A requirement mining framework to support complex sub-systems suppliers

Romain Pinquié, Philippe Veron, Frédéric Segonds, Nicolas Croué

Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 70 (2018) 410–415

www.elsevier.com/locate/procedia

28th CIRP Design Conference, May 2018, Nantes, France

A requirement mining framework to support complex sub-systems suppliers

Romain Pinquié[a,*], Philippe Véron[a], Frédéric Segonds[b], Nicolas Croué[c]

[a] *Arts & Métiers ParisTech, LSIS UMR CNRS, Aix En Provence, France*
[b] *Arts & Métiers ParisTech, LCPI, Paris, France*
[c] *Sogeti High Tech, Toulouse, France*

* Corresponding author. Tel.: +33-658-336-305; fax: +0-000-000-0000. *E-mail address:* romain.pinquie@ensam.eu

Abstract

The design of engineered socio-technical systems relies on a value chain within which suppliers must cope with larger and larger sets of requirements. Although 70 % of the total life cycle cost is committed during the concept phase and most industrial projects originally fail due to poor requirements engineering [1], very few methods and tools exist to support suppliers. In this paper, we propose to methodologically integrate data science techniques into a collaborative requirement mining framework to enable suppliers to gain insight and discover opportunities in a massive set of requirements. The proposed workflow is a five-activity process including: (1) the extraction of requirements from documents and (2) the analysis of their quality by using natural language processing techniques; (3) the segmentation of requirements into communities using text mining and graph theory; (4) the collaborative and multidisciplinary estimation of decision making criteria; and (5) the reporting of estimations with an analytical dashboard of statistical indicators. We conclude that the methodological integration of data science techniques is an effective way to gain insight from hundreds or thousands of requirements before making informed decisions early on. The software prototype that supports our workflow is a JAVA web application developed on top of a graph-oriented data model implemented with the NoSQL NEO4J graph database. As a future work, the semi-structured as-required baseline could be a sound input to feed a formal approach, such as model- and simulation-based systems engineering.

Peer-review under responsibility of the scientific committee of the 28th CIRP Design Conference 2018.

Keywords: Requirement; Specification; Data mining; Decision-Making

1. Introduction

[CONTEXT] Whether it is in our daily life or in our professional environment, information is systematically digitalized. The concept of the division of labour illustrated by Adam Smith's story of the pin-maker in his *magnum opus – The wealth of Nations* – is in today's industries supported by a digital twin that stands as the single source of truth for collaboration, both within and across functional areas of the extended enterprise from inception to disposal. Nevertheless, the so-called digital twin and Industry 4.0 trends, which are highly encouraged by economic incentives, hide inequalities. Indeed, the degree of digitalization during the beginning-of-life and end-of-life phases is rudimentary compared to the activities belonging to the middle-of-life phase, that is, design and manufacturing. For instance, the methods and tools supporting the specification of complex socio-technical systems have not significantly evolved: customers and suppliers exchange digital documents or extracts from database. It is rather surprising that requirements engineering does not receive more attention since literature agrees that most projects fail due to poor requirements engineering capabilities and that 70 % of the total life cycle cost is committed during the concept phase.

[PROBLEM] Companies providing Product Lifecycle Management (PLM) consulting and integration services have diagnosed that most of their clients must cope with large, unintelligible sets of requirements, and even more so for the sub-systems suppliers who take OEM's specifications over. To illustrate, the specification of a given system-of-interest requires several hundreds or thousands of requirements: up to 1300 in Ericsson Microwave Systems, 10 000 in Bosh and Sony Ericsson, or 50 000 in Mercedes-Benz. The problem of

2212-8271
Peer-review under responsibility of the scientific committee of the 28th CIRP Design Conference 2018.
10.1016/j.procir.2018.03.228

suppliers is two-fold. On the one hand, they struggle to gain insight and discover opportunities in a massive set of text-based requirements. On the other hand, they must (re-)structure the set of requirements according to their own requirements engineering practices.

[CONTRIBUTION] The existing commercial solutions do not tackle the aforementioned problems, they mainly focus on the management (maturity, change and diversity management, traceability with other engineering artefacts, etc.) of the requirements within a database. We consider the large sets of text-based requirements as Big Data and we assume that:

> ***[HYPOTHESIS]*** *The methodological integration of data science techniques into a collaborative requirement mining framework is an effective way to gain insight from hundreds or thousands of requirements.*

Starting from a set of specification documents, our framework aims at exploring and structuring the requirements before exporting an *as-required* configuration baseline. The *as-required* configuration baseline, which complies with the ReqIF exchange format, feeds a requirements management tool (e.g. IBM DOORS) or a Product Lifecycle Management (PLM) tool (e.g. ENOVIA, TEAMCENTER). The workflow supported by our JAVA Web application is a five-activity process including: (1) the extraction of requirements from documents and (2) the analysis of their quality by using natural language processing techniques; (3) the segmentation of requirements into communities using text mining and graph theory; (4) the collaborative and multidisciplinary estimation of decision making criteria; and (5) the reporting of estimations with an analytical dashboard of statistical indicators. An underlying data model based on the mathematical concept of graph supports the structuring of the requirements and their dependencies as well as basic Create, Read, Update, Delete (CRUD) operations.

2. Related works

Requirements engineering is usually split into two activities – development and management – which are supported by numerous technologies.

Nevertheless, such a dichotomy hides the problems which arise between an OEM and a sub-system supplier during the contractual phase, especially the problem of large sets of requirements. There are various reasons why the number of requirements endlessly increases, but a sub-system supplier cannot directly act on most of them. For instance, he cannot require from the OEM to simplify the system-of-interest or to reduce its diversity if the system is tailored to meet specific customer requirements. He can neither expect less legal texts and certification guidelines from regulation authorities, nor a simplification of the processes and organizations. All these factors tend towards more and more requirements. We should therefore wonder what we can do to tackle this common problem.

We distinguish two main approaches to deal with the staggering increase of requirements: formal *vs.* mining. On the one hand, we can concentrate on the main cause of requirements mushrooming which is poor text-based requirements. Indeed, most requirements are ambiguous, often reused statements which paraphrase design solutions. Thus, a promising alternative to limit the number of requirements is to adopt rigorous engineering methods and tools. A sort of "lean engineering". For instance, rather than writing massive specification documents or filling database with thousands of prescriptive sentences, one can prefer formal concepts, such as formal languages [2], goal-oriented approaches [3], logical models [4] or the demonstration of proof properties [5]. Encouraging, not to say forcing, engineers to deeply think about the necessary content of a requirement before giving a parsimonious definition of it is one way to reduce the number of requirements. However, sub-system suppliers cannot force OEMs to specify their systems using formal methods, they are on their own. Moreover, formal methods are of interest for complex dynamic systems, but they do not bring any added-value to specify other system properties (structure, safety, maintainability, quality, manufacturing, aesthetics, etc.) so far.

Alternatively, as for any problem, one can focus on the symptoms rather than the causes. Thus, a sub-system supplier can relinquish control on the form of the requirements delivered by an OEM and use data science techniques to explore a large set of messy requirements. An extensive number of data science techniques have been applied to requirements engineering. Natural language processing has been used to extract requirements from documents [6,7] and detect quality defects [8,9]. Rules-based and statistical learning-based text mining algorithms have also been used to cluster requirements into communities [10]. The analysis of requirements is not only purely computational, but also graphical. Indeed, data visualisation techniques have shown to be useful to explore clusters of requirements [11] and interdependencies between them [12].

An extensive literature review of the requirements engineering body of knowledge results to the following conclusions:

- There is a bewildering array of academic and commercial methods and tools to develop and manage requirements, but none of them support sub-system suppliers who must cope with large sets of requirements.
- Suppliers cannot force OEMs to use formal approaches, they must therefore help themselves to discover opportunities and to make informed decisions (e.g. Bid / No Bid).
- Data science techniques proved to be efficient to structure and explore massive specifications. However, studies strive to improve a specific feature of requirements engineering rather than propose an integrated environment that supports a workflow.

In the next section, we propose our requirement mining framework that improves the lot of sub-system suppliers.

3. A requirement mining framework

In this section, we describe our requirement mining framework from three perspectives:

- The *operational view - What the stakeholders must be able to do with the framework?*. It is a "Black-Box" definition of the framework. It identifies the stakeholders, the main

services the framework provides to them, as well as the external constraints which bound the solution space.

- The *functional view - What the framework must do so that the stakeholders can achieve their missions?* It is a decomposition of services into technical functions and exchanged data flows which can subsequently be integrated to provide the main services to the stakeholders.
- The *software view - How the framework is doing it?* It is a detailed definition of the data model, algorithms and technologies that make up our framework.

3.1. Operational view: Stakeholders, Services, Constraints

As in most collaborative software application, an *Administrator* of the database manages the projects and the associated *Managers*. The *Manager* of a project has an adjudicative role – e.g. a *Bid/No Bid* decision. Once the *Administrator* has created a project and the *Manager* associated to it, the latter creates an *Analyst* and a set of *Experts*. The *Analyst* collects and uploads the set of prescriptive documents that applies to the project. After the extraction of requirements from documents, the *Analyst* cleans the quality defects detected by the framework and which are very likely to lead to risky misunderstandings. For instance, the estimation of the cost and the time to develop a requirement that does not prescribe a minimum and a maximum level of performance is very likely to be inaccurate. Decisions within a company are often made by subject-matter experts rather than a multidisciplinary group of stakeholders. This framework is the opportunity to give the floor to each expertise and finally make an informed decision which emerges from a collaborative consensus. Thus, the *Manager* of a given project creates a set of *Experts* whose role is to estimate decisions making criteria (cost, time, risk, etc.) associated to clusters of requirements. Each *Expert* is defined by two attributes: its domain of expertise (mechanics, electronics, etc.), and its level of experience (junior or senior) to weight estimations.

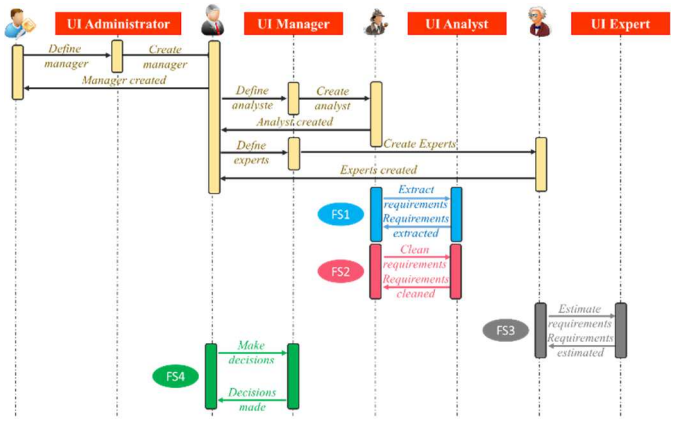

Fig. 1. Logical architecture with the logical entities and I/O information flows

Two design constraints must be fulfilled. First, the framework must be web-based as it is intended to support collaborative and geographically dispersed companies. Second, the *as-required* baseline exported from the framework must comply with the standardised ReqIF [13] exchange format to feed various requirements management or PLM tools.

3.2. Functional view: Technical functions

To provide the desired services to the stakeholders, the framework shall perform a set of technical functions enumerated in Table. 1.

Table 1. Technical functions (FTi) resulting from services (Si)

S1. To extract requirements	
	FT 11. To extract from unstructured documents.
	FT 12. To extract from semi-structured documents.
	FT 13. To detect cross-references.
S2. To guarantee reliable requirements interpretation	
	FT 21. To detect and remove intrinsic quality defects.
	FT 22. To detect and remove relational quality defects.
	FT 23. To create requirements context.
S3. To estimate requirements	
	FT 31. To define decision making criteria.
	FT 32. To segment requirements.
	FT 33. To estimate decision making criteria.
S4. To make informed strategic decisions	
	FT 41. To analyse estimations.
	FT 42. To simulate "what-if" analysis.

3.3. Software view: Web application prototype

In this section, we allocate each technical function onto a software building block solution.

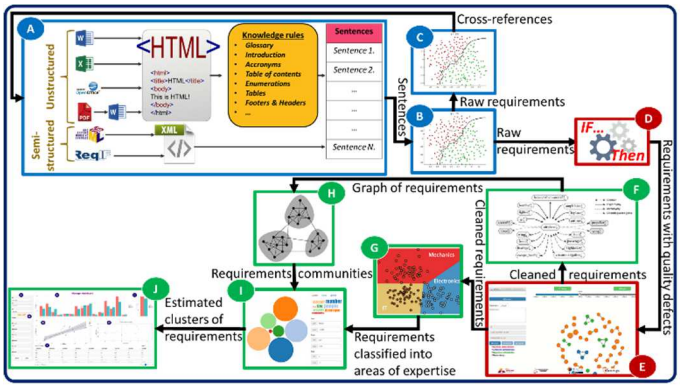

Fig. 2. Logical architecture with the logical entities and I/O information flows

FT11 is implemented by a natural language processing pipeline (Fig. 2.A) and a rules-based classifier (Fig. 2.B). The former extracts sentences from unstructured (Word, OpenOffice, etc.) specifications and prescriptive verbs (e.g. require, shall, must, need, want, desire, expect, etc.) that enable the classifier to distinguish prescriptive from descriptive statements. FT12 is satisfied by an XML/XMI parser (Fig. 2.A) that extracts the requirements from semi-structured (SysML, ReqIF, etc.) specifications. Both implementations have been thoroughly discussed in [14] and tested on two specifications containing 1368 requirements in total. We obtained 0.86 and 0.95 of precision and recall, respectively. These promising results is a first demonstration of the efficiency of our framework to automatically collect large sets of requirements.

A prescriptive statement often refers to another prescriptive document which also contains applicable requirements. For

instance, a requirement that refers to a standard. We have developed a machine learning-based classifier (Fig. 2.C) that takes the requirements and filters out a subset of requirements containing cross-references (FT13). Thus, for each requirement referring to an external document, the analyst can upload the referenced document, which in turn will be processed to extract the requirements it contains before linking them to the original requirement. Alternatively, if there is a false positive result, that is, a requirement that does not contain a cross-reference, then the analyst can simply remove it from the list. The cross-reference interdependencies are of interest to cluster requirements and navigate through them. We consider the problem of cross-references detection as a classification problem: does the requirement contains a cross-reference? To solve the classification problem, we developed a supervised machine learning-based classifier. The vector of features is as follows, is there:

- A word that refers to an external source? *e.g. specify, detail, define, accordance, set, comply, agreement, compatible, conform, refer, as per.*
- A proposition? *e.g. as, in, at, under, with, to, herein.*
- A structural element? *e.g. paragraph, §, chapter, section.*
- A mix of digit and characters? *e.g. ISO9001.*
- Multiple capital characters? *e.g. ISO.*
- A prescriptive document? *e.g. standard, policy, regulation, guideline, law.*
- A term that is not a WordNet thesaurus entry? *e.g. ISO9001.*
- A standard acronym? *e.g. ISO, IEEE, ECSS, IEC, CS, ESA, CEN, ETSI.*
- A term that contains multiple full stops? *e.g. 3.10.2.11.*
- A term that contains a dash? *e.g. ECSS-4-40.*
- Multiple brackets? *e.g. (, [, {.),], }.*

Fig. 3. Clusters of requirements containing quality defects

To train and test our machine learning classifier, we hand-crafted a training set of 500 requirements based on numerous industrial specifications. Among the 500 requirements, 270 contain a cross-reference. We applied a 10-fold cross validation using five learning algorithms (naive Bayes, decision tree J48, logistic regression, support vector machine sequential minimal optimization, and neural network virtual perceptron) with the Weka API embedded in our framework. Initial results show that SVM outperforms other alternatives with an accuracy of 91.2 %, a precision and a recall of 0.912, and a ROC area of 0.911. To make sure that the model was not overfitting the data, we replaced the initial linear function of SVM by a 2nd order and 3rd order polynomial kernel function. Results are slightly improved with an accuracy of 92% and 91.4%. Since results with more flexible functions are not plumed, we can conclude that the model does not overfit the data. Moreover, since the results are not significantly improved with a more flexible function, we can conclude that our binary classification problem is linearly separable. To appreciate the usefulness of such a capability, we encourage readers to think about the manual activity of reading thousands of pages to just identify applicable external documents which must be considered too.

Once all requirements have been extracted, we must make sure that their interpretations remain reliable not to harm subsequent experts' estimations and manager's decisions (FT21, FT22 and FT23). We consider three sources of ambiguities. First, there are intrinsic defaults, such as incomplete sentences, vague terms, or connectors (and, or, /, etc.). These defaults of quality are detected by checking a range of best practices writing rules (Fig. 2.D). Defects are presented to the analyst under the form of clusters (Fig. 3) expandable by a simple click leading to a tabular form of the underlying requirements. The analyst can directly clean or ignore the highlighted defects. Second, there are relational defects, that is, contradictions or redundancies between requirements. We can never be sure that a relational defect exists, we can only detect very likely ones. To detect redundancies, we calculate a similarity score between each pair of requirements. Highly similar requirements are very likely to be redundant or contradictory. To calculate the similarity score, we build a keyword-requirement matrix for each pair of requirements and calculate the cosine between both vectors. Keywords are lemmas of nouns, verbs, adjectives, adverbs and their synonyms queried from the WordNet thesaurus. Indeed, synonyms must be considered as exact-match would degrade performance. Nevertheless, before expanding a keyword with its synonyms, we must disambiguate it to avoid irrelevant terms. We assume that a word has a unique meaning in a given document. For instance, in a document, the term *bank* means a financial institution or a sloping land but not both. Thus, for each sense of a lemma, we calculate the intersection of the document keywords and the Synset (Synset: a set of synonyms associated to the meaning of a word) of each sense. The Synset that has the highest number of terms in common with the vector of document keywords is used to expand the keyword. So far, natural language processing tasks do not deal with syntactic and spelling mistakes. Some highly similar requirements may be contradictory rather than redundant. Among the different kind of contradictions, we focus on the ones due to antonyms, numerical values, and negations. If two highly similar requirements contain antonyms or numerical values, then they are linked by a contradiction. Antonyms and numerical values are identified with WordNet and CoreNLP, respectively. If a pair of requirements has a high similarity score and that only one contains a negation, then they are linked by a contradiction too. Negations are also detected with the CoreNLP library. If two similar requirements do not have a contradiction feature (antonym, negation, numerical value), then they are linked by a redundancy relationship. An analyst removes the relational defects by interacting with a graph where nodes are

requirements and edges are redundancies or contradictions (Fig. 2.E). It is realistic to accept that the automatic detection of intrinsic and relational defects in a large set of requirements is much more efficient than manual reviews.

Even after removing intrinsic and relational defaults, requirements remain ambiguous. To guarantee reliable interpretations, we suggest creating a requirement context (FT23). The context of a requirement includes: the document context, the cross-references context, as well as the semantic and conceptual context. The document context is the set of sentences that precedes and follows the requirements in the document and which improves the interpretation. When clicking on a requirement, a PDF version of the original document pops up so the user can directly read the relevant section. The cross-references context is the requirements linked by cross-reference relationships. Finally, the semantic (e.g. aircraft → plane) and conceptual (e.g. plane → airfield) context is defined by semantic and conceptual relationships between requirement keywords (Fig. 2.F). Semantic relationships are based on the WordNet thesaurus, whereas conceptual ones come from the ConceptNet 5 ontology. These relationships ease the structuring and exploration of requirements. The graph data model (Fig. 4) sums up the node and relationship objects.

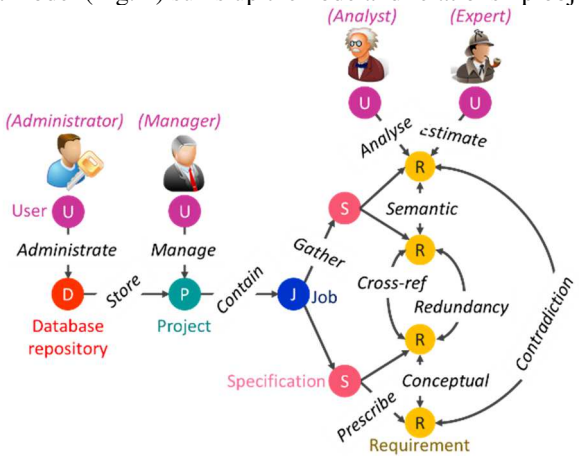

Fig. 4. Graph-oriented data model without node and relationship properties

Unambiguous requirements make a good input to estimate decision making criteria (FT31) before making informed decisions. There is no fewer than 280 decision making criteria associated to requirements [15]. We cannot select the criteria that suits us best as each company will have its own decision-making problem and a different business strategy (e.g. reuse with limited investments *vs.* innovate with substantial investments). We therefore let the manager defines the set of criteria he wants for each new project he is in charge of.

Since we deal with hundreds or thousands of requirements, we cannot reuse basic prioritization methods that consist in estimating criteria for each requirement or making pairwise comparisons. Thus, to make the activity scalable, we propose to associate the decision-making criteria to communities of requirements (FT32). We have studied three different alternatives to detect communities. The first one, which consists in classifying requirements into topics – mechanics, electronics, IT, etc – by using a supervised machine learning-based classifier (Fig. 2.G) has been presented in [14]. The second alternative is an unsupervised machine learning-based classifier. Unsupervised because we do not predefine a list of categories but search for dissimilar clusters. We used the LinLog algorithm from the Carrot2 API and applied it to 1618 requirements collected from three specifications. We obtained 109 clusters and could therefore expect approximately 15 requirements within each cluster. However, results show that the biggest cluster contains 169 requirements, whereas the smallest one contains 2 requirements. Moreover, there is a cluster named “Other Topics” that contains 436 requirements, that is, ¼ of the data. If clusters contain more than a hundred of requirements, we do not solve the issue of large sets of requirements. Additionally, with the LinLog algorithm, a requirement may belong to several clusters. Thus, experts may overestimate the decision-making criteria. Finally, when we carefully look at the cluster labels, we notice that clusters are based on sequences of terms such as “boxes shall”, “conduit shall”, “system shall”, etc. In our specific application, such assumption is irrelevant as it corresponds to co-occurrences of “Subject + Modal verb”. Clustering of requirements have been extensively studied [10]. However, most approaches are purely text-based. We believe that a community of requirements should not only include requirements that share linguistic affinities, but also requirements that are linked by other kinds of interdependencies, such as conceptual ones and cross-references. Any kind of relationship such as SysML ones (derive, trace, satisfy, etc.) can be relevant. We have therefore studied the clustering of requirements based on graph theory (Fig. 2.H). The basic approach for graph clustering is to detect local particular subgraphs (cliques, k-plexes, k-cores, k-components), but this would lead to relatively small clusters. The second approach, graph partitioning, is also inappropriate as we cannot guess the number and the size of communities *a-priori.* We therefore adopted the third approach – community detection – with which communities of requirements arise naturally from the graph topology. We used the Spectral algorithm proposed in [16] and implemented in the Jmod API.

Fig. 5. Clusters of requirements estimated (green) and to be estimated (red)

Experts estimate the criteria associated to the communities (Fig 2.I). We have preferred quantitative continuous criteria and requires a minimum and a maximum estimation by each expert (Fig. 5). A range is not only natural when we guess, but it also enables us to better appreciate the level of confidence.

To enable a manager to analyse the estimations (FT41), we propose an analytical dashboard (Fig. 2.J). In the panel A of

Fig. 6, the manager sets criteria to variables, e.g. X1 = Cost, X2 = Time, X3 = Risk. The interactive dashboard provides several descriptive statistical indicators 1-D (B and C), 2-D (D), or n-D (E). The red columns stand for the subject-matter experts (mechanics, electronics, etc.), whereas the green columns stand for the cross-functional experts (quality, marketing, etc.). In the panel F, for a given community, the manager can observe all estimations for each criterion. Finally, in the panel G, the manager can select a community and make his own estimation before simulating a *what-if* scenario (FT42). An *as-required* configuration can be exported as a standardised exchangeable ReqIF file to feed a requirements management tool.

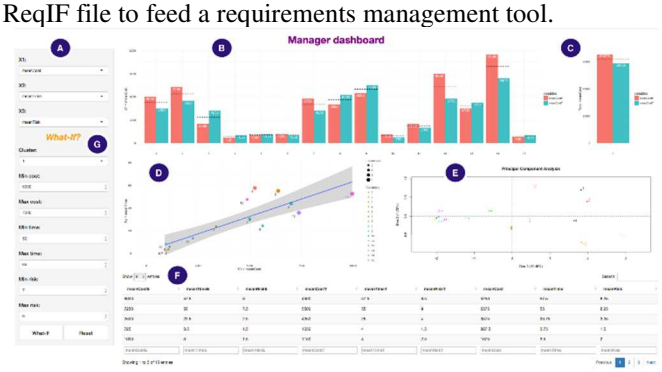

Fig. 6. Interactive analytical dashboard to make informed decisions

The prototype is a Java Web application (Fig. 7). The first layer, which enables users to interact with the software, is a Web interfaces including interactive visuals developed with D3.js. The domain layer includes all algorithms based on the various APIs previously discussed. Data are created, read, updated and deleted with the Data Access Object (DAO) Spring Data Neo4J library which interacts with the database layer implemented with the Neo4J NoSQL graph-oriented database.

Fig. 7. Architecture of the JAVA Web application prototype

4. Conclusions

In this paper, we have presented a requirement mining framework that enables sub-system suppliers to gain insight and discover opportunities in a large set of requirements. The framework relies on a five-activity workflow: (1) the extraction of requirements from documents and (2) the analysis of their quality; (3) the segmentation of requirements into communities; (4) the collaborative and multidisciplinary estimation of decision making criteria; and (5) the reporting of estimations. Finally, an *as-required* configuration baseline can be exported to feed a requirements management tool or generate inputs for model- and simulation-based systems engineering.

> ***[ANSWER]*** *Results encourage us to conclude that the methodological integration of data science techniques into a collaborative requirement mining framework is an effective way to gain insight from hundreds or thousands of requirements. Indeed, it irrefutably avoids the monopolisation of resources (cost, time and experts) to carry out essential activities with low value (e.g. the extraction of requirements and the detection of poor requirements) and provides new means to gain insight into massive sets of requirements (e.g. the detection of implicit affinities, or the classification into topics and communities).*

The framework can be improved in several ways. For instance, by detecting other types of contradictions or by dealing with graphical requirements (e.g. charts). The implementation should also be based on a distributed computing architecture to improve the computational capabilities. State-of-the-art language models (GloVe, Word2Vec) should be considered. One could also investigate how the capitalized information can be systematically reused.

References

[1] INCOSE, Systems engineering handbook: a guide for system life cycle processes and activities, V4, 2014.
[2] Abrial J.-R. The B-book : Assigning Programs to meanings. Cambridge University Press, New York, NY, USA, 1996.
[3] van Lamsweerde A. Goal-oriented requirements engineering : a guided tour. Proceedings of the fifth IEEE international symposium on requirements engineering, August 27-31, p. 249–262, 2001.
[4] Pinquié R, Micouin P, Véron P, Segonds F. Property model methodology : a case study with modelica. In: Horváth I., Pernot J.-P., Rusák, Z. editors. Proceedings of the 12th international symposium on TMCE, Aix-en-Pce, France, May 9 – 13; 2016, p. 79–91, 2016.
[5] Liliegard M, Nilsson V. Model-based testing with Simulink Design Verifier: a case study on property proving and automatic test case generation for automotive embedded systems, master's thesis, Chalmers University of Technology, Göteborg, Sweden, 2014.
[6] Sawyer P, Rayson P, Garside R.. Revere: support for requirements synthesis from documents. Information Systems Frontiers, 4(3), p. 343–353, 2002.
[7] Bernard A, Coatanea E, Christophe F, Laroche F. Design : A key stage of product lifecycle. Procedia 24th CIRP design conference, 21 :3–9, 2014.
[8] Kiyavitskaya N, Zeni N, Mich L, Berry D.M. Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. Requirements Engineering, 13(3), p. 207–239, 2008.
[9] Yang H, De Roeck A, Gervasi V, Willis A, Nuseibeh B. Analysing anaphoric ambiguity in natural language requirements. Requirements engineering, 16(3), p. 163–189, 2011.
[10] Duan C, Laurent P, Cleland-Huang J, Kwiatkowski C. Towards automated requirements prioritization and triage. Requirements Engineering, 14(2), p. 73–89, 2009.
[11] Reddivari S, Rad S, Bhowmik T, Cain N, Niu N. Visual requirements analytics: a framework and case study. Requirements engineering, 19(3) P. 257–279, 2014.
[12] Heim P, Lohmann S, Lauenroth K, Ziegler J. Graph-based visualization of requirements relationships. Requirements Engineering Visualization, 2008, REV '08, p. 51–55, September 2008.
[13] Object Modeling Group. OMG Requirements Interchange Format V1.2 (OMG ReqIF), 2016.
[14] Pinquié R, Véron P, Segonds F, Croué N. Requirement mining for model-based product design. International Journal of Product Lifecycle Management, 9(4), 305-332, 2016.
[15] Riegel N, Doerr J. A systematic literature review of requirements prioritization criteria, p. 300–317. Springer International Publishing, 2015.
[16] Newman M. E. J.. Modularity and community structure in networks. Proceedings of the National Academy of Sciences, 103(23), p. 8577–8582, 2006.