The Journal of Systems and Software xxx (2013) xxx–xxx

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Conceptual modeling of natural language functional requirements

Vidhu Bhala R. Vidya Sagar[a,*], S. Abirami[b]

[a] Mindtree Limited, India
[b] Department of Information Science and Technology, College of Engineering, Anna University, Guindy, Chennai 600025, India

ARTICLE INFO

Article history:
Received 10 March 2012
Received in revised form 27 July 2013
Accepted 11 August 2013
Available online xxx

Keywords:
Automated requirements analysis
Conceptual modeling
Natural language processing
Syntactic analysis

ABSTRACT

Requirements analysts consider a conceptual model to be an important artifact created during the requirements analysis phase of a software development life cycle (SDLC). A conceptual, or domain model is a visual model of the requirements domain in focus. Owing to its visual nature, the model serves as a platform for the deliberation of requirements by stakeholders and enables requirements analysts to further refine the functional requirements. Conceptual models may evolve into class diagrams during the design and execution phases of the software project. Even a partially automated conceptual model can save significant time during the requirements phase, by quickening the process of graphical communication and visualization.

This paper presents a system to create a conceptual model from functional specifications, written in natural language in an automated manner. Classes and relationships are automatically identified from the functional specifications. This identification is based on the analysis of the grammatical constructs of sentences, and on Object Oriented principles of design. Extended entity-relationship (EER) notations are incorporated into the class relationships. Optimizations are applied to the identified entities during a post-processing stage, and the final conceptual model is rendered.

The use of typed dependencies, combined with rules to derive class relationships offers a granular approach to the extraction of the design elements in the model. The paper illustrates the model creation process using a standard case study, and concludes with an evaluation of the usefulness of this approach for the requirements analysis. The analysis is conducted against both standard published models and conceptual models created by humans, for various evaluation parameters.

1. Introduction

The requirements gathering and analysis phase, or the requirements phase, in short, is the most critical phase in the software development life cycle (SDLC). During this phase, analysts collaborate with project stakeholders to gather the requirements for a project.

The requirements serve as inputs to further phases of the SDLC, in terms of work content, and for the planning of schedule and effort. They need to be as complete as possible. Changes to requirements during subsequent phases of the project can be more expensive than during the requirements phase. A major problem with functional requirements is that of unstated or implicit requirements, which stakeholders assume the analyst knows. Such issues in the requirements phase can cause disagreements among development teams and business teams much later, for instance, during acceptance testing, or after project launch, and can consume a lot of effort and time to correct.

To safeguard against changes made later by the requirements providers, most projects require a signoff of the requirements by all stakeholders, including customers, developers, business analysts and testers. Although this could be part of the process or a contractual requirement in the project, various issues exist that can reduce the signoff process to an 'in letter', rather than 'in spirit' activity.

The requirements analysis phase is highly dependent on personal opinions and is subjective in nature. Some stakeholders have trouble understanding the requirements presented in the form of text without the assistance of an analyst. It is important that the stakeholders are able to understand the requirements presented by the requirements analyst and comprehend the impact of the requirements in a uniform and efficient manner.

While many tools exist to render and visualize requirements models, not many tools exist to assist the requirements analyst during the requirements analysis phase. For the purpose of visualizing

Abbreviations: NLP, natural language processing; UML, unified modeling language; ER, entity relationship; EER, extended entity relationship; OOAD, object oriented analysis and design; SDLC, software development life cycle; OO, object oriented; NLTK, natural language tool kit; POS, part of speech.

* Corresponding author. Tel.: +91 442 266 3229.
E-mail addresses: vidhubhala.r.vidyasagar@gmail.com (V.B.R. Vidya Sagar), abirami_mr@yahoo.com (S. Abirami).

0164-1212/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2013.08.036

the functional requirements, the conceptual or domain model is a valuable intermediate artifact. A conceptual model "*represents 'concepts' (entities) and relationships between ideas in a problem domain*" (CORDIS, 2013). The conceptual model, as defined by Larman (2002) contains only domain entities and attributes, and represents the static model of a system. On the other hand, the UML based OOAD model resembles a UML class diagram (Booch et al., 2010) through the inclusion of class operations and inter-class relationships, which also represent the dynamic nature of the system in study. A conceptual model is considered to be efficient in visual communication because it uses less space and fewer symbols to convey maximal information, as compared to natural language requirements. The use of conceptual models is widely recommended in newer software development models like Agile Modeling (Leffingwell, 2011), and has been adopted by objected-oriented development models using UML (Ambler, 2005).

A quality framework for conceptual model validation was first proposed by LindLand et al. (1994), who presented a framework for a conceptual model covering the syntactic, semantic and pragmatic qualities of the model. Syntactic quality addresses the correctness of the model through the use of a formal syntax. Semantic quality addresses the validity and the completeness of the conceptual model against the target domain, i.e., the domain from which the functional requirements are created. Pragmatic quality addresses the efficiency of the conceptual model in ensuring that the audience comprehends the information the model tries to convey. Each quality is elaborated with goals, and the recommended means to achieve the goals. The availability of a quality framework and the parameters to measure the quality of the conceptual model, make it a suitable choice for systematic requirements analysis.

This article focuses on the automated extraction of concepts and their relations to create a conceptual model. The constructed conceptual model is based only on stated, i.e., explicit requirements . Implicit requirements do not appear in the model, and must be provided in an explicit manner by the stakeholders during the requirements phase itself. The goal is to create a useful conceptual model that the analyst can use, to help the stakeholders understand the requirements before they sign off and conclude the requirements phase of the project.

By automating the creation of the conceptual model, the analyst can focus the attention of the stakeholders and herself/himself on the analysis and refinement of the models, rather than on the creation of the same. This allows the requirements to go through multiple iterations within the same timeframe, resulting in requirements that analysts and stakeholders are comfortable with.

The basis of this work lies in the linguistic aspects of the English language. Natural language sentences that constitute the functional specifications are parsed to understand their grammatical structures. Various design components such as classes, attributes, entities and relationships are then extracted from these parsed sentences, based on textual analysis.

Our work addresses the syntactic aspects of the conceptual model quality. As a logical extension, we discuss Pragmatic quality in Vidhu Bhala et al. (2012), which deals with the uniformity of audience perception, and is achieved through visualization techniques.

The structure of this paper is as follows. In Section 2, a survey of the related work in this field is offered. In Section 3, the architecture of the system is presented in the form of a visual layout of the functional blocks. The algorithms and functionality of each functional block and modules are detailed. Section 4 gives details of the implementation. A working example illustrates each of the functional blocks. In Section 5, an evaluation of the automated conceptual model is presented, and Section 6, gives the conclusion, along with the achievements of the current research work, the shortcomings and possible future work.

2. A survey of existing work

The trend in research in the area of automation of requirements analysis indicates, that although the need for automation was realized in the early 1990s (Kurt, 1995), the concept of the automation of this area has started gaining interest only recently, as is evident from the recent appearances of journal and conference articles in this area of research. A couple of reasons can be inferred for this renewed interest in this field. One is the recent surge in research and subsequent development in automatic text analysis, like parsers for natural language, tagged corpora for specific needs, anaphora resolvers etc., which have reduced the complexities of dealing with natural language. Another reason is the evolution of diagramming models like UML, and the ongoing attempts at standardizing the content of models. With the software industry converging to a few standard and well-known formats of requirements documentation, more focused research with wider application is possible.

Yue et al. (2011) refer to the conversion of requirements into models as a transformation process. They present an extensive survey of the transformation processes. They discovered that five types of pre-processing techniques, i.e., lexical, syntactic, semantic and pragmatic analyses, and categorization were found to be used in isolation or in combination. The resulting models from the transformation processes of the papers that were surveyed, were found to have low efficiency, and lacked evaluation mechanisms. Their survey included 20 works, categorized into 16 transformation approaches. They found that only 7 of the approaches were automated. Further, only 4 of the automated works could generate class diagrams, object diagrams or conceptual models, while the rest generated other types of analysis models like sequence diagrams or state charts. The survey also indicates underlying issues related to the evaluation of the completeness of rules used for transformation, due to the ambiguity of the English language. The four approaches that generated models similar to conceptual models were based on sentence pattern matching or shallow parsing. They required stringent rules on the pattern of the input sentences in the requirements text, for example, sentences of the form – sentence, verb, object, closely relating to the use case mode of writing requirements. Sanjay et al. (2010) explored the automatic creation of domain models from inputs that users provide in a spreadsheet. These inputs are processed as rules that represent individual events connecting entities in the requirements. A tool creates an intermediate representation of the requirements from the parsed rules, from which the domain model is then created. Ambriola and Gervasi (2006) present an extensive requirements modeling and analysis tool called CIRCE, that is based on fuzzy reasoning. A customized parser is used to parse sentences from the requirements text. User glossaries are provided manually to the tool. An expert system adds semantic information to the parsed information to create a final model. In both these approaches, the nature of the input text is constrained either through the use of templates, or through the use of a text that a parser can process.

Mu et al. (2009) parsed natural language requirements using typed dependencies, and extracted requirements from the functional requirements specifications. Their objective is to assess the non-functional requirements. While it might have been possible to use a free form input text, the authors require the input text to be in a standard format. They classify 10 types of semantic cases based on the sentence structure. The output is not a conceptual model, but a customized format for their specific needs.

Most of the publications discuss solutions that impose constraints on the language used for the requirements, with very few attempting to use natural language. Elbendak et al. (2011) represent the requirements model by semi-automatic processing of requirements presented in use cases, that are written in natural language. Their work, which motivated our research, attempts to analyze