object-oriented specification language (OASIS) [4] in such a way that every element in the graphical techniques corresponds (one-to-one) to a section in the formal specification language. In this way, the formal specification acts as a high-level system repository for all the static and dynamic properties of the system (structure, behaviour and functionality). Furthermore, an application that is functionally equivalent to the OASIS specification can also be generated in an automated way. This is achieved by defining an Execution Model, which gives the pattern for obtaining a specific implementation in an imperative software development environment. A CASE Tool\(^1\) provides an operational environment that supports all the methodological aspects of the OO-Method which has been developed in the context of an R&D project carried out jointly by the Valencia University of Technology, CARE Technologies SA and Consoft SA in Spain.

The work presented in this paper is based on the union of these two approaches. It is important to justify the good properties obtained by such a combination. It is clear that TRADE provides an interesting set of\(^1\)More information about the OO-Method and its CASE tool can be found at http://www.upv.es/oomethod.