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

D4.1 OpenReq Approach to Group Decision Support

Lead contractor: TUGraz

Author(s): TUGraz

Submission date: July 2018. Resubmitted November 2018

Dissemination level: PU

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

Abstract:
Requirements Engineering related decisions are often taken by groups. One major focus of the OpenReq project is to provide techniques that support different types of group decision processes. Examples of such decision processes are prioritization and release planning, i.e., the ranking of requirements with regard to criteria such as effort, risk, and potential profit, and the related task of assigning requirements to releases (i.e., release planning). Decision making is also influenced by different types of biases which are responsible for suboptimal decisions taken by single users as well as groups. With this document, we provide an overview of the requirements for the OpenReq group decision service/component. We present algorithmic approaches to support group decision scenarios and the show role of group decision support services in the overall OpenReq infrastructure.

As required on the Consolidated Project Review Report received on October 1st, 2018, this Deliverable has been revised following the recommendations of the Reviewers and resubmitted again

Table of Contents

List of Figures

List of Tables

1.Introduction

In this section, we motivate the need of group decision support and then sketch the overall goal and vision related to the support of OpenReq group decision processes. The terming "group decision" refers to scenarios where a group of persons (in contrast to a single person) is in charge of taking a decision. Such decisions should as far as possible take into account the opinions and preferences of individual group members [JWF+2015] [FBS+2018].

1.1 Motivation

There are different reasons why decisions of single persons and groups in general, and especially in requirements engineering (RE), have a suboptimal outcome - example reasons are summarized in Figure 1.

Figure 1: Issues related to group decision processes in requirements engineering.

First, stakeholders in charge of making a decision are often uninformed, i.e., do not have the knowledge to take an optimal decision. *Second*, in many cases the opinion of experts with regard to specific topics is not taken into account. This can happen, for example, if their expertise is not obvious for those who are in charge of selecting a team of stakeholders for evaluating the quality of requirements. *Third*, different types of decision biases can lead to suboptimal decisions. For example, *anchoring* [SFL+2015] triggers a situation where the visibility of the preferences of a stakeholder has an influence on the requirement evaluation behavior of other stakeholders. *Fourth*, stakeholders are not aware of the existence of decision biases which is accompanied by an inability to trigger counter-measures. *Fifth*, decisions can be manipulated, for example, stakeholders evaluate requirements in specific ways with the goal to push their preferred requirements. *Sixth*, reasons for decisions are not well explained or not documented, for example, the choice of a specific prioritization remains unclear if the evaluations of individual stakeholders suggest a different ranking. *Seventh*, decision support tools such as release planners are not able to determine optimal solutions simply due to the fact that the underlying problem is too complex. To handle that complexity, often local search

approaches are applied that are not able to determine optimal solutions. *Finally*, as a direct or indirect result of all the mentioned aspects, groups of stakeholders are not able to identify optimal decisions, for example, prioritizations or release plans. In such contexts, group members are not able to identify the so-called *hidden profile*, i.e., the information that is crucial for being able to make an optimal decision.

1.2 Vision

Our reasons for developing decision support technologies for requirements engineering scenarios are summarized in Figure 2.

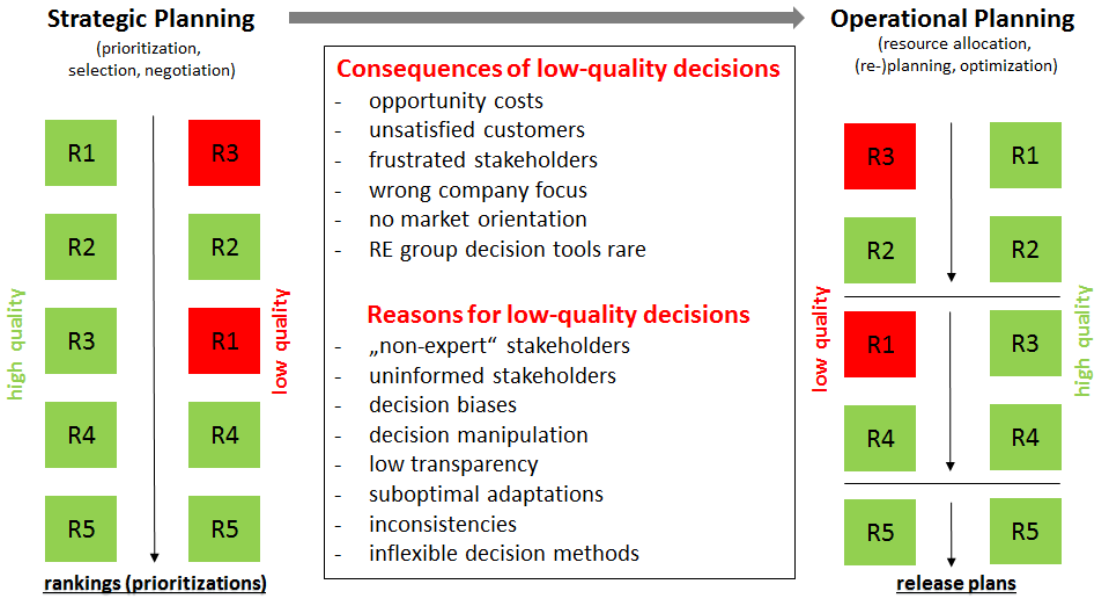


Figure 2: Basic OpenReq group decision scenarios: strategic & operational planning.
Requirements marked as “red” represent faulty prioritizations, for example, requirement “R1” received priority 3 although it is the most relevant one.

First, suboptimal prioritizations and release plans can lead to situations where important requirements get postponed and - as a result - features required by the market or improving core business processes of the customer (or both) are not available even though customers would be willing to pay the price. *Second*, customers could get unsatisfied if important software features are missing and maybe competitor solutions already provide the required functionality. *Third*, stakeholders become unsatisfied if their business processes are only supported in a suboptimal fashion. *Fourth*, due to a faulty prioritization, a startup company could select the wrong features to be included in a minimal viable product (MVP) which could result in low customer acceptance and maybe also in a situation where potential customers and/or investors decide to invest in a different solution. *Finally*, RE tools providing an in-depth support for group decision tasks do not exist but are needed to be able to better tackle the mentioned challenges [FZN+2011, FSA+TR2018].

In OpenReq, we focus on the support of two major types of decisions which are (1) *strategic planning* (often interpreted as prioritization) and (2) *operational planning* which can be implemented in terms of a release planning process [AFF+2017]. When prioritizing requirements, "high quality" prioritizations are expected where highly relevant requirements should have a corresponding high ranking. In this context, suboptimal prioritizations should be avoided, for example, a triage process [FBS+2018] should not exclude "feasible" requirements of high market potential and low risk due to a faulty evaluation of a requirement. Similar criteria hold in the context of release planning: the relevant requirements should be implemented as early as possible while taking into account existing resource constraints.

The overall OpenReq vision in this context is to develop functionalities and corresponding user interfaces that help to tackle the discussed challenges of decision making in requirements engineering. In order to provide support for the mentioned group decision scenarios, several research questions have to be answered. The research questions are related to objectives defined in the context of workpackage 4 (for an overview see Figure 3).

Research Questions	WP4 Objectives
• How to automatically estimate stakeholder expertise? • How to „algorithmically" take into account stakeholder expertise? • How to make group decisions more „liquid"?	**Group recommendation & decision technologies for RE**
• How to increase stakeholder participation in RE processes? • How to explain group recommendations (e.g., to foster consensus)? • How to avoid manipulations in group decision making?	**Bias-aware & persuasive user interfaces**
• How to support automated planning „socially aware"? • How to evaluate group recommendations? • How to exploit recommenders for efficient solution search?	**Group-aware planning**
• How to efficiently identify & explain inconsistencies to groups? • How to resolve inconsistencies „socially aware"?	**Group-aware consistency management & re-planning**

Figure 3: Objectives of workpackage 4.

First, OpenReq focuses on the development of *group recommendation and decision technologies for requirements engineering*. In OpenReq, recommender systems [FBS+2018, JZF+2010] are considered as a major means to improve the quality of RE support. Especially, group recommendation technologies are taken into account, since these technologies allow to model the preferences of a group as a whole and propose corresponding group decisions. Example research questions that have to be answered in this context are the following. *How to estimate and take into account stakeholder expertise in an automated fashion* and *how to make group decisions more "liquid"*, i.e., how to make the collection of user preferences more flexible to be able to take into account the fact that some stakeholders have more expertise and knowledge that is needed to correctly estimate the relevance of a requirement. *Second*, OpenReq focuses on the development of *bias-aware & persuasive user interfaces* that help to counteract biases and trigger stakeholder actions that are needed to assure the quality of requirement models. An example question in this context is *how to best motivate stakeholders to exchange more information that is needed to be able to take an optimal decision*. *Third*, OpenReq focuses on the development of *group-aware planning* functionalities that support release planning decisions on a group level. An example research question in this context is

how to make automated planning "socially-aware". For example, how to include group aggregation functions [FBS+2018] into solvers that determine release plans [DR2009, AFF+2017]. *Finally,* OpenReq develops group-aware consistency management & re-planning functionalities that help to deal with situations where inconsistencies have to be resolved. Examples thereof are inconsistent stakeholder preferences regarding the assignment of requirements to releases or inconsistent evaluations of requirements within the scope of a prioritization [FSA+2018].

1.3 Role in OpenReq Infrastructure

The role of group decision scenarios in the OpenReq architecture is sketched in Figure 4. Workpackage 4 (the OpenReq group decision support) is primarily responsible for providing different kinds of group decision technologies which include a group- and utility-based prioritization of stakeholder preferences. These preferences are represented in terms of evaluation dimensions (e.g., profit and risk), textual evaluations represented in terms of pro/contra arguments and related sentiments which can help to interpret textual evaluations in a more fine-grained fashion. The group decision services have a strong relationship to *workpackage 5* which acts as a provider of group-based release planning and diagnosis functionalities where the latter is a core technology that can be used to resolve inconsistencies, for example, in requirements evaluations, dependencies between requirements, and assignments of requirements to release plans. Furthermore, *workpackage 4* has a relationship to *workpackage 2* which acts as a sensing component that provides new requirements (e.g., from Twitter channels) to be taken into account in group decision processes. Finally, *workpackage 3* supports the identification of relevant stakeholders who should participate in group decision scenarios. The trial partners who currently exploit OpenReq group decision functionalities are *Siemens* (stakeholder selection and compliance assessment in bidding scenarios) and *Vogella* (prioritization in open source communities). In addition, OpenReq Live is already evaluated by different industry partners of TU Graz and also applied within TU Graz courses such as "Requirements Engineering" and "Object-oriented Analysis and Design".

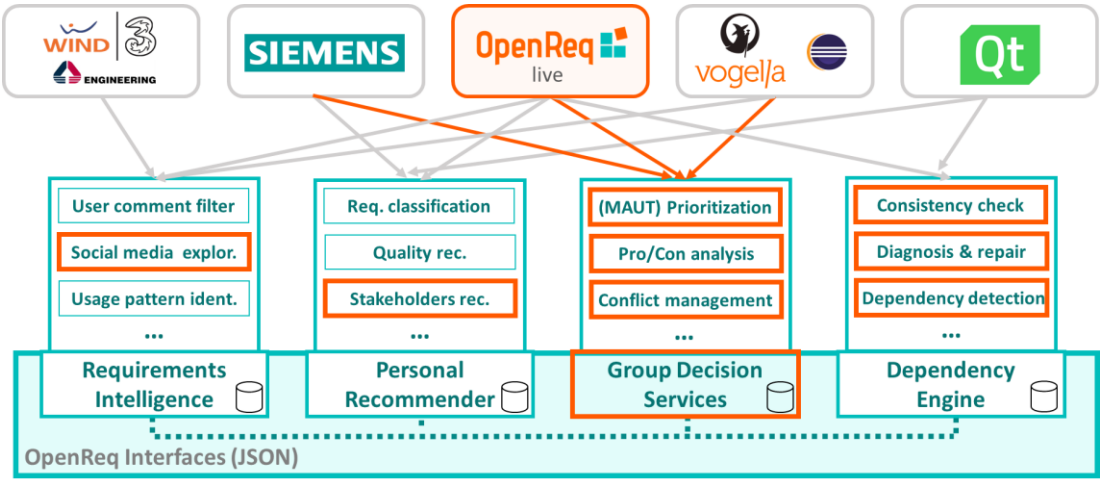


Figure 4: Role of workpackage 4 in the overall OpenReq architecture.
Orange lines denote the relationships of group decision services with other services and applications.

1.4 Structure

The remainder of the deliverable is organized as follows. In Section 2, we provide an overview of related work (*major advancements of existing work in the field are exemplified in more detail in the following sections*). Furthermore, we introduce the main requirements related to group decision support in OpenReq. In Section 3, we provide more insights into the OpenReq architecture as well as the architecture of the group decision component itself. Section 4 contains study results related to industrial decision making practices in the context of requirements engineering as well as an overview of group decision scenarios supported by OpenReq. Section 5 deals with the aspect of decision biases which can negatively influence the quality of group decisions. Explanations are an essential aspect to increase trust in a given recommendation - especially in the context of group recommendations. Section 6 deals with explanation approaches w.r.t. group decisions. Section 7 sketches interfaces between the OpenReq prototype and the other OpenReq components. Section 8 concludes the deliverable and lists relevant OpenReq group decision related publications.

2. Related Work & WP4 Requirements

2.1 Related Work

Requirements Engineering

Requirements Engineering (RE) can be considered as one of the most critical phases in software development processes. Incomplete, low-quality, and suboptimally prioritized requirements can lead to cost explosions and often to the cancellation of a software project [LEF1997]. A major reason is that stakeholders in software projects often take suboptimal decisions. For example, highly relevant requirements are implemented in late releases. Such decisions trigger, for example, opportunity costs since the earlier the software is applicable, the earlier business processes can be supported more efficiently. The reasons for suboptimal decisions are manifold. For example, the relevance of a requirement could have been underestimated simply due to the lack of decision-relevant knowledge of stakeholders who are in charge of prioritizing and release planning. The reason behind this could be a low degree of knowledge exchange between stakeholders. For similar reasons, the quality of individual requirements could have been overestimated. Due to anchoring effects [SFL+2015], the opinion of one stakeholder can have an influence on the opinions of other stakeholders. Polarization effects [ARF2018] can lead to situations where groups take decisions that are riskier compared to the riskiness of decisions taken by individual stakeholders. Such decision biases can be regarded as a major obstacle for high-quality decision making [FBS+2018].

Configuration Technologies & Constraint-based Reasoning

Configuration [FHB+2014, FAT2016] is considered as one of the most successful applications of Artificial Intelligence technologies. It is applied in many domains such as financial services [FIS+2007] and telecommunication [FHB+2014]. Configuration environments are typically single-user oriented, i.e., the underlying assumption is that a specific user is in charge of completing the configuration task. However, considering configuration as a single user task can lead to suboptimal decisions [FAT2016]. For example, release planning is a task that typically requires the engagement of a group of stakeholders where the knowledge and preferences of all stakeholders should be taken into account in order to be able to achieve high-quality decisions [DR2009, AFF+2017]. Configuration technologies can be considered as a major technology to support efficient and at the same time personalized and group-oriented prioritization, re-prioritization, release planning, and release re-planning. Thus, configuration and constraint-based reasoning can be regarded as a major technology that will be developed further within the scope of OpenReq to support group-centred decision scenarios in the RE context.

Existing configuration environments do not take into account the aspect of group configuration [FAT2016]. For non-configurable items such as movies, restaurants, personnel decisions, and music, there already exist proposals how to support related group decision processes. In this context, group recommendation heuristics [FBS+2018] are applied to support groups in their decision making activities. In order to achieve consensus, different decision heuristics are applied which propose decisions acceptable for a group as a whole. For example, the least misery heuristic proposes alternatives which do not represent an absolute no-go for one of the group members. Besides decision heuristics, standard recommendation approaches

[JZF+2010] such as matrix factorization can be applied to predict recommendations acceptable for a group as whole. These approaches rely on existing group recommendations. Based on such information about group selection behavior, corresponding recommendations can be determined for similar groups.

Software Release Planning

Software Release Planning [DR2009, AFF+2017] is a requirements engineering related activity, where groups of users (stakeholders) are deciding about the ordering in which requirements should be implemented. In this scenario, stakeholders have different preferences and knowledge regarding the implementation alternatives. Consequently, requirements-related knowledge must be exchanged as much as possible and existing contradictions in preferences and evaluations have to be resolved. “Holiday planning” is another scenario where a group is in charge of identifying a configuration that is accepted by all group members – examples of related decisions are region to visit, hotel, and activities during the stay. Product Line Scoping [SCH2000] is related to the task of determining boundaries in a product line. This task is a specific type of requirements engineering task and related decisions are crucial for the success of a whole product line effort. Investment Decisions (e.g., project funding) are often taken by a group of users who have to take into account constraints with regard to the overall amount of money that can be invested and the topics projects should deal with. The configuration task in investment decisions is to identify a bundle of project proposals that takes into account the financial limits and includes high-quality proposals.

Prioritization of Requirements

In software projects, resources are typically limited which requires the prioritization of requirements [FSA+2018]. Prioritization is often interpreted as a part of strategic planning where the focus is to select and prioritize requirements that should be included in releases (long-term planning) [AFF+2017]. Decision support in prioritization is extremely important since especially when dealing with large assortments of requirements, manual prioritization processes tend to become very costly [XJR+2012]. In this context, suboptimal prioritizations trigger time wasting due to the implementation of unimportant requirements. There are two basic approaches to prioritize requirements – for an in-depth related analysis we refer to [AFF+2017]. *First*, requirements prioritization can be interpreted as an optimization task where the overall objective is to identify the middle ground, i.e., an aggregation of individual prioritizations into a global prioritization that reflects the least possible level of dissimilarity from all stakeholder-individual prioritizations [KSM+2017]. *Second*, in contrast to approximating individual prioritizations on the basis of optimization functions, utility-based approaches focus on (1) establishing agreement with regard to the evaluation of individual requirements and (2) determining prioritizations [FSA+2018].

Prioritizations following the optimization approach are determined on the basis of individual prioritizations of stakeholders. When following a utility-based approach, preferences of stakeholders are first aggregated and a prioritization is determined thereafter. In the line of basic approaches to determine group recommendations [FBS+2018], the first approach is based on *aggregated prioritizations* where stakeholder-individual prioritizations are known and a recommendation minimizes dissimilarities between the given prioritizations. The second approach is based on *aggregated models* where stakeholder requirement evaluations are

aggregated first and a prioritization is determined on the basis of a group profile (model) derived from requirements evaluations. Aggregated models have the advantage that stakeholders are encouraged to focus their evaluations on specific relevant aspects of a requirement (e.g., dimensions such as profit, risk, and effort) and thus contribute to stable preferences and a higher degree of consensus.

Aggregated prioritizations trigger scalability issues since each stakeholder has to provide a ranked list of requirements as input for the optimization process. Furthermore, due to the computational complexity of the underlying problem, an optimal solution cannot be guaranteed and is often only approximated on the basis of local search algorithms [AFF+2017]. Utility-based approaches focus on evaluations of individual requirements on the basis of different evaluation dimensions (e.g., profit, effort, and risk). This way, stakeholders can focus on evaluating requirements they have knowledge about and the focus of prioritization is first on establishing consensus and thereafter on figuring out the most relevant prioritizations [SFL+2015].

Different algorithmic approaches can be used to support requirements prioritization – for an overview, see, for example, [ASI+2014]. Examples thereof are constraint-based reasoning [TSA1993], incremental preference learning [PSA2013], evolutionary algorithms [KSM+2017], machine learning [AB2013], [TLX+2015], and pairwise preference-based decision making [SV2000]. Utility-based prioritization based on multi-attribute utility theory [DY2005] can be implemented in different variants. *First*, requirements are simply evaluated with regard to a set of predefined interest dimensions and the overall utility of a requirement is determined as a sum of interest dimension specific utilities. *Second*, weights can be introduced to emphasize on specific interest dimensions (e.g., a lower risk is more important than high profits). *Third*, stakeholders can be enabled to define their personal evaluations and utility-based approaches should then be able to aggregate these evaluations and take into account stakeholder weights. Stakeholder weights can be interpreted as ”global”, i.e., there is a global weighting of stakeholders independent of a specific dimension or requirement. If weights are interpreted ”local”, the importance of a stakeholder can be defined on the level of individual requirements or dimensions.

Prioritization criteria differ depending on the requirements engineering scenario. The criteria effort, risk, and profit are often used in settings where a group of stakeholders engaged in the same project is in charge of completing a prioritization task [ASI+2014]. In contrast, in open-source settings, developers are in most of the cases engaged in different projects and also work for different companies. In such scenarios, prioritization is less focusing on establishing consensus between individual stakeholders but more on supporting stakeholders in identifying requirements of relevance to them and to prioritize the important ones by also taking into account global criteria. Examples of criteria in such scenarios are *personal expertise* of a developer and *importance* of a requirement for the community of the stakeholder and the open source community as a whole.

BUGZILLA[1] is an open-source based bug tracking system which supports users from different geographical locations to report their findings with regard to a given set of software components. Users can submit textual descriptions of issues (bugs – also used to represent requirements) and corresponding meta-information, for example, associated components, key-words, and dependencies to other bugs. Reported bugs can be selected by contributors to work on. There are different related approaches to support machine learning based requirements prioritization. The approaches operate on datasets including historical data of previous requirement (bug report in BUGZILLA) selections and try to predict future requirement selections thereof. Utility-based prioritization can be used in interactive scenarios (stakeholders are engaged in an interactive prioritization process) as well as scenarios where requirements are recommended but no further stakeholder interaction is needed for determining a prioritization.

Stakeholder Selection

For the definition and evaluation of project requirements, it is necessary to identify suitable stakeholders who are responsible for the development of these requirements. In addition, an early involvement of these stakeholders in the project is essential for the success of a project. This is because a low involvement of stakeholders in a project can lead to project failure. Project failures are often caused by missing or wrong assignments of stakeholders to requirements in early phases of the requirements engineering process [LKK2004]. Stakeholder recommendations can help to identify persons who are capable of providing a complete analysis and description of software requirements. Recommended stakeholders also need to bring deep knowledge about the corresponding item domain in order to provide precise evaluations of the requirements.

StakeNet [LQF2010] is an application that supports stakeholder identification on the basis of social network analysis. This approach builds a social network on the basis of a set of stakeholders. In this social network, stakeholders are represented by nodes and recommendations articulated by the stakeholders are represented by links. On the basis of such social networks, different social network measures are used for the prioritization of the stakeholders. One example of such a measure is *betweenness centrality* which measures the priority of a certain stakeholder based on the ability of this user to play a role as a broker between separate groups of stakeholders. Castro-Herrera et al. [CDC+2008] and Mobasher et al. [MC2011] introduce a content-based recommendation approach where requirements are grouped by using different clustering techniques. Subsequently, stakeholders are recommended and assigned to these groups on the basis of content-based filtering. In OpenReq, a novel stakeholder assignment approach is introduced (see Section 4). Acting as basic configuration service, it lets voters evaluate stakeholders based on different criteria/dimensions and then aggregates their votes to derive possible configurations which are then recommended as stakeholder assignment decision.

[1] www.bugzilla.org

Decision Biases in Group Decision Making

Groups have the potential to outperform individuals in terms of decision quality [OS2016,SH2014]. A collective memory of a group in many cases entails more decision-relevant knowledge compared to the memory of each individual group member. The same holds for solution knowledge: different group members are able to recall approaches to solve problems or take decisions in the past. However, groups often fail to achieve this goal [FOR2006]. One reason/explanation for/of this phenomenon are *decision biases* which are defined as a tendency to think and act in specific ways which results in deviations from rational and potentially optimal decisions [AR1998, KKM199629, OS2016]. Decision biases occur in single person decision making as well as in group decision settings. The inclusion of theories of human decision making into different types of recommender applications is still a quite young research field with a couple of open research issues [FE2014].

Group Recommender Systems

Single user recommender systems focus on the recommendation of items to individuals [JZF+2010]. In contrast, group recommender systems determine item recommendations that fit the preferences of group members [FBS+2018, JWF+2015]. Table 1 provides an overview of example group recommendation environments with documented user interfaces and evaluation studies.

System	Domain	Reference
Travel Decision Forum	Tourist Destination	[JA2004]
PolyLens	Movies	[CCK+2001]
IntelliReq	Software Requirements	[NFS+2014]
CATS	Ski holiday packages	[MSC+2006]
Choicla	Domain-independent, e.g., personnel decisions	[SFL+2015]

Table 1: Example group recommender systems.

For an in-depth discussion of group recommender applications we refer to [FBS+2018].

Jameson [JA2004] introduces a prototype application that supports groups of users to elicit and aggregate user preferences with regard to holiday destinations. Masthoff [MA2004] introduces concepts for television item sequencing for groups of users on the basis of different models from social choice theory (see also [FBS+2018]). O'Connor et al. [CCK+2001] present a collaborative filtering based approach to movie recommendation that determines recommendations for groups of users. Ninaus et al. [NFS+2014] demonstrate the application of group recommendation technologies in software requirements engineering scenarios where stakeholders are in charge of cooperatively developing, evaluating, and prioritizing requirements. Kudenko et al. [KBD2003] propose a system which helps a group of users while

purchasing a product from an electronic catalog and mediates a group discussion with the goal to achieve consensus. McCarthy et al. [MSC+2006] introduce a critiquing-based recommendation approach that supports groups of users in a skiing holiday package selection process. Finally, Choicla [SFL+2015] is a group decision support environment which includes group recommendation technologies in a domain-independent fashion – related example application domains are personnel decisions and restaurant selection. For a detailed overview of existing group recommender applications we refer to Jameson and Smyth [JS2007] and Boratto et al. [FBS+2018].

Also in the context of recommender systems, decision biases frequently occur and can lead to low-quality decisions [FE2014]. Masthoff and Gatt [MG2006] report possible approaches for the prediction of group member satisfaction with recommendations – in this context, conformity and emotional contagion are stated as major influence factors. Felfernig et al. [FZN+2012] and Stettinger et al. [SFL+2015] discuss the impact of conformity on group decision making and report an increasing diversity of the preferences of group members the later individual preferences are disclosed. Chen and Pu [CP2012] show how emotional feedback of group members can be integrated in a music recommendation system. An outcome of their study is that emotional feedback can help to enhance the mutual awareness regarding the preferences of other group members.

Knowledge exchange between group members can have a major impact on decision quality [MS2010]. The probability of discovering the relevant knowledge (knowledge of one group member not known to the other group members) to take a high-quality (if optimality criteria exist, also an optimal) decision increases with an increased frequency of information exchange between group members [WHB2004]. One possible reason for increased knowledge exchange between group members is group diversity (in terms of dimensions such as demographic and educational background). The higher the degree of diversity, the higher the probability of higher quality decision outcomes (measured, e.g., in terms of the degree of susceptibility to the framing effect [YA2011]). Schulz-Hardt et al. [SBM+2006] report the role of dissent in group decision making scenarios: the higher the dissent in initial phases of a group decision process, the higher the probability that the group manages to share the decision-relevant information. An initial study on selection criteria for preference aggregation in group decision making is reported in Felfernig et al. [FAT+2017] – a major outcome is an observed shift from consensus-based strategies such as average voting to borderline strategies such as least misery in the case of high- involvement items such as apartments and financial services. Software release plans can be considered as high-involvement items.

Explanations of Group Decisions

Using explanations in recommender systems can have various reasons: users have to be supported in taking (high-quality) decisions more quickly, developers of recommender systems want to convince users to purchase specific items, users should better understand how the recommender system works, why a specific item has been recommended, and also develop a more in-depth understanding of the item domain. Consequently, explanations are always given in order to achieve specific goals such as increasing the transparency of a recommendation or increasing a user's trust in the recommender system.

Explanations have been recognized as an important means to help users to evaluate recommendations, to take better decisions, but also to deliver persuasive messages to the user [AR1998,FBS+2018]. Empirical studies show that users appreciate explanations of recommendations [FBS+2018]. Explanations can be regarded as a means to make something clear by giving a detailed description [TM2012]. In the recommender systems context, Friedrich and Zanker [FZ2011] define explanations as information about recommendations and as means to support objectives defined by the designer of a recommender system. Explanations can be seen from two basic viewpoints [TOF2016, BM2005]: (1) the user's (group members) and (2) the recommender provider's point of view. Users of recommender systems are in the need of additional information to be able to develop a better understanding of the recommended items. Developers of recommender systems want to provide additional information to users for various reasons, for example, to convince the user to purchase an item, to increase a user's item domain knowledge (educational aspect), and to increase a user's trust in and overall satisfaction with the recommender system. Another objective is to make users more tolerant with regard to recommendations provided by the system; especially for new users/items, situations can occur where a recommendation is perceived as inappropriate. Solely providing the core functionality of recommender systems, i.e., showing a list of relevant items to users, could evoke the impression of interacting with a black box with no transparency and no additional user-relevant information [HKR2000]. Consequently, explanations are an important means to provide information related to recommendations, the recommendation process, and further objectives defined by the designer of a recommender system [CW2017,FZ2011,QSR+2017, VPB+2013]. Visualizations of explanations can further improve the perceived quality of a recommender system [GKV2009,TOF2016,VPB+2013,DR2009,DR2005].

2.2 OpenReq Requirements in WP4

Requirements regarding the OpenReq group decision support (WP4 component) have been collected from the original project requirements, the interviews held with OpenReq trial partners, and the results of a user study we conducted with different small and medium-sized companies and research organizations (see Section 4). Overall, existing requirements engineering tools provide limited support for decision making processes in groups. All collected requirements regarding the OpenReq group decision component are defined and stored in OpenReq Live. Thus, *we already use OpenReq technologies for supporting Requirements Engineering processes within OpenReq* with the nice side effect of being able to extensively test "OpenReq in OpenReq".

The OpenReq WP4 component (also denoted as "OpenReq Live" is the central component for supporting group decision processes. Major innovations that come along with the OpenReq group decision technologies are among others user interfaces that foster engagement in requirements engineering processes, approaches to the recommendation of group decisions, efficient methods for planning and also re-planning, semi-automated stakeholder selection, and decision technologies that go beyond preference-based approaches taking into account argumentation-based group decision making.

Figure 5 depicts an example screenshot of the OpenReq Live system in its current version. Requirements can be *assigned (*via *drag & drop)* to specific releases (the created releases are sorted by their release dates). Requirements not yet assigned to a specific release are available in a dedicated section "unassigned requirements" on top of the page. Figure 5 illustrates a planned Release #4. In the current version of the system, requirements can be evaluated on the basis of utility dimensions where a collected set of group preferences is aggregated to a "global" group preference which reflects a prioritization. On the basis of a pro/con analysis, requirements can be discussed in a forum. The underlying idea is that such an argumentation-based approach to group decision making helps to better detect hidden profiles representing information that needs to be available to take an optimal decision.

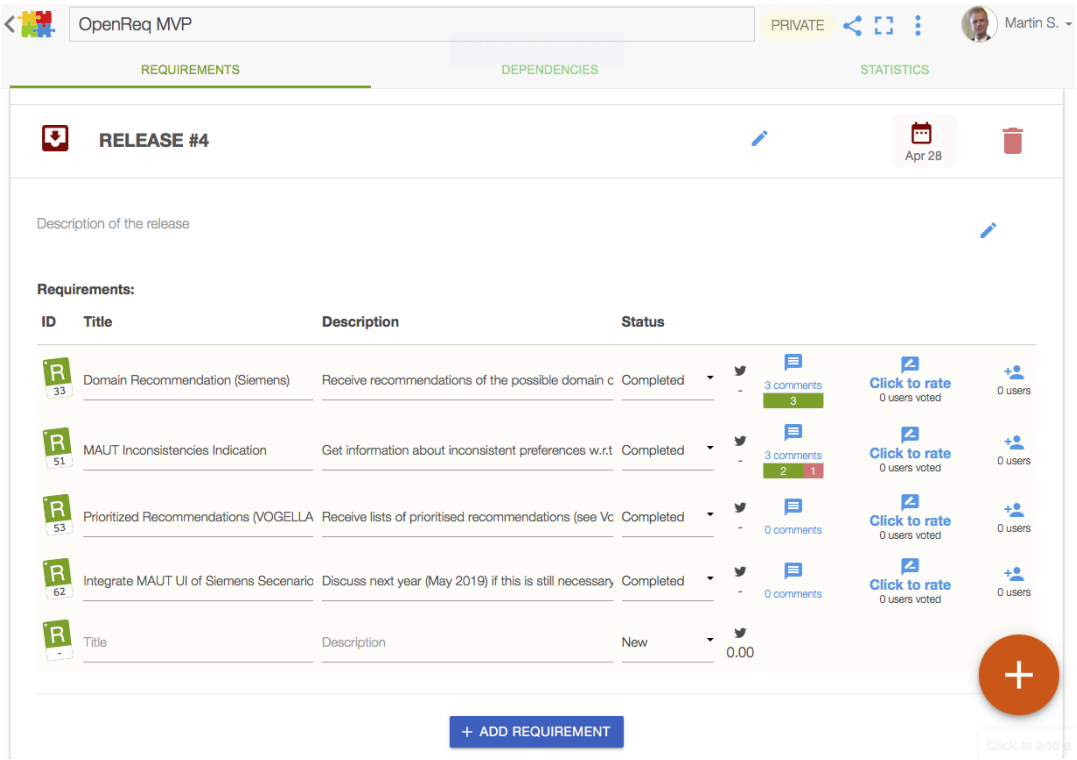


Figure 5: Fragment of the Release Plan for OpenReq Live.

Figure 5 shows all requirements assigned to release "Release #4" which is due to April, 28th 2018. On the basis of a Twitter[2] channel analysis, requirements receive a popularity value which is in indicator to which extent the given requirement is relevant to a specific Twitter community. This way, new requirements are not only evaluated by stakeholders but potentially also by a user community which can help to increase the adequateness of prioritizations. As already mentioned, stakeholders are enabled to discuss a requirement and comment on a requirement in terms of pro and con arguments. Related sentiment values (pro arguments are

[2] www.twitter.com

associated with a positive sentiment) support stakeholders when rating requirements. Stakeholders can also be assigned to requirements (recommendations for relevant stakeholders can be provided).

The following list includes important requirements related to group decision scenarios in the OpenReq project.

- *Group-based Multi-attribute Utility Theory* (MAUT - see [DY2005]): requirements should be evaluated with regard to interest dimensions which have to be predefined for each project, i.e., interest (evaluation) dimensions should be parametrizable. Standard (predefined) interest dimensions are *profit*, *effort*, and *risk*. Stakeholders are enabled to evaluate requirements on the basis of the defined dimensions. These evaluations can then be aggregated on the basis of a utility-based approach. The resulting ranking represents the prioritization preferred by the group.
- *Argumentation-based Messaging*: in order to better estimate the relevance of a requirement, concepts of argumentation-based ranking (on the basis of a structured chat interface) have to be integrated into group-based utility analysis. This way, the detection of hidden profiles can be more easily achieved simply due to an increase of the amount of available (i.e., exchanged) decision-relevant information.
- "*Domain*" *& Stakeholder Recommendation*: the underlying idea is that often it remains unclear which stakeholders should be in charge of working on or evaluating a requirement. Recommendations are needed in order to recommend stakeholders or sometimes recommend "domains" which reflect areas of expertise covered by specific persons. In the context of the OpenReq Siemens scenario, also recommendations of so-called "technical solution approaches" are required.
- *Dealing with Inconsistencies*: inconsistencies can be induced by contradicting stakeholder preferences regarding the assignment of requirements to releases but as well to inconsistent evaluations of requirements and inconsistencies between requirement dependencies and assignments of requirements to releases. Such inconsistencies have to be detected and resolved in a personalized fashion, i.e., should be reflected by adaptations that are accepted by the group of stakeholders as a whole. The OpenReq approach to tackle this challenge will be to exploit preference aggregation functions from group recommender systems to calculate repairs for inconsistencies that are relevant and acceptable for all group members.
- *Recommendations of Prioritizations*: such recommendations will be determined on the basis of utility-based approaches. An initial evaluation will take place on the basis of the Eclipse community where community members will give feedback on the quality of utility-based prioritizations. In the following, utility-based approaches will also be compared to alternative machine learning approaches.
- *Statistics and Beyond*: the overall idea is to have a visualization of the current status of the requirements engineering process. For stakeholders it is important to understand what are the open issues, which inconsistency issues have to be resolved, what are potentially hidden dependencies, which requirements have quality issues, and how well

they perform compared to the other project members and stakeholders in other projects. Thus, we do not only envision a descriptive component but also include elements that help to increase the engagement of stakeholders in requirements engineering processes.

- *Liquid Democracy based Requirements Prioritization*: the underlying idea of liquid democracy is make voting processes more flexible especially due to the fact that in some contexts non-expert stakeholders are in charge of performing, for example, requirements evaluations but prefer to transfer their "token" to other stakeholders they regard as more experienced and thus more appropriate to perform the evaluation.
- *Community Preference Extraction from Social Media*: on the basis of the integration with social media channels such as Twitter, additional preference feedback regarding the relevance of requirements can be collected. This feedback can be integrated into the group based utility evaluation of requirements.
- *Negotiation & Explanation Mechanisms*: OpenReq will provide mechanisms that help to explain decisions (e.g., prioritizations) to groups. Negotiation mechanisms will be included that help to resolve inconsistencies. In this context, repair proposals will be formulated in such a way that they are acceptable for stakeholders responsible for an inconsistency. Stakeholders should also be enabled to negotiate requirements. The impact of these concepts will be evaluated within the scope of empirical studies.

3. Overview of WP4 Services

An overview of the OpenReq architecture has already been given in Figure 4. On top, different applications (user interfaces developed by the trial partners as well as OpenReq Live) are shown with their interactions to different services mainly developed by the OpenReq university partners. Figure 6 represents the basic micro-service architecture of workpackage 4. Requirements engineering tasks are supported by the OpenReq Live UI which uses microservices of the group decision component.

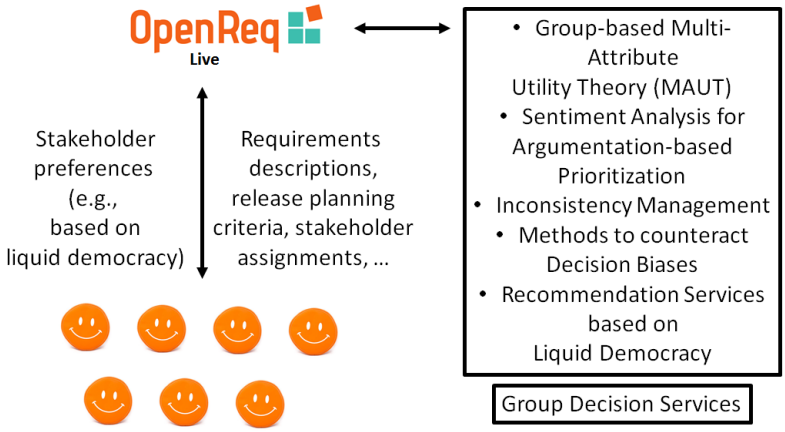


Figure 6: Overview of workpackage 4 services.

4. Group Decision Scenarios

4.1 Industrial Practice in Group Decision Making in Requirements Engineering

In this section, we present the results of an empirical study conducted with requirements engineering practitioners. The results of this study are summarized in terms of practices and open issues in industrial RE decision making. We provide an overview of techniques that can help to increase the quality of group decision making in requirements engineering. We conducted a user study with participants from software companies and research institutes in Europe and the United States (see [FSA+TR2018]). The participants had to answer specific questions about decision making processes via an online survey tool. A general insight of this study is that industrial requirements engineering tools support decisions such as prioritization, release planning, and quality decisions. However, this support is limited to entry fields associated with individual requirements, i.e., no in-depth group decision support is provided in most of the existing systems. Furthermore, market and user community analysis features are not integrated in existing tools. For example, existing industrial RE tools do not provide functionalities that support the analysis of content provided by platforms such as Twitter to figure out the relevance of requirements.

Overview of the User Study

Within the scope of our user study, we received feedback from N=186 participants working for small and medium-sized companies and research organizations in Europe and the United States. An overview of the business areas of the study participants is given in Figure 7. An overview of RE roles of study participants is given in Figure 8. Figure 9 provides an overview of the number of employees working in these companies. The majority of the participating organizations has less than 500 employees, i.e., the majority of participants are working in small or medium-sized organizations.

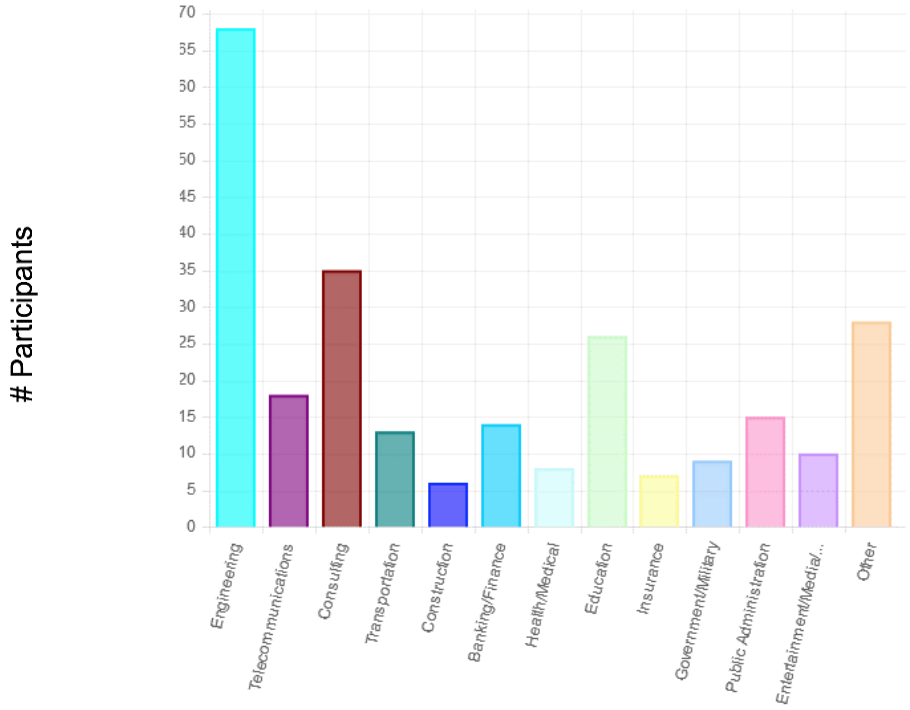


Figure 7: Business areas of study participants.

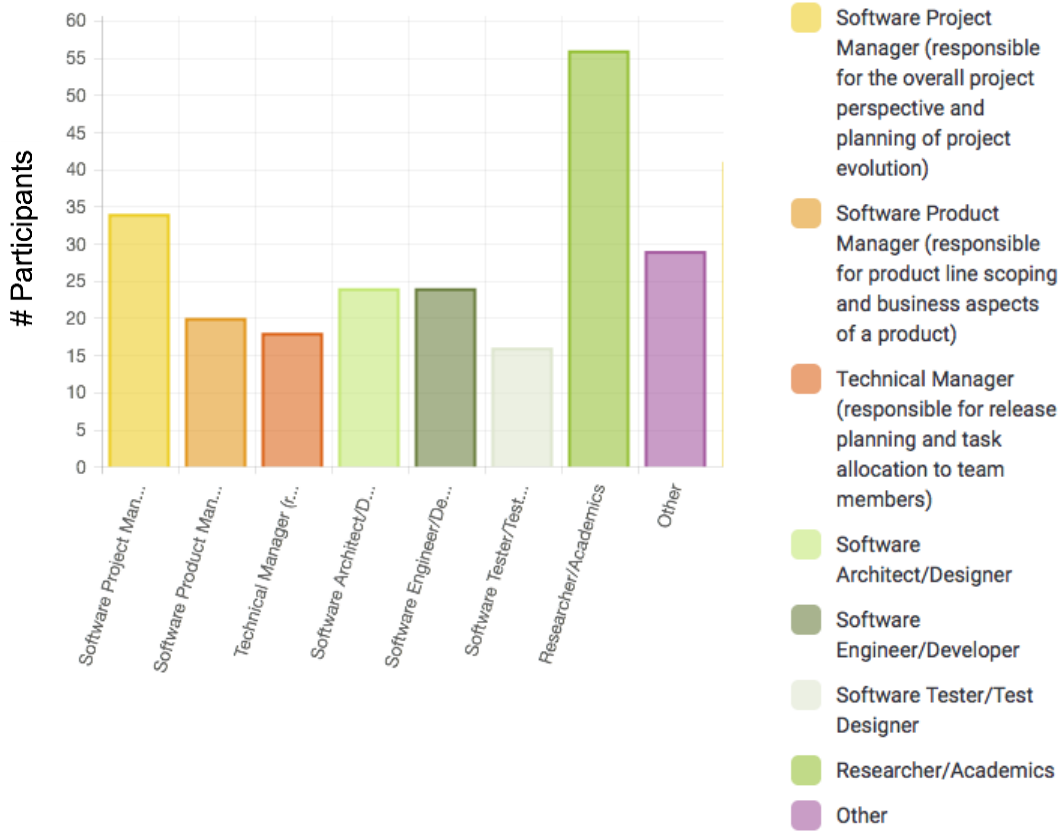


Figure 8: Company roles of study participants (roles are partially overlapping).

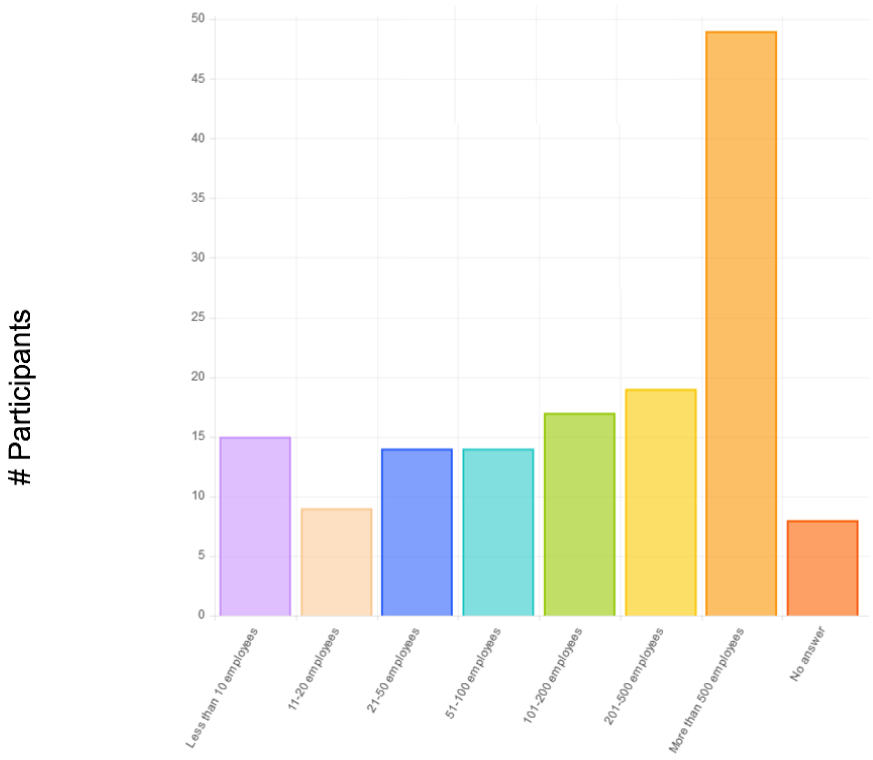


Figure 9: Number of employees in participating organizations.

An overview of the used requirements engineering tools is provided in Figure 10. OFFICE TOOLS[3] and BUGZILLA[4] are most frequently applied by the study participants. Further tools applied by the participants are IBM Doors[5] and Wiki-based environments. The category "Other" in Figure 5 entails tools such as JIRA[6] and POLARION[7].

[3] E.g., www.microsoft.com

[4] www.bugzilla.org

[5] www.ibm.com

[6] www.atlassian.com

[7] www.polarion.com

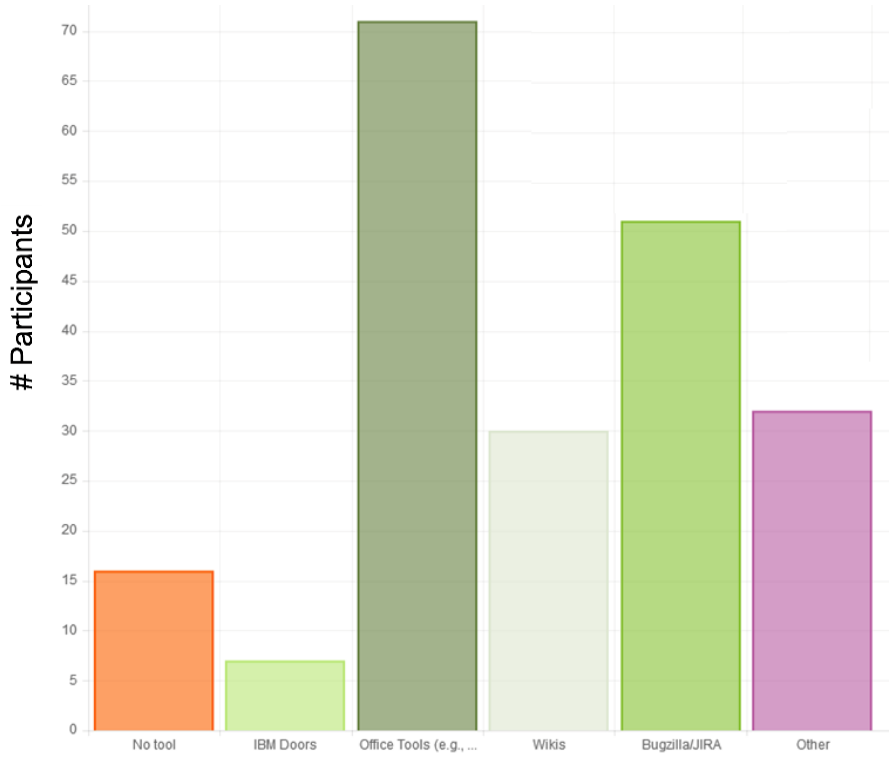


Figure 10: Used requirement engineering tools.

An overview of decision types supported by the used tools is given in Figure 11. The most frequently supported decision types are:

- prioritization,
- release planning,
- quality decisions,
- stakeholder decisions, and
- go/no-go decisions.

Note that these decisions are in most of the cases supported by entry fields that allow the specification of a requirement-specific priority or the assignment of a requirement to a specific release. There is no further support for group decision processes in terms of pro-actively supporting groups of stakeholders in their decision making processes.

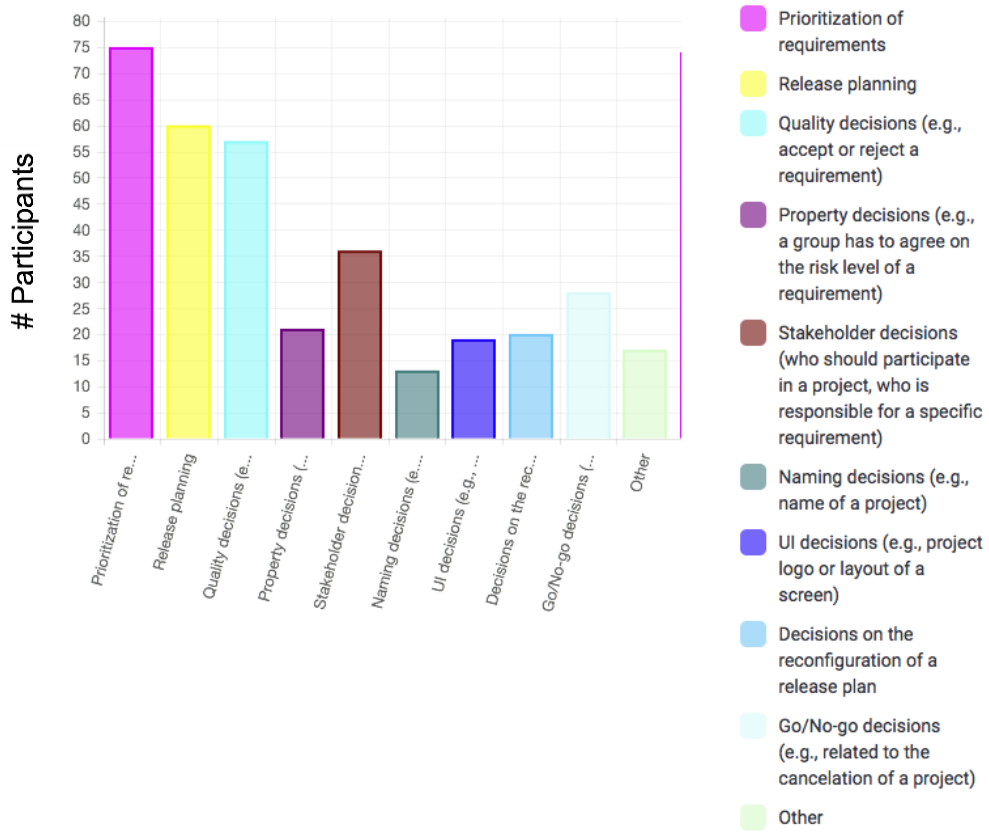


Figure 11: Overview of basic decision types supported by the used RE tools.
In most of the cases, these decisions are supported in terms of providing entry fields that allow, for example, the assignment of priorities and releases.

User Study Results

Most of the study participants reported not to use a requirements engineering tool that supports group decision making scenarios (Figure 12). In many cases, DOODLE[8] is used as a basis for decision making. Other tools that have been mentioned are: ADHOCRACY[9], STRAWPOLL[10], CHOICLA[11], SKYPE[12], and JIRA (mentioned in the Other category).

[8] www.doodle.com

[9] www.liqd.net

[10] www.strawpoll.de

[11] www.choiclaweb.com

[12] www.skype.com

Consequently, in many cases companies base their decisions on informal discussions and meetings, i.e., the underlying process is often unstructured and the related documentation is limited. Even if tools such as DOODLE or JIRA are used, no support for content-based (argumentation-based) decision making is supported by these tools. This can result in *suboptimal decisions due to a lack of a structured exchange of decision-relevant information among stakeholders* [MS2010].

Study participants reported to take part of the group decisions (see Figure 13). The most frequently reported group decision types are *prioritization*, *release planning*, *quality decisions* (e.g., which requirements should be taken into account), and *stakeholder decisions*.

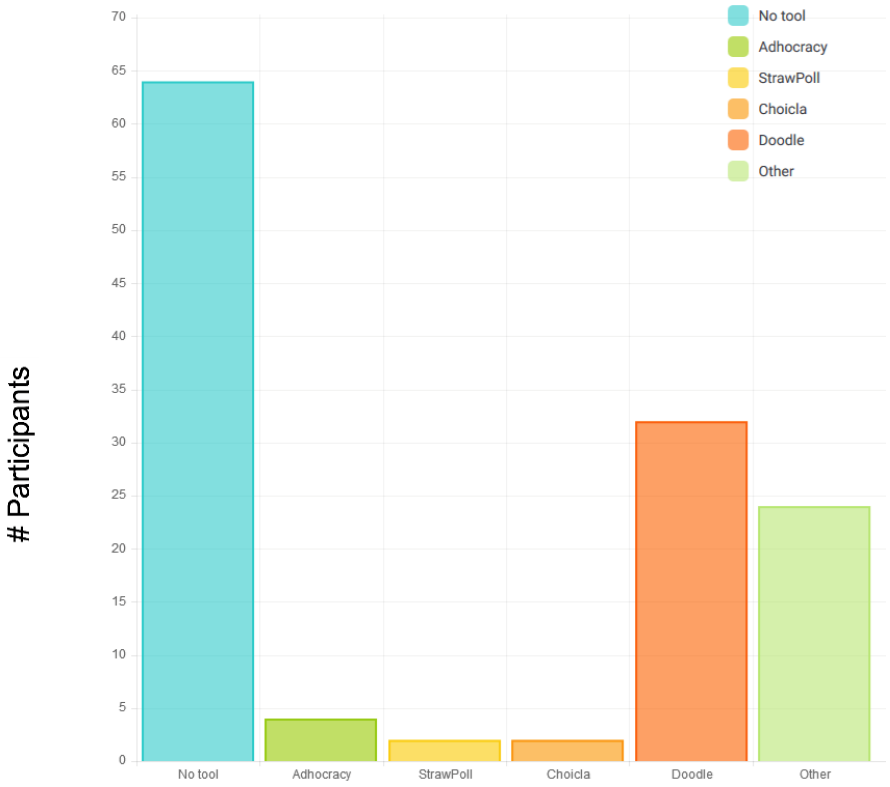


Figure 12: Used tools for supporting group decision making in RE.

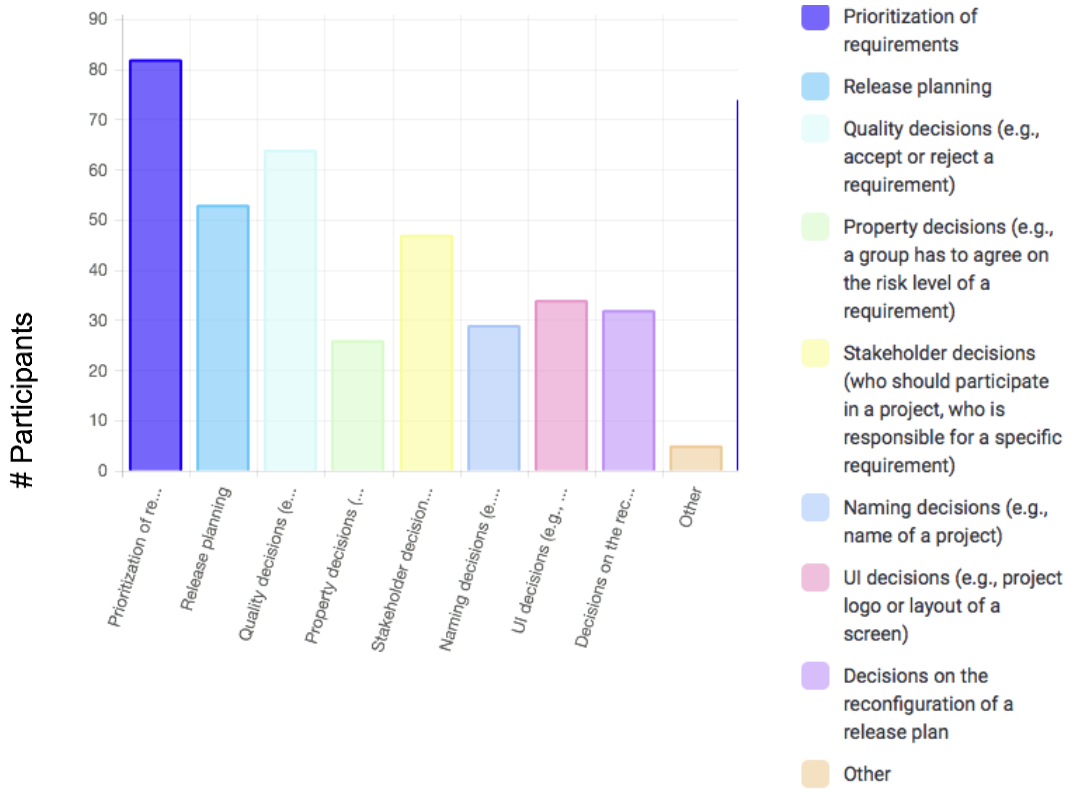


Figure 13: Overview of decisions taken by user groups in the context of RE processes.

An overview of group decisions that are currently *not tool-supported but should be* (the study participants would like to have this support) is given in Figure 14. Interestingly, most of the mentioned group decision tasks are not tool-supported but are completed manually. On the other hand, study participants emphasize that a corresponding tool support is needed.

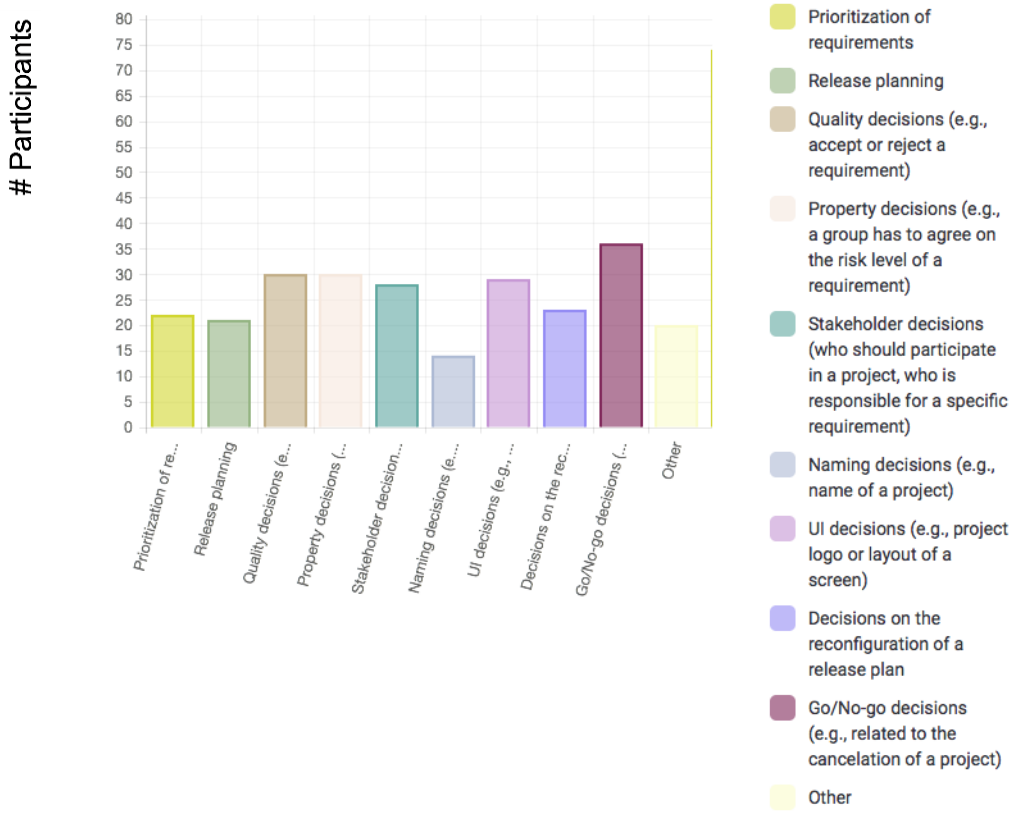


Figure 14: Overview of non tool-supported group decisions (but should be).

An overview of preferred consensus approaches is given in Figure 15.

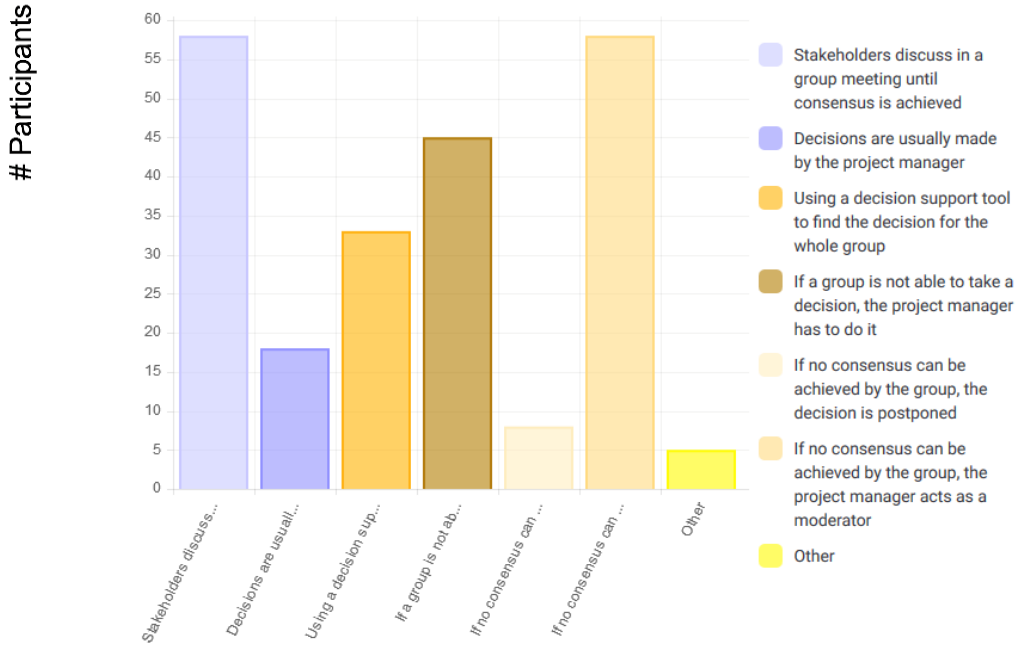


Figure 15: Overview of preferred approaches to achieve consensus in group decisions.

The most frequently used approaches are discussion sessions among stakeholders with the overall goal to repeatedly discuss an issue until consensus is achieved. Other used approaches

(and combinations thereof) are the moderation by the project manager, decision making by the project manager (if the group is not able to find a consensus), and taking decisions on the basis of a decision support tool.

Important aspects that influence the quality of a decision outcome are summarized in Figure 16. The most relevant factors that deteriorate the overall quality of a decision are limited stakeholder knowledge (e.g., stakeholders do not read decision-relevant documents), absence of decision-relevant stakeholders (e.g., these stakeholders do not attend decision- relevant meetings), *group think* (i.e., group members who do not say what they think to avoid raising issues and negative consequences), and stakeholder preference orientation (e.g., important topics are not discussed due to time restrictions and a focus on discussing stakeholder preferences instead of exchanging decision-relevant information).

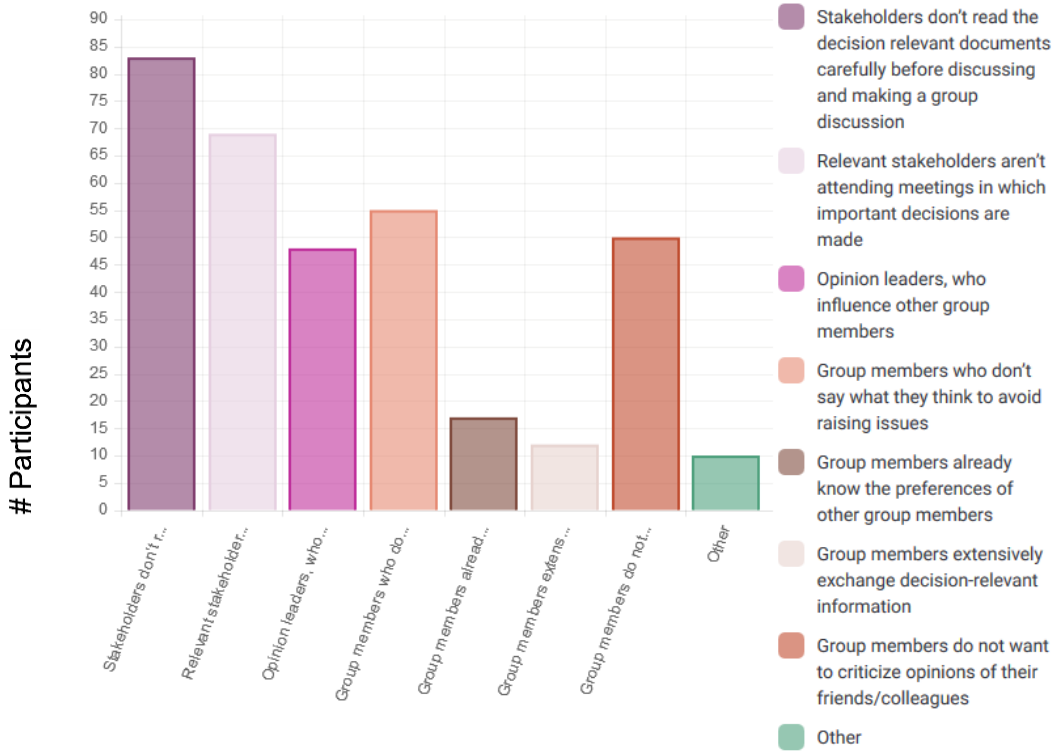


Figure 16: Factors that can negatively influence the outcome of a decision process.

Participants of the survey reported that there are different measures to improve the overall quality of group decisions (see Figure 17).

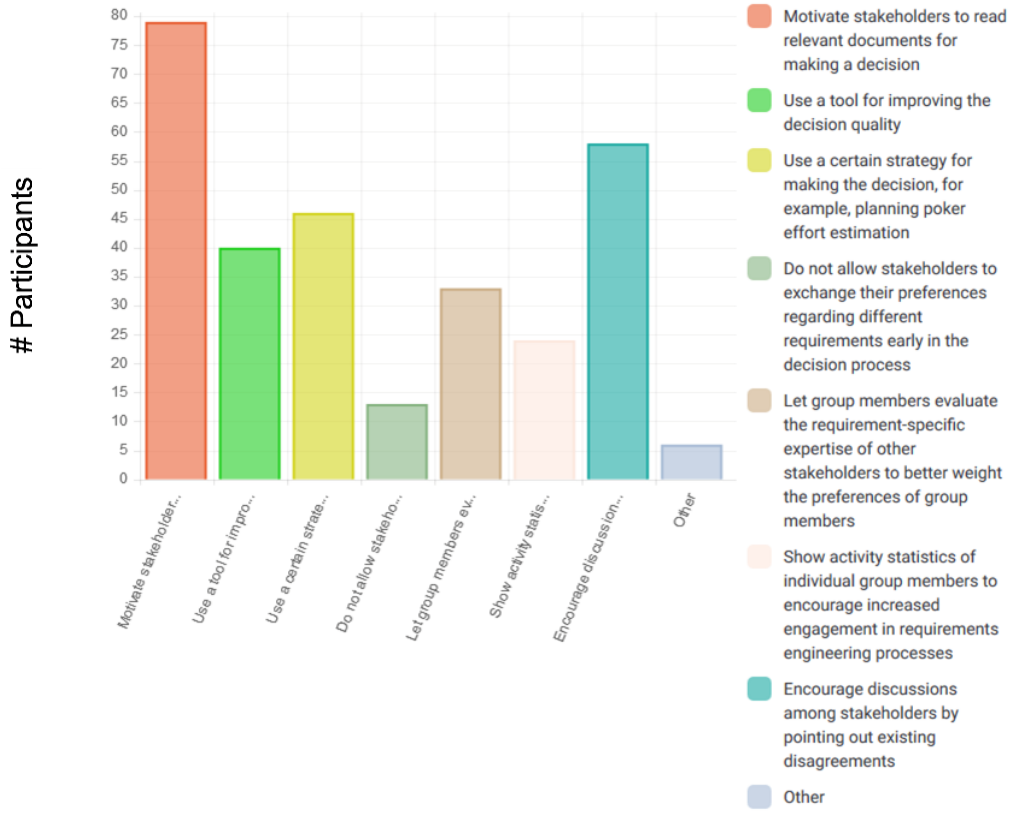


Figure 17: Measures to improve the overall quality of group decisions.

First, stakeholders have to be motivated to read decision-relevant documents. This can be achieved, for example, by gamification approaches such as performance measures (the more interactions stakeholders have with these documents, the higher the corresponding performance index). Discussions among stakeholders have to be fostered. One way to achieve this is to provide RE user interfaces that focus on decision-relevant information exchange and discussions and keep discussions on requirements prioritization and preferences in the ”background”. In this context, explanations, argumentations, and documentation of decision processes and decision outcomes can be regarded as major factors to improve decision quality and trust in decision outcomes [WHB2004].

Different types of conflicts that can occur among stakeholders are shown in Figure 18. Corresponding conflict resolution strategies are summarized in Figure 19. The state of practice focuses on manual conflict resolution strategies, i.e., automated approaches that support stakeholders in conflict management are typically not available. An example of (semi-)automated conflict resolution in release planning is provided in [FBS+2018] where stakeholder preferences are analyzed on the basis of the concepts of model-based diagnosis [FSZ2012]. In this context, important functionalities that have to be integrated in RE tools are the automated detection of conflicts (e.g., conflicting dependencies between requirements in a release plan) and the determination of corresponding resolution alternatives.

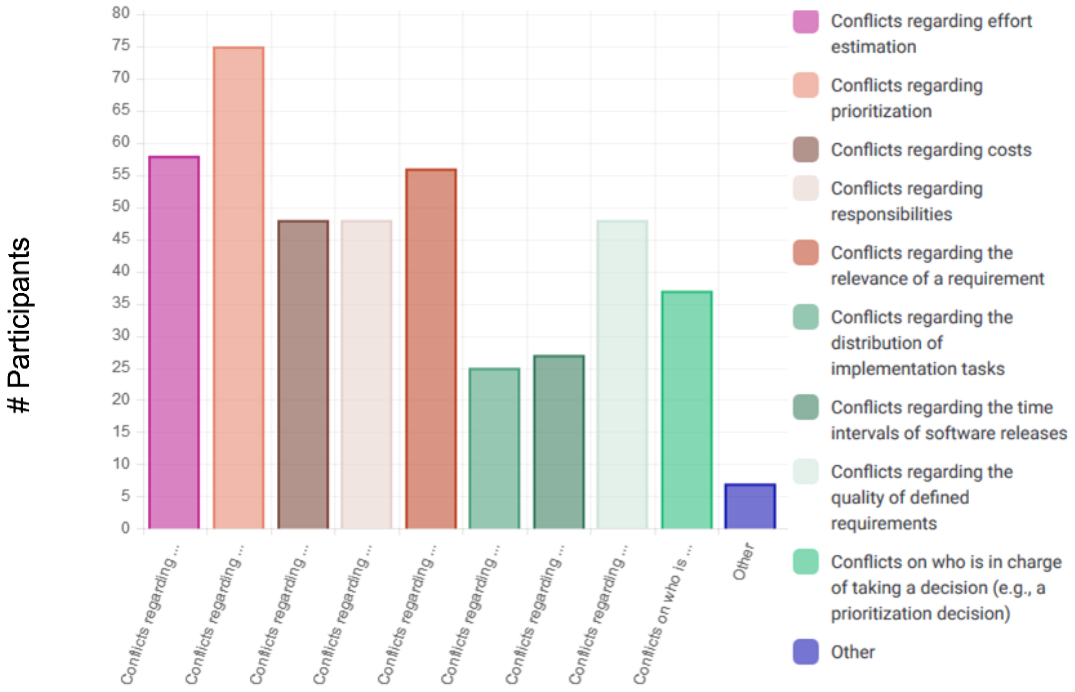


Figure 18: Different types of conflicts among stakeholders.

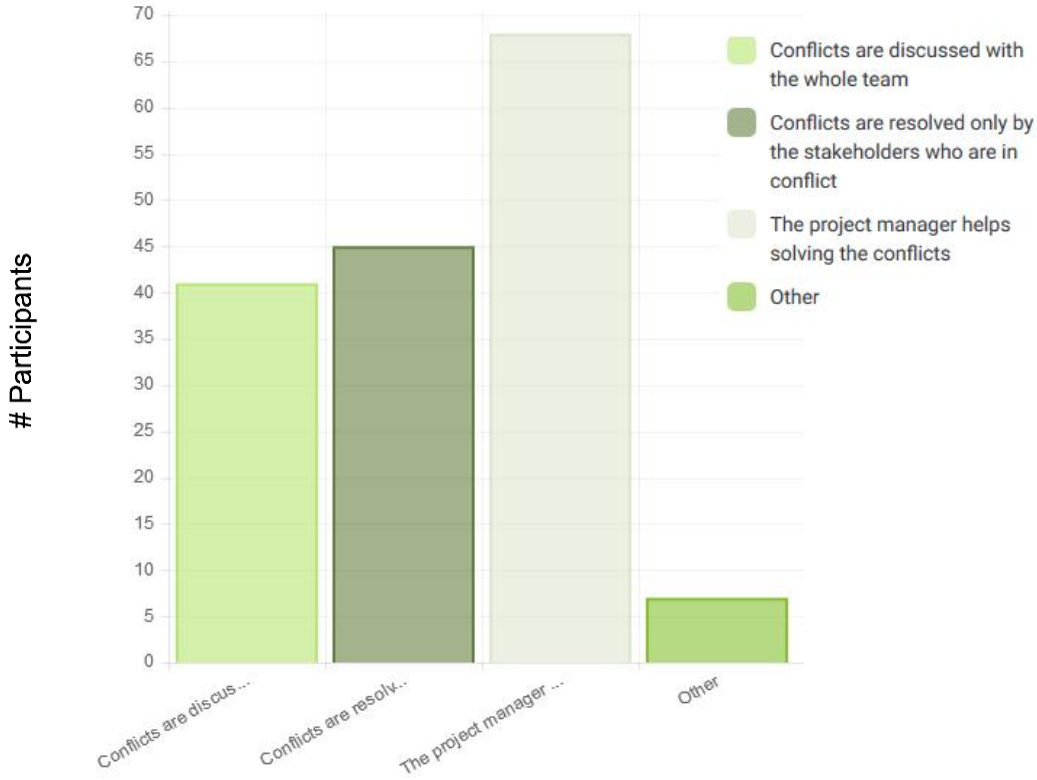


Figure 19: How to resolve conflicts among stakeholders.

Different approaches to identify project-relevant requirements are summarized in Figure 20. An important open issue in this context is that there is a low share of companies that integrate community knowledge in one way or another in their requirements engineering scenarios. In the majority of the cases, no community integration is provided (see Figure 21). An example of the mentioned community integration is the analysis of specific Twitter channels in order to figure out relevant issues discussed in online communities [WM2017].

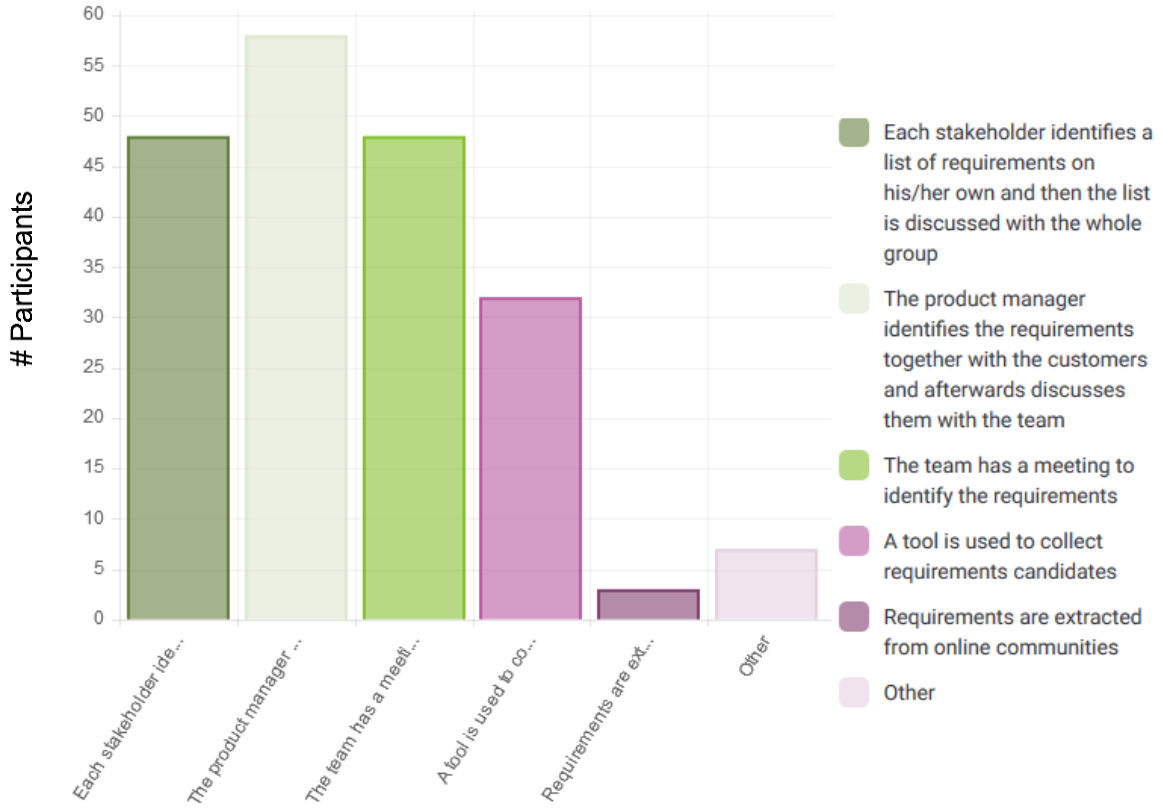


Figure 20: Ways to identify project-relevant requirements.

Different approaches to take into account the importance of individual stakeholders are summarized in Figure 22. These approaches follow a manual process in which the expertise of individual stakeholders has to be evaluated. In many cases it remains unclear how stakeholder expertise has been taken into account in a requirements-related decision. More fine-grained techniques that allow to estimate stakeholder-specific expertise on the level of individual requirements are not supported by existing RE environments. Due to this lack of flexibility, there is currently no room for flexible voting mechanisms such as liquid democracy that allow a clear expertise-focused view on the importance of requirements [JM2014].

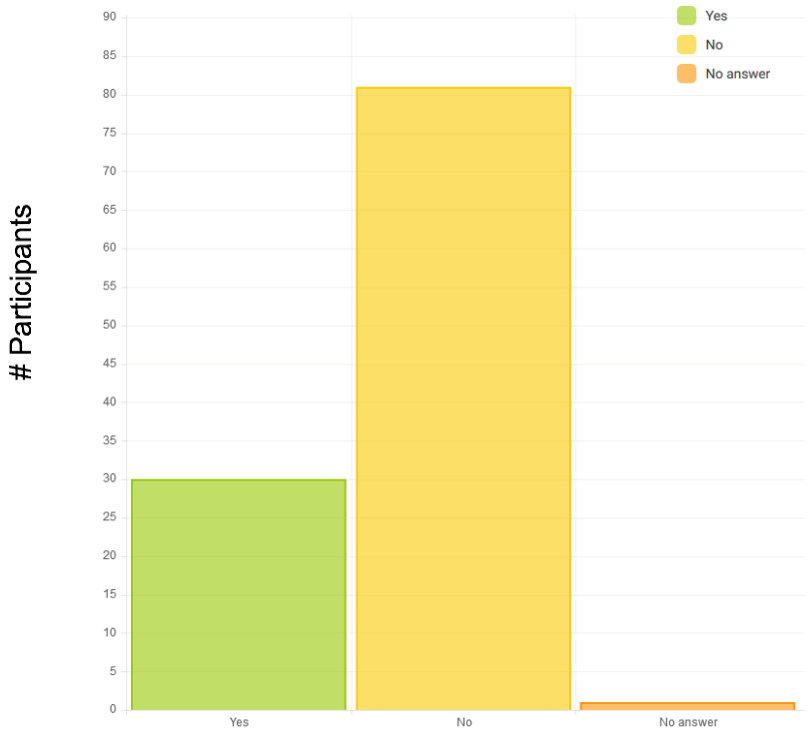


Figure 21: Share of companies integrating communities in their decision making processes.

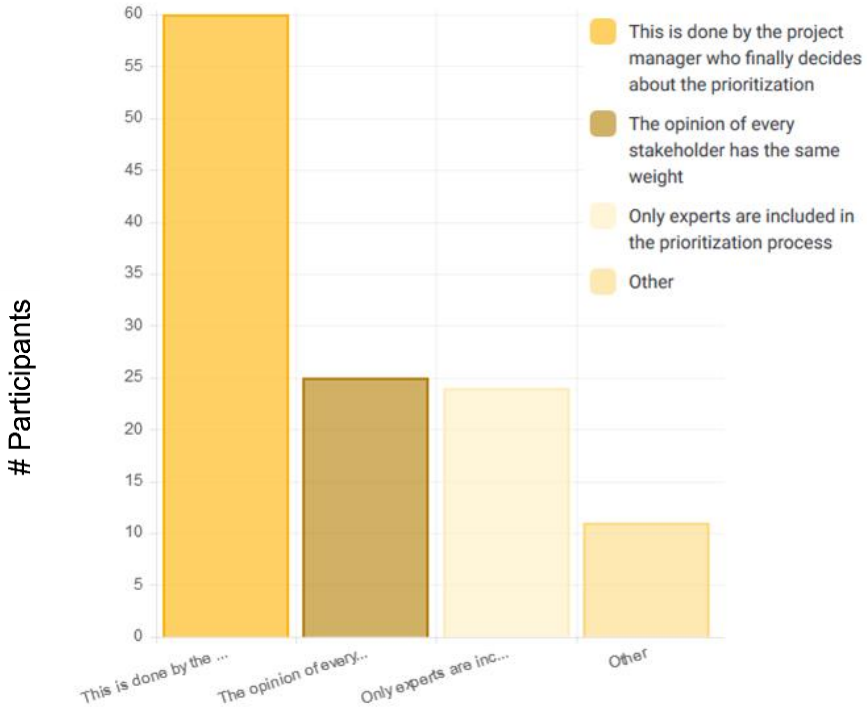


Figure 22: Ways to identify the decision-specific importance of stakeholders.

Different criteria (dimensions) that are used to evaluate requirements are summarized in Figure 23. The most frequently used ones are consistency, completeness, time efforts, understandability, and risk. Other mentioned criteria are benefit, conformity of existing standards, and feasibility (e.g., in terms of the availability of resources).

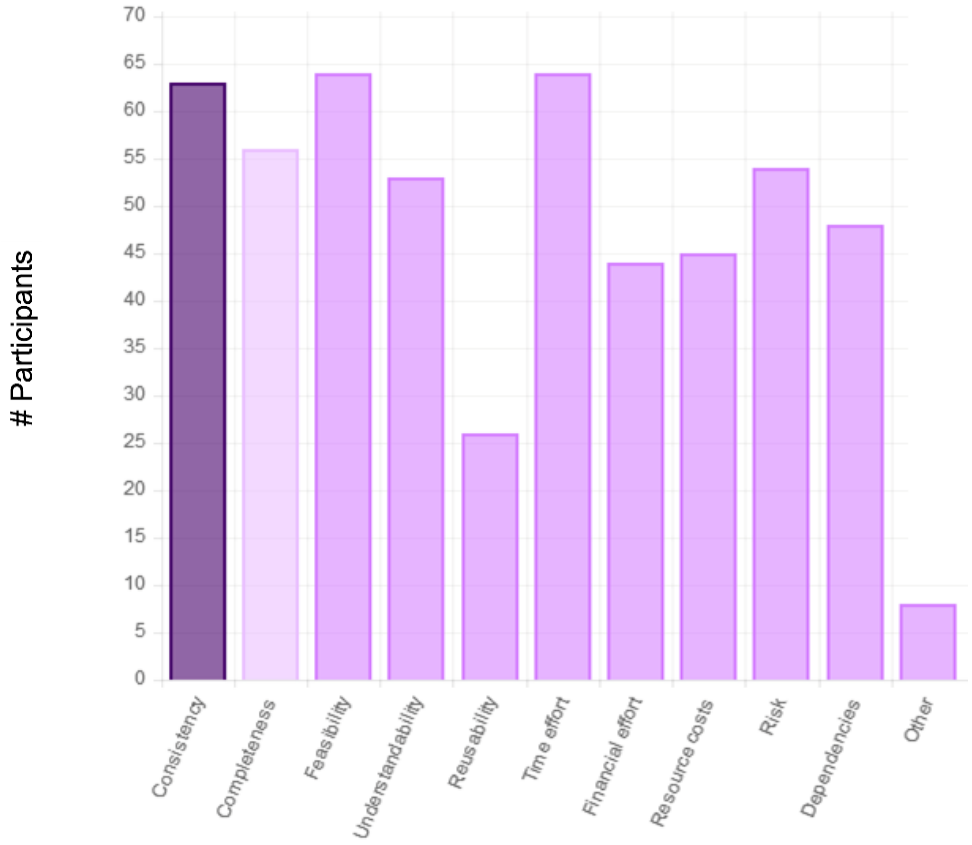


Figure 23: Different criteria (dimensions) often used for evaluating requirements.

Relevant features for supporting group decision making in RE

Summarizing the results of our study, the following features will help to tackle open issues discussed in the previous sections.

Preference-based Decisions. Such decisions are related to specific properties (dimensions) of requirements. For example, a group of stakeholders has to estimate the effort related to a requirement. Such estimation can lead to inconsistencies which have to be resolved. In this context, techniques are required that automatically propose reasonable adaptations of existing evaluations such that consensus can be achieved in the group. Such adaptations can, for example, be proposed on the basis of different types of preference aggregation functions [FBS+2018]. An important issue in this context is to figure out which aggregation functions are best in terms of achieving consensus among group members.

Prioritization Decisions. Prioritization can be considered as a specific type of preference decision [FMM+2010, FSA+2018]. An important issue in this context is to take into account the preferences of whole user communities, for example, by analyzing related Twitter channels

or app store reviews [WM2017]. The quality of prioritization decisions can be further improved by personalizing the underlying decision process. For example, *liquid democracy* concepts [JM2014] allow to transfer votes to experts who have the knowledge to take a requirement-specific decision.

Argumentation-based prioritization. A major issue in many of the existing requirements engineering environments is the lack of a content-oriented approach to prioritize requirements (i.e., no discussions on a content level are supported). Most of the used prioritization approaches are utility-based [FSA+2018] where the focus is to evaluate requirements with regard to a given set of interest dimensions. A disadvantage of a purely utility-based approach is the missing mechanism to support knowledge exchange between stakeholders. In this context, it is important to integrate mechanisms that allow to take into account argumentations for and against specific requirements.

Stakeholder Selection. Stakeholder selection can be regarded as a kind of classification decision where the task is to figure out which stakeholder should be part of a software project. A variant thereof is the assignment of stakeholders to specific requirements, i.e., to define which stakeholder is responsible for the evaluation of which requirement. Especially in large software projects with a large number of requirements, the manual assignment of stakeholders to requirements is extremely tedious. Consequently, mechanisms are needed that automatically propose the assignment of stakeholders to requirements (stakeholder selection). Such an approach helps to tackle scalability issues in large software projects and can also help to avoid faulty assignments.

Group Decision Making in Release Planning. In this context, inconsistencies between preferences have to be resolved on the basis of model-based conflict detection and diagnosis approaches [FSZ2012], which help to determine conflict resolutions acceptable for a given group of stakeholders. Examples of related inconsistencies are different preferences regarding the release assignment of a requirement and inconsistencies between release assignments and requirements dependencies. An important challenge in this context is to propose preferred conflict resolutions, i.e., conflict resolutions that will be accepted with a high probability. Especially in the context of release planning, mechanisms are needed that help to automatically resolve inconsistencies. Since release planning is a complex task, related reasoning concepts have to be optimized to be able to support response times acceptable for end-users and at the same time to determine solutions that are accepted by all stakeholders [FAT2016]. In this context, concepts have to be developed that allow to integrate theories of computational social choice [FAT2016] into diagnosis algorithms.

Taking into Account Decision Biases. Decision biases are a major reason for suboptimal decisions [5]. An important issue in this context is to develop user interfaces and algorithmic concepts that can help to counteract such biases. An overview of decision biases especially relevant in the context of group decision and group recommendation scenarios is given in [FBS+2018]. An approach to counteract such biases is presented in [AFS+2017]. One key result of the conducted user study was that a higher diversity in a recommended item set can

lead to a higher amount of messages exchanged between stakeholders within the scope of a prioritization process.

Explanations for Groups. A major issue in the context of group decision making is how to best explain a recommendation to a group. Explanations help to develop trust with regard to a proposed decision and also to increase the degree of transparency. For an overview of different goals of explanations in the context of group decision making we refer to [DR2009, FBS+2018]. In the context of requirements engineering scenarios, explanations can help in various ways. First, explanations can be used to explain a proposed conflict resolution in such a way that stakeholders accept proposed changes with regard to their individual prioritizations. Furthermore, explanations can be used to point out the major reasons/factors (dimensions in utility-based prioritization) that led to a specific prioritization. Another example is the explanation of proposed stakeholder assignments. Explanation dimensions in this context are, for example, expertise and available resources.

Summary

In this section, we summarized industrial practices in requirements engineering decision making. As a basis for our analysis, we conducted a user study with N=186 participants from companies in Europe and the United States. A major outcome of this study is that decision support mechanisms are already integrated in existing requirements engineering tools. A missing link is the provision of more group-centered decision techniques that help to avoid decision biases that can occur in groups and foster intended behavior of stakeholders such as increased knowledge exchange and focus on content related discussions instead of preference related discussions. The following sections light up more details on the previous mentioned features that should be included in future requirements engineering platforms.

4.2 Preference-based Decisions

Group-based configuration is a new approach that supports scenarios in which a group of users is in charge of designing a product or service (e.g., a software release plan). In this section, we introduce a definition of a group-based configuration task and a corresponding solution. Furthermore, we show how inconsistent situations in group-based configuration can be resolved to achieve consensus within the group. We introduce these concepts on the basis of a working example from the domain of (group-based) software release planning. In this section, we focus on introducing a formal definition of a group configuration problem and show how inconsistencies in the preferences of group members can be resolved. This model serves as a basis for solving release planning tasks in OpenReq.

Group-based Configuration & Release Planning

In the following, we introduce definitions of a group configuration task and a corresponding solution. These definitions are based on a Constraint Satisfaction Problem (CSP) [TSA1993] which is frequently used for the definition of (single user) configuration tasks. The major characteristic of group-based configuration compared to other types of group decision tasks is that the alternatives are defined in terms of a knowledge base, i.e., the alternatives are not pre-

specified. This requires new approaches to configuration and diagnosis search, and to represent the configuration task in a corresponding user interface.

Definition 1: Group-based Configuration Task. A group-based configuration task can be defined as a CSP (V ,D, C) where V is a set of variables, D represents the corresponding domain definitions, and C = PREF ∪ CKB represents a set of constraints. In this context, PREF = ∪ $PREF_i$ is the union of customer preferences $PREF_i$ and CKB represents a configuration knowledge base. We denote customer requirements as preferences (PREFS) in order to distinguish these from software requirements in the working example.

Definition 2: Group-based Configuration. A group-based configuration (solution) for a group-based configuration task is a complete set of assignments CONF = ∪ $a_i : v_i = v_{ai}$ to the variables $v_i \in V$ such that CONF ∪ PREF ∪ CKB is consistent.

Example 1: Group-based Configuration Task. For demonstration purposes, we introduce a simplified group-based configuration task from the domain of software release planning. The goal of software release planning is to assign to each software requirement a corresponding release. In this example, 9 requirements are represented in terms of variables V = {req_1, req_2, .., req_9} and releases are represented as variable domains. If we assume that three releases have been planned for completing the whole software (i.e., implementing each individual requirement), each variable has a corresponding domain [1 .. 3], e.g., dom(r_1) = [1 .. 3]. For the purpose of this example, we assume the existence of three stakeholders who are in charge of release planning – $PREF_i$ represents the preferences of stakeholder i.

The following is a complete specification of a group-based configuration task. In this task, the individual user requirements $PREF_i$ are consistent, i.e., a corresponding solution (software release plan) can be identified. In the next section we discuss how to deal with inconsistencies. The configuration knowledge base CKB includes additional constraints that describe dependencies between different software requirements req_i, for example, $req_1 < req_5$ denotes the fact that requirement req_1 must be implemented before req_5, i.e., there is a dependency between these requirements. Furthermore, the requirements req_3 and req_4 must not be implemented in the same release (e.g., due to resource constraints).

V = {req_1,..,req_9}

D = {dom(req_1) = [1 .. 3], .., dom(req_9 = [1 .. 3])}

$PREF_1$ = {$pref_{11}$: req_1 = 1, $pref_{12}$: req_2 = 1, $pref_{13}$: req_3 = 1, $pref_{14}$: req_5 = 2, $pref_{15}$: req_8 = 3}

$PREF_2$ = {$pref_{21}$: req_3 = 1, $pref_{22}$: req_4 = 2, $pref_{23}$: req_6 =3, $pref_{24}$:req_7 =3}

$PREF_3$ = {$pref_{31}$: req_5 = 2, $pref_{32}$: req_6 = 3, $pref_{33}$: req_8 =3, $pref_{34}$:req_9 =2}

CKB = {c_1 : $req_1 < req_5$, c_2 : $req_2 < req_8$, c_3 : $req_3 < req_6$, c_4 : $req_3 \neq req_4$}

Example 2: Group-based Configuration. On the basis of the example group-based configuration task, a constraint solver could determine the following solution: CONF = {a_1 : req_1 = 1, a_2 :req_2 =1, a_3 : req_3 =1, a_4 : req_4 =2, a_5 :req_5 =2, a_6 : req_6 = 3, a_7 : req_7 = 3, a_8 : req_8 = 3, a_9 : req_9 = 2}. For each requirement, the constraint solver proposes a corresponding release in the context of which the requirement should be implemented.

Resolving Inconsistencies in Group Preferences

In the example introduced in the previous section, the basic assumption is that the preferences of individual group members are consistent. However, in group-based configuration scenarios it happens quite often that the preferences of individual users differ or even contradict. In the context of release planning scenarios, it is often the case that stakeholders have different preferences regarding the implementation of specific requirements. One requirement could be favored due to the fact that the corresponding functionalities are needed by the stakeholder. Another reason could be that a stakeholder has no preferences or simply does not understand the requirement in detail. Inconsistencies between preferences can be manually resolved by showing inconsistent preferences to stakeholders and let them decide which changes should be performed. In such scenarios, minimal conflict sets are determined [JU2004] and conflict resolution is performed by users in a manual fashion.

Alternatively, conflicts between requirements can be resolved automatically by calculating minimal diagnoses (Definition 4) for minimal conflict sets (Definition 3).

Definition 3: Conflict Set. A conflict set $CS \subseteq REQ_i$ is a minimal set of requirements such that inconsistent (CS). CS is minimal if there does not exist a conflict set CS' with CS' is a conflict set and $CS' \subset CS$.

Minimal conflict sets can be exploited for determining the corresponding diagnoses [RE1987]. Assuming that $PREF_i \cup CKB$ is inconsistent, a minimal diagnosis represents a minimal set of requirements that have to be deleted from $PREF_i$ such that a solution can be found for the remaining constraints (see Definition 5).

Definition 4: Group-based Configuration Diagnosis Task. A group-based configuration diagnosis task is defined by a group-based configuration task (V, D, C = PREF ∪ CKB) where PREF ∪ CKB is inconsistent.

Definition 5: Group-based Configuration Diagnosis. A diagnosis for a given group-based configuration task (V, D, C = PREF ∪ CKB) is a set Δ such that CKB ∪ PREF − Δ is consistent. Δ is minimal if ¬∃ Δ′: Δ′ ⊆ Δ.

Example 3: Group-based Configuration Diagnosis Task. An example group-based configuration task that includes inconsistencies between different user requirements is the following.

$V = \{req_1,..,req_9\}$

$D = \{dom(req_1)=[1..3],..,dom(req_9)=[1..3]\}$

$PREF_1 = \{pref_{11} : req_1 = 2, pref_{12} : req_2 = 1, pref_{13} : req_3 = 1, pref_{14} : req_5 = 2, pref_{15} : req_8 = 3\}$

$PREF_2 = \{pref_{21} : req_3 = 2, pref_{22} : req_4 = 3, pref_{23} : req_6 = 3, pref_{24} : req_7 = 3\}$

$PREF_3 = \{pref_{31} : req_5 = 2, pref_{32} : req_6 = 3, pref_{33} : req_8 = 3, pref_{34} : req_9 = 2\}$

$CKB = \{c_1 : req_2 > req_1, c_2 : req_2 < req_8, c_3 : req_3 < req_6, c_4 : req_3 \neq req_4\}$

In this example, the requirements of the first stakeholder are inconsistent since the combination $req_1 = 2$ and $req_2 = 1$ is inconsistent with the underlying knowledge base ($req_2 > req_1$). Furthermore, there exists an inconsistency between the requirements $req_3 = 1$ (stakeholder 1) and $req_3 = 2$ (stakeholder 2).

The minimal conflict sets that can be derived from our working example are the following: $CS_1 = \{pref_{11}, pref_{12}\}$ and $CS_2 = \{pref_{13}, pref_{21}\}$. The corresponding set of alternative diagnoses (hitting sets) is the following: $\Delta_1 = \{pref_{11}, pref_{13}\}$, $\Delta_2 = \{pref_{11}, pref_{21}\}$, $\Delta_3 = \{pref_{12}, pref_{13}\}$, and $\Delta_4 = \{pref_{12}, pref_{21}\}$. A diagnosis is a minimal set of requirements from ∪ $PREF_i$ such that CKB ∪ PREF − Δ is consistent.

Diagnoses represent a set of consistency-preserving delete operations that can be applied to the set $PREF_i$ in the case that PREF ∪ CKB is inconsistent. In many cases, there exist different diagnoses that can be recommended for preserving the consistency between user requirements and the configuration knowledge base (CKB). A ranking of alternative diagnoses in the context of group configuration scenarios can be achieved, for example, by determining a candidate set of minimal diagnoses that is then ranked on the basis of different types of group decision heuristics [FBS+2018]. An example of the application of such group decision heuristics will be discussed in the following. Table 2 depicts a situation where individual user requirements are inconsistent.

stake-holder	req_1	req_2	req_3	req_4	req_5	req_6	req_7	req_8	req_9
1	$pref_{11}$: req_1=2	$pref_{12}$: req_2=1	$pref_{13}$: req_3=1		$pref_{14}$: req_5=2			$pref_{15}$: req_8=3	
2			$pref_{21}$: req_3=2	$pref_{22}$: req_4=3		$pref_{23}$: req_6=3	$pref_{24}$: req_7=3		
3					$pref_{31}$: req_5=2	$pref_{32}$: req_6=3		$pref_{33}$: req_8=3	$pref_{34}$: req_9=2

Table 2: Tabular representation of constraints in an example group-based configuration task. Conflict set CS_1 = {$pref_{11}$, $pref_{12}$} reflects inconsistent preferences of stakeholder 1 (the preferences are inconsistent with the configuration knowledge base) and conflict set CS_2 = {$pref_{13}$, $pref_{21}$} reflects a conflict between the preferences of stakeholders 1 and 2.

In order to resolve this inconsistency, the alternative diagnoses Δ_1, Δ_2, Δ_3, and Δ_4 can be applied. An open question in this context is which of the alternative diagnoses should be recommended first to the group of users – Table 3 summarizes the impact of the different diagnoses on the current preferences of stakeholders (users). For this purpose, different group decision heuristics can be applied that help to figure out alternatives acceptable for the whole group.

In the following, we exemplify three basic heuristics and show how these can influence the selection of a diagnosis. First, the *least misery* heuristic prefers alternatives (in our case diagnoses) that minimize the misery of individual users (see Formula 1 – $pref_{delta}(s, \Delta)$ denotes the number of preferences that have to be changed by user s in the context of diagnosis Δ). In our scenario, least misery for a whole group would reflect the minimum of the maximum number of preferences part of a diagnosis, i.e., the lower the least misery value the better the corresponding diagnosis. For example, if diagnosis Δ_2 is recommended, user 1 would have to adapt two of his/her requirements and user 2 would have to adapt zero. Diagnosis Δ_2 has a lower misery value since the maximum number of requirements to be adapted is 1. Obviously, user 3 is in the situation of not being affected by any of the diagnosis candidates. Second, the *average* heuristic prefers alternatives with the lowest average deviation from the original preferences (see Formula 2). Finally, the most pleasure heuristic prefers alternatives with the best outcome for one user (see Formula 3). For example, in Table 3 the *most pleasure* value of all diagnoses Δ_i is 0.0 since for user (stakeholder) 3 there does not exist a need to adapt his/her preferences in all of the diagnoses. For a detailed discussion of group decision heuristics we refer to [FBS+2018].

$$leastmisery(\Delta) = argmax_d \bigcup_{s \in users} pref_\delta(s, \Delta) = d$$

Formula 1.

$$average(\Delta) = \frac{\Sigma_{s \in users} pref_\delta(s, \Delta)}{\#users}$$

Formula 2.

$$mostpleasure(\Delta) = argmin_d \bigcup_{s \in users} pref_\delta(s, \Delta) = d$$

Formula 3.

stakeholder	$\Delta_1 = \{r_{11}, r_{13}\}$	$\Delta_2 = \{r_{11}, r_{21}\}$	$\Delta_3 = \{r_{12}, r_{13}\}$	$\Delta_4 = \{r_{12}, r_{21}\}$
1	2	1	2	1
2	0	1	0	1
3	0	0	0	0

Table 3: **Overview of the impact of the different diagnosis Δi on the current preferences of stakeholders**.
For instance, stakeholder 1 has to change two of his / her requirements if diagnosis Δ1 gets selected.

heuristic	$\Delta_1 = \{r_{11}, r_{13}\}$	$\Delta_2 = \{r_{11}, r_{21}\}$	$\Delta_3 = \{r_{12}, r_{13}\}$	$\Delta_4 = \{r_{12}, r_{21}\}$
least misery	2.0	1.0	2.0	1.0
average	0.67	0.67	0.67	0.67
most pleasure	0.0	0.0	0.0	0.0

Table 4: **Evaluation of the different diagnoses using the *least misery*, *average*, and the *most pleasure* heuristic**.
In all three heuristics the ranking criteria for the diagnoses is *less is better*.

OpenReq Contributions

Consensus in Group Decision Making. Presenting diagnoses in situations where user preferences are inconsistent with the underlying configuration knowledge base and/or the preferences of other users is a basic means to trigger discussions and achieve consensus [FZN+2012]. However, further aspects have to be taken into account in order to be able to accelerate the achievement of consensus in group decision making. Approaches that are promising in this context are, for example, the following. User interfaces have to be enriched in order to allow basic negotiation mechanisms between users. An example thereof is the following: stakeholder A is interested in having implemented requirement *reqa* as soon as possible. Furthermore, stakeholder B is interested in having implemented requirement reqb as

soon as possible. Stakeholder A would accept an earlier implementation of *reqb* if stakeholder B accepts an earlier implementation of requirement reqa. In this context, visualization concepts for the representation of the current decision situation will play a major role – alternative ways to represent decision situations are a focus of future work.

Fairness in Group Decision Making. An important issue in group decision making is fairness with regard to group members. Fairness is especially a topic within the scope of repeated decision processes where the same or similar groups are taking a decision. A related example are decisions regarding which features to be included in upcoming software releases. Fairness could be a criteria in such repeated decision scenarios, for example, in the past the requirements of specific departments have been taken more into account than those of others (although similar business impacts can be expected). Fairness also includes visualization aspects since the visualization of the current state of the decision process could help to increase fairness in group decision making, for example, by increasingly taking into account the preferences of other group members.

Predictive Search. Based on the information about already completed group decision processes, diagnosis and repair could be improved by better predicting alternatives acceptable for the whole group. In this context, different types of personalization approaches should be included that help to take into account the preferences of the whole group when determining diagnoses and corresponding repair actions. Diagnosis prediction approaches for single users are already discussed in [FHB+2014], however, in group decision scenarios further related aspects have to be taken into account. The prediction of a relevant diagnosis does not only have to take into account the selection behavior of users but also how users interacted with each other within the scope of a group decision process. Furthermore, the search for alternative configurations has to take into account group preferences, i.e., search heuristics must be learned on the basis of past group interactions.

Negotiation Mechanisms. The main challenge of negotiation mechanisms is to include these in a way that is easy to understand for users. Complex negotiation mechanisms will not be accepted by end-users, i.e., the major challenge is to propose decision and negotiation mechanisms that help to achieve high-quality decisions and consensus as soon as possible and to trigger inconsistency management only in situations where real disagreements exists. For example, if one stakeholder evaluates the risk level of a requirement with 7 (on a scale [1..10]) and the other stakeholder evaluates the same requirement with 8, there seems to be no real disagreement and the system may not have to point out an existing inconsistency.

Intelligent User Interfaces. Since group-based configuration tasks are solved in a distributed and asynchronous fashion, user interfaces should be able to take into account this situation. Figure 24 includes a screenshot of OpenReq Live. In its current version, the system supports user interfaces that foster engagement in requirements engineering processes, approaches to the recommendation of group decisions, efficient methods for planning and also re-planning, semi-automated stakeholder selection, and decision technologies that go beyond preference-based approaches taking into account argumentation-based decision making.

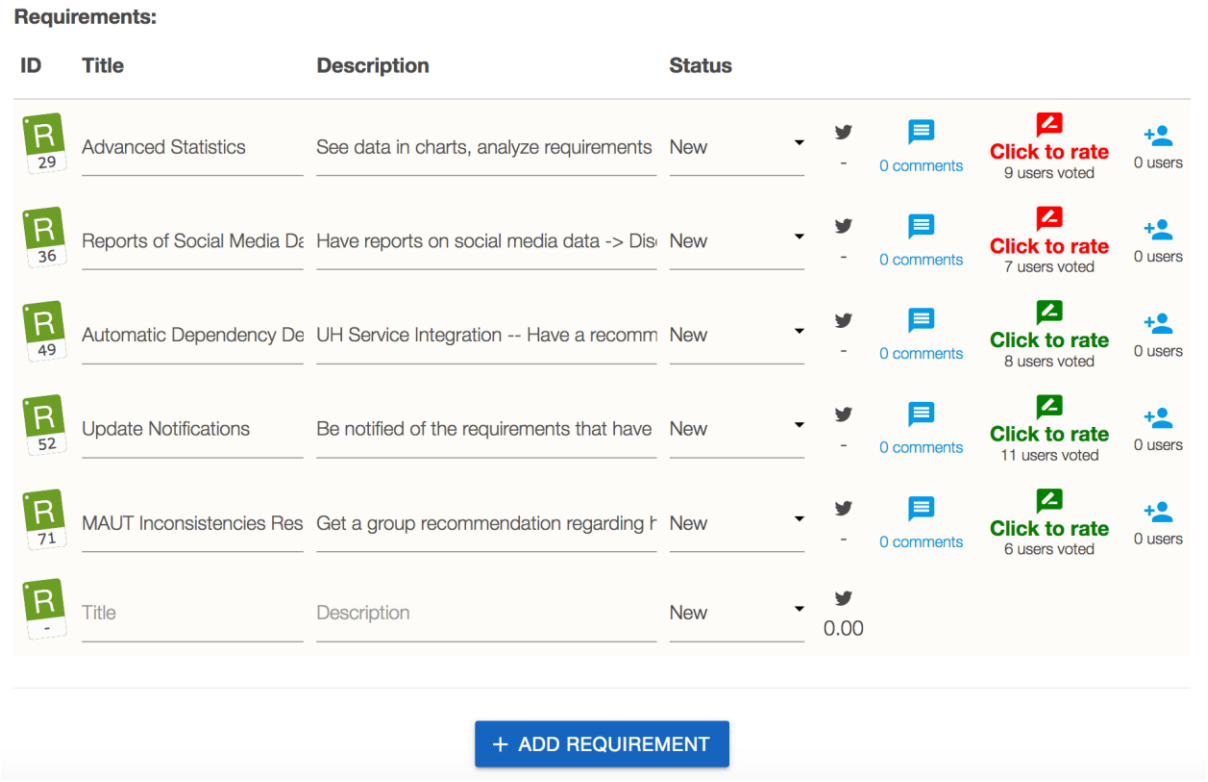


Figure 24: OpenReq Live release planning.
Each entry represents a requirement to be included in one of the defined releases. A red rate indication shows the fact that the current user did not articulate his / her preferences.

4.3 Prioritization

In this section, we focus on the OpenReq prioritization approach especially in the context of the Vogella open source trial. Requirements Engineering especially in open source communities faces the challenge of having to prioritize requirements for individual contributors in a more or less unobtrusive fashion. The main role of prioritization is to support contributors in figuring out the most relevant and interesting requirements to be implemented next and thus avoid time-consuming and inefficient search processes. In this section, we show how utility-based prioritization approaches can be used to support contributors in conventional as well as in open source Requirements Engineering scenarios. As an example of an open source environment, we use BUGZILLA[13]. In this context, we also show how dependencies can be taken into account in utility-based prioritization.

In the following, we provide an overview of application scenarios of utility-based prioritization and discuss specific aspects of requirements prioritization in open source projects. For

[13] www.bugzilla.org

scenarios that include dependencies between requirements, we show how such dependencies can be taken into account on the basis of the concepts of model-based diagnosis [FSZ2012, RE1987]. Furthermore, we present a first version of a user interface developed to support prioritization tasks for BUGZILLA users. Finally, we indicate different issues for future work to further advance the state of the art in utility-based prioritization.

Utility-based prioritization

Utility-based prioritization allows stakeholders to prioritize a requirement with regard to different interest dimensions D = {d1, d2, ..., dn}. Examples of such interest dimensions are *profit*, *risk*, and *effort*. Utility-based prioritization is based on the idea to first evaluate each requirement with regard to the set of interest dimensions (see Table 5) and thereafter calculate the individual utility of each requirement (see Formula 4). In general, the *priority* is associated with the *utility* of a requirement r which results from its total contributions to all of each individual interest dimensions d (denoted as contribution(r, d)) combined with the corresponding importance weights of individual interest dimensions (denoted as weight(d)).

interest dimension	r_1	r_2	r_3
profit	10	5	4
risk	7	2	8
effort	2	3	7

Table 5: Contribution of requirements R = {r1,r2,r3} to dimensions D = {profit, effort, risk}.

interest dimension	weights
profit	0.3
risk	0.5
effort	0.2

Table 6: Predefined weights for the dimensions D = {profit, effort, risk}.

$$utility(r, D) = \Sigma_{d \in D}(contribution(r, d) \times weight(d))$$

Formula 4.

Applying Formula 4 to the entries in Tables 5 and 6 results in the ranking depicted in Table 7 (the higher the utility with regard to the given interest dimensions, the higher the corresponding priority of the requirement).

requirement	r_1	r_2	r_3
utility	6.9	3.1	6.6
priority (ranking)	1	3	2

Table 7: Ranking of requirements with static weights.

In the previous example, the evaluation of requirements with regard to interest dimensions and the weighting of interest dimensions are assumed to be predefined (e.g., by a single stakeholder). However, requirements prioritization is often a group decision process where different stakeholders are evaluating requirements (see, e.g., Table 8) and define importance weights with regard to interest dimensions (see, e.g., Table 9). Both, stakeholder-individual evaluations of interest dimensions and importance weights have to be aggregated. Formula 5 shows the aggregation of stakeholder-individual evaluations of requirements where S refers to the set which includes all m stakeholders (i.e., S = {s1 , s2 , ..., sm }).

$$contribution(r, d, S) = \frac{\Sigma_{s \in S} eval(d, r, s)}{|S|}$$

Formula 5.

Formula 6 shows how to aggregate the stakeholder-specific importance weights (denoted as w(d, s)) which are related to individual interest dimensions d. Previous calculations did not take into account potential different degrees of stakeholder expertise, for example, a stakeholder sa could have more expertise with regard to estimating the market potential of a requirement in terms of profit as estimating the corresponding development efforts. To take into account this aspect, Formula 6 includes a factor that represents the *expertise* of a stakeholders with regard to a specific interest dimension d.

interest dimension	r_1				r_2				r_3			
	s_1	s_2	s_3	AVG	s_1	s_2	s_3	AVG	s_1	s_2	s_3	AVG
profit	5	2	2	3.0	5	1	2	2.7	2	2	6	3.3
risk	3	3	4	3.3	2	5	6	4.3	3	2	2	2.3
effort	2	3	2	2.3	3	4	2	3.0	5	6	2	4.3

Table 8: Contributions of the requirements R = {r1,r2,r3} to the dimensions D = {profit,effort, risk} (defined by the stakeholders S = {s1,s2,s3}).

$$weight(d, S) = \frac{\Sigma_{s \in S} w(d, s) \times expertise(d, s)}{|S|}$$

Formula 6.

Similar to the basic approach, the utility of a requirement (Formula 7) is determined as a combination of the contributions of a requirement to the given interest dimensions and related interest dimension importance evaluations of stakeholders.

$$utility(r, D, S) = \Sigma_{d \in D}(contribution(r, d, S) \times weight(d, S))$$

Formula 7.

stakeholder	s_1	s_2	s_3	weights
profit	0.5	0.3	0.6	0.47
risk	0.3	0.6	0.3	0.4
effort	0.2	0.1	0.1	0.13

Table 9: Preferences of stakeholders S = {s1,s2,s3} with regard to the interest dimensions D = {profit,effort,risk}.

The result of applying Formulae 5–7 to the evaluation data contained in Tables 8 and 9 is depicted in Table 10.

requirement	r_1	r_2	r_3
utility	3.03	3.57	3.03
priority (ranking)	2	1	2

Table 10: Ranking of requirements with group weights.

Utility based Prioritization in BUGZILLA

In the previous section, we took a look at different variants of utility-based prioritization. These variants were discussed on the basis of interest dimensions (evaluation criteria) typically occurring in software projects where a group of stakeholders is in charge of jointly defining and prioritizing requirements. In this section, we focus on open source scenarios where individual users (e.g., contributors in an open source platform) follow their individual interests without necessarily taking into account the preferences of other users. We now show how utility-based prioritization can be applied in such contexts.

Table 11 represents a BUGZILLA-specific evaluation of requirements (bugs) with regard to the set of interest dimensions {*cc*, *geritt*, *blocker*, *comments*}. In this context, *cc* is the number of contributors who are in the :cc list of bug-related emails, *geritt* is the number of bug-related GERITT[14] changes, *blocker* is the number of dependent bugs (dependent requirements), and *comments* refers to the number of bug-related comments. Formula 8 supports the calculation of the contribution of a requirement r to a specific interest dimension d. In sharp contrast to the previous scenarios, the contribution is not directly specified by stakeholders but derived from BUGZILLA specific information (e.g., #comments related to a requirement).

$$contribution(r, d) = eval(r, d)$$

Formula 8.

Formula 9 supports the determination of the expertise of a stakeholder which is represented in terms of the similarity between the keywords stored in the stakeholder profile and those

[14] A code review tool - www.gerritcodereview.com

extracted from the requirement description and corresponding meta-information (e.g., name of associated component/system). The similarity[15] between requirement-related keywords and meta-information and the information stored in the profile of the contributor is interpreted as expertise (see Formula 9).

$$weight(r, s) = expertise(r, s)$$

Formula 9.

In the line of the previously discussed utility functions, the overall utility of a requirement is interpreted as a combination of (1) the contributions of a requirement to a set of interest dimensions and (2) the expertise level of a stakeholder (in this context interpreted ”globally”, i.e., not on the level of individual interest dimensions).

$$utility(r, s) = \Sigma_{d \in D} contribution(r, d) \times weight(r, s)$$

Formula 10.

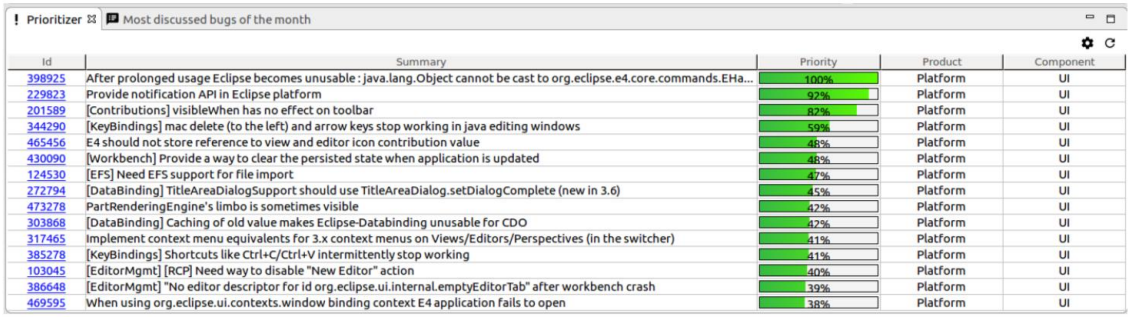
interest dimension	r_1	r_2	r_3
cc	5	2	2
geritt	3	3	4
blocker	2	3	2
comments	2	3	2

Table 11: Contribution of requirements (bugs) R = {r1,r2,r3} to the interest dimensions D = {cc, geritt, blocker, comments}.

stakeholder	s_1
r_1	0.5
r_2	0.3
r_3	0.2

Table 12:Expertise of stakeholder s1 with regard to the requirements {r1,r2,r3}
(determined, for example, on the basis of the similarity between the stakeholder profile and information associated with a requirement.)

[15] Similarity calculation is not explicitly explained here



Id	Summary	Priority	Product	Component
398925	After prolonged usage Eclipse becomes unusable : java.lang.Object cannot be cast to org.eclipse.e4.core.commands.EHa...	100%	Platform	UI
229823	Provide notification API in Eclipse platform	92%	Platform	UI
201589	[Contributions] visibleWhen has no effect on toolbar	82%	Platform	UI
344290	[KeyBindings] mac delete (to the left) and arrow keys stop working in java editing windows	59%	Platform	UI
465456	E4 should not store reference to view and editor icon contribution value	48%	Platform	UI
430090	[Workbench] Provide a way to clear the persisted state when application is updated	48%	Platform	UI
124530	[EFS] Need EFS support for file import	47%	Platform	UI
272794	[DataBinding] TitleAreaDialogSupport should use TitleAreaDialog.setDialogComplete (new in 3.6)	45%	Platform	UI
473278	PartRenderingEngine's limbo is sometimes visible	42%	Platform	UI
303868	[DataBinding] Caching of old value makes Eclipse-Databinding unusable for CDO	42%	Platform	UI
317465	Implement context menu equivalents for 3.x context menus on Views/Editors/Perspectives (in the switcher)	41%	Platform	UI
385278	[KeyBindings] Shortcuts like Ctrl+C/Ctrl+V intermittently stop working	41%	Platform	UI
103045	[EditorMgmt] [RCP] Need way to disable "New Editor" action	40%	Platform	UI
386648	[EditorMgmt] "No editor descriptor for id org.eclipse.ui.internal.emptyEditorTab" after workbench crash	39%	Platform	UI
469595	When using org.eclipse.ui.contexts.window binding context E4 application fails to open	38%	Platform	UI

Figure 25: BUGZILLA view on bugs (requirements).
Based on the presented utility-based prioritization approach, bugs are presented to BUGZILLA contributors in the order of the determined personalized priority.

requirement	r_1	r_2	r_3
utility	6.0	3.3	2.0
priority	1	2	3

Table 13:Ranking of Bugzilla bugs with static weights.

Dependencies

Utility-based recommendation approaches per se do not explicitly take into account dependencies between requirements (e.g., requirement A must be implemented before requirement B). In the discussed open source prioritization scenario, this aspect is taken into account by prioritizing requirements on the basis of the number of related dependencies, i.e., the higher the number of requirements dependent on a requirement x, the higher the ”global” relevance of x. In this context, the *blocking factor* (i.e., how many requirements depend on the implementation of requirement x) can be considered as interest dimension that has an impact on prioritization. In other words, this requirement should be implemented as soon as possible since it otherwise blocks the implementation of other requirements. This approach can also be applied in software development scenarios where a group of stakeholders (e.g., an in-house software development project) is in charge of prioritizing requirements. Such an approach helps to avoid situations where prioritizations violate dependency constraints but cannot explicitly exclude such situations.

An alternative approach is to apply repair mechanisms from model-based diagnosis [8] that help to adapt already determined prioritizations in such a way that all defined dependencies are taken into account. In the following, we will shortly sketch our approach. In order to trigger a diagnosis process, we are in the need of a predefined set of dependencies between requirements (denoted as DEP = {dep1, dep2, .., depn}). Furthermore, we assume that a prioritization (represented as sequence) P = [p1 , p2 , .., pm] determined by a utility-based prioritization approach is *inconsistent* with the given set of dependencies. In order to apply model-based diagnosis, we assume that both, the pre-defined set of dependencies and the requirement prioritization is represented in terms of constraints [19], for example, DEP = {dep1 : r3 < r1,dep2 : r3 < r2} and P = {p1 : r1 < r2,p2 : r2 < r3,p3 : r3 < r4,p4 : r4 < r5,p5 : r5 < r6}. As can be easily seen, DEP ∪ P is inconsistent. As variable domains we assume [1...#requirements].

Following the principles of model-based diagnosis, we need to detect all *minimal conflicts* induced in P by the dependencies defined in DEP. In our example, CS : {p2} is a conflict induced in P by the dependencies defined in DEP. CS is minimal, i.e., we need to adapt only one of the prioritization elements in CS such that a global prioritization can be found that is consistent with the elements in DEP. A corresponding diagnosis Δ is {p2}. In our example, we could decide to replace p2 :r2 < r3 with the corresponding repair r3 < r2. This is a repair action that helps to restore the consistency of DEP ∪ P. Our approach to the repair of inconsistent prioritizations can be used for both, *interactive prioritization* where stakeholders receive feedback on the consistency of prioritizations, and *automated prioritization* where repairs for inconsistent prioritizations are determined in an automated fashion. Important issues to improve our approach are discussed in the following.

OpenReq Contributions

In this section, we described how to support utility-based requirements prioritization. These scenarios range from *single user prioritization* where one stakeholder is in charge of completing prioritization tasks to *group-based prioritization* where the preferences and evaluations of different group members have to be taken into account. On the basis of these scenarios we showed how utility-based prioritization can be applied in the context of open source development projects. In this context, we sketched our initial implementation currently provided in the BUGZILLA environment (the Vogella trial). This implementation serves as a first version to support prioritization in BUGZILLA.

Future Work. Since prioritization is a repetitive process, we will include mechanisms that are capable of learning stakeholder weights and also the weights of individual requirements. This approach will help to further increase the prediction quality of prioritizations in terms of the probability that stakeholders accept the proposed prioritizations. In this context, we will also compare the predictive quality of utility-based approaches (i.e., approaches based on aggregated models) with machine learning based approaches and approaches that determine rankings on the basis of aggregated prioritizations. Furthermore, we will analyze which further features (interest dimensions) are useful to improve prediction quality. For example, the number of redundant bugs (issues) in can be a further important relevance indicator. A major challenge in requirements prioritization is the provision of persuasive user interfaces that increase the preparedness of stakeholders to actively engage in requirements engineering processes. Consequently we will focus on a further extension/improvement of the existing BUGZILLA requirements prioritization user interface. Finally, we will analyze in which way recommended prioritizations have to be *explained* to support specific group decision goals such as *consensus*, *fairness*, and *decision quality*.

4.4 Argumentation-based Prioritization

OpenReq Live includes functionalities that support group decision making scenarios in different stages in the context of requirements engineering. For example, within the scope of release planning, stakeholders have to discuss strategic directions and features of the requirements. A related discussion can be supported by a pro/con analysis where positive and

negative arguments with regard to specific features/ requirements can be provided and shared among the participants of the meeting. Figure 26 depicts an OpenReq Live user interface where the balance between pro-arguments (green), counter-arguments (red), and neutral-arguments (grey) are summarized.

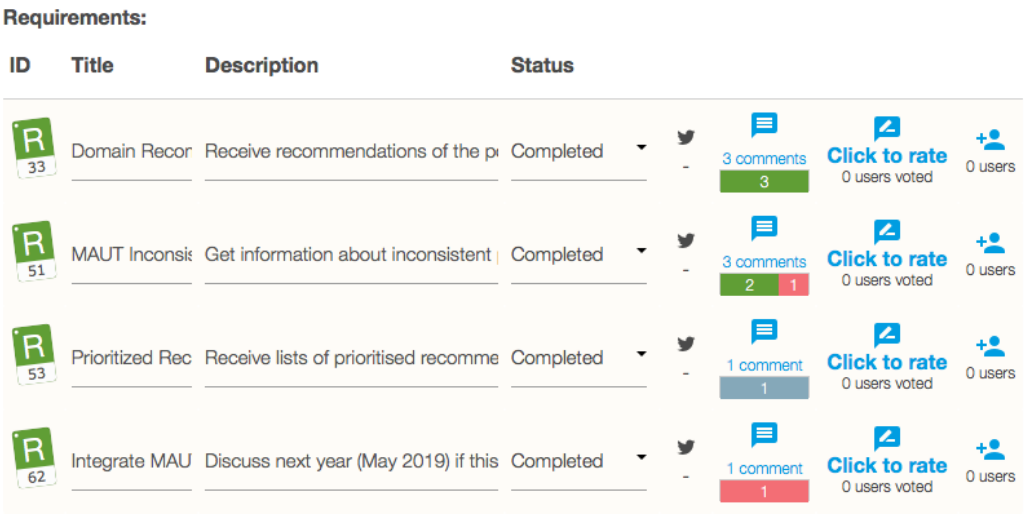


Figure 26: Pro / Con Analysis in OpenReq Live.

In this context, OpenReq Live provides a ranking functionality for proposed alternatives that is based on the concepts of multi-attribute utility theory [DY2005]. The calculation of the utility of an alternative is based on Formulae (11) and (12). This approach can be considered as a specific type of group recommendation [FBS+2018] since the preferences (arguments) of individual users are aggregated into a global ranking that is recommended to a group of users (i.e., not a single person). Since positive (pos) and negative (neg) requirement-specific arguments (args) can be supported by stakeholders (users), the degree of support is taken into account by the utility function. In this context, S_x denotes the number of supports for the current argument whereas S_{total} denotes the total number of supports of all arguments.

$$utility\ (a) = \sum_{pos \in args} support(\text{pos}) - \sum_{neg \in args} support(\text{neg})$$

Formula 11.

$$support\ (x) = 1 + \frac{Sx}{1+Stotal}$$

Formula 12.

OpenReq Contributions

In this section we provided a basic introduction to concepts that help to tackle challenges triggered by software variability in the context of requirements management processes (e.g.,

release planning). We discussed these aspects on the basis of OpenReq Livd that supports the creation and management of release plans.

4.5 Stakeholder Selection

Besides finding the right person to implement a requirement in a SW project, “Requests for Proposal” (RFP) are typical RE scenarios where a stakeholder selection is essential. Requests for proposal trigger company-internal requirements management (RM) processes in order to assure that offers comply with a given set of customer requirements. As traditional RM approaches require a deep involvement of the requirements manager of a RM project especially when it comes to assigning suitable stakeholders to requirements, the quality of the decisions and the time effort for making correct decisions mainly depends on these experts. In this section, we present a novel stakeholder assignment approach that reduces the overall involvement of these experts and also limits the uncertainty of overseeing suitable stakeholders at the same time. The assignment of responsible stakeholders is represented as a group decision task expressed in the form of a basic configuration problem. The outcome of such a task is a configuration which is represented in terms of an assignment of responsible stakeholders to corresponding requirements.

Application Scenario

Whenever an organizational unit of a large company (e.g., Siemens) decides to bid for a Request for Proposal (RFP), a new bid project for that proposal is initiated and the necessary stakeholders of the bid project are identified. RFPs for technical systems usually consist of a set of PDF or Microsoft Word documents which describe all requirements for the requested system covering technical, financial, legal, etc. aspects. Examples of stakeholders can be project managers, system architects, requirements managers, quality management departments, legal departments, engineering departments relevant for the bid, and potential external suppliers.

Within the context of a bid project, a requirements management (RM) process is initiated at the beginning. The purpose of this process is to assure that no requirement of the RFP has been overlooked. It involves the extraction of all the requirements contained in the RFP documents. The identified requirements must be assessed by the relevant stakeholders. This means that requirements concerning contracts must be assessed by the stakeholder(s) of the legal department, technical requirements must be assessed by the affected engineering department, etc. The assessment may involve statements about various criteria such as compliance, risks, approaches, etc. These statements are interpreted as evaluation dimensions in the remainder of this section. At the end, each requirement of the RFP must have been assessed by at least one appropriate stakeholder.

Traditional Requirements Management Process

The traditional requirements management process can be best explained with an example. In the following, we describe a simplified example of a traditional RM process in a rail automation context based on a conventional RM tool such as IBM DOORS.

At the beginning, the requirements manager of the bid project creates a new project in the RM tool. After that, the necessary stakeholders for the current bid project are defined. In this context, stakeholders do not necessarily correspond to persons but correspond to roles which are uniquely identified with a unique string (called Domain). These string-based identifiers are unique within the organization. Furthermore, the RM tool supports the mapping of existing roles (i.e., domain identifiers) to concrete persons within the bid project. This way, responsible persons are assigned to roles based on their skills and domain knowledge. Table 14 presents examples of domain identifiers which occur in rail automation. For such large bid projects usually more than 50 different domains are defined with the RM tool. However, in practice, most projects only use 20 different domains on average.

Domain	Stakeholder
PM	project manager
SA	system architect
RM	requirements manager
RAMS	reliability, availability, maintainability, and safety
S(ignal)	engineering department for railway signals
PS	engineering department for power supply
TVD	department for track vacancy detection
ETCS	department for European Train Control System
Test	quality management department
Supplier1	external supplier, subcontractor

Table 14: Examples of domain identifiers for rail automation.

As a next step, the requirements manager imports all the relevant documents of the RFP into the project by using the RM tool. The RM tool automatically converts each paragraph of the document into a (potential) requirement whilst the structure of the document is preserved. The requirement manager then classifies the (potential) requirements in the project as either an actual requirement or as an arbitrary comment (called prose). In general, large infrastructure projects may contain more than 10,000 (potential) requirements.

Each (actual) requirement must be assessed by at least one stakeholder. The requirements manager has to figure out which stakeholders are appropriate for which requirements and needs to assign them accordingly. However, other stakeholders may improve such initial assignments later during the assessment phase. The RM tool notifies all assigned stakeholders via e-mail to assess the requirements they are assigned to.

Table 15 shows an example of an initial assignment done by the requirements manager (RM). In this table, each row corresponds to a requirement and each column refers to a stakeholder. Each cell represents a single decision (of a stakeholder) for a stakeholder assignment (to a requirement). At the beginning, only the RM proposes assignments of potential stakeholders to requirements based on the manager's expertise and knowledge. For example, the assignment of {S, PM } to the requirement R5 in the RM column indicates that R5 has been initially assigned to the signal department (S) and to the project management department (PM) by the requirements manager (RM). As only the RM makes assignments in this initialization phase, the values of all other columns remain empty (i.e., are filled with the "-" label) until the assessment phase.

Req	RM	PM	RAMS	S(ignal)
R1	{PM}	-	-	-
R2	{PM}	-	-	-
R3	{S}	-	-	-
R4	{S}	-	-	-
R5	{S, PM}	-	-	-
...				

Table 15: Initial assignment of stakeholders to requirements done by requirements manager (RM).
The symbol "-" indicates that the other stakeholders have not made a decision yet.

Next, in the assessment phase, the affected stakeholders take a look at each of their assigned requirements in the RM tool and can either accept the requirement and assess it or they can veto the proposed assignment. Additionally, they can also propose an alternative stakeholder for the requirement or suggest (although rarely) an additional stakeholder for the requirement. For the remainder of this section, this process is hereinafter referred to as assignment feedback. After that, the requirements manager can either accept the veto and assign the requirement to a different stakeholder or decline the veto and reassign the stakeholder to the requirement. Table 16 shows an intermediate state during the assignment phase which demonstrates examples of assignment feedback given by the stakeholders PM and S(ignal):

Req	RM	PM	RAMS	S(ignal)
R1	{PM}	{PM}	-	-
R2	{PM}	{RAMS}	-	-
R3	{S}	-	-	{S, RAMS}
R4	{S}	-	-	{}
R5	{S, PM}	{S, PM}	-	{S, PM}
...				

Table 16: State of assignment during assessment phase.

- Requirement R1 has been accepted by PM
- Requirement R2 has been vetoed by PM and RAMS has been proposed by PM as alternative stakeholder
- Requirement R3 has been accepted by S(ignal), but RAMS has been proposed by S(ignal) as an additional stakeholder
- Requirement R4 has been vetoed by S(ignal)
- Requirement R5 has been accepted by all proposed stakeholders

It is important to point out the fact that in the traditional scenario, it is always the main responsibility of the requirements manager to resolve potential conflicts. Typically, this will usually involve some personal discussions with the involved stakeholders and some final decisions made by the requirement manager. These final decisions will then assure a consistent assignment of all requirements to responsible stakeholders. Table 17 presents such a final state where all conflicts have been resolved.

Req	RM	PM	RAMS	S(ignal)
R1	{PM}	{PM}	-	-
R2	{RAMS}	{RAMS}	{RAMS}	-
R3	{S, RAMS}	-	{S,RAMS}	{S, RAMS}
R4	{S}	{S}	-	{S}
R5	{S, PM}	{S, PM}	-	{S, PM}
...				

Table 17: Final state after assessment. Assignment of stakeholders to requirements.

The requirements manager periodically reminds the assigned stakeholders about their unassessed requirements. This process is repeated until all requirements have been assessed and the assessment phase is finished. Thus, the assignment of stakeholders can be considered as a manual configuration process. The outcome of this process is a configuration in terms of a consistent assignment of stakeholders to requirements they are responsible for. In our implementation, the overall goal is to achieve consensus regarding the stakeholder assignment. Future versions of our system will include further constraints that have to be taken into account in task allocation tasks as discussed in this section.

Requirements Management Process with Group Decision Support

The main idea of our novel requirements management approach is to introduce additional stakeholder votes made by artificial stakeholders (called bots). Furthermore, an intelligent group decision service is included in the RM tool to automatically aggregate all votes given by human stakeholders as well as artificial stakeholders. On a technical level, such a group decision service represents a group recommender system which generates recommendations based on aggregated votes given by group members of a group (i.e., the stakeholders). Basically, there exist different strategies on how to aggregate votes of group members [FBS+2018] such as majority, average, least-misery, etc. In addition, more sophisticated aggregation functions exist - for further information regarding preference aggregation functions we refer to [FBS+2018]. To limit the scope of this section, we assume that the group decision service is a simple group recommender using basic aggregation strategies.

The votes of the artificial stakeholders (i.e., bots) are generated by using appropriate content-based recommendation algorithms. This way, the group decision service allows to replace the traditional mainly manual stakeholder assignment process (see above) with a semi-automatic process. As a key difference to the traditional approach, the group decision service automatically aggregates the decisions of all voters and thereby allows the smart incorporation of additional (automatic) voters, i.e., intelligent recommendation services for stakeholder assignments. From an abstract point of view, the process can be interpreted as a basic configuration process. Like in the traditional RM process, the outcome of this process represents a consistent assignment of stakeholders to requirements they are responsible for.

Table 18 illustrates a possible initial state in the presence of a group decision service (GDS) and a stakeholder assignment recommendation service (denoted as RS1). In sharp contrast to the assignments made by other stakeholders, the recommendation service does not provide a binary decision for every stakeholder but a confidence value which lies in the range between

1 and 10, whereby a higher number corresponds to more confidence and a lower number corresponds to a lower level of confidence.

Req	GDS	RS1	RM	PM	RAMS	S(ignal)
R1	{PM}	{PM:9}	{PM}	-	-	-
R2	{RAMS}	{RAMS:8, PM:5}	-	-	-	-
R3	{S}	{S:8, RAMS:6}	-	-	-	-
R4	{S}	{S:5}	-	-	-	-
R5	{S}	{S:6}	{S,PM}	-	-	-
...						

Table 18: State of assignment with group decision service (GDS) and stakeholder recommendation service (RS1).
The recommendation service provides a confidence value which lies in the range between 1 and 10.

The column for the GDS shows the result of the group decision service for each requirement, i.e., the aggregated decision of all voters (including humans and bots/algorithms). Note that a clear benefit of the group decision service is that some requirements can already be assessed by the assigned stakeholders, even though they have not yet been proposed/assigned by the requirements manager. In other words, stakeholders are automatically proposed by the bots/algorithms based on their skills in the initial phase and can already evaluate their assignment to the requirements. Hence, much assignment effort is taken away from the time-pressured requirements managers and the initial phase can be significantly speeded up. Moreover, it is necessary to point out that the stakeholders GDS (perform aggregation) and RM (perform final decision) can be considered to have a special role in this evaluation process, whereas all other stakeholders only occur as voters in the process. Consequently, the major responsibility/task of a RM in this process is to review the decision suggested by the GDS and to perform the final decision about the assignment of the stakeholders to the requirements.

Potential Issues of Group Decision Support

The exact behavior of the OpenReq Live group decision support will depend on various factors. Examples of such factors include the aggregation strategy used by the group decision service to aggregate the votes (e.g., majority, average, etc.), the individual weight of the voters (e.g., ”deciders”/experts count higher than normal stakeholders), and the confidence/trust users have in different recommendation algorithms. Furthermore, the question arises how conflicting decisions (for example, stakeholder A assigns stakeholder B and B assigns A) can be resolved or supportive advice to manually resolve such conflicts can be given to the voters by the system. Moreover, the prediction quality (i.e., performance) of the artificial stakeholders (i.e., the recommender systems) plays a major role in the process. In particular, the generated recommendations should be evaluated and examined with respect to completeness. In terms of common information retrieval measures (such as precision and recall), this would, for example, mean that more emphasis should be given to the recall of the results rather than the precision achieved by the recommender. In addition to that, an appropriate recommendation algorithm should also be capable of giving negative indication by telling the RM which stakeholders are definitely not suitable to be assigned to a requirement at all. Such a negative indication can be shown as, e.g., RAMS:0. Finally, another important aspect would be to take the availability of stakeholders into account before they get finally assigned to a requirement. This adds another complexity dimension to the underlying basic configuration problem.

Group Decision Support for Bidding Processes

In this section, a slightly modified version of the aforementioned RM process based on group decision support is described. The description explains the technical implementation of this process provided by OpenReq Live serving a requirements engineering plattform.

In the initial phase, the requirements manager (RM) is asked by the system to propose suitable stakeholders for each requirement. A content-based recommender system (RS1) helps the RM to find stakeholders based on keywords extracted from former requirements those stakeholders have solved. Thereby, the recommender automatically extracts relevant keywords from the title and description text of all former requirements a stakeholder was assigned to, in order to build a user profile for the respective stakeholder. First, the title and description text is cleaned by removing special characters (such as ”.”, ”,”, ”;”, ”#”, etc.). Next, the text is split into tokens (which, basically, represent the words in the text) and stop words such as prepositions (e.g., ”in”, ”on”, ”at”, etc.) or articles (e.g., ”the”, ”a”, ”an”), are removed. After applying Part-of-speech tagging, tokens/words of classes (such as, verbs, adjectives, or numbers) that are most probably irrelevant to be used as keywords, are removed. Finally, the remaining tokens of each former requirement the stakeholder was assigned to, are merged together into a single user profile.

By applying the same procedure to new requirements, keywords for new requirements are extracted as well. Given the keywords of a new requirement and the user profiles of the individual stakeholders, a similarity between a new requirement and a stakeholder is calculated for every stakeholder provided that the stakeholder has been assigned to an (already completed) requirement. Formula 13 shows the Dice coefficient formula which is a variation of the Jaccard coefficient and used to compute the similarity between a stakeholder and a requirement. The similarity is measured by comparing the overlap of the keywords of the stakeholder’s user profile (denoted as U_a) and the relevant keywords of the respective requirement (denoted as rx) with the total number of keywords appearing in Ua as well as rx.

$$sim(U_a, r_x) = \frac{2 * |keywords(U_a) \cap keywords(r_x)|}{|keywords(U_a)| + |keywords(r_x)|}$$

Formula 13.

Stakeholders who are most similar to a given requirement are suggested by the content-based recommender to the RM. This way, the initial phase can be speeded up and the chance of overseeing suitable stakeholders for requirements at this early stage of the process, is decreased. In the next step, OpenReq Live shows a list of the initially assigned stakeholders for each requirement. Stakeholders who are assigned to a requirement can either accept or reject their assignment. In addition, the assignments of the stakeholders for the requirement can be evaluated by all stakeholders.

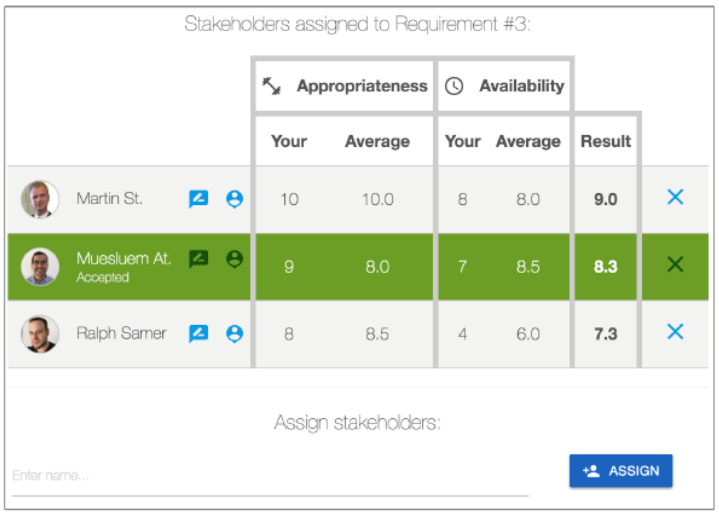


Figure 27: Evaluation of stakeholders in the OpenReq prototype.
Each stakeholder-assignment is evaluated by two evaluation dimensions (appropriateness and availability). The utility value of an evaluated stakeholder is calculated by using Formula 14.

This evaluation of a stakeholder-assignment is done based on the criteria Appropriateness and Availability (see Figure 27). Both criteria are interpreted as evaluation dimensions and stakeholders are evaluated based on both dimensions. Furthermore, an assigned stakeholder can also propose the assignment of further stakeholders to the requirement. These newly assigned stakeholders can then be evaluated again. After a new vote has been given, the group decision service (GDS) is triggered to compute a utility value for the rated stakeholder. Formula 14 shows the calculation of the utility value of an evaluated stakeholder s, whereas D is the set containing both dimensions, i.e., D = {Appropriateness, Availability}.

$$utility(s,r) = \frac{\sum_{t \in T} \frac{\sum_{d \in D} eval(s,r,d,t) \cdot weight(d)}{\sum_{d \in D} weight(d)}}{|T|}$$

Formula 14.

The formula describes the stakeholder ***s*** to be voted by other stakeholders, whereby ***T*** represents the set of stakeholders t ∈ T who evaluated s. More formally expressed, T is a set which contains the stakeholders (including s) who evaluated stakeholders, i.e., T ⊆ S. Furthermore, OpenReq Live allows the requirements manager to define different importance levels for both dimensions. In Formula 14, the importance of a dimension d ∈ D is expressed by the function weight(d). Moreover, eval(s, d, t) refers to the dimension-specific rating given by stakeholder t to stakeholder s for the requirement r. Finally, the result of utility(s, r) represents the aggregated utility of a stakeholder s for requirement r.

Once all assignments have been evaluated by a sufficient number of stakeholders, a stable state of the assignment utilities is achieved. The utility values are then used as main feedback source for the requirements manager to make the final decision about which stakeholder(s) should be assigned to the requirement.

OpenReq Contributions

The major contributions of this subsection are the following. First, we analyzed in detail a real-world scenario of a typical bid project. Second, we showed an approach to identify relevant stakeholders for specific requirements and thus generate a global assignment of stakeholders to requirements. We discussed application scenarios in the context of industry projects, ranging from traditional requirements management processes, where the assignment process of stakeholders is solely controlled by the requirements manager, to more sophisticated automated approaches where the involvement of the requirements manager is reduced to a minimum. An implementation of the latter as a basic configuration service includes artificial stakeholders as additional voters and a group decision support system as a vote aggregation component. On the basis of this scenario we showed how these two adaptations can be applied in order to improve the requirements management process such that the overall effort and the chance of overseeing stakeholders suitable for requirements can be reduced for the time-pressured requirements managers.

Future Work. As bidding processes can be seen as repetitive processes, mechanisms which are capable of learning stakeholder weights and taking individual expertise levels of stakeholders into account are considered as future work. Moreover, the set of existing evaluation dimensions can be further extended such that more fine-grained control is given to the evaluation process as well as to the group decision service. Additionally, the concept of liquid democracy will be integrated into the evaluation process (ongoing work) [JM2014]. This way, stakeholders who do not have sufficient knowledge concerning the details of a requirement can easily delegate their votes to more well-informed and experienced experts. In order to resolve conflicts, future work will include mechanisms to automatically resolve such conflicts or mechanisms which provide supportive advice to the voters and shows them how to manually resolve such conflicts. There is still plenty of room for improvement regarding the extraction of keywords used by the discussed content-based recommender system (i.e., artificial stakeholder). For example, a more descriptive and characteristic representation of the keywords can be obtained by using more sophisticated content-based approaches such as Latent Semantic Analysis (LSA) [LFL1998].

5. Decision Biases in Group Decision Contexts

The intensity of domain knowledge exchange among group members is an important factor that directly influences group decision quality. The more frequent information is exchanged among group members, the higher the quality of the corresponding decision. In this section we present results of an empirical study conducted with groups of students – the task of each group was to take a decision regarding the exam topics the group prefers (a similar task in the context of requirements engineering would be the selection of the adequat dimensions/properties for the evaluation of the requirements). This group decision had to be taken on the basis of a group decision support environment with included recommendation functionality and a discussion forum that allows for information exchange among group members. Depending on the included variant of the group recommendation algorithm, groups received recommendations that varied in terms of recommendation diversity. The results of the study show that increased recommendation diversity leads to an increased degree of information exchange among group members.

In this section, we focus on the aspect of recommendation diversity on the frequency of information exchange between group members. We integrated different recommendation strategies with a varying degree of recommendation diversity into our group decision support environment [SFL+2015] and analyzed the impact of recommendation diversity on knowledge interchange between users. The underlying idea is that too similar recommendations provide only a limited coverage of the whole item space and increased diversity helps to introduce new alternatives and to trigger discussions/information exchange with regard to these alternatives. In contrast to the mainstream in recommender systems research [JZF+2010], we do not focus on improving the prediction quality of recommendation approaches. Our aim is to investigate possibilities to exploit recommendation technologies to foster intended behavior which can also be interpreted as a kind of persuasive technology. In group decision scenarios, it is often more important to increase the performance of the group and foster group members' information exchange, than predicting decisions that will be taken by the group. Based on this idea, we analyze the impact of recommendation diversity on the degree of knowledge exchange in a group. This section analyzes three different basic group recommendation heuristics (aggregation functions) (min, avg, and max group distance) with regard to their impact on the communication behavior (knowledge exchange) within a group.

Preference Aggregation Mechanisms

Different preference aggregation mechanisms were used in our study that was conducted on the basis of our group decision support environment Choicla [SFL+2015]. This system includes different group preference aggregation mechanisms from social choice theory [18] – GD_{min}, GD_{max} and GD_{avg} (see Formulae below) have been included for the purpose of the work presented in this section. The mentioned aggregation mechanisms differ from each other especially with regard to the calculated diversity (see Formula 15). In this context, diversity(d) is interpreted in terms of the deviation of recommendations d (recommended evaluation of specific alternatives, i.e., exam modes) from the evaluations provided by individual group members (eval(u,s) where u is a user and s represents a specific alternative/item, e.g., an exam mode).

$$diversity(d) = \frac{\sum_{u \in Users} |eval(u,s) - d|}{\#Users}$$

Formula 15.

The following group aggregation mechanisms were used within the scope of our study. First, the minimum group distance (GD_{min}) determines a rating d(rating scale [1..5]) that reflects the minimum distance to the individual preferences of the group members (see Formula 16). Consequently, Formula 16 implements a low-diversity recommendation approach that tries to take into account the initial preferences of group members.

$$GD_{\min}(s) = arg \min_{d \in \{1..5\}} \left(\sum_{u \in Users} |eval(u,s) - d| \right)$$

Formula 16.

Maximum group distance (GD_{max}) returns a rating d that represents the maximum distance to the preferences of individual group members (see Formula 17). Consequently, Formula 17 implements a high-diversity recommendation approach that often neglects the preferences of individual group members.

$$GD_{\max}(s) = arg \max_{d \in \{1..5\}} \left(\sum_{u \in Users} |eval(u,s) - d| \right)$$

Formula 17.

Finally, average group distance (GD_{avg}) represents a value between maximum and minimum group distance (see Formula 18) and thus can be considered as a compromise between minimum and maximum group distance.

$$GD_{avg}(s) = \frac{GD_{\min}(s) + GD_{\max}(s)}{2}$$

Formula 18.

These aggregation functions were used as a basis for the user study discussed in the following section.

Empirical Study

Our user study on the impact of different aggregation functions on the preparedness of group members to exchange information has been conducted on the basis of the Choicla decision support environment [SFL+2015]. A screenshot of the Android version of Choicla is depicted in Fig. 28. N = 256 computer science students (12% female, 88% male) participated in the study – all students were enrolled in a software engineering course (object-oriented analysis

and design) and assigned to a group that had to implement a software within the scope of the course. Within the scope of our user study, each of these groups also had to choose a preferred exam mode for object-oriented analysis and design. An example of such an exam mode is: a theoretical question on State Charts (SC), a theoretical question on Sequence Diagrams (SD), and two practical exercises on Object-Relational Mapping (ORM).

This underlying task of the user study was chosen to get a representative amount of participants (we conducted this study in a software engineering course @ TU Graz) - as previously mentioned the similar study in the context of requirements engineering would be the selection of appropriate rating dimension which should then serve as a basis for the preference acquisition of all group members in the release planning process.

All study participants were aware about the fact that there is no guarantee that the preferred exam mode will be taken into account in upcoming exams. The task of each group was to select a specific exam mode on the basis of the Choicla decision support environment. Figure 28 depicts example screenshots of the Android version of Choicla. The study participants had the chance to choose between n=15 different exam modes which differ, first, in terms of the share of practical exercises (PE) and theoretical questions (TQ) and second, in terms of the share of specific topics. For example, PE (2xSC, 2xORM) denotes an exam mode that includes only practical exercises (i.e., no theoretical questions) related to the topics of State Charts (SC) and object relational mapping (ORM).

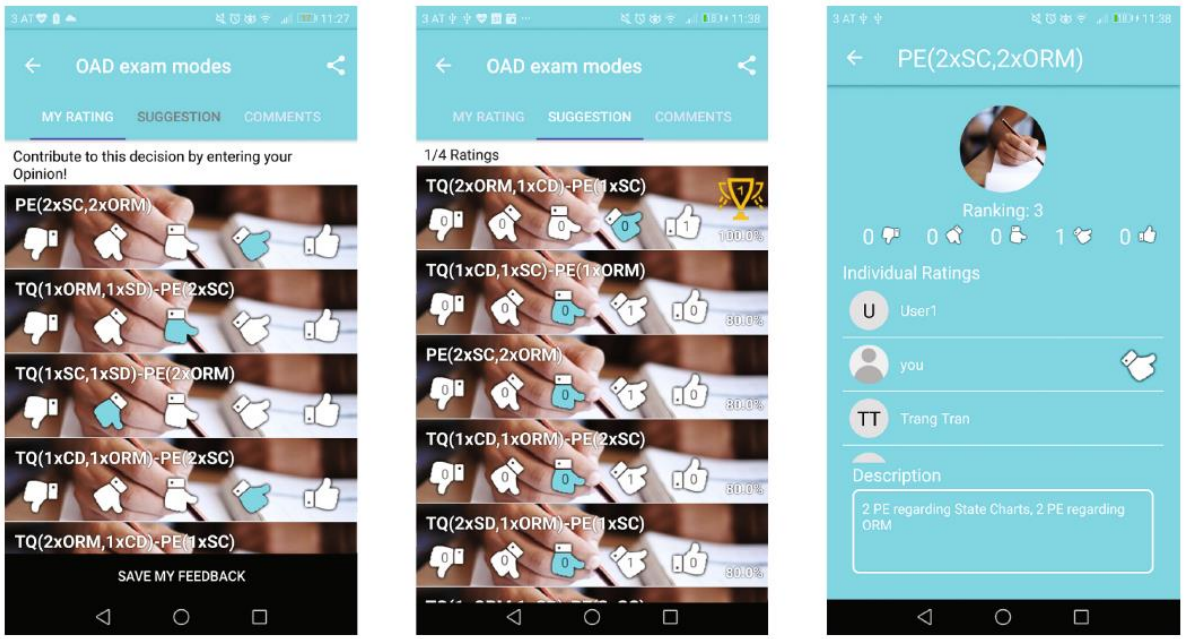


Figure 28: Choicla group decision support environment [SFL+2015] (Android version). Recommendations (suggestions) are determined on the basis of the different aggregation functions introduced in previous section.

Within the scope of the study, each group member had to define his/her own preferences with regard to the available exam modes (see Figure 29). Before a group member did not define his/her initial preferences, there was no possibility to see the preferences of the other group members (the underlying idea is to avoid anchoring biases that result from a too early preference disclosure [SFL+2015]). On the basis of a short introductory statement before starting the decision process, study participants were encouraged to take a look at the group recommendations (tab suggestion) which was done by 91.41% of the participants at least once. An overview of the assignment of individual groups to specific Choicla versions that differ in terms of the used aggregation mechanism is depicted in Table 19.

Aggregation function	#groups	#participants
GD^{min}	17	92
GD^{*}	12	69
GD^{max}	16	95
Total	45	256

Table 19: Assignment of preference aggregation mechanisms to groups.

The hypotheses analyzed within the scope of the empirical study were the following:

- H1: preference aggregation mechanisms with a higher resulting recommendation diversity increase the degree of knowledge exchange within a group. High-diversity recommendations can act as an anchor [SFL+2015] and can also induce the feeling of dissent and a corresponding need to resolve the dissent. Increased knowledge exchange between group members can increase the probability of identifying the knowledge relevant for taking an optimal decision. Examples of the different types of knowledge exchanged within the scope of a group decision processes are shown in Table 20. This table summarizes the total amount of messages exchanged between group members that can be assigned to one of the categories of content-related, preference-related, and recommendation-related. In the following we characterize these categories on the basis of related examples.
    - Content-related. A student only took a look at exercises related to a specific topic, e.g., Object-relational Mapping (ORM) and asks for further information regarding alternative topics. Another group member points out that there are only a few slides with very simple and understandable examples on the topic of state charts which are also very useful in industrial contexts.
    - Preference-related. A group member mentions that he/she prefers to include exercises related to the Unified Process (UP) compared to State Charts (SC).
    - Recommendation-related. A participant does not like the group recommendation and he/she wants to discuss assignment topics that are more acceptable for the group as a whole.

Information units exchanged between group members were analyzed manually with regard to the three mentioned categories. In the context of recommendation-related

information exchange, we evaluated the valence for recommendation-related comments, i.e., how positive/negative a recommendation was perceived.

- H2: a higher degree of knowledge exchange provides more flexibility to change initial preferences afterwards. If more decision-relevant knowledge is exchanged between the members of a group, the amount of global decision-relevant knowledge is increased. This improves the individual capabilities of taking into account additional decision alternatives. Increased knowledge exchange between group members plays a key role to overcome a discussion bias (group discussions tend to be dominated by information group members already knew before the discussion [GS2003]).

Hypothesis H1 can be confirmed, i.e., the degree of exchanged decision-relevant knowledge depends on the chosen aggregation function. The higher the diversity, the higher the number of exchanged decision-relevant knowledge (see Table 20). The number of the given comments for maximum group distance is highest (total number of comments for GD_{max} = 278, GD_{avg} = 92, GD_{min} = 49). Furthermore, also the overall time invested in taking a decision increases with the diversity of recommendations (see Table 21).

We can also confirm hypothesis H2. The flexibility of the group members to change their initial preference increases with the higher amount of knowledge exchange. Table 22 confirms hypothesis H2 which provides an overview of the changes of initial ratings depending on the supported aggregation mechanisms. The average degree of opinion adaptation of groups is highest with GD_{max}.

Function	Content		Preference		Recommendation		
	#comments	avg.	#comments	avg.	#comments	avg.	valence
GD_{min}	22	1.29	0	0	27	1.59	+4.2
GD_{avg}	31	2.58	26	2.16	35	2.92	+0.9
GD_{max}	79	4.93	91	5.69	108	6.75	−4.4

Table 20: Content-, preference-, recommendation-related comments
(#comments, avg. #comments per group, and valence [-5 .. +5] (for recommendation-related comments)).

Function	Duration (h)		proc. time (min)	
	avg.	std.dev.	avg.	std.dev.
GD_{min}	71.06	13.05	210.71	20.19
GD_{avg}	85.64	26.58	234.56	17.67
GD_{max}	101.18	19.48	278.46	16.74

Table 21: Duration (endtime-starttime) and processing time
(total time of system interaction) invested per group for decision task completion (i.e., rating of alternatives).

Function	Degree of rating adaptation
GD_{min}	0.67
GD_{avg}	1.32
GD_{max}	2.46

Table 22: Changes of initial ratings depending on included aggregation mechanism (difference between original rating and final rating).

Summarizing, the higher the diversity of preference aggregation, the higher the amount of knowledge exchange between group members. Thus, diverse group recommendations can help to increase the probability of identifying optimal solutions due to a higher probability of exchanging knowledge relevant for the optimal decision [MS2010]. This can be considered as an important aspect to be taken into account by online decision support environments.

OpenReq Contributions

The major contributions of this section are the following. We show that recommendation diversity can help to increase the degree of information exchange in group decision making. Furthermore, a higher degree of information exchange also correlates with a higher preparedness to adapt initially articulated preferences. In contrast to the mainstream of recommender systems research, we focused on the application of recommendation technologies to improve decision processes per-se. The results of our empirical study show that recommendation diversity has an impact on the frequency of information exchange between group members – the higher the diversity, the more information is exchanged between group members. Furthermore, recommendations with a higher diversity can lead to an increased preparedness of changing initially defined preferences, i.e., these recommendations can be regarded as a mechanism to counteract discussion biases. We regard this work as a contribution to establish recommender systems as a core mechanism to improve the quality of group decision processes.

A major focus will be the analysis of further aggregation mechanisms relevant in social choice scenarios [FBS+2018]. Of major relevance in this context is to answer the question on the optimal degree of recommendation diversity that helps to optimize the parameters degree of information exchange and perceived recommendation quality. Tables 23 and 24 show that the satisfaction with group recommendations decreases with a higher diversity.

Function	GD_{min}	GD_{avg}	GD_{max}
Diversity	0.84	1.38	2.23

Table 23:Diversity of group recommendations.

Function	Very satisfied	Satisfied	Average	Unsatisfied	Very unsatisfied
GD_{min}	67	12	9	2	2
GD_{avg}	17	14	12	14	12
GD_{max}	2	1	15	25	52

Table 24: Satisfaction with group recommendations.

6. Explanations for Groups

Explanation approaches focus on single users, i.e., do not have to take into account group decision immanent aspects. Explanations dedicated for groups can have further goals such as fairness (taking into account as far as possible the preferences of all group members), consensus (group members agree on the taken decision), and optimality (a group takes an optimal or nearly optimal decision -- In contrast to single-user decision making, the exchange of decision-relevant knowledge among group members has to be fostered). An important aspect in this context is that explanations are able to show how the interests of individual group members are taken into account which is not relevant in the context of single user recommender systems. Understanding the ”process behind” enables group members to evaluate the appropriateness of the way their preferences have to been taken into account by the group recommender system. Similar to explanations for single users, explanations for groups are shaped by the underlying recommendation algorithms. Explanations can be defined in a group context, for example, first, ”groups that like feature x also like feature y”, second, ”since the group rates the requirement / feature x very high, we also recommend requirement / feature y which is related to requirement / feature x”, and third, ”since the maximum effort for this requirement accepted by group members is 500 (defined by Paul) and the minimum accepted effort is 360 (defined by Joe), we recommend 460 which represents the average of all given effort ratings”.

These examples show that the chosen preference aggregation approach has an impact on the explanation style. While aggregated predictions allow to include information about the individual preferences of group members (e.g., one group member specified the lowest maximum price of 500) and thus support explanation goals such as fairness and consensus, aggregated models based approaches restrict explanations to the group level (e.g., groups that like x also likey). More advanced (hybrid) explanations can also be formulated in group recommendation scenarios, for example, ”since all group members prefer requirement X to be in Release Y, we recommend requirement X rather than requirement Z to be in Release Y. It is only a little bit more effortful but has a higher usability which is important for group member Joe. Similar groups also preferred similar requirements”.

An example of an explanation in a situation where no solution could found is: ”no release with a total effort value below 250 could be found. Therefore we recommend to include at least requirement y with an effort value of 200 since this requirement is the most important one for all group members.” Finally, the following example shows how to take into account a group’s social reality, for example, in terms of ”tactful” explanations: ”Although your preference for requirement Y is not very high, your close friend Peter thinks it is an excellent choice”. This example explanation is formulated on the level of aggregated predictions and also takes into account social relationships among group members (e.g., neighborhoods in a social network). On the level of aggregated models, an explanation can be formulated as follows: ”Some group members think that it is an excellent choice” (assuming the existence of at least some aggregated categorization of preferences such as number of likes). Taking into account the individual preferences of group members helps to increase mutual awareness among group members and thus counteract the natural tendency to focus on one’s own favorite alternatives. An approach to explain the consequences of a given recommendation is introduced by Jameson

et al. [JA2004] where emotions of individual group members with regard to a recommendation are visualized in terms of animated characters.

In the following, we sketch ways in which explanations in single user recommendation scenarios can be adapted to groups. Following the idea of categorizing explanation types along the different recommendation approaches, we discuss explanations for groups in the context of collaborative filtering, content-based filtering, constraint-based recommendation, and critiquing-based recommendation.

Consensus in Group Decisions

Situations can occur where the preferences of individual group members become inconsistent [FAT2016]. In the context of group recommendation scenarios, consensus is defined in terms of disagreement of individual group members regarding item evaluations (ratings) [FBS+2018]. To provide a basis for establishing consensus, such situations have to be explained and also visualized [FBS+2018]. In this context, diagnosis methods can help to determine repair actions that propose changes to the current set of requirements (preferences) in such a way that a recommendation can be identified. Such repairs are able to take into account the individual preferences of group members. The potential of aggregation functions (see, e.g., Table 25) to foster consensus in group decision making is discussed in [SMS2012].

Concepts to take into account consensus in group decision making are also presented in [FBS+2018]. In scenarios such as software requirements engineering [FZN+2012], there often occur misconceptions regarding the evaluation/selection of a specific requirement, for example, misconceptions regarding the assignment of a requirement to a software release. An explanation in such contexts indicates possible changes of requirements (assignments) that help to restore consistency. In group-based settings, such repair-related explanations help to understand the constraints of other group members and to decide in which way own requirements should be adapted.

aggregation strategy	description	recommendation
Additive Utilitarian (ADD) [C]	sum of item-specific evaluations	$\underset{(t \in I)}{argmax}(\Sigma_{u \in G} eval(u,t))$
Approval Voting (APP) [M]	number of item-specific evaluations above an approval threshold	$\underset{(t \in I)}{argmax}(\lvert\{u \in G : eval(u,t) \geq threshold\}\rvert)$
Average (AVG) [C]	average of item-specific evaluations	$\underset{(t \in I)}{argmax}(\frac{\Sigma_{u \in G} eval(u,t)}{\lvert G \rvert})$
Average without Misery (AVM) [C]	average of item-specific evaluations (if all evaluations are above a defined threshold)	$\underset{(t \in I : \nexists u \in G \lvert eval(u,t) \leq threshold)}{argmax}(\frac{\Sigma_{u \in G} rating(u,t)}{\lvert G \rvert})$
Borda Count (BRC) [M]	sum of item-specific scores derived from item ranking	$\underset{(t \in I)}{argmax}(\Sigma_{u \in G} score(u,t))$
Copeland Rule (COP) [M]	number wins (w) - number losses (l) in pair-wise evaluation comparison	$\underset{(t \in I)}{argmax}(\lvert w(t, I - \{t\})\rvert - \lvert l(t, I - \{t\})\rvert)$
Fairness (FAI) [C]	item ranking as if individuals ($u \in G$) choose them one after the other	$\underset{(t \in I)}{argmax}(eval(u,t))$ [in each iteration]
Least Misery (LMS) [B]	minimum item-specific evaluation	$\underset{(t \in I)}{argmax}(mineval(t))$
Majority Voting (MAJ) [B]	majority of evaluation values per item	$\underset{(t \in I)}{argmax}(majorityeval(t))$
Most Pleasure (MPL) [B]	maximum item-specific evaluation	$\underset{(t \in I)}{argmax}(maxeval(t))$
Most Respected Person (MRP) [B]	item-evaluations of most respected user	$\underset{(t \in I)}{argmax}(eval(u_{mrp}, t))$
Multiplicative (MUL) [C]	multiplication of item-specific evaluations	$\underset{(t \in I)}{argmax}(\Pi_{u \in G} eval(u,t))$
Plurality Voting (PLU) [M]	item with the highest #votes from $u \in G$	$\underset{(t \in I)}{argmax}(votings(t))$ [in each iteration]

Table 25: Basic aggregation functions for group recommendation [12, 38, 43, 44, 56].
Tie breaking rules such as random selection can to be applied. M, C, and B denote the aggregation categories majority-based, consensus-based, and borderline; u represents a user (group member), G a group, t an item, and I a set of items (e.g., requirements / features in a software project).

User-generated Explanations

User-generated explanations are interpreted as textual explanations defined by a group member (typically, the creator of a decision task) to explain, for example, why a specific alternative has been selected. The impact of user-generated explanations in constraint-based group recommendation scenarios was analyzed by Stettinger et. al [SFL+2015]. The creator of a decision task (prioritization decisions in the context of software requirements engineering) had to explain the decision outcome in a verbal fashion. In groups where such explanations were provided, this contributed to an increased satisfaction with the final decision and the perceived degree of group decision support quality [SFL+2015]. User-generated explanations are not limited to constraint- based recommendation, for example, crowdsourcing based approaches are based on the similar idea of collecting explanations directly from users.

Fairness Aspects in Groups

Fair recommendations in group settings can be characterized as recommendations without favoritism or discrimination of specific group members. The perceived importance of fairness

depending on the underlying item domain has been analyzed in [FAT+2017]. An outcome of this study is that in high-involvement item domains (e.g., decisions regarding new cars, financial services, and apartments), the preferred preference aggregation strategies (see Table 25) differ from low-involvement item domains such as restaurants and movies which are often the domains of repeated group decisions (e.g., the same group selects a restaurant for a dinner every three months). Groups tend to apply strategies such as Least Misery (LMS) in high involvement item domains, and prefer Average Voting (AVG) in low-involvement item domains. When recommending, for example, packages to groups, the task is it to recommend a set of items in such a way that individual group members perceive the recommendation as fair [SMP+2017]. One interpretation of fairness stated in [SMP+2017] is that there are at least m items included in the package that a group member likes.

An approach to take into account fairness in repeated group decisions is presented in [QRD2006] where rating predictions are adapted to achieve fairness in future recommendation settings – this adaptation also depends on the personality of a group member, for example, a group member with a strong personality who was treated less favorably last time, will be immediately compensated in the upcoming group decision. A similar interpretation of fairness is introduced in [ST2014] where fairness is defined also in the context of repeated group decisions, i.e., decisions that repeatedly take place within the same or stable groups (groups with a low fluctuation). Fairness in this context is achieved by introducing functions that systematically adapt preference weights, i.e., group members whose preferences were disregarded recently receive higher preference weights in upcoming decisions. For example, in the context of repeated decisions (taken by the same group) regarding a restaurant for a dinner, the preferences of some group members are more often taken into account than the preferences of others. In software engineering contexts it is also very common that the group constellation rather stays stable over time (only a small proportion of the group members change) -- most of the group members work in multiple projects together (e.g., follow-up projects). In such scenarios, the preference weights of individual group members can be adapted [ST2014] (see Formulae 19 - 20).

Formula 20 provides a fairness estimate per user u_i in terms of the share of the number of supported preferences in relation to the number of defined preferences. The lower the value the less the preferences of a user (group member of group G) have been taken into account and the lower the corresponding degree of fairness with regard to ui. Formula 19 reflects an approach to increase fairness in upcoming recommendation sessions: if the fairness (Formula 20) in previous sessions was lower than average, a corresponding upgrade of user-specific importance weights takes place for each dimension. For an example of adapted weights see Table 26.

$$imp'(u_i, dim_j) = imp(u_i, dim_j) \times (1 + (\frac{\Sigma_{u \in G} fair(u)}{|G|} - fair(u_i)))$$

Formula 19.

$$fair(u_i) = \frac{\#supported preferences(u_i)}{\#group\ decisions}$$

Formula 20.

user	importance (imp)			fairness (fair)	adapted importance (imp')		
	dim_1	dim_2	dim_3		dim_1	dim_2	dim_3
u_1	0.3	0.3	0.4	4/8=0.5	0.375	0.375	0.5
u_2	0.5	0.4	0.1	6/8=0.75	0.5	0.4	0.1
u_3	0.3	0.2	0.5	8/8=1.0	0.225	0.15	0.375

Table 26: An example of an adaptation of individual users' weights to take into account fairness.
In this example, the importance weights of user u1 have been increased, the weights of u2 remain the same, and the weights of user u3 have been decreased (the preferences of u3 have been favored in previous decisions).

OpenReq Contributions

We provided an overview of explanation concepts that help groups to better understand item recommendations. As has been pointed out in the pioneering work by Jameson and Smyth [JS2007], explanations play a crucial role in group recommendation scenarios. In order to support a more in-depth understanding of how explanations can be determined, we provided a couple of working examples in terms of verbal explanations.

Although extensively analyzed in the context of single-user recommendations (see, e.g., [TI2009]), the generation of explanations for groups entails a couple of open research issues. Specifically, aspects of group dynamics have to be analyzed with regard to their role in generating explanations. For example, consensus, fairness, and privacy are major aspects – the related research question is how to define explanations that best help to achieve these goals. Some initial approaches exist to explain the application of aggregation functions in group recommendation contexts (see, e.g., [NSN+2012]), however, a more in-depth integration of social choice theories into the generation of explanations has to be performed (also on the algorithmic level, for example, in the context of group-based configuration scenarios). In this context, the integration of information about personality and emotion into explanations has to be analyzed – initial related work can be found, for example, in [QSR+2017] where social factors in groups are taken into account to generate tactful explanations, i.e., explanations that avoid situations which introduce tensions in the group of requirements engineers where group members think "my opinion was taken into account less due to my personality" or "someone does not trust me".

Mechanisms that help to increase the quality of group decision processes have to be investigated [KR2012], for example, explanations could also be used to trigger intended behavior in group decision making such as exchange of decision-relevant information among group members [AFS+2017]. Finally, explaining hybrid recommendations [36] and

recommendations generated by matrix factorization (MF) approaches [AN2017] in group recommendation contexts can be considered as issues for future research. Thus, explanations for groups is a highly relevant research area with a couple of open issues for future research.

7. Interfaces between WP4 and other Components

OpenReq Live is regarded as central component supporting group decision processes. This component exploits the services of other OpenReq components to support core Requirements Engineering tasks such as prioritization, group-based release planning, and "requirements intelligence" functionalities such as analyzing Twitter channels regarding the popularity/relevance of specific requirements. OpenReq Live integrates functionalities from:

- workpackage 2 (requirements intelligence, e.g., the analysis of Twitter channels or the quality estimation of specific requirements),
- workpackage 3 (e.g., the recommendation of relevant stakeholders),
- workpackage 4 (core group recommendation components), and
- workpackage 5 (dependency detection and diagnosis / reconfiguration).

In order to show how the interfaces between the mentioned components are implemented, we show the interface between OpenReq Live and workpackage 3 (detection of similar requirements) in more detail.

Recommendation of similar requirements

This task consists of the detection and recommendation of similar requirements by analyzing requirements from other (past) projects. From an architectural point of view, the whole recommendation process includes several steps.

Besides some meta-data of requirements (such as explicitly defined skills required to solve the requirement), as main criteria to find relevant requirements, the title and description of requirements are taken into account. Given a detailed description and a title of every requirement, NLP techniques are applied to extract their relevant features. Thereby, the text is split into tokens and noisy data is removed (i.e., data cleaning). Next, stop words such as prepositions or articles, which do not represent valuable information of a requirement, are removed as well. After that, lemmatization is applied to the remaining tokens to reduce the number of tokens which share the same meaning. This way, undesired ambiguity-related issues caused by the same word appearing as plural and singular words, as verbs in different tenses, can be counteracted. Since each remaining token represents a feature, the number of features is reduced as well. This results in a less complex and more flexible recommendation model. Finally, the current NLP pipeline computes the TF-IDF (Term Frequency and Inverse Document Frequency) values of all remaining tokens, and these values are then used as features.

In the next phase, clustering is applied to group similar requirements based on the preprocessed tokens. Moreover, to better tackle the ambiguity of words (i.e., polysemy) appropriate (soft) clustering techniques such as Latent Semantic Analysis (LSA) are exploited. The recommendation system is based on hard-clustering in terms of hierarchical clustering as well as soft clustering in terms of LSA. Similar requirements which lie in the close proximity in the vector/latent space are then considered as candidates to be recommended to the requirements manager by the recommender system. This microservice includes an internal offline training to fit the model.

Sequence Diagrams

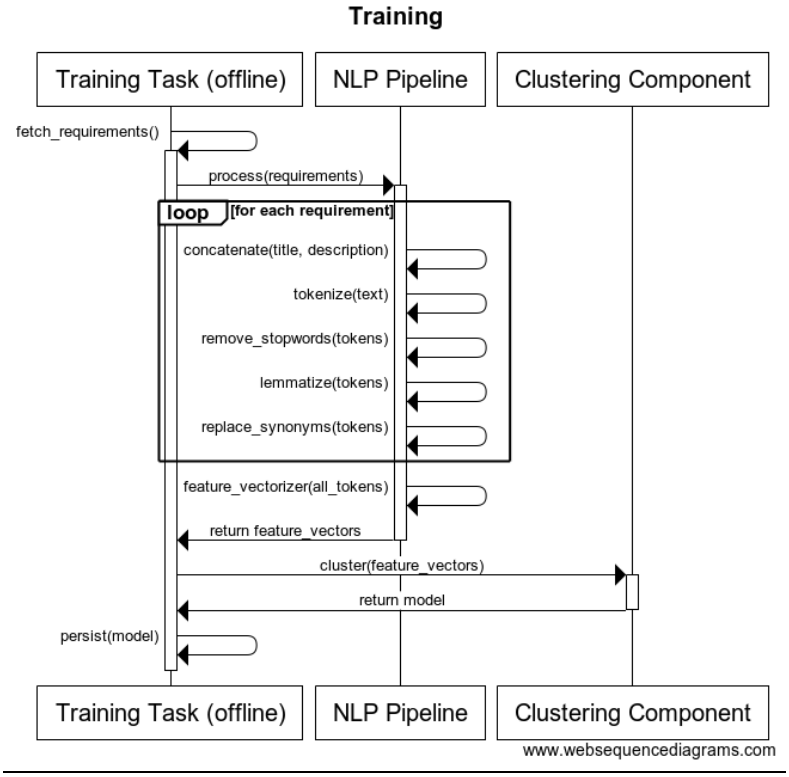


Figure 29: Sequence Diagram for offline training.

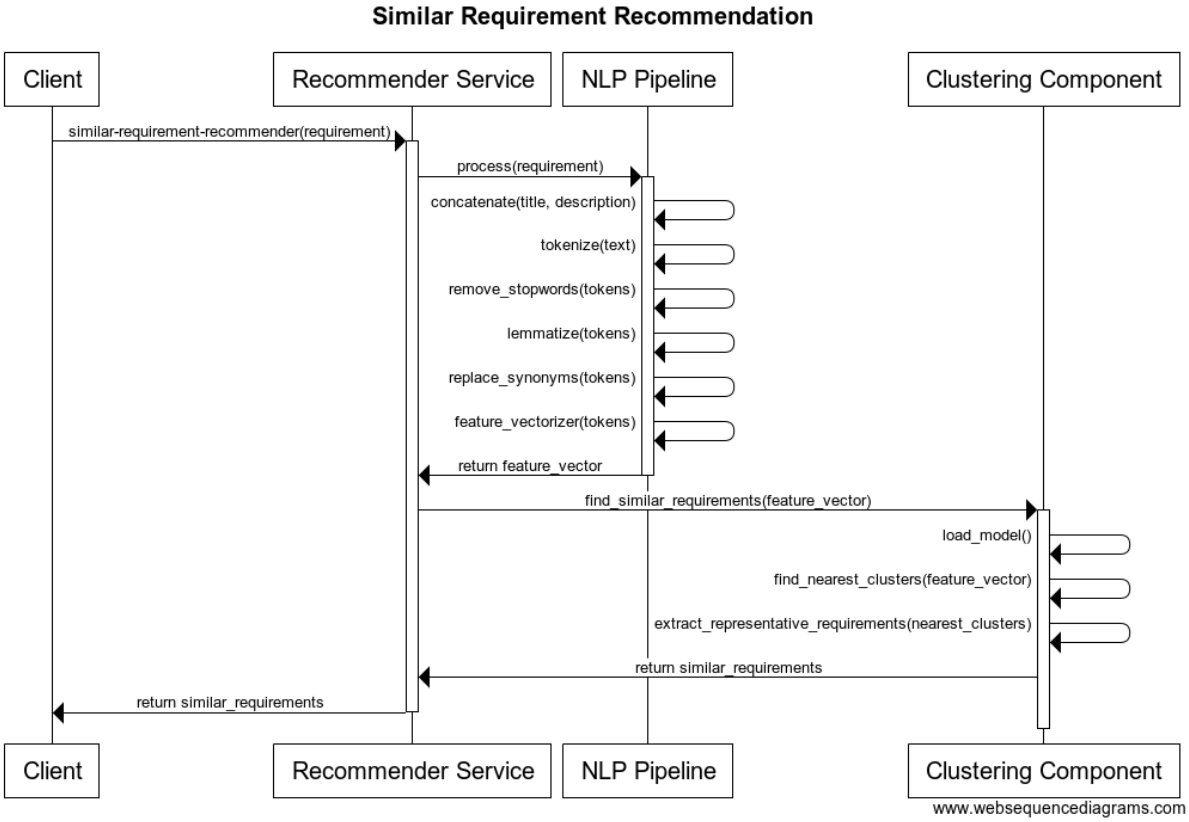


Figure 30: Sequence Diagram for similar requirements detection recommendation service.

Example Usage

URL	/tugraz/similar-requirement-recommender
Method	POST
URL params	None

Data params	{
	"project": {
	"id": "Demo",
	"name": "Demo project",
	"specificRequirements": ["1"] },
	"requirements": [
	{
	"id": 1,

	"name": "Speed Measurement", "text" : "As evaluation after a workout, the average speed must be shown. The following statistics should be displayed: average speed, maximum speed. This requires a time measurement, distance measurement, and a storage unit for storing the data.", “effort”: 2, “created_at”: 2018-06-15 }] }

Return data	{
	"project": {
	"id": "Demo",
	"name": "Demo project",
	"specificRequirements": ["28"] },
	"requirements": [
	{
	"id": "28",
	“name”: “Distance Measurement”
	"text" : "For statistical purposes, a distance measurement is necessary which requires data from a GPS sensor. This data is needed for the evaluation software and therefore stored in memory.",
	“effort”: 5,
	“created_at”: 2018-04-22
	},
	...]

Return explanation	This service returns a list of similar requirements to the given requirement (input requirement). To increase the precision the similarity calculation has access to all requirements stored in the database. The requirements in the list follow the same format as the input requirement.

8. Relevant OpenReq Publications

A. Felfernig, M. Stettinger, M. Atas, R. Samer, J. Nerlich, S. Scholz, J. Tiihonen, and M. Raatikainen. **Towards Utility-based Prioritization of Requirements in Open Source Environments**, 26th IEEE Conference on Requirements Engineering, Banff, Canada, 2018.

M. Atas, T. Tran, A. Felfernig, and R. Samer. **Socially-aware Recommendation for Over-Constrained Problems**, 31st International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, pp. 267-278, Montreal, Canada, 2018.

T. Tran, M. Atas, A. Felfernig, R. Samer, and M. Stettinger. **Investigating Serial Position Effects in Sequential Group Decision Making**, ACM Conference on User Modeling, Adaptation and Personalization, pp. 239-243, Singapore, 2018.

M. Atas, S. Reiterer, A. Felfernig, and T. Tran. **Polarization Effects in Group Decisions**, ACM Conference on User Modeling, Adaptation and Personalization, Singapore, pp. 305-310, 2018.

A. Felfernig, L. Boratto, M. Stettinger, and M. Tkalcic, **Group Recommender Systems**, Springer, March 2018.

A. Felfernig. R. Walter, J. Galindo, D. Benavides, M. Atas, S. Polat-Erdeniz, and S. Reiterer. **Anytime Diagnosis for Reconfiguration**, Journal of Intelligent Information Systems (JIIS), 2018.

M. Atas, A. Felfernig, M. Stettinger, and TNT. Tran. **Beyond Item Recommendation: Using Recommendations to Stimulate Information Exchange in Group Decisions**, 9th International Conference on Social Informatics, Oxford, UK, pp. 368-377, 2017.

S. Polat Erdeniz, A. Felfernig, M. Atas, TNT. Tran, M. Jeran, and M. Stettinger. **Cluster-Specific Heuristics for Constraint Solving**, 30th International Conf. on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2017, Arras, France, pp. 21-30, 2017.

A. Felfernig, M. Stettinger, A. Falkner, M. Atas, X. Franch, and C. Palomares. **OpenReq: Recommender Systems in Requirements Engineering**, RS-BDA'17, pp. 1-4, Graz, Austria, 2017.

A. Felfernig, M. Atas, M. Stettinger, T. Tran, and S. Reiterer. **Group Recommender Applications**, in: Group Recommender Systems - An Introduction, Springer, pp. 75-89, 2018.

A. Felfernig and M. Willemsen. **Handling Preferences**, in: Group Recommender Systems - An Introduction, Springer, pp. 91-103, 2018.

A. Felfernig, N. Tintarev, T. Trang, and M. Stettinger. **Explanations for Groups**, in: Group Recommender Systems - An Introduction, Springer, pp. 105-126, 2018.

A. Felfernig, M. Atas, R. Samer, M. Stettinger, T. Tran, and S. Reiterer. **Further Choice Scenarios**, in: Group Recommender Systems - An Introduction, Springer, pp. 129-144, 2018.

A. Felfernig, M. Atas, M. Stettinger, T. Tran, and G. Leitner. **Biases in Group Decisions**, in: Group Recommender Systems - An Introduction, Springer, pp. 145-155, 2018.

M. Tkalcic, A. Delic, and A. Felfernig. **Personality, Emotions, and Group Dynamics**, in: Group Recommender Systems - An Introduction, Springer, pp. 157-167, 2018.

9. References

[AN2017] B. Abdollahi and O. Nasraoui. Using Explainability for Constrained Matrix Factorization. In*11th ACM Conference on Recommender Systems*, pages 79–83, Como, Italy, 2017.

[AB2013] M. Alenezi and S. Banitaan. Bug reports prioritization: Which features and classifier to use? In *12th International Conference on Machine Learning and Applications*, pages 112–116, 2013.

[AR1998] D. Arnott. A Taxonomy of Decision Biases. In *Technical Report, Monash University*, pages 1–48, Caulfield East, Victoria, Australia, 1998.

[ASI+2014] P. Achimugu, A. Selamat, R. Ibrahim, and M. Mahrin. A systematic literature review of software requirements prioritization research. *Information and Software Technology*, 56(6):568–585, 2014.

[AFF+2017] D. Ameller, C. Farre, X. Franch, D. Valerio, and A. Cassarino. Towards continuous software release planning. In *24th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 402–406, 2017.

[ARF2018] M. Atas, S. Reiterer, A. Felfernig, and T. Tran. Polarization Effects in Group Decisions. In *ACM Conference on User Modeling, Adaptation and Personalization*, pp. 305-310, Singapore, 2018.

[AFS+2017] M. Atas, A. Felfernig, M. Stettinger, and TNT. Tran. Beyond item recommendation: Using recommendations to stimulate knowledge sharing in group decisions. In *9th International Conference on Social Informatics (SocInfo 2017)*, pages 1–10, Oxford, UK, 2017.

[BM2005] M. Bilgic and R.Mooney. Explaining Recommendations: Satisfaction vs. Promotion. In *ACM IUI 2005 Workshop Beyond Personalization*, pages 1–6, San Diego, CA, USA, 2005.

[CDC+2008] C. Castro-Herrera, C. Duan, J. Cleland-Huang, and Bamshad Mobasher, Using data mining and recommender systems to facilitate large-scale, open, and inclusive requirements elicitation processes, 165–168, 2008.

[CCK+2001] O'Connor, M., Cosley, D., Konstan, J., Riedl, J.: PolyLens: a recommender system for groups of users. In: European Conference on Computer-Supported Cooperative Work, pp. 199–218. ACM, 2001.

[CP2012] Chen, L., Pu, P.: CoFeel: emotional social interface in group recommender systems. In: RecSys 2012 Workshop on Interfaces for Recommender Systems, Dublin, Ireland, pp. 48–55, 2012.

[CW2017] L. Chen and F. Wang. Explaining recommendations based on feature sentiments in product reviews. In *ACM IUI 2017*, pp. 17–28. ACM, 2017.

[DR2005] G. Du, M. Richter, G. Ruhe: Identification of Question and Answer Types for an Explanation Component in Software Release Planning. 3rd International Conference on Knowledge Capture, ACM, pp. 193-194, 2005.

[DR2009] G. Du and G. Ruhe. Does explanation improve the acceptance of decision support for product release planning? In *3rd International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 56–68, 2009.

[DY2005] J. Dyer. MAUT - Multi Attribute Utility Theory. In Multiple Criteria Decision Analysis: State of the Art Surveys, volume 78 of Intl. Series in Operations Research & Management Science, pages 265–292. Springer New York, 2005.

[FAT2016] A. Felfernig, M. Atas, T. Tran, and M. Stettinger. Towards group-based configuration. In *International Workshop on Configuration 2016*, pages 69–72, Toulouse, France, 2016.

[FAT+2017] A. Felfernig, M. Atas, T.N.T. Tran, M. Stettinger, S. Erdeniz, and G. Leitner: An analysis of group recommendation heuristics for high- and low-involvement items. In: Benferhat, S., Tabia, K., Ali, M. (eds.) IEA/AIE 2017. LNCS (LNAI), vol. 10350, pp. 335–344. Springer, 2017.

[FBS+2018] A. Felfernig, L. Boratto, M. Stettinger, and M. Tkalcic, Group Recommender Systems, Springer, March 2018.

[FE2014] A. Felfernig: Biases in decision making. In: International Workshop on Decision Making and Recommender Systems 2014, pp. 32–37. CEUR Proceedings, 2014.

[FSZ2012] A. Felfernig, M. Schubert, and C. Zehentner. An Efficient Diagnosis Algorithm for Inconsistent Constraint Sets. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing (AIEDAM)*, 26(1):175– 184, 2012.

[FZN+2012] A. Felfernig, C. Zehentner, G. Ninaus, H. Grabner, W. Maaleij, D. Pagano, L. Weninger, and F. Reinfrank, Group Decision Support for Requirements Negotiation, in *Advances in User Modeling, Springer Verlag*, volume 7138 of *LNCS*, pp. 105–116, 2012.

[FHB+2014] A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen, *Knowledge-based Configuration: From Research to Business Cases*, Elsevier/Morgan Kaufmann Publishers, 1st edn., 2014.

[FMM+2010] A. Felfernig, W. Maalej, M. Mandl, M. Schubert, and F. Ricci. Recommendation and decision technologies for requirements engineering. In*ICSE 2010 Workshop on Recommender Systems in Software Engineering*, pages 1–5, Cape Town, South Africa, 2010.

[FOR2006] D. Forsyth. *Group Dynamics*. Thomson Higher Education, 2006.

[FSA+2018] A. Felfernig, M. Stettinger, M. Atas, R. Samer, J. Nerlich, S. Scholz, J. Tiihonen, and M. Raatikainen. Towards utility-based prioritization of requirements in open source environments. In 26th IEEE Conference on Requirements Engineering, pp. 406-411, Banff, Alberta, Canada, 2018.

[FSA+TR2018] A. Felfernig, M. Stettinger, M. Atas, R. Samer, T. Gruber, B. Peischl, Decision Making in Requirements Engineering: An Overview of the Industrial State-of-Practice, Technical Report, 2018.

[FZN+2012] Felfernig, A., Zehentner, C., Ninaus, G., Grabner, H., Maalej, W., Pagano, D., Weninger, L., Reinfrank, F.: Group decision support for requirements negotiation. In: Ardissono, L., Kuflik, T. (eds.) UMAP 2011. LNCS, vol. 7138, pp. 105–116. Springer, Heidelberg, 2011.

[FIS+2007] A. Felfernig, K. Isak, K. Szabo, and P. Zachar, The VITA Financial Services Sales Support Environment, in *AAAI/IAAI 2007*, pp. 1692– 1699, Vancouver, Canada, 2007.

[FZ2011] G. Friedrich and M. Zanker. A taxonomy for generating explanations in recommender systems. *AI Magazine*, 32(3):90–98, 2011.

[GKV2009] E. Gansner, Y. Hu, S. Kobourov, and C. Volinsky. Putting recommendations on the map: visualizing clusters and relations. In *ACM Conference on Recommender Systems*, pages 345– 348, New York, USA, 2009.

[GS2003] T. Greitemeyer and S. Schulz-Hardt, Preference-consistent evaluation of information in the hidden profile paradigm: Beyond group-level explanations for the dominance of shared information in group decisions, *Journal of Personality & Soc Psychology 84(2)*, 332–339, 2003.

[HKR2000] J.Herlocker, J.Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *ACM Conference on Computer Supported Cooperative Work*, pages 241–250. ACM, 2000.

[JA2004] A. Jameson: More than the sum of its members: challenges for group recommender systems. In: International Working Conference on Advanced Visual Interfaces, AVI 2004, Gallipoli (Lecce), Italy, pp. 48–54. ACM, 2004.

[JM2014] T. Johann and W. Maalej. Liquid democracy for a sustainable and scalable participation in requirements engineering, 2014.

[JS2007] A. Jameson and B. Smyth, 'The adaptive web', chapter Recommendation to Groups, 596–627, Springer-Verlag, Berlin, Heidelberg, 2007.

[JU2004] U. Junker. QUICKXPLAIN: Preferred Explanations and Relaxations for Over-Constrained Problems. In *19th National Conference on AI (AAAI04)*, pages 167–172, San Jose, CA, 2004.

[JWF+2015] Jameson, A., Willemsen, M.C., Felfernig, A., Gemmis, M., Lops, P., Semeraro, G., Chen, L.: Human decision making and recommender systems. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook. LNCS (LNAI), pp. 611–648. Springer, Boston, MA, 2015.

[JZF+2010] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender Systems – An Introduction*, Cambridge University Press, 2010.

[KKM1996] N. Kerr, G. Kramer, and R. MacCoun. Bias in judgement: Comparing individuals and groups. *Psychological Review*, 103(4):687–719, 1996.

[KSM+2017] F. Kifetew, A. Susi, D. Mutante, A. Perini, A. Siena, and P. Busetta. Towards multi-decision-maker requirements prioritisation via multi- objective optimisation. In *Forum and Doctoral Consortium Papers Presented at the 29th International Conference on Advanced Information Systems Engineering (CAiSE'17)*, pages 137–144, Essen, Germany, 2017.

[KR2012] J. Konstan and J. Riedl. Recommender systems: from algorithms to user experience. *User Modeling and User-Adapted Interaction (UMUAI)*, 22(1):101–123, 2012.

[KBD2003] D. Kudenko, M. Bauer, and D. Dengler. Group decision making through mediated discussions. In: Brusilovsky, P., Corbett, A., Rosis, F. (eds.) UM 2003. LNCS (LNAI), vol. 2702, pp. 238–247. Springer, Heidelberg, 2003.

[LFL1998] T. Landauer, P. Foltz, and D. Laham, An introduction to latent semantic analysis, 1998.

[LEF1997] D. Leffingwell. Calculating the return on investment from more effective requirements management. *American Programmer*, 10(4):13–16, 1997.

[LQF2010] 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, ACM, 2010.

[LKK2004] L. Lehtola, M. Kauppinen, and S. Kujala. Requirements prioritization challenges in practice. In *International Conference on Product Focused Software Process Improvement*, pages 497–508, 2004.

[MA2004] J. Masthoff, Group modeling: Selecting a sequence of television items to suit a group of viewers, *UMUAI*, 14(1), 37–85, 2004.

[MG2006] Masthoff, J., Gatt, A.: In pursuit of satisfaction and the prevention of embarrassment: affective state in group recommender systems. User Model. User-Adapted Interact. 16(3–4), 281–319, Springer, 2006.

[MC2011] B. Mobasher and J. Clehand-Huang, Recommender systems in requirement engineering, 81–89, 2011.

[MS2010] Mojzisch, A., Schulz-Hardt, S.: Knowing other's preferences degrades the quality of group decisions. J. Personal. Soc. Psychol. 98, 794–808, 2010.

[MSC+2006] McCarthy, K., Salamo, M., Coyle, L., McGinty, L., Smyth, B., Nixon, P.: Group Recommender Systems: A Critiquing based Approach. In: IUI 2006, pp. 267–269. ACM, 2006.

[NFS+2014] Ninaus, G., Felfernig, A., Stettinger, M., Reiterer, S., Leitner, G., Weninger, L., Schanil, W.: IntelliReq: intelligent techniques for software requirements engineering. In: European Conference on AI, Prestigious Applications of Intelligent Systems (PAIS), pp. 1161-1166, 2014.

[NSN+2012] E. Ntoutsi, K. Stefanidis, K. Norvag, and H. Kriegel. Fast group recommendations by applying user clustering. In *ER 2012*, volume 7532 of *LNCS*, pages 126–140, 2012.

[OS2016] J. Osmani. Heuristics and cognitive biases: Can the group decision-making avoid them? *Academic Journal of Interdisciplinary Studies*, 5(3):225–232, 2016.

[PSA2013] A. Perini, A. Susi, and P. Avesani. A machine learning approach to software requirements prioritization. *IEEE Transactions on Software Engineering*, 39(4):445–461, 2013.

[QSR+2017] L. Quijano-Sanchez, C. Sauer, J. Recio-Garcia, and B. Diaz-Agudo. Make it personal: a social explanation system applied to group recommendations. *Expert Systems with Applications*, 76:36–48, 2017.

[RE1987] R. Reiter, A theory of diagnosis from first principles, *AI Journal*,23(1), 57–95, 1987.

[QRD2006] L. Quijano-Sanchez, J. Recio-Garcia, and B. Diaz-Agudo. Social Factors in Group Recommender Systems. *ACM Trans. on Intelligent Systems and Technology*, 4(1):8:1–8:30, 2006.

[SMS2012] M. Salamo, K. McCarthy, and B. Smyth. Generating recommendations for consensus negotiation in group personalization services. *Personal and Ubiquitous Computing*, 16(5):597–610, 2012.

[SV2000] T. Saaty and L. Vargas. *Models, Methods, Concepts & Applications of the Analytic Hierarchy Process*. Springer, 2000.

[SCH2000] K. Schmid, Scoping software product lines, in *Software Product Lines – Experience and Research Directions*, pp. 513–532, 2000.

[SBM+2006] 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. J. Personal. Soc. Psychol. 91, 1080–1093, 2006.

[SMP+2017] D. Serbos, S. Qi, N. Mamoulis, E. Pitoura, and P. Tsaparas. Fairness in package-to-group recommendations. In *WWW'17*, pages 371–379. ACM, 2017.

[SFL+2015] M. Stettinger, A. Felfernig, G. Leitner, and S. Reiterer. Counteracting Anchoring Effects in Group Decision Making. In *23rd Conference on User Modeling, Adaptation, and Personalization (UMAP'15)*, volume LNCS 9146, pages 118–130, Dublin, Ireland, 2015.

[SFL+2015] 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)*, pages 148–157, Atlanta, Georgia, USA, 2015.

[SH2014] C. Sunstein and R. Hastie. *Wiser: Getting Beyond Groupthink to Make Groups Smarter*. Harvard Business Review Press, 2014.

[ST2014] M. Stettinger, Choicla: Towards domain-independent decision support for groups of users, in *8th ACM Conference on Recommender Systems*, pp. 425–428, 2014.

[TI2009] N. Tintarev. *Explaining Recommendations*. University of Aberdeen, 2009.

[TM2012] N. Tintarev and J. Masthoff. Evaluating the Effectiveness of Explanations for Recommender Systems. *User Modeling and User-Adapted Interaction*, 22(4–5):399–439, 2012.

[TOF2016] N. Tintarev, J. O'Donovan, and A. Felfernig. Human interaction with artificial advice givers. *ACM Transactions on Interactive Intelligent Systems*, 6(4):1–10, 2016.

[TLX+2015] Y. Tian, D. Lo, X. Xia, and C. Sun. Automated prediction of bug report priority using multi-factor analysis. *Empirical Software Engineering*, 20(5):1354–1383, 2015.

[TSA1993] E. Tsang, *Foundations of Constraint Satisfaction*, Academic Press, London, 1993.

[VPB+2013] K. Verbert, D. Parra, P. Brusilovsky, and E. Duval. Visualizing recommendations to support exploration, transparency and controllability. In *International Conference on Intelligent User Interfaces (IUI'13)*, pages 351–362, New York, NY, USA, 2013.

[WM2017] G. Williams and A. Mahmoud. Mining Twitter feeds for software user requirements. In *25th International Requirements* Engineering Conference (RE), pages 1–10, 2017.

[WHB2004] G. Wittenbaum, A. Hollingshead, and I. Botero: From cooperative to motivated information sharing in groups: moving beyond the hidden profile paradigm. In: Communication Monographs, pp. 286–310, 2004.

[XJR+2012] J. Xuan, H. Jiang, Z. Ren, and W. Zou. Developer prioritization in bug repositories. In *ICSE 2012*, pages 25–35, Zürich, Switzerland, 2012.

[YA2011] Yaniv, I.: Group diversity and decision quality: amplification and attenuation of the framing effect. Int. J. Forecast. 27, 41–49, 2011.