Knowledge-based System for Software Requirements Analysis and Management

Natalja Pustovalova, Maxim Bakaev

Novosibirsk State Technical University, Department of Economic Informatics
+7-383-346-06-79
natalja.ru@gmail.com; maxis81@gmail.com

Abstract. The paper describes the work in progress towards the development of knowledge-based system (KBS) to support developers in ensuring the quality of software requirements. Ambiguity, contradictoriness, incompleteness, poor traceability and adjustability were identified as currently most severe problems with requirements, as well as certain ignorance towards existing validation and verification techniques. Frame-based model is proposed as the most appropriate for knowledge organization in the system and its structure is derived from textual sources analysis and experts interviews. Application of concept mining methods with external domain or common sense ontologies or vocabularies serving as thesaurus is proposed, to make possible intellectualized quality assurance of requirements by the KBS.

Keywords: requirement management tools, software engineering, artificial intelligence, frame ontologies, information extraction

1 Introduction

It has almost become a commonplace that a significant share of software development projects fails or exceeds their budget. There is no lack of research on the causes of runaway projects, and unstable or incomplete requirements and poor estimations are justifiably identified as the two major ones [1; 2, p.5]. The former of the two is clearly the primary factor, and it is also widely noted that requirements errors are the most expensive to correct late in the development process or when the product is released.

Certain desired properties for well-defined requirements are described in international standards related to software development: correspondence to the needs and expectations of users and stakeholders, unambiguity for different domain experts, consistency, lack of contradictions, completeness and etc. [3]. An extensive list of approaches and methodologies concerning requirements analysis and management includes SWEBOK, PMBOOK, RUP, DOORS, CORE and so on (review is available in [3]), however there seem to be certain factors hindering their ultimate effectiveness:

1. The discrepancy on the conceptual level. There are different definitions of requirements, compiled by different experts from their diverse experiences [4, 5]. Thus, methodologies are discriminate in their understanding of a requirement, so they may mark out different levels and types of requirements, leading to radical distinctions in the development process.
2. The absence of reliable means for choosing the most appropriate approach for a given project. To make a justified choice, developer would have to be an expert in these various theoretical paradigms, as well as software development methodologies. In practice, the choice depends either on the developers' previous experience, or even on requests from stakeholders, who may be quite inept in software engineering.
3. The lack of accordance on procedures and criteria for assessing the final quality of the requirements. Although methods for requirements validation and verification do exist, they hardly provide clear control points where the quality of the requirements set could and should be estimated, before the construction of the product or its prototype has started.

These factors, in the light of objective complexity of software engineering domain, lead to such generally recognized problems related to requirements:

1. Requirements elicitation and formalization that result in ambiguous or unobvious requirements. Omitted requirements are, probably, even greater problem, as they result in omitted logic in the final product, and this kind of error, according to some estimates [1, p.75], constitute a prominent 30% of all persistent software errors.
2. Poor requirements traceability and adjustability, due to lack of established relationships between them. Nowadays, unstable requirements are less and less viewed as a problem of weak management, but more accepted as objective reality [1, p.69] – thus proper requirements management techniques have to deal with introducing changes to the requirement set.
3. Persistence of requirements errors, which significantly increase risks and costs for the project. Many requirements validation and verification techniques, such as reviews, consistency checks, etc., rely heavily on expertise, however developers are often pressed by project estimates and schedules that generally allocate few time on their involvement in requirement-cleansing [1, p.72].

2 Method

Currently, popular requirements management tools, such as Rational IBM or Borland products, offer means for managing requirements in the course of the project and integrating interim results of the development. However, effective use of these tools, including requirements quality assurance, implies relatively high level of expertise, as domain knowledge is mostly presented in user manual or online help only. Thus, the application of such products is not accessible for all developers or, given their considerable cost, economically feasible in all projects. As we believe the requirements analysis and management domain matches the generally accepted premises for expert systems deployment (e.g., see in [6, p.36]), we suggest the application of knowledge engineering methods to develop a knowledge-based system (KBS) aiding developers in ensuring the quality of requirements. The system, used by software engineers (requirements analysts), should be able to decrease the probability of errors committed by inexperienced developers as well as alleviate workload on skilled ones, by providing support in requirements elicitation, specification, validation and their management throughout the development cycle.

The application of artificial intelligence (AI) methods in software engineering remains a subject of current importance (see latest trends in [7]) and ranges from highlights of general importance of knowledge management in the field [8] to research on application of specific AI methods, such as agents [9], neural network [10] or data mining [11]. To organize knowledge in the system, we chose frame-based model, which generally contains classes (concepts) and slots (concepts' attributes or relationships between concepts) and offers the following advantages.

Taxonomy – the model, as it holds related concepts, is capable of providing unified vision of conceptually discrepant requirements analysis domain.

Relationships – the model allows all kinds of relationships between concepts (including hierarchical with inheritance), thus making available improved requirements traceability, as well as top-down formulation of requirements, from the overall software goal to detailed functional or technical requirements.

Adjustability – slots in a frame-based model may contain the so-called default assignments, which are not necessarily proper, but may be gradually modified as the project develops and requirements are clarified.

Intelligibility – it is believed that frames are consistent with how human mind works [12], so correspondingly organized requirements set may be effectively presented to inexpert stakeholders or various members of the development team.
Procedural – frame-based model may combine declarative and procedural components and thus embody automated or intellectualized requirements analysis.
Integration – formal frame-based structure facilitates the use of external ontologies, including common sense ones, so data (vocabulary, taxonomy, axioms, etc.) may be easily obtained for the software project domain.

3 Results

3.1 The frame model

To obtain the list of concepts and their relationships for the frame model, we used such popular knowledge extraction methods as textual sources analysis (with [1], [4], [5], etc.) and experts interviews. The resulting structure is in essence an ontology (conceptual model) for the requirements analysis and management domain, explicitly providing one of the possible conceptual visions of this field. Figure 1 outlines a fragment of the model concerning *Requirement* frame and certain related concepts (frames relevant to requirements management on subsequent stages of software development process are not shown). The model was implemented in popular Protégé frame ontology editor and visualized with OntoViz plug-in (please see [13] for description of the legend). After ongoing iterative revisions are completed, we plan to publish the ontology online.

3.2 The system capabilities

The description of requirements starts top-down, from business requirement corresponding to the need of target users, to more detailed requirements representing the functional capability of the software product, supplemented by technical or the so-called non-functional requirements. In the system, each requirement is represented as a frame instance, the attributes (slots) of which both allow to describe the requirement and to partake in requirements quality assurance process. The KBS in development has the following principal capabilities:

1. Controls the fill-in of the essential attributes for requirements, allows setting pre-defined or custom relationships between requirements.
2. Suggests possible approaches for composing functional requirements (e.g. use cases), project context dependent types of non-functional requirements (related to interface quality, web response time, security, etc.) and requirements validation techniques.
3. Allows modifications to the requirements, traces change history, notifies on possible effects of changes on related requirements.
4. Provides visualization for the requirements set (tables, tree view, technical documentation, etc.) as well as for related concepts, in particular for running requirements reviews with stakeholders.
5. Helps tracking the degree of the requirements fulfillment during modeling, simulation or prototyping.
6. Checks for inconsistencies in requirements priorities, development order, and pre-requisite requirements; offers conflict resolution strategies (e.g. based on priorities of involved stakeholders).
7. Tracks development estimations vs. real work efforts, builds up statistics for analysis and improved estimating and product release date forecasting.

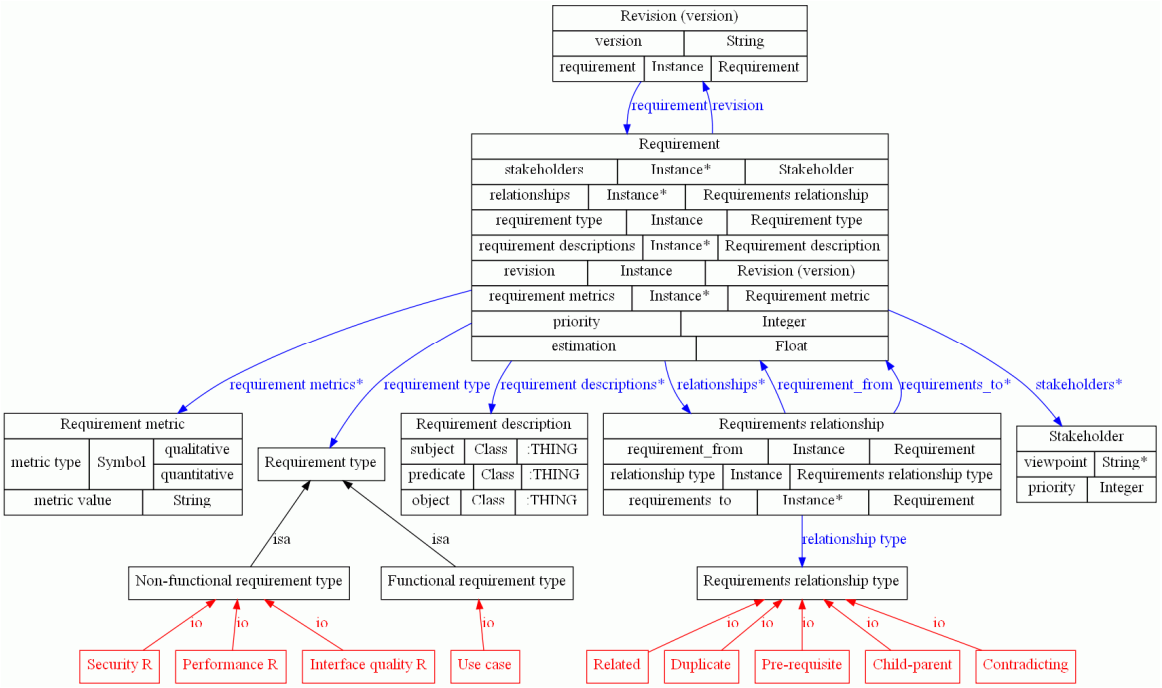

Figure 1. Frame *Requirement* and related concepts.

3.3 Requirements analysis with external ontologies

Further development of the system implies the extension of the procedural component for requirements analysis, so that the system: auto-discovers such relationships between requirements as *Child-parent*, *Related*, *Contradicting*, *Duplicate*; hints on possibly omitted requirements; checks for ambiguous wordings in requirements descriptions.

For that, we plan to utilize information extraction and concept mining techniques, with external domain or common sense ontologies or vocabularies (such as Russian WordNet) providing thesaurus (see [14], [15]). The first step would be text simplification and populating the related *Description* instances with extracted information (see Figure 1, where *Description* frame is designed according to Resource Description Framework specification), and the troubles in the processing would imply possibly unobvious wordings. Then, the KBS would be able to perform checks of different requirements to propose relationships between them, and to make comparisons between *Description* slots values and objects in the thesaurus, to uncover possibly missing requirements.

4 Conclusions

In our article, we sought to reason that application of AI methods in requirements analysis and management domain could contribute to resolving problems persisting in software development, such as ambiguous, contradicting or incomplete requirements, poorly suited for adjusting or tracing. The proposed KBS built based on frame model includes capabilities to aid in control of the requirements quality and provides means for better traceability that leads to advantages on various stages of the development. Further, with respect to requirements validation and verification, the system both offers abundant visualization aimed on ensuring stakeholder involvement (which is major factor for the project success [2, p.4]), and supplies the foundation for automatic requirements analysis. The implementation of the latter is planned with information extraction and concept mining methods, implying utilization of external ontologies (vocabularies) as thesauri.

The developed frame-based knowledge structure for the KBS may serve as a conceptual model of requirements analysis domain, and in turn get used by other

systems related to software development. So, the *Requirement* frame and related concepts were utilized in an adjacent project in Novosibirsk State Technical University: KBS used in solving practical tasks in web interface design, based on such requirements as business goals of the project and target user attributes [16].

References

1. R.L. Glass. Facts and Fallacies of Software Engineering. Addison Wesley, 2003.
2. Standish Group International. The Chaos Report, 1995.
3. V. Kulyamin, N. Pakulin, O. Petrenko, A. Sortov, A. Khoroshilov. Formalizatsiya trebovanii na praktike (in Russian). Preprint 13, ISP RAN, Moscow, 2006.
4. E. Hull, K. Jackson, J. Dick. Requirements Engineering. SpringerVerlag, 2004.
5. D. Leffingwell, D. Widring. Managing software requirements: a unified approach. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1999.
6. T.A. Gavrilova, V.F. Khoroshevsky. Bazi znanii intellektualnikh sistem (in Russian). Piter, St.Petersburg, Russia, 2000.
7. F. Meziane, S. Vadera. Artificial intelligence applications for improved software engineering development: new prospects. IGI Publishing, Hershey, PA, 2009.
8. I. Rus, M. Lindvall, Guest Editors' Introduction: Knowledge Management in Software Engineering. IEEE Software, 19(3): 26-38, May 2002.
9. N. Jennings. On agent-based software engineering. In Artificial Intelligence, 117(2): 277-296, March 2000.
10. K. Iwata, T. Nakashima, Y. Anan and N. Ishii. Improving Accuracy of an Artificial Neural Network Model to Predict Effort and Errors in Embedded Software Development Projects. Studies in Computational Intelligence, 295/2010: 11-21, 2010.
11. M. Zong Chun-mei, Li Zhi-ling. Applying data mining techniques in software development. In The 2nd IEEE International Conference on Information Management and Engineering (ICIME), Chengdu, 16-18 April 2010, pp. 535-538.
12. M. Minsky. A framework for representing knowledge. In The Psychology of Computer Vision. McGraw-Hill: P. Winston, 1975.
13. http://protegewiki.stanford.edu/wiki/OntoViz
14. A. Maedche, S. Staab. Mining Ontologies from Text. Lecture Notes in Computer Science, 1937: 169-189, 2000.
15. I. Spasic, S. Ananiadou, J. McNaught, and A. Kumar. Text mining and ontologies in biomedicine: Making sense of raw text. Briefings in Bioinformatics 6(3): 239 – 251, 2005.
16. M.Bakaev, T. Avdeenko. Knowledge-Based System for Web Interface Design. Proc. of 2nd International Conference on Information and Multimedia Technology (ICIMT 2010), Hong Kong, China, Dec 28-30, 2010, vol. 3, pp. 262-266.

Figure 1. Frame *Requirement* and related concepts.