



**Hochschule
Bonn-Rhein-Sieg**
University of Applied Sciences

Fachbereich Informatik
Department of Computer Science

Masterproject

in the Master's program Computer Science

Modernization and extension of the YAWL-Dashboard

by

Robin Steinwarz

First examiner: Prof. Dr. Andreas Hense
Supervisor: M. Sc. Markus Stuhm

Submitted: Tuesday 31st January, 2023

Statutory Declaration

Robin Steinwarz
Büttinghausenstr. 15
53129 Bonn

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 marked all material which has been quoted either literally or by content from the used sources.

The work has not yet been submitted to any examination authority and has not yet been published.

.....
Location, Date	Signature

Contents

List of Figures	iv
List of Tables	v
1 Introduction	1
1.1 Problem statement	1
1.2 Objective	1
1.3 Approach	2
2 Modernization	4
3 Conception	7
3.1 Current state of the Management Cockpit	7
3.2 Requirements analysis	8
3.2.1 Approach	9
3.2.2 Execution	11
3.2.3 Results	12
3.3 Transformation plan	14
4 Implementation	24
4.1 General approach for the implementation of user stories	24
4.2 Backend	25
4.2.1 Transformation	25
4.2.2 Data retrieval	27
4.2.3 Information values for management	29
4.3 Frontend	32
4.3.1 Transformation	32
4.3.2 View	33
4.4 Realized user stories	34
5 Summary	35
6 Bibliography	36
7 Appendix	37
7.1 Protocols	37
7.2 Complete list of requirements	46
7.3 Complete list of prioritized requirements	50

List of Figures

1	Example for a modernized code section	6
2	General architecture of the Management Cockpit (Thomas, 2017, p. 35) . .	8
3	Login page mockup	15
4	General layout mockup	16
5	Specification mainpage mockup	17
6	Cases page mockup	18
7	Work items page mockup	19
8	All cases work items queue page mockup	20
9	Tasks page mockup	20
10	Statistical specification selection page mockup	21
11	Statistical comparison of specifications page mockup	22
12	Benchmark creation page mockup	22
13	Example specification retrieval from ResourceService	24
14	Example specification attribute storing into local database	25
15	Package difference of the backend	26
16	Example route in SpecificationController for retrieval of tasks	28
17	Example method in WorkItemManager to retrieve tasks from ResourceService	28
18	Relevant progress times of a task	30
19	Package difference of the frontend	32
20	Specifications view	33

List of Tables

1	Backend component dependencies	4
2	Frontend component dependencies	5
3	Guide for the first interview	10
4	Guide for the second interview	11
5	Table of Stakeholders, based on Rupp, 2014, p. 81	11
6	Some user stories	13
7	Prioritization of user stories according to the relative-weight method, based on Gloger, 2011, S. 138, 139	14
8	API extension	27
9	Calculation of information values for management	31
10	Realized user stories	34
11	Complete user stories list, part 1	46
12	Complete user stories list, part 2	47
13	Complete user stories list, part 3	48
14	Complete user stories list, part 4	49
15	Prioritization of all user stories according to the relative-weight method, based on Gloger, 2011, S. 138, 139	50

Abbreviations

API Application Programming Interface

BPM Business Process Management

DAO Data Access Object

HTTP Hypertext Transfer Protocol

JPA Jakarta Persistence API

JSON JavaScript Object Notation

REST Representational State Transfer

UI User Interface

URI Uniform Resource Identifier

US User story

WfMS Workflow Management System

YAWL Yet Another Workflow Language

1 Introduction

1.1 Problem statement

The analysis and optimization of business processes is of strategic importance for business. This is the reason for the intensive use of Workflow Management Systems (WfMS), which enable companies to automate a large part of their business processes. However, the use of WfMS has its own problems. These are, among other things, the high costs during their use (Hofstede et al., 2010, preface).

M. Adams et. al have addressed these issues in their book 'Modern business process automation'. They developed the open-source WfMS YAWL, with its own notation of the same name, and licensed it in 2004. Its range of services includes the management of business processes, their form-based web execution and the automation of process steps. Additionally, for integration into enterprise system landscapes, organizational resources and applications can be incorporated into process automation (The YAWL Foundation, 2022). Since 2004, the YAWL core has undergone significant development, resulting in a stable and reliable system, that could have a great added value in the future. Especially for small and medium enterprises, as well as for scientific use. During this time, however, an important feature for visualizing management information, a dashboard, was not integrated into YAWL. While several other commercial products have this feature by default, giving them a competitive advantage over the YAWL WfMS.

Philipp René Thomas took a first initiative in 2017 to solve this issue, developing a new user interface (UI) for the YAWL system and integrating dashboard functionality into it. He described this component as Management Cockpit, with the goal to supply operational management with computed process information (Thomas, 2017). Therefore, in the following, when using this term, this component is meant.

In the end, the Management Cockpit was never integrated into the YAWL system. As a direct result of that, no efforts have been made to maintain and further develop the Management Cockpit since then. Accordingly, the software module is still based on industry standard technologies, but also on libraries that are severely outdated. Almost all dependencies are no longer actively supported and some contain critical, security-related vulnerabilities. The dependencies of the backend component have for example more than 95 known vulnerabilities.

In summary, the YAWL system needs a dashboard component that allows managers to monitor process efficiency. The Management Cockpit could be a good basis for solving this problem. However, not in its current form.

1.2 Objective

The goal of this project is to increase YAWL's competitiveness and thus make it more attractive to small and medium enterprises. To achieve this goal, a user-friendly designed dashboard based on the technological industry standard will be developed. The prototypical Management Cockpit from Philipp René Thomas will serve as the basis for this YAWL dashboard. This approach has a high chance to save a large amount of effort, which has already been undertaken.

Other sub-goals exist in the context of the target dashboard. The central ambition to be

achieved by the dashboard is to satisfy the information needs of the different management levels. In addition, the desired information should be easily retrievable by management personal. The dashboard must be secure and economical. It should also be able to be delivered side-by-side as part of the YAWL system. In this way, the user-friendliness of the YAWL system is preserved.

Several limitations arise from the time constraints of the project and the complexity of the subject matter. Although the integratability of the YAWL dashboard is part of this project, it is not the actual integration. For that, the component would need to be tested and signed off by the YAWL development team beforehand. This is something that can happen after the time frame of this master project. In addition, due to various risks, it is possible that not all planned functionalities can actually be implemented in the limited time.

1.3 Approach

In order to achieve the objectives stated in the previous section, a procedure consisting of several work packages was identified. The procedure and its steps reflect the chapter structure of this work. The chapters are described below.

Chapter 1 covers the first work package, which is about the Management Cockpit modernization. This first step had several reasons. First, the reuse of the Management Cockpit was intended to reduce the risk of not being able to implement the complexity of the target dashboard in the time available. Second, the initial modernization was intended to close the current security gaps of the Management Cockpit, reduce subsequent maintenance efforts, and ensure its longevity. In this context, the frontend and backend components had to be migrated to their latest technology versions. Libraries that were no longer actively supported had to be replaced. Technical debt had to be identified and removed. Last but not least, it had to be ensured that the consequential errors of the modernization were found and resolved.

Chapter 2 deals with the conceptual development of the yawl dashboard and its planning. The aim was to find out what the actual information needs of the management are and how they can be served in the dashboard.

The conception consisted of several sub-steps. First, an as-is analysis was performed to clarify the functionality and architecture of the Management Cockpit. Then, the information needs were identified in a to-be requirements analysis. For this purpose, several stakeholder interviews were conducted, resulting in various prioritized, functional and qualitative requirements for the target dashboard. The prioritization of requirements, in addition to weighting their relevance, was a tool for the later stage of implementation. The development was limited to the requirements of the highest priority. This was to further reduce the risk of project failure. The follow-up step after the as-is and to-be analysis was the conceptual planning of the dashboard development. This describes the transition from the Management Cockpit to the yawl dashboard and includes mockups as well as the modified dashboard architecture. The mockups show the target dashboard layout and visualize the application of the requirements on the web interface.

Since this chapter's work basically did not overlap with the modernization of the Management Cockpit, both steps were carried out in parallel. Within the procedures of chapter 2,

close coordination was made with a key developer of the YAWL system, M. Adams. This was to increase the likelihood of later integration. Furthermore, these findings are a first intermediate result that can be reused in future work.

Chapter 3 describes and documents the implementation of the yawl dashboard on the basis of the previously described concept. This chapter mainly communicates the realizations that have been achieved. That is, the actually implemented interfaces and components, as well as the implemented requirements. In addition, implications for the YAWL dashboard integration will be mentioned.

Chapter 4 summarizes the results of this project. It evaluates whether the goals of this project were achieved and describes complications, as well as further follow-up work.

2 Modernization

The present chapter documents the modernization of the Management Cockpit. This was carried out in three major steps. In the first step, all unsupported and deprecated dependencies were identified and replaced. In the second step, the implemented code structures were migrated to their modern dependency versions. The last step involved the analysis and adaptation of the general code structure as part of a manual technical dept analysis. The result formed the basis for the YAWL dashboard. The corresponding steps are discussed in detail in the following.

To illustrate the need for modernization, all dependencies of the Management Cockpit components are listed in the following tables. The Management Cockpit consists of a frontend and a backend component, hence two tables have been created to show each of their dependencies along with associated information. This is an identifier beginning with a P or D followed by an enumeration. The P represents a dependency that is used in production, while D represents a dependency that is used for development purposes only. This identifier is supplemented by the name of the dependency and its version used in the Management Cockpit. In addition, the support column shows active support by the letter Y and no active support by the letter N. The vulnerabilities column shows the sum of known vulnerabilities as of November 1, 2022. Finally, the new version shows the latest versions of the associated dependencies. The latest versions were retrieved on November 27, 2022.

The distinction between development and production dependencies was made, because of their different impact on security. Dependencies for development purposes have a lower priority in the context of security. This is because their code is usually not included in the production code. Therefore, their potential weaknesses cannot be exploited by attackers. Nonetheless, they should generally be updated to improve development functionality and avoid serious problems if development code accidentally makes it into production. Thus, they will be updated in this migration as well, if necessary.

ID	Name	Version	Support	Vulnerabilities	New Version
P01	spring-boot-starter	1.3.2	N	87	2.7.5
P02	spring-security-oauth2	2.0.8	N	26	-
P03	h2database	1.4.192	N	3	2.1.212
P04	jackson-core	2.6.5	N	0	2.13.4
P05	jackson-databind	2.6.5	N	49	2.13.4.1
P06	jackson-datatype-jsr310	2.6.5	N	49	2.13.4
P07	jdom	2.0.2	N	7	2.0.6.1
P08	commons-lang3	3.4	N	2	3.12.0
P09	commons-codec	1.10	N	1	1.13
P10	xerces	2.11.0	N	4	2.12.2

Table 1: Backend component dependencies

Table 1 shows the dependencies for the backend component. The vulnerabilities for the backend component were obtained from the MvnRepository, 2022 website. Maven dependency packages have their own information page on this website. The information includes details about direct and indirect vulnerabilities obtained from the U.S. govern-

ment's national vulnerability database (Center, 2022). For the table, indirect and direct vulnerabilities have been added together.

A total of 10 production dependencies were found. All spring boot starter subcomponents with the same version were grouped together. Together, all dependencies contain 228 vulnerabilities. All dependencies were updated to their highest available version.

Table 2 shows the dependencies of the frontend component. The vulnerabilities for the frontend component were determined using the npm package manager. NPM has its own database for security vulnerabilities, which can be determined for packages using the npm audit command (npm inc, 2022).

ID	Name	Version	Support	Vulnerabilities	New Version
P01	angular	4.4.4	N	9	14.2.6
P02	angular/cdk	2.0.0	N	3	14.2.6
P03	angular/material	2.0.0	N	4	14.2.6
P04	covalent/core	1.0.0	N	9	3.2.4
P05	angular2-notifications	0.7.8	N	5	-
P06	core-js	2.4.1	N	0	-
P07	font-awesome	4.7.0	N	0	-
P08	ngx-contextmenu	1.3.4	N	3	6.0.0
P09	rxjs	5.4.3	N	0	7.5.7
P10	zone.js	0.8.4	N	0	0.11.8
D01	angular/cli	1.4.4	N	98	14.2.7
D02	angular/compiler-cli	4.4.4	N	3	14.2.8
D03	angular/language-service	4.0.0	N	0	-
D04	types/node	8.0.32	N	0	-
D05	codelyzer	3.2.0	N	3	-
D06	ts-node	3.3.0	N	0	-
D07	tslint	5.7.0	N	0	-
D08	typescript	2.5.3	N	0	4.8.4

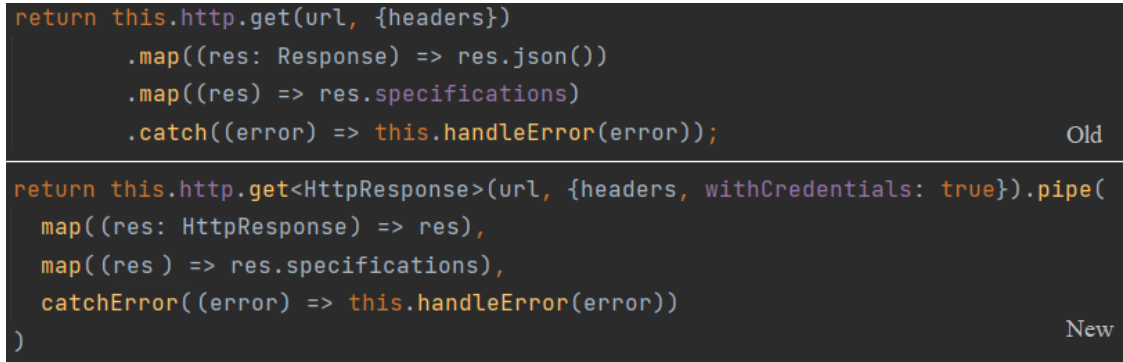
Table 2: Frontend component dependencies

A total of 10 production dependencies and 8 development dependencies were found. All angular subcomponents with the same version were summed. The production dependencies contained 33 vulnerabilities, whereas the development dependencies had 103 vulnerabilities.

All dependencies were updated to their highest possible versions. However, this was not practical or possible for all dependencies. The angular2-notifications dependency for website notifications was not upgraded for the latest version of angular. Accordingly, there is no version compatible with the updated plugins. Thus, it has been replaced by the angular-notifier dependency. The dependencies font-awesome, core-js, angular/language-service, types/node, ts-node, codelyzer, tslint were no longer needed. No new version is stated for these dependencies in the table.

Once the dependencies were updated, it was the next task to migrate the existing code back to a runnable state. To achieve this, the code was changed step by step to the current versions of the dependencies. Migration documentation from the according technologies was used to support this process. For example, Figure 1 shows the migration of a section of code in the frontend component. The upper part shows the old section, the lower part

the new section. Here it is noticeable that the structures are very similar overall, but to adapt to the new versions these slight changes had to be made.



```
return this.http.get(url, {headers})
    .map((res: Response) => res.json())
    .map((res) => res.specifications)
    .catch((error) => this.handleError(error));
```

Old

```
return this.http.get<HttpResponse>(url, {headers, withCredentials: true}).pipe(
    map((res: HttpResponse) => res),
    map((res) => res.specifications),
    catchError((error) => this.handleError(error))
)
```

New

Figure 1: Example for a modernized code section

To be able to find all remaining issues, the code was compiled and executed. This way, most existing problems were found and fixed one by one. After both the frontend and the backend were ready to run again, the functions were tested one after the other. Here, only the layout and design errors of the migration that did not lead to compilation or runtime errors had to be eliminated. In the final step, the code structures were analyzed manually on a conceptual level. No gross technical debt was found thereby. The structures of the Management Cockpit were in good condition. Nevertheless, another technological issue has been identified in this procedure. It was the use of the technology gradle for the build process of the backend component of the Management Cockpit. This is an inconsistency with the approach of the YAWL project. Maven is used there. According to Lilienthal, 2019, p. 125, technologies should be used as uniformly as possible in a project, unless great added value is achieved. This is not the case here. Therefore, the backend component was reconfigured to use maven instead.

After these steps, the Management Cockpit was ready for use again.

3 Conception

The following sections document the procedure and results during the conception phase. First, the current contents of the Management Cockpit are roughly described. This is followed by a description of the procedure for requirements analysis in order to determine what information is to be presented in the dashboard. The last section then shows how these new requirements can be integrated into the Management Cockpit.

3.1 Current state of the Management Cockpit

This section briefly describes the current state of the Management Cockpit. For this purpose, the functionality of the Management Cockpit and the system structure created for it are presented.

The Management Cockpit provides the user with two main functional areas. The first functional area allows the user to display basic specification information, current work items and organizational data. In part, the Management Cockpit also offers administrative activities here. In addition, a second functional area displays notifications for exceptional situations to the user. For example, if no work items are assigned to ongoing cases or if resources are overloaded. Beyond that, the Management Cockpit offers synchronized authentication via the YAWL resource service.

The current status of Philipp Rene Thomas Management Cockpit can be found on Github (Thomas, Philipp René, 2022). The status there is superior to the final version of the master thesis and includes various bug fixes. Furthermore, OAuth was implemented there for testing purposes. This is an alternative authorization method based on tokens. However, this authorization variant is not compatible with the current YAWL system. Due to the most recent status, the Github code is still used as a basis.

Figure 2 shows the general architecture of the Management Cockpit. Basically, the YAWL system is used merely as a data source. Here, data flows from the ResourceService of the YAWL system to the backend component of the Management Cockpit and finally to its frontend component. For persistence of computed data, the backend component still integrates its own additional database.

According to the Management Cockpit documentation, the backend component performs the following tasks. Providing the static and dynamic program data for the frontend. The processing of customization and formatting requests. The execution of monitoring processes, as well as authentication and authorization (Thomas, 2017, p. 36).

For the accomplishment of these tasks, the backend component implements some functionalities. This is the implementation of an interface, through which a communication with the ResourceService of YAWL becomes possible. Furthermore, a second Representational State Transfer (REST) interface, through which a client can use the capabilities of the dashboard via the frontend. Last but not least, the backend manages the communication with an integrated database in which designated dashboard information, also processed by the backend, is stored.

The frontend component, on the other hand, visualizes forms for authentication. It enables the management of dashlets, notifications and observances and visualizes them (Thomas,

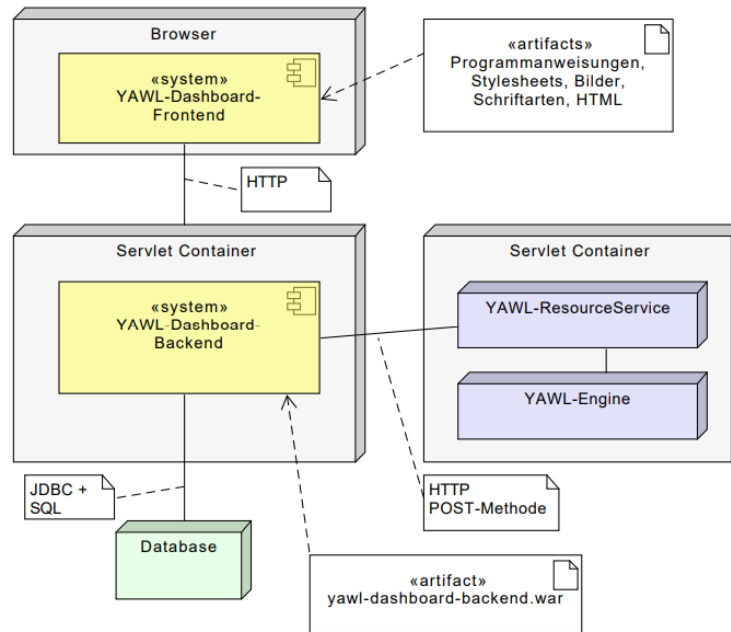


Figure 2: General architecture of the Management Cockpit (Thomas, 2017, p. 35)

2017, p. 37).

In order to perform these tasks, the frontend implements its own interface, in which services request data from the REST interface of the backend. However, it must also take into account the entity model of the ResourceService. User interaction is made possible via various Angular components that are nested within each other.

Both the frontend and the backend components have so far been considered a black box. This is because in this chapter it is mainly relevant what the Management Cockpit does and not exactly how it implements it. In the next subsections, this functionality will be revisited. A more detailed look into the individual components is given in chapter 4. There it is explained more closely, how the subcomponents of the Management Cockpit were changed to suit the goals of the YAWL dashboard.

3.2 Requirements analysis

This section describes how the requirements analysis was carried out. It first describes the organizational planning, i.e., what exactly had to be undertaken as part of this requirements analysis. This is followed by a section underpinning the approach to requirements analysis based on current literature. Finally, the execution and the collected results are documented.

Stakeholders were the direct sources of information for this requirements analysis, and discussions had to be held with them accordingly. During project implementation, two stakeholders in particular offered themselves for these discussions. At this point it must be said that normally only two stakeholders are probably not sufficient to get an accurate picture of the requirements. However, in the context of this work, this was sufficient.

Two individual interviews with one stakeholder each were planned for the obtainment of information. The first interview was intended for requirements discovery and documentation, while the second interview was intended to fine-grade and prioritize the prior results. Both interviews were conducted according to the interview techniques described in Rupp, 2014 on pages 107-109. The following subsections describe the application of the techniques and the resulting prioritized requirements.

3.2.1 Approach

Each interview consisted of several phases. The first phase of preparation involved inviting participants. This invitation included information about the context and how the interview would be conducted. The second phase was conducting the interviews. This is described in the following text sections, as this phase was different for the first and second interviews. Due to the ongoing Corona crisis and the geographical distance of the participants, the interviews were conducted digitally. All interviews lasted approximately one hour. For convenience and with the prior consent of the participants, all interviews were recorded. In the final analysis phase, the recording was converted into a comprehensible protocol. For this purpose, the verbal formulations were revised. This also had the purpose of finding the information that was mentioned between the lines. The protocols were then sent to the interviewees for approval in digital form. The interview protocols can be found in the appendix.

As mentioned in the previous text sections, there were two types of interviews. The first interview with the aim of requirements discovery was planned and structured. It consisted of a structured part in which several prepared questions were asked. They were neutral and open-ended, allowing participants to answer freely and creatively. A second reason for these open-ended questions was to allow participants to focus on the content that was most important to them.

The interview guide of the first interview is shown in the table 3. In each case, the interview began with a welcome, re-explanation of the topic, and the interview guide. This gave the participant a chance to mentally prepare for the questions and recall their own interests related to the dashboard. This was followed by around 16 minutes of asking the interview questions. At this point, a further distinction was made between two parts of questions. The first question part up to question eight aimed to bring to light conscious requirements and the second subconscious requirements. This was done in accordance with Rupp, 2014, p. 94-97. Meanwhile, the participants answers were also addressed, so that new questions developed from the course of the conversation. The dialog part was followed by an open discussion lasting five minutes, which was intended to give the participants the chance to bring out their own aspects.

Phase	Duration in minutes	Content
Start	2	Greeting and warm-up
Lead in	5	Interview guide and re-explanation of the subject matter and goals of the interview
Dialog	40	Q01: Please put yourself in the position of a tactical manager. Explain the situation of the tactical manager. Which informations does he need? Q02: How can the dashboard satisfy the need for information? Q03: Please put yourself in the position of a strategic manager for the last time. Can you also describe his perspective as opposed to the other roles? Q04: That's it for the roles. Please now answer the next questions intuitively. What interfaces do you think need to exist on the web interface? Q05: Which visualizations have to be present on the web interface? Q06: How do you imagine interactions with the dashboard? Q07: You've already mentioned some interesting ideas. Now there are certainly a lot of, let's say, more boring features that the dashboard typically comes with by default. In this last section of questions I would like to focus more on them. Therefore, please think of several core functions such as a simple login. Which functions come to mind first? Q08: Please describe a simple scenario for using the dashboard. What steps need to be taken in detail? Q09: Is there another not-so-common scenario that comes to mind?
Discussion	20	Open discussion
Closing	2	Acknowledgement and farewell

Table 3: Guide for the first interview

The second interview was used for reconciliation in order to avoid misunderstandings and possible ambiguities in the requirements set. The interview was also structured in a same way displayed in table 4. However, the dialog part of the interview was not questions, but the actual requirements identified in the first interview. The plan was to discuss all identified requirements one after the other. For any requirement, the plan was to cover multiple questions. Is the underlying problem correctly formulated by the requirement description? Are there other related requirements that have not yet been addressed? Is it possible to refine the requirement description?

The refinement process was followed by a round of prioritization. In this context, Gloger's relative weight method was used. However, participants were not asked to estimate values for relative advantage, penalty, cost, and risk (Gloger, 2011, pp. 138, 139), as indicated in the method. This discussion would have taken several hours. A time frame beyond the time resources of our participants for this project. Instead, prioritization values were estimated prior to the interview, without the participants. This had the advantage that requirements with a very high or very low priority could already be excluded from the

prioritization process. This meant that the time-limited meeting could concentrate on the requirements with a medium priority. The realization of these requirements was uncertain. Accordingly, three priority lists were prepared in advance. The first list included all requirements with a priority above two, the second list all requirements with a priority between one and two, and a third list with requirements below priority one. The plan was to present the second list to the participants and let them select the most important requirements themselves. Based on this, the prioritization would be adjusted again, resulting in the final list of prioritized requirements.

Phase	Duration in minutes	Content
Start	2	Greeting and warm-up
Lead in	3	Interview guide and re-explanation of the subject matter and goals of the interview
Evaluation	30	Reconciliation and fine-grading of Requirements
Prioritization	10	Reconciliation of priority values
Discussion	5	Open discussion
Closing	2	Acknowledgement and farewell

Table 4: Guide for the second interview

3.2.2 Execution

Within the context of the project, various stakeholders were considered for conducting the interviews. In particular, developers and scientific, as well as commercial users of the YAWL system. In addition, only requirements of technically competent sources were to be determined (Rupp, 2014, p. 90). Due to their proximity to the YAWL system and several years of experience in the field of business process management (BPM), Mr. Michael Adams and Mr. Andreas Hense were ideally suited for the requirements analysis. They took the roles of developer and scientific user. Unfortunately, a commercial user was not involved. The stakeholders are described in more detail in the table 5. The table lists various information about the stakeholders. Their names, roles and other information related to the dashboard. What relevant knowledge they bring to the requirements analysis, as well as their own goals and relevance as stakeholders.

Name	Role	Knowledge	Goals	Relevance
A. Hense	Scientific User	Detailed Knowledge about BPM and manager needs	YAWL-System benefits, YAWL functionality that supports the BPM lecture	Requirements
M. Adams	Developer	Detailed Knowledge of the YAWL-System and BPM	YAWL-System benefits, simple integrability of the YAWL-Dashboard	Requirements

Table 5: Table of Stakeholders, based on Rupp, 2014, p. 81

The first round of interviews was originally planned as a digital individual interview to avoid mutual influences. However, due to organizational complications, a digital group interview was conducted instead. To avoid mutual influence, the dialog focused on one participant, while new ideas and additions from the second participant were considered in the open discussion. Each participant contributed several new requirements.

The first part of the dialogue was particularly productive. The participant used the initial questions for his own and contributed many ideas, even anticipating answers to later questions. The questions after Q03 were more narrowly focused and resulted in better clarity. Asking separately for requirements related to unconscious knowledge was helpful, as their questions triggered a whole new set of ideas from the participants. The last section of the open discussion was particularly effective and lasted longer than planned, as the two experts used this time to discuss interesting new ideas for the dashboard. Several requirements were expressed in this section.

The second interview was also not conducted as a digital individual interview, but again as a group interview. This did not pose a problem in the second part of the requirements refinement. However, it also did not go completely according to plan, as the participants were unexpectedly very prepared. They had been sent the requirements list in advance, but it was not assumed that they had already read them before the interview. Since the participants already knew the requirements, it did not make sense to go through them again piece by piece. Instead, it was asked at the beginning if the participants had noticed any ambiguities. This was not the case. Instead of going through all the requirements, only the requirements that had already been classified as ambiguous and unclear were discussed. In this context, several user stories were further developed. In addition, three new user stories were developed. These are the user stories US05, US14 and US16.

Prioritization took place according to plan. Participants were shown the list of requirements for which implementation was not certain. After 5 minutes, both participants were able to mark several requirements as particularly important to them. These were requirements US03, US04, US08, US10, US15, US17, and US19. Based on this, the priorities were adjusted again so that those important requirements had higher priorities than the unimportant ones. To do this, the relative penalty of the important ones was increased and that of the unimportant requirements was decreased. The newly identified requirements did not change anything for the prioritization, as they all have a priority below 1.

3.2.3 Results

Requirements were extracted from the interview transcripts. There were 33 user stories in total. The appendix contains the complete list, with the table 6 showing some of the user stories. A complete view of all user stories that are subject of the implementation has been omitted here for visual reasons.

Several pieces of information are displayed in the user story table. Each User Story has been given its own context in the form of an identifier and an epic. The epic is the overarching context that connects multiple user stories on similar topics (Bergsmann, 2014, p. 61). The epics Specification, Case, Work item, Historical, Authentication, Information and Integration have been identified. In addition, there are tactical, strategic and user role types, all of which have unique requirements important to them. Each user story is associated with a role in this process. A user is the abstraction of all other role types

3 Conception

ID	Epic	Role	User story
US01	Specification	User	As a user, I want to be able to see all the specifications to get an overview of them.
US02	Specification	User	As a user, I want to be able to distinguish between specification versions so that I can recognize positive or negative statistical effects of the versions. Therefore, for each specification its version and development date should be displayed.
US03	Specification	Tactical	As a tactical manager, I would like to obtain statistical information on specifications to identify weaknesses and areas for improvement. This information includes the number of successful cases and cases with problems. It should also be shown how often the case is started per week, the summed average processing time of those cases, and the as-is planned resource time for those cases.
US06	Specification	Strategic	As a strategic manager, I want to be able to select specifications for core processes to mark the specifications that are of high importance to me.
US07	Specification	Strategic	As a strategic manager, I want to filter by specifications for core processes to be able focus on specifications which are relevant to me.
US10	Specification	Tactical	As a tactical manager, I want specifications, cases and work-items to have direct links to the respective YAWL admin interface so I can act immediately.
US11	Case	Tactical	As a tactical manager, I want to be able to see all running cases to get a quick overview of the current operational situation.
US12	Case	Tactical	As a tactical manager, I want to see the specification for each case to be able to classify and distinguish between the cases.
US13	Case	Tactical	As a tactical manager, I want to see temporal information for each case to be able to identify issues. This temporal information is the current runtime and the average runtime of those cases. The average runtime is based on historical data.

Table 6: Some user stories

As already mentioned, all requirements were prioritized according to Gloger’s relative-weight method. This was necessary not only for reasons of completeness of the requirements analysis, but also because simply too many requirements were obtained through the interviews. It would not have been possible to implement all of the requirements within the scope of the project. Gloger’s intuitive prioritization was very well suited to select only the most relevant requirements. Table 7 shows the application of the relative-weight method for all selected requirements, with a priority above 1.7. A complete list can be found in the appendix.

The priority value is based on the ratio between the relative advantage and disadvantage, as well as the ratio between the relative cost and risk. All values are called relative because each value must be seen in relation to the same values of the other user stories. This means that all priorities were estimated based on a feeling. Each estimation was given a value in the range of one to nine. Relative advantage measures the value contributed by a realized requirement. The severity of an unrealized requirement is described by the penalty attribute. Requirements that make a large contribution to solving the fundamen-

tal problem of the system receive a high penalty. The sum of the advantage and penalty is their total amount (Gloger, 2011, p.138). The risk is high for complex requirements for which realization is not easily imaginable, or for which the amount of work required to satisfy them is difficult to estimate. Planning poker values are used to estimate the time costs for user stories (Gloger, 2011, p. 144). A priority is the sum of advantage and disadvantage divided by risk plus cost. Higher values mean a higher priority (Gloger, 2011, p. 139). The requirements are implemented piece by piece in the reverse order of their priority value. Therefore, higher priorities are implemented first.

ID	Relative advantage	Relative penalty	Total	Relative cost	Relative risk	Priority
US01	5	9	14	1	1	7
US02	5	7	12	2	2	3
US03	9	6	15	5	3	1.87
US06	3	4	7	2	2	1.75
US07	3	4	7	1	1	3.5
US10	8	5	13	2	5	1.85
US11	6	7	13	3	2	2.6
US12	4	5	9	1	2	3
US13	9	6	15	3	4	2.14
US18	3	3	6	2	1	2
US19	7	5	12	5	2	1.71
US24	9	9	18	2	1	3
US25	7	6	13	1	1	6.5
US28	8	9	17	5	2	2.42
US29	7	8	15	3	2	3

Table 7: Prioritization of user stories according to the relative-weight method, based on Gloger, 2011, S. 138, 139

3.3 Transformation plan

Based on the current state of the Management Cockpit and the collected requirements, a transformation plan is presented in the following section. This transformation plan is intended to show how the requirements for the YAWL dashboard can be implemented on the basis of the Management Cockpit. For this purpose, it is first described which functionalities of the Management Cockpit could basically be reused for the realization of the requirements. Then, the views for user interaction of the YAWL dashboard are visualized using mockups. The purpose and structure of the individual mockups is explained. Furthermore, it is briefly noted if there are similar views in the Management Cockpit that can be reused. The mockups were drawn in the software balsamiq and show the implementation of most of the user stories (Balsamiq Studios, 2023). User Stories US05, US09, US26, US27 and US33 were discarded because their realization remains unlikely. User Story C01 is a constraint and thus cannot be visualized.

Basically, the requirements for the Management Cockpit at that time differed from those for the YAWL dashboard. Whereas back then it was about being able to configure tailored notifications for exceptional situations, the YAWL dashboard is more about tactical and strategic information communication. The focus is less on configurability than what is

possible with the Management Cockpit. In addition, it is not a requirement for the dashboard to be able to manage YAWL objects such as specifications, cases or work items. For this reason, large parts of the Management Cockpit are not relevant and will be removed. This applies, for example, to all views except the login view and the general layout. However, the fundamental architecture of the Management Cockpit remains relevant. In particular, all communication-relevant structures between frontend, backend, Resource-Client and Dashboard database can be built upon. Accordingly, only the new interfaces need to be constructed and the communication structures for the new relevant data need to be extended. This systematic is discussed further in chapter 4. The rest of this chapter is devoted to the mockups of the views.

Figure 3 shows the mockup of the login page. This page already exists in the Management Cockpit. It is linked to user stories US24 and US25, both of which have already been implemented. No visual adjustments had to be made.

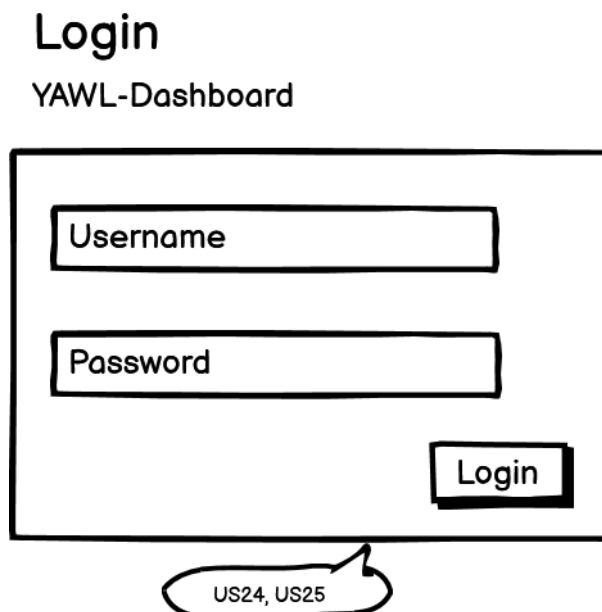


Figure 3: Login page mockup

Figure 4 shows the general layout of the YAWL dashboard. It resembles the Management Cockpit, with subpages easily accessible via sidebar navigation. The web application is a single page application. This means that all subpages are stored right away for later use. Thus, when a user clicks on a navigation link from the sidebar, that particular page is loaded from memory. Costly hypertext transfer protocol (HTTP) requests are thus avoided. The content of the page replaces the content in the large gray area of the mockup. Angular is responsible for the technical implementation of the single page application method (Google, 2023). The components of the general layout were already present in the Management Cockpit. However, they were changed in design and positioning.

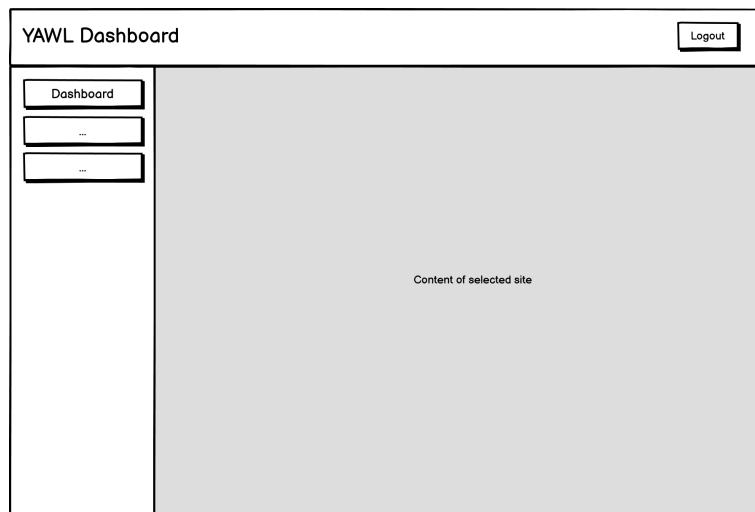


Figure 4: General layout mockup

Figure 5 shows the specification page mockup. It is the default page of the YAWL dashboard and displays a list of all specifications. Specification attributes such as the name, which is called URI in YAWL, version (US02), description, upload date, and active cases are displayed. All table fields can be sorted and filtered in ascending or descending order. An order and a filter can be selected by clicking on the header field. All applied filters appear at the top of the table as a filter chain. An element can be removed from the filter chain by a single click on the element. In addition, a general text search can be used. This allows a user to search for specific items even in very large datasets. These interactions are possible for all tables in the same way. They solve US28, US29 and US30. The interactions are not mentioned in the rest of the mockup descriptions again. This is also true for other functionality that has been mentioned in previous mockup descriptions. Several other functions are visible on the mockup page. Based on US32, it is possible to switch to the strategic view, which will be described later. Because of US06 and US07, users can mark core business processes with the switch icon in the table and filter as described above. A detail page for each specification (US31) can also be accessed using the eye icon. Finally, a link to the corresponding YAWL administration page for specifications is provided for immediate action (US10).

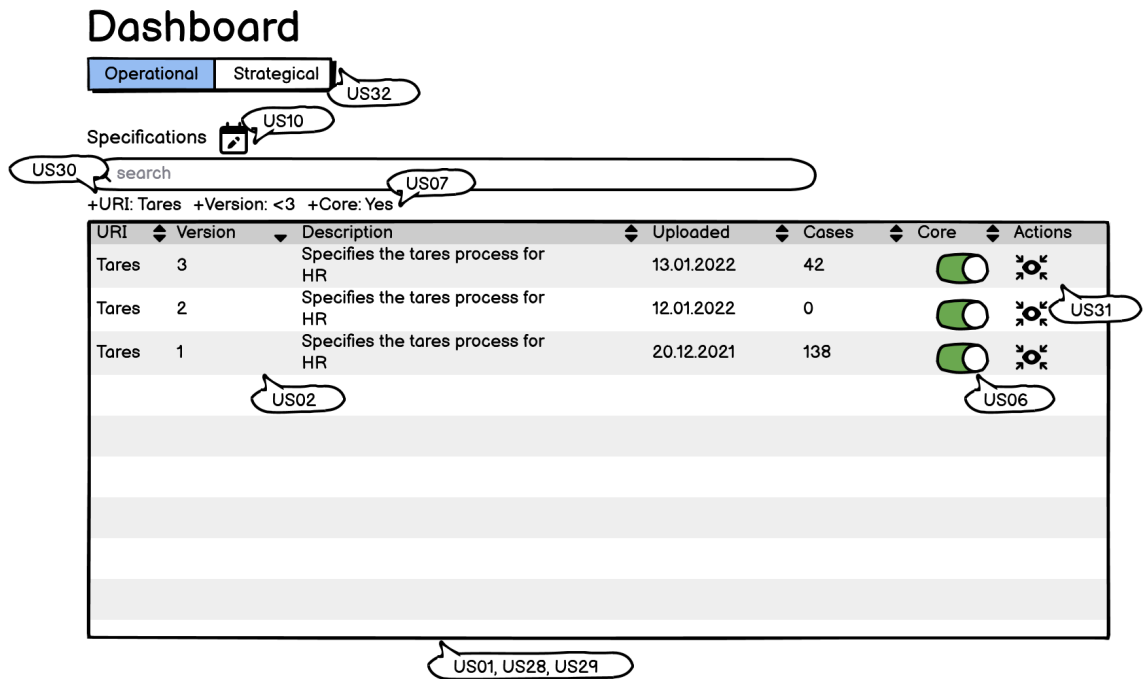


Figure 5: Specification mainpage mockup

Figure 6 shows the details page of a particular specification. The header bar allows the user to return to the general specification page and displays the identifier of the selected specification. This allows users to associate the visualized information in the bottom table with the regarding specification and solves US12. Right below the header bar is a button to be able to switch between Case and Task related information. The task page will be visualized in a later mockup.

The visualized cases in the table presented in Figure 6 show all instances of the specification ever started (US11). The case ID, start date, completion status, and age (US13) are displayed along with the number of running and queued cases. The last three information fields are intended to allow the user to get a quick impression of the case status. In the Actions column, users will find a direct link icon to the YAWL case administration page. Currently, this is the same for all cases, but it is expected that this will change in the near future (US10). Another link icon allows users to see all work items for a particular case. Similar, but not the same, is the Work Item Queue button, which leads to a corresponding modal described later. More statistical information is displayed above the case table. It shows interesting operational information related to all cases of this specification. This is the average completing time of these cases (US13), the average number of cases per day, the number of successful and unsuccessful cases, the maximum capacity of resources for these cases (US03). According to US15, a time limit can be set for each case. Exceeding this limit results in a red case row. According to US04, the average cost of such specification is displayed per week.

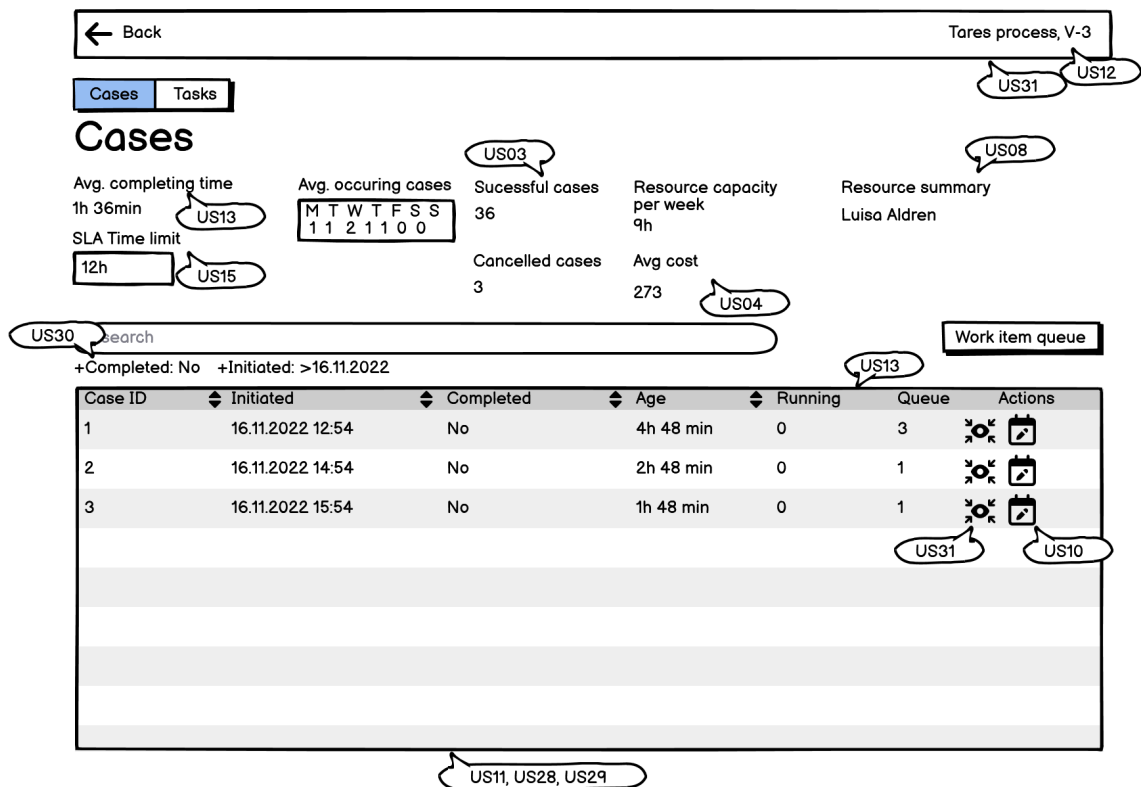


Figure 6: Cases page mockup

Figure 7 shows the details modal for a particular case and contains all work items with temporal and resource information. This solves US17, US08 and again US10, as there is again a link to the corresponding YAWL administration page in the Actions column. The plus icon, currently swapped with the minus icon, can be used to view the details of each work item, completing the hierarchical order. In addition to the average age of similar tasks, another field is provided to specify a maximum time limit. The overdue column of the table informs about exceeding the limit. This solves US18. However, if a work item is in queue, the max queue time limit is displayed instead. In addition, the average time to completion, time in queue and time to reach are displayed. The time to reach indicates how long it usually takes to reach this task in the case. Previous tasks are taken into account.

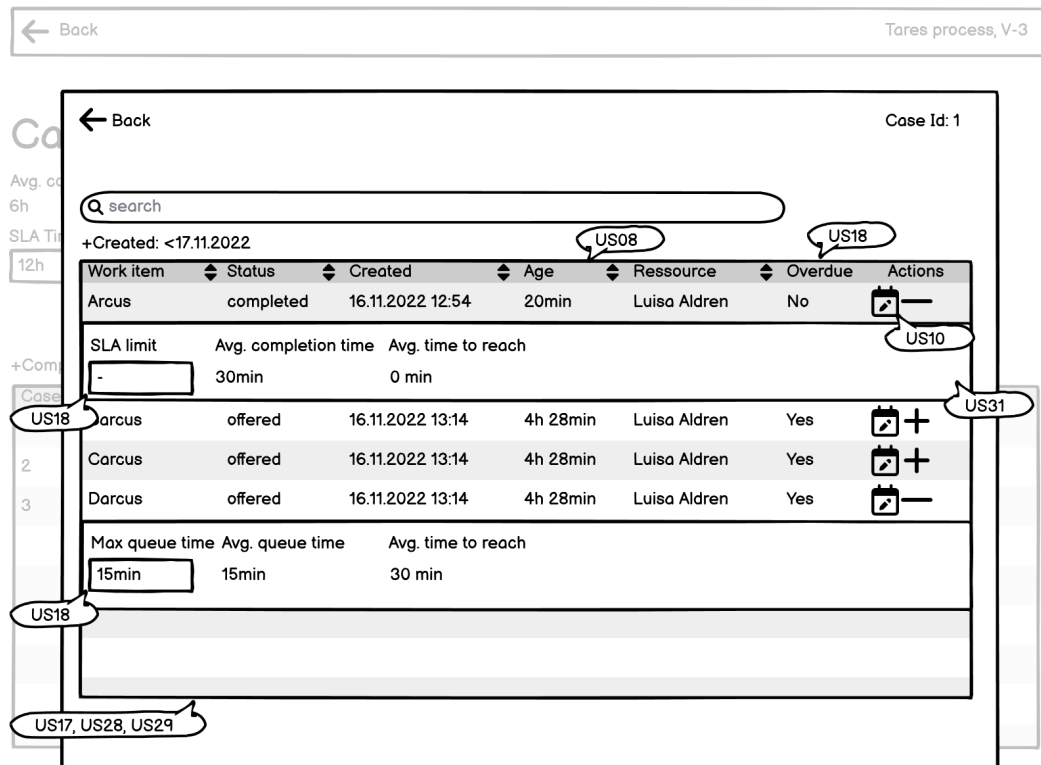


Figure 7: Work items page mockup

Figure 8 is the work item queue modal that contains the work items of all cases of a given specification. It allows users to monitor business progress, detect bottlenecks, and react quickly in case of high congestion (US19). Additional statistical information such as current queue size, average queue size, and average time until a work item is started is displayed at the top (US20).

Figure 9 shows the tasks of the selected specification by clicking the upper switch button. All tasks defined in the specification are displayed. Control flow, data flow and resource flow are disregarded here because they would be too much information to be displayed compactly. Basically, there is again a table that lists all tasks. Each task is identified in the table by its name. In addition, there are several other pieces of information in the table. First of all, there is several statistical information like the average completion time, the average occurrences per week and the average time to reach. The average time to reach solves US14. In addition, other information can be entered that has an impact on other parts of the dashboard. The average cost per resource hour replaces US04 and is the basis for the average cost of a case. Moreover, a time limit can be specified for each task (US15).

3 Conception

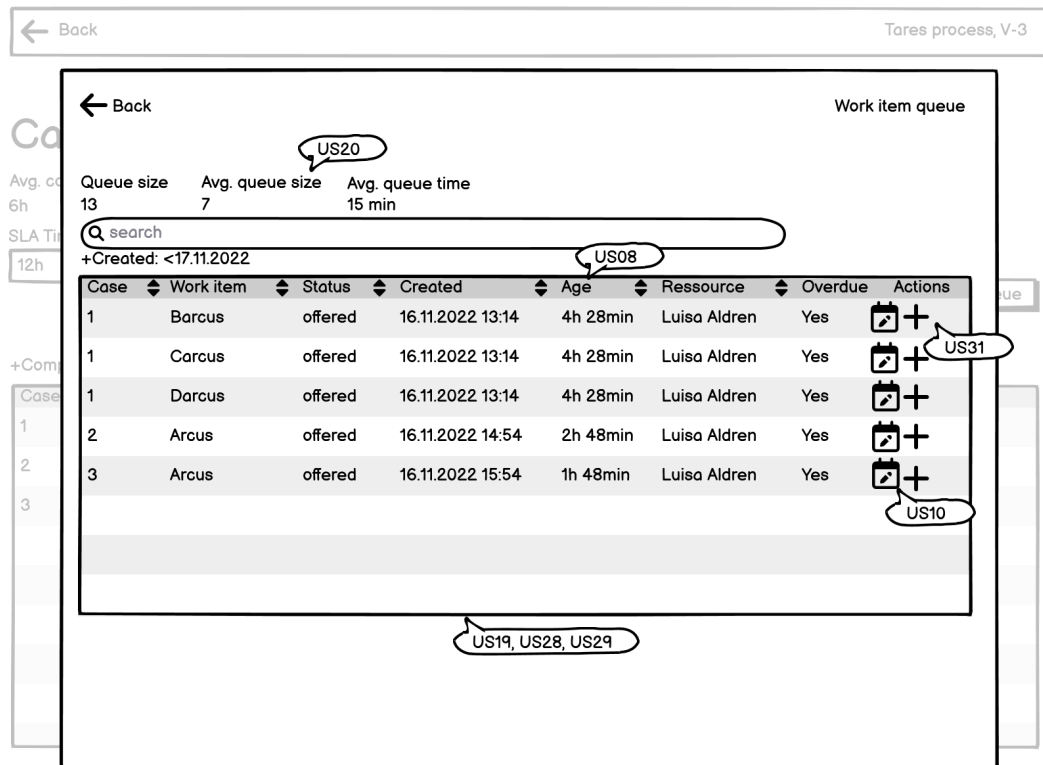


Figure 8: All cases work items queue page mockup

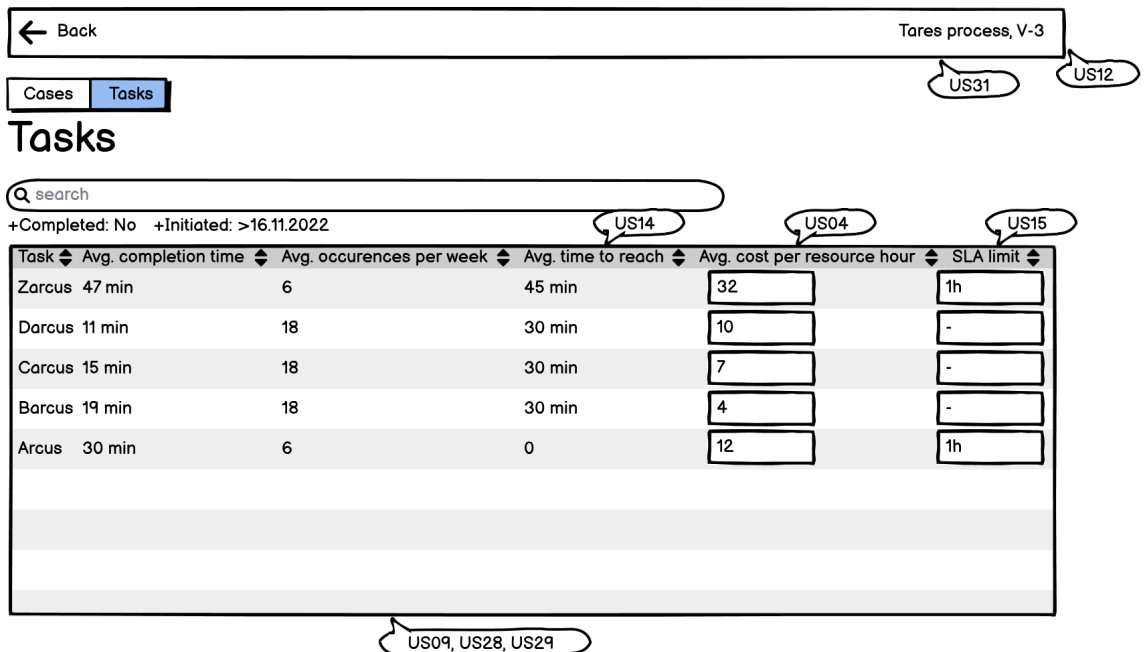


Figure 9: Tasks page mockup

Figure 10 shows the strategic dashboard view. Its purpose is to allow users to view statistical data on a historical basis. Before these statistics can be generated, some specifications must be selected. This can be a single specification or several for comparison purposes. To view the statistics, the Show Statistics button must be pressed. The statistics modal is described below. Another user story was US23. In this user story, it was specified that it should be possible to create benchmarks. The purpose of these is to provide target values for comparison to support progress monitoring. Therefore, not only specifications are displayed in the table, but also benchmarks. These benchmarks can also be selected for comparison. New benchmarks can be created by clicking on the corresponding button, which activates another modal, also described below.

Dashboard

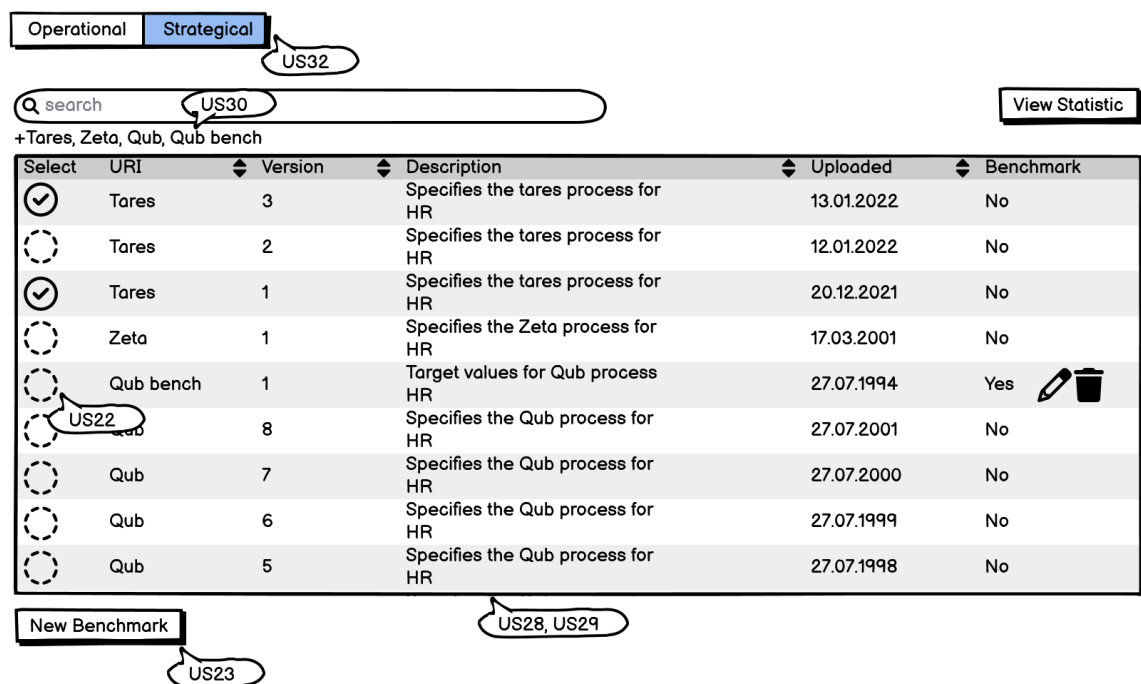


Figure 10: Statistical specification selection page mockup

Figure 11 shows the statistical modal. It solves US21 because several statistical measures can be displayed. These are average case completion time, minimum case completion time, maximum case completion time, average cases per working day, average cost, success rate, average queue time, average queue size, and average number of overdue cases. Each statistic can be selected as a type from the drop-down menu above the chart. A period of individual length and the number of periods per specification can also be configured. The bar chart visualizes the corresponding information, with the periods of each selected specification displayed sequentially.

Figure 12 shows the creation modal for a benchmark. It is a basic form in which the static specification information must be entered. After saving the benchmark, it is persisted and then only used in the strategic view, similar to a specification.

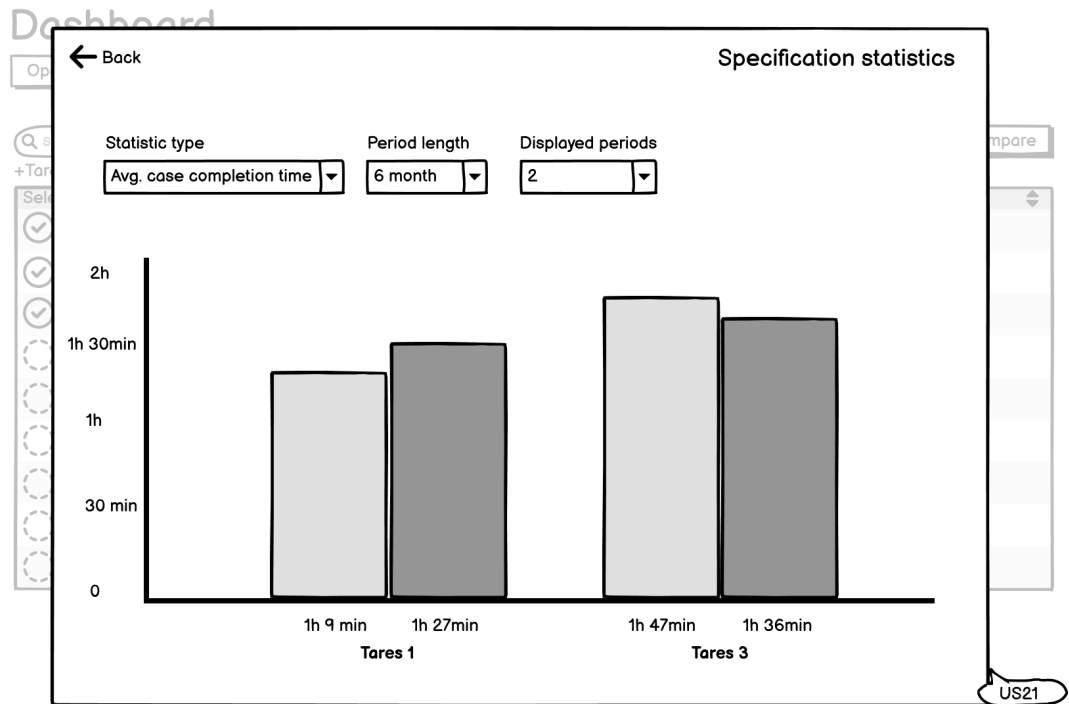


Figure 11: Statistical comparison of specifications page mockup

Edit Benchmark

Back

Benchmark name: Tares benchmark
Version: 1
Description: A very unlikely benchmark for the tares process

Avg. case completion time: 1h
Avg. cases per workday: 5
Avg. cost: 150
Case successrate: 97%
Avg. queue time: 15min
Avg. queue size: 3
Avg. overdue count: 1

Cancel Save

US23

Figure 12: Benchmark creation page mockup

The previously shown mockup views together represent a complete version of the YAWL dashboard. With their development, all necessary information has been collected as a precondition for the implementation. They will guide the implementation in the next chapter.

4 Implementation

The implementation had the goal to realize as many user stories as possible in the modernized Management Cockpit during the project time. The following subsections document this realization. The first subsection deals with the basic approach for the realization of a user story. This is explained by means of an example. The second and third subsections show how the frontend and the backend evolved. The general architecture of the Management Cockpit did not change, therefore the architecture description is not listed again, as it was already introduced in the conception chapter. Finally, a summary is given of which user stories could be implemented and which could not.

4.1 General approach for the implementation of user stories

For each user story, functionality usually had to be added in the form of code in both the backend and the frontend. The functionality requirements of a user story could essentially be divided into three types. (1) User story requires data from the ResourceService (2) User story reads/writes own dashboard data from or to the dashboard database (3) User story does not require any data. Therefore, before writing code for a user story, the first step was to investigate what data was needed and where it could be obtained. For the first two types, a new route had to be added to the backend application programming interface (API) if necessary. In addition, for the first type, the corresponding data had to be obtained from the ResourceService and serialized accordingly. For the second type, data had to be retrieved from the local database.

Finally, if all the required data could be queried via a route from the backend, the HTTP services of the frontend were adapted for those. Each service was responsible for retrieving API data about one entity. Finally, the Angular components were created or further developed. They used the data retrieved from the services to enable user interaction via web interfaces.

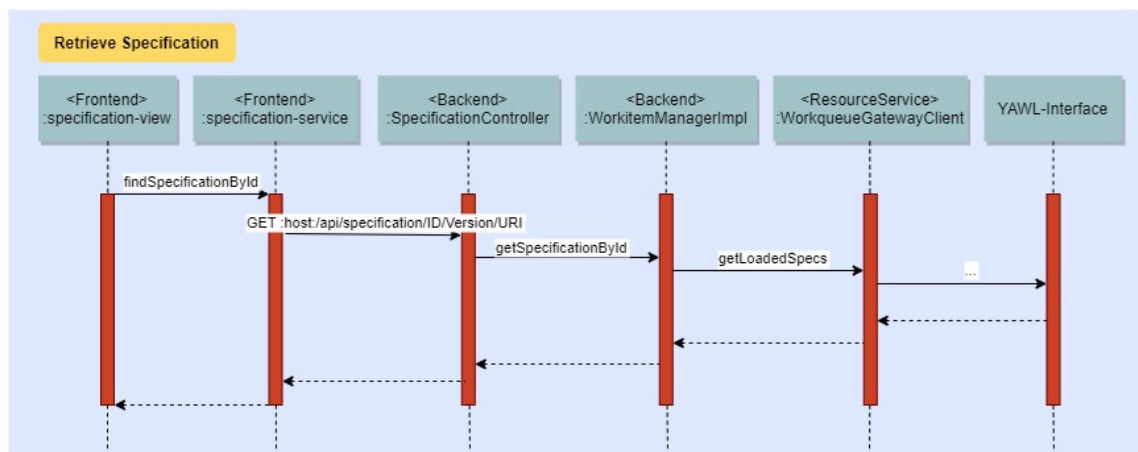


Figure 13: Example specification retrieval from ResourceService

Figure 13 shows a sequence diagram that visualizes the retrieval of specification data from the YAWL ResourceService as an example. User stories that require data from the YAWL

system were implemented in an analogous manner. Data retrieval via the ResourceService was not always straightforward. What additional hurdles there were to overcome will be explained in the next subsection.

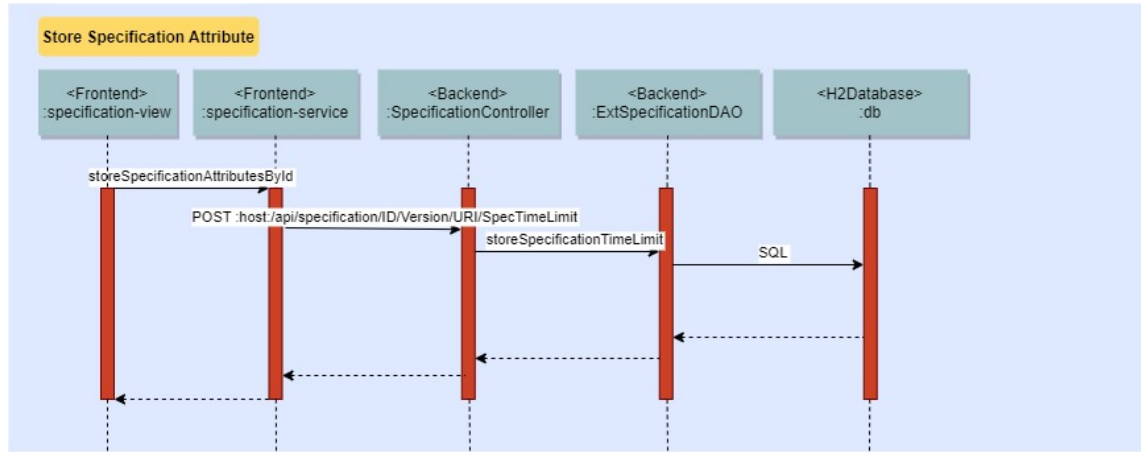


Figure 14: Example specification attribute storing into local database

Figure 14 shows a sequence diagram that describes the storage of a time limit for a specification in the local database. Compared to the data retrieval from the YAWL system, those sequences turned out to be comparatively simple. Of course, this is also due to the fact that fewer different components are involved in the data exchange.

4.2 Backend

The next subsections describe the realization of the capabilities needed for the extended backend component. First, it is shown how the backend of the Management Cockpit and its package structure were fundamentally transformed. Afterward it is shown exemplarily over some code examples, how the communication flow between frontend and ResourceService is controlled. Finally, it is clarified how the values are calculated, that are of interest from a management point of view. The goal of this section is not to give a complete picture of the adaptations in the backend. This documentation is intended only as a rough overview. For a deeper insight, the code should be consulted.

4.2.1 Transformation

As already explained in the approach, the task of the backend was to act as a data interface between frontend, local database and ResourceService. However, old functionalities related to complex notification management or configurable dashlets were no longer needed. Therefore, the corresponding package structure of the backend was adapted. This is visualized in figure 15.

The figure shows all packages, along with their use relationships, that have been removed from the YAWL dashboard. These are all objects marked in red. All objects marked in yellow, on the other hand, were kept and further developed. Each individual package is now mentioned with its rough changes.

The session package enables the authentication of a user via the ResourceService. Here,

the normal authentication via username and password has been restored, as it is common in the YAWL system. Accordingly, the token-based method OAuth was removed, which was implemented in the Management Cockpit for testing purpose.

The YAWLClient package contains the interface to the ResourceService and thereby classes that can be used to communicate with the YAWL system. Since new information was needed from the YAWL system, new connectors had to be written as well. This also included converting the received XML files into JAVA objects with marshallers. An example of an extension of the YAWLClient is the query of all events. This was necessary to be able to calculate all statistical values.

The Model and data access object (DAO) packages have been modified so that they can be used to store the new dashboard data. For example, the different time limits that dashboard users can set. Jakarta Persistence API (JPA) and Hibernate were used to fully abstract the database communication.

Last but not least, the Endpoints package has been modified. It now contains various new routes, which are described in more detail below.

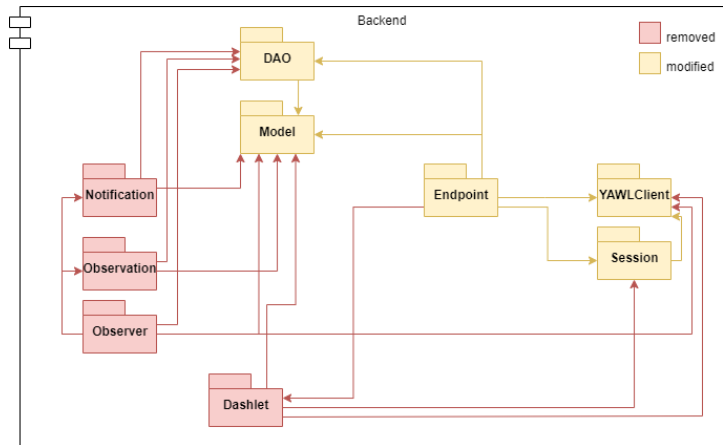


Figure 15: Package difference of the backend

To provide all the data, the REST API of the YAWL dashboard was extended with several new routes. Table 8 shows some of the most important extended routes together with their respective HTTP-Methods.

Method	Route
GET	/api/specification
GET	/api/specification/extension
GET	/api/specification/extension/uri/specificationID/specversion
POST	/api/specification/extension/uri/specificationID/specversion/core
POST	/api/specification/extension/uri/specificationID/specversion/specificationTimeLimit
POST	/api/specification/extension/uri/specificationID/specversion/taskId/costResourceHour
GET	/api/specification/task/uri/specificationID/specversion
POST	/api/specification/task/uri/specificationID/specversion/taskId/maxQueueAge
POST	/api/specification/task/uri/specificationID/specversion/taskId/maxTaskAge
POST	/api/specification/task/uri/specificationID/specversion/taskId/setAttributes
GET	/api/specification/uri/specificationID/specversion
GET	/api/specification/uri/specificationID/specversion/cases
GET	/api/specification/uri/specificationID/specversion/definition
GET	/api/specification/uri/specificationID/specversion/workitems
GET	/api/specification/uri/specificationID/specversion/caseId/workitems
GET	/api/specification/uri/specificationID/specversion/caseId/statistic

Table 8: API extension

4.2.2 Data retrieval

In the following, some code examples are presented for an example route. They are intended to show the basic implementation of such an API extension and to communicate some obstacles that had to be overcome in the process. The sample route is `/api/specification/uri/specificationID/specversion/definition`. It is used to get a list of all tasks in a specification. The backend was implemented in the Spring tech stack, so all code structures refer to common entities of these technologies (VMware, 2023).

Figure 16 shows a method in the `SpecificationController` and the direct extension of the REST API of the backend component. The Java-framework Spring offers various Java annotations to make the definition of the REST route as simple and flexible as possible. The `RequestMapping` annotation basically defines the HTTP route. It also specifies that the route is only accessible via the GET method. Finally, when the corresponding route is loaded, the Java method under the annotation is called with the corresponding parameters. The parameters are replaced by the values set in the corresponding bracketed places in the URL. The task list returned by the method is serialized into the JavaScript Object Notation (JSON) because of the `ResponseBody` annotation and stored in the body of the HTTP response. However, since this task list has to be found first and that is not the job of the REST controller, another class is used for this purpose. This is the `WorkItemManager`, a service that was automatically added to this controller via dependency injection by Spring. It has a method `getSpecificationDefinitionById` that retrieves a task list and is explained by the next figure.

The `WorkItemManager` is a service that directly uses the `ResourceInterface` of the `ResourceService`. It offers a total of four gateways with different responsibilities. A `CalendarGateway`, a `ResourceGateway`, a `LogGateway` and a `WorkItemQueueGateway`. They offer data which is associated to their name. For example, `LogGateway` handles historical data such as events triggered for a case. As we intend to gather all tasks associated to a specification, at this point, the `WorkItemQueueGateway` is of relevance. The gateway provides current data on specifications, cases and workitems. The gateway communicates

```

Robin Steinwarz *
@RequestMapping(value="/api/specification/{uri}/{specificationID}/{specversion}/definition",
    method= RequestMethod.GET)
@ResponseBody
@Transactional
public List<Task> getTaskDefinitions(@PathVariable("specificationID") String specificationID,
    @PathVariable("specversion") String specversion,
    @PathVariable("uri") String uri) {
    return workItemManager.getSpecificationDefinitionById(new YSpecificationID(specificationID, specversion, uri));
}

```

Figure 16: Example route in SpecificationController for retrieval of tasks

directly with YAWL interfaces, so there is no need to deal with the many abstractions and further hurdles of the YAWL system. Figure 17 shows the `getSpecificationDefinitionById` method in the `WorkItemManager` for collecting the task list. To obtain this list, the `WorkItemManager` calls the `getSpecData` method. This method exists in instances of the `WorkItemQueueGatewayClient`, a prebuilt interface class of the `ResourceService`. The identifier connection of this client indicates that this is again an HTTP communication, but it is not necessary to manage it directly. Unfortunately, the YAWL system does not provide REST interfaces, because then collecting data would be even easier.

The data provided by the gateway client is delivered as a string. In order to be able to continue working with them, they are transformed into a Java object. This procedure is called unmarshalling. Two additional YAWL classes, `SpecificationMarshaller` and `YMarshaller` are used for this purpose. With them, finally, a list of the YAWL class `YSpecification` can be created. However, since only a single instance of `YSpecification` can be contained in the data string, only the first list element is ever important. Since this `YSpecification` contains far too much data, only the most important ones are extracted in the next sequence of stream operations. Namely, a list of all task names, which the specification describes. From them a task list is generated, which is finally returned.

```

3 usages Robin Steinwarz *
@Override
public synchronized List<Task> getSpecificationDefinitionById(YSpecificationID ySpecificationID) {
    try (ResourceServiceSessionHandle handle = resourceManagerSessionPool.getHandle()) {
        String specData = connection.getSpecData(ySpecificationID, handle.getRawHandle());
        try {
            SpecificationData specificationData = SpecificationMarshaller.parseSpecificationDefinition(specData);
            List<YSpecification> ySpecification = YMarshal.unmarshalSpecifications(specificationData.getAsXML());
            return ySpecification.get(0).getDecompositions().stream()
                .filter(d -> d.getClass().getName()
                    .equals("org.yawlfoundation.yawl.elements.YAWLServiceGateway"))
                .map(YDecomposition::getID)
                .map(d -> new Task(d, ySpecificationID.getIdentfier(), ySpecificationID.getVersionAsString(),
                    ySpecificationID.getUri()))
                .collect(Collectors.toList());
        } catch (YSyntaxException e) {
            throw new RuntimeException(e);
        }
    } catch (IOException | JDOMException ex) {
        throw new RuntimeException(ex);
    }
}

```

Figure 17: Example method in WorkItemManager to retrieve tasks from ResourceService

With this, this small example is already shown. The task list is returned to the SpecificationController, where it is automatically serialized by Spring into a JSON object, copied into the ResponseBody and returned to the client. However, exactly this place was not always easy, because Java objects with cyclic relations cannot be serialized without further effort. In many places, therefore, such data had to be removed beforehand when it was not of importance for the Frontend.

Communication with the local database was much simpler using JPA, Hibernate and DAO objects. These technologies reduced the required code to a minimum.

4.2.3 Information values for management

As already reported in the requirements section, various information relevant to management needs to be presented in the dashboard. Much of this information is not provided by the YAWL system itself. Therefore, it was necessary to query all case and task events as process log from the YAWL system and to generate the corresponding values from these events. Which values these were and how they are calculated is now shown. Before this can be done, however, the information that was available for its calculation is discussed below.

Basically, the individual events contain six different pieces of information. The specification key, a case id, a task id, a resource id, an event type and a timestamp. For case events, the event types `launch_case`, `complete_case` and `cancel_case` exist. For task events there are the event types `offer`, `allocate`, `start`, `complete`, `autotask_start`, `autotask_complete`, `cancelled_by_case`, `suspend` and `resume`. Suspend and resume are not considered here for simplification. Additionally, the information pieces task id and a resource id are only given if it is a task event. Thus, a distinction between task and case event was possible. Just on the basis of this information, all final values were calculated.

Here, it is to be considered that there can be a multiplicity of events related to each workitem and each case instance. This means that the events and their times had to be assigned to their case or workitem first. These related events are now called reference events. The process of mapping all reference events to their case or workitem was not straightforward, as there was unfortunately no common ID. Instead, the tasks were assigned a deeper decomposition order in case they occurred multiple times, as may be the case with a multiple instance task. A decomposition order arranges tasks according to the sequential arrangement in which they may occur in the process. Accordingly, the first task has the decomposition order 1. Two next tasks, which are connected to the first task via the control flow, could have the decomposition order "1.1" and "1.2". Thus, all reference events could be attached to their workitem or case instance based on their decomposition.

With these reference events related to their case or workitem, further progress times could be calculated, which were relevant for the calculation of the information values. For a case, this is the initiation time and the end time, respectively corresponding to the timestamps of the events `launch_case` and `complete_case`.

Figure 18 shows the progress times for tasks. For a task, the time created is the smallest timestamp of a reference event offered or allocated, because unfortunately no creation event is created. The queue time describes the time that passes until a resource is actually actively dedicated to the concrete task of a workitem. Accordingly, it is the subtraction of the creation timestamp from the start timestamp. The completion time describes only the time a resource spends actively processing a workitem. It is formed from the subtraction

of end and start timestamp. Finally, the lead time describes the time from the creation of a workitem to its completion. The lead time is the end timestamp minus the creation timestamp.

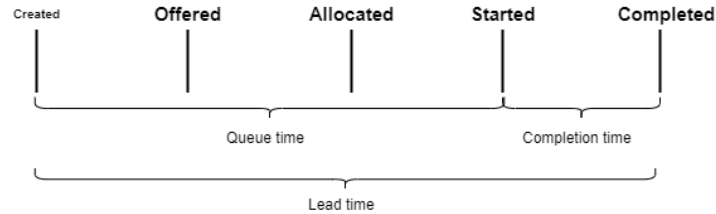


Figure 18: Relevant progress times of a task

With this basic information in mind, it can now be shown how the individual target values were actually calculated. This is done by table 9. In the table, each individual target value is associated with the process element to which it is assigned. S means specification, C means case and T stands for task. In the column Calculation, a short summary of the calculation procedure is given in natural language. In addition, no cases or tasks that have been canceled are included in the statistics. They are not meant in relation to cases. Only if they are explicitly addressed, they are meant.

4 Implementation

Level	Target Value	Calculation
S	Ressources summary	All participants involved in the events of the specification so far
S	Avg. case completion time	All ages of all cases summed up and divided by all cases. Only cases from this specification
S	Successful cases count	The number of all cases from this specification that have an end timestamp
S	Active cases count	The number of all cases from this specification that have no end timestamp
S	Unsuccessful cases count	The number of all cancelled cases from this specification
S	Avg. occurrences of cases per day of week	The sum of all cases initiated on a weekday divided by all weeks in which a case occurred. Only the cases of this specification
S	Avg. ressource capacity needed per Week	The average completion time of each task multiplied by the average occurrences of the task in a week summed up. Only the tasks of this specification
S	Avg. cost per week	The average completion time of each task in hours multiplied by the average occurrences of the task in a week multiplied by its specified costs per week hour summed up. Only the tasks of this specification
C	Age	The end time stamp minus the start time stamp. Unless the end timestamp is not given. Then the current timestamp minus the start timestamp
C	Queued workitem count	Sum of all workitems of a case whose start time stamp is not available
C	Running workitem count	Sum of all workitems of a case whose end time stamp is not available
T	Common resources	All participants involved in the events of the task of this specification so far
T	Decomposition order	Usually part of the case id of common events
T	Avg. queue time	The queue time of all workitems with the same name divided by the number of all workitems. Workitems of all cases from this specification are meant
T	Avg. completion time	The completion time of all workitems with the same name divided by the number of all workitems. Workitems of all cases from this specification are meant
T	Avg. lead time	The queue time plus completion time of all workitems with the same name divided by the number of all workitems. Workitems of all cases from this specification are meant
T	Avg. time to reach	The average lead time of all tasks summed up, which are further back in the decomposition order. Only tasks from this specification
T	Avg. occurrences per day of week	The sum of all workitems initiated on a weekday divided by all weeks in which this workitem occurred. Only workitems with the same name as the task and of this specification are meant

Table 9: Calculation of information values for management

These information values for management are calculated in the backend and displayed in various places in the frontend.

4.3 Frontend

The next subsections describe the realization of the new capabilities needed for the frontend component. These capabilities were to retrieve the corresponding data via REST routes from the backend and to provide the user interfaces that were already shown as mockups in the previous chapter. For this purpose, a brief summary of the frontend transformation is given. After this summary, a developed view is shown as an example, and it is briefly explained how those were developed in principle. For a more detailed picture, the code should be consulted.

4.3.1 Transformation

As visualized in the figure 19, the frontend of the Management Cockpit contained various modules for managing notifications and data that are no longer relevant. These modules are marked in red in the figure and have been removed from the frontend. This also applies to the red arrows, which all indicate use relationships. Additionally, a new module named new-dashboard was created, which contains all components for visualizing the mockups. This module and the other reused components will now be briefly discussed.

For the new-dashboard module, highlighted in green, the Angular components dashboard-new, specification-view, case-view, task-view, workitemqueue-dialog, workitems-dialog and timestampform were developed. These Angular components consist of different documents that together form a self-contained part of the user interface (Google, 2023). Almost each of these components represents exactly one mockup and visualizes its respective data. Before visualization, the data is received from the backend interface, which is in the yawl package.

The common package serves as an outsourcing of aiding structures. For example, this is the basic layout of the YAWL dashboard that was customized.

The YAWL package contains the communication interface to the backend. Various extensions have been made here to take full advantage of the extended API of the backend. For example to get the statistical data.

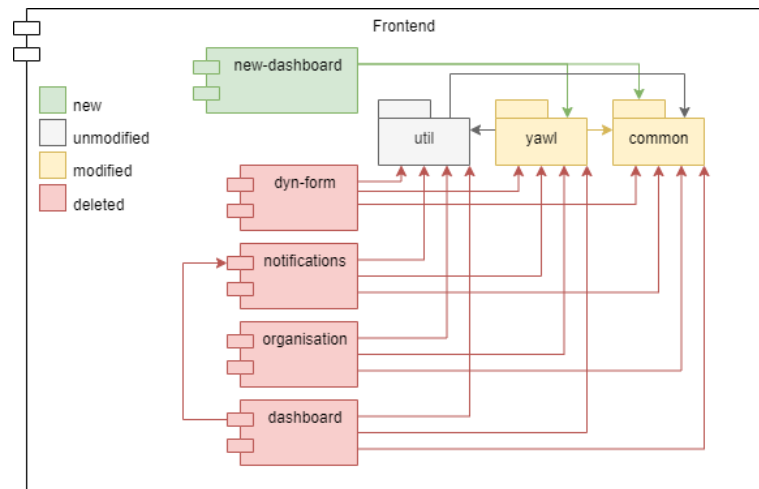


Figure 19: Package difference of the frontend

4.3.2 View

Each view in the frontend consists of one or more nested Angular components. The figure 20 shows one example view. That is the view which corresponds to the Specification mainpage mockup. It is available via the route `/dashboard` and shows the Angular component specification-view. Only Angular material components were used to implement the interfaces to achieve a simple, user-friendly and effective design. Angular material components are standardized and predesigned web elements. One of these is for example the table in the figure. Unfortunately, this interface also shows a circumstance that had to be addressed during implementation. Some data could not be obtained through the ResourceService. This included the time when a specification was uploaded. This column remains empty. However, it was implemented anyway, as it is hoped that this presumably small adjustment of the ResourceService will still take place.

Dashboard

☒ Operational ☐ Strategic

Specifications 






URI	Specification Version ↓	Documentation	Uploaded	Active Cases	Core Case	Actions
ReiseangebotEntwicklung	0.79	Workflow für die Entwicklung eines Reiseangebots	-	2	<input type="checkbox"/>	
ReiseangebotEntwicklung	0.78	Workflow für die Entwicklung eines Reiseangebots	-	2	<input checked="" type="checkbox"/>	
ReiseangebotEntwicklung	0.77	Workflow für die Entwicklung eines Reiseangebots	-	0	<input checked="" type="checkbox"/>	
ReiseangebotEntwicklung	0.76	Workflow für die Entwicklung eines Reiseangebots	-	0	<input type="checkbox"/>	
_1	0.1	No description provided	-	0	<input type="checkbox"/>	

Figure 20: Specifications view

As mentioned earlier, all interfaces were developed almost exactly according to their mock-ups and implemented in Angular. However, visualizing all of those interfaces is beyond the scope of this documentation. Therefore, the following section instead summarizes which user stories were developed.

4.4 Realized user stories

Unfortunately, as predicted, the project time was too short to realize all user stories. Table 10 shows the achievement of each functional user story. There were 24 user stories realized out of 33. Except for those and US22, US29, US30, all user stories were realized with a priority higher than 1. Due to the technologies used, the YAWL dashboard is easily transformable into a war-file. Accordingly, the constraint C1 is fulfilled.

ID	Realized	ID	Realized	ID	Realized
US01	Y	US12	Y	US23	N
US02	Y	US13	Y	US24	Y
US03	Y	US14	Y	US25	Y
US04	Y	US15	Y	US26	N
US05	N	US16	Y	US27	N
US06	Y	US17	Y	US28	Y
US07	Y	US18	Y	US29	N
US08	Y	US19	Y	US30	N
US09	Y	US20	Y	US31	Y
US10	Y	US21	N	US32	Y
US11	Y	US22	N	US33	N

Table 10: Realized user stories

5 Summary

The goal of this work was to increase the competitiveness of the YAWL system by developing a dashboard. The identified work packages for this were the modernization of the Management Cockpit, its conceptual further development and its implementation. Basically, all work packages were successfully completed. The Management Cockpit was successfully modernized. The technologies were updated and the associated code migrations were carried out. The requirements elicitation and the associated conceptual planning phase were also successfully completed. Valuable insights into dashboard requirements were gathered via requirements engineering methods, which may be of value beyond this project and implementation. In addition, the realization of most user stories was shown in mockups, which could also be of value beyond the project. Through the implementation phase, most user stories could be realized. However, especially the user stories related to strategic information are not realized. Yet this was already foreseeable during the requirements phase and agreed with the developers of the YAWL system. With this in mind, the central goal to provide a valuable new dashboard for the YAWL system is fulfilled.

Nevertheless, a new major version of the YAWL system was released during the project phase. As a result, the YAWL dashboard would need to be adapted again for the latest YAWL version. It has not been verified that the current version of the YAWL Dashboard is compatible with this latest version.

This project has done some work that can help increase the competitiveness of the YAWL system. However, it is up to the developers of the YAWL system to decide whether to actually integrate the YAWL dashboard after the project is complete. Only then will the full added value of this work actually materialize.

For the time after this time-limited project, there are several pieces of work that can still be completed. That is the implementation of all remaining user stories and the migration of the YAWL dashboard to YAWL version 5.0. With a bit of luck, nothing further needs to be done for this. Then the YAWL dashboard can be finally integrated by the YAWL developers.

6 Bibliography

- Balsamiq Studios, L. (2023). Quick and easy wireframing tool [(Last access: 13.01.2023)]. <https://balsamiq.com/wireframes/>
- Bergsmann, J. (2014). *Requirements engineering für die agile softwareentwicklung: Methoden, techniken und strategien*. dpunkt.verlag, Heidelberg.
- Center, U.-C. S. O. (2022). National vulnerability database [(Last access: 27.10.2022)]. <https://nvd.nist.gov/>
- Gloger, B. (2011). *Scrum: Produkte zuverlässig und schnell entwickeln, 3., aktualisierte aufl.* Carl Hanser Verlag GmbH Co KG.
- Google. (2023). Angular [(Last access: 13.01.2023)]. <https://angular.io/guide/router-tutorial>
- Hofstede, A. H. M., Aalst, W. M. P., Adams, M., & Russell, N. (2010). *Modern business process automation: Yawl and its support environment*. Springer Berlin, Heidelberg.
- Lilienthal, C. (2019). *Langlebige software-architekturen: Technische schulden analysieren, begrenzen und abbauen, 3. auflage*. dpunkt.verlag.
- MvnRepository. (2022). Mvn repository [(Last access: 27.10.2022)]. <https://mvnrepository.com/>
- npm inc. (2022). Npm - build amazing things [(Last access: 27.10.2022)]. <https://www.npmjs.com/>
- Rupp, C. e. a. (2014). *Requirements-engineering und -management: Aus der praxis - von klassisch bis agil, 6. auflage*. Carl Hanser Verlag.
- The YAWL Foundation. (2022). Yawl bpm website [(Last access: 14.10.2022)]. <https://yawlfoundation.github.io/>
- Thomas, P. R. (2017). Masterthesis -.
- Thomas, Philipp René. (2022). Yawl-dasboard [(Last access: 15.11.2022)]. <https://github.com/Floaz>
- VMware, I. (2023). Hear from the spring team this january at springone [(Last access: 13.01.2023)]. <https://spring.io/>

7 Appendix

7.1 Protocols

Below is a list of all interview transcripts conducted as part of the project.

Protocol 1

Participants: Michael Adams, Andreas Hense

Interviewer: Robin Steinwarz

Date: 10.11.2022

Time: 9:00-10:30

Type of the interview: digital group interview

Steinwarz	"Thank you both for your participation. I would like to record this meeting to be able to capture all the requirements between the lines as well, if that's okay with you."
Hense, Adams	"Okay."
Steinwarz	"Fantastic. For clarity, I'll briefly summarize the goal of today's meeting. We are meeting to document your functional requirements for the YAWL dashboard. We will do this in a structured way. For this purpose, I have prepared some questions that I would like to ask you one by one. At this point I would like to add that today we want to find all requirements. Not only those that are realizable in the time frame of this work, but really any valuable idea that comes to your mind. And if you both agree, I would like to focus on Mr. Adams first."
Hense, Adams	"Yes, that's fine."
Steinwarz	"Okay, let's start the dialog right away. Please put yourself in the position of a tactical manager. Explain the situation of the tactical manager. What information does he need?"
Adams	"In my opinion, the dashboard is important. Typical things you want to see there are the cases that are currently running and what those cases are. So what specification they solve. Also, some statistics about their timing. For example, how long they've been running. The average run time of those cases. Maybe you can compare that to some historical benchmark case instances. Also looking for bottlenecks. In other words, showing work items that have been with a participant for an extended period of time. Additionally, the number of successful cases, cases with problems, problems with data, or maybe problems with control flow."
Steinwarz	"How can the dashboard satisfy this need for information?"

- Adams "There are several ways to visualize this information. A simple way would be a tabular view showing some statistics against specifications. So how many work items are running, how long have they been running? You could try to make it fancier by using some graphs or dials or gauges to show if times are within a threshold or if they're over or under, if they're meeting certain service level agreements. Possibly a bar chart or two as well."
- Steinwarz "Is the manager able to configure these graphics?"
- Adams "I'm not sure. I mean, to some extent it would be an interesting functionality. Maybe instead you could offer the possibility to switch between the regular tabular view and a detailed view for a specification."
- Steinwarz "Did I understand you correctly? Is it that configuration is not necessary for the information displayed on some kind of main page?"
- Adams "It would certainly be a nice to have, but not base functionality"
- Steinwarz "Now please put yourself in the position of a strategic manager. Can you also describe his perspective in comparison to the other role?"
- Adams "Strategic management is more interested in trends over time. Rather than just looking at current cases. It would be more about how cases have trended over a 6 to 12 month period. Are our specifications fast? Is the efficiency decent? Do we have benchmarks and are we exceeding them? Are there places where we can improve the cases? So it's about mid- to long-term aspects of improvement. What can we do better? What can we do cheaper?"
- Steinwarz "So are they looking at the same information, but in a historical sense?"
- Adams "I think it could be the same. A line chart would be a good way to show extracted historical data. And I think the strategic view would also have tabular data representations, but only for core business specifications."
- Steinwarz "So there is a difference between normal and core business process specifications. Is this distinction already implemented in the YAWL system?"
- Adams "No. But there could be some form of selection that strategic managers make. This would enable them to see the cases that are strategically important to their meetings. Certain departments might be interested in certain specifications."
- Steinwarz "That's it for the roles. Now please answer the next questions intuitively. How do you imagine interacting with the dashboard?"

- Adams "Again, the basic functionality would just be a view with some tables where you can select rows and view detail views. That's really the only interaction you need. Maybe some additional functionality to compare specifications, for example in terms of their resources. Yes, there could also be a resource view for specifications. Specifications could also be broken down into activities or tasks, where additional resource and performance information is stored. Perhaps statistics on how long a task has been running. As for interaction, it's really just a tabular view with some details. Anything else would be unnecessary and not basic functionality."
- Steinwarz "Please describe a simple scenario for using the dashboard. What steps need to be taken in detail?"
- Adams "After the login, there would be a table with the list of specifications, if you click on one of this lines you see the details. And this is basically it."
- Steinwarz "Is there another not-so-common scenario that comes to mind?"
- Adams "I think if there are issues with a process, someone should be able to break it down. To understand if it's a data-related problem or a resource-related problem? Is it because we have more instances of a process than we have resources to handle them? Are work items queued up waiting to be started, but resources are busy with other tasks. Perhaps some queueing theory. How much time passes before a task is started and completed. How many tasks are in the queue on a typical day or week? Is that number increasing or decreasing? With this information, you could answer whether you need to allocate more resources to certain activities by looking at the throughput of each activity."
- Steinwarz "Do you have ideas for user management?"
- Adams "This is done with the YAWL system."
- Steinwarz "Which users can log in to the dashboard?"
- Adams "For now, administrators will serve. But actually you want users to be able to use the dashboard that are in between a normal user and an administrator. Because some kind of line manager may not want to see all the administrative functions of YAWL. So there should be a new privilege in YAWL for this purpose."
- Steinwarz "Perfect. Thank you very much for all your answers. We have gone through all the questions and will now continue with an open discussion. Mr. Hense, please express your opinion now as well."
- Hense "That's good!"
- Adams "My approach would be to start very simple. Just a simple table with all current cases in it."
- Steinwarz "The work is actually based on the work of Philipp Jan Thomas, so fortunately such a table already exists. My current goal is to modernize his version of the dashboard and build on it."
- Adams "Of course. Must have been many iterations of Spring Boot since then!"

- Hense "I do indeed have some points that came to mind during the conversation. Perhaps we can discuss these. First of all, the tabular views should be sortable by their content headings. It should also be possible to filter the tables. Especially for different versions of specifications. Filter chains could also be a nice feature to drill down to a specific question. That's one thing. The other thing is the issue that Mr. Adams mentioned earlier, tracking trends over time. It would be very interesting to see statistics on all the versions of a specification over a period of time. That could lead to information about whether some versions had a positive or negative impact on turnaround times or something like that."
- Adams "Absolutely, that's a good idea!"
- Hense "Also, it would be very good to see, for example in a line chart, when exactly a specification was uploaded. Another point that stood out to me was the idea that you can take immediate action on work items. For example, to unblock a situation. In this scenario, I would like a way to go directly from the dashboard to the administrative board in the regular YAWL system."
- Adams "Yes, you would definitely need administrative rights for that. What kind of measures do you think need to be taken?"
- Hense "For example, if a work item is blocked, someone is on vacation, or something similar; you want to quickly assign the work item to someone else. Or you want to cancel the operation. In these cases, a bridge to the regular administration web page would be good."
- Adams "If the user has the can-manage-cases privilege and the dashboard privilege, he can certainly do this."
- Hense "Sure, if we are talking about a normal administrator, we don't have this problem. But you are right. For those fine-grained rights, that needs to be well thought out and included in the concept. I was thinking of a process owner role that would be able to view the dashboard and edit some specifications. I mean, that doesn't exist right now and would kind of break up the admin role. I mean, that's not something that we should implement right away."
- Adams "We could do something similar with the can-manage-cases privilege, because it already exists. We could allow users with that privilege to see the dashboard. That fits quite well."
- Hense "I believe there are also team and group privileges for cases."
- Adams "Yes, they are at the task level."
- Hense "There are certainly a lot of privileges. My idea was to figure out which ones are relevant to the dashboard. Maybe you could introduce a feature that guarantees that a user only sees information related to a specific domain. That would increase the focus of that particular user."
- Adams "These permissions are also called can-view-team-items and can-view-group-items. There are also some other permissions that allow users to start work items in a certain way. This is really the task level."

- Hense "Right, there are many privileges, and I think it might be useful to explore whether these different privileges can add value to the dashboard in terms of the context in which it is used."
- Adams "Yes. The user rights set up can be seen in the participant data pages, while the task rights can be seen in the editor in the resourcing dialog."
- Hense "Yes, in some tabs after the first one."
- Steinwarz "Are there any restrictions on integration?"
- Adams "No. If everything you build can be built into a .war file, it integrates quite well."
- Steinwarz "Okay. Is OAuth necessary for the dashboard?"
- Adams "No. The standard authentication is completely sufficient."
- Steinwarz "Ah okay, I was just wondering because Philipp Rene Thomas version uses it. But I couldn't find the authority within the YAWL system that authorizes token requests."
- Adams "No, Mr. Thomas did this as an extension. The basic authentication is sufficient."
- Hense "There is one more thing I would like to ask. Philipp has created notifications in his work. Notifications that pop up when certain conditions are met. Do we want to keep those notifications? I mean, we have all these views, we can set benchmarks and we can look at historical data. For example, we already know now based on the information presented that some executions take longer than usual. The question remains, do we really need these notifications?"
- Adams "In my opinion, the notifications feature is a nice-to-have. It is not absolutely necessary. But if you could set a parameter, like 'I want to receive a notification when case 26 runs longer than a week'. That would be a nice-to-have, but not an essential implementation candidate."
- Hense "Yes, this could be an additional lot of work as well, since you would have to set all these parameters to get the right level of notification diversity."
- Adams "Yes, you would have to be able to configure it."
- Hense "Speaking about this topic. Do YAWL users currently have email addresses? Urgent notifications could also be sent via email."
- Adams "Currently, we don't have that. But it could be another improvement in the future."
- Steinwarz "Okay, I think that's it for now. Thank you for your time, your patience, and all the good answers. If it's okay with you, I'd like to have a second conversation in about three weeks. The second conversation will be to fine tune the requirements and prioritize."
- Adams "Yes, that would be fine with me."
- Hense "Yes, I think it would be useful if we could see some mockups by then."
- Steinwarz "Of course, I will prepare them."

[A short excerpt from a follow-up discussion with A. Hense and R. Steinwarz]

- Hense "Regarding the possibility to take immediate action in the dashboard again the following example. For example, if a work item is too old, then you can go to its detail view in the dashboard and see which resources the work item is assigned to. In addition there is for example a link that, when clicked, immediately takes you to the corresponding YAWL user interface, where you can assign the work item to someone else."
- Steinwarz "Ah! Now I understand what exactly you mean."
- Hense "Other things I thought of would be a comparison between the age of the currently selected work item and the average age of the other items. Also a separation of tactical and strategic information. By the way, in the case view all activated cases should be displayed with the corresponding work items. Here it should be possible to filter between active, completed and the case version."

Protocol 2

Participants: Michael Adams, Andreas Hense

Interviewer: Robin Steinwarz

Date: 06.12.2022

Time: 9:00-10:00

Type of the interview: digital group interview

- Steinwarz "Thank you both for your second participation. If you don't mind, I would like to record this session again to make it easier for me to take the transcript."
- Adams, "Yes."
- Hense
- Steinwarz "Today's topic is fine-tuning and prioritizing the requirements I prepared during our last interview. We will start by resolving the questions that came up as I wrote them down. Feel free to make notes of any information that comes to mind. After that, we will focus on prioritization. I have prepared three lists for this purpose. One with requirements that are certainly feasible within the scope of this work, another list with uncertain requirements, and a final one with requirements that are certainly not feasible. From the list of uncertain requirements, you can both select some that I will then prioritize further. Do you have any questions now?"
- Adams "Yes, is the document that you are presenting right now the same one that you sent us?"
- Steinwarz "It is the same, but I have recently made some minor changes to it. The list of requirements received is the same. Have you already checked them?"
- Adams "Yes, I took a quick look at it and it looks very good."
- Steinwarz "Oh, have you read through them yet?"
- Adams "Yes, I know them."
- Steinwarz "That's perfect! Because Mr. Hense knows them, too. This allows us to really focus on the ambiguous parts that I've marked in red. Without hesitation, let us start right away."
- Steinwarz Readout of US04 - "Is that what you mean? Please give me a brief explanation on this subject."
- Adams "It would be really good for an organization to measure the cost of a case. But cost refers to resources, the time a resource spends on a task, taking into account salary, machine costs, and so on. This can be assigned as an average rate per minute for each task. It is fine if this is done in a very rough way. An approximate value would be good."
- Steinwarz "So it's really just a value that is assigned to a resource hour and then totaled in the operation, right?"

- Adams "Yes. Of course, it would be good to pursue this at a finer level, where the cost of each resource can be considered. But that is certainly beyond the scope."
- Steinwarz "Perfect. I think I know what you mean."
- Hense "Michael, does it have anything to do with the cost service?"
- Adams "Yes, the cost service does just that, but it is really complicated at a low level. There has never been a user interface created for the cost service. You can do this using XML, but this is rather a future extension."
- Hense "Yes, that is true."
- Steinwarz "Okay, let's move on to US07" - Reading US07 - "I understood that for work items. There can be one or more resources associated, but I didn't understand it for specifications."
- Adams "I guess it just means aggregating the resources currently associated with a case's workload."
- Steinwarz "All right. Next." - Readout of US08 - "I was not sure how this requirement was meant. Do you mean a visualization of tasks with their control flow and resource information?"
- Hense "I think it is not necessarily the complete relationship between tasks at the specification level, but rather a list of all tasks in a specification with information about them. For example, what tasks exist for a specification, what resources are associated with it, what is the history of that task."
- Steinwarz "Okay, I think it's clear now what you mean. Let's move on to the Epic Case." - Reading out US12 - "Is there any other temporal information on the case that you would like to get?"
- Adams "Well, I think you can break it down to several levels. For example, specification level, case level, and work item level. For example, it would be very interesting to know how long it takes on average to reach a certain point in a specification, to compare it with the current situation. Because this would indicate some problems with the current cases. For example, one could see the difference between young and older employees."
- Hense "Yes, that makes perfect sense to me, especially in terms of being able to see statistics up to a certain point. But it's certainly not trivial."
- Adams "Admittedly, some of this information is not easy to come by."
- Steinwarz "Good. Let's move on to the next user story" - Reading US13 - "Can they also be set for any other object?"
- Adams "Again, it would be great to be able to set this at the task level. For example, if we have a sales process where we have to meet customer expectations, it would be good to have a soft limit and a hard limit so that we get a warning if it looks like we are not meeting our agreements. Again, I'm just brainstorming here."

Steinwarz "Is the soft limit based on average completion time?"
Adams "Yes, that could be an option that we have."
Steinwarz "Okay, let's move on." - Reading of US15 - "The question here is what does too long mean?"
Adams "Too long again means a limit that a manager must configure."
Steinwarz "Like the soft limit?"
Adams "Yes, very similar, but at the task level. It could be the average time again."
Hense "For each task there is a waiting time and an execution time, for which we can of course configure maximum values."
Adams "Yes, and this can be easily determined by looking at the state of the work item."
Steinwarz "Okay. Next." - Reading of US16 - "Are the cases relevant in this case?"
Adams "No, it is only work items."
Steinwarz "Okay, okay. Next." - Reading US17 - "We're talking about Work items here too, right? It's the same problem as last time?"
Adams "Yes, right."
Steinwarz "Okay, then let's drop this user story and move on." - Readout from US18 - "This is where I had the question of exactly what was meant in terms of temporal information. I had a few ideas, but...."
Adams "Average, minimum and maximum duration per case. It should also be possible to specify individual time spans. Not only 6 or 12 months. For example 9 months or 12weeks."
Steinwarz "Really good, I'll write it down. We are now on our last user story." - Reading out US20 - "I need to quickly remember what the problem was."
Adams "I interpreted the problem to mean that benchmarks can be collected by looking at the history of past cases to get statistical information. But I also want to set my own benchmarks for some targets."
Steinwarz "Ah, yes. Then we are also done with this user story and with the first part. Let's continue with the prioritization."
[...]
Steinwarz "Please review the list of uncertain requirements for implementation and let me know the requirements that are most important to you."
Adams, "Okay."
Hense
[...]
Steinwarz "Are you ready?"
Adams, "Yes, we are."
Hense
Steinwarz "Good. First of all, please give us your impression, Mr. Adams."
Adams "Important to me are US03, US04, US08, US15, US19 whereas US06, US10, US17, US22, US30, US31, US32 are only nice to have."
Steinwarz "Okay. Mr.Hense?"
Hense "For me, US03, US10, US15, US17 are important and US19 is especially important."
Steinwarz "Thank you both. I will adjust the prioritization accordingly. That was the last productive part, but please let me show you the mockups I prepared."
[...]

7.2 Complete list of requirements

ID	Epic	Role	User story
US01	Specification	User	As a user, I want to be able to see all the specifications to get an overview of them.
US02	Specification	User	As a user, I want to be able to distinguish between specification versions so that I can recognize positive or negative statistical effects of the versions. Therefore, for each specification its version and development date should be displayed.
US03	Specification	Tactical	As a tactical manager, I would like to obtain statistical information on specifications to identify weaknesses and areas for improvement. This information includes the number of successful cases and cases with problems. It should also be shown how often the case is started per week, the summed average processing time of those cases, and the as-is planned resource time for those cases.
US04	Specification	Strategic	As a strategic manager, I would like to configure the resource costs in the specifications for each task to see the total cost of a case for reference in the statistical view. This cost is an estimated average amount per hour.
US05	Specification	Strategic	As a strategic manager, I would like to configure the resource cost for each employee to get a more accurate amount of the cost for each case in the statistical view.
US06	Specification	Strategic	As a strategic manager, I want to be able to select specifications for core processes to mark the specifications that are of high importance to me.
US07	Specification	Strategic	As a strategic manager, I want to filter by specifications for core processes to be able focus on specifications which are relevant to me.
US08	Specification	User	As a user, I want to see resources on specifications and work items to be able to search for improvement opportunities. Resources are the assigned employees. At the specification level, this means a summary of all matching resources.

Table 11: Complete user stories list, part 1

ID	Epic	Role	User story
US09	Specification	User	As a user, I want to be able to see the tasks in a specification to have a deeper level of abstraction for details. I want to see what tasks are associated with a specification, what resources are associated with the task, and what temporal information is available about the task. By temporal information is meant the average processing time.
US10	Specification	Tactical	As a tactical manager, I want specifications, cases and work-items to have direct links to the respective YAWL admin interface so I can act immediately.
US11	Case	Tactical	As a tactical manager, I want to be able to see all running cases to get a quick overview of the current operational situation.
US12	Case	Tactical	As a tactical manager, I want to see the specification for each case to be able to classify and distinguish between the cases.
US13	Case	Tactical	As a tactical manager, I want to see temporal information for each case to be able to identify issues. This temporal information is the current runtime and the average runtime of those cases. The average runtime is based on historical data.
US14	Case	Tactical	As a tactical manager, I would like to see the average time it takes to reach a certain task in order to answer other resource-related questions.
US15	Case	Tactical	As a tactical manager, I want to be able to set limits on timed case values to quickly see if cases are meeting their service level agreements. These limits can be set for a specifications and tasks.
US16	Case	Tactical	As a tactical manager, I want a soft time limit on cases and tasks so I can be notified and react quickly to avoid overdues. The time limit can base on the average completion time.
US17	Work item	Tactical	As a tactical manager, I want to see the work items of all the cases to get an idea of the processing status of the case.
US18	Work item	Tactical	As a tactical manager, I want to be able to identify bottlenecks to prevent the pile-up of work in time. These are work items that have already been assigned to an employee for too long. This can be realized analogously to the soft limit.
US19	Work item	Tactical	As a tactical manager, I want to see queues of work items that are still waiting for resource allocation to be able to monitor work pile-up. In addition, the time already passed per queue item.

Table 12: Complete user stories list, part 2

ID	Epic	Role	User story
US20	Work item	Tactical	As a tactical manager, I would like to see statistical information on the queue information to evaluate the associated operational performance. This includes the average time until an work item is started and completed. Also, the typical queue size for a typical business day.
US21	Historical	Strategic	As a strategic manager, I want historical information to compare operational performance over time periods that can be individually specified. This information is averaged temporal information to our specifications. Examples include the average, minimum and maximum duration per case.
US22	Historical	Strategic	As a strategic manager, I want to have current operating values displayed in relation to benchmark values in order to have an actual and target comparison with which I can perform controlling.
US23	Historical	Strategic	As a strategic manager, I want to be able to select benchmarks from past performance periods and additionally set my own benchmarks to configure target values myself.
US24	Authentication	User	As a user, I want to be able to log in to access the information that is relevant to me.
US25	Authentication	User	As a user, I must be an administrator to log into the YAWL system. This is to prevent resources from seeing performance information of other resources that are not authorized to do so.
US26	Authentication	User	As a user, I want fine-grained function inclusion and exclusion based on privileges assigned to me so I can focus on my relevant function groups.
US27	Information	User	As a user, I want to be able to configure a custom page for myself, to have a way to bundle all my most relevant information in one place.
US28	Information	User	As a user, I want to have the information presented in a structured way to grasp it quickly. Dashboard information can be visualized in tables or graphs, dials or gauges.

Table 13: Complete user stories list, part 3

ID	Epic	Role	User story
US29	Information	User	As a user, I want all tables to integrate search and filter options to be able to quickly find the information I need.
US30	Information	User	As a user, I want to be able to use filter chains to search efficiently in large tables.
US31	Information	User	As a user, I want to get a general and fine view of work items, cases and specifications to be able to focus only on the detailed information that really interests me. This way I save my cognitive capacities.
US32	Information	User	As a user, I want the separation of tactical and strategic information view to see only those values that are valuable for my role.
US33	Notification	User	As a user, I want to be able to configure individualized notifications to alert me to certain situations. These notifications should be visible on the user interface as well as sent via email.
C01	Integration	-	As a developer of the YAWL system, I want the YAWL dashboard to be automatically delivered in a .war document. This allows integration with the YAWL system.

Table 14: Complete user stories list, part 4

7.3 Complete list of prioritized requirements

ID	Relative advantage	Relative penalty	Total	Relative cost	Relative risk	Priority
US01	5	9	14	1	1	7
US02	5	7	12	2	2	3
US03	9	6	15	5	3	1.87
US04	5	3	8	2	3	1.6
US05	2	1	3	21	5	0.11
US06	3	4	7	2	2	1.75
US07	3	4	7	1	1	3.5
US08	6	4	10	3	3	1.66
US09	5	4	9	21	9	0.3
US10	8	5	13	2	5	1.85
US11	6	7	13	3	2	2.6
US12	4	5	9	1	2	3
US13	9	6	15	3	4	2.14
US14	7	3	10	5	5	1
US15	3	5	8	2	3	1.6
US16	5	2	7	5	1	1.16
US17	5	9	14	5	4	1.55
US18	3	3	6	2	1	2
US19	7	5	12	5	2	1.71
US20	8	2	10	8	2	1
US21	8	2	10	8	3	0.9
US22	7	2	9	4	2	1.5
US23	4	1	5	5	2	0.71
US24	9	9	18	2	1	3
US25	7	6	13	1	1	6.5
US26	4	2	6	13	5	0.33
US27	1	1	2	21	9	0.06
US28	8	9	17	5	2	2.42
US29	7	8	15	3	2	3
US30	5	4	9	3	3	1.5
US31	6	4	10	5	2	1.42
US32	6	4	10	5	2	1.42
US33	3	1	4	21	8	0.19

Table 15: Prioritization of all user stories according to the relative-weight method, based on Gloger, 2011, S. 138, 139