A Requirements Specification Case Study with ProjectIT-Studio/Requirements

David Ferreira
INESC-ID, Instituto Superior Técnico
Rua Alves Redol 9
Lisbon, Portugal
david.ferreira@inesc-id.pt

Alberto Rodrigues da Silva
INESC-ID, Instituto Superior Técnico
Rua Alves Redol 9
Lisbon, Portugal
alberto.silva@acm.org

ABSTRACT

The early stages of the software development process are crucial to overcome the generalized unsuccess among IT projects. Therefore, we embraced the goal of creating a CASE tool for requirements specification. Our approach is based on a controlled natural language, seeking to achieve a higher level of rigor and quality of requirements specifications. Moreover, this tool belongs to a broader workbench, which covers the software development life-cycle, providing guidance according to SE best practices. This paper presents an illustrative specification case study, emphasizing the tool's support and its advantages.

Categories and Subject Descriptors

D.2.1 [**SOFTWARE ENGINEERING**]: Requirements/ Specifications—*Languages*; D.2.2 [**SOFTWARE ENGINEERING**]: Design Tools and Techniques—*Computer-aided software engineering (CASE)*

General Terms

Design, Experimentation, Languages, Verification

Keywords

Requirements, Specification, Validation, CASE Tool, Controlled Natural Language, NLP Techniques

1. INTRODUCTION

The software development process is still an immature activity, presenting several critical issues that compromise its success. Its main cause derives from the non adoption of a systematic approach founded on the best practices proclaimed by the SE community. Requirements, in their essence, describe what the system should do or, at least, a constraint on its behavior; hence they are critical for the development process success. Our proposal seeks to reduce these negative aspects by introducing a controlled natural language for requirements specification, supported by a CASE tool. Our language is based on the identification of common linguistic patterns of requirements documents. By using a natural language approach, non-technical users can easily specify, in a semi-formal manner, the intended requirements. Our CASE tool provides on-the-fly validation and determines the specification's syntactical consistency.

This paper highlights our tool's benefits, in terms of productivity and quality, through a small case study. Section 2 overviews the ProjectIT initiative and the ProjectIT-Studio/Requirements tool. The case study is presented in Section 3. Finally, Section 4 presents conclusions, based on this experience, about our research and future work issues.

SAC'08, March 16-20, 2008, Fortaleza, Ceará, Brazil

2. PROJECTIT OVERVIEW

The emphasis on software development projects should be focused in the project management, requirements engineering, and design activities. Consequently, the effort in production activities, like software programming and testing, should be minimized and performed as automatically as possible. Therefore, to substantiate this vision, we started an initiative in the area of requirements engineering and model-driven development, which we have named ProjectIT [1]. One of its results is ProjectIT-Studio, which is a modular and extensible workbench that we have developed to support and test our research ideas [2]. It provides an integrated environment that supports some of the most important tasks of the software development life cycle, such as requirements specification, architecture definition, system design and modeling, and code development.

ProjectIT-Requirements [3] is the component of the ProjectIT architecture that deals with requirements issues. Its main goal is to develop a model for requirements specification, which, by raising their specification rigor, facilitates the reuse and integration with development environments driven by models. Its text editor, PIT-Studio/Requirements, implements the vision to build a tool for writing requirements documents, like a word processor that detects and gives feedback from errors violating the grammar rules.

3. MYORDERS2 CASE STUDY

This section presents a practical application of the main RSL concepts, emphasizing the benefits of the ProjectIT approach and its support with the PIT-Studio/Requirements tool. It follows a simple case study, named MyOrders2, which is a simple IS, specified and designed in a platform-independent way, to support an abstract scenario of inter-

action between customers and suppliers. This example provides a concrete proof-of-concept of the tool benefits.

From a user point of view, the specification process is simple, since our tool offers a powerful and interactive experience for requirements specification, in which users are assisted by error reporting mechanisms and a comprehensive set of views while writing them. This is achieved by constantly validating the input against the specification model. Code 1, depicts the typical structure of a PIT-RSL requirements specification document, stressing the several types of sections and the hierarchical relationships between them.

After defining the Section Introduction, the user must specify the main application unit region, called System context, followed by the system's name, in this case MyOrders2. At this level, as well as in other inner nested levels, users can specify as many Section Comments as they want because they are neither parsed nor stored, since they only provide an auxiliary annotation mechanism for enriching the specifications description. At System scope level, the user can also specify a Section Imports, one per System region. This specific section type is used to declare import entries, which when parsed by the *Fuzzy Matching Parser*, will make it invoke the *Structural Parser* import mechanism to include its contents in the document's underlying conceptual model. The Section Business Entities is of paramount importance because it is where one can define the domain model of the system currently under specification. Afterwards, MyOrders2 actors hierarchy specification is presented, as well as the features involved on, which appear in the form of pre-defined actions, such as the typical CRUD operations.

When the requirements engineer finishes the specification of the requirements document, a corresponding UML 2.0 model can be generated automatically. Currently, only the domain model (class diagram) and actors hierarchy (a part of use case diagrams) can be generated. This UML model can be further refined during the design phase [2].

Code 1: "MyOrders 2" specification document.

```
1 Section Introduction
  Description of a simple orders management software information system.
  Main objective: to provide a simple case study for proof-of-concept of
    the ProjectIT-Studio/Requirements.
End Section

2 System "MyOrders2"

  2.1 Section Comments
    This will describe the domain view, corresponding to the static/
      structural view of the system.
  End Section

  2.2 Section Imports
    - use \Documents\Invoice.
    - use \Documents\Document.
  End Section


  2.3 Section Business Entities
    <business entities pre edit point>

    2.3.1 A thirdParty is an entity.
    2.3.2 Each thirdParty has a 'company name' and a list of affiliates.

    2.3.3 An affiliate is an entity.
    2.3.4 Each affiliate has an 'affiliate category', and a 'contact name',
       and an address, and a 'contact title'.

    2.3.5 An address has a city, and a 'zip code', and a country, and a
      phone, and a fax.
    2.3.6 An 'affiliate category' has a description.

    2.3.7 A supplier is a thirdParty.
    2.3.8 Each supplier has a list of 'order details', and an integer
      quality, and an integer responsiveness.

    2.3.9 A customer is a thirdParty.
    2.3.10 A customer is the same as a client.
    2.3.11 Each costumer has an integer importance and an integer
      flexibility.
    2.3.12 Each costumer has one or more markets.
    2.3.13 Each market has a list of markets.

    2.3.14 A market is an entity.
    2.3.15 A market has a name.

    2.3.16 Each supplier provides a list of products.
    2.3.17 Each product is provided by one or more suppliers.

    2.3.18 A product is an entity.
    2.3.19 A product has a 'product name', and a double 'unit price', and
      an integer 'order units', and an integer 'stock units', and boolean
      discontinued.

    2.3.20 A customer emits a list of orders.
    2.3.21 Each order is emitted by one customer.

    2.3.22 An order is an entity.
    2.3.23 An order has a 'order code', and an integer 'order number', and
      a date 'order date', and a date 'required date', and a date 'shipped
       date'.

    2.3.24 An order is composed by a list of 'order detail'.

    2.3.25 An 'order detail' is an entity.
    2.3.26 An 'order detail' has a supplier, and an integer quantity, and a
       double 'unit price', and a double discount.

    <business entities pos edit point>
  End Section

  2.4 Section Functional Requirements
    <functional requirements pre edit point>

    2.4.1 Section Actor Declaration
      2.4.1.1 The user is an actor.
      2.4.1.2 The uUnRegister is a user.
      2.4.1.3 The uRegister is a user.
      2.4.1.4 The uManager is a uRegister.
      2.4.1.5 The uAdmin is a uRegister.
      2.4.1.6 The Person is the same as user.
    End Section

    2.4.2 Section Actors Definition
      2.4.2.1 The user can create, and read, and update, and edit, and
        delete a thirdParty.
      2.4.2.2 uManager can create, and edit, and delete a customer.
      2.4.2.3 uRegistered can create a customer.
    End Section

    <functional requirements pos edit point>
  End Section

End System
```

4. CONCLUSIONS AND FUTURE WORK

This paper presents our approach to deal with RE issues, whose vision is realized by a specification language and a supporting CASE tool. To better explain the concepts and to demonstrate its respective tool support, the paper introduces the academic and proof-of-concept MyOrders2 example, which shows in a practical way how to define a system's specification. Despite it still lacks some usability aspects, it allows the specification of a considerable part of the envisioned system's domain model and actors hierarchy, which is noteworthy when considering the complexity of NLP.

There are still open issues to address in our research, namely enhancing the language's extensibility and reuse mechanisms, improving artifacts traceability, and support the knowledge capture from the natural language specifications.

5. REFERENCES

[1] A. Silva. O Programa de Investigação ProjectIT (whitepaper), October 2004.

[2] A. R. d. Silva, J. Saraiva, D. Ferreira, R. Silva, and C. Videira. Integration of RE and MDE Paradigms: The ProjectIT Approach and Tools. *IET Software: On the Interplay of .NET and Contemporary Development Techniques*, 2007. (to appear).

[3] C. Videira, D. Ferreira, and A. Silva. Using linguistic patterns for improving requirements specification. In *ICSOFT 2006*. INSTICC Press, September 2006.