A Collaborative Requirement Mining Framework to Support OEMs

Romain Pinquié, Philippe Veron, Frédéric Segonds, Nicolas Croué

A Collaborative Requirement Mining Framework to Support OEMs

Romain Pinquié[1(✉)], Philippe Véron[1], Frédéric Segonds[2], and Nicolas Croué[3]

[1] Arts et Métiers ParisTech, LSIS, UMR CNRS 7296, Aix-en-Provence, France
{romain.pinquie,philippe.veron}@ensam.eu
[2] Arts et Métiers ParisTech, LCPI, Paris, France
frederic.segonds@ensam.eu
[3] Keonys, Toulouse, France
nicolas.croue@keonys.com

Abstract. With the fastidiously ever-increasing complexity of systems, the relentless, massive customisation of products and the mushrooming accumulation of legal documents (standards, policies and laws), we can observe a significant increase in requirements. We consider the tremendous volume of requirements as big data with which companies struggle to make strategic decisions early on. This paper proposes a collaborative requirement mining framework to enable the decision-makers of an Original Equipment Manufacturer (OEM) to gain insight and discover opportunities in a massive set of requirements so as to make early effective strategic decisions. The framework supports OEMs willing to uncover a subset of key requirements by distilling large unstructured and semi-structured specifications.

Keywords: Requirements · Exploration · Cooperative engineering · Collaborative decision making · Visual analytics · Framework

1 Introduction

The development of complex systems prompts organisations to build partnerships to pool knowledge and resources and to share risks.

Within the supply chain, a stakeholder is either a system integrator[1] or an Original Equipment Manufacturer (OEM)[2]. Nonetheless, any stakeholder at tier N can, in turn, be an integrator of OEMs' subsystems developed at tier $N-1$, and an OEM that provides a subsystem to an integrator at tier $N+1$.

The development of a system-of-interest starts with the requirements development phase from the point of view of the integrator. Requirements analysts elicit hundreds or thousands of requirements to specify the design, manufacturing, use, maintenance and disposal of a complex system-of-interest. For instance,

[1] An integrator is sometimes known as an acquirer, a customer, a contractor or a contracting authority.

[2] An OEM is sometimes known as a subsystem provider, a subsystem supplier or a

at Mercedes-Benz, the size of a system-of-interest specification varies from 60 to 2000 pages and prescribes between 1000 and 50 000 requirements [1]. Once the system-of-interest is fully specified, systems architects develop functional and physical architectural alternatives. As soon as architects have committed themselves to one preferred architecture, requirements analysts apportion the performance requirements of the current hierarchical level to the functions of the next lower level. This recursive decomposition process, which aims at simplifying the problem to be solved, results in various Stakeholder Requirements Specifications (StRSs)[3] whose implementations require domain-specific knowledge from various OEMs. Therefore, an OEM is the one that takes over all the requirements of a given subsystem.

In its Chaos Manifesto report [2], the Standish Group points out that there is never enough time or money to do everything. Thus, the report argues that focusing on the 20 % of the features and functions that give organisations 80 % of the value will maximise the investment and improve overall user satisfaction. It is therefore strongly recommended to reduce the scope and spend more time focusing on high-value requirements rather than completing 100 % of them. Such a strategy is currently fashionable since the report indicates that the features and functions developed went down, with 74 % of the specified requirements completed in 2010, dropping to 69 % in 2012. However, as Sawyer et *al.* (2002) [3] state, there is no tool that supports the OEMs willing to adopt such a strategy. Hence, so far, when an OEM collects the StRS and the applicable documents to which the StRS refers to, it has no other alternative than go through the documents to identify and prioritise a subset of key requirements.

2 Literature Review and Proposition

2.1 Related Work

An up-to-date literature review of the existing visual requirements analytics[4] frameworks can be found in Reddivari et *al.* [5] and Cooper et *al.* [6]. Coatanéa et *al.* [7] and Lash [8] extract requirements and relationships from unstructured specifications. Zeni et *al.* [9] propose GaiuST, a framework that supports the extraction of legal requirements for regulatory compliance. Few articles broach the topic of contradictions in natural language [7,10]. However, much research attempts to diagnose quality defects by using NLP and text mining techniques [11–17]. Recent work has been done to classify requirements thanks to machine learning [18] or linguistics analysis [16]. Zhang et *al.* [19] suggest an approach to qualify and quantify customer value for value-based requirements engineering. Numerous requirements prioritisation techniques, which are reviewed in [20], have been developed to rank requirements. Finally, there is an stimulating interest to use recommendation systems in requirements engineering [21].

[3] One StRS for each system element.

[4] Reddivari et *al.* [5] coined the term *visual requirements analytics* that is the use of *visual analytics* applied to requirements engineering. Visual analytics is "the science of analytical reasoning facilitated by interactive visual interfaces" [4].

2.2 Limitations of Existing Solutions

In current commercial requirements engineering software can be divided into four main categories: (1) the everlasting document-based specifications (e.g. Word, Excel, PDF) that are easy to use and cheap; (2) the centralised database-based specifications (e.g. Rational DOORS[5], ENOVIA V6 Requirements Central[6]) that ease collaboration and configuration management; the model-based systems engineering specifications (e.g. SysML) that offer new elements such as relationships between requirements; and (4) the requirements editors and analysers that assist analysts with ensuring the quality of requirements (e.g. The Requirements Quality Suite[7]).

Feedback from industrialists reveals that these technologies benefit the systems integrators whose main tasks are requirements writing and management, but they do not help the OEMs willing to distil a large set of requirements. Moreover, they do not offer any contextual view of the requirements and their interdependencies, but only offer a graphical table that imitates the form of a spreadsheet. We can also notice that requirements analysis tools mainly focus on the processing task, but provide very limited visual analytics capabilities. Conversely, visual analytics solutions concentrate on the depiction of requirements' features without including advanced processing capabilities. Finally, to the best of our knowledge, current solutions do not provide any statistical capabilities that may help uncovering patterns in a set of decision-making attributes associated to requirements.

This literature review prompts us to claim that there is an absolute necessity to imagine a framework that supports OEMs willing to distil a large set of requirements by integrating computational engineering with visual analytics.

2.3 Proposition

We propose a collaborative requirement-mining framework that addresses the **problem** of an OEM receiving a request for proposal consisting in a StRS and the referenced applicable documents, all of them prescribing a massive set of requirements that cannot be implemented within cost and schedule constraints.

The **mission** of the framework is to support the business analysts willing to distil a large set of requirements, that is, to create an optimised Sytem Requirements Specification (SyRS) that is ready to be managed in configuration thanks to requirements management tools. By using the "distil" we mean to get and show only the most important part of a large set of requirements. We qualify the SyRS as optimised because it is the outcome of the distilling process that results in a subset of **key requirements**. The key requirements gather:

– **legal requirements:** non negotiable requirements that are required for regulatory compliance.

[5] http://www-03.ibm.com/software/products/fr/ratidoor.

[6] http://www.3ds.com/products-services/enovia/products/v6/portfolio/d/collaborative-innovation/s/governance-user/p/requirements-central/.

[7] http://www.reusecompany.com/requirements-quality-suite.

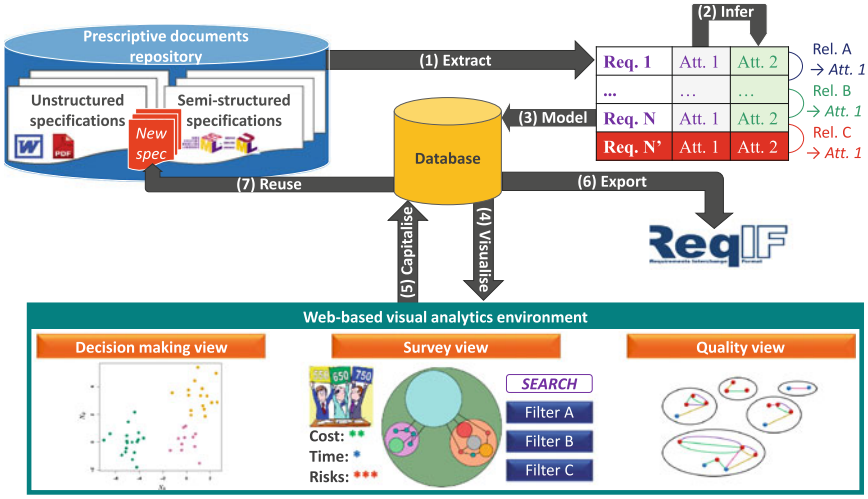

Fig. 1. An over-simplified operational scenario of the collaborative framework.

- **basic requirements:** non negotiable requirements without which no one would buy the product.
- **added-value requirements:** negotiable requirements corresponding to key product differentiators.

Figure 1 illustrates the **operational scenario** that we propose to answer the question "What is the subset of key requirements?". To begin, for a given project, the project manager collects the unstructured and semi-structured documents corresponding to the StRS and the applicable documents and places them into a common repository. Then, the operational scenario is as follows:

1. **Extract** explicitly stated attributes (e.g. original author, version, statement, original document source title, etc.) that make up each requirement and relationship from unstructured and semi-structured specifications. We use an attribute scheme to structure each requirement as a set of attributes so as to be compliant with the meta-model of the standardised Requirements Interchange Format (ReqIF)[8].
2. **Infer** implicit requirements attributes, such as the business category (e.g. mechanics, electronics, I&T, marketing, safety, etc.), the functional *vs.* non-functional category, the textual cross-references to the referenced applicable prescriptive documents (e.g. The system shall comply with the *CS.25*) standard. Implicit relationships such as the linguistic interdependencies (e.g. synonymy, hypernymy, hyponymy, etc.) among the requirements keywords, the potential redundancies and contradictions, and the occurrence of proscribed terms can also be created.
3. **Model** the requirements and the relationships as a property graph data model into a graph database.

[8] http://www.omg.org/spec/ReqIF/.

4. **Visualise** the requirements and the relationships from different perspectives enabling contextual and multidimensional exploration of the attributes through interactive functionalities. Users can interact with visuals, perform the classic CRUD[9] operations, and save changes to the database. When the data set is updated by saving changes in the database, the processing tasks have to be restarted to update the data model and the visuals. So far we have identified five interactive visuals:
   - **Quality view:** highlights the quality defects that are not only inherent to a requirement statement (e.g. ambiguities, incompleteness, etc.), but also among a set of requirements (e.g. redundancies and contradictions).
   - **Cross-references view:** illustrates the interdependencies among the StRS and the referenced applicable documents.
   - **Survey view:** helps the domain-experts to retrieve the requirements that are relevant to their profile so as to estimate various decision-making criteria (e.g. cost, time, risks, quality, etc.).
   - **Decision making view:** supports analysts for uncovering unanticipated patterns in the dataset resulting from a statistical analysis of the decision-making criteria estimated by domain-experts in the *survey view*.
   - **Configuration management view:** depicts the history of changes (created, modified and deleted requirements and relations).
5. **Capitalise** the information and the strategic decisions made by the business analysts by keeping an historical record of subsequent projects.
6. **Export** into a ReqIF compliant XML data file the subset of key requirements that will be managed in configuration throughout the downstream system's life cycle phases thanks to requirements management tools.
7. **Reuse** the capitalised information throughout the entire distilling process whenever a new specification is added in the repository or whenever a new set of specifications corresponding to a new request for proposal is received.

In the next sections we introduce the framework according to two perspectives: *functional* - WHAT functions need to be performed to fulfil the mission? - and *software* - HOW shall the framework accomplish the required functions?

3 The Collaborative Requirement Mining Framework

3.1 Functions of the Framework - "WHAT"

In this section we present a Functional Flow Block Diagram (see Fig. 2) that not only defines the functions that the framework must perform to fulfil its mission, but also depicts their logical, sequential relationship, as well as what flow goes in and out of each function. It is the definition of "what" the framework must do without guessing a particular answer to "how" the functions will be performed.

The first step consists in extracting the explicit attributes that make up each requirement (F1 and F2). For unstructured specifications (F1), attributes

[9] Create, Read, Update and Delete.

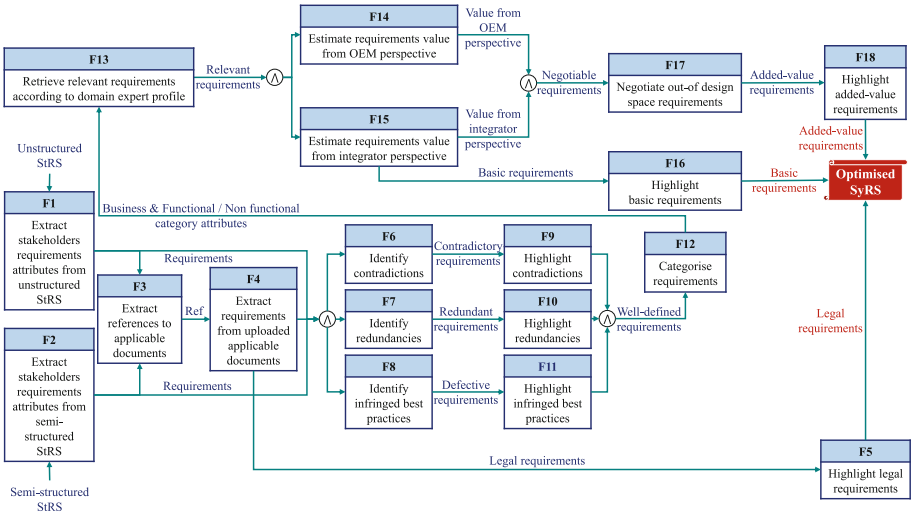

Fig. 2. A FFBD depicting the lowest abstraction level of the functional architecture.

include a unique identifier for each requirement, the metadata corresponding to the author and version of each specification, as well as the statement of each requirement. For semi-structured specifications (F2) such as SysML requirements diagrams saved as an XML file, we only extract the statement attribute and the "derive" relationship elements. Then, the requirement statements are processed to extract the textual cross-references that refer to applicable documents (F3) from which the software extracts additional requirements. Legal requirements have to be highlighted (F5) in a visuals because they are non-negotiable key requirements. It is therefore necessary to find a solution to distinguish legal requirements from non-legal requirements. Once all requirements have been extracted, a diagnosis (F6, F7 and F8) is launched to identify quality defects (e.g. contradictions and redundancies, ambiguities, etc.) to avoid over-specification with unnecessary requirements and misinterpretations during the survey phase. The cleaning of requirements defects requires another visual, the *quality view*, to illustrate contextual potential redundancies and contradictions. Once cleaned, the requirements are used to infer implicit attributes such as the business (e.g. mechanics, IT, etc.) and functional *vs.* non functional categories (F12). Such classes ease the retrieval of requirements (F12) whose value needs to be estimated from the OEM (F14) and the integrator (F15) perspectives. The estimation of decision-making criteria results in either a basic requirement that is non negotiable from the point of view of the integrator or a requirement that needs to be negotiated (F15). The negotiation phase can be approached as a constrained optimisation problem where integrator's constraints need to be weakened to find a solution belonging to the OEM's feasibility space that is maximum in value but that was originally outside of the scale of constraints [22].

3.2 Implementation of the Framework - "HOW"

In the previous section we detailed what functions the framework must perform to fulfil its mission. Now we shall explain how the functions can be implemented.

To extract the textual content and the metadata we use a specific parser according to the extension of the uploaded StRS. The parser Apache Tika[10] for .doc (Word), .odf (OpenOffice) and .pdf; the parser Apache POI[11] for .xls (Excel) and to extract figures that potentially are graphical requirements; and a simple XML parser (JDOM[12]) for SysML requirement diagrams.

The Stanford CoreNLP[13] toolkit offers a variety of natural language processing (NLP) functions such as tokenisation, sentence splitting, POS-tagging and lemmatisation allowing the identification of sentences. A binary knowledge engineering classifier classifies the sentences as requirements or non-requirements based on prescriptive word matching (shall, must, should, require, have to, need, want, desire, etc.)[3]. As previously explained, any XML parser can do the job for a semi-structured SysML specification. The extraction of textual cross-references to applicable documents is an information extraction problem. We can solve this challenge by building a probabilistic sequence model that is either discriminative such as Conditional Random Fields (CRF) and Maximum Entropy Markov Models (MEMM), or discriminative such as Hidden Markov Models (HMM). Stanford CoreNLP and the Java library Mallet[14] embed the CRF algorithm. A feature function that takes in as input a requirement statement, the position i of a term in the current statement, the class of the current token, and the class of the previous token can be used to build a linear-chain CRF. By defining two classes, *class A* corresponding to strings consisting in a blend of a few capital letters and digits such as "ISO900" or "CS25" standing for prescriptive document titles, and *class B* corresponding to chunks usually used to refer to content in external documents such as "in accordance with", "as specified in", "as set out in", it is likely that a term belonging to *class A* in the sequence model will be a cross-reference given the fact that the previous term belongs to *class B*. In other words, if we find a chunk such as "as specified in" followed by a term that mixes a few capital letters and digits such as "CS25", then we infer that the second term is a reference to an external document. Once extracted, cross-references are displayed to require the user to upload the referred applicable documents which, in their turn, will be processed to extract the requirements they prescribe. While uploading the required applicable documents, the user can manually set a binary (*legal* vs. *non-legal*) switch so as to specify whether it is a legal document. If it is set as a legal document, then the extracted requirements are classified as key legal requirements.

To diagnose quality defects (e.g. redundancies, contradictions, ambiguities) that pollute specifications and often lead to wrong decisions we use natural

[10] https://tika.apache.org/.
[11] https://poi.apache.org/.
[12] http://www.jdom.org/.
[13] http://nlp.stanford.edu/software/corenlp.shtml.
[14] http://mallet.cs.umass.edu/.

language processing techniques. To identify redundancies and contradictions we, at first, carry out a thesaurus-based lexical semantic disambiguation of keywords (nouns, verbs, adjectives and adverbs). The identification of the meaning of keywords enables us to reliably select a subset of similar requirements thanks to a sentence-level similarity function. Then, we use the POS-tagger of Stanford CoreNLP to find negations (e.g. no, not, not, 't, etc.) and numerical values (e.g. 40, 50) in the statements, and we look for the antonyms of each keyword by querying the thesaurus WordNet[15]. An analysis of the numerical dependencies in each requirement with the Stanford CoreNLP dependency analyser[16] enables us to only keep the statements that include digital values related to a physical dimension (e.g. 40 N or 50 K). With ad-hoc algorithms we build: (1) pairs of requirements that contain physical numerical values, (2) pairs of requirements with a negation and requirements without a negation, and (3) pairs of requirements whose keywords are antonyms. Finally, we combine the similarity function with these numerical, negation and antonymy pairs. If a pair of requirements has a similarity score that is higher than the threshold (e.g. 0.8), then there is potentially a contradiction or a redundancy. Such an approach increases recall rather than accuracy. Interactive visuals representing clusters of similar and potentially contradictory requirements facilitates the cleaning of quality defects. This interactive *quality view* also depicts all the defects due to the infringement of best practices (e.g. ambiguities, incompleteness, etc.) in requirement writing.

Well-defined requirements are the basis for inferring new implicit attributes that help a domain-expert to retrieve requirements that belong to his domain of expertise. One attribute is the business category (mechanical, IT, etc.) a given requirement belongs to. This attribute can be inferred with supervised statistical learning text categorisation algorithms. The Support Vector Machine (SVM) algorithm is usually recognised as the best algorithm for text categorisation. Furthermore, Knauss and Ott (2014) demonstrated that a semi-supervised SVM model outperforms a supervised approach [18]. However, instead of building a time-consuming training set with annotated requirements as was done in [18], we propose to train a multi-class SVM model with sentences from domain-specific (mechanics, electronics, etc.) dictionaries and handbooks. In addition, either a formal linguistic analysis implemented with the syntactic meta-language Backus-Naur Form (BNF)[16] or a supervised machine learning decision tree [23] can be implemented to infer another implicit attribute that categorises the statements as *functional* or *non-functional*. These attributes coupled with the Apache Lucene[17] search engine, and the visualisation of semantic relationships (synonyms, hypernyms, hyponyms, etc.) among requirements are functionalities of the *survey view* that enable domain experts to retrieve the most relevant requirements according to their profile during the estimation phase.

Once assessed, quantitative decision-making criteria can be analysed with principal component analysis, whereas qualitative variables can be analysed

[15] https://wordnet.princeton.edu/.
[16] http://nlp.stanford.edu/software/stanford-dependencies.shtml.
[17] https://lucene.apache.org/core/.

with multiple correspondence analysis. A *decision-making view* communicates the results of the statistical multivariate analysis by uncovering unanticipated interesting patterns in the data set. These patterns ease the identification of a subset of added-value requirements that motivate business analysts' decisions.

4 Conclusion and Future Work

In this paper, we have presented a collaborative requirement mining framework that supports OEMs willing to distil a large set of requirements. Its computational capabilities and interactive visuals should help analysts to gain insight and discover opportunities in a massive set of requirements. Computational capabilities include the automated extraction of requirements and relationships, the inference of implicit attributes, the analysis of requirements' quality, a statistical multivariate analysis, and the capitalisation and reuse of information and decisions made. On the other hand, five interactive visuals should improve exploration through contextual views illustrating requirements and their interdependencies and multidimensional views depicting the numerous attributes that make up each requirement and relationship. Furthermore, the originality of this framework is to focus on the contractual phase that brings a system integrator and OEM(s) in one single collaborative digital environment.

The prototyping of a web application is under development. The MVC design pattern JSF 2[18] has been selected. The graph database Neo4j[19] has been chosen to store requirements and relationships in a property graph data model. The Object-Graph Mapping Spring Data Neo4j[20] makes it possible to work with annotated POJO entities. Finally, the JavaScript library D3.js[21] has been adopted to create the web-based interactive visuals.

References

1. Houdek, F.: Managing large scale specification projects. In: 19th International Working Conference on Requirements Engineering Foundation for Software Quality, REFSQ 2013, Essen, Germany, 8–11 April 2013 (2013)
2. The Standish Group: Chaos manifesto report (2013)
3. Sawyer, P., Rayson, P., Garside, R.: REVERE: support for requirements synthesis from documents. Inf. Syst. Front. J. **4**(3), 343–353 (2002)
4. Thomas, J., Cook, K.: Illuminating the Path: Research and Development Agenda for Visual Analytics. IEEE Press, Los Alamitos (2005)
5. Reddivari, S., Rad, S., Bhowmik, T., Cain, N., Niu, N.: Visual requirements analytics: a framework and case study. Requir. Eng. **19**(3), 257–279 (2014)
6. Cooper, Jr., J.R., Lee, S-W., Gandhi, R.A., Gotel, O.: Requirements engineering visualization: a survey on the state-of-the-art. In: 4th International Workshop on Requirements Engineering Visualisation, pp. 46–55. IEEE, Altanta (2010)

[18] https://javaserverfaces.java.net/.
[19] http://neo4j.com/.
[20] http://projects.spring.io/spring-data-neo4j/.
[21] http://d3js.org/.

7. Coatanéa, E., Mokammel, F., Christophe, F.: Requirements models for engineering, procurement and interoperability: a graph and power laws vision of requirements engineering. Technical report, Matine (2013)
8. Lash, A.: Computational representation of linguistics semantics for requirements analysis in engineering design. MSc thesis, Clemson University (2013)
9. Zeni, N., Kiyavitskaya, N., Mich, L., Cordy, J.R., Mylopoulos, J.: GaiusT: supporting the extraction of rights and obligations for regulatory compliance. Requir. Eng. **20**(1), 1–22 (2015)
10. de Marneffe, M., Rafferty, M., Manning, C.: Finding contradictions in text. In: 46th Annual Meeting of ACL, Columbus, OH, pp. 1039–1047 (2008)
11. Carlson, N., Laplante, P.: The NASA automated requirements measuring tool: a reconstruction. Innov. Syst. Softw. Eng. **10**(2), 77–91 (2014)
12. Christophe, F., Mokammel, F., Coatanéa, E., Nguyen, A., Bakhouya, M., Bernard, A.: A methodology supporting syntactic, lexical and semantic clarification of requirements in systems engineering. Prod. Dev. **19**(4), 173–190 (2014)
13. Génova, G., Fuentes, J.M., Llorens, J., Hurtado, O., Moreno, V.: A framework to measure and improve the quality of textual requirements. Requir. Eng. **18**(1), 25–41 (2013)
14. Kiyavitskaya, N., Zeni, N., Mich, L., Berry, D.M.: Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. Requir. Eng. **13**(3), 207–239 (2008)
15. Körner, S.J., Brumm, T.: Natural language specification improvement with ontologies. Semant. Comput. **3**(4), 445–470 (2009)
16. Lamar, C.: Linguistic analysis of natural language engineering requirements. MSc thesis, Clemson University (2009)
17. Yang, H., de Roeck, A., Gervasi, V., Willis, A., Nuseibeh, B.: Analysing anaphoric ambiguity in natural language requirements. Requir. Eng. **16**(3), 163–189 (2011)
18. Knauss, E., Ott, D.: (Semi-) automatic categorization of natural language requirements. In: Salinesi, C., van de Weerd, I. (eds.) REFSQ 2014. LNCS, vol. 8396, pp. 39–54. Springer, Heidelberg (2014)
19. Zhang, A., Auriol, G., Eres, H., Baron, C.: A prescriptive approach to qualify and quantify customer value for value-based requirements engineering. Comput. Integr. Manuf. **26**(4), 327–345 (2014)
20. Achimugu, P., Selamat, A., Ibrahim, R., Mahrin, M.N.: A systematic literature review of software requirements prioritization research. Information Softw. Technol. **56**(6), 568–585 (2014)
21. Felfernig, A., Ninaus, G., Grabner, H., Reinfrank, F., Weninger, L., Pagano, D., Maalej, W.: An overview of recommender systems in requirements engineering. In: Maalej, W., Thurimella, A.K. (eds.) Managing Requirements Knowledge, pp. 315–332. Springer, New York (2013)
22. Cheung, J., Wong, J., Forrester, J., Eres, H.: Application of value-driven design to commercial aero-engine systems. In: 10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, Fort Worth, TX (2010)
23. Hussain, I., Kosseim, L., Ormandjieva, O.: Using linguistic knowledge to classify non-functional requirements in SRS documents. In: Kapetanios, E., Sugumaran, V., Spiliopoulou, M. (eds.) NLDB 2008. LNCS, vol. 5039, pp. 287–298. Springer, Heidelberg (2008)