ECAI 2020
G.D. Giacomo et al. (Eds.)

doi:10.3233/FAIA200477

Intelligent Recommendation & Decision Technologies for Community-Driven Requirements Engineering

Ralph Samer and **Martin Stettinger** and **Alexander Felfernig**[1] and **Xavier Franch**[2] and **Andreas Falkner**[3]

Abstract. Requirements Engineering (RE) represents a critical phase in the management and planning of software projects. One of the main reasons for project failure is missing or incomplete RE. In order to reduce the risk of project failure, there exists a high and urgent demand for applying intelligent technologies in RE. Since the RE process is mainly decision- and community-driven, *Recommender Systems* are supposed to be applied in this particular context to support stakeholders in decision-making and, hence, to increase the quality of the decisions taken by the stakeholders. This paper introduces a variety of innovative recommendation tools developed within the scope of the European Horizon 2020 research project OPENREQ. Moreover, we give an overview of user studies conducted to evaluate our approaches and present final results of selected studies. The study results indicate that the developed concepts have the potential to significantly improve the quality of requirements definition and requirements prioritization.

1 Introduction

Requirements Engineering (RE) plays an important role in software development. In general, RE represents a branch of systems engineering which deals with the definition and fulfilment/implementation of desired properties and constraints of software-intensive systems. The major phases of the RE process are the *elicitation and definition of requirements*, the *negotiation of requirements*, *quality assurance*, and *release planning*. RE can be considered as a critical phase in a software project since poor (or missing) RE can (1) cause a software project to miss important deadlines (due to a late discovery of serious issues in the RE model), (2) lead to increased costs which can exceed the project budget, or (3) even result in project failure [18, 22]. In fact, research shows that the follow-up costs can add up to 40% of the overall project costs [18]. In the worst case, the project might miss important deadlines or even fail. Consequently, RE constitutes a high risk factor for the success of a project. Hence, there is a high demand for applying intelligent technologies to support stakeholders in RE, in order to mitigate these project risks. In particular, recommender systems (RS) are predestined to support stakeholders in different decision-making scenarios which represent the core foundation of the RE process [12, 22]. RS can support stakeholders in RE tasks such as, *requirements definition, release decisions, stakeholder identification*, and *dependency detection* [22, 23]. Beyond the use of RS in RE, further intelligent and automated solutions based on artificial intelligence can be used to support stakeholders in RE.

This paper presents innovative applications of intelligent recommendation and decision technologies in RE which are based on artificial intelligence and were developed within the scope of the European research project OPENREQ[4]. The major aim of OPENREQ is to address the aforementioned issues by providing an innovative and intelligent tool support, which might change the way RE stakeholders think about and work with requirements. Following the objective to foster high quality decision-making, OPENREQ offers intelligent solutions for all phases of the RE process. OPENREQ has even the potential to update current software engineering methodologies and introduce new roles in software organizations. For instance, with OPENREQ the boundaries between marketing, RE, and maintenance should be reconsidered. The outcomes of the OPENREQ project touch a broad set of different communities. OPENREQ provides actionable feedback for novel contributions, software practitioners for the scientific community as well as solid foundations for the open-source community. The work presented in this paper deals with the analysis and extension of current methodologies on (1) stakeholder and user involvement in a software life-cycle, and (2) distributed requirements engineering and management. We provide a framework for processes and methodologies that support stakeholders in using the OPENREQ platform to achieve high efficiency and quality in requirements elicitation and management. The focus lies on extending agile, reuse-driven methodologies, community-centred participative product development, and open innovation methodologies. The core contributions of the developed recommendation tool suite presented in this paper, provide genuine added value for traditional software development institutions and open source communities in terms of (1) more efficient information exchange among stakeholders, (2) increased RE quality by providing quality-related feedback and advanced conflict-resolution, and (3) reduced risk of project failure by providing immediate feedback to requirements engineers early in the process.

The remainder of this paper is structured as follows. Section 2 provides an overview of the developed OPENREQ tool suite covering a broad set of different intelligent recommendation technologies. In Section 3, we show the user interface of the RE platform OPENREQ!LIVE which provides a central platform for the use of the recommendation tools. Section 4 presents the design and results of several user studies that evaluate the different recommendation approaches. Section 4 outlines related work and provides some ideas regarding future work for recommender systems in the field of RE. Finally, the paper is concluded with Section 6.

[1] Institute of Software Technology, Graz University of Technology, Austria, email: {rsamer,martin.stettinger,felfernig}@ist.tugraz.at
[2] Universitat Politècnica de Catalunya, Spain, email: franch@essi.upc.edu
[3] Siemens AG, Vienna, Austria, email: andreas.a.falkner@siemens.com

[4] https://www.openreq.eu

2 OpenReq Recommendation Technologies

A broad collection of different recommendation tools has been developed within the scope of the research project OPENREQ. Thereby, we focused on the techniques which are of the highest relevance for the RE process. These recommendation tools are based on common recommendation concepts such as *content-based filtering* [25], *collaborative filtering* [7, 15], *knowledge-based recommendation* [11], and *group recommendation approaches* [21, 9]. The objective of the developed recommendation approaches is to improve the efficiency and the quality in requirements elicitation and management. Our approaches are supposed to support stakeholders working on different RE-related tasks such as the *assignment of stakeholders to requirements*, the *elicitation of requirements*, the *identification of requirement dependencies*, and the *prioritization of requirements*. The developed techniques and tools follow the community-driven OPENREQ approach for modern software RE. Figure 1 presents the general flow of the OPENREQ approach and demonstrates on a basic level how the participants (users, communities, and stakeholders) interact with the RE process. As shown in the figure, the basic idea of the OPENREQ approach is that users and communities provide valuable feedback (implicit and explicit) which can be analyzed and used as learning input for OPENREQ's intelligent technologies. All stakeholders (requirements engineers, developers, and users) provide expertise and define preferences which are considered as input for the RE process. Beyond this, OPENREQ aims to use knowledge from previous/past projects and the history of current projects as input to further optimize decision support provided by the developed recommendation and decision support tools. An overview of these tools is given in the remainder of this section. All tools were developed as standalone open-source web services[5].

2.1 Requirement Elicitation

The elicitation of requirements represents a task in the initial phase of the RE process, where requirements of a software project are jointly defined and collected by the project stakeholders. The traditional way is that the stakeholders provide the fundamental elements for the definition of a requirement which can be textual descriptions, scenario descriptions, use cases, or mock-up illustrations of prototypical user interfaces. Based on these elements, stakeholders select the relevant requirements. However, the process of distinguishing between the elements which define a new requirement and those elements that further explain or describe an already defined requirement, is often very time-consuming for stakeholders. To support this task, OPENREQ provides a classification service called *OpenReq Classification Service* [8] which is based on supervised machine learning. It focuses on the classification of textual descriptions and allows to determine whether a piece of text (paragraph) as a part of a formatted text document (Microsoft Word) defines a new requirement (denoted as REQ) or represents a description (denoted as PROSE) which is related to a previously defined requirement. Paragraphs classified as PROSE are automatically linked to the corresponding requirement. The hierarchy of classified requirements and PROSE paragraphs is then converted into a format suitable for the requirement management tool *IBM doors*[6]. Misclassified samples have to be manually corrected by requirements managers.

Another approach supporting the elicitation of requirements has been developed by Stanik et al. [33, 34]. The *Orchestration service* of OPENREQ applies this approach in order to intelligently extract requirements from social media channels. The basic idea of the approach is to bridge the gap between requirements engineers and customers in an agile development process that relies on continuous feedback from the customer. The service opens new feedback channels for customers to report issues and ideas regarding the developed software products. The service intelligently analyzes and classifies all messages (tweets) which appear in these channels by using supervised machine learning [34]. The classification model can either be trained by using traditional machine learning or deep learning. Messages can be of any kind of requirement such as a *feature request* (new feature which should be included in the next release), a *bug report* (a bug which leads to malfunction of the software), or an *inquiry* (a question related to different aspects such as usability, compatibility, or questions on how to use the software). All other messages which do not represent (or do not refer to) a requirement are classified as *irrelevant*. All tweets are preprocessed and cleaned in order to avoid obvious spam messages (i.e., *irrelevant*) at an early stage of the classification process. Further (less obvious) spam messages or any other kind of unclear messages are detected by the prediction model and automatically classified as *irrelevant*. The requirements engineers are responsible to review the list of recommended requirements (messages classified as requirements) and select the final candidates which are then included in the requirements model of the software product.

Most software projects contain reusable parts of the functionality (e.g., user authentication systems used in online applications) which often represent core components of other software projects as well. Consequently, the requirements related to the implementation of such reusable components can be reused in new software projects. Recommender systems aim to support stakeholders in the definition of new requirements by suggesting requirements which are related to the content of already defined requirements [23]. The presentation of reusable requirements (extracted from other software projects that are related to the current project) represents a large potential for recommender systems during requirement elicitation. OPENREQ provides a recommendation component called *similar-related requirements recommendation service* which is a web service that addresses this aim. The main goal of the service is (1) to foster the systematic reuse of requirements such that the efficiency of the RE process can be improved, and (2) to analyze the requirements of a project in order to detect duplicates and related requirements. The service recommends requirements that are similar or related to a given set of requirements by intelligently analyzing the given requirement set of the current project (*inter-project analysis*) as well as requirements of different existing software projects (*cross-project analysis*). The service compares the semantics of the words from the description of the requirements to find and recommend closely related requirements. An additional part of the *cross-project analysis* is the recommendation of requirements of reusable components from other projects. Typical software projects usually have a large assortment of requirements (more than 100 or 1,000 requirements) which makes the elicitation of requirements to become one of the most important and time-consuming tasks in RE. OPENREQ tackles the aforementioned issues and allows stakeholders to save much time and to reduce the risk of overseeing important requirements in a project. It is important to mention that the late discovery of requirements can cause serious consequences for a software project which further underlines the use of recommenders in this critical phase of the RE process.

[5] Source code is published on GitHub: https://github.com/OpenReqEU
[6] https://www.ibm.com/us-en/marketplace/requirements-management

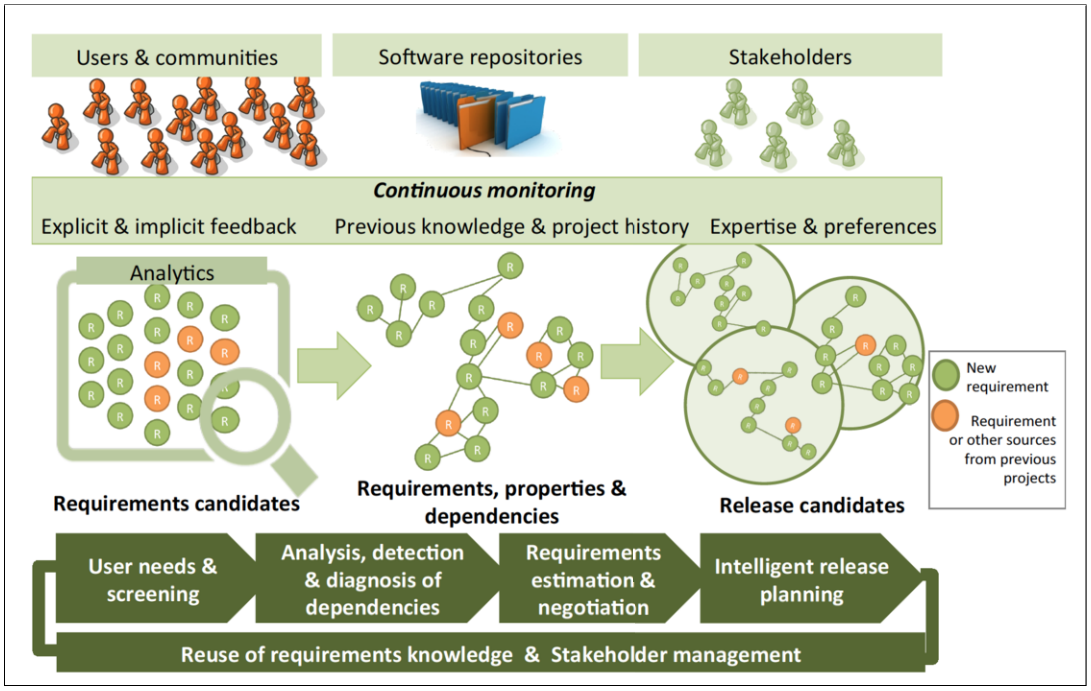

Users & communities
Software repositories
Stakeholders
Continuous monitoring
Explicit & implicit feedback
Previous knowledge & project history
Expertise & preferences
Analytics
Requirements candidates
Requirements, properties & dependencies
Release candidates
New requirement
Requirement or other sources from previous projects
User needs & screening
Analysis, detection & diagnosis of dependencies
Requirements estimation & negotiation
Intelligent release planning
Reuse of requirements knowledge & Stakeholder management

Figure 1. Overview of OPENREQ's Requirements Engineering approach.

2.2 Requirement Dependency Detection

The identification of dependencies between requirements represents another important task in RE. In this task, requirements are typically analyzed pairwise, in order to find all dependent requirement pairs. There exist different types of dependencies/relationships (e.g., *requires*, *includes*, *excludes*, or *similar* [30, 14]) expressing different meanings. The early and complete discovery of all dependencies has a significant impact on the successful completion of a software project [19]. Incomplete, inconsistent, or incorrect dependencies can induce serious consequences in a project, such as increased costs which may exceed project budget, lead to missed project deadlines, or even project failure [30, 27]. As already mentioned before, software projects often have a large number of requirements which must be fulfilled in order to successfully complete the project. The number of requirement pairs k to be analyzed for dependencies is a function of quadratic order of the number of requirements n (more formally, $k = \binom{n}{2}$). Hence, the manual analysis of k pairs quickly turns out to be a herculean task/effort for (human) stakeholders. Moreover, with an increasing number of requirements in a project, the probability is very high that a manually defined set of dependencies is incomplete and contains inconsistencies. The complete awareness of all (correct) dependencies directly affects the *release planning* of a software product since the dependencies express relevant information about the compatibility between the requirements as well as the chronological time order in which the requirements should be implemented. An extensive consideration of all dependencies fosters a more fruitful release planning which helps to avoid effortful re-designs and re-implementations later in the project. Consequently, the application of automated technologies which assist the stakeholders in finding requirement dependencies is essential for a software project.

In order to address this issue, two different recommendation approaches have been implemented within the scope of OPENREQ. The first approach is represented by OPENREQ's *Dependency Recommender* service which analyzes the requirements set of a software project to find dependent requirement pairs. The underlying approach uses *Latent Semantic Analysis* [5, 17] to transform the textual descriptions of the requirements into a low dimensional semantic-space representation (for more details, see [30]). This way, noisy words can be filtered out and more emphasis is placed on semantically-related requirements. The pairs of closely semantically-related requirements are considered as dependent requirements. The main advantage of this service is that it does not require any labelled training data and can be used once all requirements have been defined in a project. This is due to the reason that the underlying approach uses unsupervised learning (soft-clustering). Moreover, fine-tuned parameters (e.g., the minimal similarity distance between two requirements) avoid that two too closely related (i.e., similar or duplicate) requirements are falsely considered as a dependent pair.

Besides this unsupervised approach, we have developed another approach that is based on supervised machine learning and requires labelled dependency data in order to learn a prediction model (see [2]). This approach goes beyond the level of similarity-based recommendation, and uses probabilistic features which take statistical aspects from the area of *Information Theory* into account. In contrast to the previous approach, the main benefit of this approach is a significantly increased prediction quality (see results in [30]).

2.3 Prioritization and Evaluation of Requirements

A correct prioritization and allocation of all resources and requirements is the fundamental basis for a smooth and efficient schedule in every software project. This involves the evaluation and prioritization of a project's requirements, the assignment of suitable stakeholders to the requirements (see Section 2.4), and release planning (see Section 2.5). In particular, an efficient support of prioritization decisions is essential for a software project. This is due to the reason that a manual prioritization of a large number of requirements is a very time-consuming and effortful process [38]. However, the prioritization of requirements is often performed by a single person or a very small group of stakeholders (e.g., requirements/project manager).

According to research in the field of group recommender systems, more information exchange between decision-makers as well as more people involved in a decision can significantly increase the probability of better prioritizations [32, 36]. Following these scientific empirical observations, we developed new approaches for group recommendation user interfaces which (1) trigger more stakeholder engagement and (2) foster information exchange between the stakeholders (see [31]). The approaches are based on the concept of multi-attribute utility theory (MAUT) [9, 23] which represents a multi-dimensional rating scheme. MAUT for groups extends the basic utility-based recommendation for single users to multi-user scenarios where the preferences of the individual stakeholders are aggregated into a recommendation such that the whole group is satisfied with the recommendation. In our approach, we use the three interest dimensions *(profit, effort, risk)* which must be evaluated by every stakeholder for every requirement individually. The preferences of a single user are the user's ratings of all specified dimensions. Strongly diverging ratings of different group members within the same dimension indicate a strong disagreement and are considered as a conflict by the group recommender. The group recommender automatically detects all of such conflicts and presents them to the group members who are involved in the conflicts. For each conflict of a requirement, the group members are required to discuss the conflict and (after the discussion) to reevaluate the requirement's dimensions which are affected by the conflict. This helps to trigger more communication between stakeholders which positively influences the quality of the requirement prioritizations. Once all requirements have been evaluated and all conflicts have been resolved by the stakeholders, a utility/priority value is determined for every requirement. Based on this priority value, an ordered list of requirements is presented to the stakeholders for further action (e.g., release planning). In sharp contrast to traditional group recommendation approaches which do not take into account the aspects of *delegate voting* (*liquid democracy*) [16], our approach aims to make voting processes more flexible by allowing to transfer voting rights to stakeholders who are the experts with respect to specific requirements and interest dimensions. In other words, the approach harnesses the stakeholders' knowledge and its algorithm allows them to prioritize and delegate at scale.

Another OPENREQ service which supports the prioritization and evaluation of requirements is the *Social Popularity Indication* service. It provides further relevant input for the evaluation of the requirements by estimating the relevance of a requirement given its overall sentiment and popularity in social media networks[7]. The tool automatically extracts messages from *Twitter* which are related to the textual content of a requirement. The cleaned messages are then further analyzed considering the sentiment, in order to obtain a relevance score for every requirement. This relevance score is updated every day and refers to the social popularity which serves as an indication on how relevant/popular a requirement is for potential users/markets that a software company may want to address.

[7] At the moment, the support for social networks is limited to *Twitter*.

2.4 Stakeholder Recommendation

The task of assigning suitable stakeholders to requirements is essential for the success of a software project [20]. A complete and correct assignment in the early phases of the RE process is indispensable. With an increasing number of requirements, this task can become very challenging for requirements managers. Stakeholder recommendations can assist requirements managers in the identification of suitable persons who are capable of implementing the requirement or providing a more detailed analysis and description of the requirement. Inspired by existing research [20], OPENREQ comes up with two new content-based recommendation approaches which have been implemented as different services.

As described by Samer et al. [28] (see also [24]), OPENREQ's *Stakeholder Recommendation* service implements the first approach. In contrast to traditional stakeholder assignment where requirements managers decide on who is responsible for a requirement, the basic idea followed by this approach is to involve more stakeholders in the assignment decision process. This includes human and artificial stakeholders. Content-based recommenders act as artificial stakeholders and propose suitable stakeholder candidates for each requirement based on learnt user profiles from historical data. The (human) stakeholders (including the requirements manager and expert stakeholders working in the domain related to the requirement) can enrich the list of already proposed candidates (if necessary) by adding further stakeholders to the candidate list. Moreover, the human and artificial stakeholders are asked to evaluate all proposed candidates individually. Given the complete evaluation of the proposed candidates as input produced by the combined power of the intelligent service and the expert knowledge of the stakeholders, a group recommendation system preselects final candidates to be assigned to the requirement. The requirements manager can then either accept the final candidates proposed by the group recommender (no action is required) or choose an alternative candidate (action is required) in exceptional cases. The main benefit of this group-based evaluation approach represents the potential to significantly reduce the overall involvement of the requirements managers in this particular task and it can also lower the risk of overseeing suitable stakeholders.

Our second approach is implemented by the service *Issue Prioritizer* and it addresses a slightly different scenario where stakeholders receive a list of recommended requirements based on a user profile (see [29]). Such scenarios typically occur in large open-source projects or in large commercial projects with highly fluctuating groups of employees/developers. Our approach uses content-based filtering and is based on supervised machine learning. The service builds a keyword profile based on "old" requirements/issues resolved by a stakeholder/contributor in the past and aims to find new requirements which are similar in terms of the content. In contrast to standard content-based filtering, we do not use similarity metrics but exploit the potential of machine learning by using large requirement/issue datasets. Based on the keyword profile, relevant requirements matching the stakeholder's interests are recommended using supervised classification techniques. Thereby, our content-based classification approach accepts a single feature vector (consisting of keywords of the current requirement and the keywords of the current stakeholder's keyword profile) as input and uses binary classification to predict whether the given requirement is suitable for the stakeholder or not (for more technical details, we refer to [29]). Since the approach is fully automated and does not require any input from other stakeholders (such as evaluations of proposed candidates or suggestions for candidates), the approach is more suitable to be applied in projects where stakeholders/developers can start to work on a new requirement immediately (without waiting for permission from a requirements manager) which is typically the case in large open-source projects. Moreover, the approach also supports flexible working environments where onboarding of newcomers (i.e., new contributors who want to join the project, but are not yet known by the community) is of high importance for the project (see also [35]).

2.5 Release Planning and Configuration

Release planning deals with the assignment of requirements to releases. It usually follows the principle of *requirements triage* which is accounted for by the fact that there is only a limited amount of available resources in a project [23]. According to *requirements triage*, requirements are classified into (1) requirements which should not be assigned to any release, (2) requirements which have to be included in an early (or the next) release, and (3) requirements whose assignment to an early release is optional.

On an abstract level, this procedure can be regarded as a configuration problem. Raatikainen et al. [26] and Felfernig et al. [14] present a recommendation approach in terms of a configuration system which has been developed within the scope of OpenReq. In their prototype service (named *Release Planning and Consistency Check* service), the aforementioned requirements triage settings (the three types) are modelled as constraints and serve as main input for the service. Further constraints are dependencies defined between the requirements (see Section 2.2), the release dates/deadlines, and the maximum time capacity of each release which limits the number of requirements that can be assigned to the release (based on the specified time effort of the requirement). In addition to these constraints, the complete sets of requirements and releases are used as additional input for the service. Based on the input, the service uses knowledge-based configuration techniques to automatically generate and suggest a list of different release plan candidates which satisfy all defined constraints and represent possible/feasible solutions. The requirements manager can review the recommended list and select one solution as final release plan.

2.6 Quality Assurance

Quality assurance deals with the aspect-oriented evaluation of requirements. The aspects which should be evaluated, represent qualitative attributes such as *feasibility* (economic vs. technical feasibility), *consistency* (no requirement must conflict with another), *completeness* (the requirements model must include all necessary requirements), *understandability* (readability/understandability quality of the requirement's description), and *reusability* (for future software projects) [13]. Quality Assurance in RE represents the backbone of the RE process and is a highly important measure for preventing mistakes and defects in the development of software products. To that end, our OpenReq team has developed two services which facilitate and foster quality assurance in software projects.

The OpenReq *Orchestration service* (introduced in Section 2.1), also provides methods and statistics of collected community data from social media channels. More precisely, the service presents general statistics of the analyzed tweets/messages and can automatically keep track of recent activities and changes in the social media channels which are linked to a software project. The statistics visualized by the tool serve as a fundamental means for requirements managers to identify, analyse, and understand the users' desires and the reported problems the users face while using the software product.

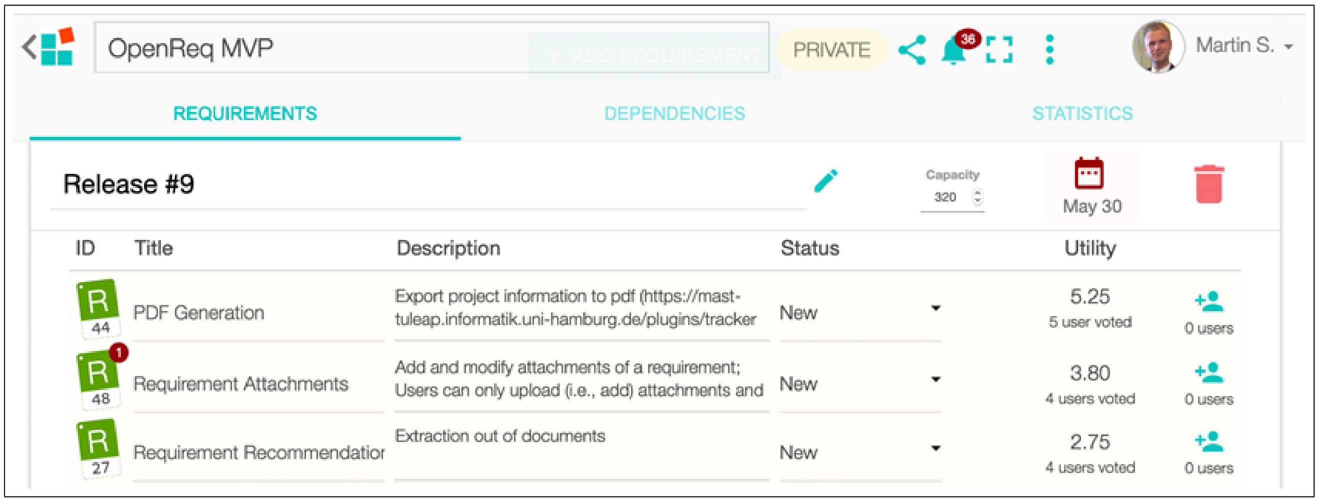
Since requirements are usually described using natural language, the textual descriptions of the requirements can often become inherently ambiguous. The serious consequences caused by such ambiguities are often misleading information, inconsistent descriptions, or poor understandability of the requirements. This can further lead to fatal mistakes or defects in the development phase. Common approaches to detect ambiguities are often rule-based and follow certain guidelines. The service *improving requirements quality* is based on this idea and assists stakeholders in improving ambiguous and unclear definitions of their requirements. Besides pure ambiguity detection, the service provides detailed explanations for each detected ambiguity to the user in terms of basic graphical visualizations. These visualizations represent indications that mark requirements that "smell" which means that the stakeholders should look again at the textual formulations (e.g., if there are some ambiguities introduced by natural language). Further research will be needed to fine-tune the introduced concepts. The ambiguity analysis of the service focuses on syntactic and semantic issues in the wording and phrasing of the requirements. This way, the service helps to improve the quality of a project's requirements model. This avoids serious consequences (mentioned above) which can cause increased time effort and follow-up costs.

3 OpenReq User Interface

In order to provide stakeholders a convenient and user-friendly central access to OpenReq's recommendation tool suite presented in Section 2, we developed a web-based RE platform called OpenReq!Live. OpenReq!Live is a free collaborative RE platform[8] which fosters the cross-fertilization of ideas between stakeholders and allows them to jointly manage their projects and take full advantage of the recommendation power provided by OpenReq's tool suite. OpenReq!Live is capable of tackling all of the everyday RE tasks stakeholders face in their software projects. From a technical viewpoint, the platform represents a central hub which combines the functionality of the most relevant OpenReq services. The platform provides all necessary functionality to create and update the requirements model of a project which includes requirements, releases, dependencies, and further release-related constraints (such as the maximum requirement capacity defined for a release or the release date/deadline) of a project. Moreover, on a project's main page, the platform presents a compact overview of the project which illustrates the defined project structure at a glance (see Figure 2). The users of the platform can modify and update the requirements and releases directly on this page.

The evaluation of the requirements defines the basis for the utility-based prioritization discussed in Section 2.3. Figure 3 shows an example of an argumentation-based MAUT rating interface. Stakeholders are asked to provide textual feedback (in terms of arguments) for every requirement. The stakeholders argue on issues related to the requirement and manually classify the sentiment of their arguments (*positive* (PRO), *neutral* (NEUTRAL), or *negative* (CON)). Moreover, based on the type of issue, every argument has to be assigned to at least one interest dimension $\{$*profit*, *effort*, *risk*$\}$ (see Section 2.3 for more details). Strongly diverging arguments assigned to the same interest dimension indicate strong disagreement and appear as a conflict in the interface. The involved stakeholders have to discuss the issue and resolve the conflict by reevaluating the requirement (i.e., providing improved arguments based on the outcome of the discus-

[8] https://live.openreq.eu



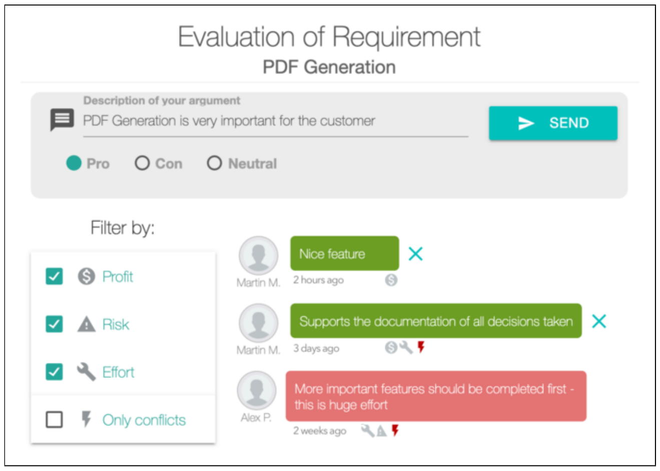
ID	Title	Description	Status	Utility	
R 44	PDF Generation	Export project information to pdf (https://mast-tuleap.informatik.uni-hamburg.de/plugins/tracker	New	5.25 5 user voted	0 users
R 48 (1)	Requirement Attachments	Add and modify attachments of a requirement; Users can only upload (i.e., add) attachments and	New	3.80 4 users voted	0 users
R 27	Requirement Recommendatior	Extraction out of documents	New	2.75 4 users voted	0 users

Figure 2. Overview of an example project in OpenReq!Live. Requirements (they consist of a unique ID, title, description, and status) are listed on the page and ordered by a utility value. Each release has constraints such as the deadline of the release and a maximum capacity value (in hours) which limits the possible number of requirements that can be assigned to the release. The OpenReq services (see Section 2.3) have been integrated into the user interface. For example, the red labelled numbers indicate issues (such as requirement duplicates, ambiguities in a requirement's description text, etc.) reported by some of the services.

sion). This leads to more information exchange between stakeholders and has an positive impact on the quality of the prioritization process. After the eliminiation of all conflicts, the system computes a utility/priority value for each requirement using MAUT and ranks the requirements based on their priority (see Figure 2).

Besides OpenReq!Live, a plugin for the Eclipse open-source community has also been developed within the scope of OpenReq. The plugin aims to bring RE for open-source communities to the next level by helping the communities to reduce time and effort in the smart assignment of their requirements/issues to suitable developers as well as to attract many newcomers in onboarding scenarios. The plugin runs inside the Eclipse IDE[9] (*integrated development environment*) which is the favored IDE of most developers in the community, and fetches requirements/issues relevant for the user/developer from OpenReq's issue prioritizer service (see Section 2.3). The service connects to the web API of the issue tracking platform Bugzilla[10] and extracts new unresolved issues/requirements from the Eclipse project. The plugin is available for download on the Eclipse Marketplace[11] and provides a graphical user interface that shows the recommended requirements/issues to the developers (see Figure 4). Moreover, the plugin also allows developers to give feedback on each recommendation. Developers can give feedback in three different ways: *like button* (indicates that the recommendation was helpful), *dislike button* (indicates that the recommended issue is not relevant for the developer), and *snooze button* (indicates that the issue is not of highest relevance at the moment and the developer should be reminded about the issue again a few weeks later). Our recommendation service uses the individual feedback provided by the developers as additional input, in order to further improve the prediction quality.

To summarize, the developed user interfaces have large potential to enhance and speed up RE by making intelligent technologies developed in OpenReq accessible in a user-friendly fashion for software development institutions as well as open-source communities. All user-interfaces are applicable on different platforms, such as Windows, Linux, and MacOS. In addition to that, OpenReq!Live runs on a wide variety of web browsers and also supports various computing devices including different mobile and desktop devices.



PDF Generation is very important for the customer

Martin M. — Nice feature — 2 hours ago

Martin M. — Supports the documentation of all decisions taken — 3 days ago

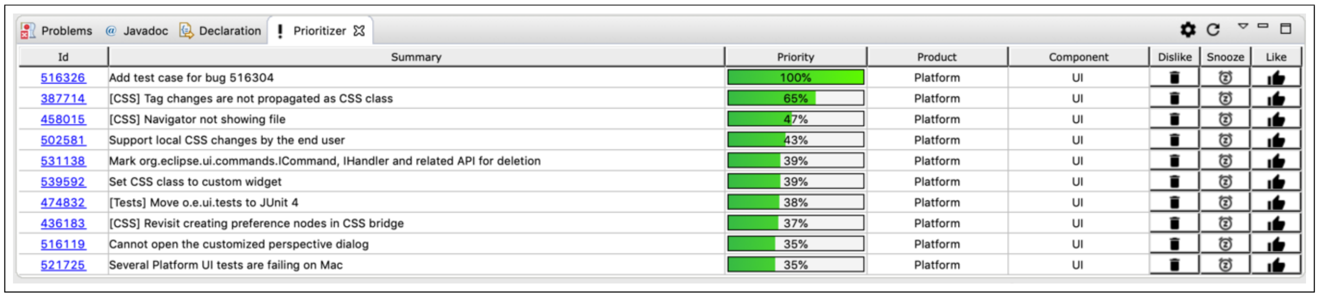
Alex P. — More important features should be completed first - this is huge effort — 2 weeks ago

Figure 3. Argumentation-based rating interface which allows stakeholders to exchange arguments for/against a requirement. Each argument must be assigned to one interest dimension. Negative arguments are highlighted in red, positive arguments in green, and neutral arguments in orange.

4 User Studies and Benefits

In this section, we present representative evaluations and study results of a narrowed list of selected empirical and user studies. All studies have been conducted within the scope of the OpenReq research project and follow the purpose to evaluate the performance,

[9] http://www.eclipse.org
[10] https://www.bugzilla.org
[11] https://marketplace.eclipse.org/content/openreq-eclipse-ide-bug-prioritizer



Id	Summary	Priority	Product	Component	Dislike	Snooze	Like
516326	Add test case for bug 516304	100%	Platform	UI	🗑	⏰	👍
387714	[CSS] Tag changes are not propagated as CSS class	65%	Platform	UI	🗑	⏰	👍
458015	[CSS] Navigator not showing file	47%	Platform	UI	🗑	⏰	👍
502581	Support local CSS changes by the end user	43%	Platform	UI	🗑	⏰	👍
531138	Mark org.eclipse.ui.commands.ICommand, IHandler and related API for deletion	39%	Platform	UI	🗑	⏰	👍
539592	Set CSS class to custom widget	39%	Platform	UI	🗑	⏰	👍
474832	[Tests] Move o.e.ui.tests to JUnit 4	38%	Platform	UI	🗑	⏰	👍
436183	[CSS] Revisit creating preference nodes in CSS bridge	37%	Platform	UI	🗑	⏰	👍
516119	Cannot open the customized perspective dialog	35%	Platform	UI	🗑	⏰	👍
521725	Several Platform UI tests are failing on Mac	35%	Platform	UI	🗑	⏰	👍

Figure 4. ECLIPSE plugin that recommends relevant requirements/issues to the active developer using content-based filtering based on supervised learning.

usability, or prediction quality of the developed tools and approaches which have been discussed in Section 2. Most of the presented user studies were conducted in real-world scenarios with industry partners and some of our evaluated recommendation tools are also currently used by these industry partners.

Issue Prioritizer Service I (Section 2.4). We conducted a small-scale usability study (N=11 participants) with the developed Eclipse Plugin that calculates individual issue lists for the contributors of the Eclipse project by taking into account the individual preferences and the history. The persons who participated in our study were developers of the public Eclipse open-source project and 7 out of 11 participants (63.63%) either rated the usability to be good (2 participants) or even excellent (5 participants). Moreover, the quality of the results of the plugin satisfied all of the participants' expectations (5 participants (45.45%) stated that the recommendations were helpful and for 6 participants (54.55%) the recommendations were very helpful). By using the plugin, the participants of the user study could also take a look at their keyword profile. All participants stated that suitable keywords were found and the list of recommended requirements/issues was very accurate. Moreover, the participants were asked to estimate their perceived time savings in finding suitable requirements by using the plugin (compared to manual finding) within a range between 0 and 100, whereby 0 refers to no time savings and 100 refers to significant/high time savings. The results show that the perceived individual time savings were estimated very high on average (average: 74.18, median: 91.00, standard deviation: 31.92). Regarding future improvements, some of the participants mentioned that they would also see huge benefits if the developed tool could also be connected to other issue tracking systems (such as GITHUB issues or JIRA), in addition to BUGZILLA.

Issue Prioritizer Service II (Section 2.4). To demonstrate the potential of the content-based recommendation approach used by our *Issue Prioritizer Service*, we trained and tested our tool with different classifiers (*Naive Bayes* (our baseline), *Decision Tree*, and *Random Forest*). We used three large issue/requirement datasets of different open-source projects (ECLIPSE (N=141117), MOZILLA (N=751961), and LIBREOFFICE (N=47542)) to compare the performance of the different classifiers. The models used optimized hyper-parameter combinations determined via Grid search. In sharp contrast to already existing issue recommendation approaches, our approach is generally applicable which means that it does not focus on a single target group (such as newcomers). Considering this difficulty, the results show that our recommendation approach performs significantly better with *Random Forest* classification (in case of Eclipse, precision: 0.88, recall: 0.55, f1-score: 0.68) than already available and comparable approaches. Furthermore, the approach based on *Random Forest* classification also considerably outperforms our Naive Bayes baseline (in case of Eclipse, precision: 0.53, recall: 0.29, f1-score: 0.38) as well as the simple baseline of the best random predictor for this specific recommendation task (in case of Eclipse, precision: 0.10, recall: 1.0, f1-score: 0.18). Note that we also compared our approach with a standard content-based approach that searches for k-nearest neighbor requirements based on different similarity metrics (e.g., cosine similarity). However, the results of this standard approach were quite poor and only slightly above the best random predictor baseline. For this reason, we did not include the standard content-based approach as baseline and decided to use *Naive Bayes* which represents a more reliable baseline for this specific recommendation task. Further detailed results and the scores achieved on the other two datasets (MOZILLA and LIBREOFFICE) are presented in [29]. The bottom line is that the developed approach represents a good orientation towards future work which will focus on further improving the performance by using deep learning.

Argumentation-based rating interface (Section 2.3). To evaluate our argumentation-based rating approach, we conducted a large-scale user study with N=313 students in a RE-related course at our university [31]. The students worked in groups of 4–6 students (60 teams) and each team developed a tourist information software. All teams had to use OPENREQ!LIVE to maintain their software project and apply OPENREQ!LIVE's prioritization functionality (see also Section 3 and Figure 2). In this study, we compared three different prioritization approaches/versions: a *one-dimensional rating interface* (baseline), a *multi-dimensional rating interface* (MAUT-based version), and our *argumentation-based rating interface* (see Figure 3). We randomly assigned 20 teams to each rating version and disguised all requirement ratings/arguments of other team members until the current user evaluated the requirement to avoid psychological (cognitive) biases related to the hidden profile theory (see [10, 37]). To counteract further biases related to anchoring effects, we did not inform the 60 groups and our 4 study assistants (who were responsible for the final assessment and grading of the student projects) about the existence of different UI variants during the study [32, 37] (double-blind). In order to ensure comparability of the study results and to compute a utility/priority value in each rating version, we used equal rating scales in all three versions. In the one- and multi-dimensional rating interfaces, students could rate a requirement within the range of 1–5 points. In the case of arguments, there existed three different sentiment levels (positive, neutral, negative) and we assigned 5 points to positive, 3 points to neutral, and 1 point to negative arguments. The evaluation results show that the argumentation-based rating interface helped to significantly increase the communication and interaction rate (includes created ratings and adaptations of existing ratings) among participants in the context of requirements prioritization. For example, groups that were confronted with the argumentation-based rating interface had 10.4 interactions per requirement on average (standard deviation: 1.03), whereas the interaction rates of the groups using the multi-dimensional rating interface (avg.: 5.4, std. dev.: 0.65) as well as the one-dimensional version (avg.: 4.5, std. dev.: 0.61) were significantly lower. The comparison of the grades the teams achieved at the end of the course and the quality of the developed software also reveals interesting differences/findings. The results of our analysis indicate that the teams that used the argumentation-based and multi-dimensional rating interfaces performed significantly better than our baseline (the one-dimensional version) in terms of grades and software quality (for more details, see [31]). Even though the explanatory power of the evaluation results is partly limited due to the university setting (since the study was conducted with students in a university course), the results are very significant and expressive which allows to draw the conclusion that argumentation-based rating interfaces foster information exchange among stakeholders and have a positive impact on the quality of requirements evaluation and prioritization.

OpenReq Classification Service (Section 2.1). The classification of whether a piece of text defines a requirement (REQ) or represents a textual description (PROSE) of a requirement is treated as a binary classification task. The utility of the OPENREQ CLASSIFICATION SERVICE using this binary classification approach, was evaluated with ten datasets of different projects (30,000 requirements altogether). The datasets consist of samples labelled as REQ and PROSE which represent requirements and descriptive paragraphs of real-world industry projects. All datasets have been cleaned from irrelevant information, manually labelled, and evaluated by a small group of requirements managers and RE experts working at Siemens in Vienna (our industry partner). The finally obtained datasets represent a solid ground truth that reflects correctness and clearness of the data for the training of a reliable prediction model. Due to the nature of the datasets, there existed a high class-imbalance between REQ and PROSE classes, whereby REQ represented the majority class in all datasets (around 90%). The labelled datasets were used to evaluate our requirement classification tool. Our partner Siemens perceived the achieved level of prediction quality very promising (considering the highly imbalanced data) and thus integrated the developed approach in their (RE-related) business processes used in their real-world commercial operations (see [8] for detailed results). Hence, the outcome of this work represents a significant work reduction for requirements managers and experts during requirement elicitation.

Orchestration service (Section 2.1). The underlying approach was evaluated in a series of classification experiments. In these experiments, classifiers based on traditional supervised machine learning as well as deep learning were trained to find requirements-relevant information in English and Italian tweets as well as in English app reviews. All classifiers were trained to distinguish between the classes *problem reports*, *inquiries*, and *irrelevant* (see Section 2.1). The Twitter datasets consisted of labelled tweets (around 10,000 in English and 15,000 in Italian) from the telecommunication domain and the app review datasets included around 6,000 app reviews. The learnt prediction models performed well on all datasets. The best results were achieved on the dataset consisting of English app reviews (*problem report*: precision: 0.83, recall: 0.75, f1 score: 0.79; *inquiry*: precision: 0.68, recall: 0.76, f1 score: 0.72; *irrelevant*: precision: 0.88, recall: 0.89, f1 score: 0.89). In general, the results indicate that, within the used experimental settings, the approaches based on traditional machine learning could achieve comparable results to deep learning. For more detailed results, we refer to [33].

5 Related and Future Work

Related Work. In order to address the problems discussed in Section 1, a lot of research has been conducted with respect to the aforementioned aspects. Different scientific works identify a need for intelligent tool support to help requirements engineers and stakeholders in the different stages of the RE process for complex projects [3, 22, 23]. A recommendation approach, for example, may be helpful to suggest requirements to a stakeholder, who already dealt with the same topics in the past. Alenezi et al. [1] present a content-based approach which is used to predict and recommend relevant bugs based on the experience of the stakeholder. As for finding dependencies between requirements, existing research shows that *natural language processing* (NLP) techniques can be applied [4, 6]. Although there already exist several tools for parts of the RE process, the unique nature of the projects makes it rather complex to find methods which are valid for every project. Ninaus et al. [23] present a small RE platform called INTELLIREQ which applies recommendation techniques to support stakeholders in different RE tasks. INTELLIREQ goes one step further and aims to tackle this challenge by providing intelligent tools, approaches, and techniques for different RE scenarios to the RE community. INTELLIREQ utilizes basic recommendation techniques to support stakeholders in common RE tasks. However, when compared to OPENREQ's RE platform OPENREQ!LIVE, INTELLIREQ only provides a small and limited set of very basic features which do not go beyond the level of semi-automated learning.

Future Work. The OPENREQ!LIVE user interface will be continuously improved in terms of integrating functionalities to automatically annotate, evaluate, and group requirements as well as to open its application for existing Requirements Engineering tools such as, IBM DOORS by providing interfaces to such systems. Furthermore, text processing techniques are still an active research topic which receives contributions from several domains. In our future work, we plan to focus on this issue and compare some of our developed content-based recommendation approaches with more sophisticated approaches based on deep learning.

6 Conclusion

This paper provided an overview of new RE recommendation tools developed in the context of the European research project OPENREQ. The RE process can be viewed as a comprehensive decision-driven process consisting of different phases in which many stakeholders are involved. A high complexity in this process implicates a high risk of project failure. In this context, we presented the OPENREQ project as an example of a research project in the field of RE. We introduced the OPENREQ approach which addresses the aforementioned issue and presented *OpenReq*'s RE recommendation tool suite consisting of innovative recommendation solutions which are applicable in different RE-related tasks. The variety of intelligent technologies and services shown in the paper fit seamlessly into OPENREQ!LIVE which is a RE platform that provides a central interface for stakeholders to access the most relevant OPENREQ services in a user-friendly way. The OPENREQ project is important, not only for tool support, but its user studies can guide research in directions that are supported by actual real-world cases. the OPENREQ approaches are supported by the evaluation of OPENREQ methods by industry partners from three different areas (*railway safety systems*, *telecommunications*, *distributed open-source development*). The evaluation results clearly prove the point that the developed concepts are able to significantly improve the RE process, in terms of more information exchange among stakeholders, reduced costs and time effort, and increased quality of the process output.

ACKNOWLEDGEMENTS

The work presented in this paper has been conducted within the scope of the Horizon 2020 project OPENREQ (732463).

REFERENCES

[1] M. Alenezi, S. Banitaan, and M. Kenneth, 'Efficient bug triaging using text mining', *Proceedings of the 5th International Conference on Computer Engineering and Technology, 2013.*, **8**, 2185–2190, (01 2013).

[2] M. Atas, R. Samer, and A. Felfernig, 'Automated identification of type-specific dependencies between requirements', in *2018 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2018, Santiago, Chile, December 3-6, 2018*, pp. 688–695, (2018).

[3] C. Castro-Herrera, C. Duan, J. Cleland-Huang, and B. Mobasher, 'A recommender system for requirements elicitation in large-scale software projects', in *Proceedings of the 2009 ACM Symposium on Applied Computing*, SAC '09, pp. 1419–1426, New York, NY, USA, (2009). ACM.

[4] R. Chitchyan and A. Rashid, 'Tracing requirements interdependency semantics', in *Workshop on Early Aspects*, (Jan 2006).

[5] S. Deerwester, S.T. Dumais, G. Furnas, T. Landauer, and R. Harshman, 'Indexing by latent semantic analysis', *Journal of the American Society for Information Science*, **41**(6), 391–407, (09 1990).

[6] G. Deshpande, 'Sreyantra: Automated software requirement interdependencies elicitation, analysis and learning', in *41st International Conference on Software Engineering (ICSE'19)*, Canada, (2019).

[7] M.D. Ekstrand, J.T. Riedl, and J.A. Konstan, 'Collaborative filtering recommender systems', *Found. Trends Hum.-Comput. Interact.*, **4**(2), 81–173, (February 2011).

[8] A. Falkner, C. Palomares, X. Franch, G. Schenner, P. Aznar, and A. Schoerghuber, 'Identifying requirements in requests for proposal: A research preview', in *Requirements Engineering: Foundation for Software Quality*, eds., Eric Knauss and Michael Goedicke, pp. 176–182, Cham, (2019). Springer International Publishing.

[9] A. Felfernig, L. Boratto, M. Stettinger, and M. Tkalcic, *Group Recommender Systems – An Introduction*, Springer, 2018.

[10] A. Felfernig, L. Boratto, M. Stettinger, and M. Tkalčič, *Biases in Group Decisions*, 145–155, 03 2018.

[11] A. Felfernig, M. Jeran, G. Ninaus, F. Reinfrank, S. Reiterer, and M. Stettinger, *Basic Approaches in Recommendation Systems*, 15–37, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[12] A. Felfernig, G. Ninaus, H. Grabner, F. Reinfrank, L. Weninger, D. Pagano, and W. Maalej, 'An overview of recommender systems in requirements engineering', in *Managing Requirements Knowledge*, chapter 14, 315–332, Springer, (2013).

[13] A. Felfernig, G. Ninaus, H. Grabner, F. Reinfrank, L. Weninger, D. Pagano, and W. Maalej, *An Overview of Recommender Systems in Requirements Engineering*, 315–332, 04 2013.

[14] A. Felfernig, J. Spöcklberger, R. Samer, M. Stettinger, M. Atas, J. Tiihonen, and M. Raatikainen, 'Configuring release plans', in *Proceedings of the 20th Configuration Workshop, Graz, Austria, September 27-28, 2018.*, pp. 9–14, (2018).

[15] D. Goldberg, D. Nichols, B.M. Oki, and D. Terry, 'Using collaborative filtering to weave an information tapestry', *Commun. ACM*, **35**(12), 61–70, (December 1992).

[16] T. Johann and W. Maalej, 'Democratic mass participation of users in Requirements Engineering?', in *23rd International Requirements Engineering Conference (RE)*, pp. 256–261, Ottawa, ON, Canada, (2015).

[17] T.K. Landauer, P.W. Foltz, and D. Laham, 'An introduction to latent semantic analysis', *Discourse Processes*, **25**(2-3), 259–284, (1998).

[18] D. Leffingwell, 'Calculating the return on investment from more effective requirements management', **10**, 13–16, (1997).

[19] J. Li, R. Jeffery, K.H. Fung, L. Zhu, Q. Wang, H. Zhang, and X. Xu, 'A business process-driven approach for requirements dependency analysis', in *Business Process Management*, eds., Alistair Barros, Avigdor Gal, and Ekkart Kindler, pp. 200–215, Berlin, Heidelberg, (2012). Springer Berlin Heidelberg.

[20] S.L. Lim, D. Quercia, and A. Finkelstein, 'Stakenet: Using social networks to analyse the stakeholders of large-scale software projects', in *Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ICSE '10, pp. 295–304, New York, NY, USA, (2010). ACM.

[21] J. Masthoff, 'Group Recommender Systems: Aggregation, Satisfaction and Group Attributes', *Recommender Systems Handbook*, 743–776, (2015).

[22] B. Mobasher and J. Cleland-Huang, 'Recommender systems in requirements engineering', *AI Magazine*, **32**(3), 81–89, (Jun. 2011).

[23] G. Ninaus, A. Felfernig, M. Stettinger, S. Reiterer, G. Leitner, L. Weninger, and W. Schanil, 'Intellireq: Intelligent techniques for software requirements engineering', in *Proceedings of the Twenty-first European Conference on Artificial Intelligence*, ECAI'14, pp. 1161–1166, Amsterdam, The Netherlands, (2014). IOS Press.

[24] C. Palomares, X. Franch, and D. Fucci, 'Personal recommendations in requirements engineering: The openreq approach', in *Requirements Engineering: Foundation for Software Quality*, eds., Erik Kamsties, Jennifer Horkoff, and Fabiano Dalpiaz, pp. 297–304, Cham, (2018). Springer International Publishing.

[25] M.J. Pazzani and D. Billsus, *Content-Based Recommendation Systems*, 325–341, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[26] M. Raatikainen, J. Tiihonen, T. Männistö, A. Felfernig, M. Stettinger, and R. Samer, 'Using a feature model configurator for release planning', in *Proceedings of the 22Nd International Systems and Software Product Line Conference - Volume 2*, SPLC '18, pp. 29–33, New York, NY, USA, (2018). ACM.

[27] G. Ruhe, *Product release planning: methods, tools and applications*, CRC Press, 2010.

[28] R. Samer, M. Atas, A. Felfernig, M. Stettinger, A.A. Falkner, and G. Schenner, 'Group decision support for requirements management processes', in *Proceedings of the 20th Configuration Workshop, Graz, Austria, September 27-28, 2018.*, pp. 19–24, (2018).

[29] R. Samer, A. Felfernig, and M. Stettinger, 'Towards issue recommendation for open source communities', in *2019 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2019, Thessaloniki, Greece, October 14-17, 2019*, WI '19, pp. 164–171. ACM, (2019).

[30] R. Samer, M. Stettinger, M. Atas, A. Felfernig, G. Ruhe, and G. Deshpande, 'New approaches to the identification of dependencies between requirements', in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 1265–1270, (Nov 2019).

[31] R. Samer, M. Stettinger, and A. Felfernig, 'Group recommender user interfaces for improving requirements prioritization', Technical report, Institute of Software Technologies, Graz, Austria, (2020).

[32] S. Schulz-Hardt, F. Brodbeck, A. Mojzisch, R. Kerschreiter, and D. Frey, 'Group decision making in hidden profile situations: Dissent as a facilitator of decision quality', *Journal of Personality and Social Psychology*, **91**(6), 1080–1093, (2006).

[33] C. Stanik, M. Haering, and W. Maalej, 'Classifying multilingual user feedback using traditional machine learning and deep learning', *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW), Jeju Island, South Korea*, 220–226, (2019).

[34] C. Stanik and W. Maalej, 'Requirements intelligence with openreq analytics', in *2019 IEEE 27th International Requirements Engineering Conference (RE), Jeju Island, South Korea*, (2019).

[35] C. Stanik, L. Montgomery, D. Martens, D. Fucci, and W. Maalej, 'A simple nlp-based approach to support onboarding and retention in open source communities', *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 172–182, (2018).

[36] M. Stettinger, A. Felfernig, G. Leitner, S. Reiterer, and M. Jeran, 'Counteracting Serial Position Effects in the CHOICLA Group Decision Support Environment', in *20th ACM Conference on Intelligent User Interfaces (IUI2015)*, pp. 148–157, Atlanta, Georgia, USA, (2015).

[37] A. Tversky and D. Kahneman, *Judgment under Uncertainty: Heuristics and Biases*, 141–162, Springer Netherlands, Dordrecht, 1975.

[38] J. Xuan, H. Jiang, Z. Ren, and W. Zou, 'Developer prioritization in bug repositories', pp. 25 – 35, Zürich, Switzerland, (2012).