Patterns and metamodel for a natural-language-based requirements specification language€

Carlos Videira [1], Alberto Rodrigues da Silva[2]

[1] INESC-ID, Universidade Autónoma de Lisboa,
Rua de Santa Marta, nº 56, 1169-023 Lisboa, Portugal
cvideira@acm.org
[2] INESC-ID, Instituto Superior Técnico,
Rua Alves Redol, nº 9 –1000-029 Lisboa, Portugal
alberto.silva@acm.org

Abstract. Software requirements engineering is an essential activity for the successful development of information systems. Requirements can be specified using different techniques, but taking into account the different stakeholders that have to deal with requirements information, a successful approach has to balance the need to use a widely understood requirements notation, with the importance of eliminating the frequent ambiguity and imprecision, by adding rigor to the specification. In this paper we present the metamodel of a controlled natural language for interactive systems requirements specification, called ProjectIT-RSL, based on the identification of the linguistic patterns that are most frequently used in requirements documents written in natural language.

Keywords. Requirements; Requirements Specification Languages; Controlled Natural Languages

1 Introduction

The development of information systems is a complex process usually initiated with the identification and specification of the requirements of the system to be developed, and in particular of its software components. Requirements describe what the system should do, which is obviously critical for the success of the whole development process. Several surveys and studies (such as [7]) have emphasized the costs and quality problems that can result from mistakes in the early phases of system development, such as inadequate, inconsistent, incomplete, or ambiguous requirements.

Software requirements are normally described using a requirements specification language. These languages can be classified through many perspectives; one that is

€ The work presented in this paper is partially funded by FCT (Portuguese Research and Technology Foundation), project POSI/EIA/57642/2004 (Requirements engineering and model-based approaches in the ProjectIT research program).

Proceedings of the CAiSE'05 Forum - O. Belo, J. Eder, J. Falcão e Cunha, O. Pastor (Eds.)

particularly interesting for our work is the level of formality used, ranging from the formal initiatives to those completely informal. The ideal solution should combine the benefits of simultaneously using natural language and a precise (not necessarily formal) approach for requirements specification, which leads to the idea of using a controlled natural language. A controlled natural language is a subset of the words used in our common natural language, where the selected terms have their usual meaning, but can also be interpreted in specialized contexts (normally using tools).

As a result of the experience gathered from previous research and practical projects, the Information Systems Group of INESC-ID [http://gsi.inesc-id.pt/], in Lisbon, Portugal, started an initiative in the area of requirements engineering, named ProjectIT-Requirements [9], which proposes a new approach to successfully achieve some of the goals of this discipline. One of the results of this project is a new requirements specification language, called ProjectIT-RSL [8]. The goal of this paper is to describe the current status of its metamodel, and the underlying patterns. Section 2 presents an overview of the requirements patterns we have identified. Section 3 describes the most important concepts of the metamodel. Section 4 presents previous related work, describes some open issues, and identifies the work to be performed in the future.

2 ProjectIT-RSL Patterns

The analysis of the interactive systems requirements can lead to the identification of a number of frequently occurring patterns. Most of the sentences are used to specify what the system should (a verb frequently found in requirements specifications) do, and present an operational perspective of the system. This is why they follow a simple and common structure, where a *subject* performs an *operation* (expressed by a verb), which affects an *object*. More elaborated sentences can also include a condition, which describes a constraint about a simpler fact. So we started with some basic patterns, described below using an EBNF notation; the list presented here is representative of our current requirement patterns and not complete.

```
<Requirement>: <Simple Requirement> | <Conditional Requirement>
<Simple Requirement>: <Subject> <Operation> <Object>
<Subject>: <System> | <Actor>
<Object>: <Entity> | <Entity property> | <Entity>* | <Entity Property>*
<Operation>: <Action> | <Complex Operation>
<Action>: <Interaction Action> | <Database Action> | …
<Interaction Action>: manages | enters | shows | gets | creates | …
<Database Action>: inserts | updates | deletes
<Complex Operation>: <Operation>*
```

The above list introduces the patterns for higher level requirements, but also demonstrates that, using some of them, we can describe detailed operations. One important goal we wanted to achieve is to provide a single (or at least integrated) technique for the specification of early and late requirements. This is also demonstrated by the list of patterns involved in the specification of conditional requirements.

```
<Conditional Requirement>: <Condition> <Requirement>
```

```
<Condition>: <Simple Condition> | <Complex Condition>
<Simple Condition>: <Condition Type> <Expression> <Comparison> <Expression>
<Complex Condition>: <Condition> <Logical Operator> <Condition>
<Condition Type>: if | while
<Logical Operator>: or | and | not
<Comparison>: greater than | less than | equal to | …
```

These simple patterns enable the representation of requirements sentences such as:

```
System manages entities.
User creates invoices.
User enters invoice date.
System gets invoice number.
If invoice total greater than 1000€, then invoice discount equals 10%.
```

The introduction of rigour in the specification, even when using natural language, imposes some rules that necessarily make a requirements specification different from another used without such restrictions. We can see from the above patterns that we try to simplify the description, for example by eliminating verbs frequently found in requirements specification, such as *shall*, *will* or *must*, or by eliminating words such as *the*. In this area there are still some issues under discussion, for example, a complete definition about the use of singular or plural forms of verbs.

3 ProjectIT-RSL Metamodel

After the identification of requirement's patterns for this type of systems, we derived a metamodel that can adequately represent the patterns identified. ProjectIT-RSL is currently a simple language represented by a metamodel and a grammar that defines the rules to map the metamodel's concepts into sentences, which are validated by the requirements tools.

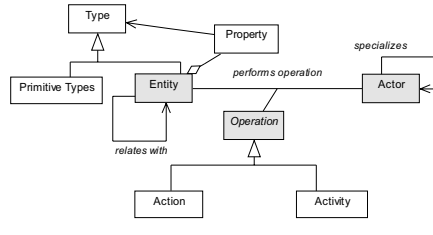

Figure 1 – ProjectIT-RSL metamodel – basic concepts

The main concepts of our language (the ontology) are represented in Figure 1:

- **Actors** are active resources (e.g., an external system or an end-user role) that perform operations involving one or more entities.
- **Entities** are static resources affected by operations (e.g., a client or an invoice). Entities have **Properties** that represent and describe their state.
- **Operations** are described by their respective workflows, which consist of a sequence of simpler Operations that affect Entities and their Properties. Operations

are specialized in **Actions**, which are atomic and primitive (and provided by default by our framework)**,** and **Activities**, which are not atomic.

The metamodel includes other concepts whose goal is to define the organization of the requirements, and which are represented in Figure 2.

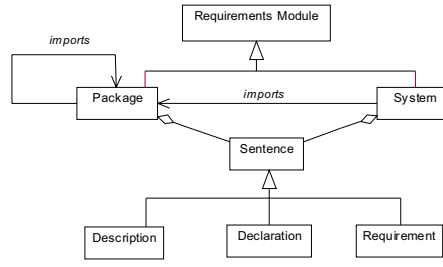

Figure 2 – ProjectIT-RSL metamodel – organizational concepts

All requirements information is expressed in terms of **Sentences** (which represents every sentence of our text), which can be a **Requirement**, a **Description** or a **Declaration**. These statements are included in requirements documents, which we call **Requirements Modules** that are specialized in:

1. **Packages** allow the specification of information about concepts (Actors, Entities, Operations, Requirements) that can be reused in other requirements documents.
2. A **System**, which represents a software component (either a complete application or a reusable software component) with which an actor interacts, by executing operations to access or manage entities and their properties.

Requirements describe what the system is expected to do, and are represented by a number of concepts, as Figure 3 shows.

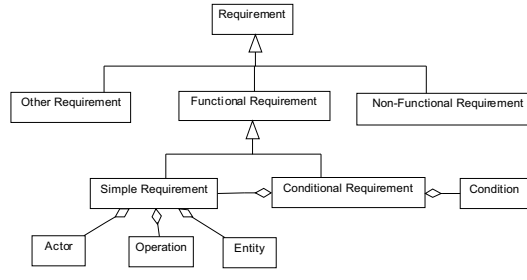

Figure 3 – ProjectIT-RSL metamodel – requirements classes

Requirements are currently specialized in:

1. **Functional Requirements** are the current focus of our research and specify the functional features of the system**;** these requirements are themselves specialized into (1) **Simple Requirements**, which is basically a relationship between an Actor, an Operation and an Entity or an Entity's Property, and (2) **Conditional Requirements,** which add a condition to a Simple Requirement.
2. **Non-Functional Requirements** (e.g., for timing restrictions)

3. **Other Requirements** (e.g., to include information about the need to use a specific database management system).

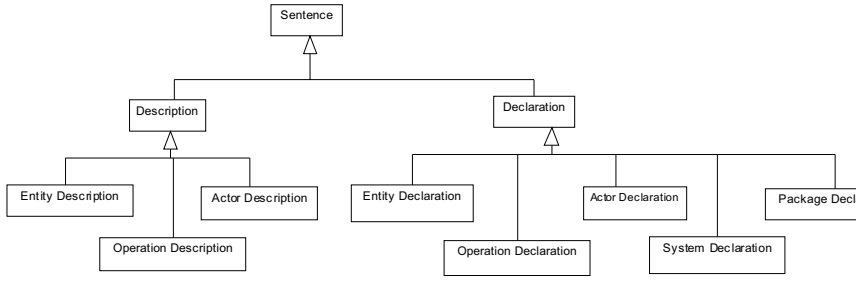

Figure 4 – ProjectIT-RSL metamodel – description and declaration classes

Besides the requirements sentences, a requirements specification in ProjectIT-RSL can also include two other types of sentences, represented in our metamodel by **Description** and **Declaration** classes (Figure 4); the first represents the sentences that begin the detailed information about a concept, such as an entity (**Entity Description**), an actor (**Actor Description**) or an operation (**Operation Description**). Declarations represent sentences that enumerate the entities, the operations and the actors of the system (**Entity Declaration**, **Operation Declaration** and **Actor Declaration**, respectively), as well as those name the requirements module being defined, a System or Package (**System Declaration** and **Package Declaration**). For simplicity reasons, there are other classes in our current metamodel that are not shown in the above figures, such as the classes that represent the description of the properties of an Entity.

4 Open Issues and Future Work

We recognise that there are a number of issues in ProjectIT-RSL that need to be addressed. One of the most important questions still under discussion is the level of resemblance to natural language that we will adopt; the option is between a description very close to natural language (the current option), or one that adopts a style similar to a programming language. Other types of requirements will have to be supported by our language. As Figure 3 has shown, he have already included in the metamodel two classes for Non-Functional Requirements and Other Requirements, for future use. However, as we have not provided any feature in our tools to validate these requirements and to take them into consideration by code generation techniques, they have not been object of current research, which is a situation we will change in the future.

An area considered of prime importance in this project is the support of requirements reuse. Besides the possibility of reusing a previously specified operation in the same requirements module, we have developed other techniques, such as “including” requirements modules in new modules. The information contained in the included module can be reused by the including module. This poses other questions, such as the

visibility of the information of the included module, which are currently being discussed.

In our work, we have been influenced by a number of different approaches that have researched on the elaboration of requirements specification using patterns of natural language. Approaches such as [1] and [6] reduce the level of imprecision in requirements by using a limited number of sentence patterns to specify a requirement for a particular domain. In [5] the authors propose a set of requirements patterns for embedded software systems; these patterns are represented in UML and cover both structural and behavioral aspects of a requirements specification. Their work influenced the one described in [2], where the authors have also identified natural language patterns that can be used to specify functional requirements of embedded systems, from which they developed a requirements statements metamodel. A formalization process based on the analysis of natural language sentences in order to create precise conceptual models is described in [4]. Attempto Controlled English (ACE), first described in [3], is one of the approaches that use a controlled natural language to write precise specifications that, for example, enable their translation into first-order logic.

In the near future we will concentrate on the open issues of ProjectIT-RSL, above all in the development of the requirements reuse mechanisms and in advancing tool support. In a further iteration, it is our intention to include support for the specification of the goals of the system, and from them deriving the requirements specification

References

1. Ben Achour, C., *Guiding Scenario Authoring*, Proceedings of the 8th European-Japanese Conference on Information Modeling and Knowledge Bases, pp. 152–171, IOS Press, Vamala, Finland, May 1998
2. Denger, C., *High Quality Requirements Specifications for Embedded Systems through Authoring Rules and Language Patterns*, M.Sc. Thesis, Fachbereich Informatik, Universität Kaiserslautern, Kaiserslautern, Germany 2002
3. Fuchs, N., Schwitter, R., *Attempto Controlled English (ACE)*, CLAW 96, First International Workshop on Controlled Language Applications, University of Leuven, Belgium, March 1996
4. Juristo, N., Morant, J., Moreno, A., *A formal approach for generating oo specifications from natural language*, The Journal of Systems and Software, Vol. 48, pp. 139-153, 1999
5. Konrad, S., Cheng, B., *Requirements patterns for embedded systems*, Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE02), Essen, Germany, September 2002
6. Rolland, C., Proix, C., *A Natural Language Approach for Requirements Engineering*, Proceedings of the 4th Int. Conf. Advanced Information Systems, CAiSE 1992
7. The Chaos Report, available at http://www.standishgroup.com
8. Videira, C., Silva, A., *The ProjectIT-RSL Language Overview*, UML Modeling Languages and Applications: UML 2004 Satellite Activities, Lisbon, Portugal, October 2004
9. Videira, C., Silva, A., *ProjectIT-Requirements, a Formal and User-oriented Approach to Requirements Specification*, Actas de las IV Jornadas Iberoamericanas en Ingeniería del Software e Ingeniería del Conocimiento - Volumen I - pp 175-190, Madrid, Spain, November 2004