



Thomas Woger, BSc

Intelligent Methods for Software Release Planning

Master's Thesis

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Software Engineering and Management

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dipl.-Ing. Dr.techn. Alexander Felfernig

Institute for Softwaretechnology

Graz, December 2020

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used.

Date

Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Datum

Unterschrift

Acknowledgement

First of all, I would like to thank my supervising professor Univ.-Prof. Dipl.-Ing. Dr.techn. Alexander Felfernig for his continuous feedback and helpful support through the creation process of this thesis.

I also want to acknowledge and thank Dipl.-Ing Müslüm Atas, who supported me with helpful suggestions and discussions during the implementation of this application.

Furthermore, I would like to give my profound appreciation to my parents, who steadily supported me through the years of my study.

In addition, I'm grateful for all my friends, who turned my study into an incredible experience and finally, I would like to thank my girlfriend Tamara. She encouraged me through the writing process and gave me the necessary motivation to finish this thesis.

Thomas Woger
Graz, 2020

Abstract

During the creation of releases for products or projects, companies are confronted with complex decisions. They must determine the appropriate requirements which have to be added to the releases at the correct time. The preferences of all stakeholders must be considered during the planning. At the same time, stakeholders must also consider their available resources for the releases. Therefore, methods are needed to support the planning of the releases, including, involving the stakeholders more during the planning and observing the available resources. This thesis introduces a release planning approach, which takes into account the preferences of stakeholders during the planning process. Stakeholders can decide on different criteria such as business relevance or effort of requirements. These evaluations build the base for the prioritisation and assignment of requirements to releases. The prioritisation is determined using the relevance values of requirements, which are calculated with the help of Multi-Attribute Utility Theory (MAUT). During the evaluation process, stakeholders are supported by recommendations and techniques that help to achieve consensus. Furthermore, dependencies between requirements and the availability of the resources are considered during release planning.

Zusammenfassung

Während der Releaseplanung für Produkte oder Projekte sind Firmen oft mit schwierigen und komplexen Entscheidungen konfrontiert. Dabei müssen die Entscheidungsträger die richtigen Requirements auswählen, die zur richtigen Zeit in den Releases implementiert werden sollen. Hierbei müssen vor allem die Wünsche der Stakeholder während der Planung berücksichtigt werden. Zur gleichen Zeit müssen aber auch die verfügbaren Ressourcen im Auge behalten werden. Aus diesem Grund werden Methoden für die Releaseplanung benötigt, die einerseits die Stakeholder während der Planung mehr einbinden und andererseits die verfügbaren Ressourcen im Auge behalten. In dieser Arbeit wird ein Ansatz zur Releaseplanung vorgestellt, welcher die Stakeholder während der Priorisierung der Requirements stärker einbindet. Die Stakeholder können während der Planung die Requirements anhand verschiedener Kriterien wie Business-Relevanz oder Aufwand bewerten. Diese Bewertungen bilden die Basis für die Priorisierung der Requirements und die daraus resultierenden Zuweisungen zu den Releases. Die Priorisierung wird anhand der Relevanzwerte der Requirements durchgeführt. Diese Werte werden mit Hilfe von Multi-Attribute Utility Theory (MAUT) berechnet. Dieser Ansatz berücksichtigt die verschiedenen Wichtigkeiten von Stakeholdern anhand von Gewichten. Die Stakeholder werden während des gesamten Abstimmungsprozesses durch Empfehlungstechniken und Techniken zur Unterstützung des Gruppenkonsens begleitet. Des Weiteren werden die Abhängigkeiten zwischen Requirements und die Verfügbarkeit von Ressourcen während der Zuteilung der Requirements zu Releases berücksichtigt.

Contents

Abstract	vii
1 Introduction and Motivation	1
2 Related work	5
2.1 Planning game	5
2.2 Evolve II	6
2.3 INTELLIREQ	8
3 System architecture and overview	11
3.1 Architecture	11
3.2 Employed technologies and patterns	13
3.2.1 Spring	13
3.2.2 Spring Boot	13
3.2.3 Spring MVC	14
3.2.4 Spring Security	14
3.2.5 Spring Data	15
3.2.6 Thymeleaf	15
3.2.7 MySQL	15
3.2.8 WebSocket API	16
3.2.9 JavaScript	16
3.2.10 jQuery	17
3.2.11 Stomp	17
3.2.12 Bootstrap	17
3.2.13 D3.js	19
4 Used algorithms	21
4.1 Majority voting	21
4.2 Average voting	22
4.3 Median voting	23
4.4 Least distance (LDIS) voting	24
4.5 Multi Attribute Utility Theory (MAUT)	24
4.6 Interface for further algorithms	26

Contents

5	Procedure of release planning	27
5.1	Planning flow	28
5.2	Registration and login	30
5.2.1	Profile Page	31
5.3	Projects	33
5.3.1	Project overview	33
5.3.2	Project detail	34
5.4	Stakeholder	37
5.4.1	Assignment	37
5.5	Requirements	39
5.5.1	Requirement overview	40
5.5.2	Modelling	41
5.5.3	Dependencies	43
5.5.4	Evaluating a requirement	44
5.5.5	Recommendation	47
5.5.6	Support for finding a consensus	50
5.6	Releases	51
5.6.1	Release overview	52
5.6.2	Release detail	53
5.6.3	Assignment of requirements	54
6	Issue recognition and notifications	57
6.1	Issue overview	57
6.2	Issue creation	59
6.2.1	Automatic issues created by the system	59
6.2.2	Issues created manually by stakeholders	60
6.3	Notifications and visualization of conflicts	61
7	Statistics	65
7.1	Requirements assigned to releases	65
7.2	Disagreement of requirements	66
7.3	Optimisation chart	68
8	Release optimisation and diagnosis	71
8.1	Prioritisation	71
8.2	Dependency resolution	74
8.3	Assignment process	76
8.4	Diagnosis	80
9	Evaluation	83

10 Conclusion	85
10.1 Results	85
10.2 Future work	85
Appendix	89
List of Figures	89
List of Tables	90
Bibliography	93

1 Introduction and Motivation

Despite the fact that software solutions nowadays are important for almost every business, many software projects fail. The failure of these projects can have different causes. Possible factors for a project's failure include cancelled projects, exceedance of the project budget or the lack or incomplete delivery of the required features [WK04; BB05; LV01]. Furthermore, poorly defined requirements or poorly planned resources can lead to project failures [WK04; Cha05]. An simple example of a poorly planned resource is the lack of availability of developers.

The previously listed failures are associated with the first phase of the software development process: Requirement Engineering. Suboptimal requirement engineering is considered one of the major reasons, why projects fail and is therefore one of the most critical ones [HL01; JBK04]. There are various projects, such as OpenReq¹, which are trying to address the problems of requirement engineering. The OpenReq Requirements Engineering project is a EU Horizon 2020 project with the aim to build efficient recommendation and decision systems for the support of requirement engineering processes [Fel+17]. The main tasks of requirement engineering are organisational activities for quality assurance, the election and definition of requirements, negotiation and release planning [Som10; Fel+13].

In the release planning phase, requirement defects can occur. In a requirement defect, the software works as expected on the developers' side, but it may be too difficult for the users to use or the customers may not be completely satisfied [LV01]. Much work must be invested to solve these issues in the implementation phase. The costs can be up to 100 times more expensive than solving these defects during the analysis phase [BB05]. Improving the defining and collecting requirements must be an important goal to reduce these costs. For this reason, requirement engineering is considered one of the most important and challenging phases of a software project [Fel+13]. These facts also illustrate why there is an increasing demand for smart solutions to advance the requirement engineering process [Fel+10b; Ren+13; MT09]. Therefore, the following work focuses on the release planning part of the requirements engineering process.

¹<https://openreq.eu/> (Retrieved 18.10.2020)

1 Introduction and Motivation

The main task of release planning is to create a schedule for predefined requirements, which should be implemented during a predefined release period [REP03; Fel+13; Fel+18b]. During this process, companies are often confronted with the challenge of choosing the right requirements for the next releases [BRW01; RB05]. A release plan must fulfil different criteria. The plan must fit into the budget, not exceed available resource capacities and availabilities. The plan should provide a maximum business value for the company, by considering the preferences of stakeholders and by considering the dependencies between requirements [RS05; Sva+10]. Therefore, the company must master different challenges during the planning process for the next releases.

Important stakeholders also often perform pressure on the company to select their favoured requirements [Dag+05]. These stakeholders frequently have different opinions about the importance of the requirements. This leads to even more complex decision processes [RM05]. Furthermore, it is often not possible to implement all requirements, due to different bottlenecks [Akk+08; Ruh10].

Complex and large software projects often lack efficient triage processes [Fel+13]. The term 'triage' originates from the medical field. In the medical context, victims of catastrophes are categorised into three different groups [Dua+09]. All victims who have no chance of recovery are in the first group. The second group includes those who recover with the help of treatment, and the last group includes those who can recover without any treatment [Dua+09]. When this process is used in the context of release planning, the demands that must be fulfilled have to be identified, in order to obtain an optimal release [Dav03; Fel+13].

By considering the restrictions of a release plan, stakeholders must prioritise requirements [Dua+09]. To receive favourable solutions, studies have demonstrated that such decisions should be made by the affected groups [And98; JBK04; CC12]. For this purpose, stakeholders must work together to achieve a feasible solution [LQF10]. To accomplish this challenge, the stakeholders need intelligent support in the prioritisation of requirements.

If a release plan is suboptimal, for example, it does not take into account customer preferences, customers could be dissatisfied, which could cause the project to fail [BRW01; HL01]. Additional costs triggered by suboptimal release plans can increase the total project cost by up to 40% [Lef97]. This argument reveals, that release planning is a complex process with many involved stakeholders, in which the effective prioritisation and planning of the suggested requirements is extremely important.

Different models are available for the creation of a release plan. Two known approaches for the creation of a release plan are ad-hoc-planning and systematic-planning [Lin+08]. For efficient planning, the ad-hoc-approach is already out of date because software technologies are growing so fast that managers are not able

to productively handle those changes [JBK04]. There are also models which were created by the combination of these two approaches, named hybrid approaches [RM05]. These models combine the knowledge of experts with the strength of mathematical models [RN04]. These hybrid models try to demonstrate that the mathematical models alone are not enough and that the knowledge of experts is also needed to make correct decisions for the release plan.

A general guideline for a software life cycle with the planning process exists, but this guideline does not describe how the requirements can be assigned to achieve optimal business value [08; RS05]. For this reason, different release planning techniques have been created (like [BRW01; Akk+08; RN04; Ruh10; RM05; Nin+14]). These models focus on different criteria, which influence the quality of the release plan.

Release plan optimality can be affected by several criteria. These criteria include business value and quality of the requirements as well as stakeholder preferences [Akk+05; Ruh10].

Promising hybrid approaches are the combination of the involvement of the stakeholders' opinions of the requirement's importance with the use of linear programming techniques [RS05; Akk+08]. To achieve effective prioritisation of requirements, these should be evaluated on various criteria. Some useful prioritisation criteria are *consistency* (no incompatible requirements), *feasibility* and *effort* for the implementation, and *reusability* in future projects [Fel+13].

During requirements prioritisation, conflicts between the preferences of individual stakeholders can occur. To resolve these conflicts, tools must support mediation between the conflicting parties [Nin+14; Fel+11].

This thesis presents the Intelligent Release Planner (IRP) application, a software to support release planning decisions of managers. The software guides users through the whole release planning process. The basis of the model utilises on hybrid approach. Stakeholders evaluate requirements based on different evaluation (interest) dimensions. In this context, a recommender system helps stakeholders to determine the significant requirements in a large collection and to generate a useful decision [Buro2; BFG11]. Stakeholders have a platform to discuss the requirements during the prioritisation process. They are also able to state advantages or disadvantages of the requirements (pro/con analysis), which should help to achieve a consensus. Furthermore, the developed release planning software includes automatic conflict detection. If the stakeholder evaluations are contradictory, stakeholders are encouraged to resolve the conflict on the basis of a recommendation.

The remainder of this thesis is organised as follows. In Chapter 2, similar research projects for release planning are presented. Next, Chapter 3 addresses the

1 Introduction and Motivation

software architecture of the system, and all important technologies, which were used for the system, are described. In Chapter 4, all the used algorithms are presented. The procedure of release planning with all its steps in the implemented software is demonstrated in Chapter 5. This chapter starts with the definition of the projects, continues to the evaluation of the requirements and ends in the representation of the planned releases. Chapters 6, 7 and 8 deal with more specific topics of the release planning software. In Chapter 6, the recognition of conflicts between stakeholder preferences and their representation is demonstrated. All available statistics, which should help the responsible managers to select the preferred releases, are described in Chapter 7. Chapter 9 presents a related paper, which analyses argumentation-based interfaces. Finally, Chapter 10 provides a brief conclusion and discusses issues for future work.

2 Related work

This chapter provides an overview of existing methods and tools that support an efficient release planning.

2.1 Planning game

In agile software development, the term 'planning game' is well known. The approach is embedded in extreme programming (XP) [BGoo]. The development team and business' representatives arrange user stories in such a way that they can be published in the next release.

In consideration of the business' value, the selection of the stories for a release is based on the estimates of the development team [FHC06]. Determining the scope for the next releases is performed in iterations, with a combination of the developers' estimates and business' priorities [BGoo]. Therefore, the planning goal is to achieve maximum business value by selecting the most important stories of a software.

During the game two parties are involved. These sides are the development team and customers, and each party has its own responsibilities and members [PEM03; Kar+04]. On the business side, the business team must decide about [BGoo]:

- Scope - What should be implemented?
- Priority - Prioritisation of the stories.
- Composition of the releases - Which stories must be included to obtain maximum business value?
- Dates of releases - When is the best time to introduce the release to gain a benefit for the business?

On the other side, the developers must decide about [BGoo]:

- Estimates - Duration of the feature implementation.
- Consequences - Impacts of software architecture decisions on the business.
- Process - Organisation of the team.
- Detailed scheduling - Defining the implementation sequence of stories in a release.

2 Related work

According to Beck [BG00], the planning game is structured into three phases. In the first phase, the 'Exploration phase', an overview of the system's functionalities is given and the stories are defined. In this phase the business side defines the stories, and the development team estimates the stories and if necessary, divides them into smaller software pieces.

The second phase is the 'Commitment phase'. In this phase, the business side defines the priority of the stories, which are also evaluated by the development team with regard to their risks. The business team defines the next release on the basis of this information.

The last phase is the 'Steering phase'. Based on the learning of the development and business side, the release plan is constantly being updated. Therefore, in each iteration, the most appropriate stories are selected. In this context, old stories can be replaced by new stories. Furthermore, the stories are re-estimated by the development team if there were new insights.

The planning game is easy to apply and delivers a prioritised sequence of features. Furthermore, studies have demonstrated that the approach works well in projects with a smaller number of features, but in larger projects the approach becomes more complex and the failure rate increases [ARSo4; Kar+04].

One of the biggest differences between the IRP application and the planning game approach is that all prioritisations are made in a group of stakeholders, who must be present during the prioritisation process. In the IRP application, stakeholders are supported by recommendation technologies. Unlike the IRP application, which can handle bigger releases, the planning game is especially useful for small release circles and a small number of features.

2.2 Evolve II

Evolve II is a systematic decision support approach to create a release plan [Ruh10]. The method is an enhancement of Evolve, which was first presented in [GR04]. Evolve II has been developed under the assumption that the isolated formulation of mathematical models or the isolated selection and knowledge of the decision makers are not sufficient to create a high-quality release plan.

Therefore, Evolve II combines computational strength of computers with expert knowledge. To determine solutions, integer linear programming (ILP) [Sch86] techniques are employed [Ruh10].

The Evolve II approach follows an evolutionary problem-solving process, which is organised in three phases [Ruh10]:

- modelling,
- exploration, and
- consolidation.

Based on real world problems, models are created during the 'modelling' phase, which build the input for computational algorithms. In the modelling phase, possible features are defined with a description as well as all their dependencies. Also, budget and resource constraints are determined [Ruh10; ARSo4].

Next, is the 'exploration' phase, where the goal is to gain a greater knowledge of the problem's structure and to reduce the number of solutions. During the exploration, iterations occur to determine the possible alternative solutions out of a large set of solutions. The determination is executed with the help of pre-defined criteria and using optimisation techniques [Ruh10].

The last phase is 'consolidation'. This phase relies on expert knowledge and know-how of human decision makers. Alternative solutions from the exploration phase are checked by experts. The decision makers evaluate the solutions and select the most appropriate one [Ruh10].

In the background of the three phases, 13 steps are being performed [NR14]. Five of them are optional. These steps are predominant, where the suggested alternative plans are defined and analysed in more detail taking into account the available resources.

The mandatory steps start with the definition and selection of the features and the planning of the weights used as planning criteria [Ruh10]. After these stages, the features are prioritized by the stakeholders according to the defined voting criteria [NR14]. Based on the prioritisation and resource constraints, alternative release plans are created. These plans are prioritised by the stakeholders and a final plan is selected.

To summarize, Evolve II is a complex approach for release planning. The method provides the possibility to plan more than one release and intensively involves the stakeholders during planning. Evolve II also allows the definition of resource constraints and dependencies between features. This method is integrated in the *ReleasePlanner*TM system [Ruh10].

The IRP application, developed within the scope of this thesis, is also based on the combination of expert knowledge and optimisation approaches. In Evolve as well as IRP, stakeholders are involved in early phases of release planning. In addition to the Evolve approach, the stakeholders in the IRP application are supported by recommendation technologies, which determine recommendations based on the preferences of other stakeholders.

Furthermore, the IRP application offers a platform for discussions to support the

achievement of a consensus amongst the stakeholders.

2.3 IntelliReq

INTELLIREQ was introduced by Felfernig et al. [Fel+11] to assist in various requirement engineering challenges. The tool's target is to support the early phases of requirements engineering by modelling and prioritising requirements [Nin+14]. Different recommendation technologies are applied during the engineering process. The tool's output is a consistent set of requirements with corresponding dependencies, estimates, prioritisations, and planned releases [Nin+14].

During the requirement engineering process, stakeholders are strongly involved in INTELLIREQ and can define and adapt their preferences [Fel+11]. Stakeholders can discuss their preferences and see recommendations for the requirements. The prioritisation of requirements is also performed by stakeholders. During the prioritisation phase, a consensus must be present for each requirement [Nin+14].

In INTELLIREQ, requirements are modelled with a description, evaluations with regard to predefined criteria and dependencies [Nin+14]. The detailed view of a requirement is depicted in Figure 2.1. The stakeholders can also view the preferences of other stakeholders and can discuss their evaluations. The recommendation of preferences is made by group recommendation technologies, more precisely with the help of the majority voting heuristic (see also Chapter 4) [Nin+14].

INTELLIREQ employs model-based diagnosis (MBD) of inconsistencies [Rei87; FSR13] to identify conflicts between stakeholder preferences [Nin+14]. To display the status of preference elicitation, traffic lights are introduced. These lights are also employed to indicate other issues in a requirement model or release. If a traffic light is red, it signals that further evaluations or actions are required.

The assignment of requirements to releases is performed by stakeholders. If there are still unassigned requirements, the system can suggest an assignment. Based on group recommendation techniques, it is also possible to let the system assign all requirements to releases automatically [Nin+14].

Summarizing, INTELLIREQ offers a wide spectrum of functionalities to support requirement engineering. The quality of release plans as well as the time effort for planning can be improved by using recommendation technologies. Furthermore, the diagnosis of conflicts and support of finding a consensus between stakeholders can improve the quality of the release plans.

The biggest difference between INTELLIREQ and the IRP application developed in this thesis is the involvement of the stakeholders. While stakeholders can discuss requirements and evaluations in both applications, IRP supports also stakeholders with giving approvals or disapprovals to comments of other stakeholders. All the

Meta Data

Description - Bluetooth

The watch needs a bluetooth connection. This is necessary to connect the watch to other devices. Example: heart rate chest strap

Meta Data	Adjust	Info
Risk: 4 (Average)	<input type="range"/>	
Feasibility: 8 (High)	<input type="range"/>	
Cost: 5 (Average)	<input type="range"/>	
Relevance: 4 (Average)	<input type="range"/>	
Priority: 1 (Low)	<input type="range"/>	
Duration: h	<input type="text" value="20"/>	
Preferred Release:	<input type="text" value="R2 - 2014-03-13"/>	

Select Meta Data Type to view
Priority

All ratings for Priority

Picture	Name	Rating
	IRManagement	4
	IRKunde	7
	AlexanderFelfernig	1

Recommendation

Majority Recommendation: 1

Average Recommendation: 4

Save

Figure 2.1: Detailed view of a requirement with all its criteria and corresponding evaluations of stakeholders. [Nin+14]

2 Related work

positive and negative comments and arguments are also taken into account in the recommendation of the requirements and the release plan (see Chapter 8).

3 System architecture and overview

This chapter presents the system architecture of the implemented release planning software IntelligentReleasePlanner (IRP). The composition of the software modules and the employed software technologies are discussed. In Section 3.1, the system structure of the software with the whole data flow is presented. Thereafter, in Section 3.2 all relevant technologies used for the implementation of the IRP application are discussed.

3.1 Architecture

The architecture of the software is based on a client-server model. The main advantage of this model is, that the user only requires a web browser on his or her computer. Furthermore, the user does not have to install the IRP application on his or her computer or mobile device.

The architecture is based on the MVC model of Spring (see Section 3.2.3). The data flow of the software in combination with Spring is complex and depicted in Figure 3.1. Figure 3.1 lists all steps, which are performed by the IRP application. The data flow follows thirteen steps:

1. After the user enters the application URL or submits a form, a HTTP request is sent to the server.
2. The **Dispatcher Servlet** receives the request from the client. This component is the front Controller and receives all incoming requests. After the reception of the call, the Dispatcher delegates the request to the corresponding component. In this case, the receiving component is the **Handler Mapping** component.
3. The **Handler Mapping** component serves as a consultant for the Dispatcher. The Handler must find the correct recipient for the request. For this purpose, the Handler parses the URL and routes the request to the correct Controller with the corresponding Handler method.
4. The responsible Controller processes the incoming request. In this step, the task is to prepare the correct model for the view. If necessary, additional business logic is applied.
5. The **Service Layer** provides the connection to the repository queries.

3 System architecture and overview

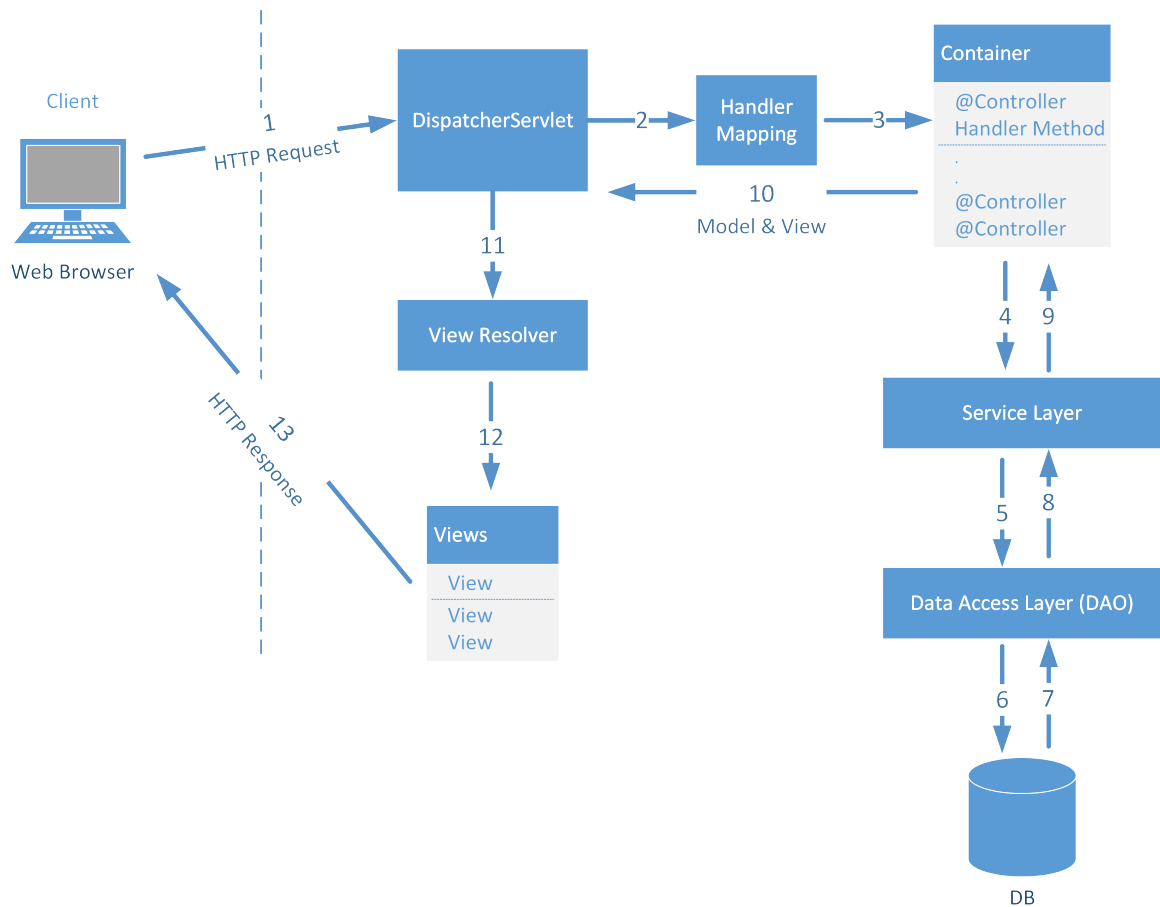


Figure 3.1: Architecture of the IRP system.

6. To retrieve the information from the database, the **Data Access Layer (DAO)** is utilised. This layer utilises the Spring Data framework (see Section 3.2.5).
7. The database delivers the data requested from the query.
8. The result is returned to the Service in the predefined format.
9. If necessary, additional operations are performed on the result and then passed on to the calling Controller.
10. The gathered information is still in a raw format and therefore handled back to the Dispatcher.
11. To create the appropriate page, the Dispatcher forwards the data to the **View Resolver**.
12. The **View Resolver** determines the correct view with the help of the view name.
13. The view is responsible for the presentation of the data and creates the response with the provided data in HTML format. Therefore, the attribute values of the models are set in the view and in order to generate the views in

the IRP software, the Thymeleaf template engine (see Section 3.2.6) is utilised. In the end, the web browser displays the rendered page.

The **Dispatcher Servlet** is the central contact point for all requests. Before the request can be processed by the Dispatcher, the data is revised by filters of the Spring Security framework (see Section 3.2.4). The filters determine whether the requesting user is authorized and authenticated, and only if this is the case, the request is forwarded. If the user is not authenticated, he or she is rerouted to the login page (see Section 5.2). To ensure effective security, this behaviour is applied to each request, .

3.2 Employed technologies and patterns

This section presents the employed software technologies in the IRP software. The thirteen main technologies are presented in the following subsections, which are predominantly JAVA¹ and JavaScript (see Section 3.2.9) based frameworks.

3.2.1 Spring

The Spring² framework is an open source library and was developed for the Java environment. The framework attempts to improve and relieve the development process for software applications. One of the main features of Spring is the usage of the *inversion of control* pattern (IoC). The framework with its containers manages the lifetime and resource allocation for an object.

3.2.2 Spring Boot

One of the Spring projects is Spring Boot³. The project is built on the top of the Spring framework and the starting point for the creation of various applications. Spring Boot enables developers to more easily set up and configure new web-based and standalone applications. To arrive at a runnable application in Spring, several configuration and XML files must be created manually. These configurations are now done in the background by Spring Boot.

Additional main features of Spring Boot are 'standalone' and 'opinionated'. Spring Boot is absolutely 'standalone'. Only by running one command the application

¹<https://java.com/en/> (Retrieved 01.10.2020)

²<https://spring.io/> (Retrieved 01.10.2020)

³<https://spring.io/projects/spring-boot> (Retrieved 01.10.2020)

3 System architecture and overview

is started on an embedded web server. For this purpose, the framework contains various embedded web servers, which do not have to be configured.

At the 'opinionated' feature Spring Boot creates configurations for the included third-party libraries by itself, so a developer does not have to invest time to set up the libraries each time.

3.2.3 Spring MVC

The Spring Model View Controller (MVC)⁴ framework is one of the first Spring projects and is utilised to build Java web applications and RESTful web services. MVC is a design pattern, which defines three interconnected components:

- Model: manages the data of the application.
- View: responsible for the representation of data.
- Controller: validates input commands and manages models and views.

The pattern specifies that in an application the Model must be separated from the View and the Controller. This separation enables a parallel development of the individual components and an easy reuse of the code. Spring provides its own implementation of the pattern.

The core of the framework is based on the *Front Controller* pattern with the Dispatcher Servlet. The Dispatcher receives the input of the user and searches for the responsible Controller. The distinction between Controllers is defined by the URL. Therefore, all the methods of the Controllers, which should be available to user requests, are mapped with an URL. The Controller is then responsible for further processing of the request.

3.2.4 Spring Security

Spring Security⁵ is a project of Spring. The project provides authorization and authentication features for Java applications. Furthermore, the framework provides a huge number of authentication features and customization options. The framework also provides protection against known attacks such as cross site request forgery. Spring Security is applied in the IRP software for the authentication of users and an automatic decryption of user passwords. Before a request is forwarded to a servlet, it must run through the security filters of the framework and only then the

⁴<https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html> (Retrieved 01.10.2020)

⁵<https://spring.io/projects/spring-security> (Retrieved 01.10.2020)

request is routed to the correct servlet. The framework is also responsible for the encryption and decryption of passwords.

3.2.5 Spring Data

Spring Data⁶ enables developers to easily access various relational and non-relational databases. First, developers must only define a repository interface and the framework automatically creates the according repository. After the definition of these interfaces, Spring Data generates the data access to the databases in the background. Furthermore, database schema updates are executed easily with the Spring Data framework.

Spring Data can dynamically derive database queries only by defining method names in the interfaces. Therefore, simple queries do not have to be created manually anymore.

Furthermore, the framework provides extensive auditing possibilities. Due to these advantages, Spring Data is utilised for all database accesses in the IRP software.

3.2.6 Thymeleaf

Thymeleaf⁷ is employed in the IRP application for the dynamic generation of the HTML5 views. The framework is a server-side Java template engine and is applicable for web-based and standalone applications. One of the main benefits of the engine is the complete integration of the Spring framework. Therefore, it is possible to combine the models of Spring with the Thymeleaf templates.

Thymeleaf provides the same possibilities as JSP and JSTL, but the use of Thymeleaf is much easier and not as complex as with JSP. Furthermore, many other templates can be processed, such as text, JavaScript, or CSS files.

3.2.7 MySQL

MySQL⁸ is employed as database technology for the IRP software. MySQL is a broadly used database system also available as Open Source distribution. The complete communication between application and database is handled with the Spring Data framework (see Section 3.2.5).

⁶<https://spring.io/projects/spring-data> (Retrieved 01.10.2020)

⁷<https://www.thymeleaf.org/> (Retrieved 01.10.2020)

⁸<https://www.mysql.com/> (Retrieved 01.10.2020)

3.2.8 WebSocket API

In the IRP application, the WebSocket API⁹ from Spring is utilised to provide a group chat function to stakeholders. A WebSocket is a bi-directional and persistent connection between a client and a server [FM11].

The WebSocket can be initiated by a request of the web browser over http. When the connection is established, both sides can send data to each other, until the connection is closed by one of these sides. In IRP, the API is utilised for the server side, and Stomp (see Section 3.2.11) is utilised for handling of the connections on the client side.

3.2.9 JavaScript

JavaScript¹⁰ is a dynamic and interactive script programming language and predominantly employed to generate or modify web pages. With the help of JavaScript, HTML elements can be changed or inputs validated. The scripts of the programming language are executed in the clients' web browsers. Therefore, no additional requests to the servers are required to modify a current page.

JavaScript also provides the possibility to present the content in a well formatted structure. These structures can be tabs, dialogues, or tables. Furthermore, not all the content must be displayed instantly to the user, so that the content is only visible after clicking on a button or after hovering over an image. These controlling possibilities allow presenting web sites more clearly, and only if required, additional information is displayed. After the client receives the content of the page from the server, the client is responsible for the appropriate presentation of the site. Therefore, the server must only send the data to the client and must not care about the representation, which also saves computation time.

The JavaScript functions can be 'registered' on events of the user input. Thus, it is possible to recognize mouse clicks or text inputs immediately and react accordingly. Additionally, animations can be created with this technology.

JavaScript is the basis for several large and extensive frameworks. Examples thereof are jQuery (see Section 3.2.10) and the styling framework Bootstrap (see Section 3.2.12).

In IRP, JavaScript is employed for an improved and dynamic user interface appearance. Also, form validations are made with this framework.

⁹<https://docs.spring.io/spring-framework/docs/5.0.x/spring-framework-reference/web.html#websocket-server> (Retrieved 01.10.2020)

¹⁰<https://www.w3schools.com/js/default.asp> (Retrieved 01.10.2020)

3.2.10 jQuery

jQuery¹¹ is a JavaScript library which provides an effective and easy access to HTML elements. The library also supports developers with an effective event handling, AJAX communication, and animations. With the improvement of the use of AJAX calls, requests to the server are easy to implement. jQuery also provides various plugins for an improved user interface handling. Therefore, this library was utilised in the IRP application to access HTML elements.

3.2.11 Stomp

The simple text-orientated messaging protocol (STOMP)¹² provides an interoperable format to connect clients with servers. With the help of WebSockets, the clients can exchange data with message brokers of servers. The protocol is available on various client and server platforms and in different programming languages. IRP employs the JavaScript version of the protocol in combination with the WebSocket API (see Section 3.2.8).

3.2.12 Bootstrap

Bootstrap¹³ is a JavaScript (see Section 3.2.9) and CSS-based framework to improve the user interface appearance in web browsers. With the help of this framework, a responsive design can be created easily. The framework also provides many design templates, buttons, forms and navigation elements. Bootstrap is compatible with all modern web browsers and can be integrated easily in all HTML views. Furthermore, Bootstrap plugins for various user interface functions are available. Some of the important features of Bootstrap, which are included in the IRP software, are:

- grids,
- tabs,
- modals, and
- datatables.

With the help of the **Grid** feature, the application gains a responsive design, which also improves the view for mobile devices. The user screen is partitioned by Bootstrap into twelve columns per row, which is demonstrated in Figure 3.2. If there are more than twelve columns in a row, the columns are wrapped to a new

¹¹<https://jquery.com/> (Retrieved 01.10.2020)

¹²<http://stomp.github.io/stomp-specification-1.2.html> (Retrieved 01.10.2020)

¹³<https://getbootstrap.com/> (Retrieved 01.10.2020)

3 System architecture and overview

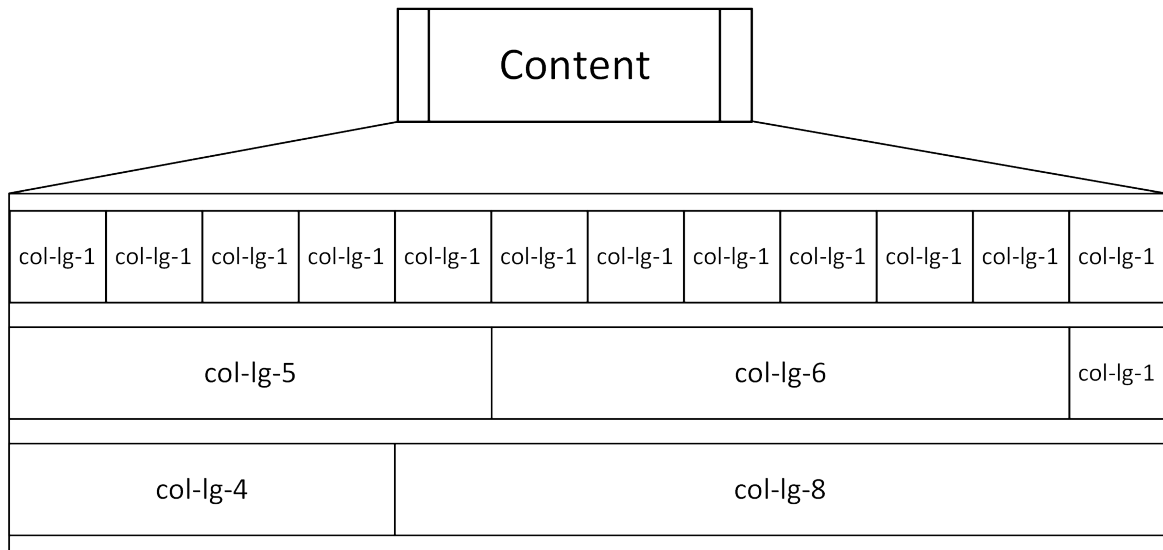


Figure 3.2: Partitioning of the screen by Bootstrap in different column sizes.

line. Depending on the screen width of the browser, the columns are adjusting their widths by themselves. If there is a demand of larger columns, they can be merged to larger columns. Depending on the screen size, Bootstrap enables to define the size of the merged columns. The adjustable size feature has the advantage that on smaller screens more columns can be merged so that the whole content is still displayed in a row.

The IRP software uses the **Tabs** feature on almost every page. The advantage of tabs is that they provide an effective overview of the content. This is especially important if information units are connected to each other, but the number of data entries is too high to fit on one page.

To obtain an effective overview of the content, not all information is displayed from the beginning. For this purpose, **Modals** are employed. The information is displayed in own dialogues after clicking the button. When the dialogue appears, the background turns grey and the focus of the user is directed to the modal. In the IRP application, modal dialogues are utilised for different purposes. For example, they are employed to confirm user decisions, such as the deletion of requirements, or to offer a selection dialogue for adding responsible stakeholders to a requirement or project.

The Bootstrap **DataTables**¹⁴ are based on basic HTML tables, but they are extended with additional features and styles. The functions for the tables are provided by an own library. DataTables enable displaying a large number of connected data in a structured and well formatted way. Furthermore, the table provides a pagination function, such that not all data rows are listed at once. The rendering is done by

¹⁴<https://datatables.net/manual/styling/bootstrap> (Retrieved 01.10.2020)

the library and allows to modify each column of the table individually. These are only the most important features of Bootstrap. There are many more functions available in the framework.

3.2.13 D3.js

D3.js¹⁵) is a JavaScript (see Section 3.2.9) based library, which enables developers to visualise data in diagrams. The library also offers the possibilities to create tables or interactive SVG charts. The libraries base relies on basic web standards. Therefore, no additional frameworks are required, and the library can be employed in all modern web browsers. The framework binds data to the DOM and transforms them according to the defined settings. One of the main advantages of the library is the flexibility of the settings for charts. The data can be transformed in every imaginable chart and, if required, also be animated.

Examples and the libraries structure are presented in Chapter 7.

¹⁵<https://d3js.org/> (Retrieved 01.10.2020)

4 Used algorithms

Aggregation strategies (also called social choice functions) are important for group recommendation scenarios [Mas11; Fel+17]. These functions aggregate stakeholder preferences in a way that an acceptable suggestion for the stakeholders can be achieved [Mas11; Fel+17].

With the use of aggregation functions a stronger consensus between stakeholders can be achieved [Fel+11; Mas11]. Furthermore, the quality of decisions can also be increased [Ste+13; Ste+15].

In the following sections, five group recommendation algorithms (aggregation strategies) are presented. These are majority voting, average voting, median voting, least distance and MAUT.

These algorithms represent basic group recommendation heuristics and were already applied in various application domains [Mas11; FN12; Ste+13].

In the following sections, examples for the recommendation approach are provided. The examples consist of five stakeholders (S_1, \dots, S_5) and three requirements (R_1, R_2, R_3). In this context every stakeholder states his or her preferences with regard to the given set of requirements. Stakeholder preferences are represented as individual evaluations of interest dimensions. An evaluation is always associated with exactly one stakeholder. Stakeholders evaluate requirements on a scale from 1 (very unimportant) to 5 (very important). For an easier understanding of the heuristics, only one voting criterion is used in the example evaluation process, which is the *priority* criterion.

4.1 Majority voting

This algorithm focuses on the most popular item of an evaluation [Mas04a; Sen+11; Fel+18b]. Therefore, the algorithm chooses the preference with the highest evaluation [Mas11]. Even though the algorithm is simple, it is still powerful in group decision making, because a majority is behind a recommendation [HK05; Fel+11]. In the context of release planning, the algorithm returns a recommendation for a criterion, which represents the majority stakeholders [FN12]. Each recommendation

4 Used algorithms

is related to exactly on requirement.

An example of the recommendation of a priority is demonstrated in Table 4.1. In this example, it is shown that requirement R_2 should have priority 5, since this priority was preferred by the majority of the stakeholders.

Requirement	S ₁	S ₂	S ₃	S ₄	S ₅	Recommendation
R_1	1	3	2	5	2	2
R_2	5	4	5	5	2	5
R_3	2	1	1	3	2	1

Table 4.1: Recommendation of the priority, according to the majority heuristic.

Studies have demonstrated that this heuristic performs well with a small evaluation scale and that it is not possible to manipulate the recommendation [Fel+11; FN12; Jamo4; Nin+14]. The heuristic prevents the manipulation of the result, because it does not intensively consider individual evaluations. In other algorithms high or low evaluations can increase or decrease the overall rating dramatically.

4.2 Average voting

This algorithm calculates the average value (see Formula 4.1) for the group recommendation of a requirement [FN12]. For the calculation of the recommendation, the heuristic sums up all evaluations of the requirement, $evals(r)$, and divides the result by the number of the evaluations for the requirement. If the result is a floating number, the final result is, depending on the value, rounded (see Formula 4.2).

$$AVG(r) = \frac{\sum_{v \in evals(r)} v}{|evals(r)|} \quad (4.1)$$

$$Recommendation(r) = \begin{cases} \lfloor AVG(r) \rfloor & AVG(r) < 0.5 \\ \lceil AVG(r) \rceil & AVG(r) \geq 0.5 \end{cases} \quad (4.2)$$

An example of a recommendation based on average voting is presented in Table 4.2.

This heuristic performs effectively with homogeneous groups, where evaluations are similar to each other [Mas11]. If evaluations are diverse, the result of the recommendation is often not accurate [Mas11]. The problem is that average rating is manipulable and as a consequence, high or low evaluations can change the result dramatically [Jamo4].

Requirement	S1	S2	S3	S4	S5	Recommendation
R_1	1	3	2	5	2	3
R_2	5	4	5	5	2	4
R_3	2	1	1	3	2	2

Table 4.2: Recommendation of the priority, according to the average heuristic.

4.3 Median voting

This algorithm determines the median value (see Formula 4.3) for the group recommendation of a requirement. As a preparation for the calculation, all evaluations, $evals(r)$, are ordered from the smallest value to the largest. To detect the middle of the set, Formula 4.4 is applied. Depending on the set's count, n , whether or not it is even, the value, x , is set. If the set is even, the next larger value is chosen. The result of the heuristic is then the middle value of all evaluations [Mas11].

An example of a recommendation based on median voting is presented in Table 4.3. The algorithm shows that for this example 5 is the recommended priority of R_2 , because this evaluation is exactly in the middle of the ordered set of evaluations.

$$MED(r) = evals(r) \left[\frac{x}{2} \right] \quad (4.3)$$

$$x = \begin{cases} n + 1 & \text{odd} \\ n + 2 & \text{even} \end{cases} \quad (4.4)$$

Requirement	S1	S2	S3	S4	S5	Recommendation
R_1	1	3	2	5	2	2
R_2	5	4	5	5	2	5
R_3	2	1	1	3	2	2

Table 4.3: Recommendation of the priority, according to the median heuristic.

This heuristic is not manipulatable, because it separates the lower half from the higher half of the evaluations [Mas11]. Therefore, single divergent evaluations can not change the outcome. To change the result of this algorithm dramatically, more than half of the evaluations must be manipulated.

4.4 Least distance (LDIS) voting

This heuristic (see Formula 4.5) determines the priority, p , with the lowest overall distance to other evaluations, v , in a set of evaluations, $evals(r)$, for a requirements group recommendation [Ste+15]. If there exist two or more lowest distance priorities, the lowest priority is then chosen for the recommendation.

An example of a recommendation based on the priority evaluation criterion is presented in Table 4.4. The algorithm shows that for this example 5 is the priority recommendation for R_2 , because the sum of the distances of this evaluation has the lowest value.

$$LDIS(r) = \underset{v \in evals(r)}{argmin}(p, \sum |v - p|) \quad (4.5)$$

Requirement	S1	S2	S3	S4	S5	Recommendation
R_1	1	3	2	5	2	2
R_2	5	4	5	5	2	5
R_3	2	1	1	3	2	2

Table 4.4: Recommendation of the priority, according to the least distance heuristic.

Studies have demonstrated that the least distance algorithm provides good results for homogenous group decisions [FN12; Mas11].

4.5 Multi Attribute Utility Theory (MAUT)

To prioritise requirements, the priority criterion alone is mostly not sufficient. There are also other important factors such as *feasibility*, *cost* or the *risk* for the implementation. To take into account all these criteria for the recommendation, the MAUT model [KR93] can be applied. The model supports the derivation of a utility value in group-based settings [Ste+15; Fel+18a]. The formula for the calculation of the utility value is shown in Formula 4.6.

To distinguish between the criteria and to define their importance, each criterion, c , has an assigned weight, $w(c)$. For example, it is more important that a release is feasible than a maximised profit. Also, to distinguish between the evaluation importance of stakeholders for a requirement, r , stakeholders can also have different weights, $w(v)$.

4.5 Multi Attribute Utility Theory (MAUT)

MAUT creates a sum of all individual MAUT values from all evaluations, $v \in evals(r)$. First, the sum of the preferences for all criteria is created. In this context $eval(v, c)$ refers to a requirement-specific preference of a stakeholder for a specific criterion, c . Each criterion is normalised to a scale 1..10 where 1 = very low and 10 = very high.

An example of the evaluation of requirements is presented in Table 4.5. To demonstrate the algorithm, the previous Table 4.4 has been extended with a second criterion, the *feasibility* (F) criterion. The bold evaluations indicate that a stakeholder has a higher importance for a requirement and therefore has a higher weight. The bold evaluations have a weight, $w(v)$, of 2 and all others a weight of 1.

$$Utility(r) = \frac{\sum_{v \in evals(r)} w(v) * \frac{\sum_{c \in criteria} eval(v,c) * w(c)}{\sum_{c \in criteria} w(c)}}{\sum_{v \in evals(r)} w(v)} \quad (4.6)$$

Furthermore, each criterion has its own weight, which are shown in Table 4.6.

Requirement	S1		S2		S3		S4		S5	
	P	F	P	F	P	F	P	F	P	F
R_1	1	2	3	3	2	3	5	3	2	2
R_2	5	4	4	3	5	3	5	5	2	4
R_3	2	1	1	3	1	2	3	2	2	2

Table 4.5: Example evaluations for the MAUT algorithm. P = Priority, F = Feasibility

	Priority	Feasibility
Weight	2	1

Table 4.6: Example weights of evaluation criteria.

$$\begin{aligned}
 Utility(R_1) &= \frac{\sum_{v \in evals(r)} w(v) * \frac{\sum_{c \in criteria} eval(v,c) * w(c)}{\sum_{c \in criteria} w(c)}}{\sum_{v \in evals(r)} w(v)} \\
 &= \frac{1}{1 + 2 + 2 + 2 + 1} * \left(1 * \frac{1 * 2 + 2 * 1}{2 + 1} + 2 * \frac{3 * 2 + 3 * 1}{2 + 1} \right. \\
 &\quad \left. + 2 * \frac{2 * 2 + 3 * 1}{2 + 1} + 2 * \frac{5 * 2 + 3 * 1}{2 + 1} + 1 * \frac{2 * 2 + 2 * 1}{2 + 1} \right) \\
 &= \frac{1}{8} * \left(\frac{4}{3} + \frac{18}{3} + \frac{14}{3} + \frac{26}{3} + \frac{6}{3} \right) = \frac{1}{8} * \frac{68}{3} = 2.83
 \end{aligned} \quad (4.7)$$

With the provided evaluations in Table 4.5, the utility values of the requirements are calculated. An example of the calculation of the utility value for a requirement,

4 Used algorithms

R_1 , is presented in Formula 4.7. The further utility values of the requirements are $R_2 = 4.16$ and $R_3 = 1.83$. According to the calculated utility values, the requirement R_2 has the highest relevance. R_2 is then followed by R_1 and R_3 .

During the optimisation process in the IRP software, the MAUT algorithm is employed for the prioritisation of the requirements (see Chapter 8.1).

4.6 Interface for further algorithms

There are many further group recommendation algorithms available [Maso4b; Mas11; Ata+18]. In this context, the architecture of the IRP system was chosen, so that the implementations of existing recommendation algorithms can be easily added, extended, or replaced.

5 Procedure of release planning

The demand from customers for new and improved product features is omnipresent [Ruh10]. The selection of the appropriate features for the next releases to satisfy customer needs is often challenging. Many parameters must be considered in the context of feature selection, which is the major motivation for intelligent release planning support.

The assignment of requirements to releases during release planning must be accomplished in such a way that all conditions such as risk and technical constraints are satisfied [ARSo4].

Ruhe and Saliu [RS05, p.47] have stated the four main characteristics of an effective release plan:

- 'provide maximum business value by offering the best possible blend of features in the right sequence of releases,
- satisfy most important stakeholders involved,
- be feasible with available resources, and
- reflect existing dependencies between features.'

These characteristics show that many specifications must be fulfilled and considered for an effective release plan. First, the time horizon and objects for the planning must be identified [RM05]. Furthermore, the dependencies of the requirements and releases as well as the involvement of stakeholders during the planning process must be considered.

Deciding which requirement should be implemented in which release is often a challenge. Existing dependencies between requirements are particularly difficult to detect and require the involvement of special dependency detection functionalities [Fel+17; Tsa14].

On the other hand, an effective release plan can save up to 30% of the product development budget [Ruh10]. According to Ruhe [Ruh10], a release plan also has further benefits:

- Reduction of risk: With continuous planning, future risks can be detected more easily and quickly. There is more time to identify countermeasures to reduce risks [Ruh10].

5 Procedure of release planning

- Reduction of uncertainty: By analysing the release plan, ambiguities can be detected and reduced [Ruh10].
- Support for better decisions: By gathering information during planning and defining the responsibilities for the completion of tasks, more effective decisions can be made [Ruh10].
- Providing information: When being confronted with the release plan, all stakeholders receive an overview of the next steps of the product [Ruh10].

A release plan also provides information regarding future directions and dependencies in the project [Ruh10]. These benefits and advantages demonstrate that a detailed release plan is important. However, to achieve these benefits, continuous investment in planning must be made.

In this context, the IRP system aims to provide the following support:

- support managers in the definition of requirements,
- support managers in the recognition of dependencies,
- support managers in the involvement of stakeholders in the requirement prioritisation process,
- support stakeholders with recommendations,
- support stakeholders in the identification of a consensus, and
- support managers in the optimisation of releases.

The following sections present the procedure of release planning in the IRP application. Section 5.1 provides a complete overview of the planning process. The follow-up sections present all release planning steps in detail. Thereafter, special functions such as conflict detection (see Chapter 6) and release optimisation (see Chapter 8) are presented.

5.1 Planning flow

The IRP process consists of nine steps. The whole planning flow with all its steps and their connections is shown in Figure 5.1.

1. **Planning Project Settings:** Defining all necessary project parameters. This step includes the definition of all involved stakeholders. Furthermore, the project's *start* and *end* date as well as a formal description of the project must be defined. Furthermore, all evaluation dimensions for the requirements (see Section 5.5.4) are defined here.
2. **Define releases:** All releases which must be planned, are created with the key parameters of releases. These key parameters consist of a *start* and *end* date of

5.1 Planning flow

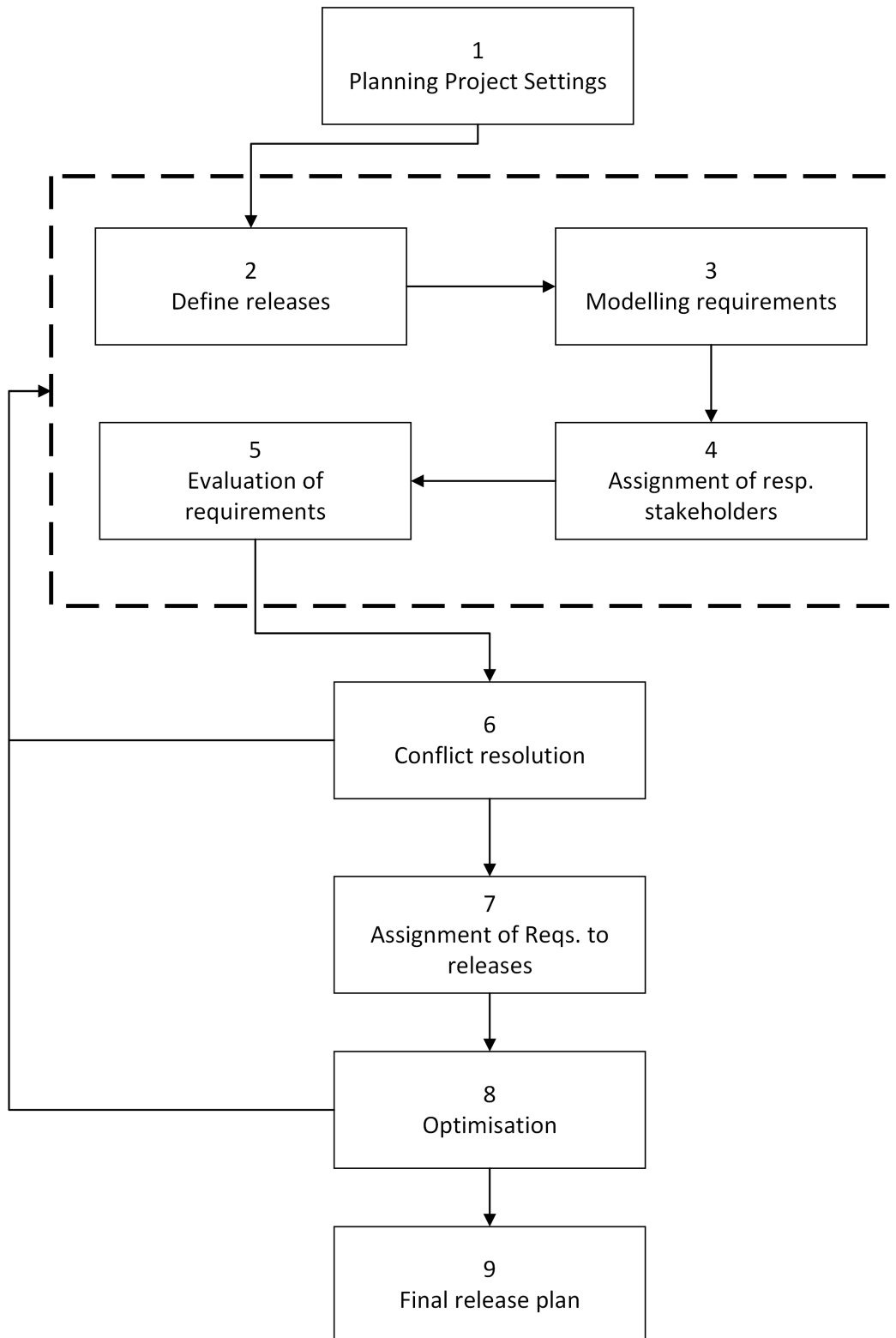


Figure 5.1: Overview of the IRP release planning flow with all steps and connections.

5 Procedure of release planning

the release, a formal description, and the available resource capacity for the release.

3. **Modelling requirements:** In this step, the requirements with their key parameters, which are candidates for the next releases, are created. The parameters are an exact description of the requirement and the already known dependencies between the requirements.
4. **Assignment of responsible stakeholders:** Requirements can have responsible stakeholders. These stakeholders should have solid background knowledge about the requirements.
5. **Evaluating requirements:** Every stakeholder can evaluate requirements. Related decisions can be supported by recommendations. During the evaluation, stakeholders can present positive or negative comments (see Section 5.5.6) about a requirement.
6. **Conflict resolution:** During the evaluation process, conflicts between individual stakeholder preferences can occur. These conflicts are presented, and stakeholders should try to achieve a consensus.
7. **Assignment of requirements to releases:** The project manager can manually assign requirements to releases.
8. **Optimisation:** The system displays optimisations for the releases in order to achieve a higher business value. In addition, the current assignment of requirements to the releases is analysed and further conflicts are revealed.
9. **Final plan:** At the end, a final plan is created.

Steps 6 and 8 demonstrate that the process can also move backwards. If a refinement is needed, managers can return to the modelling and evaluation steps. In the following sections, a detailed description of the individual steps is given.

5.2 Registration and login

When the user enters the application URL, a login page is displayed, which is presented in Figure 5.2. If the user had no account until now, he or she can switch the active tab to the Register tab.

All actions in the IRP software require a valid account. Therefore, before a user is allowed to use the IRP software and collaborate on projects, he or she must register. For registration purposes, new users must enter following personal information:

- first name,
- last name,
- email,
- username,

- password,
- confirm password, and
- password reset question.

The system determines if the entered username is available. The password must be at least six characters long. If one of the criteria is not fulfilled, an error message is displayed to the user. If the user forgets his or her password a question must be answered to reset the password. Therefore, the user must create a reset question and a corresponding answer. After successful registration, the user can log in.

The screenshot shows the login page of the OpenReq Intelligent Release Planner. At the top left is the OpenReq logo, and at the top right are the language options 'EN | DE'. The main heading is 'Intelligent Release Planner'. Below this is a welcome message: 'Welcome to the Intelligent Release Planner. Sign in or register to start planning releases or to collaborate in existing projects. The planner guides you through the whole release planning process. It starts by the creation and prioritization of new requirements. And ends by the visualization of the assigned requirements, with its relevance values for the releases and with suggestions for improvements.' There are two buttons: a large teal 'Login' button and a smaller blue 'Register' button. Below the buttons is the 'Login' section, which includes a 'Username' input field, a 'Password' input field, an orange 'Login' button, and a blue link for 'Forgot password?'.

Figure 5.2: Login page of the IRP software.

5.2.1 Profile Page

After the successful registration of the user, he or she is always able to edit his or her personal information. To achieve this, the user must select the 'Profile Settings' option of the menu of the header, which is illustrated in Figure 5.3.

In the header, the user is also able to change the language of the application. At the

5 Procedure of release planning

moment, German and English are available, whereas the primary language of the software is English.



Figure 5.3: Header of the IRP application with the profile settings menu.

After the user is on the profile page, which is presented in Figure 5.4, he or she can adapt his or her personal information.

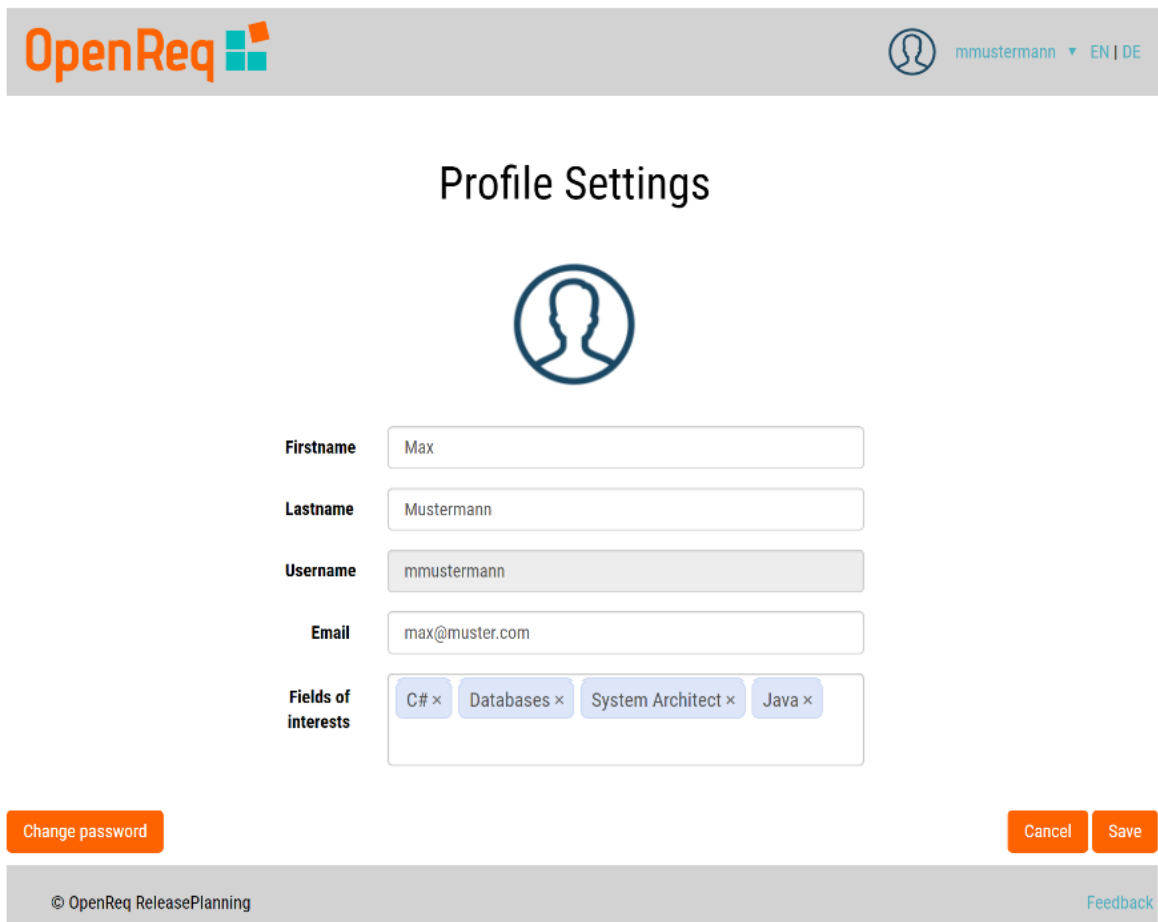
The image shows the 'Profile Settings' page in the OpenReq application. The header is the same as in Figure 5.3. The main content area has the title 'Profile Settings' and a profile icon. Below the icon are several form fields: 'Firstname' with the value 'Max', 'Lastname' with 'Mustermann', 'Username' with 'mmustermann', and 'Email' with 'max@muster.com'. There is also a 'Fields of interests' section with four tags: 'C# x', 'Databases x', 'System Architect x', and 'Java x'. At the bottom, there are three buttons: 'Change password', 'Cancel', and 'Save'. The footer contains the copyright notice '© OpenReq ReleasePlanning' and a 'Feedback' link.

Figure 5.4: Profile page of 'Max Mustermann', a user of the IRP application.

Besides the data which were entered during the registration, the user can also add

a profile picture and enter his or her skills.

The profile picture of the current logged-in user is always displayed in the header of each page. Furthermore, the profile pictures are also displayed in the tables of assigned stakeholders (see Section 5.4.1).

In the 'Skills' text box, tags can be added. These tags describe and highlight users' special knowledge. For example, if the user is an expert in creating database schemes and queries, that user has a 'Database' tag. With the help of those tags, managers are supported during the selection of stakeholders (see Section 5.4.1) responsible for specific requirements.

On the profile page, the user can change the password.

5.3 Projects

A project contains all the requirements and releases for a specific product. A user in the IRP application can collaborate in various projects. The role of the collaboration can range from a domain expert to the manager of the project.

5.3.1 Project overview

After a successful login, the user is redirected to the project overview. All projects for which the user has been added as a stakeholder, are listed in this overview. Figure 5.5 presents an example of the project overview.

To provide the user a short overview of the projects, every record of the list displays the name, release statistic, and start and end dates of the projects. The 'Releases finished' column compares the completed releases with the number of all available releases of the project.

Depending on the stakeholder's role, two different buttons are presented. If the user is the creator of the project, he or she can delete the project. If this is the case, a button with a trash bin is visible to the user. Otherwise, the user can only leave the project. After the stakeholder leaves the project, he or she is not able to see the project anymore, unless another stakeholder adds him or her again.

On the project overview page, the users are also able to create new projects using the 'Create project' button.

5 Procedure of release planning

OpenReq mmustermann ▾ EN | DE

Projects

Show entries Search:

Name	Releases finished	Date	
Recommendation Project	2/3	Jan 17 - Mar 19	
Intelligent Release Planer	0/0	Jan 18 - Apr 19	
OpenReq	0/0	Jan 18 - Dec 19	

Showing 1 to 3 of 3 entries Previous 1 Next

[Create project](#)

© OpenReq ReleasePlanning [Feedback](#)

Figure 5.5: Project overview of the IRP application.

5.3.2 Project detail

After clicking on a project in the project overview (see Section 5.3.1), the stakeholders are redirected to the detail page of the project. A project detail page always consists of four configuration tabs. The tabs are the following:

- Requirements (see Section 5.5),
- Releases (see Section 5.6),
- Issues (see Chapter 6), and
- General.

In the 'General' tab, configurations for the project can be made, which are illustrated in Figure 5.6.

The screenshot displays the 'General' tab of a project in the IRP application. The header shows the OpenReq logo and user information (mmustermann | EN | DE). The navigation bar includes tabs for Requirements, Releases, Issues, and General. The main content area is divided into several sections:

- Basic Settings:** Contains a large red 'iST' logo, a 'Name' field with the value 'Intelligent Release Planer', a 'Start date' field with the value '01.01.2018', an 'End date' field with the value '30.04.2019', and a 'Description' text area. A 'Save' button is located at the bottom right of this section.
- Requirement properties scheme:** A section for defining evaluation criteria.
- Stakeholders:** A section for listing project stakeholders.
- Statistics:** A section for project statistics.
- Attachments:** A section for project attachments.

The footer contains the copyright notice '© OpenReq ReleasePlanning' and a 'Feedback' link.

Figure 5.6: Detail page with the selected 'General' tab of a project in the IRP application.

This tab is grouped into five sections.

The first section, the 'Basic Settings', defines the name of the project and the start and end date. A description can be added to give the stakeholders an overview of the project. Furthermore, a picture can be assigned to the project. This picture is then presented in each list of the project.

The next group is the 'Requirement properties scheme'. Here, all MAUT evaluation criteria with a description are listed, as shown in Figure 5.7.

5 Procedure of release planning

The screenshot displays a web interface titled "Requirement properties scheme". It features a table with the following columns: Name, Weight, Max. Value, Invert result, and Description. The table contains four rows of criteria. The "Invert result" column has checkboxes, with the last three rows checked. At the bottom right of the table area, there are two orange buttons: "New Criterion" and "Save".

Name	Weight	Max. Value	Invert result	Description
Business Relevance	1,5	10	<input type="checkbox"/>	Business Relevance of the mentioned requirement.
Effort	2,0	50	<input checked="" type="checkbox"/>	Effort in hours to develop the mentioned requirement.
Risk	1,0	10	<input checked="" type="checkbox"/>	Risk of not being able to successfully implement the mentioned requirement
Cost	0,5	5000	<input checked="" type="checkbox"/>	Incurred costs for developing the mentioned requirement

Figure 5.7: Detail view of the 'Requirement properties scheme' section.

This group contains all the evaluation criteria for requirements. The project manager can add or remove criteria. Furthermore, detailed settings for the criteria can be defined. These settings can be evaluation scales and the weight of MAUT evaluation criteria (see Chapter 8). Further information regarding evaluation criteria is presented in Section 5.5.4.

The 'Stakeholders' group provides the possibility to add and manage all the stakeholders of the project. More information about the stakeholders is presented in Section 5.4.

The 'Stakeholders' group is followed by the 'Statistics' group. In this group, three statistics are available for the project. More information about related diagrams is provided in Chapter 7.

The last group is the 'Attachments'. Here, different project-relevant files can be provided for users. A detailed view of the 'Attachments' group is presented in Figure 5.8. Depending on the file type, the files are displayed differently.

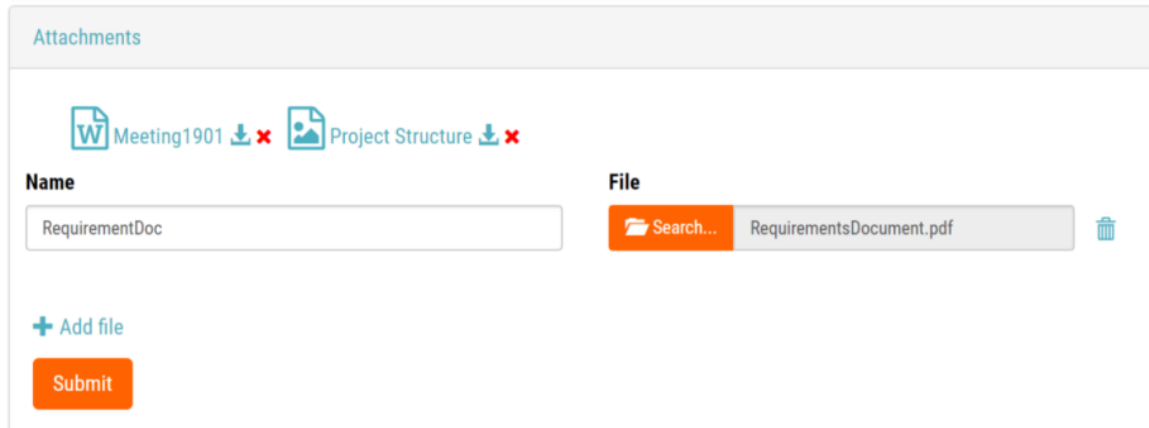


Figure 5.8: Detail view of the 'Attachments' section.

5.4 Stakeholder

A stakeholder is someone who is affected by or affects the outcome of the releases of a project [Ruh10]. As already mentioned, affected stakeholders often put pressure onto managers of a project to provide their favoured requirements earlier [Dag+05]. This shows that the decisions regarding the assignment of requirements to releases often can become complex [RM05].

Therefore, to achieve a feasible and effective solution, stakeholders must work together. Thus, it is preferred to involve them during the planning process. In the IRP application, stakeholders are involved during the modelling and prioritisation of requirements.

A problem of involving stakeholders during requirements prioritisation is that not every stakeholder has the same importance and knowledge about requirements and the project [RS05]. Therefore, different approaches exist to classify stakeholders and to give a higher weight to the evaluations of 'important' stakeholders [BRW01; RS05].

5.4.1 Assignment

To enable stakeholders to collaborate in a project, they must be added as stakeholders to the project, which can be achieved in the 'Project Detail' page in the 'General' tab. Figure 5.9 presents a view of a project with assigned stakeholders. In this view, the manager can add and remove stakeholders from the project.

To add stakeholders, the 'Add stakeholder' button must be pressed. Then, a modal

5 Procedure of release planning

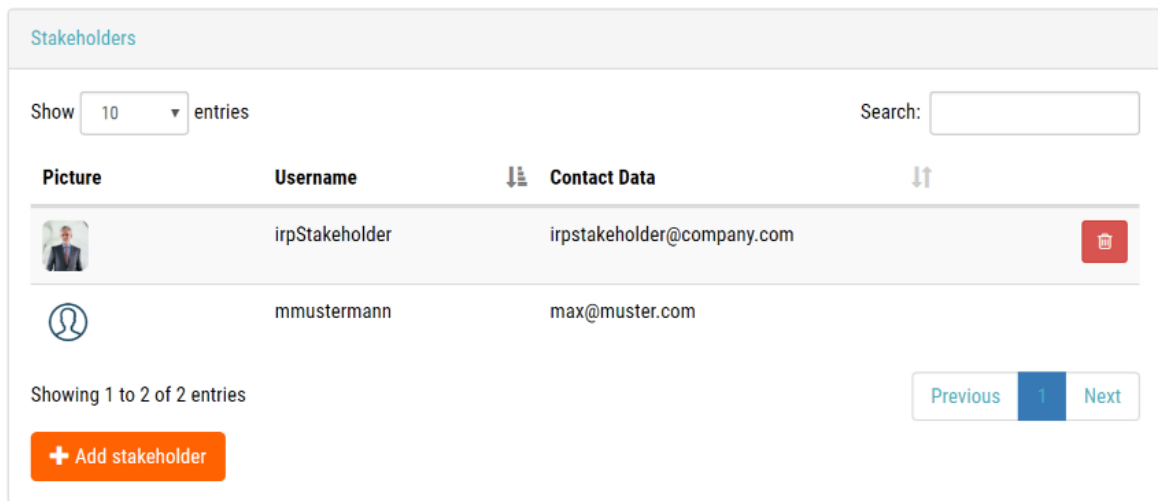


Figure 5.9: List of stakeholders, who were added to a project in the IRP application.

dialogue with all available stakeholders appears (see Figure 5.10). This list is filterable by the names and the skills of stakeholders.

The IRP application provides the possibility to add stakeholders as 'responsible stakeholders' to individual requirements, because of their knowledge or importance. These responsible stakeholders are more involved in the added requirements. If there is new information available for the requirements, such as further descriptions or additional attachments, the responsible stakeholders receive notifications. Furthermore, since the responsible stakeholders have more knowledge, their evaluations are given more weight during the prioritisation process (see Chapter 8.1). Only stakeholders, who were already added to the project can be assigned to a requirement as responsible stakeholders.

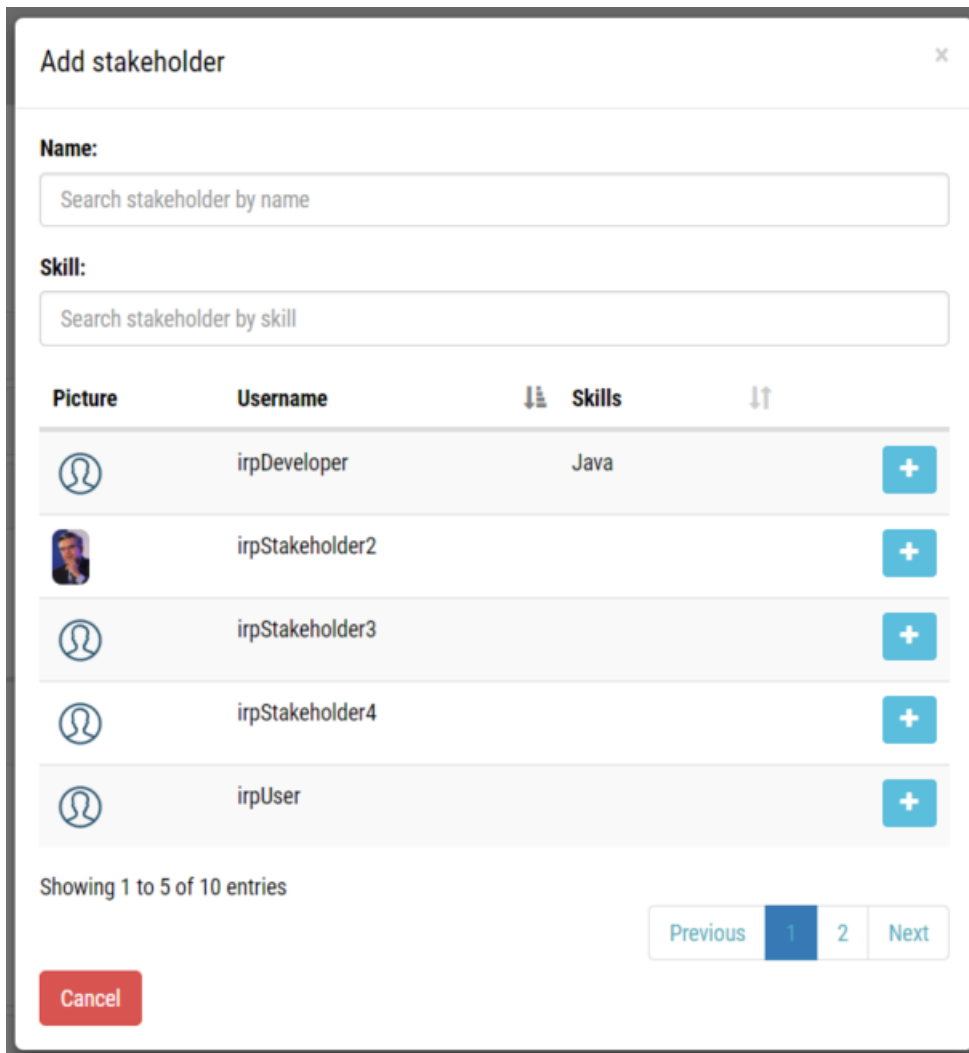


Figure 5.10: List of stakeholders, who can be added as responsible stakeholders to a project in the IRP application.

5.5 Requirements

Sommerville and Sawyer [SS97, p.4] have provided a definition for requirements:

'Requirements are defined [...] as a specification of what should be implemented. They are descriptions of how the system should behave, or of a system property or attribute. They may be a constraint on the development process of the system.'

Therefore, requirements can be viewed as decision alternatives for the functionality of the product [AW03].

5 Procedure of release planning

Features are utilised to provide communication and understanding between customers and developers [Ruh10]. Features are abstract and are often one or more capabilities of a system, which present a value to the users [WB13]. Usually, a feature consists of several requirements.

An example of a feature or requirement is 'Support for the consensus finding during the requirements evaluation process'. The abstract description is enough for a user but not for the developers. Therefore, a feature is divided into several smaller requirements to gain more detailed information. The division of the smaller requirements is implemented as follows:

- providing a chat with comment function,
- highlighting positive and negative comments with different colours, and
- providing a recommendation for each evaluation criterion.

5.5.1 Requirement overview

The first tab of the project details is the 'Requirements' tab. All requirements of the project are listed here. The overview with its classification of the requirements is presented in Figure 5.11.

A requirement can have four different states:

- new,
- planned,
- completed, and
- rejected.

In the 'new' state, a requirement was recently created and has not been assigned to a release. In the 'next' state, the requirement has been assigned to a release. In the 'completed' state, the work for the requirement has been finished. Requirements can also be 'rejected', which is possible, for example, if the requirement is out of the scope of the project or if the functionality is already provided by the product. 'Rejected' requirements are still listed in the overview to provide a history of all the rejected requirements.

The screenshot displays the 'Requirements' tab in the 'Projects/Intelligent Release Planer' application. At the top, the OpenReq logo is on the left, and the user 'mmustermann' is logged in on the right. The main navigation bar includes 'Requirements', 'Releases', 'Issues', and 'General'. The 'New Requirements' section shows a list of requirements with columns for 'Title' and 'Name'. Two requirements are listed: 'Req#0' with the title 'UI design of the login page' and 'Req#1' with the title 'User registration'. Each entry has a red trash icon. A 'Create Requirement' button is located below the list. Navigation controls for 'Previous' and 'Next' are also present. At the bottom, there is a footer with '© OpenReq ReleasePlanning' and a 'Feedback' link.

Figure 5.11: Overview of the 'Requirements' tab in the IRP application.

5.5.2 Modelling

The modelling of a requirement is performed in four sections. A requirement detail view, in which users can manage and model the requirement information, is presented in Figure 5.12. In the first section, 'Information', the users are able to define a short name for the requirement. After the creation of a requirement, the system assigns the requirement a unique ID. This ID should help the user to identify a requirement throughout the project. In addition, a textual description of the requirement must be added. Furthermore, users have the possibility to add tags to a requirement, which has the advantage that the requirement can be more appropriately classified by its attributes. Through the classification with tags, the system can provide an overview of the requirements.

In the 'My Evaluation' section, stakeholders can evaluate the MAUT criteria (see Section 5.5.4). More information and a detailed view of this modelling section is provided in Section 5.5.4.

Every stakeholder can evaluate requirements, but the requirements can also have

5 Procedure of release planning

The screenshot shows a web interface for managing requirements. It features a main form with the following sections:

- Information**:
 - Title**: A text input field containing "Req#1".
 - Name**: A text input field containing "User registration".
 - Description**: A text area containing the text: "To use the software a user has to register him- or herself. A user has to register with following information The register page is a tab on the login page."
 - Tags**: A list of tags, currently showing "UI x" and "HTML x".
 - State**: A dropdown menu currently set to "New".
 - Save**: An orange button located at the bottom right of the form.
- My Evaluation**: A section header below the main form.
- Responsible Stakeholders**: A section header below "My Evaluation".
- Attachments**: A section header at the bottom of the page.

Figure 5.12: Detailed view of a requirement in the IRP application.

'Responsible Stakeholders'. These stakeholders are added to a requirement due to their special knowledge or their connection to the requirement. The assigned user is then more involved during the modelling of the requirement.

He or she receives notifications (see Chapter 6) if some information has been changed or added. Furthermore, this user's evaluations are weighted higher than those of other stakeholders.

The last modelling section is 'Attachments'. Here, different relevant files for the requirement can be provided.

5.5.3 Dependencies

Dependencies are used to describe relationships between requirements [Fel+13; Tsa14]. The importance of dependencies has been demonstrated by a study by Carlshamre et al. [Car+01]. The study, which analysed software projects, has revealed that only about 20% of the analysed requirements had no dependency on other requirements. This indicates that dependencies are important during the planning process and that they must be carefully considered during the assignment of requirements to releases. Further impacts include a dramatic increase in the follow-up costs if dependencies are identified to late [Fel+17].

In the IRP system, stakeholders can model three different types of dependencies:

- requires,
- coupled, and
- excludes.

The 'requires' or 'weak precedence' dependency defines that for requirements A and B, B must be in the same release as A or at least in a previous release [Ruh10; VMT+07], which means that B cannot be later than A. Such a dependency makes sense, for example, if there are requirements such as 'Overview of registered users' (A) and 'Registration of users' (B). If A requires B, the dependency declares that requirement A makes only sense, if users can already register on the system.

A 'coupled' dependency defines that for requirements A and B, both requirements should be in the same release because of the strong connection to each other [Ruh10]. In this case, requirement A requires B, and B requires A. An example of such a dependency is that a user has to log in to a system (A), but to be able to log in, he or she must register (B) first. Therefore, these requirements are only meaningful if they are implemented in the same release.

The 'excludes' dependency, expresses that requirement A or B is allowed in a release, but not both [VMT+07; VDL98]. This dependency defines, that if requirement A is in release C, requirement B cannot be in the same release and vice versa.

The modelling of the dependencies in the IRP system can be made in the 'Dependency' tab of the requirement detail view and is illustrated in Figure 5.13.

5 Procedure of release planning

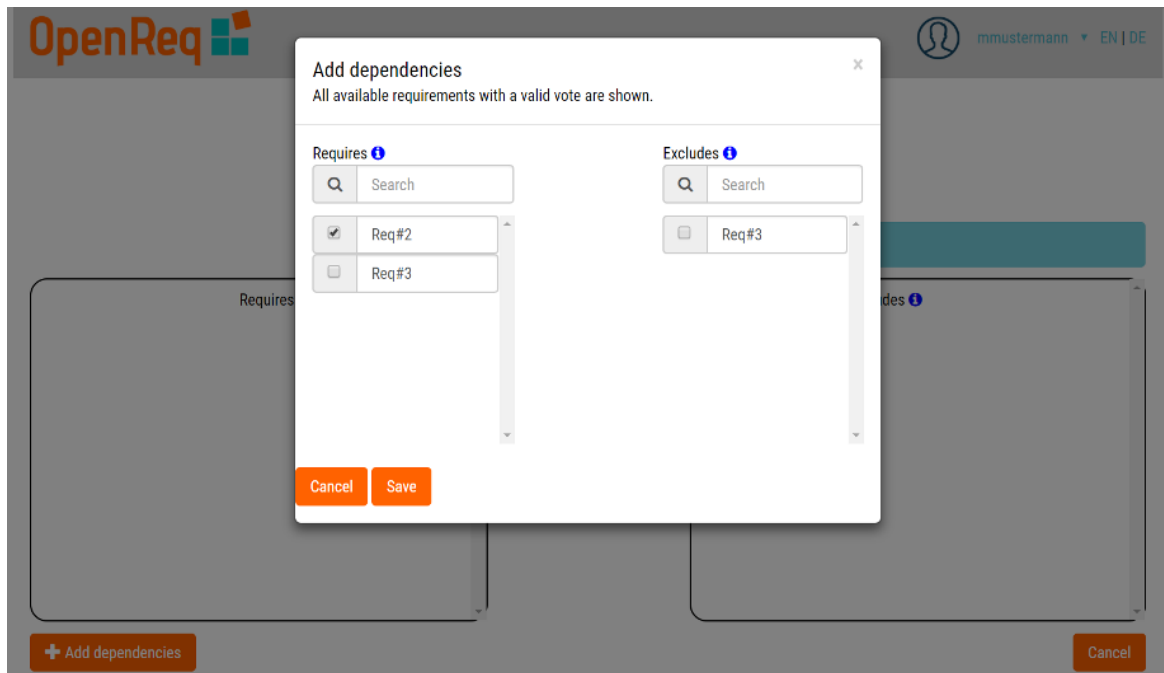


Figure 5.13: Detail view of an assignment of a dependency to a requirement in the IRP application.

5.5.4 Evaluating a requirement

Providing the ideal solution that satisfies the customers' expectations is not easy. Many aspects such as timelines, resources, and especially costs must be considered during planning. Knowing which requirement should be assigned to which release is a difficult process. Therefore, different techniques for the prioritisation of requirements exist [Dua+09].

One of these techniques is to categorise requirements in three different priority groups [Brago; Dua+09]. Such a categorisation can be mandatory, desirable, or inessential. Requirements which must be in a software such that many stakeholders are satisfied are mandatory. If there are not enough mandatory requirements included in the software, stakeholders and customers will not accept the software. All requirements which are marked as 'desirable' have a benefit for the software but are not necessary. These requirements increase the satisfaction level of stakeholders but not as much as mandatory requirements.

Easy implementation and usage are benefits of this categorisation, but the small evaluation scale can be a problem for larger and more complex projects [Dua+09].

Other techniques use scoring methods to calculate a priority value [Dua+09]. One of these techniques is described by Wiegers [Wie99]. In this technique, the

manager evaluates requirements on a scale from 1 to 9. The evaluation is based on the values of cost and risk of the implementation. The problem in this approach is that during the evaluation of the requirements, no stakeholders are involved, so the manager does not know if the customers will be completely satisfied at the end.

Other techniques extend the presented approaches through the involvement of stakeholders during the evaluation process. With these techniques, stakeholders can evaluate a specific criterion such as the priority of a requirement. After completing an evaluation step and with the help of maximisation algorithms, the evaluated requirements can be prioritised [RS05; Ruh10].

In the IRP application, all stakeholders are encouraged to evaluate the available requirements. The prioritisation is based on a multi-score method, which means that all requirements have more than one interest dimension. The available criteria with their evaluation scales are introduced in the following section. To maximise the outcome of the evaluation process and to obtain the best result for each requirement, stakeholders are supported by group recommendations, which are described in Section 5.5.5.

Evaluation criteria (interest dimensions)

The selection of requirements for a release is difficult. The challenge for the selection is to obtain the best requirements for a specific release. Therefore, requirements must be prioritised. To create an appropriate prioritisation, various prioritisation criteria must be considered [Ruh10; Fel+13]. Example evaluation criteria are the *urgency* of a requirement and various types of *risks* such as *user acceptance*. Further criteria, which should be considered for the prioritisation, are *costs* and *effort* of the implementation of the requirement.

The IRP application provides three evaluation criteria at the start of the project, which are listed in Table 5.1. These criteria are defined by the system and cannot be deleted, but managers are able to add further criteria (see Section 5.3.2).

An example of the evaluation section of the requirements is displayed in Figure 5.14. This figure demonstrates, that a manager has added an additional evaluation criterion: *cost*. An indicator displays the status of an evaluation. If it is green, the evaluation of the current criterion is consistent with the evaluations of other stakeholders. If there is a conflict between the evaluations of different stakeholders, the indicator turns red, and an issue is created (see Chapter 6).

The criteria have different evaluation scales, which enable an improved user experience, and the system obtains more detailed information about the criteria where necessary. An overview of the scales, which are created by the system, with a

5 Procedure of release planning

The screenshot shows a 'My Vote' section with four evaluation criteria, each with a slider and a green checkmark indicating a successful vote. The criteria and their values are:

Criterion	Value	Status
Business Relevance	9	✓
Cost (€)	487	✓
Effort	9	✓
Risk	2	✓

An orange 'Vote' button is located at the bottom left of the section.

Figure 5.14: Detail view of the evaluation section with all evaluation criteria of a requirement in the IRP application.

corresponding description are presented in Table 5.1. Except for the effort criterion,

Name	Description	Scale
Business relevance	Business relevance of the mentioned requirement.	1 =>Low relevance; 10 =>High relevance
Effort	Effort in hours to develop the mentioned requirement	1 =>Low effort; 50 =>High effort
Risk	Risk of not being able to successfully implement the mentioned requirement	1 =>High risk; 10 =>Low risk

Table 5.1: System evaluation criteria of a requirement with their descriptions and scales.

each criterion has the same 10-point evaluation scale. The maximum implementation effort of a requirement can be 50 hours, which corresponds to one and a half working weeks of a developer. If the effort exceeds 50 hours, it indicates that the requirement is too complex and must be split up into smaller pieces. These settings can be changed by the manager.

5.5.5 Recommendation

This section presents different decision support approaches and the recommendation technique, which is integrated in the IRP application.

Decision support or recommender systems are already used in different environments (e.g., health care or logistics) and are often in action for complex or dynamic areas [Ruh10]. These systems try to generate more transparent decisions and to provide new decision alternatives for all stakeholders.

Burke [Buro2] has described a recommender system as:

‘any system that produces individualized recommendations as output or has the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options’.

Furthermore, recommender systems can identify and provide items to users when the environment is already too complex and not easy to understand for a user, therefore he or she is not able to make a decision [Buro0; BFG11; Nin+14].

One of the key purposes of group recommender systems is to provide recommendations in group decision scenarios such as software release planning [Fel+11]. Furthermore, these systems are often able to visualize the preferences of group members and to reveal conflicts. With the help of those systems, quality assurance can be improved [Fel+13]. However, it must be considered that recommendation systems are only supportive technologies and are not able to resolve inconsistencies. For this reason, communication between the individual stakeholders is still important and cannot be automated by systems. Consequently, release planning processes depend heavily on the information which is provided and exchanged by stakeholders.

Recommendation techniques

In this section, five types of recommendation techniques are presented. These techniques are used to support stakeholders in their decisions. The techniques are content-based filtering (CBF) [PB97], clustering [Wit+16], collaborative filtering (CF) [Kon+97], group recommendation (GR) [JBK04] and knowledge-based recommendation (KBR) [Buro0; FBo8].

In a CF [Kon+97] recommender system, items are suggested to a stakeholder, which were already rated highly by similar users. This technique is a frequently used recommendation technique and collects the ratings of nearest neighbours of the current user [Fel+13; Nin+14]. The neighbours of a user can be users with a

5 Procedure of release planning

similar rating behaviour for similar products, which means that if users A and B evaluated requirements in a similar fashion, user A also receives suggestions for requirements which were rated positively by B [Nin+14].

Collaborative filtering also supports the stakeholders in their understanding of requirements. The system can present information of stakeholders, which is helpful during the evaluation process. Furthermore, other requirements, which might also be interesting for a stakeholder, can be identified by the recommender system. A basic implementation of CF is user-based CF [Kon+97]. In this implementation, the k-nearest neighbours are determined. In user-based CF, neighbours are users who are interested in similar requirements [Kon+97]. The technique creates a prediction of a rating for the stakeholder with the help of the votes of the nearest neighbours.

Another recommendation technique is content-based filtering (CBF) [PB97]. In this approach, previous preferences for items are evaluated to recommend new items [Fel+13]. Examples of preferences can be previously viewed categories or frequently used keywords. An example of this approach is online shops. After the customer has bought or viewed an item of a specific category, the system recommends further products of this category.

In the context of release planning, this approach can be used to determine similar requirements or to suggest previous requirements of other projects for reuse purposes.

Clustering can be used to identify similar items. An example of clustering is the k-means algorithm [HTF09]. The number of clusters, which are searched by the algorithm, is described by k.

In the context of release planning, clustering can be used to find similar requirements. For example, to determine the similarities between requirements, it is possible to extract the textual description. Then, with the help of the descriptions and distance metrics, clusters can be formed by the algorithm. The result provides similar requirements in the individual clusters.

With the help of knowledge-based recommendation (KBR), a group of items can be suggested to users [Buroo]. Different formal knowledge types are evaluated for the determination of these items [Nin+14].

For the recommendation of items, pre-defined rules are used by the system [FB08; Fel+13]. The rules allow to explain why items have been recommended and/or why no recommendations could be found [Fel+09]. A user specifies some criteria, and the system recommends an item according to those criteria.

In the context of release planning, this technique can be used to assist consistency management [Fel+09; Fel+10a]. Consistency management is important in cases where it is not possible to determine a release plan. This situation is possible when the preferences of stakeholders are contradictory.

The last technique discussed here is group recommendation [JBK04; Fel+11]. These technologies recommend items to a group of stakeholders. For example, the recommender system suggests a restaurant for dinner to a group of users.

Group recommenders try to achieve or at least support a consensus among all group members [Nin+14]. By considering different aspects of decision-making, such as the evaluation of decision alternatives, group recommender systems support the decision process [JBK04; Nin+14]. The technique uses heuristics to determine relevant alternatives for the group.

In the context of release planning, a group recommender system is important for the evaluation and selection of requirements. The reason is that the system attempts to foster consensus between the stakeholders by providing a plan acceptable for all stakeholders.

Recommendation approaches in IRP

In the IRP application, requirements are specified on a textual level. Stakeholders can evaluate each requirement against various evaluation criteria. The application provides a recommendation for each criterion.

For this purpose, group recommendation techniques are used to support stakeholders in the construction of their preferences. For the support of the requirement evaluation, the system uses basic recommendation functionalities, which are based on heuristics. A recommendation of a criterion (evaluation dimension) of a requirement in the IRP application is illustrated in Figure 5.15.

The recommendation is only visible to the stakeholders when they hover over the blue information icon. Then, a small hint is displayed to the user with a basic description and a recommendation for the current criterion.

To recommend a evaluation value to stakeholders, a combination of two algorithms is used (see Formula 5.1). The two algorithms were already presented in Chapter 4, the median heuristic (see Section 4.3) and the least distance method (see Section 4.4).

$$Recommendation(r) = \begin{cases} Median(r) & conflict(r) == true \\ LDIS(r) & conflict(r) == false \end{cases} \quad (5.1)$$

Depending on the evaluation condition of the requirement, the system uses the heuristic with the most appropriate aggregation approach. First, the system checks whether a conflict exists for a specific criterion. If there is a conflict, the median heuristic (see Section 4.3) is used by the system; this has the advantage that the conflicting evaluations do not affect the recommendation too much. Due to the fact that the least-distance heuristic (see Section 4.4) is more manipulatable, this heuristic is only applied if there is no conflict for the given criterion [FN12].

5 Procedure of release planning

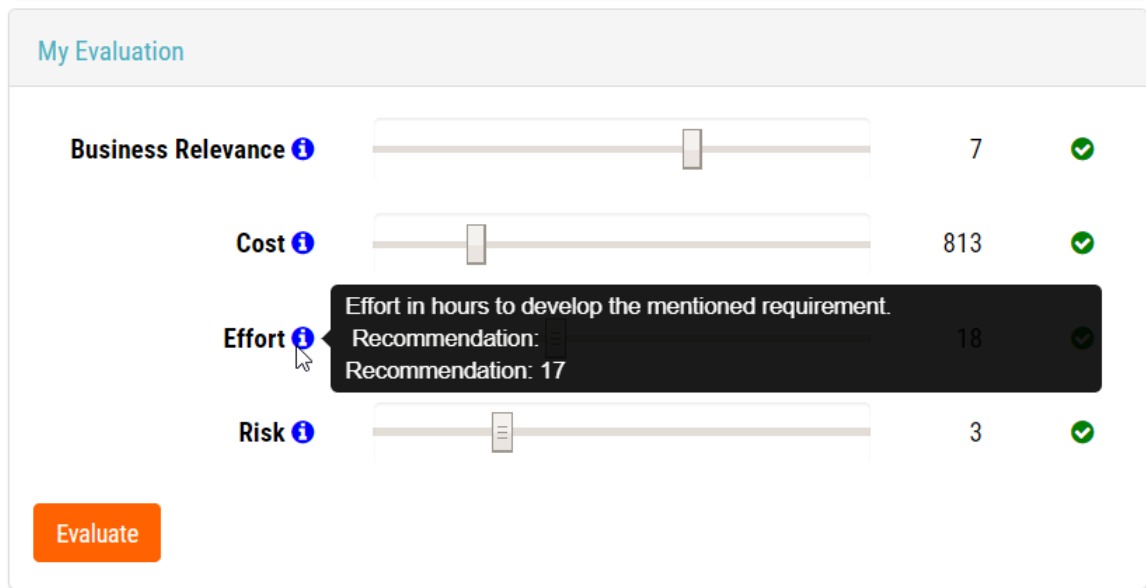


Figure 5.15: Recommendation of the effort criterion in the IRP application.

5.5.6 Support for finding a consensus

The recommendations depend on a high quality of information contributed by stakeholders. Effective release planning heavily relies on the exchange of information and the interaction between stakeholders [PCo8; Fel+13].

Therefore, in addition to the recommendation of individual criteria, the IRP system provides a chat to support the identification of a consensus. This chat relieves the communication and discussions between the stakeholders. Each requirement has its own chat in the detail view, which is displayed in Figure 5.16. This feature allows stakeholders to discuss each requirement separately.

A comment can be positive, negative, or neutral. The messages are highlighted differently depending on the type of message. Positive messages are marked green, negative ones are marked red, and neutral ones are marked grey. With this method, stakeholders can see at first glance the different meanings of the comments.

If a new chat message for a requirement arrives, all the responsible stakeholders are informed. More information about notifications is provided in Chapter 6.

Every chat message consists of five parts. The profile picture of the transmitting stakeholder is displayed on the left side of a message. Next, the username with the chat message is visible.

An important key feature of the chat is that stakeholders can provide their support for comments. The support of stakeholders is displayed by likes or dislikes of

Comments

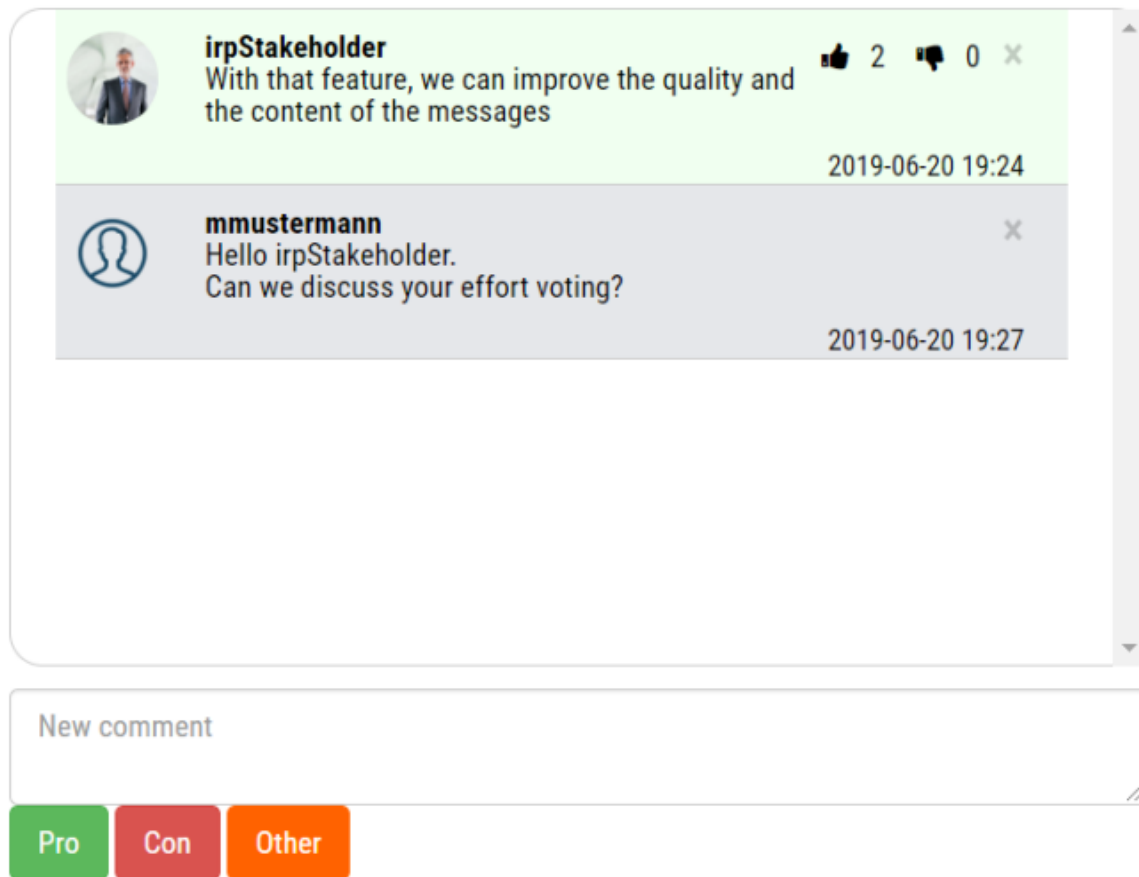


Figure 5.16: Chat regarding a requirement of the IRP application.

comments. This way, important comments can be highlighted, and the consensus of the stakeholders can be indicated. Positive and negative comments are also influencing the prioritisation of requirements (see Chapter 8).

Furthermore, every chat message contains a time stamp. The messages are sorted chronologically, meaning that the newest messages are at the top of the chat window. This sorting allows the stakeholders to see the newest messages first.

5.6 Releases

Using release planning, smaller development periods are performed sequentially [GR04]. As a result, instead of only one big release with all functionalities, smaller releases are published. The greatest benefit of this method is that more important

5 Procedure of release planning

requirements can be delivered first. Therefore, the definition of releases is important because they specify a schedule for the implementation of requirements [REP03]. Important properties, which must be considered for the releases, are the development time and resources, such as the capacity and staff availability.

5.6.1 Release overview

The second tab of the project detail view is the 'Releases' tab. In this tab, all the created releases of the current project are listed. The release overview with its classification of the releases is illustrated in Figure 5.17. The form contains four different sections with tables. Each section represents the state of the corresponding releases. Therefore, a release has four possible states:

- new,
- planned,
- completed, and
- rejected.

The screenshot displays the 'Releases' tab in the OpenReq Intelligent Release Planner. The interface features a header with the OpenReq logo and user information (mmustermann, EN | DE). The main title is 'Projects/Intelligent Release Planer'. Below the title are four tabs: 'Requirements', 'Releases' (selected), 'Issues', and 'General'. The 'New releases' section shows a table with one entry: 'Release 01' with a date range of 'Jan 18 - Feb 18' and an available capacity of '300' hours. The table has columns for 'Title', 'Name', 'Date', and 'Available capacity (h)'. Below the table is a 'Create Release' button and a 'Showing 1 to 1 of 1 entries' message. The bottom section contains links for 'Planned releases', 'Completed releases', and 'Rejected releases'.

Figure 5.17: Release overview in the IRP application.

In the 'new' state, a release was recently created, and no requirement has been assigned to this release yet. In the 'planned' state, at least one requirement has been assigned to the release. In the 'completed' state, all the requirements have

been finished. Releases can also be 'rejected', which is possible, for example, if the release is out of scope of the project or the release was mistakenly created. They are still listed in the overview to provide a history of all rejected releases. If a release is rejected and already had some requirements assigned, then all requirements will be removed from this release and receive the state 'new'.

Furthermore, the list enables stakeholders to see at first glance the available capacity and the planned implementation duration of releases. Each table also provides a search and sort function for the releases.

5.6.2 Release detail

By clicking on a release in the release overview, a detail view is displayed. An example of a detail view is presented in Figure 5.18.

Each release has some basic properties: ID, name, start and end dates, maximum capacity, status, and a description. The ID always starts with 'Rel#', followed by an individual number. The start and end dates of the release indicate the development period of a release. To be able to plan resources, each release also has its own maximum capacity, which is described in hours. This capacity implies that the sum of all the added requirements is not allowed to exceed this value. Furthermore, each release features its own 'Attachment' section, where further information can be added. Another field on the form is the status field. This is a read-only field that is set by the system through different events.

The managers are also able to reject a release. By pressing the button 'Reject', the status of the release is set to be rejected, and all assigned requirements are removed from the release.

If the implementation of the assigned requirements of a release has been finished, the release can be completed by the managers. For this purpose, the button 'Complete' is available to the managers. This button is only visible if the release is in the 'planned' state. After the activation of the button, the release and all its assigned requirements are set to the 'completed' state.

The screenshot shows the OpenReq application interface. At the top left is the OpenReq logo. At the top right, there is a user profile icon for 'mmustermann' and language options 'EN | DE'. The main heading is 'Projects/Intelligent Release Planer' followed by 'Rel#0'. Below this, there are two tabs: 'General' (active) and 'Assigned Requirements'. The 'General' tab contains an 'Information' section with the following fields:

Title	Rel#0
Name	Release 01
Start date	01.01.2018
End date	01.02.2018
Maximum capacity (h)	300
Status	New

Below these fields is a large text area for 'Description'. At the bottom right of the 'Information' section is a 'Save' button. Below the 'Information' section is an 'Attachments' section. At the bottom right of the entire form are 'Cancel' and 'Reject' buttons.

Figure 5.18: Release detail view in the IRP application.

5.6.3 Assignment of requirements

The second tab ('Assigned Requirements') of the release detail view enables managers to add requirements to the release. An example of the view with an 'Add Requirement' dialogue is presented in Figure 5.19.

The 'Assigned Requirements' view contains a table with all the requirements which were assigned to this release. With the button 'Add Requirement', a dialogue appears where all requirements in the state 'new' and with valid evaluations are listed. The requirements are ordered by their IDs. Multiple requirements can be added at

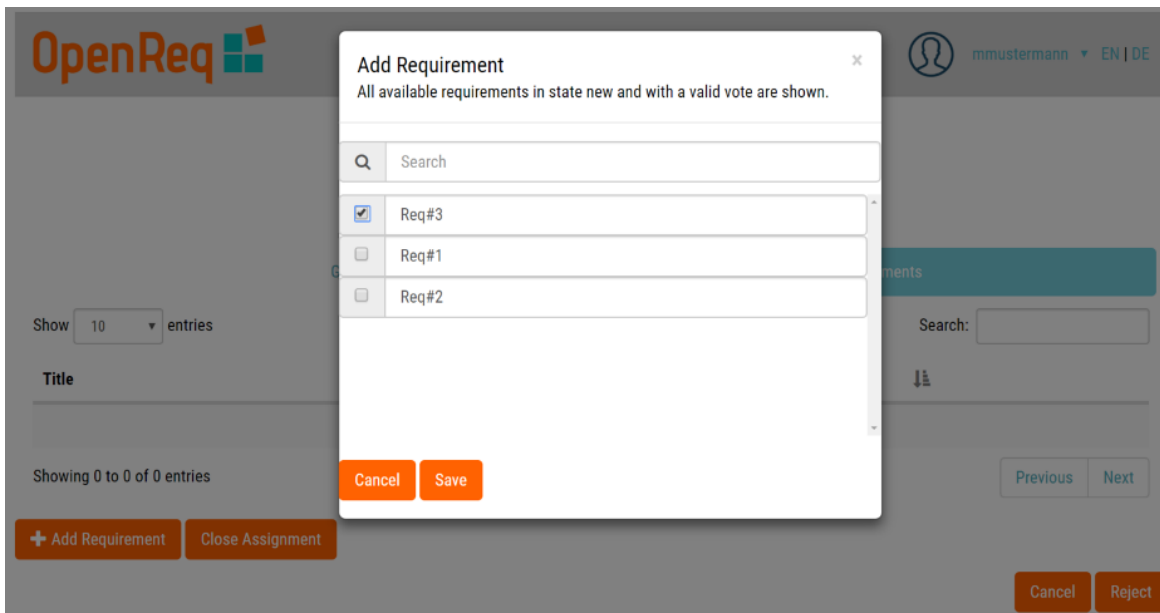


Figure 5.19: The assignment of a requirement to a release in the IRP application.

once to the release. As soon as requirements have been added to a release, the state of the requirements changes to 'planned'. The status of the release also changes to 'planned'. Furthermore, it is possible to lock the release assignment of requirements. To achieve this, the button 'Close Assignment' is provided.

6 Issue recognition and notifications

Conflicts and inconsistencies often occur during group decisions [Fel+17]. The reason is that stakeholders must agree on the release assignment of requirements. An example of such an inconsistency can be that two stakeholders have completely opposite preferences regarding the release assignment of a requirement [Fel+17]. Not only the assignment of requirements creates possible inconsistencies, but also evaluations. The whole group must agree on a given set of evaluations. Therefore, it is important to support the identification of a consensus between stakeholders. Approaches to support this are conflict *detection* and *diagnosis* techniques [FSZ12; Rei87]. These techniques can find conflicts between the preferences of stakeholders and provide possible solutions [Fel+17].

This chapter focuses on diagnosis and visualization aspects of conflicts in the context of release planning. In the first section, the issue panel, which provides an overview of all the *issues* in the project, is presented. The next section demonstrates the creation of issues. Furthermore, this section provides an overview of all automatically created issues on the basis of an example. In the IRP application, stakeholders are also able to create issues manually, which is demonstrated in Section 6.2.2. The notification feature, which visualises conflicts and informs all stakeholders about new chat messages, requirements or new information data, is presented in Section 6.3. A further diagnosis approach for the optimisation of releases is presented in Chapter 8.

6.1 Issue overview

Each project features its own issue overview, which is visible to all members of a project. All project-relevant issues are listed in this view. An example of an issue overview is provided in Figure 6.1. The issues are sorted by their creation dates and the newest issues are displayed at first. This sorting enables stakeholders to see the latest and most relevant issues at first glance.

Each issue consists of five parts:

- Name: A short title, which describes the topic of the issue

6 Issue recognition and notifications

OpenReq mmustermann EN | DE

Projects/Intelligent Release Planer

Requirements 2 Releases **Issues** General

Only my Issues Only open Issues

Show entries Search:

Name	Description	Assigned to	Responsible stakeholders	Created on
High dissent!	High dissent between stakeholder evaluations. It is strongly advised to discuss the evaluations with the stakeholders.	Req#1	mmustermann	Mar 19
High dissent!	High dissent between stakeholder evaluations. It is strongly advised to discuss the evaluations with the stakeholders.	Req#1	mmustermann	Mar 19
High dissent!	High dissent between stakeholder evaluations. It is strongly advised to discuss the evaluations with the stakeholders.	Req#0	mmustermann	Mar 19
High dissent!	High dissent between stakeholder evaluations. It is strongly advised to discuss the evaluations with the stakeholders.	Req#2	mmustermann	Mar 19

Showing 1 to 4 of 4 entries (filtered from 10 total entries) Previous 1 Next

Figure 6.1: Issues overview page in the IRP application.

- **Description:** A longer text, which provides a more thorough understanding for the issue
- **Assigned to:** An issue can be assigned to a requirement or a release. This field displays the ID field of the related component with the corresponding issue
- **Responsible stakeholders:** Displays all affected stakeholders of this issue
- **Created on:** Lists the month and the year when the issue was created

The stakeholders can navigate to the affected component by clicking on the name in the 'Assigned to' column.

An issue can be in the state 'open' or 'resolved'. Depending on this state, the representation of the issues changes. The message of the issue is displayed in a light grey if the issue is already resolved. If the text is still in black, this indicates that the stakeholders must still complete some tasks to resolve this issue. Open issues are also listed in other parts of the application (see Section 6.3).

To view only relevant and open issues, stakeholders are able to filter the list of issues. The stakeholders can filter the issue list to only see open issues by using the checkbox 'Only open issues'. Furthermore, it is possible to only display issues which were assigned to the current stakeholder. In addition, combination of both filters is possible.

6.2 Issue creation

There are two different ways to create issues in the IRP application. Issues are created either by the *system* or *manually* by the stakeholders. The modes are presented in the following sections.

6.2.1 Automatic issues created by the system

Automatic issues are created by the system because of specific inconsistencies in stakeholder preferences. Areas in which it is possible that issues occur are the assignment of requirements to releases and evaluation of requirements.

If too many requirements are assigned to a release, the sum of the capacity of the assigned requirements may exceed the maximum capacity of the corresponding release. This result means that the developers are not able to implement all requirements, or the managers must re-plan the resources for this release. Therefore, the system automatically creates an issue. By creating an issue, managers are informed and can react accordingly.

The other area which is monitored by the system is the evaluation of requirements. If a stakeholder has a preference which is too different from the other evaluations, an issue is created for all affected stakeholders. The creation of issues supports finding a consensus among stakeholders and improving the quality of the preferences. Each criterion of the evaluation section is monitored individually by the system. This monitoring means that stakeholders who are affected by an issue can concentrate on the discussion of the affected evaluation criterion. An example of a conflict with regard to a criterion is presented in Figure 6.2.

Table 6.1 lists examples of stakeholder preferences related to the criterion effort. Furthermore, the average value of the preferences is provided. As already discussed in Chapter 5.5.4, stakeholders can evaluate the effort criterion within a range from 1 to 50. A conflict regarding a preference of a stakeholder for a specific criterion of a requirement, R_e , is defined in Formula 6.1. The average of all preferences is

6 Issue recognition and notifications

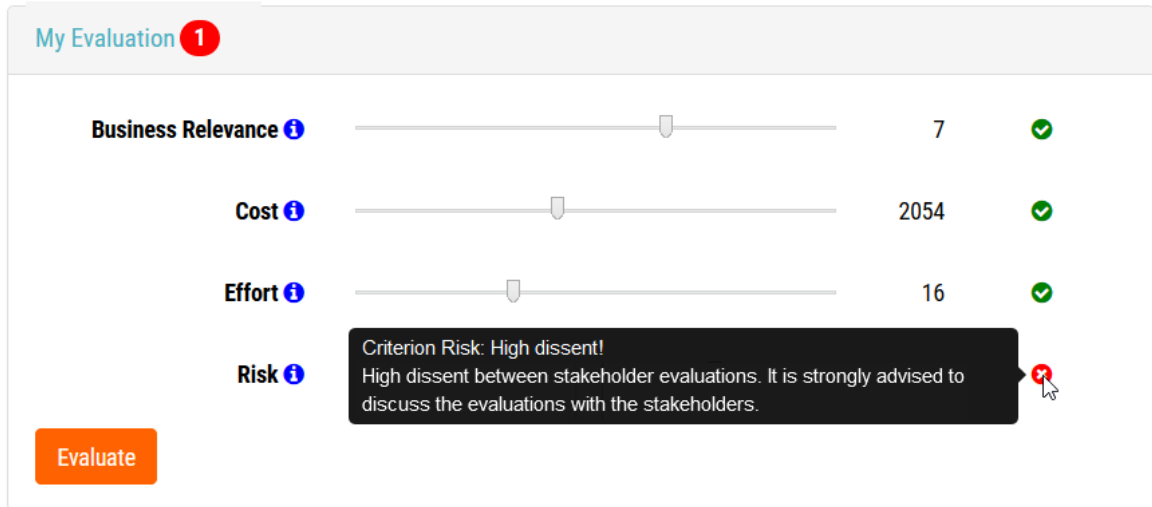


Figure 6.2: Visualization of the conflicts in the evaluation panel in the IRP application.

	S1	S2	S3	S4	S5	S6	AVG
R_1	40	30	35	40	45	20	35
Normalised	8	6	7	8	9	4	7

Table 6.1: Example of stakeholders' evaluations for the effort (max. value 50) criterion. The average value for the requirements were calculated and all preferences were normalised (base = 10).

compared to the individual preference, $pref$, of a stakeholder. The deviation of the preference is not allowed to be less than or greater than 2. If a deviation exceeds the limit, an issue is automatically created by the system. All criteria are normalised to a base of 10.

$$Conflict(R_e) = \begin{cases} true & pref > AVG + 2 \text{ or } pref < AVG - 2 \\ false & otherwise \end{cases} \quad (6.1)$$

According to Formula 6.1, the preference of S6 does not conform to the specification and produces an issue. After a new evaluation, the system checks each preference for a constraint violation. If there is a violation, the system creates an issue for the stakeholder.

6.2.2 Issues created manually by stakeholders

In addition to *automated issued generation*, user can create issues also *manually*. This feature is also intended to support the finding of a consensus between the stakeholders by highlighting conflicts. There can be various reasons for users to

manually create an issue.

If in the opinion of a stakeholder a requirement has been assigned to the wrong release, the stakeholder can create an issue and assign this to the responsible manager. Another example of a manually created issue could be that a stakeholder wants a further evaluation of a criterion. Then, the stakeholders who were assigned to the issues must discuss the criterion. Another reason for an issue could be missing information, such as lack of sufficient description or additional attachments.

To create a manual issue in the IRP application, a user must navigate to the issue overview page. The form for a manual issue is pictured in Figure 6.3.

Figure 6.3: The manual issues creation in the IRP application.

At the manual creation, the user enters a title and a description of the issue. Furthermore, either a release or a requirement must be associated with the issue. During the next step, the user selects all the stakeholders who will be notified that a new conflict exists. The 'Responsible stakeholders' field is a multi-select field, which means that more than one stakeholder can be selected, and supports the user with an auto-complete.

If a conflict has been resolved, it can be marked as 'resolved'. For this reason, a 'Resolve' button exists for every manually created issue in the issue overview. After the issue is closed, the issue is no longer displayed at the assigned release or requirement (see Section 6.3).

6.3 Notifications and visualization of conflicts

Notifications were introduced to keep all stakeholders of a project updated. These notifications help the stakeholders identify changes at the requirements and releases very quickly.

Notifications are created for following events:

- a new issue,
- new chat message,

6 Issue recognition and notifications

- the addition of a responsible stakeholder to a requirement, and
- new attachments for a requirement or release.

The system distinguishes between issue- and event-based notifications. For event-based notifications, such as a new chat message, an orange dot appears. The first time that a stakeholder is able to see the notification information is on the project overview page, which is illustrated in Figure 6.4. The orange dot is always present at the left side of the project name. If the user hovers over the dot, a tooltip is displayed with the count of new event-based notifications.

	Name	Releases finished	Date	
	Recommendation Project	2/3	Jan 17 - Mar 19	
	 Intelligent Release Planer 	0/1	Jan 18 - Apr 19	
	OpenReq	0/0	Jan 18 - Dec 19	

Figure 6.4: Visualization of notifications and issues on the project overview page.

Notifications guide users to the triggering origin. This means, that the notification is shown in each tab and section which is associated with the source requirement or release. An example of this guidance is displayed in Figure 6.5. In this example, the event notification leads the user to 'Req#2'.

The screenshot shows the OpenReq Intelligent Release Planner interface. At the top, there is a header with the OpenReq logo and a user profile icon for 'mmustermann' with language options 'EN | DE'. Below the header, the main title is 'Projects/Intelligent Release Planer'. There are four tabs: 'Requirements' (active, with a red badge '2'), 'Releases', 'Issues', and 'General'. Under the 'Requirements' tab, there is a sub-section 'New Requirements' with a red badge '2'. It includes a search bar and a table of requirements. The table has columns for 'Title' and 'Name'. The first row is 'Req#2' with the name 'User roles restrictions - Reader' and a red badge '1'. The second row is 'Req#0' with the name 'UI design of the login page' and a red badge '1'. The third row is 'Req#1' with the name 'User registration'. The fourth row is 'Req#3' with the name 'Usermanagement'. Below the table, there is a 'Create Requirement' button and a list of links for 'Planned Requirements', 'Completed Requirements', and 'Rejected Requirements'. At the bottom, there is a footer with the text '© OpenReq ReleasePlanning'.

Figure 6.5: Visualization of conflicts on the project page.

For issues, a red badge is displayed on the right side of a name, which is also presented in Figure 6.5. The badge contains a number indicating the number of conflicts for that area. Users can immediately see all the areas with a high number of conflicts.

Conflicts in the evaluation panel are also visualised. This visualization displays open issues for all criteria of a requirement. Each criterion has its own issue indicator. For this purpose, a red dot with a cross inside was introduced. The dot visualises a conflict for this specific criterion. However, if the dot is green, everything is acceptable, and the preferences of the stakeholder are consistent with the

6 Issue recognition and notifications

preferences of other stakeholders.

A difference between event-based and issue-based notifications is the duration of the visibility of the notifications. An event-based message disappears after the affected stakeholder has seen it, but an issue does not disappear until it is resolved. The reason is that event-based notifications only help users as informational messages. On the other hand, issue notifications warn the stakeholders about inconsistencies and mark these areas until the inconsistencies have been resolved.

7 Statistics

Statistics are mathematical equations and important in many application areas. Such areas can be medical, financial, or also production related. For example, in the medical field, medications are tested, and statistical evaluations are executed to assure that the medication is helpful and not hazardous.

Statistics are applied to collect, analyse, interpret, and visualise raw data. With the help of these methods, large datasets can be simplified, which helps users to better understand the analysed aspects.

By visualization of data, the users can recognize abnormal changes in the dataset at first glance. Keeping these facts in mind, the IRP application offers users three different statistics with a visualization of the datasets.

The statistics reach from the assignment of the requirements to the releases, up to the disagreement between the requirement evaluations. All these statistics are accessible via the project detail page.

In the following sections, all statistics are presented with a description and a visualization example.

7.1 Requirements assigned to releases

The 'Requirements assigned to Releases' statistic displays a brief overview of all planned releases. The statistic is visualised as a bar chart. The statistic displays which requirements are in the state 'new' or 'planned' as well as the number of assigned requirements for each release. This visualization helps managers to easily see the distribution of requirements over releases. Also, an overview for those releases, which have only a small number of requirements, is given. A small number of assigned requirements is mostly a sign of free capacities in a release. These capacities can be filled up with available requirements.

For the implementation of these statistics the JavaScript library d3js (see Chapter 3.2.13) is utilised. This library provides the possibility to visualise data with the help of HTML, CSS, and SVG.

7 Statistics

To create a chart with d3js, each line must be created by the code. The IRP application creates the axes and the bar elements separately. An example of the visualization is presented in Figure 7.1.

On the x-axis of the chart, the IDs of the analysed releases are displayed. The ID of the release is chosen as label, because it distinctly identifies the releases. Another reason for this is, that the name of the release could be too long for the labels. On the y-axis, the number of assigned requirements to a release is displayed. In Figure 7.1 it becomes apparent, that 'Rel#1' contains 8 requirements and 'Rel#2'

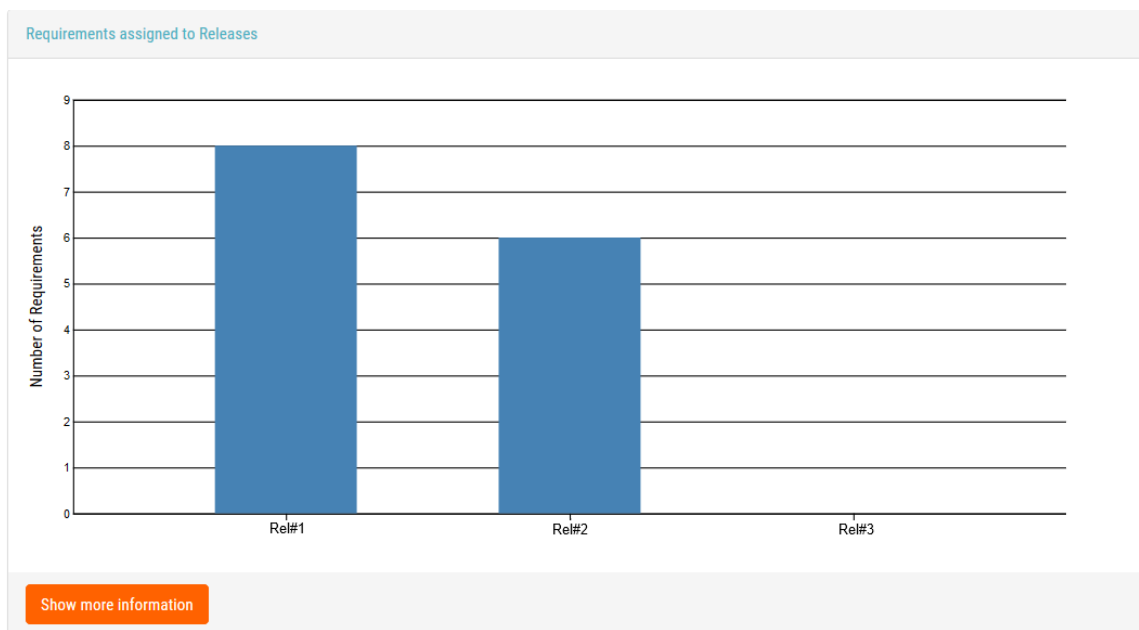


Figure 7.1: Example statistic: assigned requirements to a release.

contains six. Furthermore, 'Rel#3' has no requirements, which indicates that more requirements can be planned for this release.

Users can see a description of the statistic by clicking on the 'Show more information' button. After pressing this button, a text with a description for this statistic will be displayed.

7.2 Disagreement of requirements

The 'Disagreement of Requirements' statistic is important to gain a detailed overview of the rating behaviour of all project members. By analysing the rat-

ing data, managers gain insight into preferences of stakeholders.

For a manager it is important to stay informed about the level of disagreement between all project stakeholders [Ruh10]. If the disagreement between stakeholders is high, it is often a signal that there are problems and that further analyses are required. On the other hand, if the disagreement is low, it is an effective indicator that the consensus for this specific requirement is strong. Therefore, this statistic is an effective analysis option to detect problems and to revise the overall consensus of the stakeholders. To analyse the consensus of each criterion, managers can select the criteria separately.

As at the 'Requirements assigned to Releases' (see Section 7.1) statistic, the `d3js` (see Chapter 3.2.13) library is employed for visualization. An example of the visualization of this statistic for the 'priority' criterion is demonstrated in Figure 7.2.

The `d3js` library creates the minimum and maximum points of preferences of all stakeholders separately as well as the connection between the average preferences. On the x-axis of the chart, the IDs of the analysed requirements are displayed. This statistic analyses all requirements in the state 'new' and 'planned'. Furthermore, the statistic also includes all requirements without an evaluation. The inclusion of non-evaluated requirements has the benefit, that the managers can see all requirements at first glance, which have not been evaluated until now. On the y-axis, the evaluation scale of the selected criterion is displayed. The maximum value corresponds to the maximum evaluation value of the selected criterion.

For each requirement, three value points are created by the system. The highest point with the largest value represents the largest preference for this criterion and requirement. The point in the middle of the dataset represents the average of all preferences, and the lowest point displays the value with the least preference for this criterion. A strong consensus among stakeholders can be assumed, if the distance between all points is small.

According to this description, it becomes apparent that in Figure 7.2, requirement 'Req#2' has the largest disagreement in the context of the priority criterion. This is an indicator for managers to revise the requirement. Possible problems could be that the requirement is not well defined or that a discussion between the stakeholders is necessary. On the other hand, the evaluation of 'Req#3' seems to be well accepted by the stakeholders.

To support users in understanding this statistic, a description is provided. After pressing the 'Show more information' button a text with a description for this statistic is displayed.

7 Statistics

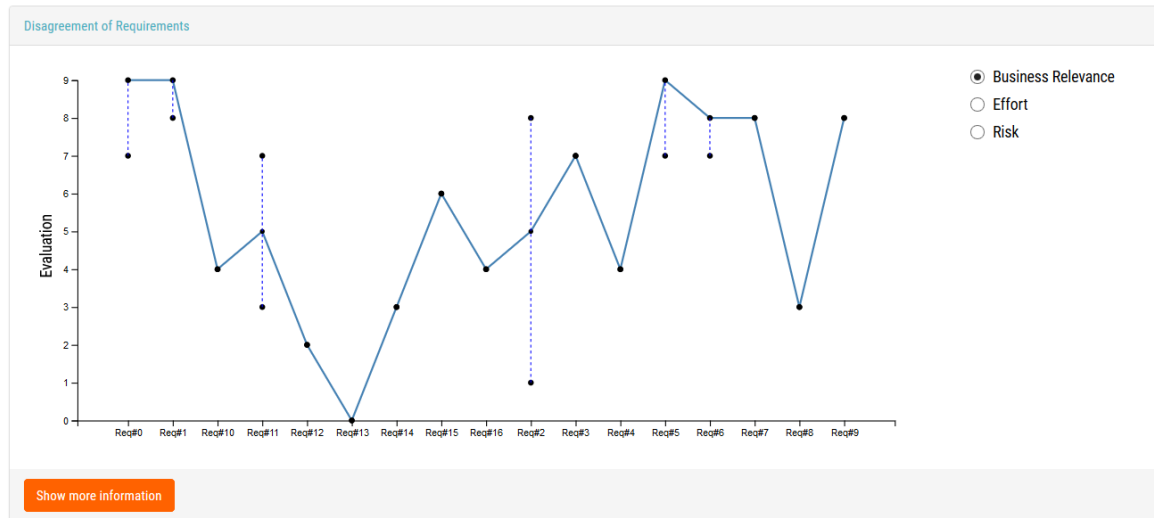


Figure 7.2: Statistic of requirements based on the disagreement between stakeholder preferences for a specific criterion.

7.3 Optimisation chart

The 'Optimisation' chart is important to obtain an overview of the planned releases. The chart compares the relevance values of releases, which were manually planned by managers.

This statistic provides a useful feedback for managers regarding the quality of the releases. If the relevance values of the optimal calculation are higher than the manual relevance values, this indicates, that improvements are possible. Furthermore, it could indicate, that there are still conflicts. The exact calculation of relevance values is exemplified in Chapter 8.

For the creation of this chart, the JavaScript library Chart.js¹ was employed. This library is easy to utilise and to integrate in HTML sites. The rendering of the chart is done in a canvas element. Chart.js creates the charts with the help of configurations. This has the advantage, that developers do not have to create the objects themselves and only must specify the configuration for the chart. At the moment, Chart.js supports eight different chart types (e.g. bar, line, scatter). The design of these charts can be efficiently changed and also legends for the diagram can easily be created with this library.

The 'Optimisation' chart is a chart with two lines, which is presented in Figure 7.3. The base builds the relevance value of the requirements, which is presented

¹<https://www.chartjs.org/> (Retrieved 01.12.2020)

7.3 Optimisation chart

in Formula 8.2 of Chapter 8. In Chapter 8, an example of the calculation of the relevance values is presented.

On the x-axis of the chart, the IDs of the analysed releases are displayed. Here the ID is chosen as label, because it clearly identifies the releases. This graphic displays all releases that are in the 'new' or 'planned' state. On the y-axis, the relevance values of the releases are displayed. The relevance values of the releases are composed of the sum of the relevance values of the added requirements. If the user hovers over a release, a legend appears for that release, which displays the optimal and manual relevance values for that specific release.

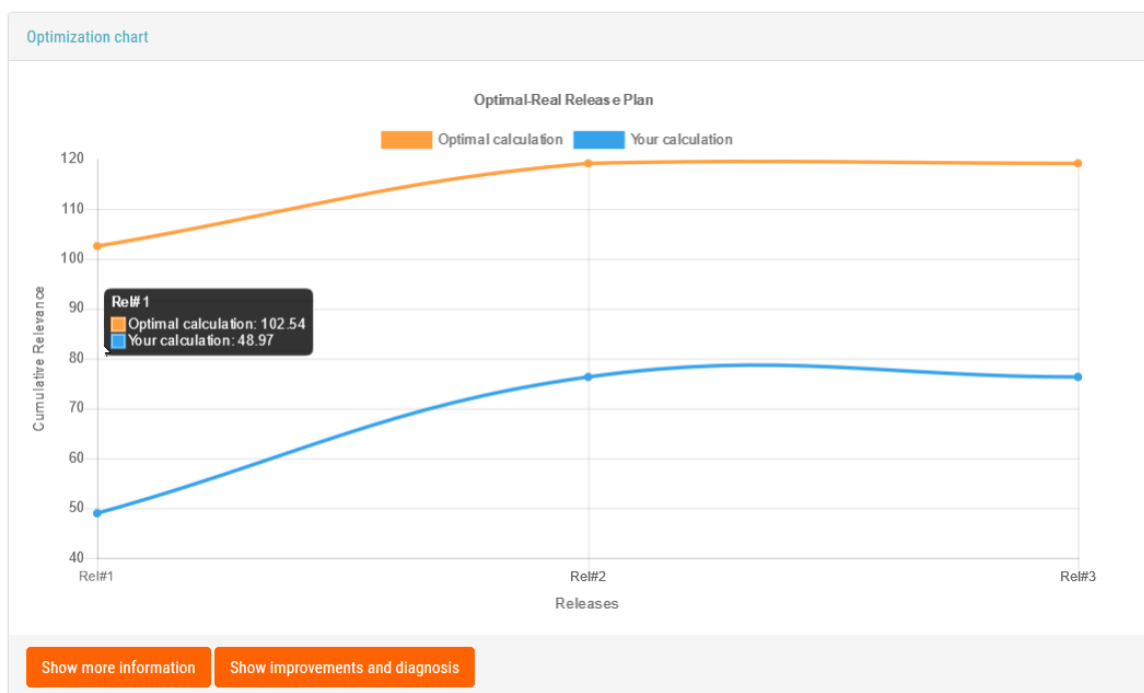


Figure 7.3: Chart for the optimisation of planned releases.

In Figure 7.3, it becomes apparent that there is a large gap between the optimal and manual relevance values for 'Rel#1'. For optimal calculation the legend displays a relevance value of 102.54 and for the current releases 48.97. This can be an indicator for managers, that important requirements are assigned to wrong releases. Furthermore, the chart demonstrates that the relevance value between 'Rel#2' and 'Rel#3' is stagnating. A stagnation is possible, when the analysed release contains no requirements. If the relevance values stagnate, the manager can plan or create further requirements for this release. It is also visible that the optimal calculation starts with a large value but increases not much higher. The

7 Statistics

reason for this are the assigned requirements. At the beginning, important requirements with a large relevance value are assigned to earlier releases and the later the releases are planned, the lower the relevance values of the requirements become.

An exact analysis and diagnosis of the planned releases is provided by the 'Show improvements and diagnosis' button. The managers also receive in this page suggestions for improvements of the planned releases. The optimisation and improvement page is presented in Chapter 8.

8 Release optimisation and diagnosis

Release plans often possess many constraints. These restrictions can be concerning available resources, deadlines, or the interdependencies between requirements. Because of those constraints, release plans require decisions regarding the release order of requirements [Fel+13]. To support managers in their decisions, all requirements are prioritised by stakeholders. By prioritising requirements, all unimportant items are separated from the important ones, but this is only one of the first steps. Managers must then consider the dependencies between requirements and the availability of resources.

Managers always try to achieve the maximum business value from the planned releases by considering all constraints. In addition, they attempt to satisfy all important stakeholders. Therefore, they assign the requirements with the highest priorities to early releases. The planning is challenging for managers due to a possibly large set of constraints [Fel+13]. For this reason, optimisation algorithms are needed. The algorithms can help to improve the assignment of requirements to releases. This improvement can be achieved by considering all the stakeholder' preferences for the requirements and their dependencies.

This chapter is organised as follows. In Section 8.1, the prioritisation of the requirements is presented. This section also presents the algorithm for the calculation of relevance values; this calculation is demonstrated by an example.

The next section reveals how all the dependencies of a requirement are detected and resolved by the system.

In the last section, the assignment of requirements to releases is presented.

8.1 Prioritisation

One way to systematically analyse the importance of requirements is prioritisation. A ranking can be derived from this prioritisation. This process is often not easy due to inconsistencies between the evaluations of stakeholders, but with the utilisation of consensus and recommendation techniques (see Chapter 5.5.5), the prioritisation process can be streamlined.

To achieve an optimal assignment of requirements to releases, all requirements are prioritised according to the stakeholders' evaluations. For the ranking of requirements, a *relevance value* is created. The creation of the relevance values is based on Multi Attribute Utility Theory (MAUT) (see Chapter 4.5). This approach enables the utility value calculation of an item, which is described by more than one criterion. Furthermore, available criteria and stakeholder evaluations can be differentiated with regard to their importance.

The IRP application differentiates between two stakeholder types. These types are 'project stakeholders' and stakeholders 'responsible' for specific requirements (see Chapter 5.4). Depending on the type of stakeholder, the weights, $w(v)$, for calculating the relevance value are different, which is shown in Formula 8.1. Formula 8.1 demonstrates, that stakeholders who are assigned to a requirement have a weight of two for this specific item and all others have a weight of one.

$$w(v) = \begin{cases} 2 & \text{assigned as a responsible stakeholder to the requirement } r \\ 1 & \text{otherwise} \end{cases} \quad (8.1)$$

During the evaluation process and when no additional criteria have been added, stakeholders state their preferences regarding a requirement in three different evaluation criteria (see Chapter 5.5.4). The criteria can have different weights (see Table 8.1).

Name	Weight
Business relevance	1.5
Effort (h)	2.0
Risk	1.0

Table 8.1: Evaluation criteria of a requirement with corresponding weights.

The criterion *Effort* has the highest relevance. This criterion has a weight of 2. The *Business relevance* follows with a weight of 1.5. The criterion *Risk* has the lowest importance. The reason for this distribution is that the IRP application focuses more on the operability than on possible failures of the releases.

At least one stakeholder must evaluate each requirement such that a relevance value can be derived. An example of an evaluation result is presented in Table 8.2. Table 8.2 displays the result of the evaluation process for the following example with five stakeholders and four requirements. The bold evaluations indicate that a user is a responsible stakeholder for a requirement and therefore has a higher weight. The criteria are defined with their abbreviation from left to right: Business relevance (BR), Effort (E) and Risk (R). Because of the high priority and low risk

Stakeholder	R1			R2			R3			R4		
	BR	E	R	BR	E	R	BR	E	R	BR	E	R
S1	5	35	8	4	20	6	8	30	2	8	45	3
S2	4	30	7	6	15	7	9	35	3	6	30	3
S3	2	40	8	6	10	8	9	25	5	7	45	3
S4	2	40	9	3	15	9	8	30	2	3	50	2
S5	3	45	9	3	35	8	9	35	2	6	40	4

Table 8.2: Example of a result of the evaluation process for requirements.

evaluations, it is already apparent that requirement R3 received the highest evaluations. After the evaluation process is completed, the system calculates the relevance value for the requirements.

The relevance value is determined using the utility approach MAUT (see Chapter 4.5) and determined through the consideration of positive and negative comments for the requirement. The calculation of the relevance value for a requirement is demonstrated in Formula 8.2.

$$\text{Relevance value}(r) = MAUT(r) + AB(r) \quad (8.2)$$

The positive and negative comments to a requirement are considered with an *argumentation-based* (AB) value. This value includes all agreements to a comment, which were given by other stakeholders. The consent of other stakeholders is represented as likes (see Chapter 5.5.6).

To determine the AB value, a base value for the comments is created by the system. The calculation is presented in Formula 8.3.

$$AB(\text{base value}) = \sum_{cp \in \text{comments}(pos)} \left[\left(\sum_{l \in \text{Likes}(cp)} l - \sum_{d \in \text{Dislikes}(cp)} d \right) + 1 \right] - \sum_{cn \in \text{comments}(neg)} \left[\left(\sum_{l \in \text{Likes}(cn)} l - \sum_{d \in \text{Dislikes}(cn)} d \right) + 1 \right] \quad (8.3)$$

In this formula, all the positive and negative comments with all their likes and dislikes are considered in the calculation. This formula checks if the sum of the positive comments, cp , with all their likes, l , and dislikes, d , is greater than the sum of all negative comments, cn . During the calculation of the base value, a rule is applied to Formula 8.3, which is presented in Formula 8.4.

$$value = \begin{cases} value & \text{if } \left(\sum_{l \in \text{Likes}(cp)} l - \sum_{d \in \text{Dislikes}(cp)} d \right) + 1 > 0 \\ 0 & \text{otherwise} \end{cases} \quad (8.4)$$

With this rule, negatively rated comments, which are stated as dislikes, have no influence on the overall AB value. After the creation of the base value, the AB value listed in Table 8.3 can be determined.

Base value of the comments	Value
≤ 2	0.2
$> 2 \ \& \ < 5$	0.5
> 5	1.0

Table 8.3: The AB value depending on the calculated base value of the comments.

Table 8.3 presents all the AB values for the comments. These values are added when the positive comments predominate the negative comments. However, negative comments also influence the relevance values. If the negative comments predominate the positives, the relevance value is decreased by the AB value.

By using Formula 8.2, the following relevance values in decreasing order can be derived for the requirements from Table 8.2:

$R_3 = 6.28$, $R_2 = 4.89$, $R_4 = 4.53$ and $R_1 = 2.83$.

As assumed, R_3 receives the highest relevance value. The following requirement is R_2 followed by R_4 . These two requirements are close to each other. In this case, a positive or negative comment could change the ranking of the requirements. The last position is occupied by R_1 .

8.2 Dependency resolution

During the modelling of requirements, stakeholders define *constraints* between individual requirements (see Chapter 5.5.3). These constraints are used to describe the relationship between the requirements [Fel+13; Tsa14]. Since dependencies define whether requirements are related or mutually exclusive, not all requirements can be implemented in the same release. Therefore, the system must resolve and consider the constraints during the optimisation process to be able to recommend the most effective solution [FB08; RS05].

An example of the definition of some dependencies between the requirements of this example is provided in Table 8.4. Table 8.4 indicates that requirement R_3 is associated with two requirements in the form of 'requires' dependency. Furthermore, R_2 has two 'requires' dependencies. R_4 has no dependencies and can therefore be excluded for the dependency resolution.

Requirement	Requires
R ₃	R ₁ , R ₂
R ₂	R ₁ , R ₃
R ₄	

Table 8.4: Examples of requirements and related constraints.

To be able to consider all dependencies, the system builds a dependency tree for the specific requirement. In this example, the system starts with the requirement R₃. All steps which are performed by the system and the final dependency tree are displayed in Figure 8.1. The system starts with the requirement R₃ and adds it as

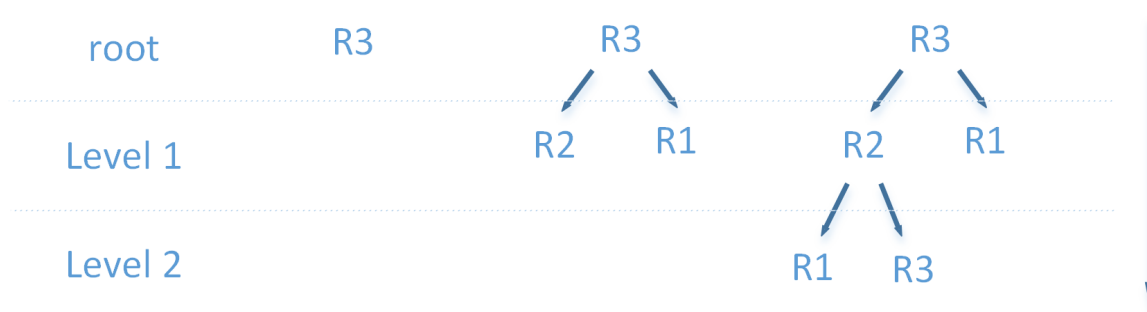


Figure 8.1: An example of the creation of a dependency tree.

the root of the tree. In this case, the system analyses all the 'requires' and 'excludes' dependencies of R₃. During the analyses, the system finds the requirements R₁ and R₂, which are connected via 'requires' dependencies. These two requirements are added as children of R₃ to the tree in Level 1. Next, the dependencies of R₁ and R₂ are analysed by the system. At this level, the system detects that R₂ has 'requires' dependencies with R₁ and R₃. These two requirements are then added to Level 2. Furthermore, the system checks if there are existing relationships. The system recognises that R₃ is already in the same path at the higher nodes and therefore, a strong relationship exists between R₂ and R₃. In the next step, the application analyses Level 2. The first requirement is R₁, which has no dependencies and is therefore not further analysed. However, R₃ has dependencies, but these dependencies have already been added. In this case, the dependencies of R₃ are no longer considered on the lower levels.

Next, the algorithm starts from the second level of the tree and checks if the predecessor of the current node and at least one child is already registered for the same release. If this is the case, the current node is also registered for the same release. The finished result for R₃ is that R₃ must be in the same release as R₂, and R₁ must be at least in an earlier or the same release as R₃.

This procedure is repeated for all created requirements until each requirement has

its dependency tree.

8.3 Assignment process

In order to achieve the greatest value for the business, an optimal assignment of requirements to the releases is important. During the assignment, various factors, such as the dependencies of the requirements and capacity of the releases, have to be considered. Therefore, a process is applied to achieve an assignment with all constraints considered, which is presented in Figure 8.2. The process consists of nine steps and various decisions. The steps are explained in this section.

8.3 Assignment process

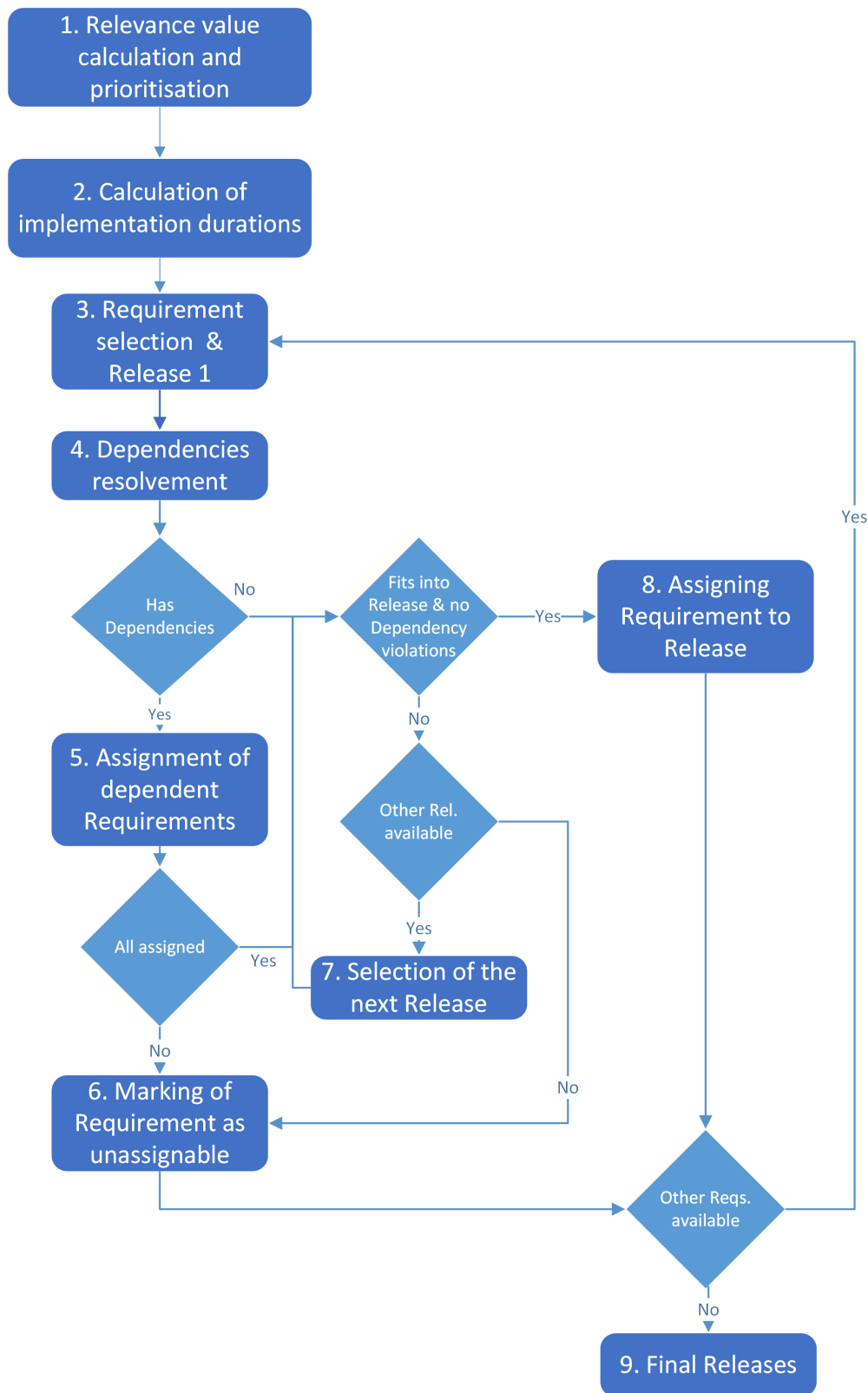


Figure 8.2: Process of the IRP application to assign available requirements to releases.

During the release planning it must be clarified, which requirement must be assigned to which release. For this purpose, the relevance values of the requirements (see Section 8.1) build the base for the assignment. Therefore, the **first task** of the process, the **Relevance value calculation and prioritisation** task, is to calculate the requirements relevance values and prioritise requirements according to the values. The aim of the approach is to have requirements with higher relevance values in the earlier releases rather than requirements with lower relevance values. Therefore, the approach sorts the requirements according to their relevance value and starts the assignment with the requirement with the highest relevance value.

Once the relevance values have been calculated, a further step is required before the requirements can be assigned to the releases. To be able to assign requirements to releases, managers need to know how long the implementation of a requirement takes, thus the capacity consumption of the requirements must also be calculated, which is performed in step **two**, **Calculation of implementation durations**. Otherwise, the sum of the capacities of the assigned requirements could exceed the available capacity of the release.

The IRP application uses a modified average algorithm for the calculation of the implementation duration, $h(r)$, of a requirement, which is shown in Formula 8.5. The calculation considers the weight of the stakeholders, $w(v)$, too.

$$h(r) = \frac{\sum_{v \in evals(r)} h(v) * w(v)}{\sum_{v \in evals(r)} w(v)} \quad (8.5)$$

By using Formula 8.5 the following implementation durations (h) were created for the requirements from Table 8.2: R1 = 38.13, R2 = 20, R3 = 30 and R4 = 43.13.

After the implementation durations and relevance values were created for the requirements, the requirements can then be assigned to releases. The assignment is performed iteratively, whereas the steps 3 to 8 are repeated until every requirement is assigned to a release or marked as unassignable. In the **third step**, **Requirement selection & Release 1** the approach takes the first requirement from the prioritised requirements and the release with the earliest release date.

Before a requirement can be assigned to a release, all dependencies must be resolved. For this reason, before a requirement can be assigned, all its 'coupling' or 'requires' requirements must be in the same or a previous release and its 'exclude' requirements must be in another release. Therefore, a dependency tree is created for the current requirement, which is presented in Section 8.2. The resolution of the dependencies is performed in step **four**, **Dependencies resolvment**, of the assignment process.

Depending on the current requirement, whether it has dependencies or not, different steps are performed next. In order for the current requirement to be assigned, all dependencies have to be resolved. The assignment of dependent requirements

is executed in step **five, Assignment of dependent Requirements**. In step five, the dependent requirements also pass through the steps three to eight. If not all dependent requirements were assigned after step five, the current requirement is marked as 'unassignable' in step **six, Marking of Requirement as unassignable**, and the process starts again with step three and the next requirement. If a requirement has no dependencies or all dependent requirements were assigned successfully, the system performs further checks. The system then verifies that no dependencies are violated and the capacity of the currently selected release is not exceeded.

When all preconditions are fulfilled, the current requirement is assigned to the release, which is executed in step **eight, Assigning Requirement to Release**. Otherwise, if the current requirement does not fit into the release and another release is available, the next one is tried for the assignment of this requirement.

The next release is selected in step **seven, Selection of the next Release**. The procedure is repeated until all requirements are assigned to a release or marked as unassignable. At the final step **nine, Final Releases**, all requirements are added to the available releases.

To demonstrate the process flow, two releases are available for this example, which both have a maximum capacity of 100. After the process has been applied to the requirements, the following release assignment appears, as demonstrated in Table 8.5. After the prioritisation and calculation of the capacity, the application with the

Release	Maximum Capacity	Planned Capacity	Requirements	Sum of the relevance values
Release1	100	88.13	R1, R2, R3	14.00
Release2	100	43.13	R4	4.53

Table 8.5: The result of the requirement assignment to releases.

assignment of the requirement starts with the highest relevance value, which is R3. However, R3 has a 'coupled' dependency with R2, which indicates that they must be in the same release; R3 also has a dependency with R1. For the developers to implement R3, R1 must also be completed. Since all three requirements fit into the same release, all three will be added to Release1. The last requirement which is evaluated by the system is R4. R4 has no dependencies, but the available capacity of Release1 is not sufficient for this requirement; therefore, the system adds this requirement to Release2.

The result is that Release1 has three requirements with a planned capacity of 88.13. If an additional requirement would be created that would not exceed the capacity of Release1, it could be added to this release. Release2 also has enough free capacity for further requirements.

8.4 Diagnosis

In the IRP application, managers can manually assign requirements to releases. The assignment of managers does not guarantee that there are no constraint violations or that there are no better assignment possibilities available. With the help of the application-created releases (see Section 8.3), the system can compare the assignments of managers. The IRP application allows managers to analyse their planned releases. They can check whether the assignment of the requirements to the releases is optimal or whether there are possible improvements.

The diagnosis page can be accessed via the button 'Show improvements and diagnosis' (see Chapter 7.3). An example of the page is displayed in Figure 8.3. The analysis is made for each planned release, and Figure 8.3 demonstrated that two releases were analysed. The figure indicates that the requirements have not been assigned perfectly. For each release, the maximum capacity is illustrated on the diagnosis page. Furthermore, the requirements relevance values, overall capacity consumption and optimisation suggestions are displayed.

Projects/Project Demo/Improvements

Release1

Maximum capacity (h) 100

Diagnose

One or more dependencies are violated. A higher relevance value is possible. There is more than 40% free capacity available.

Stakeholder assignment

Show entries Search:

ID	Description
Req#2	This requirement needs the following requirements in this or previous releases: [Req#1].
Req#3	This requirement needs the following requirements in this or previous releases: [Req#1].

Showing 1 to 2 of 2 entries Previous Next

Suggestions for an optimal release plan

Show entries Search:

ID
Req#1
Req#2
Req#3

Showing 1 to 3 of 3 entries Previous Next

Relevance value: 11.17

Capacity consumption (h) 50.0

Relevance value: 14.0

Capacity consumption (h) 88.13

Release2

Use improvements

Cancel

© OpenReq ReleasePlanning
Feedback

Figure 8.3: The diagnosis page of the IRP application for two releases.

The following criteria are checked by the system: capacity of a release, dependency

violations and assignment of requirements to a release. First, the system determines whether the maximum capacity of a release has been exceeded or if there is too much free capacity left. If one of those possibilities is the case, the application then displays the results in the 'Diagnose' box. An overview of all results is displayed in this box.

The dependencies are analysed in the next step. In case of a violation, a resolution suggestion is shown next to the requirement.

The last step is the analysis of the requirement assignments. The system compares the managers' requirement assignments with the systems' calculated assignment. If an improvement could be made, the system displays the suggestion next to the affected requirement.

Furthermore, managers can apply all suggestions to the requirements with the button 'Use improvements'. Then, all improvements are applied to the analysed requirements.

9 Evaluation

The impact of argumentation-based interfaces as presented in this master thesis (see Chapter 5.5.6) has been evaluated, for example, in Samer et al. [SSF20]. The major results of this study can be summarized as follows.

The aim of the study, presented in Samer et al. [SSF20], was to show how argumentation-based user interfaces can improve the quality of software requirement prioritisation. Therefore, an empirical evaluation was performed in Samer et al. [SSF20] to demonstrate the positive impact of argumentation-based interfaces. The evaluation revealed that argumentation-based user interfaces improve communication frequency between stakeholders. Due to higher communication frequency, a better refinement of requirements was achieved, which led to an increased stakeholder knowledge regarding the underlying software requirements.

Furthermore, argumentation-based interfaces increase the interaction rate on requirements, which means that stakeholders change (create, update, delete) their evaluations more often. The reason for this is increased information exchange and, as a consequence, increased stakeholder knowledge. This leads to more decision relevant information, such as qualitative arguments, for stakeholders. In this context, requirements are more frequently evaluated.

The evaluation showed that the success rate of projects can increase by argumentation-based interfaces. This is possible since argumentation-based interfaces foster a more detailed requirement analysis. Furthermore, by having all decision-relevant information available, requirements are better refined and prioritized.

Summarizing, argumentation-based interfaces can improve decision quality [SSF20]. This is achieved by fostering communication and information exchange between stakeholders. With this, stakeholders have more decision-relevant information available, which leads to a higher-quality prioritization.

10 Conclusion

This chapter presents the results of this master's thesis and possible future work and improvements.

10.1 Results

During this master's thesis, various goals were achieved.

A web-based application was created to involve the stakeholders at the early stages of the requirements' release selection. The users could create and manage releases. For this purpose, an argumentation-based MAUT ranking was introduced. The MAUT-based ranking enabled the stakeholders to evaluate for more evaluation criteria, which had the positive impact that the evaluation of the requirements could focus on different business values.

During the ranking process, the stakeholders were also supported in their decisions through recommendation techniques. Furthermore, a diagnosis of the requirements and dependencies was implemented to detect possible conflicts early in the evaluation process. A further improvement is the statistics for releases and the optimisation functions.

To support reaching a consensus among stakeholders, a chat was introduced. In this chat, requirement-relevant discussions could be held, and the advantages or disadvantages of a requirement could be stated. The stakeholders were also able to agree or disagree on arguments through likes or dislikes. These metadata were then used to calculate the optimal relevance value of a requirement during the prioritisation process.

10.2 Future work

An improvement for the application could be the introduction of a trophy or point system for the users. The user could receive trophies through effective prioritisa-

10 Conclusion

tions or by evaluating on many requirements. The trophies could provide a positive motivation for the stakeholders. This motivation could lead to a higher participation rate in the evaluations. Users could then also be suggested as responsible stakeholders for certain requirements based on previous assignments or trophies received.

A point for improvement is the reuse of requirements. The system could recommend already existing requirements through some metadata. The metadata could then be used to identify possible requirements which could fit into the current project [Fel+13]. The identified requirements would then be suggested to the manager. This approach could be based on CBF (see Chapter 5).

Another improvement could be the suggestion of more than one release plan. Each release plan could focus on different criteria or resolve the dependencies differently. By providing more plans, a manager could select the one which is most appropriate for the company, or stakeholders could evaluate the plans. This evaluation could also be done through more evaluation rounds. An approach was already presented by Ruhe in [Ruh10].

Another approach could be the clustering of the requirements into different categories, such as a category for 'views'. This clustering could enable the managers to assign stakeholders with specific experiences in this topic to the requirements as responsible stakeholders. Because of their expert knowledge, a more effective prioritisation could be achieved. On this topic, the system could also suggest possible responsible stakeholders for a requirement. The suggestion could be based on the metadata of the stakeholders and requirements.

The introduction of a staffing module could be another improvement. With the help of this module, available developer resources could be planned efficiently because not all the resources are needed at the same time and should only be involved at a specific point. Depending on the focus of the business value, different approaches are available [RRZ09; Kap+08].

Appendix

List of Figures

2.1	INTELLIREQ requirements detail view	9
3.1	Architecture of the IRP system	12
3.2	Bootstrap grid system	18
5.1	Overview of the IRP release planning flow with all steps and connections.	29
5.2	Login page of the IRP software.	31
5.3	Header of the IRP application with the profile settings menu.	32
5.4	Profile page of 'Max Mustermann', a user of the IRP application.	32
5.5	Project overview of the IRP application.	34
5.6	Detail page with the selected 'General' tab of a project in the IRP application.	35
5.7	Detail view of the 'Requirement properties scheme' section.	36
5.8	Detail view of the 'Attachments' section.	37
5.9	List of stakeholders, who were added to a project in the IRP application.	38
5.10	List of stakeholders, who can be added as responsible stakeholders to a project in the IRP application.	39
5.11	Overview of the 'Requirements' tab in the IRP application.	41
5.12	Detailed view of a requirement in the IRP application.	42
5.13	Detail view of an assignment of a dependency to a requirement in the IRP application.	44
5.14	Detail view of the evaluation section with all evaluation criteria of a requirement in the IRP application.	46
5.15	Recommendation of the effort criterion in the IRP application.	50
5.16	Chat regarding a requirement of the IRP application.	51
5.17	Release overview in the IRP application.	52
5.18	Release detail view in the IRP application.	54
5.19	The assignment of a requirement to a release in the IRP application.	55
6.1	Issues overview page in the IRP application.	58
6.2	Visualization of the conflicts in the evaluation panel in the IRP application.	60
6.3	The manual issues creation in the IRP application.	61

List of Figures

6.4	Visualization of notifications and issues on the project overview page.	62
6.5	Visualization of conflicts on the project page.	63
7.1	Example statistic: assigned requirements to a release.	66
7.2	Statistic of requirements based on the disagreement between stakeholder preferences for a specific criterion.	68
7.3	Chart for the optimisation of planned releases.	69
8.1	An example of the creation of a dependency tree.	75
8.2	Process of the IRP application to assign available requirements to releases.	77
8.3	The diagnosis page of the IRP application for two releases.	80

List of Tables

4.1	Recommendation of the priority, according to the majority heuristic.	22
4.2	Recommendation of the priority, according to the average heuristic.	23
4.3	Recommendation of the priority, according to the median heuristic.	23
4.4	Recommendation of the priority, according to the least distance heuristic.	24
4.5	Example evaluations for the MAUT algorithm. P = Priority, F = Feasibility	25
4.6	Example weights of evaluation criteria.	25
5.1	System evaluation criteria of a requirement with their descriptions and scales.	46
6.1	Example of stakeholders' evaluations for the effort (max. value 50) criterion. The average value for the requirements were calculated and all preferences were normalised (base = 10.	60
8.1	Evaluation criteria of a requirement with corresponding weights. . .	72
8.2	Example of a result of the evaluation process for requirements. . . .	73
8.3	The AB value depending on the calculated base value of the comments.	74
8.4	Examples of requirements and related constraints.	75
8.5	The result of the requirement assignment to releases.	79

Bibliography

- [o8] “ISO/IEC/IEEE International Standard - Systems and software engineering – Software life cycle processes.” In: *IEEE STD 12207-2008* (2008), pp. 1–138. DOI: 10.1109/IEEESTD.2008.4475826 (cit. on p. 3).
- [Akk+05] Marjan van den Akker et al. “Determination of the Next Release of a Software Product: an Approach using Integer Linear Programming.” In: *CAiSE Short Paper Proceedings*. Jan. 2005 (cit. on p. 3).
- [Akk+08] J.M. Akker et al. “Software product release planning through optimization and what-if analysis.” In: *Information & Software Technology* 50 (Jan. 2008), pp. 101–111. DOI: 10.1016/j.infsof.2007.10.017 (cit. on pp. 2, 3).
- [And98] Steve Andriole. “The politics of requirements management.” In: *IEEE Software* 15 (Dec. 1998), pp. 82–84. DOI: 10.1109/52.730850 (cit. on p. 2).
- [ARSo4] Amandeep, Günther Ruhe, and Mark Stanford. “Intelligent Support for Software Release Planning.” In: *Product Focused Software Process Improvement*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 248–262. DOI: 10.1007/978-3-540-24659-6_18 (cit. on pp. 6, 7, 27).
- [Ata+18] Muesluem Atas et al. “Liquid Democracy in Group-based Configuration.” In: Jan. 2018 (cit. on p. 26).
- [AW03] Aybüke Aurum and Claes Wohlin. “The fundamental nature of requirements engineering activities as a decision-making process.” In: *Information and Software Technology* 45 (2003), pp. 945–954. DOI: 10.1016/S0950-5849(03)00096-X (cit. on p. 39).
- [BB05] Barry Boehm and Victor R Basili. “Software defect reduction top 10 list.” In: *Foundations of Empirical Software Engineering: The Legacy of Victor R. Basili* 426.37 (2005) (cit. on p. 1).
- [BFG11] Robin Burke, Alexander Felfernig, and Mehmet H Göker. “Recommender Systems: An Overview.” In: *Ai Magazine* 32 (2011), pp. 13–18. DOI: 10.1609/aimag.v32i3.2361 (cit. on pp. 3, 47).
- [BG00] Kent Beck and Erich Gamma. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, 2000 (cit. on pp. 5, 6).

Bibliography

- [Bra90] John W Brackett. *Software Requirements*. Tech. rep. CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 1990 (cit. on p. 44).
- [BRW01] Anthony Bagnall, Victor Rayward-Smith, and I.M. Whitley. “The next release problem.” In: *Information and Software Technology* 43 (Dec. 2001), pp. 883–890. DOI: 10.1016/S0950-5849(01)00194-X (cit. on pp. 2, 3, 37).
- [Buro0] Robin Burke. “Knowledge-Based Recommender Systems.” In: *Encyclopedia of library and information systems* 69 (2000), pp. 175–186 (cit. on pp. 47, 48).
- [Buro2] Robin Burke. “Hybrid Recommender Systems: Survey and Experiments.” In: *User Modeling and User-Adapted Interaction* 12 (Nov. 2002), pp. 331–370. DOI: 10.1023/A:1021240730564 (cit. on pp. 3, 47).
- [Car+01] P. Carlshamre et al. “An industrial survey of requirements interdependencies in software product release planning.” In: *Proceedings Fifth IEEE International Symposium on Requirements Engineering*. Aug. 2001, pp. 84–91. DOI: 10.1109/ISRE.2001.948547 (cit. on p. 43).
- [CC12] Iván Cantador and Pablo Castells. “Group Recommender Systems: New Perspectives in the Social Web.” In: *Recommender Systems for the Social Web*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 139–157. ISBN: 978-3-642-25694-3. DOI: 10.1007/978-3-642-25694-3_7 (cit. on p. 2).
- [Chao5] Nicolas Charette. “Why Software Fails [Software Failure].” In: *IEEE Spectrum* 42 (2005), pp. 42–49. ISSN: 0018-9235. DOI: 10.1109/MSPEC.2005.1502528 (cit. on p. 1).
- [Dag+05] J. Dag et al. “A Linguistic-Engineering Approach to Large-Scale Requirements Management.” In: *Software, IEEE* 22 (Feb. 2005), pp. 32–39. DOI: 10.1109/MS.2005.1 (cit. on pp. 2, 37).
- [Davo3] Alan M Davis. “The art of requirements triage.” In: *Computer* 36 (2003), pp. 42–49. DOI: 10.1109/MC.2003.1185216 (cit. on p. 2).
- [Dua+09] Chuan Duan et al. “Towards automated requirements prioritization and triage.” In: *Requirements Engineering* 14.2 (2009), pp. 73–89. DOI: 10.1007/s00766-009-0079-7 (cit. on pp. 2, 44).
- [FB08] Alexander Felfernig and Robin Burke. “Constraint-based recommender systems: technologies and research issues.” In: *Proceedings of the 10th international conference on Electronic commerce*. ACM. 2008, p. 3 (cit. on pp. 47, 48, 74).

- [Fel+09] Alexander Felfernig et al. “Plausible Repairs for Inconsistent Requirements.” In: vol. 9. 2009, pp. 791–796 (cit. on p. 48).
- [Fel+10a] Alexander Felfernig et al. “Diagnosing Inconsistent Requirements Preferences in Distributed Software Projects.” In: *Software Engineering (Workshops)*. 2010, pp. 495–502 (cit. on p. 48).
- [Fel+10b] Alexander Felfernig et al. “Recommendation and Decision Technologies for Requirements Engineering.” In: *Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering*. Association for Computing Machinery, 2010, pp. 11–15. DOI: 10.1145/1808920.1808923 (cit. on p. 1).
- [Fel+11] Alexander Felfernig et al. “Group Decision Support for Requirements Negotiation.” In: *International Conference on User Modeling, Adaptation, and Personalization*. Springer. July 2011, pp. 105–116. ISBN: 978-3-642-28508-0. DOI: 10.1007/978-3-642-28509-7_11 (cit. on pp. 3, 8, 21, 22, 47, 49).
- [Fel+13] Alexander Felfernig et al. “An Overview of Recommender Systems in Requirements Engineering.” In: Apr. 2013, pp. 315–332. ISBN: 978-3-642-34418-3. DOI: 10.1007/978-3-642-34419-0_14 (cit. on pp. 1–3, 43, 45, 47, 48, 50, 71, 74, 86).
- [Fel+17] Alexander Felfernig et al. “OpenReq: Recommender Systems in Requirements Engineering.” In: *Proceedings of the Workshop Papers of i-Know 2017: co-located with International Conference on Knowledge Technologies and Data-Driven Business 2017 (i-Know 2017): Graz, Austria, October 11-12, 2017*. Oct. 2017, pp. 1–4 (cit. on pp. 1, 21, 27, 43, 57).
- [Fel+18a] Alexander Felfernig et al. “Configuring Release Plans.” In: *Proceedings of the 20th Configuration Workshop, Graz, Austria, September 27-28, 2018*. 2018, pp. 9–14 (cit. on p. 24).
- [Fel+18b] Alexander Felfernig et al. *Group Recommender Systems - An Introduction*. Springer, Cham, 2018. ISBN: 978-3-319-75066-8. DOI: 10.1007/978-3-319-75067-5 (cit. on pp. 2, 21).
- [FHC06] Brian Fitzgerald, Gerard Hartnett, and Kieran Conboy. “Customising agile methods to software practices at Intel Shannon.” In: *European Journal of Information Systems* 15 (2006), pp. 200–213. DOI: 10.1057/palgrave.ejis.3000605 (cit. on p. 5).
- [FM11] Ian Fette and Alexey Melnikov. *The WebSocket Protocol*. RFC 6455. RFC Editor, Dec. 2011, pp. 1–71. URL: <https://tools.ietf.org/pdf/rfc6455.pdf> (cit. on p. 16).

Bibliography

- [FN12] Alexander Felfernig and Gerald Ninaus. "Group Recommendation Algorithms for Requirements Prioritization." In: *2012 Third International Workshop on Recommendation Systems for Software Engineering (RSSE)*. 2012, pp. 59–62. DOI: 10.1109/RSSE.2012.6233412 (cit. on pp. 21, 22, 24, 49).
- [FSR13] Alexander Felfernig, Monika Schubert, and Stefan Reiterer. "Personalized Diagnosis for Over-Constrained Problems." In: *IJCAI 10.5591/978-1-57735-516-8/IJCAI11-454*. 2013, pp. 1990–1996 (cit. on p. 8).
- [FSZ12] Alexander Felfernig, Monika Schubert, and Christoph Zehentner. "An efficient diagnosis algorithm for inconsistent constraint sets." In: *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 26 (2012), pp. 53–62. DOI: 10.1017/S0890060411000011 (cit. on p. 57).
- [GR04] Des Greer and Guenther Ruhe. "Software Release Planning: An Evolutionary and Iterative Approach." In: *Information and Software Technology* 46 (2004), pp. 243–253. DOI: 10.1016/j.infsof.2003.07.002 (cit. on pp. 6, 51).
- [HK05] Reid Hastie and Tatsuya Kameda. "The Robust Beauty of Majority Rules in Group Decisions." In: *Psychological review* 112 (2005), pp. 494–508. DOI: 10.1037/0033-295X.112.2.494 (cit. on p. 21).
- [HL01] Hubert F. Hofmann and Franz Lehner. "Requirements engineering as a success factor in software projects." In: *IEEE Software* 18 (2001), pp. 58–66. ISSN: 0740-7459. DOI: 10.1109/MS.2001.936219. URL: <https://doi.org/10.1109/MS.2001.936219> (cit. on pp. 1, 2).
- [HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009. ISBN: 978-0-387-84858-7. DOI: 10.1007/978-0-387-84858-7 (cit. on p. 48).
- [Jam04] Anthony Jameson. "More than the Sum of Its Members: Challenges for Group Recommender Systems." In: *Proceedings of the Working Conference on Advanced Visual Interfaces*. Association for Computing Machinery, 2004, pp. 48–54. DOI: 10.1145/989863.989869 (cit. on p. 22).
- [JBK04] Anthony Jameson, Stephan Baldes, and Thomas Kleinbauer. "Two methods for enhancing mutual awareness in a group recommender system." In: *Proceedings of the working conference on Advanced visual interfaces*. ACM, 2004, pp. 447–449. DOI: 10.1145/989863.989948 (cit. on pp. 1–3, 47, 49).

- [Kap+08] Puneet Kapur et al. "Optimized staffing for product releases and its application at Chartwell Technology." In: *Journal of Software Maintenance and Evolution: Research and Practice* 20 (2008), pp. 365–386. DOI: 10.1002/smr.379 (cit. on p. 86).
- [Kar+04] Lena Karlsson et al. "Requirements prioritisation: an experiment on exhaustive pair-wise comparisons versus planning game partitioning." In: *Proceedings of the 8th International Conference on Empirical Assessment in Software Engineering (EASE 2004)*. Vol. 10. 2004, pp. 145–154. DOI: 10.1049/ic:20040407 (cit. on pp. 5, 6).
- [Kon+97] Joseph A Konstan et al. "GroupLens: Applying Collaborative Filtering to Usenet News." In: *Commun. ACM* 40 (1997), pp. 77–88. DOI: 10.1145/245108.245126 (cit. on pp. 47, 48).
- [KR93] Ralph L. Keeney and Howard Raiffa. *Decisions with Multiple Objectives: Preferences and Value Trade-Offs*. Cambridge University Press, 1993. DOI: 10.1017/CB09781139174084 (cit. on p. 24).
- [Lef97] Dean Leffingwell. "Calculating Your Return on Investment from More Effective Requirements Management." In: *Cutter IT Journal* 10 (Apr. 1997) (cit. on p. 2).
- [Lin+08] Markus Lindgren et al. "Key Aspects of Software Release Planning in Industry." In: Apr. 2008, pp. 320–329. ISBN: 978-0-7695-3100-7. DOI: 10.1109/ASWEC.2008.4483220 (cit. on p. 2).
- [LQF10] Soo Ling Lim, Daniele Quercia, and Anthony 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*. Association for Computing Machinery, 2010, pp. 295–304. DOI: 10.1145/1806799.1806844 (cit. on p. 2).
- [LV01] Soren Lauesen and Otto Vinter. "Preventing Requirement Defects: An Experiment in Process Improvement." In: *Requirements Engineering* 6 (2001), pp. 37–50. DOI: 10.1007/PL00010355 (cit. on p. 1).
- [Mas04a] Judith Masthoff. "Group Modeling: Selecting a Sequence of Television Items to Suit a Group of Viewers." In: vol. 14. 2004, pp. 93–141. DOI: 10.1023/B:USER.0000010138.79319.fd (cit. on p. 21).
- [Mas04b] Judith Masthoff. "Group Modeling: Selecting a Sequence of Television Items to Suit a Group of Viewers." In: *User Modeling and User-Adapted Interaction* 14 (Feb. 2004), pp. 37–85. DOI: 10.1023/B%3AUSER.0000010138.79319.fd (cit. on p. 26).

Bibliography

- [Mas11] Judith Masthoff. "Group Recommender Systems: Combining Individual Models." In: 2011, pp. 677–702. DOI: 10.1007/978-0-387-85820-3_21 (cit. on pp. 21–24, 26).
- [MT09] Walid Maalej and Anil Kumar Thurimella. "Towards a Research Agenda for Recommendation Systems in Requirements Engineering." In: 2009 *Second International Workshop on Managing Requirements Knowledge*. 2009, pp. 32–39. DOI: 10.1109/MARK.2009.12 (cit. on p. 1).
- [Nin+14] Gerald Ninaus et al. "INTELLIREQ: Intelligent Techniques for Software Requirements Engineering." In: *Proceedings of the Twenty-first European Conference on Artificial Intelligence*. ECAI'14. Prague, Czech Republic: IOS Press, 2014, pp. 1161–1166. ISBN: 978-1-61499-418-3. DOI: 10.3233/978-1-61499-419-0-1161 (cit. on pp. 3, 8, 9, 22, 47–49).
- [NR14] Maleknaz Nayebi and Guenther Ruhe. "An Open Innovation Approach in Support of Product Release Decisions." In: *Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering*. CHASE 2014. Hyderabad, India: Association for Computing Machinery, 2014, pp. 64–71. ISBN: 978-1-4503-2860-9. DOI: 10.1145/2593702.2593709 (cit. on p. 7).
- [PB97] Michael Pazzani and Daniel Billsus. "Learning and Revising User Profiles: The Identification of Interesting Web Sites." In: *Machine learning* 27 (1997), pp. 313–331. DOI: 10.1023/A:1007369909943 (cit. on pp. 47, 48).
- [PCo8] Pearl Pu and Li Chen. "User-Involved Preference Elicitation for Product Search and Recommender Systems." In: *AI Magazine* 29 (Dec. 2008). DOI: 10.1609/aimag.v29i4.2200 (cit. on p. 50).
- [PEM03] F. Paetsch, A. Eberlein, and F. Maurer. "Requirements engineering and agile software development." In: *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003*. 2003, pp. 308–313. DOI: 10.1109/ENABL.2003.1231428 (cit. on p. 5).
- [RB05] B. Regnell and Sjaak Brinkkemper. "Market-Driven Requirements Engineering for Software Products." In: *Engineering and Managing Software Requirements* (Jan. 2005). DOI: 10.1007/3-540-28244-0_13 (cit. on p. 2).
- [Rei87] Raymond Reiter. "A Theory of Diagnosis from First Principles." In: *Artificial Intelligence* 32 (1987), pp. 57–95. DOI: 10.1016/0004-3702(87)90062-2 (cit. on pp. 8, 57).

- [Ren+13] Dominik Renzel et al. "Requirements bazaar: Social requirements engineering for community-driven innovation." In: *2013 21st IEEE International Requirements Engineering Conference (RE)*. 2013, pp. 326–327. DOI: 10.1109/RE.2013.6636738 (cit. on p. 1).
- [REP03] Günther Ruhe, Armin Eberlein, and Dietmar Pfahl. "Trade-off Analysis for Requirements Selection." In: *International Journal of Software Engineering and Knowledge Engineering* 13.04 (2003), pp. 345–366. DOI: 10.1142/S0218194003001378 (cit. on pp. 2, 52).
- [RM05] Guenther Ruhe and J. Momoh. "Strategic Release Planning and Evaluation of Operational Feasibility." In: *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*. Feb. 2005, 313b–313b. DOI: 10.1109/HICSS.2005.561 (cit. on pp. 2, 3, 27, 37).
- [RN04] Guenther Ruhe and An Ngo-The. "Hybrid Intelligence in Software Release Planning." In: *Int. J. Hybrid Intell. Syst.* 1 (Sept. 2004), pp. 99–110. DOI: 10.3233/HIS-2004-11-212 (cit. on p. 3).
- [RRZ09] Md. Mainur Rahman, Guenther Ruhe, and Thomas Zimmermann. "Optimized assignment of developers for fixing bugs an initial evaluation for eclipse projects." In: *2009 3rd International Symposium on Empirical Software Engineering and Measurement*. 2009, pp. 439–442. DOI: 10.1109/ESEM.2009.5316025 (cit. on p. 86).
- [RS05] Guenther Ruhe and Moshood Saliu. "The Art and Science of Software Release Planning." In: *Software, IEEE* 22 (Dec. 2005), pp. 47–53. DOI: 10.1109/MS.2005.164 (cit. on pp. 2, 3, 27, 37, 45, 74).
- [Ruh10] Günther Ruhe. *Product Release Planning: Methods, Tools and Applications*. Auerbach Publications, June 2010. ISBN: 9780429126772. DOI: 10.1201/EBK0849326202 (cit. on pp. 2, 3, 6, 7, 27, 28, 37, 40, 43, 45, 47, 67, 86).
- [Sch86] Alexander Schrijver. *Theory of Linear and Integer Programming*. Wiley Interscience Series, 1986 (cit. on p. 6).
- [Sen+11] Christophe Senot et al. "Evaluation of Group Profiling Strategies." In: *IJCAI International Joint Conference on Artificial Intelligence*. Vol. 2011. 2011, pp. 2728–2733. DOI: 10.5591/978-1-57735-516-8/IJCAI11-454 (cit. on p. 21).
- [Som10] Ian Sommerville. *Software Engineering*. Addison-Wesley, 2010. ISBN: 978-0-13-703515-1 (cit. on p. 1).
- [SS97] Ian Sommerville and Pete Sawyer. *Requirements Engineering: A Good Practice Guide*. Jan. 1997 (cit. on p. 39).

Bibliography

- [SSF20] Ralph Samer, Martin Stettinger, and Alexander Felfernig. "Group Recommender User Interfaces for Improving Requirements Prioritization." In: *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization*. Association for Computing Machinery, July 2020, pp. 221–229. DOI: 10.1145/3340631.3394851 (cit. on p. 83).
- [Ste+13] Martin Stettinger et al. "WE-DECIDE: A Decision Support Environment for Groups of Users." In: *Recent Trends in Applied Artificial Intelligence*. 2013, pp. 382–391. DOI: 10.1007/978-3-642-38577-3_39 (cit. on p. 21).
- [Ste+15] Martin Stettinger et al. "Counteracting Serial Position Effects in the CHOICLA Group Decision Support Environment." In: *Proceedings of the 20th International Conference on Intelligent User Interfaces*. Association for Computing Machinery, 2015, pp. 148–157. ISBN: 9781450333061. DOI: 10.1145/2678025.2701391 (cit. on pp. 21, 24).
- [Sva+10] Mikael Svahnberg et al. "A systematic review on strategic release planning models." In: *Information & Software Technology* 52 (Mar. 2010), pp. 237–248. DOI: 10.1016/j.infsof.2009.11.006 (cit. on p. 2).
- [Tsa14] Edward Tsang. *Foundations of Constraint Satisfaction: The Classic Text*. BoD–Books on Demand, 2014. ISBN: 978-3-73-572366-6 (cit. on pp. 27, 43, 74).
- [VDL98] Axel Van Lamsweerde, Robert Darimont, and Emmanuel Letier. "Managing Conflicts in Goal-Driven Requirements Engineering." In: *IEEE Transactions on Software Engineering* 24.11 (1998), pp. 908–926. DOI: 10.1109/32.730542 (cit. on p. 43).
- [VMT+07] Cristina Vicente Chicote, Begoña Moros, Ambrosio Toval, et al. "REMM-Studio: an Integrated Model-Driven Environment for Requirements Specification, Validation and Formatting." In: *Journal of Object Technology* 6 (2007). DOI: 10.5381/jot.2007.6.9.a22 (cit. on p. 43).
- [WB13] Karl Wieggers and Joy Beatty. *Software Requirements*. Microsoft Press, 2013. ISBN: 978-0735679665 (cit. on p. 40).
- [Wie99] Karl Wieggers. "First Things First: Prioritizing Requirements." In: *Software Development* 7.9 (1999), pp. 48–53 (cit. on p. 44).
- [Wit+16] Ian H Witten et al. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2016 (cit. on p. 47).
- [WK04] Linda Wallace and Mark Keil. "Software project risks and their effect on outcomes." In: *Communications of the ACM* 47 (2004), pp. 68–73. DOI: 10.1145/975817.975819 (cit. on p. 1).