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

D6.1 OpenReq User Interface Approach

Lead contractor: TUGraz

Author(s): TUGraz, ENG

Submission date: December 2017

Dissemination level: PU

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

Abstract: This deliverable focuses on a description of the "OpenReq User Interface Approach" for the "OpenReq Prototype". The OpenReq Prototype is developed following a "minimal viable product" approach where the feedback of the OpenReq partners and other user communities accessed during OpenReq studies is directly taken into account when prioritizing the functionalities of future releases. The idea of following a "minimal viable product" approach was generated out of discussions with the advisory board in a plenary meeting of OpenReq in December 2017 in Vienna. The overall goal of the OpenReq consortium is to be able to support the prioritization and release planning of OpenReq features for the next releases on the basis of the OpenReq Prototype.

The focus of this document is to show selected user interface designs that help to provide support in different requirements engineering scenarios, for example, the automated extraction of requirements from free-text documents, the planning of upcoming releases, the motivation of stakeholders to more intensively engage into requirements engineering processes, and the indication of issues such as inconsistencies in stakeholder preferences related to the assignment of requirements to releases. A major focus of the User Interface (UI) design is to follow a responsive design style [Bernacki et al. 2016] which allows the application of OpenReq functionalities also in mobile environments. The OpenReq Prototype is based upon the latest web-technologies and will run on a Spring Boot[1] application including the Thymeleaf[2] as well as the Bootstrap framework[3]. A major goal of the "OpenReq Prototype" is to show OpenReq functionalities in an integrated fashion which goes beyond the application in the individual OpenReq trials.

We want to emphasize that the UI elements presented in this deliverable represent a showcase of OpenReq which can serve as a reference design for the trial scenarios that are based on individual UI elements, for example, the Siemens trial UI is based on a Doors[4] integration. Similar as the "minimal viable products" also the designs included in this deliverable will be improved repeatedly depending on the feedback from trial partners, usability studies, and the feedback of OpenReq communities and users engaged in the integration of OpenReq functionalities in the "Open Call". The OpenReq prototype will also serve as a basis for different user studies [Chen and Pu 2012] planned in the context of Work Package 4 (Group Decision Support).

[1] projects.spring.io/spring-boot/
[2] www.thymeleaf.org/
[3] getbootstrap.com
[4] www.ibm.com/us-en/marketplace/rational-doors

Table of Contents

List of Figures

1. THE OPENREQ USER INTERFACE APPROACH

Individual trials in OpenReq will be based on different User Interface (UI) technologies based on the available technical infrastructure. For example, the Siemens trial is based on a IBM Doors integration and therefore restricted by the corresponding UI technology provided by this environment. Comparable UI restrictions exist for the WINDTRE, QT and Vogella user interfaces (QT will develop a JIRA and Vogella an Eclipse plugin). In this OpenReq prototype, we focus on the design of User Interface elements (screens) than can be integrated (in adapted form) within the different OpenReq trial implementations and show the core functionalities of the OpenReq product.

The UI approach documented in this deliverable serves as a collection of reference designs that support OpenReq UI developers (and beyond) in the design of their individual user interfaces. Furthermore, these designs are the basis of the OpenReq prototype user interface which serves as a component that helps to demonstrate in an integrated fashion the application of the developed OpenReq recommendation and decision technologies for requirements engineering. The basic approach in this context will be a development process based on immediate feedback from trial partners and user communities that will help to efficiently develop highly relevant features immediately applicable in different requirements engineering scenarios. The presented UI elements have to be regarded as reference examples, i.e., not all interfaces will be immediately implemented and provided. Much more, selected features will be integrated into a sequence of MVPs that will be iteratively extended to a full-fledged OpenReq requirements engineering UI.

2. DESCRIPTION OF USER INTERFACE SCREENS FOR THE OPENREQ PROTOTYPE

The screen designs follow the idea of the model-view-controller architecture, i.e., a strict separation of user interface, data, and control structures. The UI designs derive from corresponding scenario descriptions (associated with the OpenReq requirements). Each scenario is described by an exemplified interaction sequence between an OpenReq stakeholder and the prototype system. These exemplified interactions are then basis for deriving a corresponding user interface design. Within the scope of this deliverable we will not provide an introduction into scenario modeling techniques but more focus on an explanation of the resulting screen designs. A major focus of the presented UI elements is to allow *responsive design*, i.e., to allow user interfaces that support intuitive interaction processes also in mobile scenarios.

The following figure (Figure 1) illustrates the user interface navigation of the prototype by means of a diagram, similar to an activity diagram in UML. Note that for simplification reasons, not every described screen is addressed in this diagram. The described figure numbers in the diagram indicate the detail screens of this document described below.

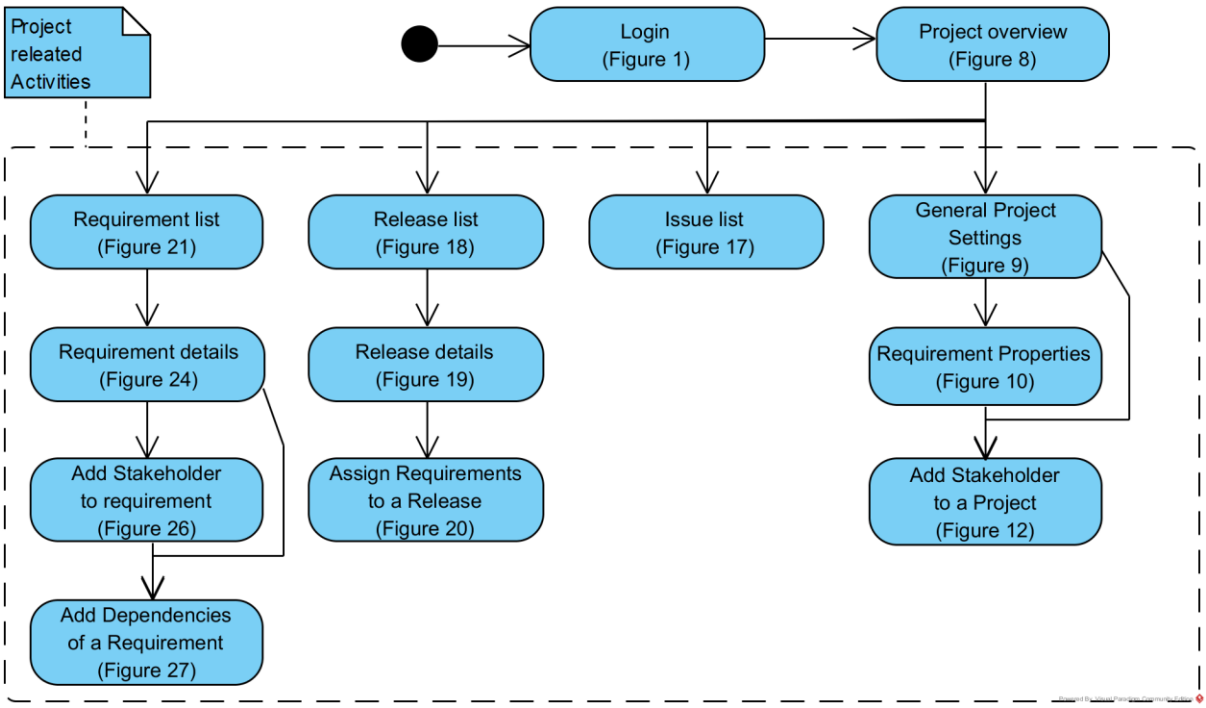

Figure 1: Activity diagram illustrating the user interface navigation

In the following, we discuss major elements of the proposed OpenReq Prototype user interface. As mentioned, these elements serve as a showcase for core functionalities of OpenReq without being too specific on a certain industry scenario. It should be a UI which could be shown to everybody without having to explain the whole background of the scenario. Different UI designs will be presented and discussed within the scope of the following sections.

2.1. User authentication and registration

This section describes the interfaces related to OpenReq user management, such as login of registered users and registration of new users. Also the adopted user profile is described, including the achievements and the scores.

Login Screen. This screen shows a simple login form that lets stakeholders authenticate against the OpenReq server. Furthermore, users can click on “forgot password” link in order to retrieve a new automatically generated password via email.

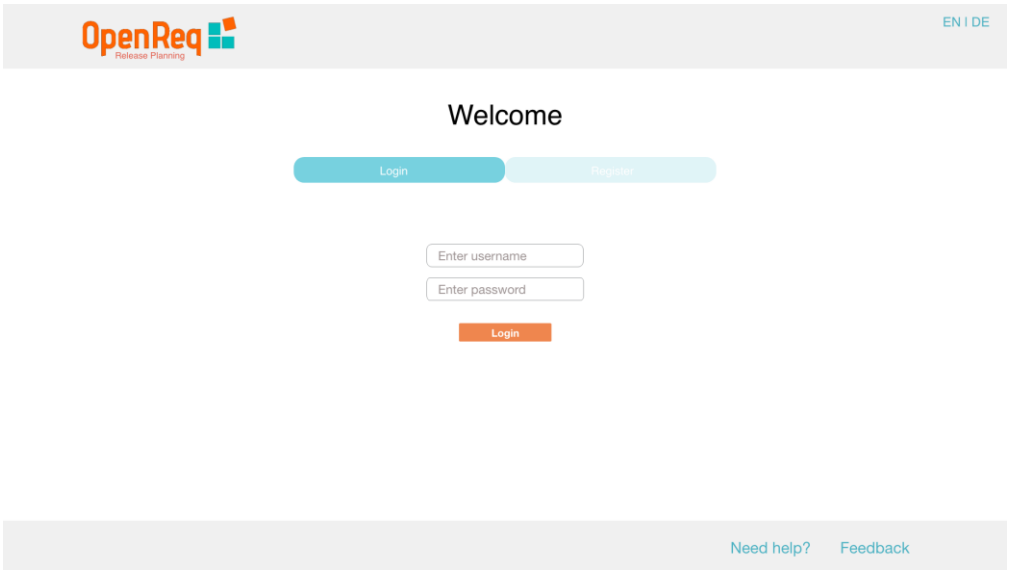

Figure 2: Login Screen.

Registration. This form let’s stakeholder users to create a new account. After filling out all form fields an additional captcha check is performed.

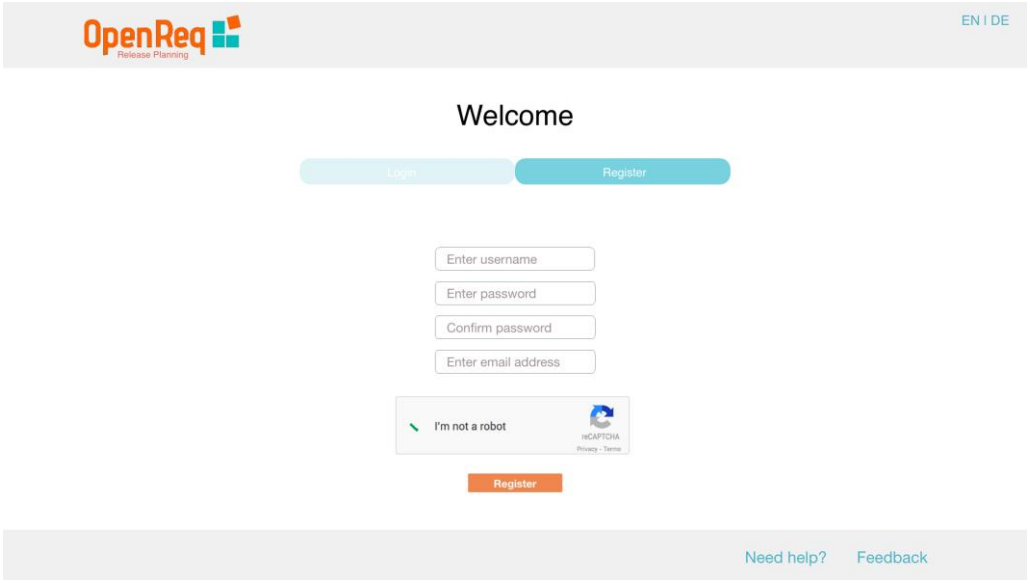

Figure 3: Registration.

Profile Settings. Users can update their personal profile and, for example, let the system know which skills they have (see Figure 4). Moreover, the user's password can be changed (see Figure 5).

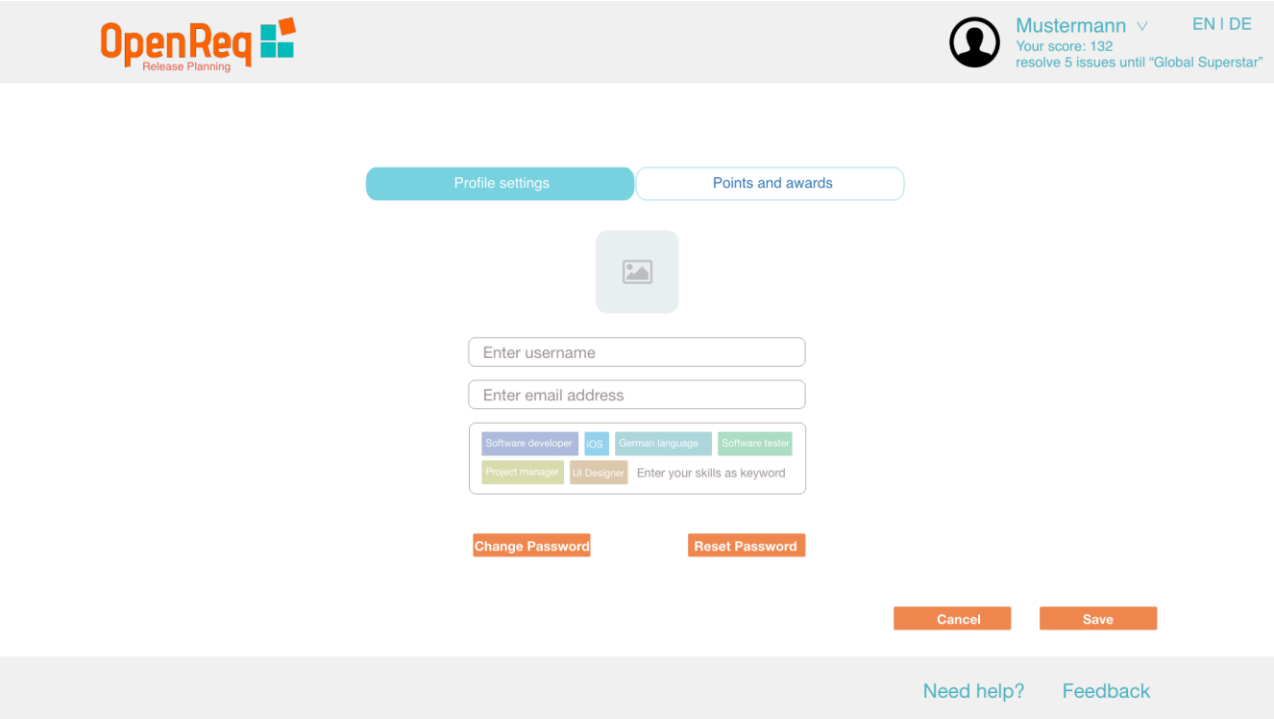

Figure 4: Profile Settings.

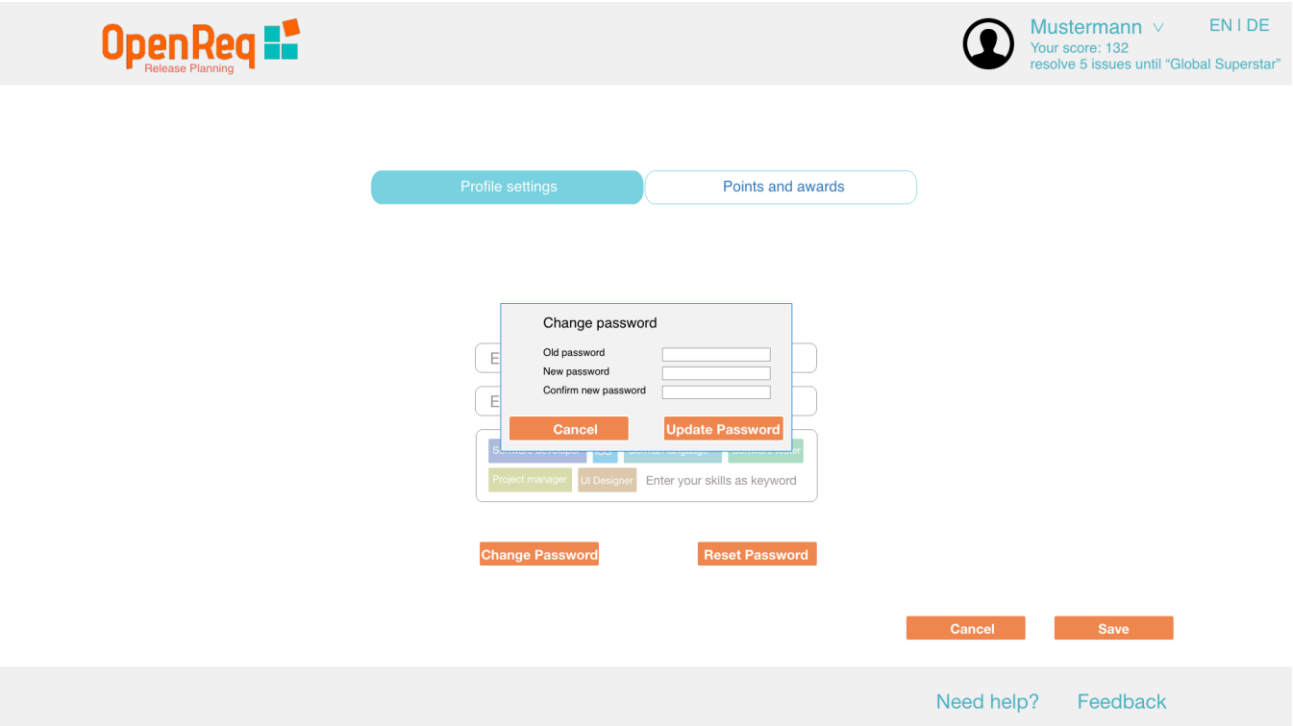

Figure 5: Change of Username and Password.

The following screen depicted in Figure 6 shows all current achievements of the user. It shows a score to motivate the stakeholder to enhance his/her contribution. Moreover, it summarizes the problem solving skills the user has achieved so far. We want to emphasize that the scoring scheme in OpenReq will be adapted conform to the results received from user studies and usability tests. The elements included in the following screen must be regarded as example, however, we are aware of the fact that more in-depth related analyses are needed.

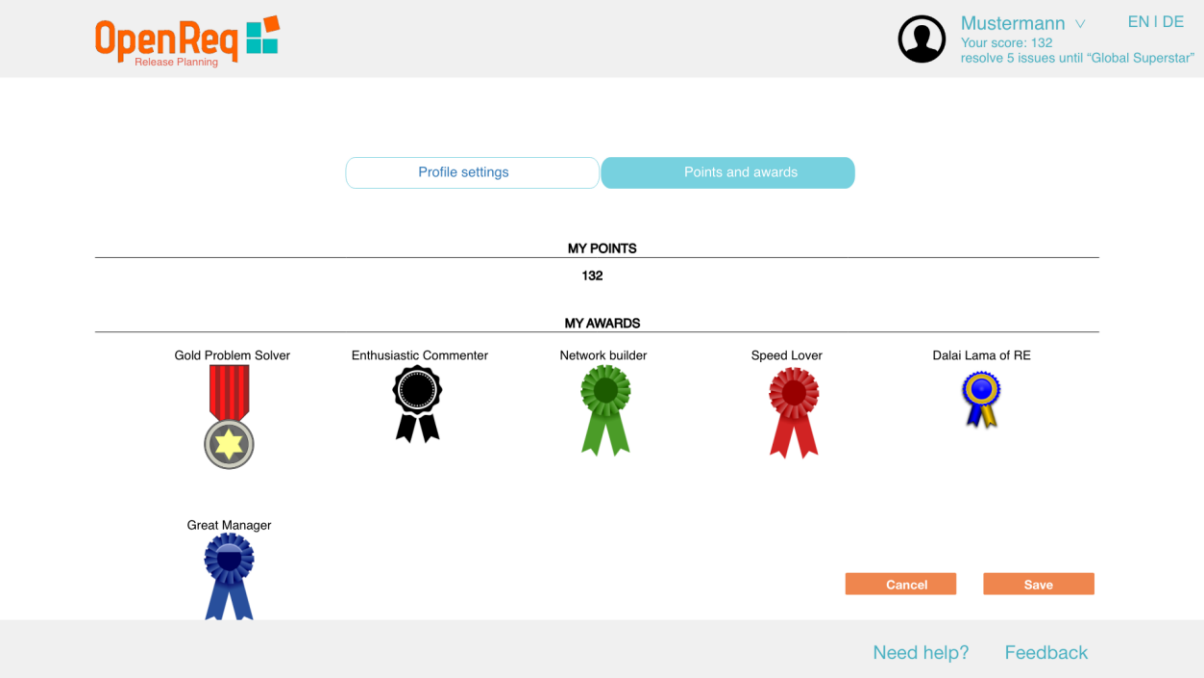

Figure 6: OpenReq user interfaces will be based on the idea of motivation concepts in order to increase user engagements while developing a requirements model.

2.2. Managing projects

A "Project" is the major organizational instance in the OpenReq prototype, i.e., if one wants to start building a new requirements model, a corresponding project has to be created. A project can be regarded as a kind of container that entails a set of requirements and further related information such as stakeholders assigned to the projects and releases in which the software should be developed.

The starting point for OpenReq prototype users is an overview of the currently defined projects. The following screen (see Figure 7) illustrates an overview of current projects in which the user is involved. Each enlisted project is described by a picture, name, number of finished releases, begin- and end-date of the project and user's activity. "***Create project***" button creates a new project and "**Show Archive**" link shows archived projects of the user. The user's activity status in a project can range, e.g., from "Active" to "Superstar" (motivational concepts will be further detailed and developed within the scope of the project). In addition, a user is allowed to delete his/her own projects or leave projects. On the top of the page, the current user's name, achieved score and overall activity is shown.

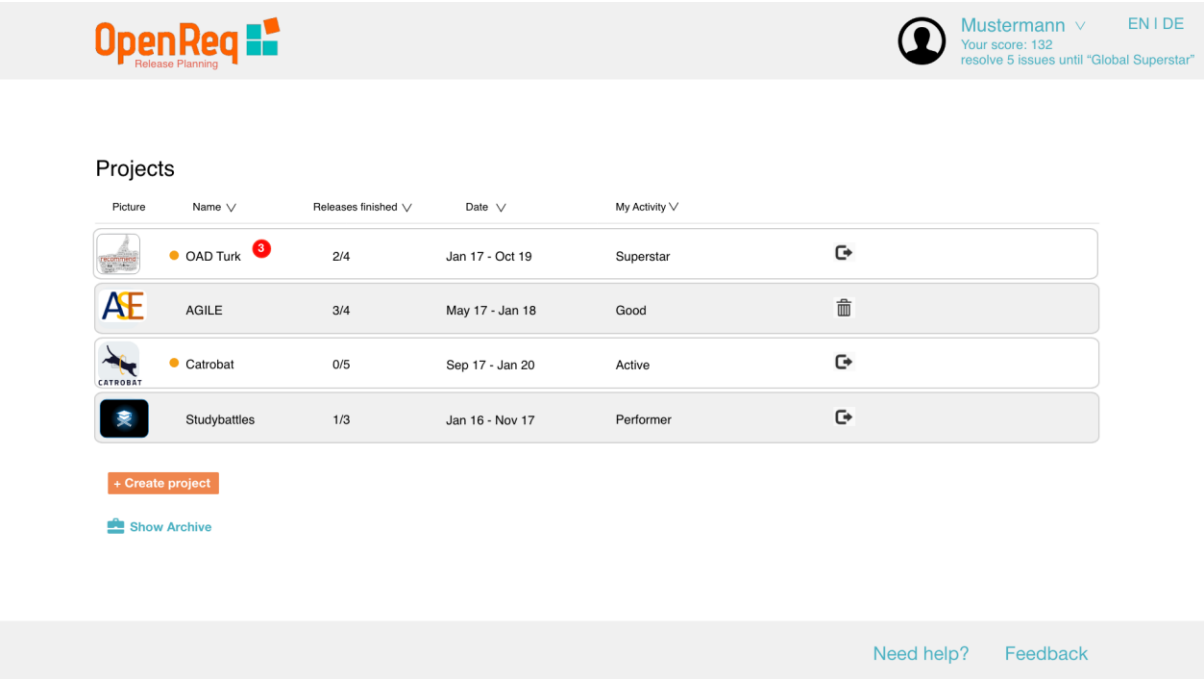

Figure 7: A project list in OpenReq.

Project Details. After selecting a project from the project list, the corresponding project-specific details can be seen. Each project can be characterized by four different dimensions ("tabs"): "Requirements" are the central element that describe on a textual basis the expectations of stakeholders with regards to the intended functionalities (and beyond) of a software. "Releases" can be used to organize requirements around iterative development processes; "Issues" are collecting different types of issues raised within the scope of the requirements engineering process such as contradicting evaluations of requirements properties; and finally, "General" provides general project information such as requirement properties, stakeholders, statistics and attachments.

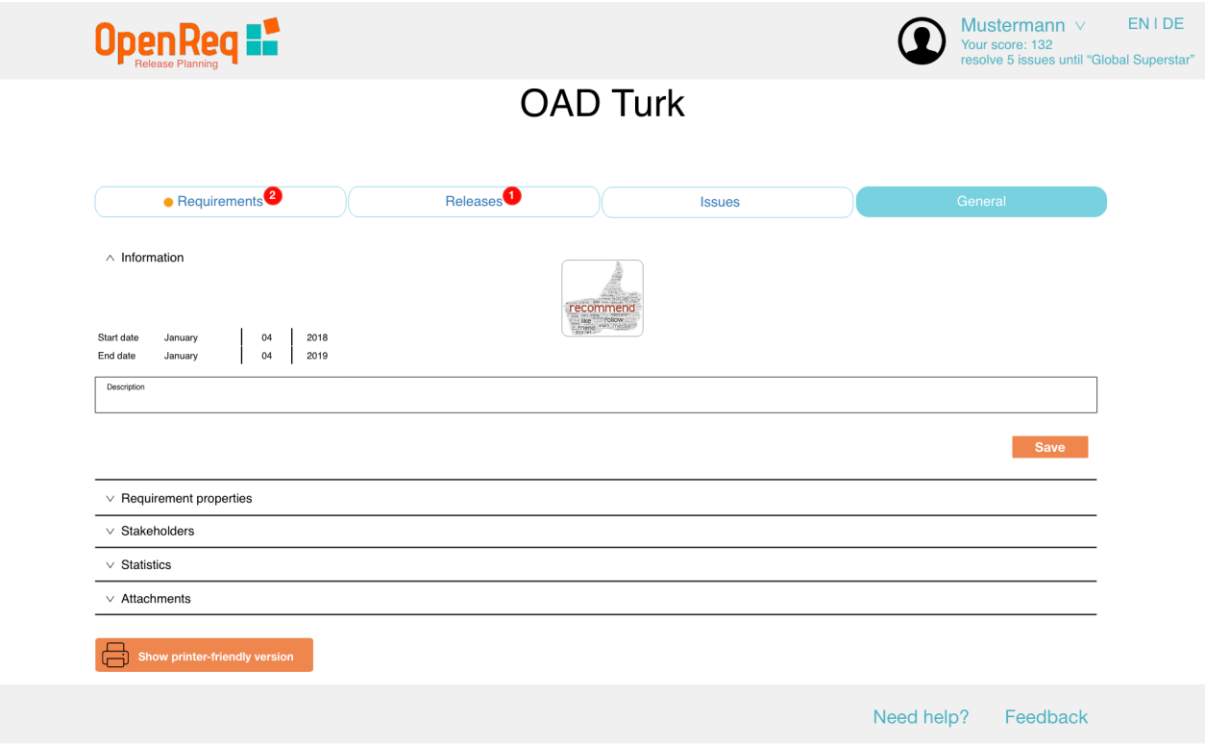

Figure 8: View "inside" an OpenReq project (red circles indicate sections with updates in).

Requirement Properties. This screen (Figure 9) shows the defined "requirement properties" (or more precisely property types) for a specific project. All requirements will be evaluated with regard to these properties.

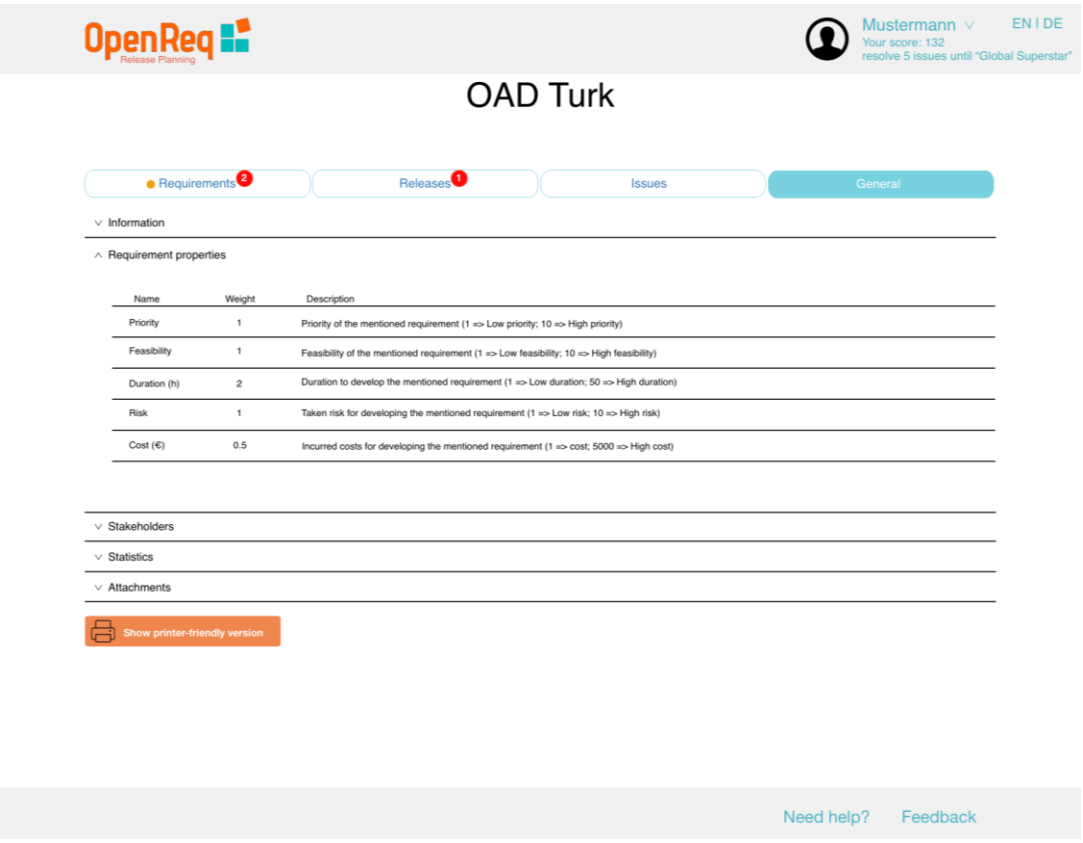

Figure 9: Properties used to evaluate/describe a requirement.

Stakeholders. The screen in Figure 10 depicts a list of stakeholders involved in a project. Stakeholder can be manually assigned to a project. Stakeholders can also be selected with the support of a stakeholder recommendation user interface that supports the search and recommendation of stakeholders, for example, search on the basis of keywords (tags) and recommendations based on information about stakeholder engagements and corporations in previous software projects.

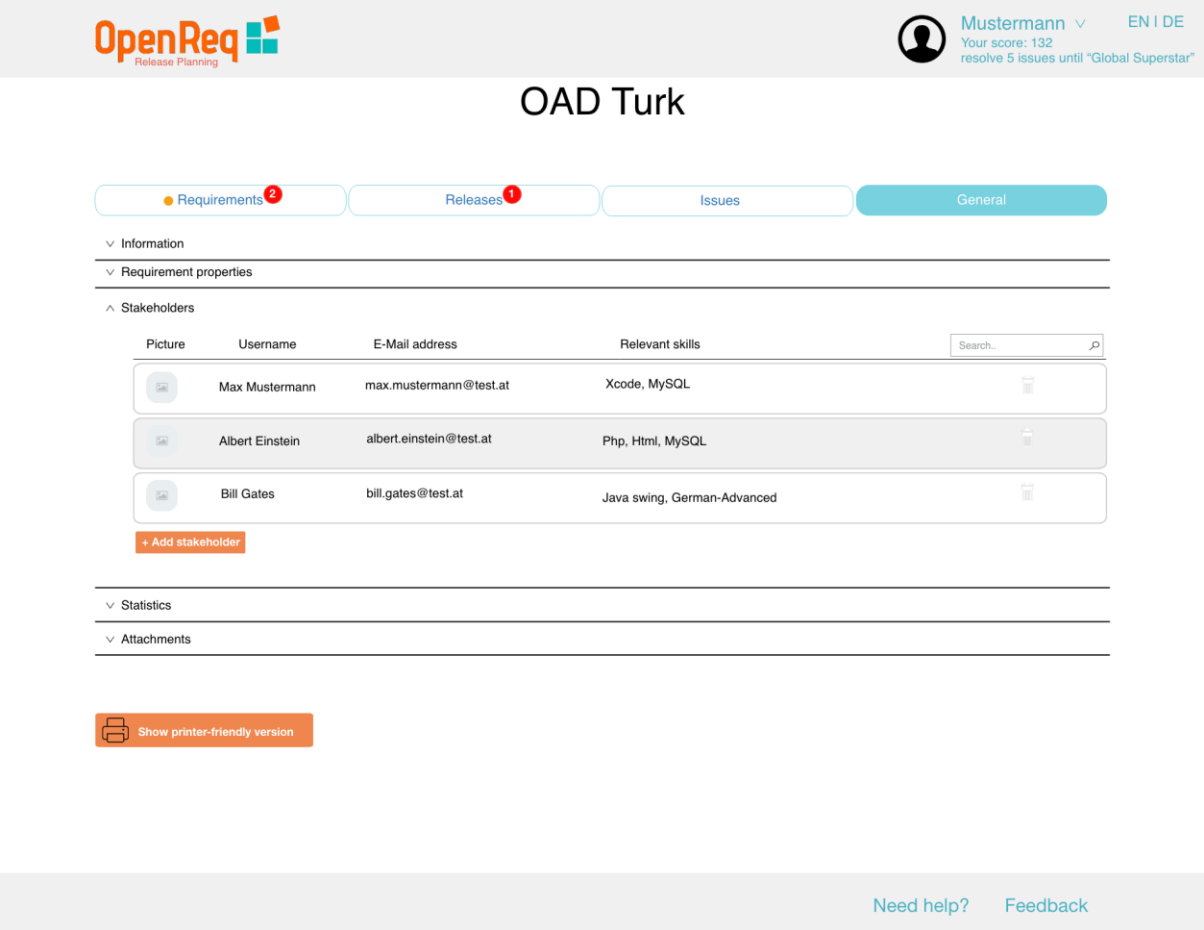

Figure 10: Stakeholders involved in an OpenReq project.

Adding stakeholders. The next screen (see Figure 11) shows how to assign a stakeholder to a project. Stakeholders can be searched by name, skills or both combinations. Additionally, recommended stakeholders are shown and can as well be selected and assigned to the new OpenReq requirements engineering project.

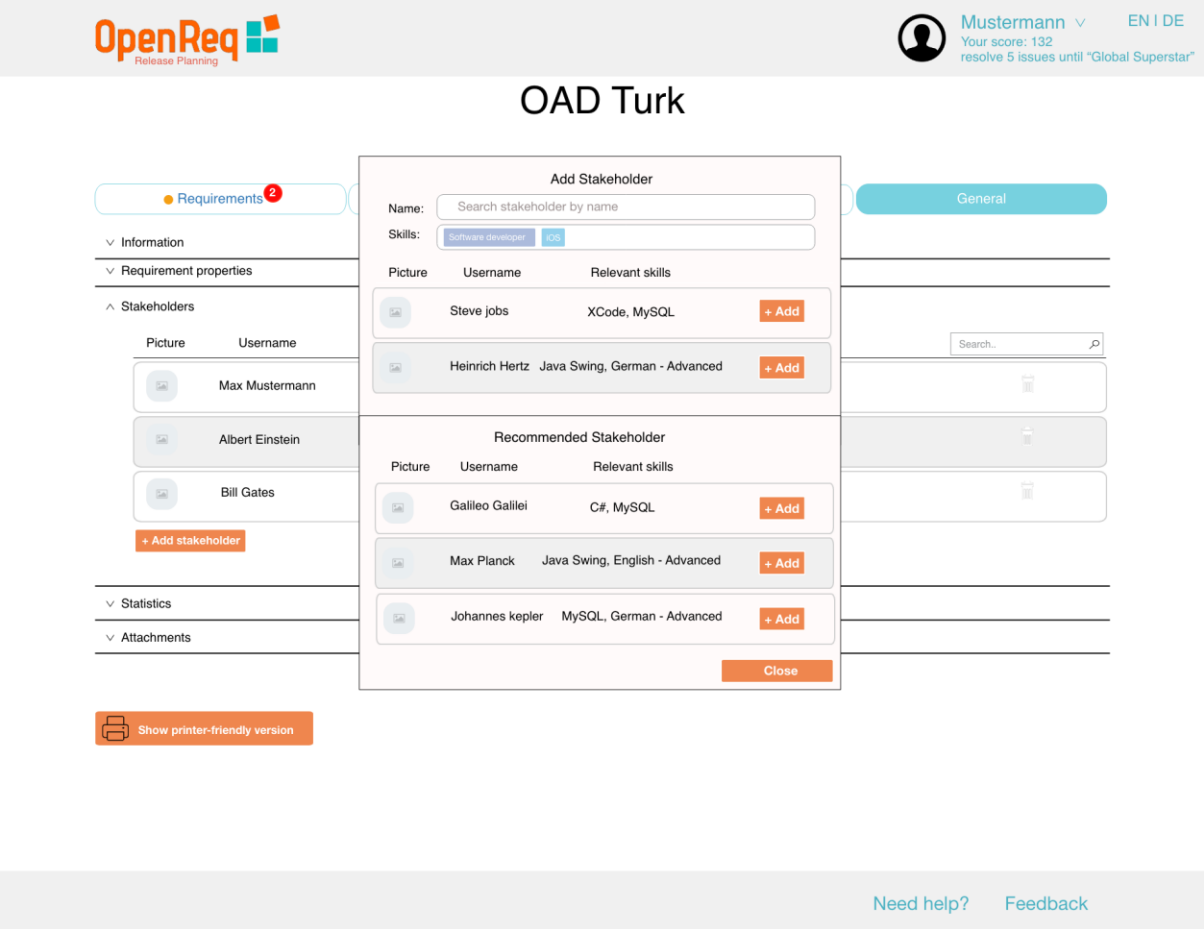

Figure 11: Adding stakeholders.

Statistics. For analysis purposes, statistics are offered by the OpenReq user interface. For example, the following (very basic) statistics can be included.

- "Optimization Chart" which shows a comparison between the optimal and actual time consumption of a project per release (see Figure 12).
- Level of "Disagreement" with regards to the evaluation of requirements (see Figure 13)
- Overview of the assignment of requirements to releases (see Figure 14)
- "Release overview", i.e., an overview of the defined releases (see Figure 15)

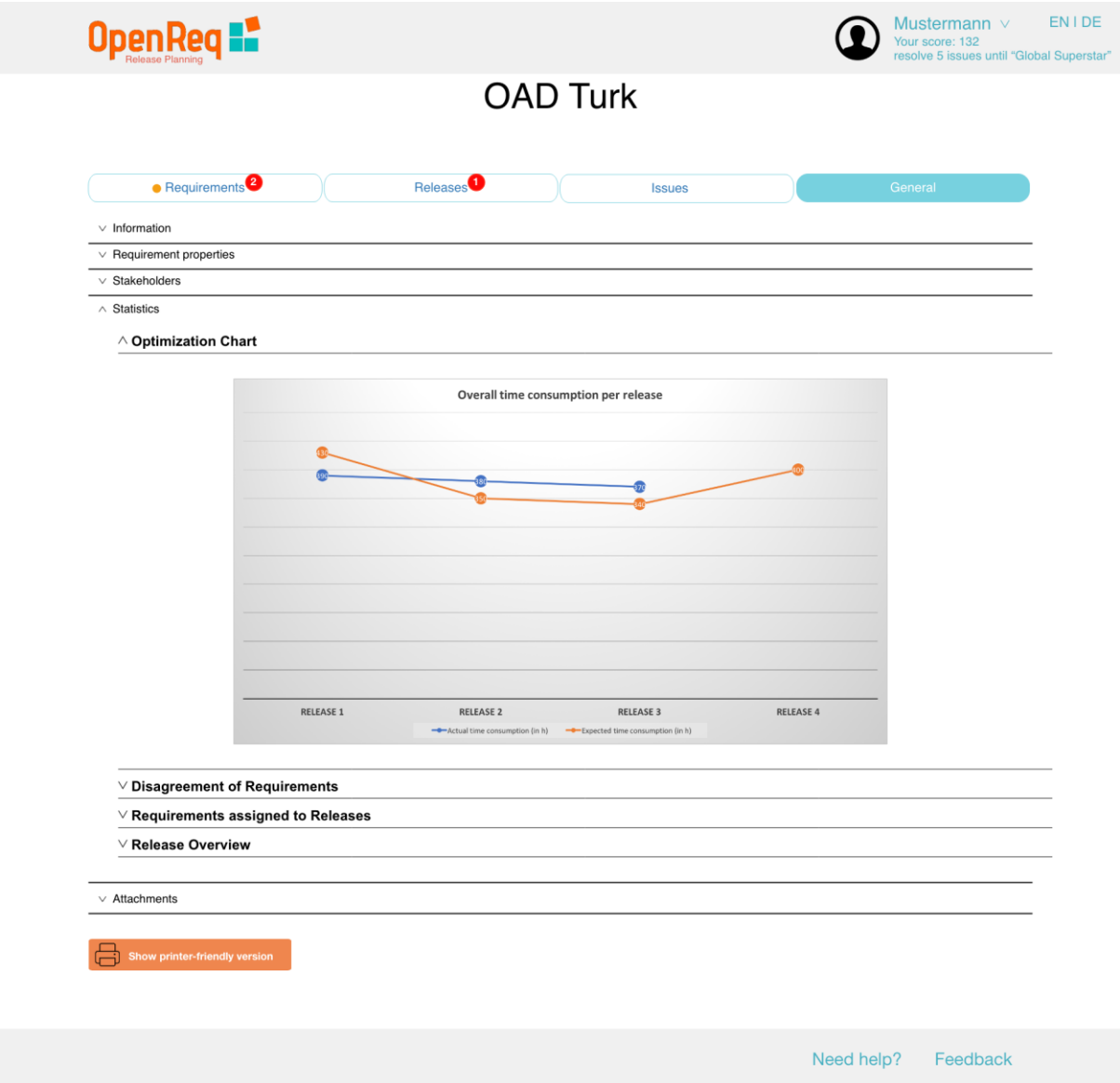

Figure 12: Optimization chart-comparison of possible optimal release plans and current one.

The level of disagreement with regards to the evaluation of different requirement property values can be shown as depicted in Figure 13. Such a "disagreement chart" can help to analyze the individual stakeholder evaluations of requirements with regards to the defined requirement properties such as "Priority", "Feasibility", "Duration", "Risk", and "Cost". These properties/dimensions can be used to perform a group-based (stakeholder-based) utility analysis within a given set of candidate requirements [Felfernig et al. 2018, Winterfeldt and Edwards 1986].

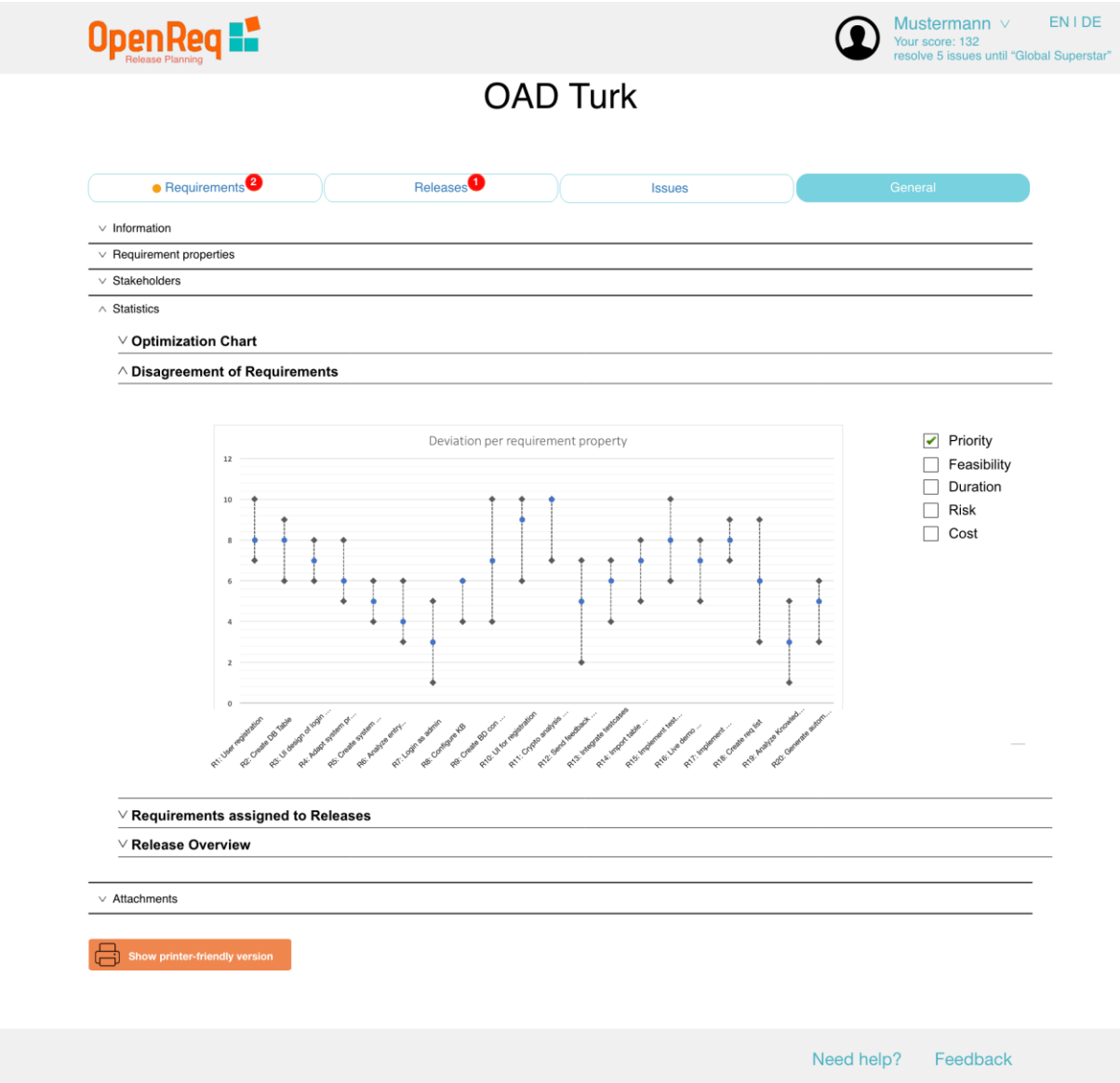

Figure 13: Visualizing disagreements.

Figure 14 provides a very basic overview of the number of requirements assigned to individual releases (and corresponding effort estimation as duration information). More complex visualizations will be included in future versions of the OpenReq user interface. For example, visualizations regarding the risk-level of individual releases or indicators regarding the current status of a releases, i.e., how high is the probability that a release will be delivered in time.

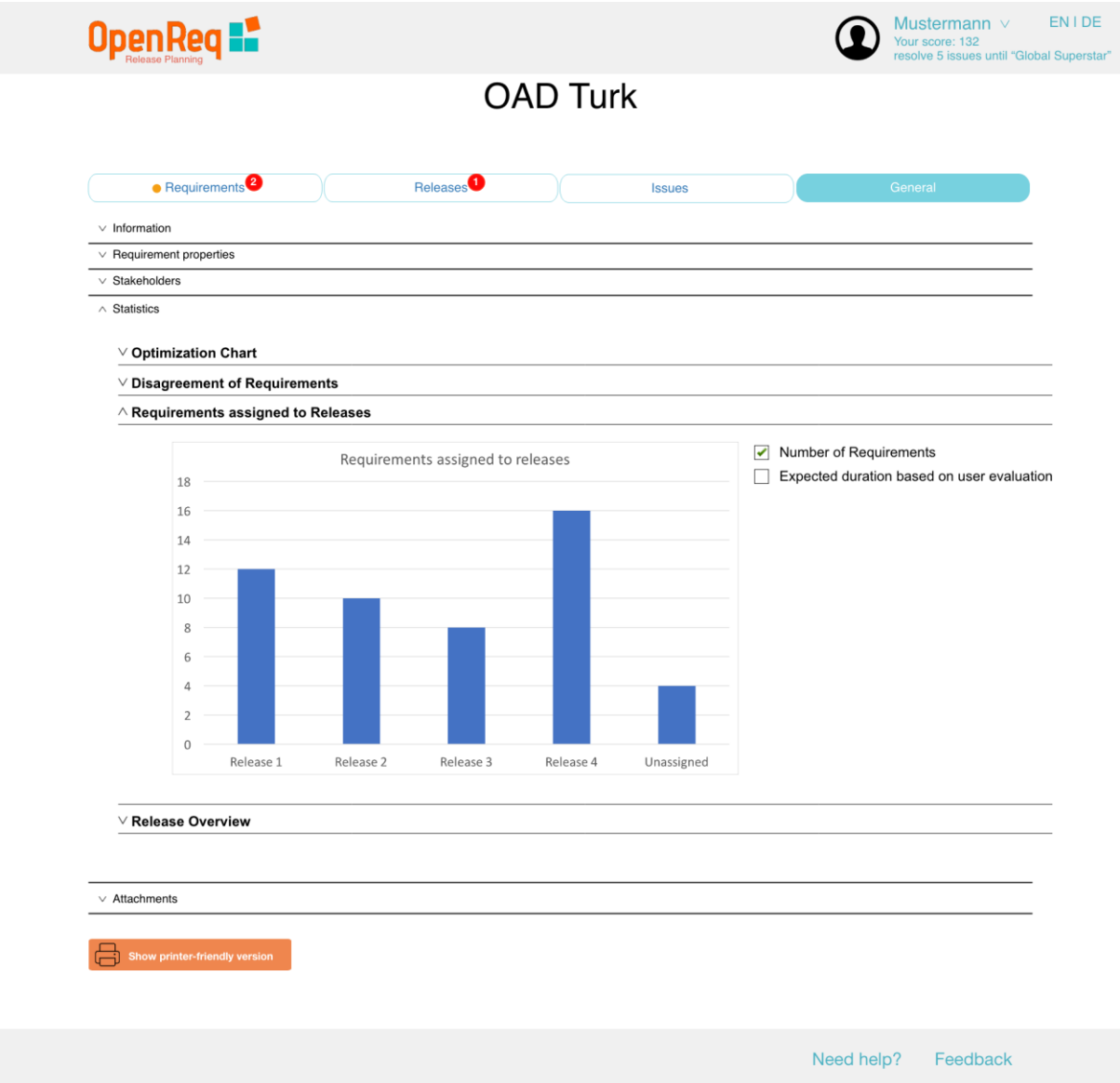

Figure 14: A simple overview of releases.

Figure 15 shows a model of individual releases, their assigned requirements and corresponding defined dependencies [Ninaus et al. 2014]. Note that such dependencies could have also been detected by the dependency detection engine developed within the scope of work package WP5. This screen illustrates the assigned requirements and their dependencies per release. It supports the comparison of multiple releases (with assigned requirements and defined dependencies).

Figure 15: Overview of releases, included requirements, and their dependencies.

Issues: Manually defined or automatically detected issues can be shown to stakeholders. Figure 16 exemplifies the representation of issues, for example:

- ready to vote requirements are available
- the overall capacity of a release is exceeded by the current assignments of requirements to releases.

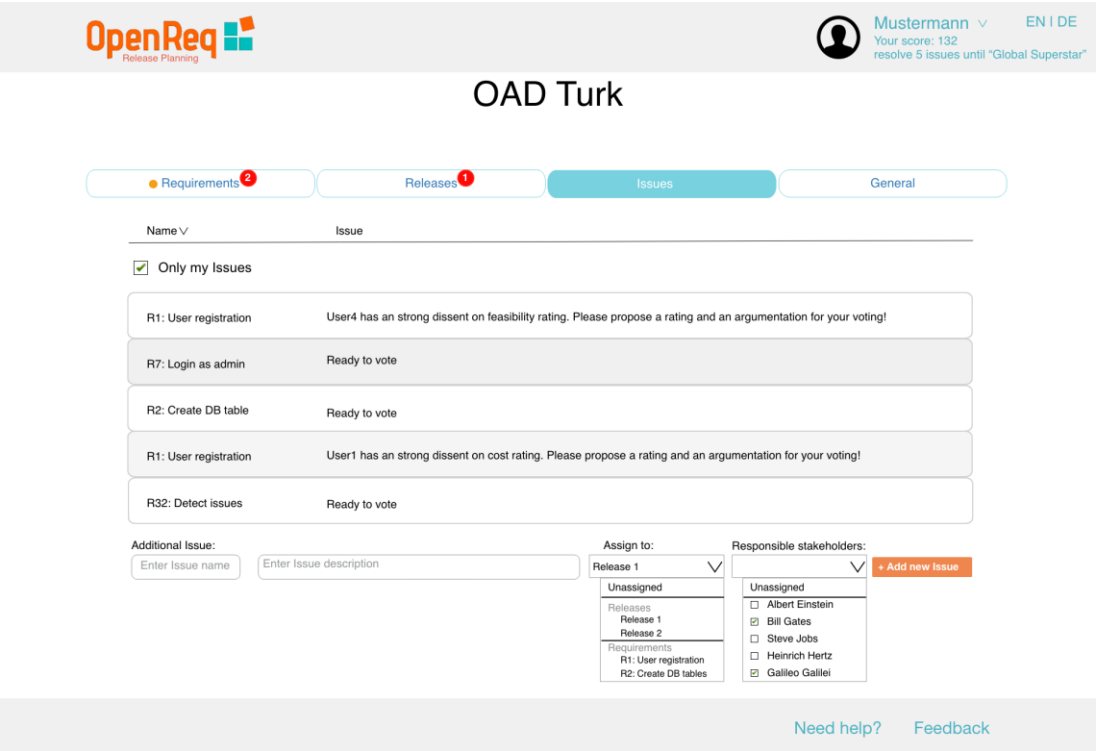

Figure 16: Management of Issues.

Releases: An overview of project releases is provided in Figure 17. Each release can be categorized as *new* (a release which begins in the future and has no requirements assigned), *planned* (Requirements are assigned to a release), *completed* (the release is already deployed/finished), or *rejected* (rejected releases by the project stakeholders). Additionally, begin- and end-date and capacity consumption of a release can be seen.

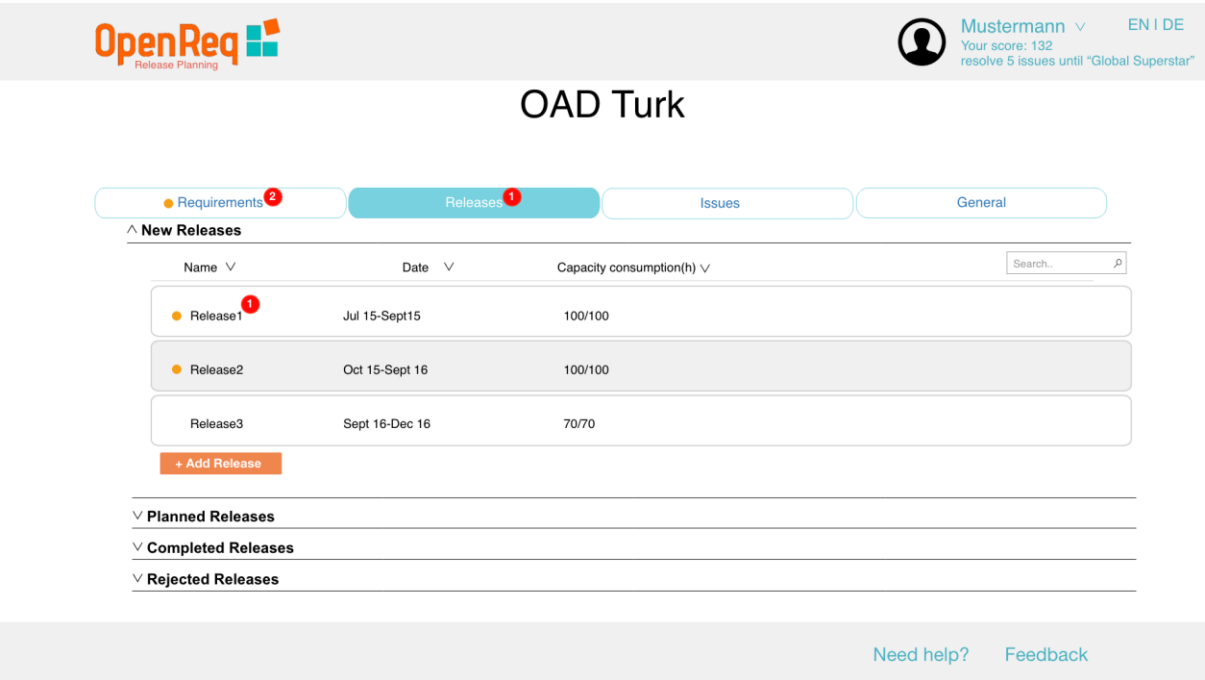

Figure 17: Overview of Releases.

Release details. The details of a release are depicted in Figure 18. Each release will be presented in two tabs:

- **General**: Shows general information of the release, such as start date, end date, maximum capacity in hours, description and its attachments.
- **Assigned requirements**: Shows the requirements assigned to a release.

Figure 18: Release details.

Assigned requirements. The following screen shows all the assigned requirements of a release. A click on "Close assignment" prevents further assignments of requirements for the specific release.

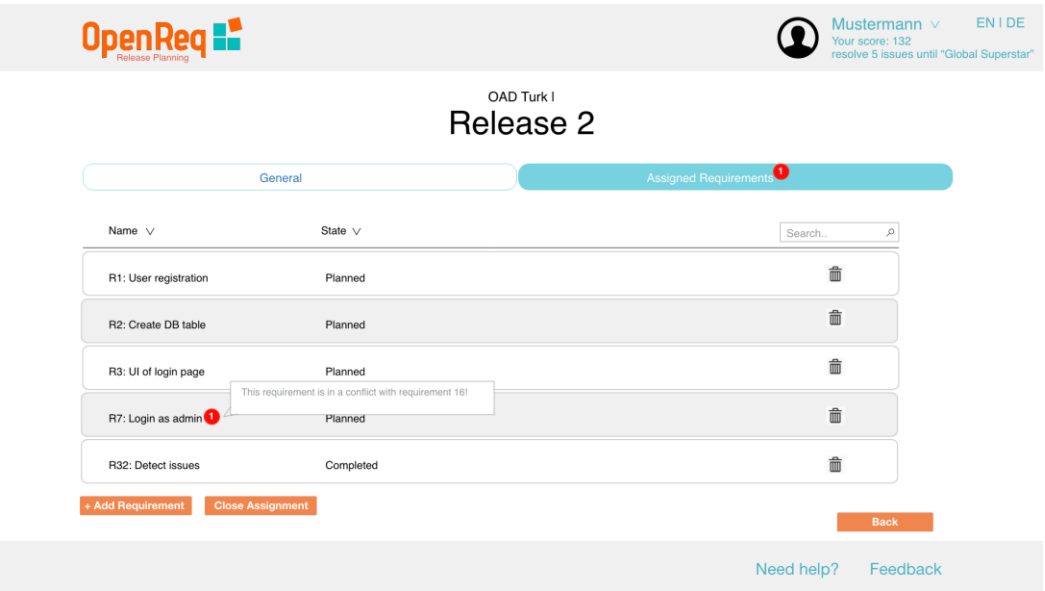

Figure 19: Assigned requirements.

Requirements. Each requirement can be categorized as *new* (a requirement which is not assigned to any release), *recommended* (requirements which are automatically suggested by the system), *planned* (requirements which are assigned to a release) , *completed* (requirements which are assigned to a completed release), or *rejected* (requirements rejected by the project stakeholders).

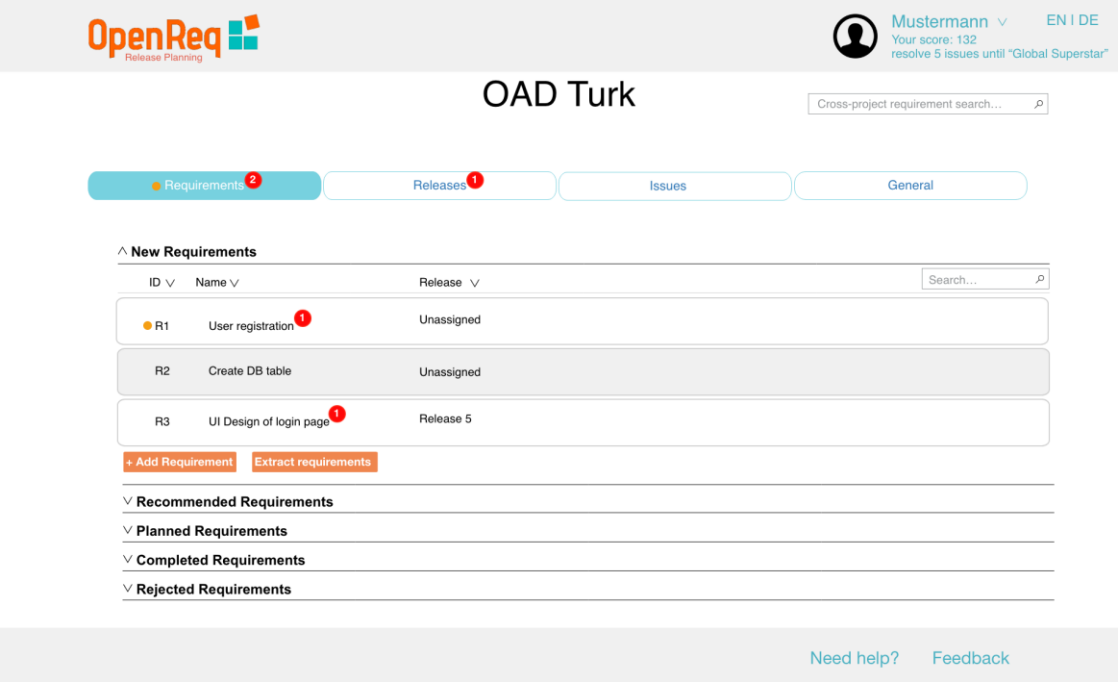

Figure 20: Overview of requirements.

Figure 21 shows a tooltip which explains an issue regarding a requirement. In this case the tooltip says: "*High dissent between stakeholder votings. It is strongly advised to discuss the votings with the other stakeholders. Please provide an argumentation for your rating on feasibility!*".

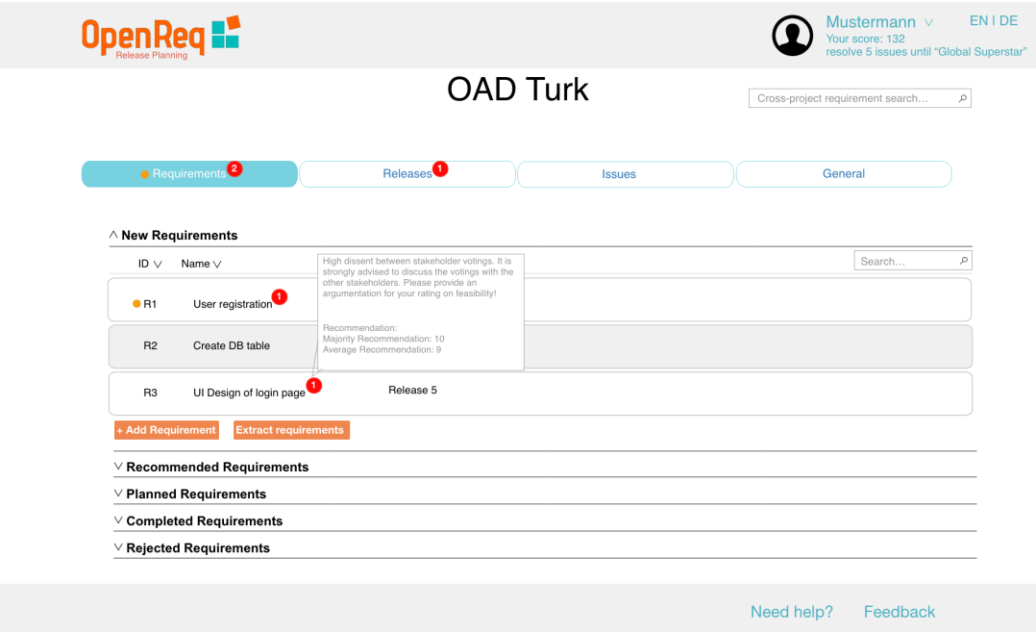

Figure 21: Representation of tooltips.

Recommended requirements. Figure 22 shows requirements which are automatically recommended by the system using the recommendation engine of OpenReq.

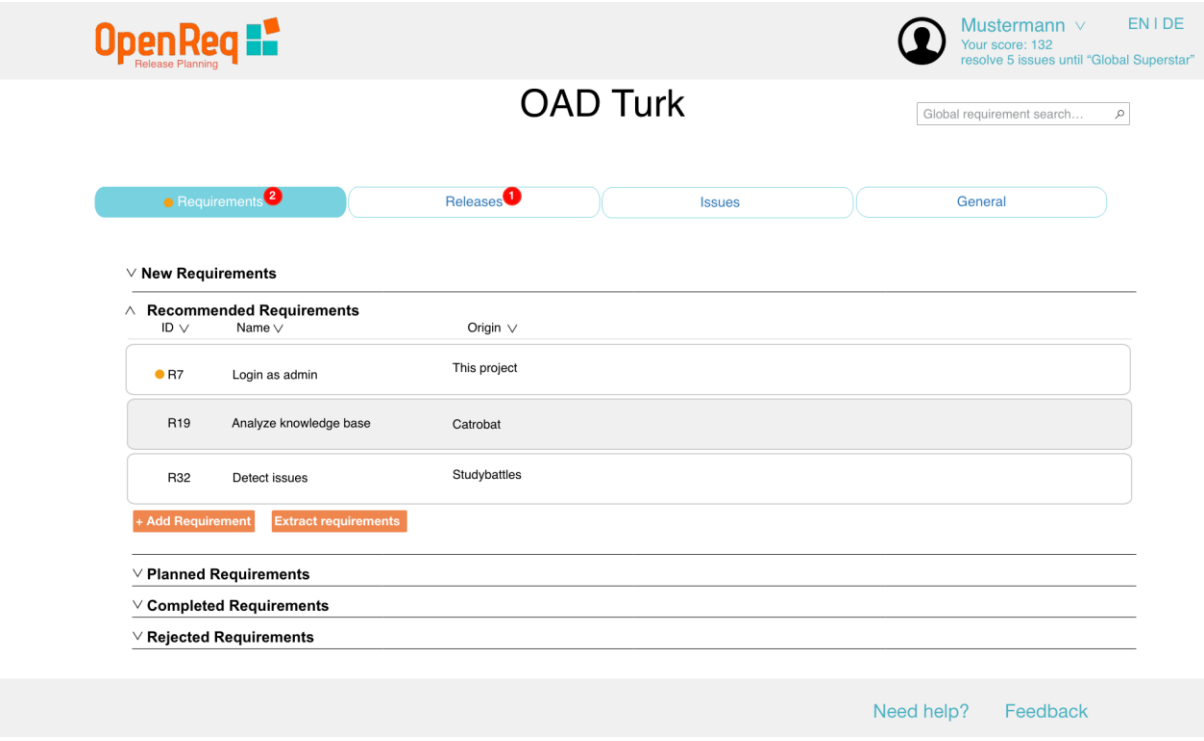

Figure 22: Recommendation of requirements.

The example screen depicted in Figure 23 includes an overview of basic properties of a requirement (click on Requirement "R1 - User registration" in Figure 21) and related indications of inconsistencies, for example, in terms of contradictory votings [Felfernig et al.

2018] or hidden dependencies that could endanger the quality/consistency of a requirements model.

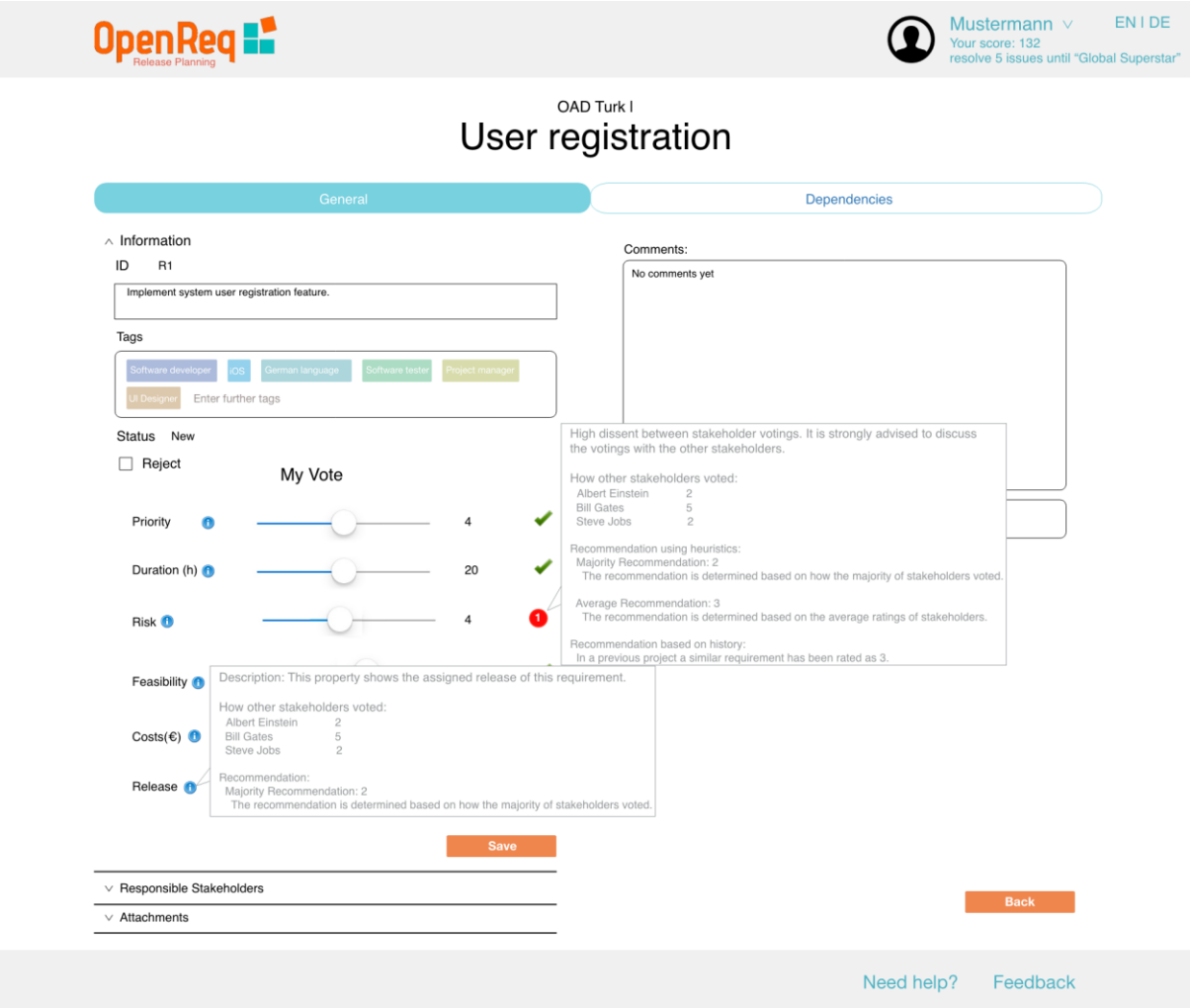

Figure 23: Properties of requirements (including indicators of inconsistencies).

The next screen (Figure 24) shows all stakeholders (experts) responsible for a requirement.

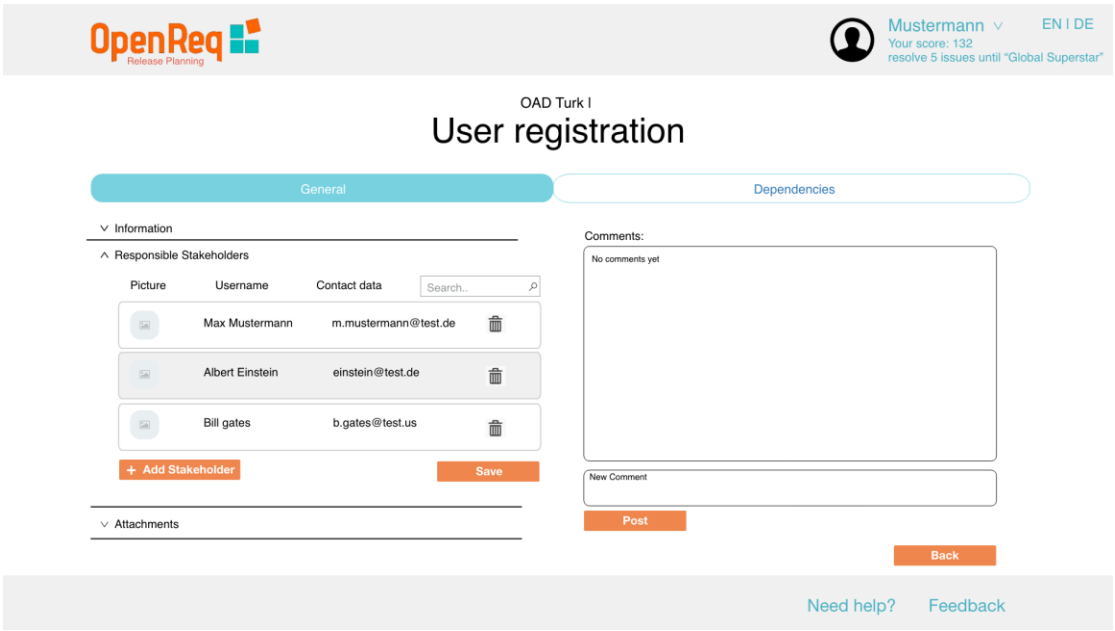

Figure 24: Responsible stakeholders.

Adding responsible stakeholders. This screen shows how to add a responsible stakeholder to a requirement (responsibility in this context is primarily related to quality assurance). Stakeholders can be searched by name, skills or both combinations. In addition, stakeholders for a requirement will be recommended by the system.

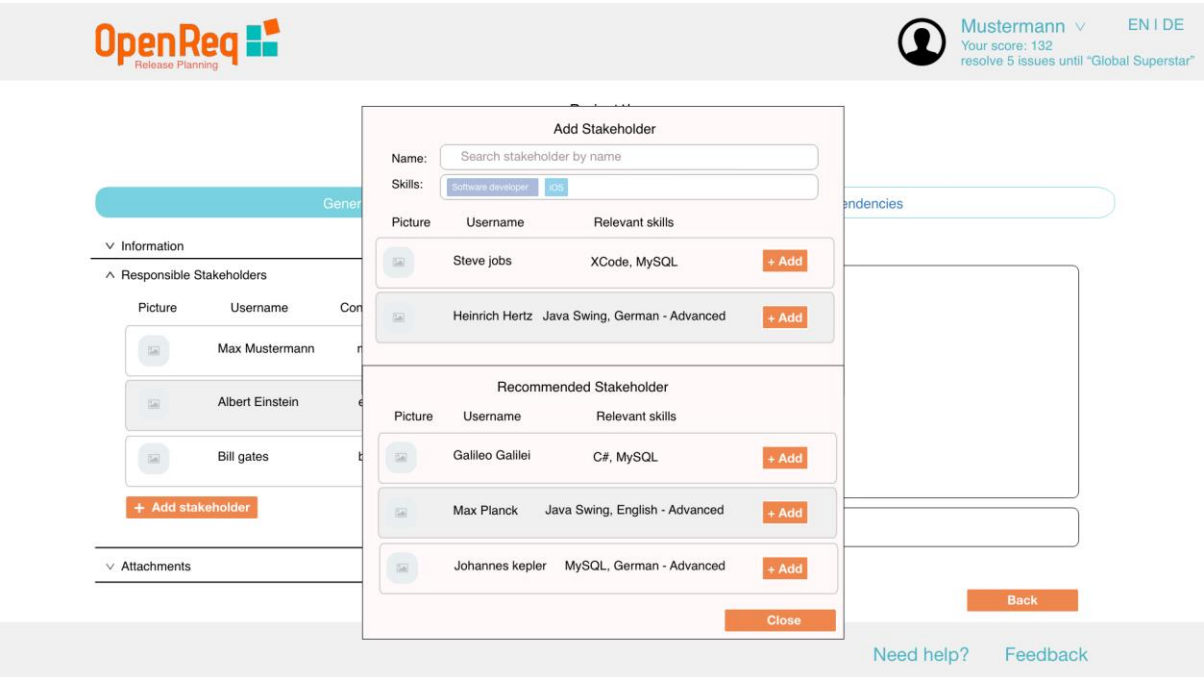

Figure 25: Stakeholder responsibilities.

Dependencies between requirements. Figure 26 shows dependencies of a requirement. A dependency can be, for example:

- "Requires" which means the current requirement requires another requirement, i.e., a specific requirement can only be implemented if another requirement has already been implemented.

- "Excludes" which means that one requirement must not be combined with another requirement in the same release (local "excludes"). On the global level, the implementation of one requirements excludes the implementation of another one (the excluded one) in the same software system.

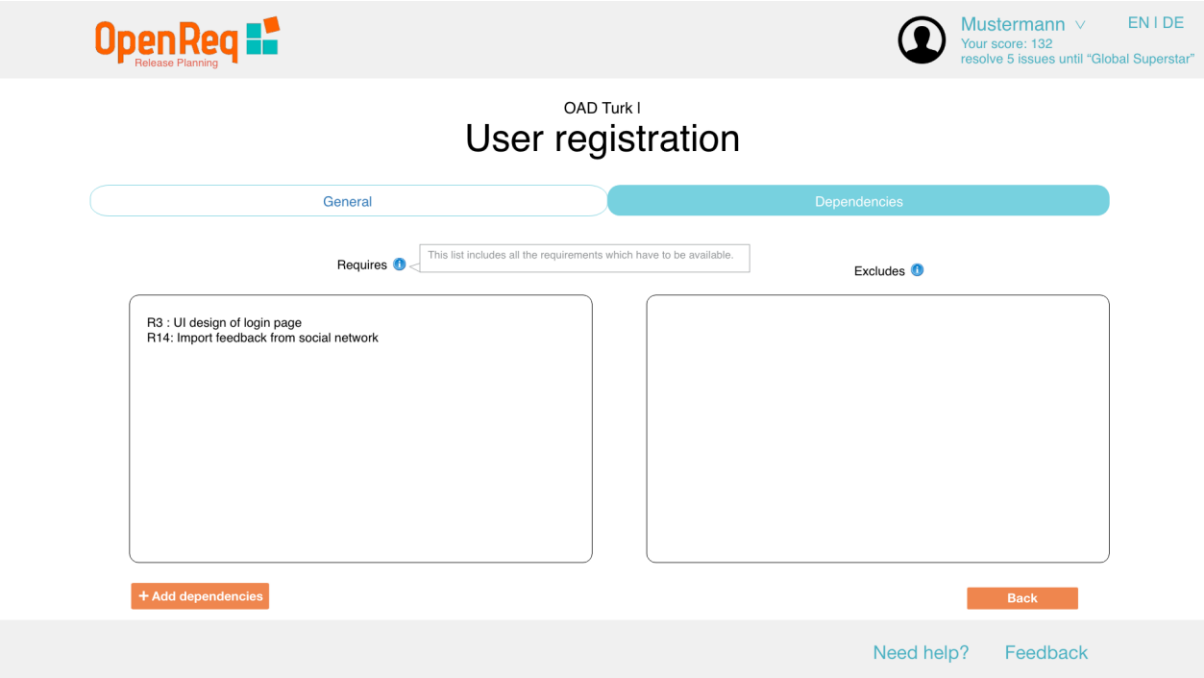

Figure 26: Representing dependencies.

Adding dependencies. Figure 27 shows how to add a dependency to a requirement. Manually or automatically detected dependencies can be added. The dependency detection engine is responsible for identifying "hidden" dependencies, i.e., those which were not yet identified by stakeholders.

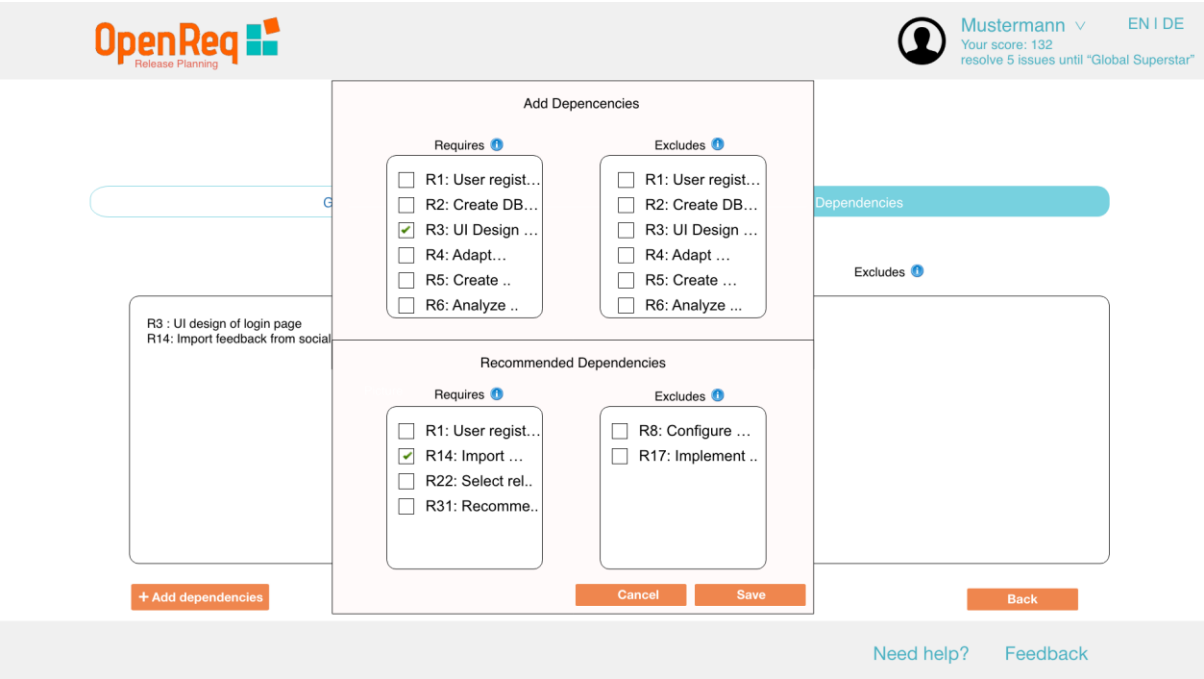

Figure 27: Definition of dependencies.

3. RECOMMENDATION APPROACH AND UI

Recommendation technologies will serve as core technologies to make OpenReq requirements engineering user interfaces more “intelligent”. Recommendations can be provided in two basic modes (see [Felfernig et al. 2010]). *Push* recommendations are proposed to stakeholders without the needed to explicitly trigger such recommendations. An example thereof are recommendations to resolve inconsistencies between stakeholder preferences (e.g., related to the assignment of requirements to releases). *Pull* recommendations are, for instance, recommendations for stakeholder assignment. For example, when defining a new project, the stakeholder recommendation functionalities can be activated the support the search for and the recommendation of individual stakeholders who should additionally be integrated into the project.

Beside the differentiation of push and pull recommendations, different basic recommendation algorithms can be used [Felfernig et al. 2014]. Depending on the application scenario, one of those algorithms will be chosen to provide user support. For example, in the context of a group-based release planning scenario, group recommendations have to be selected that are able to (1) propose release plans that satisfy the preferences of all or a majority of stakeholders, and (2) propose changes to given inconsistent requirements that allow to find a consistent release plan.

A more detailed discussion of different recommendation approaches used as a basis for building OpenReq technologies is outside the scope of this deliverable. In this context, we refer to the analyses of related work provided in OpenReq component-specific deliverables. We also want to point out that the application of recommendation technologies will not only be limited to the recommendation of basic items such as requirements, releases, and stakeholders but also focus on recommendations that help to trigger behaviour change of stakeholders, for example, increasing the preparedness of information exchange which is a major precondition for high-quality group decision making (see, e.g., [Atas et al. 2017]).

4. USER INTERFACES OF OPENREQ TRIALS

The following section describes the user interface approaches of the OpenReq trial partners.

4.1. Qt Trial User Interface

The Qt Company uses JIRA as it's main requirements management tool. Thus creating a JIRA interface interacting with the OpenReq services is needed. The interface will be used to present the results of the OpenReq services to the user as suggestions, so that the user can select what results to take into the main JIRA requirements database.

As the Qt trial starts in the beginning of the second year of the project period, the first initial tests will be done with simple web interfaces that the research partners have created.

The implementation for the JIRA plugin will start at the beginning of 2018, and the target is to have a working prototype by the summer. At the writing of this document there is no example to share yet.

4.2. Siemens Trial User Interface

Presently, Siemens uses DOORS for requirements management in bid projects. The first addressed task in the trial is to support classification, i.e. the decision whether a requirement (candidate) in DOORS is really a requirement (DEF) or not (Prose). This is accomplished by an extra menu entry in the application specific DOORS UI: It calls the corresponding OpenReq service via a DXL script and puts the result for each element (i.e. the confidence that it is a requirement, where 1.0 indicates 100% DEF and 0.0 means 100% Prose) into the classification column (see Figure 28).

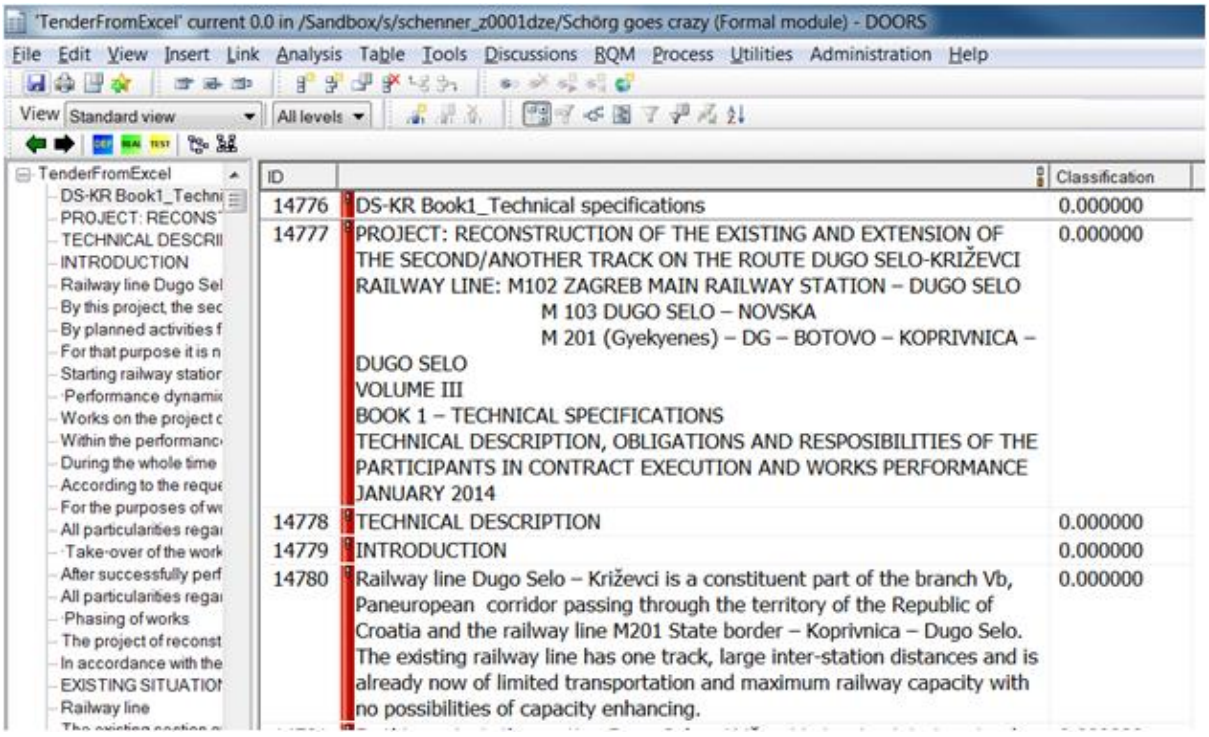

Figure 28: Integrating classification result in DOORS UI.

In order to easily evaluate all relevant OpenReq services, starting with classification (as mentioned above) and domain assignment (i.e. stakeholder assignment), we implemented a simple UI in Angular (see Figure 29).

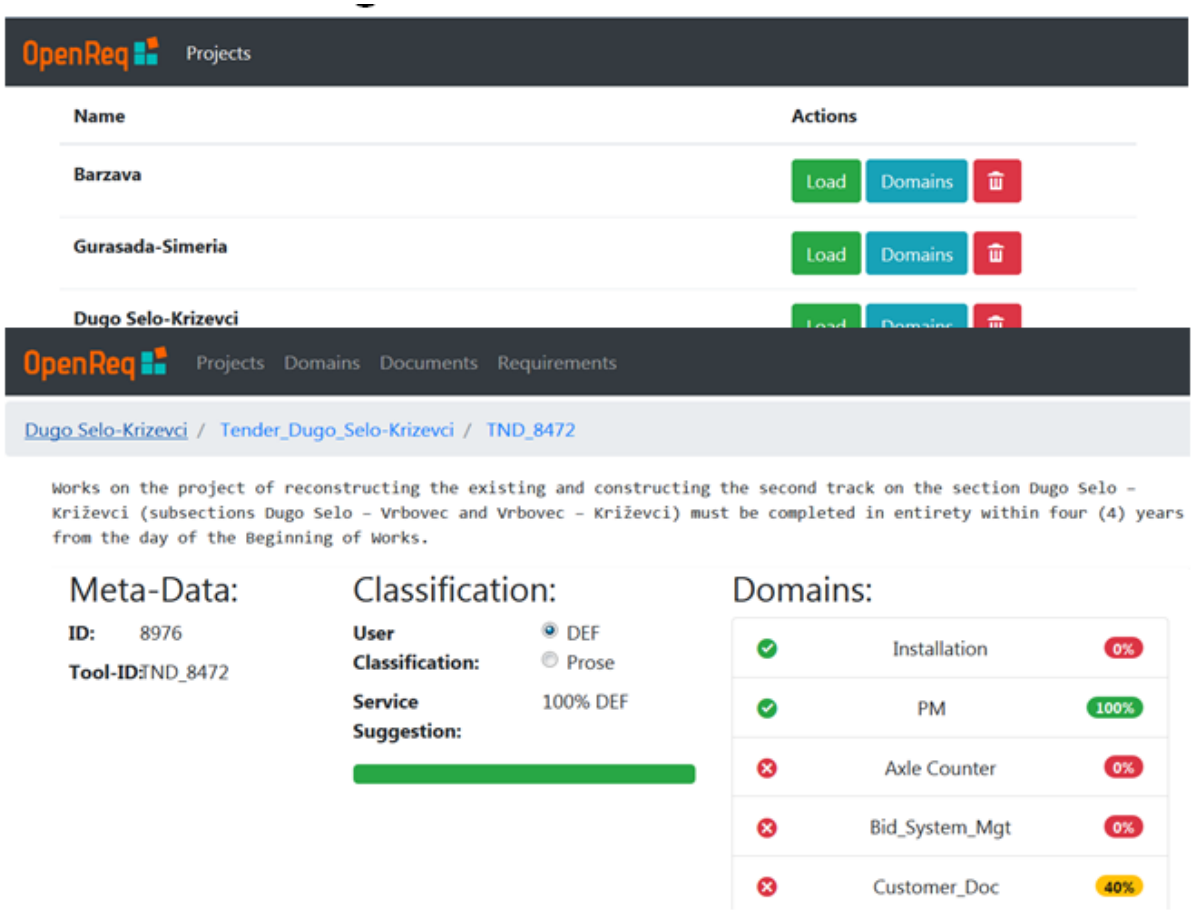

Figure 29: Siemens Trial UI (screens for project selection and properties).

4.3. Wind Tre Trial User Interface

The aim of Wind Tre trial is to extract customers needs from user-generated content (e.g. Social Network). In order to easily evaluate OpenReq services a simple UI will be implemented with social network data analysis.

4.4. Vogella Trial User Interface

To have a big impact in a large open source community with OpenReq the Vogella company will integrate the OpenReq functionalities into the Eclipse IDE. We consider this as an added trial to the initial three trials described in the DoA. This requires that openReq provides an interface to extract the requirements from Bugzilla.

Requirements in Eclipse can be displayed in a so called ‘view’, as a ranked list and sorted by different criterias, e.g., id, rank, assignee, priority etc. The number of fields depends on the fields in OpenReq, the UI in Eclipse will not enhance the data. User can edit requirements and upload them back to OpenReq, given that this functionality will be supported by the OpenReq implementation.

Integration with the Mylyn issue tracking is desired and will be evaluated during this trial.

The following figure (Figure 30) depicts the interface which will be used to present the results of the OpenReq services to the user as suggestions, so that the user can select what results to take into the requirements database.

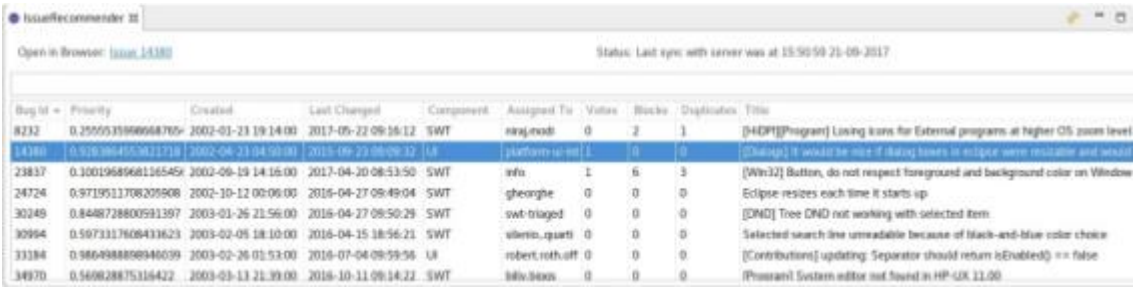

Figure 30: List of requirement suggestions

The user will be able to activate the OpenReq functionality within Eclipse. These settings will be done in the preferences of the IDE (see Figure 31).

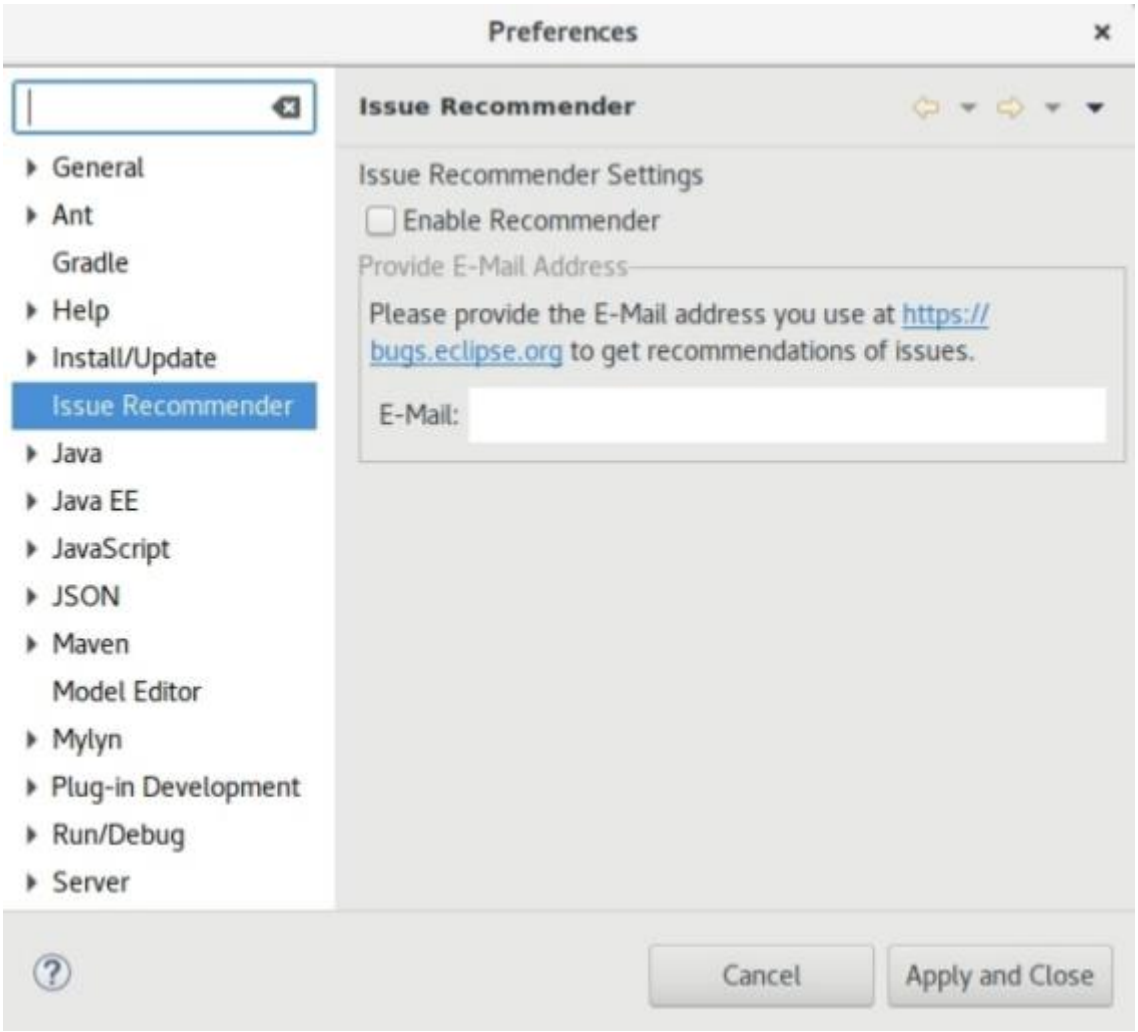

Figure 31: Eclipse IDE preferences for activating the OpenReq functionality

5. REFERENCES

- 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 (SocInfo 2017), Oxford, UK, pp. 368-377, 2017.
- J. Bernacki, I. Blazejczyk, A. Indyka-Piasecka, M. Kopel, E. Kukla, and B. Trawinski. Responsive Web Design: Testing Usability of Mobile Web Applications, Asian Conference on Intelligent Information and Database Systems (ACIIDS 2016), pp. 257-269, 2016.
- L. Chen and P. Pu. Evaluating recommender systems from the user's perspective: survey of the state of the art, User Modeling and User-Adapted Interaction, 22(4–5):317–355, 2012.
- A. Felfernig, L. Boratto, M. Stettinger, and M. Tkalcic. Group Recommender Systems, Springer, to appear in March 2018.
- A. Felfernig, M. Jeran, G. Ninaus, F. Reinfrank, and S. Reiterer, Basic Approaches in Recommendation Systems, Recommendation Systems in Software Engineering, Springer Verlag, pp. 15-37 Springer, 2014.
- A. Felfernig, W. Maalej, M. Mandl, F. Ricci, and M. Schubert, Recommendation and Decision Technologies For Requirements Engineering, ICSE 2010 Workshop on Recommender Systems in Software Engineering, Cape Town, South Africa, pp. 1-5, 2010.
- G. Ninaus, A. Felfernig, M. Stettinger, S. Reiterer, G. Leitner, L. Weninger, and W. Schanil. IntelliReq: Intelligent Techniques for Software Requirements Engineering, European Conference on Artificial Intelligence, Prestigious Applications of Intelligent Systems (PAIS), pp. 1161-1166, 2014.
- D. Winterfeldt and W. Edwards. Decision analysis and behavioral research, Cambridge University Press, Cambridge, 1986.

Appendix A

Figure 32 shows the data structure of the OpenReq prototype. This initial data structure will be synchronized with the OpenReq ontology in future versions of the software.

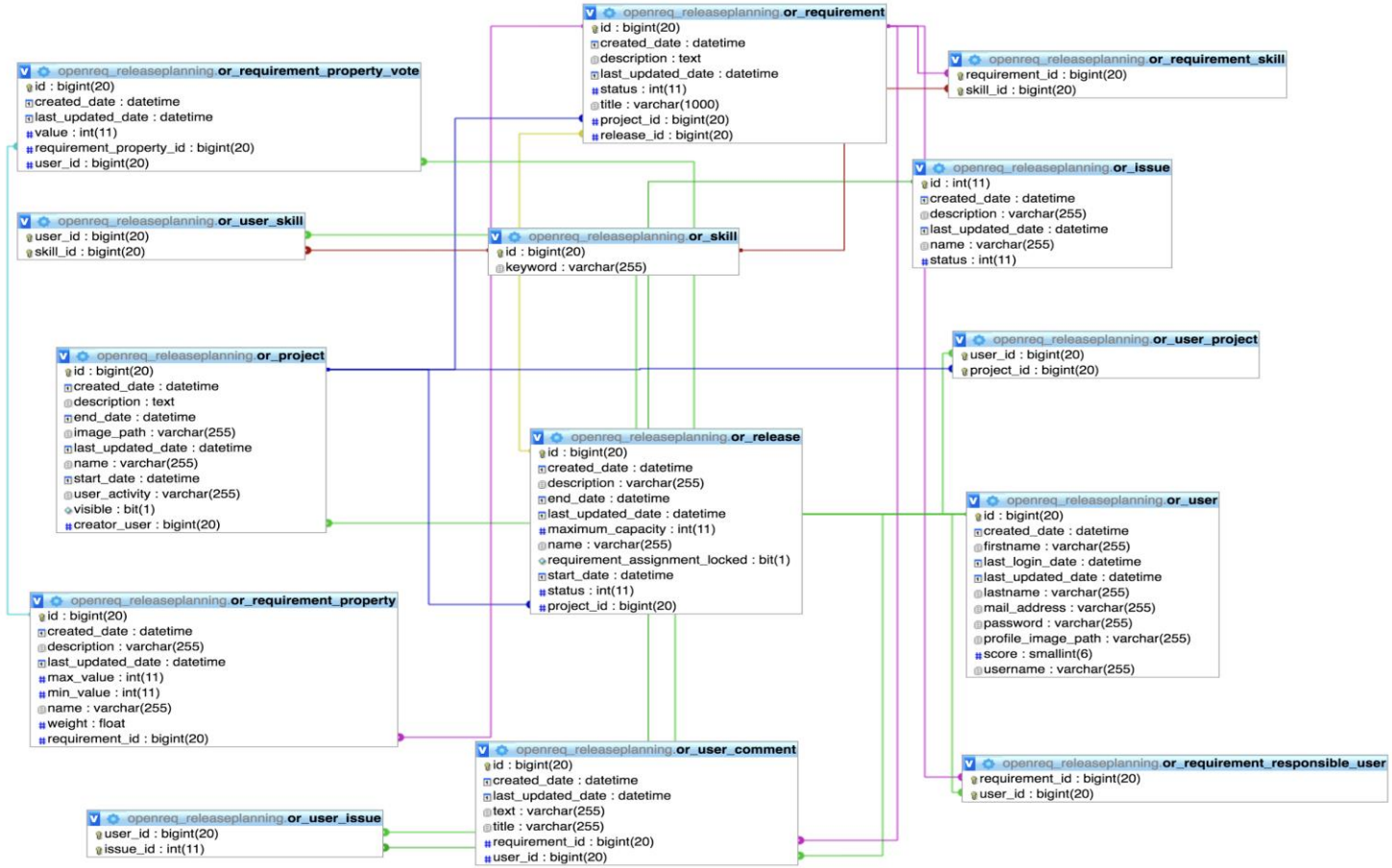

Figure 32: Data structure of the OpenReq prototype.