Grant Agreement nº	732463
Project Acronym:	OpenReq
Project Title:	Intelligent Recommendation Decision Technologies for Community-Driven Requirements Engineering
Call identifier:	H2020-ICT-2016-1
Instrument:	RIA (Research and Innovation Action
Topic	ICT-10-16 Software Technologies
Start date of project	January 1st, 2017
Duration	36 months

D1.1 Knowledge Base for OpenReq related work

Lead contractor:	TUGraz
Author(s):	TUGraz, HITEC, UH, UPC, Qt
Submission date:	December 2017
Dissemination level:	PU

Project co-funded by the European Commission under the H2020 Programme.

Abstract: This document provides an overview of the OpenReq Knowledge Base including a summary report related to its deployment and population. Please note that we see this as an ongoing task for the whole duration of the project, i.e., the OpenReq Knowledge Base will be maintained over the whole project and beyond (with the goal to establish the Knowledge Base as a central information platform for topics related to recommendation and decision technologies in requirements engineering). Deliverable D1.1 is regarded as a first summary (at month M12) that will be continuously extended by all project partners. The OpenReq partners have defined the goal to contribute, besides other publications, an overview of the state-of-the-art in recommendation and decision technologies for requirements engineering in the form of a journal publication in the project year 3. The overall roadmap to achieve this objective is documented. We also want to emphasize that detailed literature reviews are included in the topic-specific deliverables.

Table of Contents

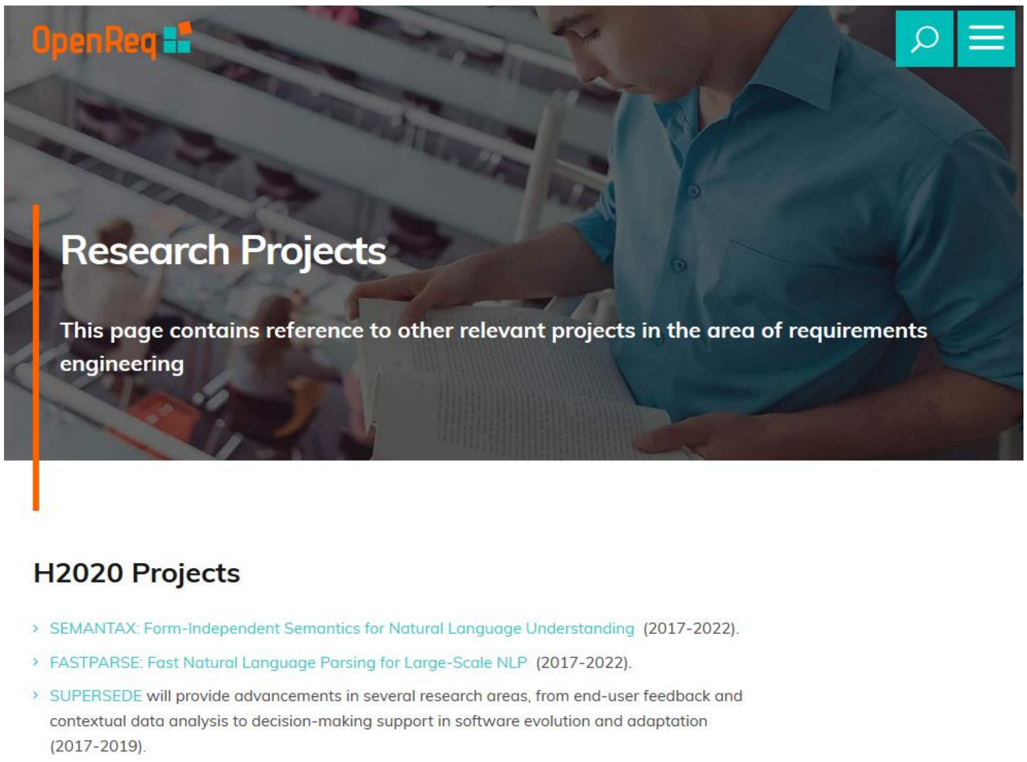
List of Figures

List of Tables

1. THE OPENREQ KNOWLEDGE BASE: OVERVIEW

The OpenReq Knowledge Base (see the screenshot in Figure 1) can be found at http://openreq.eu/knowledge-base/. It consists of the following sections[1]:

- **Communities & Initiatives:** This section provides a first overview of communities of interest including requirement engineering researchers and practitioners.
- **Journals, Conferences & Workshops:** This page contains links to journals, conferences, and workshops that are in the interests and scope of the OpenReq project.
- **Literature:** This section includes both publications from the OpenReq project and publications related to the topics of OpenReq, i.e., recommendation and decision technologies in requirements engineering.
- **Organizations & Tools:** This page contains references to organizations operating in the field of requirements engineering as well as relevant tools for researchers and practitioners. The "Tools" section is divided into following three subsections:
  - Requirement management tools
  - Eclipse-based requirement tools
  - Issue trackers
- **Research Projects:** This page contains references to other relevant projects in the area of requirements engineering.



Research Projects

This page contains reference to other relevant projects in the area of requirements engineering

H2020 Projects

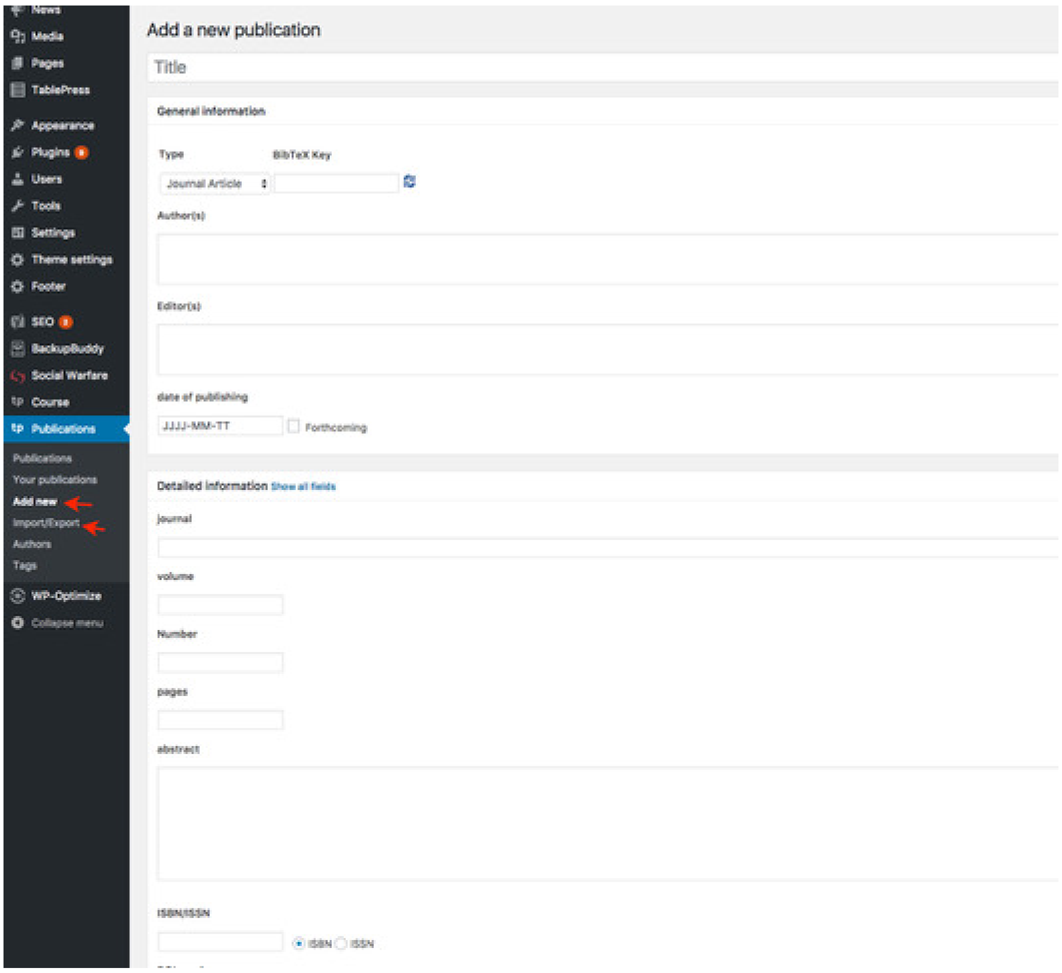
- SEMANTAX: Form-Independent Semantics for Natural Language Understanding (2017-2022).
- FASTPARSE: Fast Natural Language Parsing for Large-Scale NLP (2017-2022).
- SUPERSEDE will provide advancements in several research areas, from end-user feedback and contextual data analysis to decision-making support in software evolution and adaptation (2017-2019).

Figure 1: Screenshot of the OpenReq Knowledge Base.

[1] Sections and / or content will be extended over the whole project period and beyond

2. Technical Infrastructure of the Knowledge Base

The OpenReq Knowledge Base is integrated into the OpenReq web-page (www.openreq.eu) at a dedicated “Knowledge Base” section (www.openreq.eu/knowledge-base/). It is based on a Wordpress[2] instance and uses personalized features / plugins / extensions of Wordpress. All pages of the Knowledge Base (except for “Literature”) are Wordpress pages. Entries in the “Literature” page are searchable and exportable in BibTex format. Furthermore, “Literature” items can be added through the “Publications” menu on the left, either manually via “Publication → Add New" or by importing / pasting a BibTex via “Publication → Import/Export”. The following screenshot (Figure 2) illustrates how to add and / or import publications to the Literature part of the OpenReq Knowledge Base.



Type: Journal Article

date of publishing: JJJJ-MM-TT

Figure 2: Screen for adding new Publications to the Literature Section of the OpenReq Knowledge Base

[2] https://wordpress.com/

3. Roles of the OpenReq partners in populating the Knowledge Base

The OpenReq Knowledge Base has a central role in the OpenReq project - different OpenReq partners (listed below) are taking care of extending and maintaining different parts of the Knowledge Base. Updates are both done on a periodical fashion (i.e., every time a new paper has been accepted) by the responsible partners, and if a partner requests an update in a specific section.

- **Communities & Initiatives** → HITEC
- **Conferences & Journals** → UH
- **Literature** → TU Graz
- **Organizations & Tools** → vogella, Qt, WINDTRE, ENG
- **Research Projects** → UPC
- **Companies** → SIEMENS

4. Access to the OpenReq Knowledge Base

We will develop and enrich the OpenReq Knowledge Base over the whole project period and beyond. Our goal in this context is to establish the Knowledge Base as a central platform providing information on the state-of-the-art in requirements engineering research with a special focus on the recommendation and decision making issues. The OpenReq Knowledge Base is free to use for everyone via the project web-page (www.openreq.eu/knowledge-base/). Feedback can be given also via the project's web-page. The consortium plans to maintain active the Knowledge Base after the administrative end of OpenReq and will study alternatives to this effect, to be detailed as a part of the future OpenReq Exploitation Plan.

5. Publicity / Marketing

The OpenReq Knowledge Base is broadcasted to a wider audience through communication and dissemination activities of the project. The project communication and dissemination activities are described in general in Deliverable D8.1 Exploitation, Dissemination and Communication Plan.

The D8.1 Exploitation, Dissemination and Communication Plan outlines the external communication channels available to the project as the project website with its news section, social media channels (Twitter, Instagram, LinkedIn and YouTube), publications and reports and direct communications with stakeholders like the Advisory Board.

As the Knowledge Base is an expert and researcher-centric tool, the target audience is defined as researchers and requirements management professionals. Their interest in the Knowledge Base is easy access to essential requirements management related information.

The Knowledge Base will be a topic of a news post on the OpenReq project website. After the first year, news items about the different types of content in the Knowledge Base (as described in the overview section of this document) will be posted as news items. Also a bi-annual review of what has been added will be posted as a news item.

Items in the Knowledge Base will be posted as separate items in the project social media channels, for instance on Twitter and LinkedIn. As social media is more fast paced and the content is shorter, single items with links to the Knowledge Base will be posted.

The Knowledge Base should also provide easy ways to share links to the content in the Knowledge Base by the people using it. Methods to post links from the content of the Knowledge Base to social media sites like Twitter, Facebook and LinkedIn will be implemented. Organic sharing by users of the Knowledge Base is good for making the Knowledge Base more widely known.

These activities are in line with the overall communication and dissemination structure of the project defined in Work Package 8.

6. Recommendation and Decision Technologies for Requirements Engineering: State-of-the-Art

The overall goal of this section is to provide an overview of the state of the art in recommendation and decision technologies for requirements engineering. This overview will be adapted/extended within the scope of the first two years of OpenReq. Thereafter, we plan to transfer our documentation of OpenReq research results and related work into a corresponding journal publication (the concrete journal is not yet decided). In the following subsections we provide an overview of the state-of-the-art research that exists related to recommendation and decision technologies in requirements engineering. In the context of each subsection, we also discuss OpenReq research goals related to the advancement of the existing state-of-the-art.

6.1. Approach to state of the art survey

The literature section of the Knowledge Base will contain, among others, the primary studies used to review, in a systematic way, the main topics of the project, namely Requirements Intelligence, Group Decision Making, Recommendation Technologies in Requirements Engineering, Dependency Management, and Patterns and Requirements Reuse (see following sections).

In this section, we report an approach which represents a blueprint process for the review of the state-of-the-art. The example procedure presented below is based on the procedure illustrated in Petersen et al. [2015].

In particular, the approach is similar to a mapping of the literature in which the goal is to scope the research field and derive a research roadmap. The result of the systematic mapping study (SMS) offers an overview of the research area through a classification based on topics, research venues, and research institutions. Notably, the goal of the SMS is not to synthesize evidence in order to answer a specific research question (e.g., which requirements elicitation methodology works better) but rather to structure the research available in the field of requirements engineering according to the project topics.

The systematic process for collecting and analyzing the data (i.e., scientific literature) is presented in Figure 3.

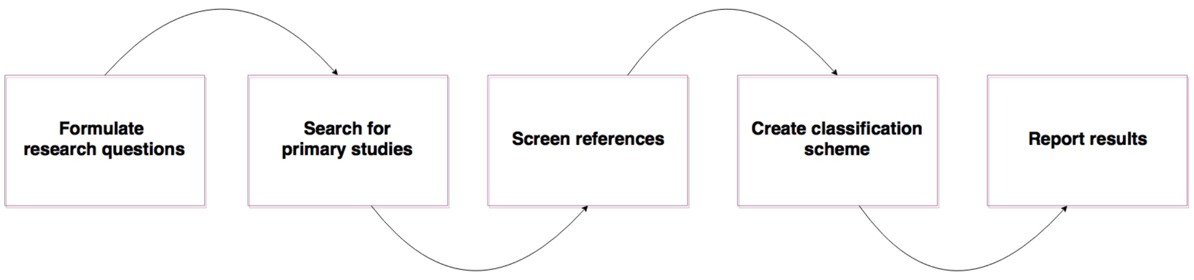

Formulate research questions
Search for primary studies
Screen references
Create classification scheme
Report results

Figure 3: Steps in the systematic mapping study process.

Formulate research questions
The goal is to identify and quantify the amount of research and the type of research performed on a topic during a specific time-span. The research questions are usually phrased as "What are the characteristics of X", "What is the trend of X over time", "How is X researched in the community".

Primary studies search
Primary studies are research papers presenting new results rather than reviewing existing one (e.g., literature surveys). Gathering a representative sample of primary studies, which will eventually be analyzed, is a fundamental step. Strategies to perform the search for primary studies are: i) automatic search using online databases (e.g., Scopus, Web of Science, IEEEXplore); ii) manual search in the proceedings of relevant conferences and journals; iii) forward and backward snowballing—i.e., leveraging the references contained in the set obtained from i) and ii) and their citations to include further publications. A recommended strategy to identify keywords to create a search string is the *PICO* approach [Kitchenham et al. 2007]. In particular, *P*opulation and *I*ntervention refers to the general research area (i.e., published studies in requirements engineering) and the second refers to the specific technology/procedure (e.g., recommender and decision making systems). At the current stage, as we want a broad overview of the field, we do not consider *C*omparison with other techniques/technologies and *O*utcomes (e.g., in terms of a specific construct, such as productivity). It is a good practice to let experts (e.g., participants in the project consortium) validating the set of primary studies (e.g., missing references).

Reference screening
This step deals with the application to inclusion and exclusion criteria to assure the quality of the studies that are used to answer the research questions. For example, one inclusion criteria for the reviews on the project topic would be that the paper explicitly mentions (e.g., in the title, abstract or introduction) requirements engineering as its focus and that it is published in a specific timeframe of interest. On the other hand, some criteria for excluding a paper are: i) it reports summaries of conferences or editorial; ii) it is not peer-reviewed; iii) it is not written in English; iv) it belong to grey literature (e.g., blog posts); v) its full-text is not available. For papers reporting the same study, only the most recent one is usually considered; likewise, several studies reported in a single paper are considered individually. It can be the case that criteria new emerge during the review process. Therefore, articles should be removed once they do not meet criteria at any point during the review.

Classification scheme
The classification scheme follows the research questions and it is based on the data extracted from the papers obtained after screening. The data extraction consists in reading the paper while looking for information mapped on the research questions. For example, i) the specific topics investigated in the paper; ii) the requirements engineering phases in which the topic is investigated; iii) the research methods used to investigate the topic; iv) the evaluation approach; v) the settings of the study. Once the data is collected, a general classification (e.g., by topic) can be used to answer the research question—i.e., by calculating the frequency of each identified topic. Or by creating a time-based classification where each year represents a class;

or, where each outlet (e.g., scientific conference, journal) is mapped to a class. However, it is possible to get other interesting insights based on the extracted data—e.g., papers reporting the use of recommender systems for requirement elicitation in a specific domain of interest for the project such as telco.

Reporting of results

The result of the classification above (also called mappings) are usually reported using, for example, bubble plots representing the frequency of papers belonging to one or several classes.

This makes it possible to easily visualize which classes have been emphasized in past research and which are not; therefore, showing possibilities for future research. A simpler way of reporting the results of a mapping is using frequency tables, although these are limited to 2-dimensions/classes at a time.

6.2. Requirements Intelligence

In OpenReq, we define requirements intelligence (RI) similar to business intelligence (BI). For BI, we use the definition of [Negash 2004]: “BI systems combine data gathering, data storage, and knowledge management with analytical tools to present complex internal and competitive information to planners and decision makers.” RI follows the same definition but focuses on requirements of software products. RI is the systematic collection, analysis, processing, and visualization of requirements and user feedback coming from natural text such as issue trackers, app reviews, or legacy requirements. The goal of RI is to provide new insights about software products based on diverse data sources of natural language and metadata. In order to describe the state of the art, we separate RI into the topics mining, classification, clustering, and visualization of requirements (a deeper literature analysis can be found in D2.1).

With **text mining**, we are able to retrieve requirements or related information to those in natural text. A survey by Mich et al. [2004] found that 79% of companies use unstructured natural language in their requirements documents; 16% use structured natural language (e.g., using templates); 5% use formal approaches. Therefore, the application of natural language processing (NLP) to requirements engineering has attracted a lot of attention from software engineering researchers and practitioners [Bates 1995]. Already in the late 1990s, NASA built a tool that leverages NLP for requirement engineering (Automated Requirements Measurement, or ARM) [Carlson and Laplante 2014]. In the area of software engineering, several approaches have been developed in the recent years with many focusing on app stores as shown in [Harman et al. 2012], [Johann et al. 2017], [Pagano and Maalej 2013]. An interesting source of information can be found in user reviews, comments, and social media in general, as people use these platforms to describe their desires and issues online [Guzman and Maalej 2014], [Williams and Mahmoud 2017], [Guzman et al. 2017].

The next step is the **classification of requirements**, which deals, among others, with the classification of requirements into functional (FR) and non-functional (NFR) as shown by the authors [Eckhardt et al. 2016], [Sommerville and Sawyer 1997], and [Kurtanovic and Maalej 2017]. In addition, RI will take advantage of the research done in the identification of bug reports, feature requests, and non-informative text in user reviews and tweets [Maalej et al.

2016], [Williams and Mahmoud 2017]. When RI successfully identified these classes, they have to be **clustered** to gain additional insights such as: how many people are having a similar issues, how many different issues and feature wishes exist, and which reports are simply duplicates. The identification of duplicates is an important task as it helps to reduce the number of open issues. Research has tackled this problem by introducing automated approaches for detecting duplicates [Wang et al. 2008], [Jalbert and Weimer 2008] and by analysing how harmful duplicates are [Bettenburg et al. 2008]. The processed data must then be **visualized** in order to give insights to stakeholders. [Abad et al. 2016] emphasize on the importance of the visualization in requirements engineering and present a systematic literature review (SLR) in which the authors analysed 26 primary papers. [Cooper Jr et al. 2009] present a survey on the state of the art in visualization in requirements engineering. They tackle the issue of effective visualization by surveying 29 research papers presented in requirements engineering Visualization (REV) workshops from 2006 to 2008. In their work they relate visualizations to 1) the lifecycle of requirements engineering including context and groundwork, structured specification, evolution, and maintenance and 2) to requirements engineering activities along the lifecycle such as elicitation, verification, and validation.

The use of NLP in the area of requirements engineering still has many open issues that we want to tackle within OpenReq. One of the open issues is the combination of different types of documents such as software feature descriptions and the feedback related to the software. For example, Johann et al. [2017] looked into the description of mobile apps and their reviews found in app stores. In OpenReq, we want to investigate further into the topic of connecting different types of documents by refining existing approaches and by looking into different domains investigating the impact of user feedback on the actual project (by e.g., finding issues reported by users and checking whether and at what point of time they were addressed by the developers). While one focus lies on the analysis of user feedback for example, in social media we also plan to work on intelligence approaches for legacy/conventional requirements documents. Part of this analysis will be the automatic classification of requirements into functional and non-functional, the identification of user rationale, and the automatic extraction of features discussed in these documents.

6.3. Recommendation Technologies in Requirements Engineering

The importance of requirements engineering has been illustrated by several studies (e.g., [Boehm and Basili 2001], [Arnuphaptrairong 2011] and [Calleam Consulting 2015]), which have shown that requirements-related issues, such as poorly specified requirements and incomplete and changing requirements, are the root cause of many failed projects. To address these problems, researchers and practitioners have applied recommenders systems to the requirements engineering stage. In this section, we briefly describe several works in which recommendation technologies have been applied to requirements engineering (see Table 1). These approaches have been identified while doing a systematic mapping. More details about the systematic mapping and these approaches can be found in deliverable D3.1.

Table 1: Recommendation technologies in requirements engineering: state-of-the-art summary

Activity	Recommendation	Recommendation approach	Reference
Elicitation & Specification	Requirements	Content-based filtering (based on clustering)	[Dumitru et al. 2011]
		Knowledge-based (based on domain ontologies)	[Kumar et al. 2010]
	Requirements (security approach)	Knowledge-based	[Romero and Richardson 2004]
	Requirements, Stakeholders	Collaborative filtering, Content-based filtering	[Castro-Herrera et al. 2009, 2010]
		Collaborative filtering, Social network analysis	[Lim and Finkelstein 2012]
Quality assurance	Completeness	Content-based filtering (based on clustering)	[Dumitru et al. 2011]
	Consistency	Knowledge-based	[Felfernig et al. 2013]
	Redundancy	Clustering	[Cleland-Huang et al. 2009]
Negotiation & Planning	Prioritization	Clustering	[Cleland-Huang et al. 2009]
Maintenance	Contributors	Collaborative filtering (based on Markov chains)	[Gaeul et al. 2009]
		Text mining (similarity)	[Weiß et al. 2007]
		Text mining (similarity)	[Nagwani and Verma 2012]
		Classification	[Lamkanfi et al. 2010]
		Classification (naive bayes)	[Cubranic and Murphy 2014]
		Classification (support vector machines)	[Anvik et al. 2006]
	Effort	Text mining (similarity)	[Olga and Cohen 2009]
	Requirements	Classification	[Antoniol et al. 2008]
	Severity level	Classification	[Fitzgerald et al. 2011]
		Classification (naive bayes)	[Di Luca et al. 2002]
		Text mining (machine learning)	[Menzies and Marcus 2008]

As can be seen in Table 1, the approaches about recommender systems in the field of requirements engineering deal with the following activities:

- Elicitation and Specification of requirements. In this case, the main aim is to recommend requirements that could be included in a software project (e.g., [Dumitru et al. 2011] and [Kumar et al. 2010]) and to recommend stakeholders that can contribute to the requirements (e.g., [Lim and Finkelstein 2012]). In the case of [Romero and Richardson 2004], recommendations are about a whole set of requirements that deal with the security of the system.
- Quality assurance of requirements. The goal of these approaches is to make recommendations that help to increase the quality of the requirements by making them more complete, more consistent or less redundant (e.g., [Dumitru et al. 2011] and [Felfernig et al. 2013]).
- Negotiation and Planning of requirements. The work here (i.e., [Cleland-Huang et al. 2009]) focuses on recommending prioritizations of the requirements to help in the negotiation and planning stage.
- Maintenance of requirements and feature requests. The first group of works in this activity aim to identify contributors that may help to solve feature requests (e.g., [Nagwani and Verma 2012] and [Cubranic and Murphy 2014]). Other works aim at predicting the effort necessary to solve a feature request or at identifying if a feature request corresponds to an actual requirement or to a bug ([Olga and Cohen 2009] and [Antoniol et al. 2008], respectively). Finally, other works aims at recommending the severity level of a feature request, so the ones with a highest severity are resolved earlier (e.g., [Di Luca et al. 2002] and [Fitzgerald et al. 2011]).

As for the recommendation approaches used, classical recommender systems technologies are used (such as collaborative filtering, content-based filtering and knowledge-based), usually changing a little bit the approach to adapt it to the requirements engineering context (e.g., [Dumitru et al. 2011], [Gaeul et al. 2009] and [Kumar et al. 2010]). However, other types of recommendation techniques, usually based in machine learning algorithms, are more prone in the area of requirements engineering. Specifically, these techniques are: text mining (i.e., processing unstructured textual information to extract meaningful numerical indices so it is accessible to the data mining algorithms) (e.g., [Menzies and Marcus 2008]), clustering (i.e., grouping of a particular set of objects based on their similar characteristics) (e.g., [Cleland-Huang et al. 2009]) and classification (i.e., identifying a class or category of an item) (e.g., [Cubranic and Murphy 2014]).

As presented in Table 1, almost all the proposed approaches focus on a specific RE task but do not support the needs of stakeholders in different requirements-related tasks through the different RE stages. OpenReq aims to incorporate recommendation technologies to support the stakeholders' needs in different RE stages. Specifically, the overall *research objective* of OpenReq is giving recommendations related to:

- *screening and recommendation of relevant requirements*: identification of actual requirements, i.e., to recognize text that contains "actual" requirements in contrast to the one that does not bring valuable information; identification of similar requirements, in the same project or from previous ones; and, finally, identification of related requirements.
- *the improvement of requirements quality*: measurement of the quality of requirements to identify bad quality requirements; and giving of tips for improving the quality of requirements, being the goal of these tips reducing ambiguity and improving completeness and adherence to templates.
- *the prediction of requirements properties*: the focus here is to predict key properties of requirements, such as priority.
- *the identification of relevant stakeholders*: detection of stakeholders that can cooperate in the definition of requirements.

Additionally, the recommendations will be context-aware, meaning that the current context of the stakeholders will be taken into account when providing the recommendations.

6.4. Group Decision Making and Recommender Systems

Requirements engineering is considered as one of the most critical phases in a software development process and poorly implemented requirements and related decisions both constitute a major source of project failure [Hofmann and Lehner 2001]. Requirements can be considered as a verbalization of different decision alternatives related to the functionality of a software system. Basic types of decisions to be taken within the scope of a software project are the following [Regnell et al. 2001]:

- *Quality decisions*, e.g., is the requirement understandable?
- *Preference decisions*, e.g., which requirements should be assigned to which release? [Ruhe 2005]

- *Classification decisions*, e.g., which stakeholders should be responsible for the quality assurance of a specific requirement?
- *Property decisions*, e.g., what is a realistic estimate of the implementation effort related to a specific requirement?

Decision making tasks in software projects become increasingly difficult due to the increasing size and complexity of the underlying processes and artefacts. As a consequence, recommendation and decision technologies are needed that help to ensure high-quality decisions which are a major precondition for successful businesses [Felfernig et al. 2010]. Since many requirements-related decisions are taken in groups, this aspect has also to be taken into account on the level of decision support tools. Existing group decision support in requirements engineering focuses on different types of discussion forums and decision support based on static preferences of individual group members [Ninaus et al. 2014]. Basic preference aggregation mechanisms help to somehow summarize the current preference levels of group members but in now way take into account the current status of a group decision process in a holistic fashion. For example some group members could be extremely disappointed with the current prioritization since they know that crucial business processes are not taken into account and - as a direct consequence - potential profits are lost. It could also be the case that risky decisions are taken which significantly increase the probability of project failure.

Summarizing, the major challenges in the context of group decision making are to take into account in an adequate fashion the preferences of individual group members (stakeholders), to be able to determine preferences of stakeholders in an intelligent fashion (e.g., on the basis of analyzing discussion forums and expertise level of individual group members), and also to be able to pro-actively foster stakeholder behavior that increases the probability of resulting high-quality software requirements. In the context of such scenarios, the overall *research objectives* of OpenReq are the following:

- *eliciting preferences of stakeholders in an intelligent fashion*: stakeholder preferences do not remain stable but often change within the scope of requirements engineering process. Our goal in this context is use community preference information from different sources such as discussion forums and social media to better figure the “real” preferences of user communities and - as a consequence - to be able to do a better prioritization and release planning.
- *identifying and counteracting decision biases*: decision biases are a major reason for suboptimal decisions. These exist in single user as well as in group decision scenarios [Felfernig et al. 2018]. Our goal in this context is to develop countermeasures of detected biases in requirements engineering scenarios.
- *explaining recommendations for increasing transparency and trust*: in any case, recommendations have to be explained to assure trust. Especially in the context of recommendations to groups, new concepts have to be developed that take into account group-specific properties such as personal relationships and diverging interests.
- *pro-actively supporting the achievement of consensus and finding a way out from “no-solution-could-be-found” dilemmas*: Analysis and synthesis approaches have to be

adapted for group settings. In OpenReq, this will especially take place in the context of group-based release planning.

- *efficient reasoning (synthesis) and diagnosis (analysis) methods for interactive settings*: diagnosis approaches will be developed that can be used in interactive settings, i.e., provide the needed response times below one second.
- *recommendation of actions to foster intended stakeholder behavior*: the goal is to develop recommendation technologies that trigger behavior change, i.e., do not only focus on the recommendation of items (e.g., requirements) to groups.

6.5. Dependency Management

Requirements and their properties, including dependencies, or more precisely interdependencies, are captured and documented or modelled during requirements specification or analysis. As a means, *natural language* is in practice a common means for representing requirements knowledge [Hiisilä et al. 2015] [Hotomski et al. 2016] [Méndez and Wagner 2015] [Palomares et al. 2017] [Raatikainen et al. 2011] [Sikora et al. 2011]. Typically, a requirement is a expressed by free form sentences. An archetypal form is a “shall”-sentence. An example of a more specific form are structured NL — for example EARS [Mavin et al. 2016] proposes five structured NL requirements templates. Beyond natural language, different types of requirements modeling notations have been proposed but not that popular. These often include a graphical notation. Goal modeling as exemplified by i* [Dalpiaz et al. 2016], is one group of modeling methods that have been proposed for requirements modeling but gained little success in practice [Mavin et al. 2017]. Likewise, feature modeling [Kang et al. 1990] proposes another family of methods that can be used for representing requirements. UML and SysML are examples of more general modeling languages that are capable of expressing requirements.

Requirements can typically contain a set of *properties* such as priority and effort. Properties can also be called attributes. We use the term *meta-data* for covering additional characteristics, such as change history. Each property, or more precisely a property type, has a value in a requirement. Requirements in different projects generally have different sets of property types. For instance, an analysis found about 280 different property types and concluded that there is no single property type that can be generally relied on to be applicable in every situation [Riegel and Doerr 2015].

Interdependencies of requirements are considered a special class of requirements traceability focusing only on information about requirements [Dahlsted and Persson 2005, Zhang et al. 2014]. In general, requirements traceability is defined as: “ability to describe and follow the life of a requirement, in both forward and backward direction, ideally through the whole system life cycle” [Gotel and Finkelstein1994]. Thus, interdependency is so-called *horizontal* traceability between the information of same type [Dahlstedt and Persson 2005]. Interdependencies are provided with a few typologies [Pohl 1996] [Carlshamre et al. 2001] [Dahlstedt and Persson 2005] [Zhang et al. 2014]. These taxonomies share many similarities with each other, but vary especially in terms of number of interdependency types they present. In general, interdependencies are considered binary relationships between two requirements. Interdependencies can be normative such as *requires* but also value-based such as *increases* *value of*. In addition, there are a few industrial studies that emphasize that interdependencies are important in requirements engineering [Lehtola et al. 2004, Vogelsang and Fuhrmann 2010] in general. However, a more detailed typology of the interdependencies is not provided in these studies. Respectively, the existing studies on interdependencies in requirements engineering seem to utilize the above mentioned taxonomies or subsets of types defined in these taxonomies. It is common that the exact types or semantics of the interdependency are not covered but referred as "(inter)dependency" without any explicit semantics.

Requirements are typically stored in a specific requirements management system. There is a large number of systems as shown in the list of 91 non-discontinued RMS [Birk and Heller 2017]. We used three reports published by consultant companies in 2016-2017 [Murphy et al. 2016, LeClair et al. 2016, Beatty et al. 2016] and one blog post on the subject [Birk et al. 2017] to identify the RMSs with significant market share and market presence. These tools are Caliber, IBM® Rational® DOORS® Next Generation, Helix RM, inteGREAT (now named Modern Requirements4TFS), Jama, Jira, Polarion® REQUIREMENTS™, and TopTeam.

Additional important activities in requirements and product management that are primarily involved in, or depended on requirements interdependency knowledge management are requirements prioritization and release management. Release planning is "concerned with selection and assignment of requirements in sequences of releases such that important technical and resource constraints are fulfilled" [Svahnberg et al. 2010]. This is typically done by a person or team that negotiates priorities of requirements and decides of what each release should comprise of. In a systematic review of strategic release planning models [Svahnberg et al. 2010], 24 different models are distinguished. Requirements interdependencies in general are clearly the most common requirement selection constraint that is taken into account in release planning, yet further details about nature of about interdependencies or interdependencies management are not given. The empirical validation for the majority of models is also noted to be immature. In Open Source development communities, setting development priorities and defining releases can be a collaborative, communicative and consensus-based process [Raymod 1999] [Hippel and Krogh2003] for which the RMS needs to provide functionalities, such as commenting and voting for the requirements' priorities. There are several strategies for defining releases, which can vary from feature-driven, time-based, frequent and further [GomesdaSilva et al. 2017].

In this context, the research objectives of OpenReq include:

- By using existing and (automatically) extracting new interdependencies, construct a knowledge representation of the entire set of requirements.
- Automatically construct the knowledge representation from requirements stored in a modern RMS.
- Enable analysis for the requirements beyond single requirements or its properties such as in the cases of conflicting or required requirements.
- Empirically assess the practical needs for requirements knowledge management at a high, such as product management, level.
- Empirically assess and refine practical application of requirements knowledge management in the context of modern RMS.

- Provide semantic reasoning concepts for a personalized diagnosis and repair of inconsistent requirements models.

6.6. Patterns and Requirements Reuse

Requirements reuse consists on taking advantage of requirements knowledge obtained from previous IT projects and later on using this knowledge in a new one. Requirement patterns are assets that provide a structured representation to reuse knowledge about specific requirements. Other assets associated to the term requirement pattern are not considered in this summary since they are not the target of the OpenReq project. Firstly we include here a summary of the existing approaches to requirements reuse, and after that a summary of the practical experiences presented in the literature of the application of requirement reuse and patterns. For a detailed description of the state of the art see deliverable D5.1.

The existing proposals of requirements reuse can be mainly characterized by: the part of a requirements specification that they allow to reuse, the language used to express the requirements to be reused, the size of the part of the specification that can be reused, the abstraction of the knowledge to reuse, and the scope where the reuse can be done.

- The proposals classified according the part of a specification that they allow to reuse are proposals that reuse requirements knowledge (e.g., [Withall 2007], [Franch et al. 2013]), requirements classification knowledge (e.g., [Panis 2015], [Franch et al. 2013]) and/or requirements interdependencies (e.g., [Franch et al. 2013], [Konrad et al. 2002]).
- From the proposals that allow to reuse requirements knowledge, they assume that these requirements are expressed in natural language (e.g., [Withall 2007]), use cases (e.g., [Chung et al. 2006], [Dehlinger et al. 2005]), and domain models (e.g., [Toval et al. 2002]). More formal representation of requirements are also used in some approaches, such as i* models (e.g., [Bourgada et al. 2004]).
- The size of reusable artifacts varies from individual requirements (e.g., [Daramola et al. 2012]) to requirement clusters (e.g., [Konrad et al. 2002]), to parts of requirement specifications and even to complete requirement specifications (e.g., [Chung et al. 2006]).
- The abstraction of knowledge to reuse in the proposals varies from no abstraction — such as templates that are natural language sentences with no required structure (e.g., [Hauksdottir et al. 2012]) — to a high level of abstraction — for instance in the case of patterns or feature models based on the notion of variability of requirements (e.g., [Withall 2007], [Mannion et al. 1999], [Franch et al. 2013]).
- There are proposals of reuse that are specific for a particular domain without any aim at generalization, such as for web applications (e.g., [Wahono et al. 2002]), embedded systems (e.g., [Konrad et al. 2002]), or security requirements (e.g., [Daramola et al. 2012], [Jensen et al. 2009]), and other that are generals for any domain.

The existent proposals that are reporting an industrial validation can be classified depending on the type of validation done (i.e., the type of empirical study carried out):

- Case studies that test the usefulness of the proposal, making special emphasis on the percentage of requirements that has been reused (e.g., [Issa et al. 2010], [Mahmoud et al. 2010], [Myklebust et al. 2014], [Renault et al. 2009]).
- Interviews of experts that explore the usefulness, advantages and disadvantages of the approach (e.g., [Issa et al. 2010], [Lam 1997]).
- Industrial experiences that explain how requirements reuse is being applied in real industrial settings (e.g., [Daramola et al. 2012], [Hauksdottir et al. 2012], [Hauksdottir et al. 2016], [Heumesser et al. 2003], [Panis 2015], [Zuccato et al. 2011]).

In this context, the overall *research objective* of OpenReq is achieving requirements reuse by using software requirement patterns (SRPs). Therefore, it is necessary to define the structure of SRP that adapts to the needs of the project and to construct a catalogue of SRPs ready to reuse. In addition, it is necessary to define different ways to use this catalogue: some might envisage more traditional approaches (e.g., directly browsing or searching the SRP catalogue), while others might be more automatic (e.g., proposing SRPs that are dependent or related to the a specific requirement already entered in the Openreq system).

7. Conclusion

In this deliverable we provided an overview of the OpenReq Knowledge Base including a summary report related to its deployment and population. The goal of the OpenReq Knowledge Base is to establish a central information platform for topics related to recommendation and decision technologies in requirements engineering. To achieve this goal it is essential to maintain the OpenReq Knowledge Base overall the whole project and beyond.

8. References

1. G. Antoniol, K. Ayari, M. Di Penta, F. Khomh, and Y. G. Guéhéneuc. Is it a bug or an enhancement?: a text-based approach to classify change requests. Proceedings of the IBM Centre for Advanced Studies Conference on Collaborative Research, 2008. J. Anvik, L. Hiew, and G.C. Murphy. Who should fix this bug? ACM/IEEE International Conference on Software Engineering, pp. 361–370, 2006.
2. T. Arnuphaptrairong. Top ten lists of software project risks: Evidence from the literature survey, International Multi Conference of Engineers and Computer Scientists, 2011.
3. B. Boehm, and V. Basili. Software Defect Reduction Top 10 List, Journal Computer, vol. 34, no. 1, p. 135–137, 2001.
4. Calleam Consulting. Why do projects fail. 101 Common Causes, 2015. [Online]. Available: http://calleam.com/WTPF/?page_id=2338. [Accessed December 2017].
5. C. Castro-Herrera C., J. Cleland-Huang J., and B. Mobasher. Enhancing Stakeholder Profiles to Improve Recommendations in Online Requirements Elicitation. 17th IEEE International requirements engineering Conference (RE'09), pp. 37–46, 2009.
6. C. Castro-Herrera, and J. Cleland-Huang. Utilizing Recommender Systems to Support Software Requirements Elicitation. 2nd International Workshop on Recommendation Systems for Software Engineering (RSSE'10), ACM, 2010.
7. J. Cleland-Huang, H. Dumitru, C. Duan, and C. Castro-Herrara. Automated support for managing feature requests in open forums. Communications of the ACM - A View of Parallel Computing, 52(10), pp. 68–74, 2009.
8. D. Cubranic, and G.C. Murphy. Automatic bug triage using text categorization. 16th International Conference on Software Engineering & Knowledge Engineering, pp. 92–97, 2004.
9. G.A. Di Lucca, M. Di Penta, and S. Gradara. An Approach to Classify Software Maintenance Requests. IEEE International Conference on Software Maintenance, pp. 93-102, 2002.
10. H. Dumitru, M. Gibiec, N. Hariri, J. Cleland-Huang, B. Mobasher, and C. Castro-Herrara. On-demand feature recommendations derived from mining public product descriptions. 33rd International Conference on Software Engineering, pp. 181-19, 2011.
11. A. Felfernig, W. Maalej, M. Mandl, F. Ricci, and M. Schubert, Recommendation and Decision Technologies For requirements engineering, ICSE 2010 Workshop on Recommender Systems in Software Engineering, Cape Town, South Africa, pp. 1-5, 2010.
12. A. Felfernig, M. Schubert, and S. Reiterer. Personalized diagnosis for over-constrained problems. International Joint Conference on Artificial Intelligence, pp. 1990–1996, 2013.
13. A. Felfernig, L. Boratto, M. Stettinger, and M. Tkalcic. Group Recommender Systems, to appear in March 2018, Springer, 2018.
14. C. Fitzgerald, E. Letier, and A. Finkelstein. Early failure prediction in feature request management systems. IEEE 19th International requirements engineering Conference

(RE'11), pp. 229-238, 2011.
J. Gaeul, K. Sunghun, and T. Zimmermann. Improving bug triage with bug tossing graphs. European Software Engineering Conference, pp. 111–120, 2009.

15. H. Hofmann and F. Lehner. Requirements engineering as a success factor in software projects. IEEE Software, 18(4):58–66, 2001.
16. A. Lamkanfi, S. Demeyer, E. Giger, and B. Goethals. Predicting the severity of a reported bug. Mining Software Repositories, pp. 1–10, 2010.
17. S. Lim, and A. Finkelstein. Stakerare: Using social networks and collaborative filtering for large-scale requirements elicitation. IEEE Trans. on Softw. Eng., 38(3), pp. 707-735, 2012.
18. M. Kumar, N. Ajmeri, and S. Ghaisas. Towards Knowledge Assisted Agile Requirements Evolution. 2nd International Workshop on Recommendation Systems for Software Engineering, pp. 16–20, 2010.
19. T. Menzies, and A. Marcus. Automated severity assessment of software defect reports. IEEE International Conference on Software Maintenance, pp. 346–355, 2008.
20. N.K. Nagwani, and S. Verma. Predicting expert developers for newly reported bugs using frequent terms similarities of bug attributes. International Conference on ICT and Knowledge Engineering, pp. 113–117, 2012.
21. G. Ninaus, A. Felfernig, M. Stettinger, S. Reiterer, G. Leitner, L. Weninger, and W. Schanil. IntelliReq: Intelligent Techniques for Software requirements engineering, European Conference on Artificial Intelligence, Prestigious Applications of Intelligent Systems (PAIS), pp. 1161-1166, 2014.
22. M.W. Olga Baysal, and G. R. Cohen. A Bug You Like: A Framework for Automated Assignment of Bugs. 17th International Conference on Program Comprehension (ICPC '09), pp. 297–298, 2009.
23. B. Regnell, B. Paech, C. Aurum, C. Wohlin, A. Dutoit, and J. N. och Dag. Requirements means decision!, 2001
24. J. Romero-Mariona, H. Ziv, and D.J. Richardson . SRRS: A Recommendation System for Security Requirements. International Workshop on Recommendation Systems for Software Engineering, 2004.
25. C. Weiß, R. Premraj, T. Zimmermann, and A. Zeller. How Long Will It Take to Fix This Bug?. 4th Working Conference on Mining Software Repositories (MSR'07), 2007.
26. Negash, Solomon (2004) "Business Intelligence," Communications of the Association for Information Systems: Vol. 13 , Article 15. Available at: http://aisel.aisnet.org/cais/vol13/iss1/15
27. L. Mich, M. Franch, and P. Novi Inverardi. 2004. Market Research for Requirements Analysis Using Linguistic Tools. 9 (01 2004), 40–56.
28. M. Bates. 1995. Models of natural language understanding. Proceedings of the National Academy of Sciences of the United States of America, 92(22), 9977–9982.
29. N. Carlson and P. Laplante. 2014. The NASA Automated Requirements Measurement Tool: A Reconstruction. Innov. Syst. Softw. Eng. 10, 2 (June 2014), 77–91. https://doi.org/10.1007/s11334-013-0225-8
30. M. Harman, Y. Jia, and Y. Zhang. 2012. App Store Mining and Analysis: MSR for App Stores. In Proceedings of the 9th IEEE Working Conference on Mining Software

Repositories (MSR ’12). IEEE Press, Piscataway, NJ, USA, 108–111. http://dl.acm.org/citation.cfm?id=2664446.2664461

31. T. Johann, C. Stanik, A. M. Alizadeh, and W. Maalej. 2017. SAFE: A Simple Approach for Feature Extraction from App Descriptions and App Reviews. In 2017 IEEE 25th International requirements engineering Conference (RE) (2017-08-01). pp. 21-30.
32. D. Pagano and W. Maalej. 2013. User feedback in the appstore: An empirical study. In 2013 21st IEEE International requirements engineering Conference (RE). 125–134. https://doi.org/10.1109/RE.2013.6636712
33. E. Guzman and W. Maalej. 2014. How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Reviews. In 2014 IEEE 22nd International requirements engineering Conference (RE). 153–162. https://doi.org/10.1109/RE.2014.6912257
34. G. Williams and A. Mahmoud, "Mining Twitter Feeds for Software User Requirements," *2017 IEEE 25th International requirements engineering Conference (RE)*, Lisbon, 2017, pp. 1-10. doi: 10.1109/RE.2017.14
35. E. Guzman, R. Alkadhi, N. Seyff, “An exploratory study of Twitter messages about software applications”. In 2017 requirements engineering, 387-412. https://doi.org/10.1007/s00766-017-0274-x
36. Z. Kurtanović and W. Maalej. 2017. Automatically Classifying Functional and Non-Functional Requirements Using Supervised Machine Learning. In 2017 IEEE 25th International requirements engineering Conference (RE) (2017-08-01). to appear.
37. I. Sommerville and P. Sawyer. 1997. requirements engineering: A Good Practice Guide (1st ed.). John Wiley & Sons, Inc.
38. J. Eckhardt, A. Vogelsang, and D. Méndez Fernández. 2016. Are non-functional requirements really non-functional?: an investigation of non-functional requirements in practice. In Proc. of the 38th International Conference on Software Engineering. ACM.
39. W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik. 2016. On the Automatic Classification of App Reviews. Requir. Eng. 21, 3 (Sept. 2016), 311–331. https://doi.org/10.1007/s00766-016-0251-9
40. X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun. 2008. An approach to detecting duplicate bug reports using natural language and execution information. In 2008 ACM/IEEE 30th International Conference on Software Engineering. 461–470. https://doi.org/10.1145/1368088.1368151
41. N. Jalbert and W. Weimer. 2008. Automated duplicate detection for bug tracking systems. In 2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN). 52–61. https://doi.org/10.1109/DSN.2008.4630070
42. N. Bettenburg, R. Premraj, T. Zimmermann, and 3. Sunghun Kim. 2008. Duplicate bug reports considered harmful… really?. In 2008 IEEE International Conference on Software Maintenance. 337–345. https://doi.org/10.1109/ICSM.2008.4658082

43. J.R. Cooper Jr., S. Lee, R.A. Gandhi and O. Gotel (2009). requirements engineering Visualization: A Survey on the State-of-the-Art. 2009 Fourth International Workshop on requirements engineering Visualization (rev'09).
44. Z. S. H. Abad, M. Noaeen, and G. Ruhe. 2016. requirements engineering Visualization: A Systematic Literature Review. In 2016 IEEE 24th International requirements engineering Conference (RE). 6–15. https://doi.org/10.1109/RE.2016.61
45. S. Withall. 2007. Software requirement patterns. *Microsoft Press*.
46. X. Franch, C. Quer, S. Renault, C. Guerlain, C. Palomares. 2013. Constructing and using software requirement patterns *Managing requirements knowledge*, Springer.
47. S. Konrad, B.H. Cheng. 2002. Requirements patterns for embedded systems *IEEE Joint International Conference on requirements engineering (RE)*, 127–136.
48. L. Chung, S. Supakkul. 2006. Capturing and reusing functional and non-functional requirements knowledge: a goal-object pattern approach *IEEE international conference on information reuse and integration (IRI)*, 539–544.
49. M.C. Panis. 2015. Reuse of Architecturally Derived Standards Requirements *IEEE International requirements engineering Conference (RE)*, 296–304.
50. J. Dehlinger, T.R. Lutz. 2005. A product-line requirements approach to safe reuse in multi-agent systems *4th international workshop on Software engineering for large-scale multi-agent systems (SELMAS)*, 1-7.
51. A. Toval, J. Nicolás, B. Moros, F. García. 2002. Requirements reuse for improving information systems security: a practitioner’s approach, *requirements engineering*, 6(4).
52. S. Bouraga, I. Jureta, S. Faulkner. 2014. Requirements engineering patterns for the modeling of Online Social Networks features *IEEE 4th International Workshop on Requirements Patterns (RePa)*, 33-38.
53. O. Daramola, G. Sindre, T. Stalhane. 2012. Pattern-based Security requirements specification using ontologies and boilerplates *IEEE second international workshop on requirements patterns (RePa)*, 54–59.
54. D. Hauksdottir, A. Vermehren, J. Savolainen. 2012. Requirements reuse at Danfoss 20th *IEEE international conference on requirements engineering (RE)*, 309–314.
55. M. Mannion, B. Keepence, H. Kaindl, J. Wheadon. 1999. Reusing single system requirements from application family requirements *International Conference on Software Engineering*, 453-462.
56. R.S. Wahono, J. Cheng. 2002. Extensible requirements patterns of web application for efficient web application development. *IEEE First International Symposium on*, Cyber Worlds 412-418.
57. J. Jensen, I.A. Tondel, M.G. Jaatun, P.H. Meland, H. Andresen. 2009. Reusable security requirements for healthcare applications *International conference on availability, reliability and security (ARES)*, 380–385.
58. A.A. Issa, A. Al-Ali. 2010. Use Case Patterns Driven requirements engineering *Second International Conference on Computer Research and Development*, 307-313.
59. W. Lam. 1997. Achieving requirements reuse: A domain-specific approach from avionics *Journal of Systems and Software*, 38(3), 197-209.

60. A. Mahmoud, N. Niu. 2010. An experimental investigation of reusable requirements retrieval *IEEE International Conference on Information Reuse & Integration*, 330-335.
61. T. Myklebust, N. Lyngby, R. Bains, G.K. Hanssen. 2014. CoVeR maintenance and maintainability requirements by using tools *Reliability and Maintainability Symposium*, 1-6.
62. D. Hauksdottir, A. Vermehren, J. Savolainen. 2012. Requirements reuse at Danfoss 20th *IEEE international conference on requirements engineering (RE)*, 309–314.
63. D. Hauksdóttir, B. Ritsing, J.C. Andersen, N.H. Mortensen. 2016. Establishing Reusable Requirements Derived from Laws and Regulations for Medical Device Development *IEEE 24th International requirements engineering Conference Workshops (REW)*, 220-228.
64. N. Heumesser, F. Houdek. 2003. Towards systematic recycling of systems requirements *25th International Conference on Software Engineering*, 512-519.
65. A. Zuccato, N. Daniels, C. Jampathom. 2011. Service Security Requirement Profiles for Telecom: How Software Engineers May Tackle Security *Sixth International Conference on Availability, Reliability and Security*, 521-526.
66. Beatty, J., Stowe, M.J., Cardenas, A., Reinhart, D., & Bartlett, J. Requirements Management Tools Evaluation Report. Seilevel Report. Seilevel report. 2016
67. A. Birk, G. Heller, List of Requirements Management Tools. The Making of Software, http://makingofsoftware.com/resources/list-of-rm-tools. Accessed November 2017. 2017
68. A. Cesar Brandão Gomes da Silva, G. de Figueiredo Carneiro, F. Brito e Abreu, and M. Pessoa Monteiro, Frequent Releases in Open Source Software: A Systematic Review Information, Multidisciplinary Digital Publishing Institute, 8, 109, 2017
69. P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, B. and J. Natt och Dag, An industrial survey of requirements interdependencies in software product release planning Proceedings Fifth IEEE International Symposium on requirements engineering, 84-91 2001
70. Å. Dahlstet, and A. Persson, Requirements Interdependencies: State of the Art and Future Challenges Engineering and Managing Software Requirements, Springer Berlin Heidelberg, 95-116, 2005
71. F. Dalpiaz,X. Franch and J. Horkoff, iStar 2.0 Language Guide. arXiv:1605.07767 https://arxiv.org/pdf/1605.07767v3.pdf, 2016
72. O. Gotel and C. Finkelstein An analysis of the requirements traceability problem Proceedings of IEEE International Conference on requirements engineering, 94-101, 1994
73. H. Hiisilä, M. Kauppinen and S. Kujala, Challenges of the Customer Organization's requirements engineering Process in the Outsourced Environment – A Case Study requirements engineering: Foundation for Software Quality, Lecture Notes in Computer Science, vol 9013, 2015
74. E. Hippel and G. Krogh Open source software and the "private-collective" innovation model: Issues for organization science. Organization science, 14(2), 209-23 2003

75. S. Hotomski, E. Charrad and M. Glinz, An Exploratory Study on Handling Requirements and Acceptance Test Documentation in Industry IEEE 24th International requirements engineering Conference (RE), 116-125, 2016
76. K. Kang, S. Cohen, J. Hess, W. Novak and Peterson, Feature-Oriented Domain Analysis (FODA) Feasibility Study Software Engineering Institute, Software Engineering Institute, 1990
77. A. LeClair, K. Bittner, C Mines, and T. Turrisi, TechRadar™: Modern Software Requirements Management Tools, Q2. Forrester report. 2016
78. L. Lehtola, M. Kauppinen and S. Kujala Requirements Prioritization Challenges in Practice Product Focused Software Process Improvement: 5th International Conference, 497-508, 2004
79. A. Mavin, P. Wilksinson, S. Gregory, S. and E. Uusitalo, Listens Learned (8 Lessons Learned Applying EARS) 2016 IEEE 24th International requirements engineering Conference, 276-282, 2016
80. A. Mavin, P. Wilkinson, S. Teufl, H. Femmer, J. Eckhardt, and J. Mund, Does Goal-Oriented requirements engineering Achieve Its Goal? 2017 IEEE 25th International requirements engineering Conference, 174-183, 2017
81. D. Méndez and S. Wagner. Naming the pain in requirements engineering: A design for a global family of surveys and first results from Germany Information and Software Technology, Volume 57, 616-643, 2015
82. T. Murphy, M. Revang and L. Wurster, Market Guide for Software Requirements Definition and Management Solutions. Gartner report. ID: G00276075. 2016
83. C. Palomares, C. Quer, C. and X. Franch, Requirements reuse and requirement patterns: a state of the practice survey Empir Software Eng, 22(6), 2719-2762, 2017
84. K. Pohl, Process-centered requirements engineering John Wiley & Sons, Inc., 1996
85. M. Raatikainen, T. Männistö, T. Tommila and J. Valkonen, J."Challenges of requirements engineering — A case study in nuclear energy domain IEEE 19th International requirements engineering Conference, 253-258, 2011
86. E. Raymond, "The cathedral and the bazaar." Philosophy & Technology 12(3), 1999
87. N. Riegel and J Doerr, A systematic literature review of requirements prioritization criteria International Working Conference on requirements engineering: Foundation for Software Quality, 300-317 , 2015
88. E. Sikora, B. Tenbergen and K. Pohl requirements engineering for Embedded Systems: An Investigation of Industry Needs requirements engineering: Foundation for Software Quality (REFSQ), Lecture Notes in Computer Science, vol 6606, 2011
89. M. Svahnberg, J. van Gurp and J. Bosch, A taxonomy of variability realization techniques Software --- Practice and Experience, 35, 705-754, 2005
90. A. Vogelsang and S. Fuhrmann, Why feature dependencies challenge the requirements engineering of automotive systems: An empirical study 2013 21st IEEE International requirements engineering Conference (RE), 267-272, 2013
91. H. Zhang, J Li, L. Zhu, R. Jeffery, Y. Liu, Q. Wang, and M Li, Investigating dependencies in software requirements for change propagation analysis Information and Software Technology, 56, 40-53, 2014

92. K. Petersen, S. Vakkalanka, and L. Kuzniarz, “Guidelines for conducting systematic mapping studies in software engineering: An update,” *Information and Software Technology*, vol. 64, no. C, pp. 1–18, Aug. 2015.
93. B. Kitchenham, S. Charters, Guidelines for Performing Systematic Literature Reviews in Software Engineering, Tech. rep., Technical report, EBSE Technical Report EBSE-2007-01, 2007
94. *Software Release Planning*. Günther *Ruhe*. University of Calgary. 2500 University Drive NW. Calgary, AB T2N 1N4, Canada.