

# The Design and Implementation of a Key Performance Indicator Dashboard for KE-chain

S.J.A. van Bekhoven

M.A. Haisma



# THE DESIGN AND IMPLEMENTATION OF A KEY PERFORMANCE INDICATOR DASHBOARD FOR KE-CHAIN

by

**S.J.A. van Bekhoven**  
**M.A. Haisma**

in partial fulfillment of the requirements for the degree of

**Bachelor of Science**

in Computer Science

at the Delft University of Technology,

to be defended publicly on Tuesday July 1, 2014 at 10:00 AM.

Supervisor:	Dr. C. Hauff,	TU Delft
	Ir. T. Henrich,	KE-works
Thesis committee:	Dr. C. Hauff,	TU Delft
	Dr.ir. FFJ. Hermans,	TU Delft
	Ir. T. Henrich,	KE-works

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

# ACKNOWLEDGEMENTS

We would like to thank our TU Delft coach Claudia Hauff for her critical opinions, which have greatly improved the quality of our thesis. We like to thank our coach at KE-works, Tijn Hendrich, for his feedback on the design and use of the application. We thank Joost Schut for setting up the client interviews at Fokker Aerostructures. A special thanks goes out to all the developers and consultants at KE-works for supplying us with essential feedback during the daily scrums, demonstrations and several questionnaires.

## SUMMARY

KE-works is a six years old company which aims to optimise the product development process in industrial applications. To accomplish this, KE-works deploys a web-application called KE-chain. KE-chain is an engineering workflow management system with the objective to increase the efficiency of the product development process through better control, more efficient distribution, access and use of product-related information. Users have the possibility to set-up a project, manage the tasks belonging to this project, and control the workflow and information distribution. With KE-chain users are able to create structure in the heap of information that composes their product and, when used right, improve the process of their project development.

One of the key elements in optimising the product development process is the monitoring of the available data to give users insight in the status of the project. Currently it is difficult to get a good overview of a project within KE-chain and it is not possible to see what tasks are critical at a certain moment. A common way of showing the status or performance of systems is the use of Key Performance Indicators (KPI's). These indicators, for example in the form of a graph or a table, can quickly give information about the performance of a system. KE-works has decided that it wants to give its users an overview in the form of a project-specific dashboard with KPI widgets. Therefore the assignment is to design and develop an integrated KPI dashboard into KE-chain.

To design the KPI dashboard, which we named KE-board, we shortly researched the field of Performance Measurement to get an overview of the different approaches for the design of KPI's. As a basis for the design we have adopted the Lean methodology [1] which has been used by KE-works in the past. In our research we have actually connected the Lean wastes to measures in KE-chain. To do this, we have chosen a bottom-up approach, which means we started by identifying the available data, after which we extracted several groups of measures. We have interviewed several clients of KE-works, the users of KE-chain. From these interviews we deducted which groups of measures were important for which user roles. To verify which measures are of importance for these dashboards, we have questioned and interviewed the consultants of KE-works. By combining the results of the interviews and the questionnaires we designed 7 KPI widgets.

Finally, we created KE-board and integrated it into KE-chain in five weeks of implementation. After that we have evaluated the complete dashboard by interviewing the consultants of KE-works. On top of that, we have sent them a questionnaire in which they rated the functionality of the widgets to see if they contribute to their purpose and achieve the goals that we set for them. KE-board has been received well by the management and employees of KE-works and according to the extensive evaluation we can state that it definitely contributes to the optimization of the product development process in KE-chain.

# CONTENTS

<b>List of Figures</b>	<b>vi</b>
<b>List of Tables</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 KE-works & KE-chain . . . . .	1
1.2 Problem definition . . . . .	1
1.3 Contributions. . . . .	1
1.4 Outline . . . . .	2
<b>2 Project Methodology</b>	<b>4</b>
2.1 KE-chain . . . . .	4
2.1.1 Application overview . . . . .	4
2.1.2 User roles in KE-chain . . . . .	7
2.2 Project goal & Approach . . . . .	7
2.3 Planning . . . . .	7
2.4 Project Roles . . . . .	8
<b>3 Performance Measures</b>	<b>9</b>
3.1 Lean Methodology . . . . .	9
3.2 Performance Measurement Systems . . . . .	10
3.3 Extraction of Performance Measures . . . . .	10
3.3.1 Available data in KE-chain . . . . .	11
3.3.2 From wastes to measures . . . . .	11
3.4 Conclusion . . . . .	14
<b>4 Key Performance Indicators</b>	<b>15</b>
4.1 Client Interview . . . . .	15
4.1.1 Set-up . . . . .	15
4.1.2 Results. . . . .	16
4.1.3 Conclusion . . . . .	18
4.2 Questionnaire . . . . .	18
4.2.1 Set-up . . . . .	19
4.2.2 Results. . . . .	19
4.2.3 Conclusion . . . . .	21
4.3 Design of KPI's . . . . .	23
4.3.1 KPI's for Engineers . . . . .	23
4.3.2 KPI's for Project Managers . . . . .	24
4.4 Specification of KPI's. . . . .	25
<b>5 Design of KE-board</b>	<b>28</b>
5.1 Requirements. . . . .	28
5.1.1 High-level requirements . . . . .	28
5.1.2 Functional requirements. . . . .	28
5.1.3 Non-functional requirements . . . . .	29
5.2 User experience . . . . .	29

5.3	Use cases . . . . .	31
5.3.1	View dashboard . . . . .	31
5.4	Technical design . . . . .	32
5.4.1	Back-end . . . . .	32
5.4.2	Front-end . . . . .	34
5.5	Testing . . . . .	37
<b>6</b>	<b>Implementation of KE-board</b>	<b>38</b>
6.1	Methods and tools . . . . .	38
6.2	Sprint evaluations . . . . .	39
6.2.1	Sprint 1 . . . . .	39
6.2.2	Sprint 2 . . . . .	40
6.2.3	Sprint 3 . . . . .	40
6.3	Feedback from SIG . . . . .	41
6.4	Test results . . . . .	42
6.5	Meeting the requirements . . . . .	43
6.6	Functionality . . . . .	43
<b>7</b>	<b>Evaluation</b>	<b>48</b>
7.1	Development methodology . . . . .	48
7.2	Planning . . . . .	48
7.3	Product . . . . .	49
7.3.1	Interview . . . . .	49
7.3.2	Questionnaire . . . . .	49
7.3.3	KE-board . . . . .	49
7.3.4	Widgets . . . . .	50
7.3.5	Conclusion . . . . .	53
7.4	Feedback from KE-works . . . . .	54
7.5	Conclusion . . . . .	55
<b>8</b>	<b>Recommendations</b>	<b>56</b>
8.1	KE-board . . . . .	56
8.2	Development . . . . .	57
	<b>Bibliography</b>	<b>58</b>
	<b>Appendices</b>	<b>60</b>
<b>A</b>	<b>KE-chain non-functional requirements</b>	<b>60</b>
<b>B</b>	<b>Follow-up questions</b>	<b>62</b>
B.1	Participant 1 . . . . .	62
B.2	Participant 2 . . . . .	62
B.3	Participant 3 . . . . .	63
<b>C</b>	<b>General Planning per Week</b>	<b>65</b>
<b>D</b>	<b>Use Cases</b>	<b>67</b>
<b>E</b>	<b>Specification of KPI's</b>	<b>69</b>
<b>F</b>	<b>SIG feedback (Dutch)</b>	<b>72</b>
E1	Aanbevelingen . . . . .	72
E2	Hermeting . . . . .	72

<b>G</b>	<b>Original project description</b>	<b>74</b>
<b>H</b>	<b>Widget Evaluation Questionnaire</b>	<b>75</b>
H.1	Hot Tasks Widget . . . . .	75
H.2	Open Tasks Widget . . . . .	77
H.3	Progress Over Time Widget . . . . .	79
H.4	Tasks In Progress Widget. . . . .	81
H.5	Tasks Not Started Widget . . . . .	82
H.6	Costly Changes Widget. . . . .	83

# LIST OF FIGURES

1.1	KE-chain with KE-board opened, user is dragging a widget. . . . .	2
2.1	Simplified Domain Model KE-chain . . . . .	4
2.2	KE-chain home view . . . . .	5
2.3	KE-chain form view . . . . .	6
2.4	KE-chain workflow view . . . . .	6
4.1	Progress over time widget . . . . .	26
5.1	Design of the dashboard with a widget. . . . .	30
5.2	The process of editing the properties of a widget. . . . .	31
5.3	Back end class diagram . . . . .	33
5.4	An example of a portal with a couple of portlets. . . . .	35
5.5	Front end class diagram. . . . .	36
6.1	Number of lines over time . . . . .	39
6.2	Back-end Code Coverage . . . . .	42
6.3	KE-board default engineer dashboard . . . . .	44
6.4	A widget is being dragged to another location . . . . .	44
6.5	Properties of a widget can be opened. . . . .	45
6.6	Properties of a widget can be changed and saved. . . . .	45
6.7	Select a widget from the list and click 'add widget' to add it. . . . .	46
6.8	Clicking the help icon on a widget. . . . .	46
6.9	The help text is being displayed on a widget. . . . .	47
6.10	A widget can be removed by pressing the cross button . . . . .	47
6.11	The removal of a widget need to be confirmed. . . . .	47
7.1	Progress over time widget . . . . .	50
7.2	Hot tasks widget . . . . .	51
7.3	Open tasks widget . . . . .	52
7.4	Tasks not started widget . . . . .	52
7.5	Costly changes widget . . . . .	53
7.6	Tasks in progress widget . . . . .	53



# LIST OF TABLES

2.1	Project Timeline as defined in Plan of Action . . . . .	8
3.1	Measure types according to lean methodology . . . . .	11
3.2	Measures regarding completion of information . . . . .	12
3.3	Measures regarding changes . . . . .	12
3.4	Measures regarding time . . . . .	13
3.5	Measures regarding usage of information . . . . .	13
3.6	Measures regarding criticality . . . . .	13
3.7	Measures regarding the amount of work done . . . . .	14
3.8	Number of potential measures per waste . . . . .	14
4.1	Questionnaire results . . . . .	20
4.2	Follow-up questions conclusions . . . . .	22
4.3	Project progress over time . . . . .	26
5.1	Use case: view dashboard . . . . .	31
7.1	Actual Project Timeline . . . . .	49
7.2	Average score per widget . . . . .	50
D.1	Use case: view dashboard . . . . .	67
D.2	Use case: add KPI widget to dashboard . . . . .	67
D.3	Use case: remove KPI widget from dashboard . . . . .	68
D.4	Use case: edit properties of KPI widget . . . . .	68
D.5	Use case: Drag KPI widgets to arrange dashboard . . . . .	68
E.1	Project progress over time . . . . .	69
E.2	Critical tasks . . . . .	69
E.3	Hot Tasks . . . . .	70
E.4	Open Tasks . . . . .	70
E.5	Tasks in progress . . . . .	70
E.6	Tasks not started . . . . .	71
E.7	Costly changes . . . . .	71

# 1

## INTRODUCTION

### 1.1. KE-WORKS & KE-CHAIN

KE-works is a six years old company which aims to optimise the product development process in industrial applications. To accomplish this, KE-works deploys a web-application called KE-chain. KE-chain is an engineering workflow management system with the objective to increase the efficiency of the operation product development process through better control, more efficient distribution, access and use of product-related information. Users have the possibility to set-up a project, manage the tasks belonging to this project, and control the workflow and information distribution. With KE-chain users are able to create structure in the heap of information that composes their product and, when used right, reduce the lead time<sup>1</sup> of the project.

### 1.2. PROBLEM DEFINITION

One of the key elements in optimising the product development process is the monitoring of available data to give users insight in the status of the project. Currently the information available to users in KE-chain is limited to the *progress* and the *number of changes* of each task and the ways of viewing this information are quite limited. The application works great for projects with a small number of tasks, but when a project consists of tens of tasks the current views are an inefficient way to check on these parameters and it is difficult to get a good overview. Furthermore it is not possible to see how projects and tasks have evolved over time or which tasks are critical at a certain moment. KE-works has noticed that its customers are looking for a better overview and more enhanced ways of analyzing their projects. A common way of showing the status or performance of systems is the use of Key Performance Indicators (KPI's). These indicators, for example in the form of a graph or a table, can quickly give information about the performance of a system and are based on one or more measures. KE-works has decided that it wants to give its users an overview in the form of a project-specific dashboard with KPI widgets. Therefore the assignment is to design and develop an integrated KPI dashboard into KE-chain.

### 1.3. CONTRIBUTIONS

In order to design the KPI dashboard, which we named KE-board, we shortly researched the field of Performance Measurement to get an overview of the different approaches for the design of KPI's.

---

<sup>1</sup>time from the start until the end

As a basis for the design we have adopted the Lean methodology [1] which has been used by KE-works in the past. The Lean methodology is based on reducing waste in a manufacturing process, but can be translated to the development process as well, which has been partly done by KE-works. In our research we have actually connected these wastes to measures in KE-chain. To do this, we have chosen a bottom-up approach, which means we started by identifying the available data, after which we extracted several groups of measures. For all of these groups we defined a broad list of measures which we connected to the corresponding wastes.

To design KE-board, we have interviewed several clients of KE-works, the users of KE-chain. From these interviews we deduced which groups of measures were important for which user roles and concluded that there exists a need for two separate dashboards; one for project managers and one for engineers. To verify which measures are of importance for these dashboards, we have questioned the consultants of KE-works. By combining the results of the interviews and the questionnaire we designed 7 widgets.

Finally, we created KE-board and integrated it into KE-chain in five weeks of implementation. After that we have evaluated the complete dashboard by interviewing the consultants of KE-works. On top of that, we have sent them a questionnaire in which they rated the actual functionality of the widgets to see if they contribute to their purpose and achieve their goals that we set for them. KE-board has been received well by the management and employees of KE-works and according to the extensive evaluation we may state that it definitely contributes to the optimization of the product development process.

Figure 1.1 shows the final product, KE-board: a customizable KPI dashboard for KE-chain.

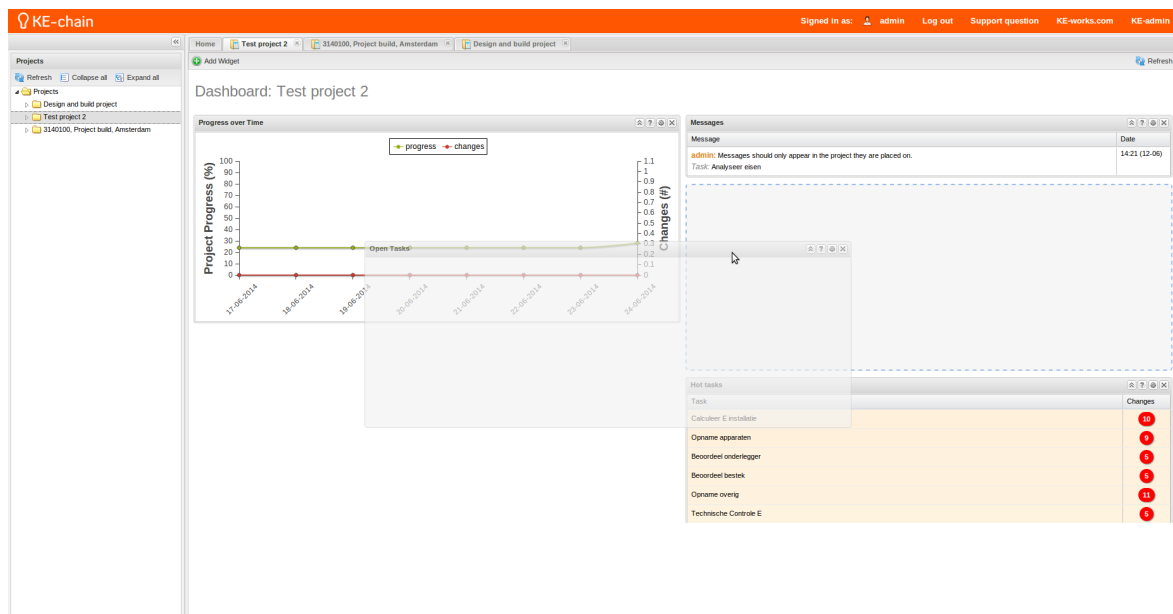


Figure 1.1: KE-chain with KE-board opened, user is dragging a widget.

## 1.4. OUTLINE

In chapter 2 we explain the project in more detail and elaborate on our approach. Then, in chapter 3 we discuss the lean methodology and describe how we extracted the list of measures for the design of the KPI's. Following up on that, in chapter 4 we discuss the interviews and questionnaire from which we conclude the measures to use in the KPI's after which we design the final KPI's. In chapter 5

we create the design of KE-board. In this chapter we go into the requirements, construct use cases and explain the technical design. In chapter 6 we report on our implementation process, such as the problems we encountered during building our application and we present the final product. We discuss KE-board as a whole in chapter 7 by evaluating both the complete dashboard as well as the widgets themselves. Finally in chapter 8 we establish a number of recommendations for KE-board and KE-works in general.

# 2

## PROJECT METHODOLOGY

In this chapter we discuss the project in more detail. We first explain what the KE-chain application is and how it works by giving a simplified example. After that we define the goal and sub-goals of the project according to the assignment, describe our approach and discuss our planning.

### 2.1. KE-CHAIN

First we shortly describe the KE-chain application in more detail. Therefore we discuss the overall data structure and the user interface of KE-chain. After that we take a look at the user roles.

#### 2.1.1. APPLICATION OVERVIEW

In Figure 2.1 you can find a simplified domain model, on which we base further explanation in this section.

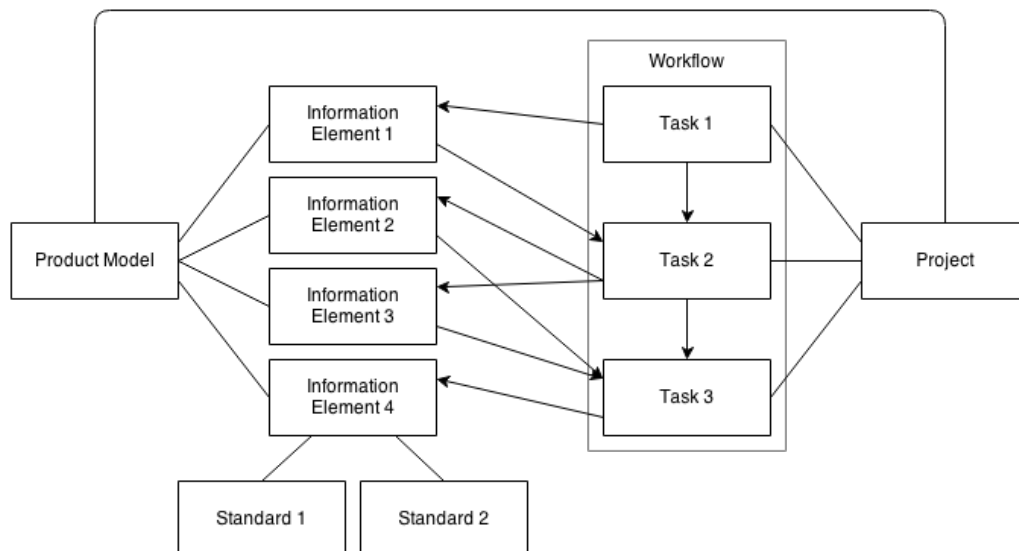


Figure 2.1: Simplified Domain Model KE-chain

KE-chain is an engineering workflow management system based on *projects*. In KE-chain these

projects can be viewed as you can be seen in Figure 2.2 in the left column. In this column there is a tree displayed that holds the project currently active in KE-chain. The large table in the middle of the screen is the view that holds the tasks that are assigned to the user that views it.

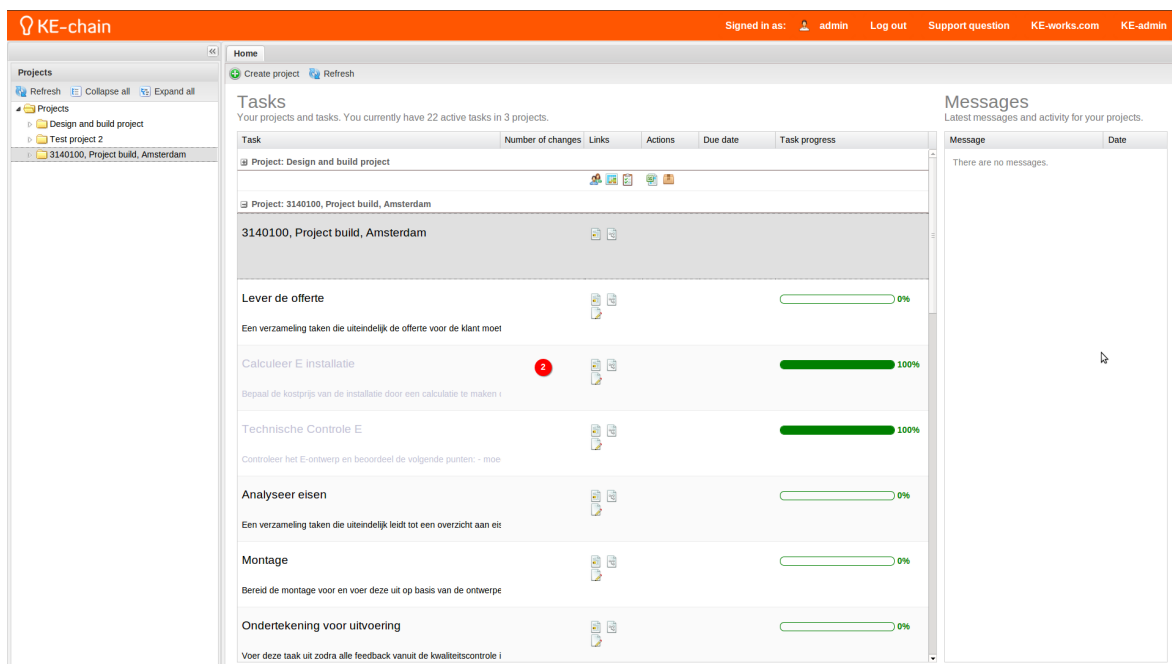


Figure 2.2: KE-chain home view

Each project results in the design of a product once completed. To accomplish this, all projects are based on a *product model*, which contains *information elements* that need to have data entered into them to complete the design of the product. To make the collection of this information manageable, *tasks* can be created which all have some of the information elements as inputs and/or outputs that are put in a form. In KE-chain tasks can be completed as can be seen in Figure 2.3 which shows the form view of a task.

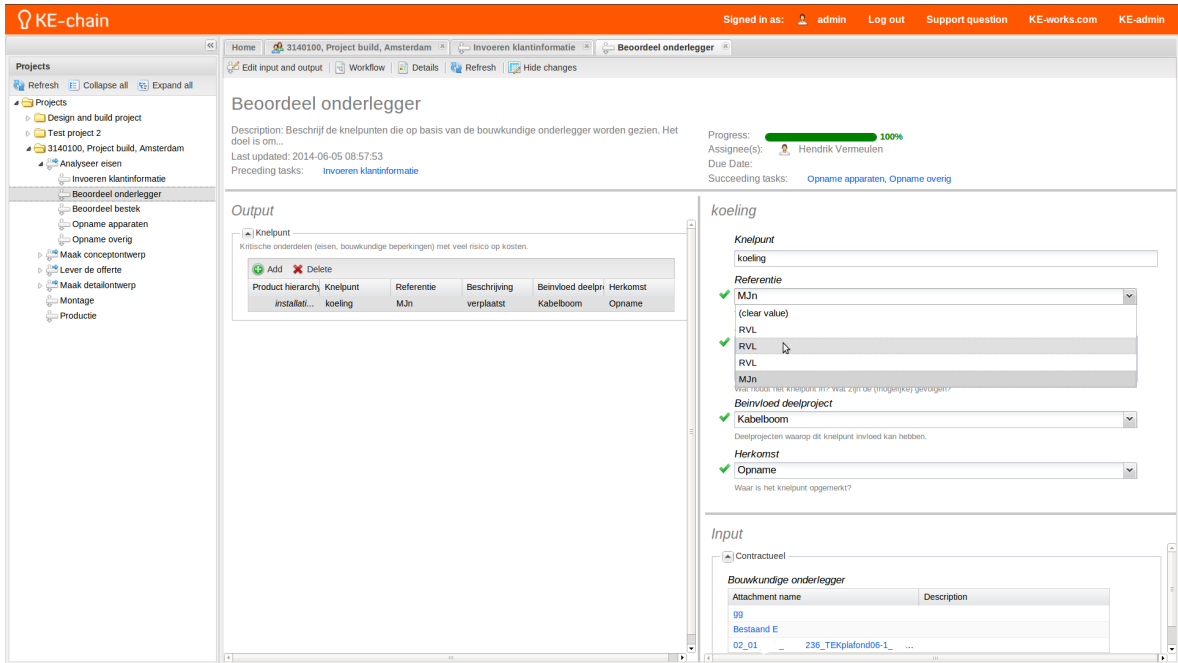


Figure 2.3: KE-chain form view

When defining information elements as output for one task and as input for another task, one creates an information flow. To make sure the tasks are being executed in the right order, they are connected to each other in a *workflow* as can be seen in Figure 2.4.

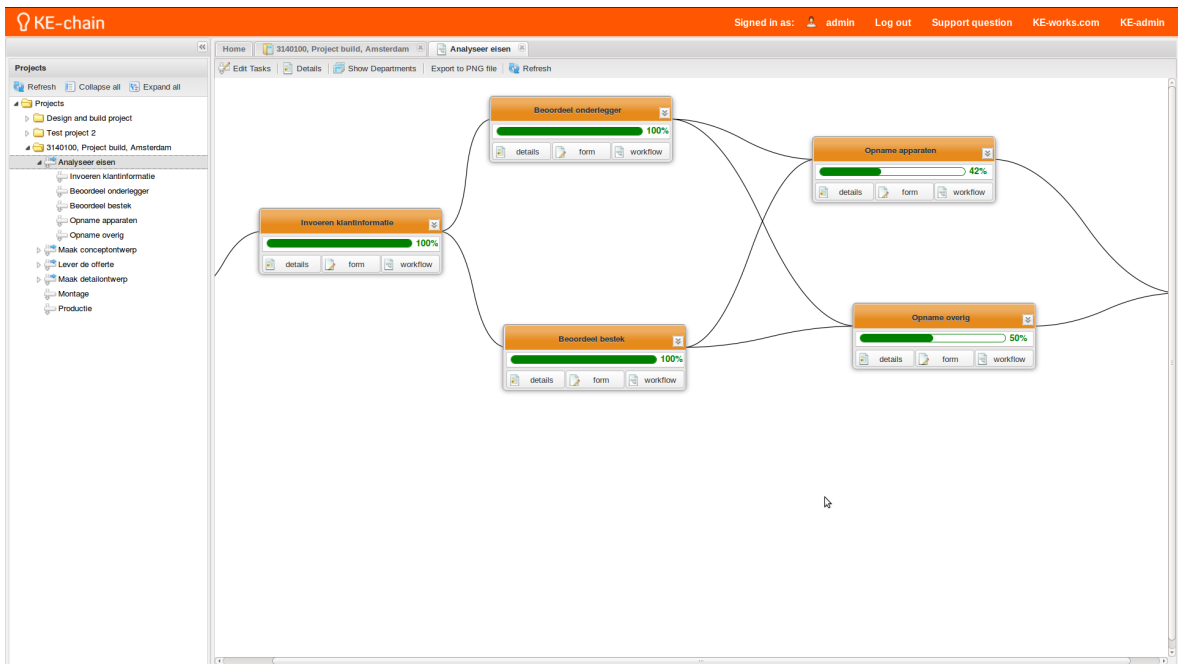


Figure 2.4: KE-chain workflow view

Some of the information elements can be integers, real numbers or dates, but in other cases *standards* are used to make sure the data is consistent, can be used throughout and over multiple projects. An example of this can be seen in Figure 2.3 where a selection is made from a combobox.

### 2.1.2. USER ROLES IN KE-CHAIN

In KE-chain multiple user roles are defined for the different types of people that use the application. For KE-board it is important to look at the different types of users that are defined in KE-chain. There are two user roles that are relevant to our project:

**Project manager** The project managers configures the project in advance, assigns tasks to engineers and keeps the overview.

**Engineer** The engineer is assigned to tasks by the project managers and enters the data in the forms to complete tasks.

## 2.2. PROJECT GOAL & APPROACH

As is described in the previous section the assignment is to design and develop an integrated KPI dashboard into KE-chain. The main goal of the KPI dashboard, from now on called *KE-board*, is to give the users of KE-chain a real-time overview of the status of a project within KE-chain to eventually be able to optimize the product development process. We have divided this goal into the following sub-goals:

- 1. Research related work on performance measurement** Investigate what has been done in the field of performance measurement and see if it can be useful for researching which measures and KPI's to use for KE-chain.
- 2. Extraction of Performance Measures** According to the results of the previous sub-goal decide how to transform the available data in KE-chain to performance measures and perform this transformation.
- 3. Verification of Performance Measures** Investigate according to interviews and questionnaires which measures are of importance for the optimization of the product development process.
- 4. Design of KPI's** Design the KPI's according to the outcome of the previous sub-goal.
- 5. Design of KE-board** Create a design for KE-board in which is defined how KE-board will look, where it will be placed in KE-chain, how it can be used and what it will contain.
- 6. Implementation of KE-board** Development of KE-board while working with the KE-works development team.
- 7. Evaluation of KE-board** Check if the users of KE-chain are satisfied with the functionality of KE-board and find out where possible chances lie to enhance KE-board in the future.

## 2.3. PLANNING

According to the five sub-goals we have split the project up into four phases: research, design, implementation and evaluation. In Table 2.1 you can see that we set up the project with two weeks of research, one week of design, five weeks of implementation and two weeks of evaluation. To get a better a more detailed view of the planning, we have created a general planning per week including deliverables which can be found in Appendix C.



Week #	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10
Research										
Design										
Implementation										
Evaluation										

Table 2.1: Project Timeline as defined in Plan of Action

## 2.4. PROJECT ROLES

The development team for KE-board consists of two team members, namely Sjoerd van Bekhoven and Michiel Haisma. Because of the relatively small team no fixed role distribution was made in advance. However, since Michiel had worked with the JavaScript framework used at KE-chain before, he has done most of the front-end development during the implementation phase. This resulted in the fact that Sjoerd has done most of the back-end development. During the research, design and evaluation phase, work has been equally distributed over both team members.

# 3

## PERFORMANCE MEASURES

Our research goal is to investigate which KPI's (Key Performance Indicators) are useful to indicate the performance of the product design process within KE-chain. In this chapter we take the first step towards the creation of these indicators. Therefore we first investigate which measures are available and useful to give a proper indication. To do so, we discuss the lean methodology which is well-known at KE-works, this methodology focuses on the wastes that emerge within engineering processes. After that we take a short look at related work in Performance Measurement Systems and common ways measures are extracted. Finally we extract the measures that we use based on the existing data in KE-chain and the lean methodology.

### 3.1. LEAN METHODOLOGY

The key initiative of lean engineering is the identification and elimination of waste. The 7 types of waste that can be identified were first described by Ohno [1]. Each of these types were established to identify waste in manufacturing processes, but they can also be applied to the *design process*. The types of waste are: defects, overproduction, inventory, transportation, waiting, motion and over-processing [2].

KE-chain is used for (the optimisation of) engineering product design processes. Since in these processes it is known that waste may occur, the lean methodology can be applied. The aim of our project is to reduce these wastes through monitoring and control. By giving project managers and engineers more knowledge about where and how waste is occurring, we enable them to pinpoint and remove that waste. We now give an overview of the waste types and how they can be identified in the product design process in which KE-chain is active:

**Defects** Defects in the product may occur as a result of wrong or missing information. Defects may be prevented through standardization and the use of complete and correct information.

**Overproduction** When something is produced before it is needed, it is regarded overproduction. Any kind of drawing or documentation that is created before it is needed, is regarded as waste.

**Inventory** Anything that has to be stored for longer than it needs to be, is waste. This type of waste often occurs as a logical consequence of overproduction. However, because of the way digital information can be stored, in design processes this isn't a very large problem.

**Transportation** Paper trails and design approvals that have to be mailed, carried and read by several people is a waste. All kinds of information that have to be transported (transferred) fall under

this category.

**Waiting** If an engineer has to wait for his work to arrive, that is referred to as waiting waste. Often, due to unbalanced workflows, some engineers are done with their portion before other members of the team are. They fill up their time creating inventory or doing something else. Good balanced workflows and efficient processes can reduce the waiting time

**Motion** This form of waste describes all kinds of work that have to be done that deducts from the efficiency of an engineer. This often occurs in software when there have to be a lot of mouse clicks and routines performed. This waste include things like saving a document to a PDF and e-mailing it, instead the entire team should have direct access over that information.

**Over-processing** This form of waste occurs from using too complex software that offers a lot of functionality that is not used. Simplicity is key to lean.

### 3.2. PERFORMANCE MEASUREMENT SYSTEMS

In the last 25 years the research field of performance measurement has been growing because of the ongoing desire for measurement and quantification [3]. Several frameworks have been proposed for the design of a Performance Measurement System (PMS), for example the SMART method [4], the Performance Measurements Matrix [5] and the Balanced Scorecard [6]. The latter is generally seen as the most renowned. It suggests that the main strategy and vision of a company should be translated into a balanced set of measures regarding the financial, internal business, customer and innovation perspective. Combining these four categories gives managers great insight in company inter-relationships which leads to improved decision making and problem solving [6].

There are many more frameworks to be found, but in general the approach is to define the main goal of a company and translate that into a set of measures in some sort of way. In *Towards a corporate performance measurement system* [7] an approach is mentioned in which the main goal is translated to list of business questions which is translated in specific measures while in *Designing a performance measurement system: a case study* [8] a top-down approach is used in which based on the main goal some global measures are defined which are specified more while the level within the company becomes lower. The main goal of KE-chain is to improve the efficiency of engineering workflows. At KE-works the lean methodology is adopted to achieve this goal by reducing the seven wastes. Therefore we use this same methodology to identify the right measures by translating the seven wastes of lean into actual measures in KE-chain.

### 3.3. EXTRACTION OF PERFORMANCE MEASURES

For the extraction of the data we have evaluated two approaches. The first is to start with the general goal of KE-chain, see which measures could be linked to the seven wastes of Lean and check which of these measures could possibly be extracted from the available data in KE-chain. When applying this method we noticed that a lot of measures we wanted to use were not available in KE-chain, which quickly narrowed down the list and was therefore not very satisfying. Therefore we chose another option which is to use a bottom-up approach. This means we start by identifying the available data in KE-chain. From this data we define certain groups of measures which can be used to reduce waste. Finally, we create a list of measures that we consider useful in some way and can be calculated with the available data for every of the groups we defined.

### 3.3.1. AVAILABLE DATA IN KE-CHAIN

Our system will use data that is already being saved in the database. In this section we have a look at what data is already at our disposal. The database includes collections for projects, product models, tasks, information elements and standards as described in Chapter 2.

Currently, most actions performed in KE-chain are being recorded through *archiving*. This means that every time some kind of information is changed, the old version of that information is being kept. In the database, this means that the previous version of a document is transported into the archive and a new document is created and kept as the working copy.

In addition to the archives, there is an activity log. In this log all occurring changes to the task forms are stored. However, because it does not log all activities users perform in KE-Chain. For our purposes, it may be easier to use only the archives. The main reason for this is, is that the archives form a consistent and predictable source of information. The main developer of KE-Chain informed us that the activity log was mainly added for performance purposes, since change calculation can take too long for normal page-loading times. For this reason we may choose to use the activity log if necessary.

### 3.3.2. FROM WASTES TO MEASURES

To create a list of possible measures we look back at the lean methodology. We match the seven wastes with aspects of KE-chain to come to several groups of measures as summarized in Table 3.1.

Waste type	Measure group
Defects	Completion of information Changes per instance
Overproduction	Usage of information
Inventory	Criticality of instances
Transportation	Time Usage of information
Waiting	Time
Motion	Work performed
Overprocessing	Usage of information

Table 3.1: Measure types according to lean methodology

The *defects* in KE-chain can be seen as missing or wrong information or as a change that is made after the task has been completed, so that succeeding tasks need to be revised as well. A form of *overproduction* is working on a task before it is necessary to do so. The information of this task will be worthless until the produced information is actually needed. *Inventory* in KE-chain can be mapped to completed tasks that are piling up. It can therefore be useful to see which tasks are critical to do to improve the process. Improving *transportation* is difficult, because we can not measure whether users have to convert files or have to enter the same information multiple times. We could measure the amount of messages that people are sending towards each other within a project or extract data from the usage of information. *Waiting* time or time in general is one of the most important wastes to reduce and can be measured efficiently. *Motion* is difficult to map and measure within KE-chain, but could be seen as the number of clicks a user has performed within a task or project. *Over-processing* could be seen as the modification of information that is not necessary, which will in reality be difficult to measure, but could be matched to the usage of information.

In the rest of this section we briefly describe each measure group and make a list of the possible

measures that could be used.

## COMPLETION

Each task in KE-chain consists of a form with multiple items or fields which contain information. Measuring the extent to which a task or project is completed can be useful. Possible measures are summarized in Table 3.2.

#	Measure	Lean waste type
1.1	Completion of task	Defects
1.2	Completion of task over time	Defects/waiting
1.3	Completion of project	Defects
1.4	Completion of project per department	Defects
1.5	Completion of project over time	Defects/waiting
1.6	Number of tasks in progress	Defects
1.7	Number of tasks in progress over time	Defects/waiting

Table 3.2: Measures regarding completion of information

## CHANGES

When information in a certain task is changed, this may influence other tasks. A cost vector may be calculated by taking into account the completion of tasks succeeding the task the change occurred in. If information stays unchanged until the end of the project, we can conclude the information was correct the first time. Possible measures are summarized in Table 3.3.

#	Measure	Lean waste type
2.1	Number of changes per task	Defects
2.2	Number of changes per field per task	Defects
2.3	Number of changes per task over time	Defects/waiting
2.4	Number of changes of project	Defects
2.5	Number of changes of project over time	Defects/waiting
2.6	Relative number of changes per task wrt other tasks	Defects
2.7	Average effect of changes on other tasks per task	Defects
2.8	Average effect of changes on other tasks per department	Defects
2.9	Changes that have the highest cost, $\text{change} \cdot \text{cost vector}$	Defects/overprocessing
2.10	Number of unchanged fields per task	Defects
2.11	Number of unchanged fields per project	Defects

Table 3.3: Measures regarding changes

## TIME

Measuring time when completing tasks or project can be useful in many cases. For these measures we define the following definitions: *lead time* is the time it takes to complete an instance from beginning to end, *waiting time* is the time an instance has to wait for predecessors to complete and *working time* is the time users have been actually working on an instance. Furthermore time is used in combination with a lot of other measures to see how they developed over time. Possible measures are summarized in Table 3.4.

#	Measure	Lean waste type
3.1	Lead time per task	Waiting
3.2	Waiting time per task	Waiting/transportation
3.3	Working time per task	Waiting
3.4	Lead time of project	Waiting
3.5	Waiting time of project	Waiting/transportation
3.6	Working time of project	Waiting
3.7	Average and SD of lead time of tasks	Waiting
3.8	Average and SD of waiting time of tasks	Waiting
3.9	Average and SD of working time of tasks	Waiting
3.10	Earliness of information	Overproduction

Table 3.4: Measures regarding time

### USE OF INFORMATION

When more frequent standards are used, less energy is wasted creating new standards or using custom values, decreasing motion waste. Using standards also helps to reduce defects. However, standards that are created but never used, contribute to inventory and overproduction wastes. Possible measures are summarized in Table 3.5.

#	Measure	Lean waste type
4.1	Number of non-standards introduced per project	Motion/defects
4.2	Relative usage of standards per project	Motion
4.3	Relative usage of standards per task	Motion
4.4	Standards most used per task	Defects
4.5	Standards most used per project	Defects
4.6	Standards most used over all projects	Defects
4.7	Standards most used per user	Defects
4.8	Standards never used	Inventory
4.9	Relative usage of standards per user	Defects/motion
4.10	Form size per task	Inventory

Table 3.5: Measures regarding usage of information

### CRITICALITY

When users see KE-board they should be able to see which task they should start working on in a glance. Therefore it is necessary to define which tasks are urgent. Possible measures are summarized in Table 3.6.

#	Measure	Lean waste type
5.1	Tasks with most succeeding tasks	Waiting
5.2	Succeeding tasks that have not been started yet	Inventory
5.3	Tasks that are due now/this week	Waiting
5.4	What information is most/least used	Overproduction/overprocessing

Table 3.6: Measures regarding criticality

### WORK PERFORMED

If tasks can be performed with less expansion of energy, that would be very useful to know. Possible measures are summarized in Table 3.7.

#	Measure	Lean waste type
6.1	Number of clicks per task	Motion
6.2	Number of clicks per user per project	Motion
6.3	Number of clicks per user relative to changes	Motion
6.4	Number of clicks per task relative to changes	Motion
6.5	Number of messages send per task	Transportation
6.6	Number of messages send per project	Transportation

Table 3.7: Measures regarding the amount of work done

### 3.4. CONCLUSION

From the existing data in KE-chain we have extracted 48 measures of which we think they can be considered somewhat useful for the reduction of the corresponding waste. In table 3.8 we show the seven wastes and the number of corresponding potential measures.

Waste	Number of measures
Defects	22
Overproduction	2
Inventory	3
Transportation	4
Waiting	16
Motion	8
Over-processing	2

Table 3.8: Number of potential measures per waste

We may safely state that the wastes defects and waiting are covered well with the current measures. Furthermore we concluded in the previous section that transportation, overproduction and overprocessing might be measured by checking the usage of information. However, in KE-chain we can only see if an information element that is output of one task is used as input for another task. Unfortunately it is not possible to check if the data is actually used to create new output. Therefore it is difficult to create measures for these wastes. Inventory may be covered by creating measures that show criticality. However, it is not possible to indicate the criticality of for example tasks by one measure, but this can be done by combining multiple measures. In the next chapter, criticality will be contained in one of the KPI's and therefore we expect the waste inventory to be reduced when KE-board is implemented. In KE-chain the only motion that could be directly measured is the number of clicks. Therefore the number of measures we have derived from this type of waste is quite limited.

From the data that is currently available in KE-chain the list is quite exhaustive. In the evaluation we found that more static measures like the number of open tasks and the number of tasks that is not assigned to a user could be helpful measures as well although they do not always correspond to a certain type of waste. Overall the list of measures we have created is a good starting point for defining the KPI's as is discussed in the next chapter. There we rate and discuss these measures with expert users of KE-chain (managers and engineers) to find out which measures should be included in the KPI's for KE-board.

# 4

## KEY PERFORMANCE INDICATORS

In the previous chapter we have identified numerous measures which we consider to be useful in some way to measure the performance of the product design process. To get a better understanding of which measures can be of value to measure product development process performance, we consult the expert users of KE-chain. We have interviewed a selection of clients of KE-works who use KE-chain on a daily basis, and have sent a questionnaire to the consultants at KE-works. In this chapter the results of both the interview and the questionnaire are discussed after which we conclude which KPI's will be implemented in the dashboard.

### 4.1. CLIENT INTERVIEW

In chapter 2 we have seen that there are two types of users that are going to use KE-board: Engineers and project managers. To get a good view of what a project manager and an engineer want to see on their KE-board, we have interviewed a selection we think gives a good reflection of the users of KE-chain:

1. A low-level process manager (Process Specialist Production Planning) at Fokker Aerostructures.
2. A high-level process manager (Senior Program Manager) at Fokker Aerostructures.
3. A consultant at KE-works who has worked with the process managers at Kersten Retail.

With this selection of users we have both covered long-term (Fokker Aerostructures) and short-term (Kersten) projects. Furthermore we have seen management at a high and low level. We have not interviewed an engineer, but since the low-level process manager and the consultant from KE-works are in direct contact with the engineers, we expect them to have enough knowledge to give us a proper understanding of the engineers needs.

#### 4.1.1. SET-UP

The main goal of the interviews is to get a clear view of the measures that are used to monitor process performance, what measures could be of used in the future and how KE-board could contribute to that. Therefore we have created a number of questions as a guidance during the interviews. To give an idea, this is a selection of the questions we have asked:

- How do you use KE-chain within your function? What are the elements you most use?



- What measures do you use at this time to monitor process performance?
- What measures are most valuable for your job?
- Are you able to easily identify bottlenecks?
- Are you able to easily identify if a project is on schedule?
- Do you often have to look for the right information or changes?
- What are measures you would like to control but can not right now?
- Are there other problems or inconveniences that could be improved with KE-board?
- Do you think KE-board could be of value for you?
- What do you think of the following measures?

The interviews were held by the project team (Michiel Haisma en Sjoerd van Bekhoven) and took around 1 hour each.

#### 4.1.2. RESULTS

We have recorded all interviews and created transcriptions. The transcriptions have not been included in this report because of privacy reasons. We discuss the general conclusions of the interviews in this section.

##### INTERVIEW 1: LOW-LEVEL PROCESS MANAGER LONG-TERM PROJECTS

The interviewee is a process manager at production preparation and has the lead of 45 production preparation employees and 15 product preparation managers. The interviewee has not used KE-chain intensively until now, but will be using it for an upcoming project. However, the interviewee is able to show us some of the KPI's he uses to monitor the performance of the processes at his department. The KPI's are collected from a diversity of software applications once a week and merged into one Excel-file. From there, the interviewee extracts some graphs and tables to get insight in the data. The measures that the interviewee is using right now or wants to use are:

- Time tasks are open, categorised per three months. When a task is open for more than six months, our interviewee starts asking questions.
- Changes; what changes emerge, how often do changes emerge within tasks. Furthermore the interviewee would like to be able to categorize changes to zoom in to the cause of the change.
- The use of standards; it is costly if some standards are only used very little while they could be replaced with others. This is not possible right now, but with the use of KE-chain it will be. It would be very helpful if that information could be monitored with KE-board.
- Maturity of data; give all data a status as "draft" or "released". This is not available in KE-chain at this moment, but could be very helpful.

Conclusions are that the interviewee wants to see KPI's about tasks that are open, number and causes of changes, maturity of data and the usage of standards. Furthermore when changes happen the KPI or dashboard should call for action.

## INTERVIEW 2: HIGH-LEVEL PROCESS MANAGER LONG-TERM PROJECTS

The interviewee is process manager at the level of timeline, scope, budget and risk management. His department trains and controls project managers at lower levels. They use KE-chain as a scoping tool and not as a workflow management tool that other clients use it as. However, the interviewee has a lot of experience and therefore we decided to see what information we could extract from the KPI's he works with. The measures pointed out by the interviewee are:

- Cost Performance Indicator (CPI) and the Schedule Performance Indicator (SPI). In these two global indicators nearly everything can be noticed, since they represent cost and time. For example, the number of hours that is worked (working time) can be noticed in the CPI, the number of hours that a project takes (lead time) can be noticed in the SPI and costly changes can be noticed in both.
- Float, the number of days that the projects predicted end date differs from the initially set end date.
- Progress of tasks within projects over time.
- Critical path analysis; in what order are tasks completed, is there any unused information.

Conclusions are that time and cost based KPI's are most important on high level. Occurring changes and standard usage are too low level. The results of the low level KPI's should be only represented in time and cost.

## INTERVIEW 3: PROCESS MANAGER SHORT-TERM PROJECTS

The interviewee is a consultant at KE-works who has set-up short-term projects and has worked a lot with the interviewees of KE-chain. He advises us to keep the scope within the project and focus on KPI's that show data at the current moment, not in retrospect. Therefore he thinks that all KPI's with some property over time are of less value, because you mainly look at what has happened and not at what is happening right now. Important measures as pointed out by the interviewee are:

- Reliability of data; if interviewees enter data when no input data has been generated yet, that data is probably worth less or at least based on assumptions.
- Task progress; not only look at the output of a task, but also give insight of the progress on the input of a task.
- Changes; the number of changes is interesting, but input and output changes should be split. Also: what is the effect of these changes?
- Waiting times; distinguish the waiting of information and the waiting of people, that is something different.
- Standards; looking at the usage of standards is very important for process optimization.
- Criticality; do not only show which tasks are critical, but also why they are critical.

Conclusions are that the most important KPI's are the one according to usage standards, impact of changes and progress per task, not per project. Furthermore input and output completion and changes should be split and the focus should lie within the project in the current moment.

### 4.1.3. CONCLUSION

In the interviews we have seen three different perspectives on managing engineering projects. In the interview with the low-level manager the interviewee mentioned that he did not really see much difference between the dashboards for an engineer and a project manager. However, in the interview with the high-level project manager we noticed a serious difference in needs. The low-level manager not only wanted monitoring, but also wanted the KPI's to call for action. The high-level manager on the other hand just wanted to see trends and high-level KPI's such as cost and time. Therefore we have assumed that the low-level manager can be matched with the engineering role in KE-chain and the high-level manager can be matched with the project manager role in KE-chain. Based on this we have chosen to implement a separate dashboard for the engineer and the project manager, of which the most important groups of measures are summarized in this section.

**Engineer Dashboard** From the first and third interview we have derived three measures that we consider the most important for the dashboard for *Engineers*:

- Progress / time: how long have tasks been open, progress of tasks, separate input & output.
- Changes: where have they emerged, where should I look for outdated data?
- Criticality: what task to start on, why is it critical, due dates

**Project Manager Dashboard** We may conclude that the most important measures for a high-level project manager are time and cost. However, in KE-chain it is not possible to monitor cost, so the main focus lies on time. Therefore we have selected the following measures for the dashboard for *Project Managers*:

- Progress / time: progress of project, progress of tasks, due dates, are we on track?
- Usage of standards: which are most and least used, non-standards introduced.
- Time: waiting and lead times, where do the bottlenecks lie?
- Changes: where do they occur, which changes are costly for the project.

**Missing** We faced two parameters that according to the interviews should be in KE-board, but are not available in KE-chain:

- Cost: cost overview of project, per task, per department.
- Maturity: how reliable is data, when did it achieve a certain maturity status such as 'released'.

Finally there is the need for a monitoring level above multiple projects. Especially when we look at bottlenecks and usage of standards it is very useful to compare notable events with other projects to see if they are systematical. However, since KE-board is scoped to be within projects, we do not look at this possibility.

## 4.2. QUESTIONNAIRE

Now that we know which information users find important on their dashboard, we need to find out which specific measures are useful to measure performance. To get a good view of the quality and usefulness of the measures mentioned in Section 3.3.2 we have questioned consultants / employees

at KE-works with a questionnaire. In this section we discuss the set-up and results of this questionnaire.

#### 4.2.1. SET-UP

We created a questionnaire in which for each measure and its corresponding waste, two questions were to be answered:

1. How would you grade the usefulness of this measure?
2. How would you grade the usefulness of this measure in achieving reduction of the corresponding waste?

The first question could be answered with: “Not useful at all”, “A little bit useful”, “Sometimes useful”, “Very useful” and “Critical to have”. The second question could be answered with: “Not contributing”, “Barely contributing”, “Contributing”, “Contributing heavily” and “Critical”. The answers correspond to the values 1 to 5.

The questionnaire was sent to the management and consultants at KE-works (8 in total) and was responded by the CCO (Chief Commercial Officer) and two consultants. Although the number of respondents is low, this selection seems to be fairly good, since we now have a customer-based and commercial approach.

#### 4.2.2. RESULTS

The answers to the first question have been summarized in table 4.1 for all measures. In this table we find each measure and the number of times it has been scored 1, 2, 3, 4 or 5, followed by the mean of the answers. Since the number of respondents is rather low, it is difficult to draw conclusions about the mean if it is not higher than 4 or lower than 2. Therefore we have looked at significant differences, for example if the respondents all had different widely varying answers or if one respondent had a completely different answer than the other two. According to these differences, we have asked follow-up questions to the respondents to get a better view of their opinion. The results of the follow-up questions can be found in appendix B. The conclusions of the questionnaire are presented in the next section.

#	Measure	1	2	3	4	5	$\mu$
1.1	Completion of task	0	0	0	1	2	4.67
1.2	Completion of task over time	0	0	0	2	1	4.33
1.3	Completion of project	0	1	1	0	1	3.33
1.4	Completion of project per department	0	1	1	1	0	3.00
1.5	Completion of project over time	0	0	1	2	0	3.67
1.6	Number of tasks in progress	0	1	0	1	1	3.67
1.7	Number of tasks in progress over time	0	1	0	2	0	3.33
2.1	Number of changes per task	0	0	0	2	1	4.33
2.2	Number of changes per field per task	0	0	2	1	0	3.33
2.3	Number of changes per task over time	0	0	1	1	1	4.00
2.4	Number of changes of project	0	1	0	2	0	3.33
2.5	Number of changes of project over time	0	0	1	2	0	3.67
2.6	Relative number of changes per task wrt other tasks	0	2	0	0	1	3.00
2.7	Average effect of changes on other tasks per task	0	0	1	1	1	4.00
2.8	Average effect of changes on other tasks per department	0	0	3	0	0	3.00
2.9	Changes that have the highest cost, change*cost vector	0	0	0	1	2	4.67
2.10	Number of unchanged fields per task	0	1	1	0	1	3.33
2.11	Number of unchanged fields per project	0	1	1	1	0	3.00
3.1	Lead time per task	0	0	0	1	2	4.67
3.2	Waiting time per task	0	0	0	1	2	4.67
3.3	Working time per task	0	1	0	1	1	3.67
3.4	Lead time of project	0	1	1	0	1	3.33
3.5	Waiting time of project	0	0	2	1	0	3.33
3.6	Working time of project	0	0	2	1	0	3.33
3.7	Average and SD of lead time of tasks	0	0	1	2	0	3.67
3.8	Average and SD of waiting time of tasks	0	0	1	2	0	3.67
3.9	Average and SD of working time of tasks	0	0	1	2	0	3.67
3.10	Earliness of information	0	0	1	2	0	3.67
4.1	Number of non-standards introduced per project	0	0	0	2	1	4.33
4.2	Relative usage of standards per project	0	0	0	2	1	4.33
4.3	Relative usage of standards per task	0	1	1	1	0	3.00
4.4	Standards most used per task	0	1	0	1	1	3.67
4.5	Standards most used per project	0	0	0	2	1	4.33
4.6	Standards most used over all projects	0	0	0	1	2	4.67
4.7	Standards most used per user	0	0	2	0	1	3.67
4.8	Standards never used	0	0	0	2	1	4.33
4.9	Relative usage of standards per user	0	1	1	1	0	3.00
4.10	Form size per task	0	2	0	1	0	2.67
5.1	Tasks with most succeeding tasks	0	1	1	0	1	3.33
5.2	Succeeding tasks that have not been started yet	0	1	1	1	0	3.00
5.3	Tasks that are due now/this week	0	0	0	0	3	5.00
5.4	What information is most/least used	0	0	1	1	1	4.00
6.1	Number of clicks per task	1	0	0	1	1	3.33
6.2	Number of clicks per user per project	1	0	0	2	0	3.00
6.3	Number of clicks per user relative to changes	1	0	0	2	0	3.00
6.4	Number of clicks per task relative to changes	1	0	0	1	1	3.33
6.5	Number of messages send per task	1	0	2	0	0	2.33
6.6	Number of messages send per project	1	0	2	0	0	2.33

Table 4.1: Questionnaire results

### 4.2.3. CONCLUSION

From the results of the interviews we have concluded which groups of measures are important for the engineer and project manager dashboard. Now we will conclude which precise measures are of importance for KE-board according to the results of the questionnaire.

First of all there are 13 measures which all three participants have graded as (very) important and thus have a mean higher or equal than 4. These measures will definitely be included in KE-board:

- Completion of task (4.67)
- Completion of task over time (4.33)
- Number of changes per task (4.33)
- Number of changes per task over time (4.00)
- Average effect of changes on other tasks per task (4.00)
- Changes that have the highest cost vector (4.67)
- Lead time per task (4.67)
- Waiting time per task (4.67)
- Number of non-standards introduced per project (4.33)
- Relative usage of standards per project (4.33)
- Standards most used per project (4.33)
- Standards most used over all projects (4.67)
- Standards never used (4.33)

According to the follow-up questions in appendix B we have decided to include 8 other measures. Our considerations about the included and not included widgets are summarized in table 4.2. The following measures will be included in KE-board together with the measures mentioned before:

- Completion of project (3.33)
- Completion of project over time (3.67)
- Lead time of project (3.33)
- Working time per task (3.67)
- Number of changes per project over time (3.33)
- Number of changes per project (3.67)
- Tasks with most succeeding tasks (3.33)
- Number of unchanged fields per task (3.33)

<b>Measure</b>	$\mu_{score}$	<b>Arguments</b>	<b>Included?</b>
Completion of project	3.33	According to the answers of participant 1 and 3 this should be included since it gives a general idea of the progress of the project.	X
Completion of project over time	3.67	Same as previous.	X
Lead time of project	3.33	How long does my project take and how long are we busy right now seems to be an important question according to participant 1.	X
Working time per task	3.67	According to the answer of participant 3 this could be of importance to see if a task needs more resources.	X
Number of changes per project over time	3.33	According to the answer of participant 3 this could be of importance to see if a task needs more resources.	X
Number of changes per project	3.67	Same as previous.	X
Tasks with most succeeding tasks	3.33	The assumption of participant 2 that KE-chain does this is true, but only static, not dynamic. Participant 3 thinks it is of value.	X
Number of unchanged fields per task	3.33	Participant 1 thinks this is not important because it gives no actual information. However, participant 2 thinks that it could be interesting for project managers.	X
Number of tasks in progress over time	3.33	Participant 1 mentions that there are always be much tasks in progress which seems right.	
Number of tasks in progress	3.67	Same as previous	
All measures regarding to number of clicks	3.00	According to participant 1 and 3 this could be of value when you could read effort/mutations. However, we think the number of clicks does not represent effort accurately.	
Completion of project per department	3.00	Participant 1 sees this as a valuable measure, but according to the interviews we have decided not to measure this since it is not really used by the companies.	
Standards most used per task	3.67	Only useful over multiple projects while our scope is within projects.	
Standards most used per user	3.67	However participant 2 thinks this is of value, we think this measure is too detailed and does not provide overview.	

Table 4.2: Follow-up questions conclusions

Now that we know which groups of measures are considered important for the user-specific dashboards and know which measures should be included in KE-board, we have come to the final step: the design of the Key Performance Indicators.

### 4.3. DESIGN OF KPI'S

In this section we come to the final design of the KPI's by combining the results from both the interviews as well as the questionnaire. In general we have noticed that the clients as well as the consultants seem to be looking for more than just monitoring. They do not only want to see how a project is going, they also want to know immediately what they should do to make it better. Therefore, for both dashboards, we have to search for a good mix of KPI's which not only give the possibility to monitor the project, but also shows users what to do (for engineers) or what to make people do (for managers).

#### 4.3.1. KPI'S FOR ENGINEERS

From the interviews we found that the most important measures for the dashboard for Engineers are mainly progress, changes and criticality. If we combine this information with the outcomes of the questionnaire, we are able to filter out the following measures for the Engineer dashboard:

1. Completion of task (input)
2. Completion of task (output)
3. Number of changes per task
4. Lead time per task
5. Waiting time per task
6. Changes that have the highest cost vector
7. Completion of project (over time)
8. Number of changes per project (over time)
9. Tasks with most succeeding tasks

We now combine these measures in four KPI's. The first one is the *Progress Over Time* KPI, which gives an overview of the progress of the project regarding completion and changes. To create a call for action, we have chosen to defined the other KPI's regarding tasks. It is not possible to give all information about tasks in one list or grid, so therefore we have split them up in the *Critical Tasks*, *Hot Tasks* and the *Open Tasks* KPI. The Critical Tasks KPI shows the tasks that have not been completed, have a large lead time and have a lot of succeeding tasks. The Hot Tasks KPI shows the tasks with the highest amount of changes combined with the impact of these changes. Finally the Open Tasks KPI shows tasks that have been started and have high input and output progress values and therefore need to be finished. These KPI's give a solid overview of all tasks that need attention and since the tasks in the lists are clickable, the KPI's include the possibility for immediate action. The following measures are used per KPI:

- Project Over Time
  - Completion of project (over time)
  - Number of changes per project (over time)
- Critical Tasks
  - Completion of task (output)
  - Lead time per task



- Tasks with most succeeding tasks
- Hot Tasks
  - Number of changes per task
  - Changes that have the highest cost vector
- Open Tasks
  - Completion of task (input)
  - Completion of task (output)

#### 4.3.2. KPI'S FOR PROJECT MANAGERS

When looking at the interviews we have concluded that the measures should mainly be about project progress, due dates, usage of standards, waiting times and changes. The usage of standards is a very interesting measure, but is especially interesting at above-project level when comparing multiple projects. It does not really give actual project status information and is more interesting for analysis after the project has been completed to learn from for new similar projects. Therefore we have chosen not to take these measures into account and focus on actual status information. The call for action is less important for Project Managers, but we think it is still valuable to combine the project status with a call to action. Combined with the information from the questionnaire, the following measures have been filtered for the Project Managers dashboard:

- Completion of task
- Number of changes per task
- Number of changes per task over time
- Changes that have the highest cost vector
- Waiting time per task
- Working time per task
- Lead time per task
- Completion of project (over time)
- Number of changes per project (over time)
- Tasks with most succeeding tasks

To summarize these measures into KPI's we have tried to find a good mix between monitoring and action. In the dashboard for Engineers we have defined the *Project Over Time* KPI that is very suitable for project managers as well. For the mix between monitoring and action we have defined two KPI's regarding tasks which are the *Tasks in Progress* and *Tasks not Started* KPI's. The *Tasks in Progress* KPI gives an overview of all tasks that are currently in progress including information about their lead time, working time and remaining time left before their due date. The *Tasks not Started* KPI gives an indication of tasks that could possibly be worked on, but are not active at this moment. The last KPI is *Costly Changes*, which show an overview of the actual changes that have emerged focusing on the costly ones. The following measures are used per KPI:

- Project Over Time
  - Completion of project (over time)

- Number of changes per project (over time)
- Tasks in Progress
  - Completion of task (output)
  - Lead time per task
  - Working time per task
  - Remaining time per task
- Tasks not Started
  - Waiting time per task
  - Completion of task (input)
  - Completion of task (output)
- Costly Changes
  - Changes that have the highest cost vector
  - Tasks with most succeeding tasks
  - Number of changes per task

#### 4.4. SPECIFICATION OF KPI'S

Now that we have identified which KPI's to implement, we have to specify their exact functionality and purpose. How to do this is one of the questions that has been studied in the field of performance measurement systems. Neely et al. have been handling this and concluded that just specifying a certain formula is not all there is to do. They propose a performance measure record sheet for each measurement to be able to design and test the measures more easily. These sheets are specified according to ten elements: title, purpose, relates to, target, formula, frequency, who measures, source of data, who acts on this data and what to they do [9]. Since this method is easy to adopt and clarifies the chosen KPI's, we adjust it for KE-chain and use it. In KE-chain the parts *who measures* and *source of data* are obviously superfluous since the data is contained in KE-chain already. Because we also want to clarify which way of visualization we want to use for the implementation phase, we add the *visualization method*. So to specify the KPI's, we focus on the following nine points, which we shortly specify:

**Title** What does the measure contain? The title of the measure should be clear and self-explanatory so that a user immediately knows what the measure is about.

**Purpose** Why is this measure necessary? What can a user of KE-chain do with or read from it?

**Relates to** To which of the objectives of KE-chain does this measure relate? In this case: which kind of waste could it reduce?

**Target** This is the one property that we think should be adapted for each company using KE-chain. It should state a certain goal in the form of *ten percent shorter total production time at the end of the year*, which is different for each company.

**Formula** How will the measure be calculated? The most challenging element which ultimately calculates the value of the measure.

**Refresh rate** How often should this KPI be measured?

**Who acts on the data** For which kind of user is this measure meant in KE-chain? Project manager or engineer?

**What do they do** One of the most important properties, since the only way to improve the engineering process is to act on the outcomes of the indicators. At KE-works this is mainly the task of the consultants which supervise the projects and therefore we will not be researching this property thoroughly. However, we give some advise towards what we think would be possible to do with the information gathered by each measure.

**Visualization method** How will the measure be shown to the user?

In the following table we specify a KPI's as mentioned in section 4.3. For the complete list of KPI specifications, please refer to appendix E.

<b>Title</b>	Project progress over time
<b>Purpose</b>	Inform of current state of project and amount of changes being introduced in relation to the completion.
<b>Role</b>	Engineer
<b>Relates to</b>	1.1, 1.2, 1.3, 1.4, 1.5, 2.1, 2.3, 2.4, 2.5
<b>Target</b>	Improve project development.
<b>User-defined properties</b>	[unit] Day, week, month
<b>Formula</b>	Calculate completion of project per [unit], calculate number of changes per [unit] and show it over time from project start date until now
<b>Refresh rate</b>	Half hour
<b>Who acts on the data</b>	Project Manager or Engineer
<b>What do they do</b>	See if the project is on track and look for problem sources if problems have occurred.
<b>Visualization method</b>	Two line graphs

Table 4.3: Project progress over time

The above specification has resulted in the widget found in figure 4.1.

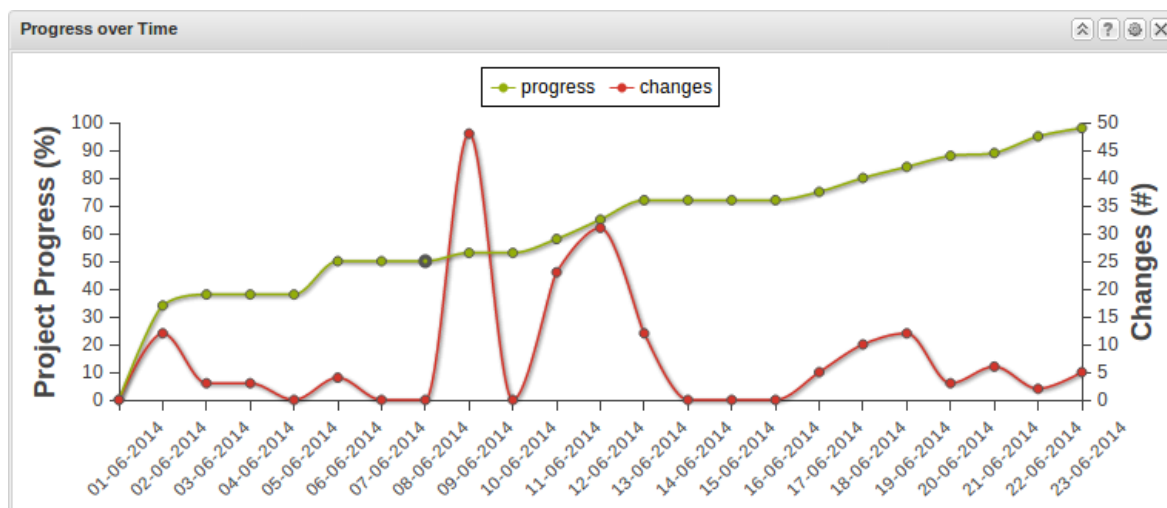


Figure 4.1: Progress over time widget

That concludes the definitions of the KPI's we want to bring to the users. In the next chapter, we focus on the design of KE-board. This includes the requirements, user experience and the technical details.

# 5

## DESIGN OF KE-BOARD

In this chapter we discuss the design of KE-board. First, we specify the requirements that KE-board has to meet, followed by the general user experience. After that, we further specify the user experience in the form of use cases. Finally we explain how KE-board will be integrated in both the back-end and front-end of KE-chain and elaborate about our testing approach.

### 5.1. REQUIREMENTS

The requirements specified in this section are all rated by the MoSCoW model. This means that each requirement belongs to one of the MoSCoW categories.

- M Must have, this category specifies the requirements that have to be included in order for the project to be a success.
- S Should have, this category includes requirements that have a high priority to be included.
- C Could have, this category specifies the requirements that would be a great addition to the system, but are not strictly required in order to have a feasible product.
- W Won't have, requirements that will not be implemented are included in this category. These requirements might be reviewed for future implementation.

#### 5.1.1. HIGH-LEVEL REQUIREMENTS

1. M - The system displays a dashboard with visualizations of KPI's related to KE-chain projects and contents.
2. M - The system distinguishes two user roles: Project-manager and Engineer.
3. M - The system shall provide information tailored to the user role and state of the system.
4. S - The system provides a customizable dashboard for every user.

#### 5.1.2. FUNCTIONAL REQUIREMENTS

5. M - The system shows a field with KPI widgets to the user.
6. M - The system provides a default list with standard KPI components to the user.

7. M - The system provides two basic dashboards which are initially generated when the user first opens the dashboard.
8. S - The user can drag KPI widgets within the dashboard to arrange them.
9. S - The user can add KPI widgets to the dashboard.
10. S - The user can remove KPI widgets from the dashboard.
11. S - The system saves the state of the dashboard automatically after adding, removing or moving a widget.
12. S - The user can edit the properties of each KPI widget.
13. S - The system updates the widgets automatically in real time.
14. C - The user can create custom KPI widgets using some parameters.
15. W - The system provides financial or product property parameters.

### KPI WIDGETS

As described in the previous sections, the following KPI widgets will be implemented for the two role-based dashboards:

- Engineer Dashboard
  - Progress and Changes over time
  - Critical Tasks
  - Hot Tasks
  - Open Tasks
- Project Manager Dashboard
  - Progress and Changes over time
  - Tasks in progress
  - Tasks not being worked on
  - Costly changes

#### 5.1.3. NON-FUNCTIONAL REQUIREMENTS

5. M - The dashboard is shown to the user within reasonable loading time.
6. M - The system user experience and design must comply with the current design of KE-chain.
7. M - The system is compliant with the non-functional requirements of KE-chain as specified in appendix A.

## 5.2. USER EXPERIENCE

The major goal of our system is to provide the user with a dashboard that informs the user of everything that is going on in the projects. For this, we define a dashboard that is used as a landing page for each project. This means that the dashboard is shown when a project is clicked.

When the dashboard is opened, the user is presented with a screen that is divided in multiple columns. The columns are filled with the dashboard widgets. Each widget holds some data. This could be ranging from a list of tasks the user is assigned to, to showing interactive graphs of the project's progress. The widgets can be dragged and dropped to be rearranged to the user's preference, but a default layout is provided, specific to the type of user that accesses the dashboard (engineer or project manager) as you can see in figure 5.1. The mockups were made with the tool Balsamiq [10].

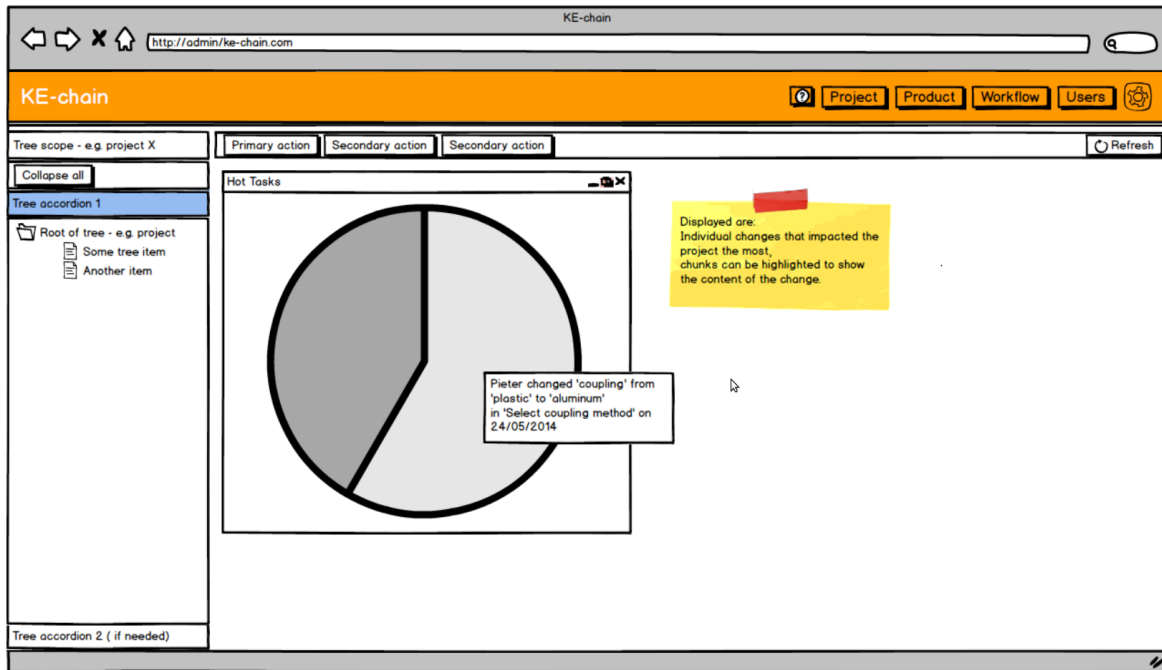


Figure 5.1: Design of the dashboard with a widget.

In addition, the user can open up a window that shows the available widgets, which can be added to the dashboard. Some of the widgets may require a minimal amount of configuration, such as setting the time-frame, interval or the maximum number of items. However, all widgets will be added with default settings, so the user does not have to set them if they don't need to.

To configure a widget, a user can press a little gear icon on the widget. Then the view of the widget will be changed into a property-changing view, where the user can edit the widgets properties. When the user is finished changing the properties it can save them and the widget will return to it's original state as can be seen in figure 5.2

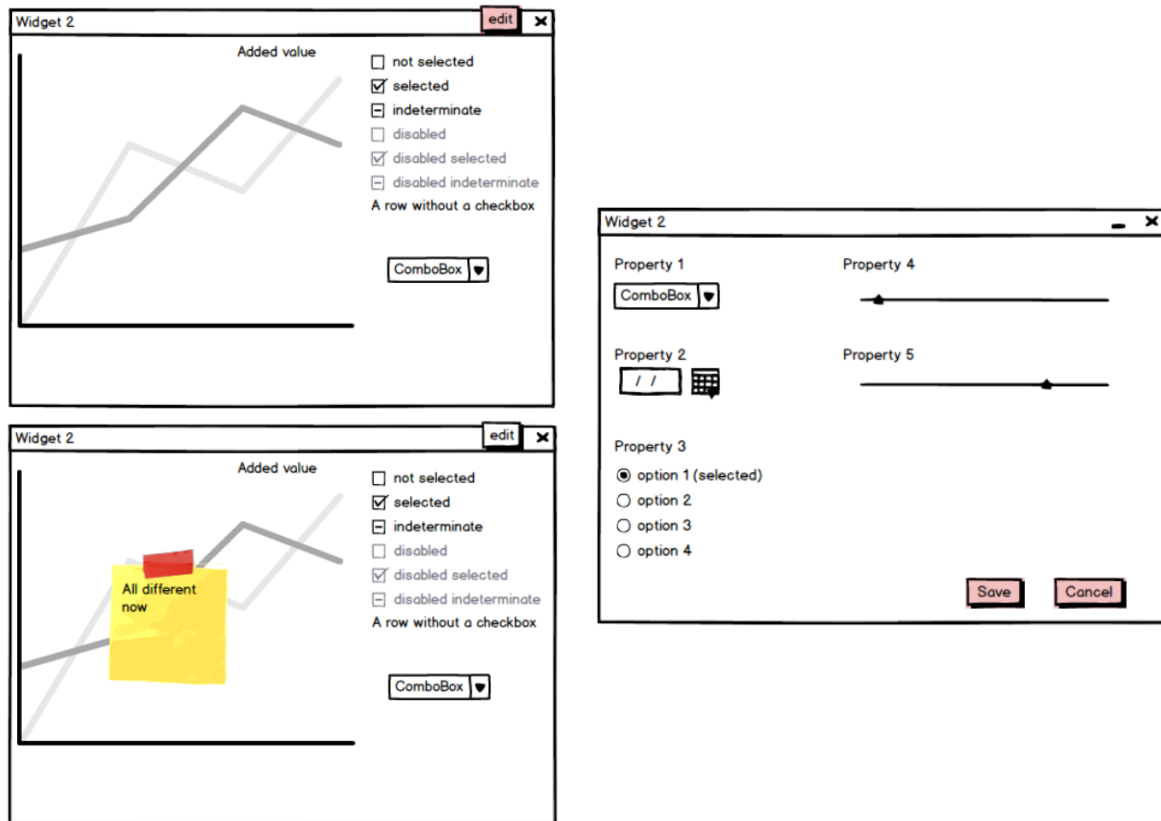


Figure 5.2: The process of editing the properties of a widget.

On the aspect of interactivity: Some widgets may have interactive content. This interactivity comes in two flavors. First of all, the user may mouseover a graph or other element of the widget to get more information. Example: The user mouses over a dot of scatter-plot in order to reveal what task the dot corresponds to. Secondly, the user may use the widget as a means of navigation. In this case the widget provides links to forms, workflows or other views already existing in KE-chain.

### 5.3. USE CASES

In the table below you find one of the use cases we defined. For the complete list of use cases, please refer to appendix D.

#### 5.3.1. VIEW DASHBOARD

<b>Actors</b>	Project Manager / Engineer
<b>Entry conditions</b>	None
<b>Activity diagram</b>	User navigates to the dashboard.
<b>Exit conditions</b>	Dashboard has been displayed.
<b>Exceptions</b>	If it is the first time the user visits the dashboard, a dashboard lay-out according to the role of the user is generated.

Table 5.1: Use case: view dashboard



## 5.4. TECHNICAL DESIGN

Now that we have established the requirements and use cases for our system, in this section we will discuss the design of the technical implementation. In subsection 5.4.1 we elaborate on the back-end architecture including the database design. Then in subsection 5.4.2 we talk about the front-end architecture.

### 5.4.1. BACK-END

Since KE-board will be integrated in KE-chain, we are obligated to work with the existing language and framework: Python [11] and Django [12]. These tools serve our purposes fine. The layout of each dashboard will be saved on the server, and it's layout can be retrieved by the front-end. For the widgets on the dashboard we will create an interface that allows the widgets to retrieve their data and properties. In addition, the back-end interface will supply functions that allows the user to edit their dashboard and widgets, so the state of the dashboard will be saved. Some early tests showed that calculating the data that needs to be showed can take quite long (over 10 seconds). Therefore we implement an asynchronous system that allows the data to be calculated at set intervals, so that when the data for a widget is requested, it can be immediately returned, because it has already been calculated and stored in the database.

The asynchronous system we use to handle the execution of our calculations is called Celery [13]. This system was previously not implemented in KE-chain, but it had been implemented before on a fork of KE-chain. This means there is some knowledge present at KE-works that we can use to implement Celery into KE-chain.

We use two major features of Celery: Scheduling and delays. The scheduling function allows us to set fixed intervals or moments that python functions can be executed. We use this to set an interval and let Celery call a function that will generate the data for our widgets. The delay function allow us to generate the data for a widget, and let it be executed by a Celery worker, so it will not delay the control flow. This is used when we want to call a function that generates the data for a widget (for instance when a user adds a widget to a dashboard) but does this in the background. Then the function will finish some time later, but the user is not forced to wait until the data is generated when he adds a widget, as this process can take some time.

In the figure 5.3 you can see the back-end class diagram. You can see that all classes are contained in modules. The most important relationship in this diagram is *Dashboard* has 0 or more *Widgets*. The front-end is contacts the back-end via the views. Celery functions are located in the *Tasks* file in the Tasks module. Only the relevant KE-chain classes are displayed in this diagram, with no inter-dependencies.

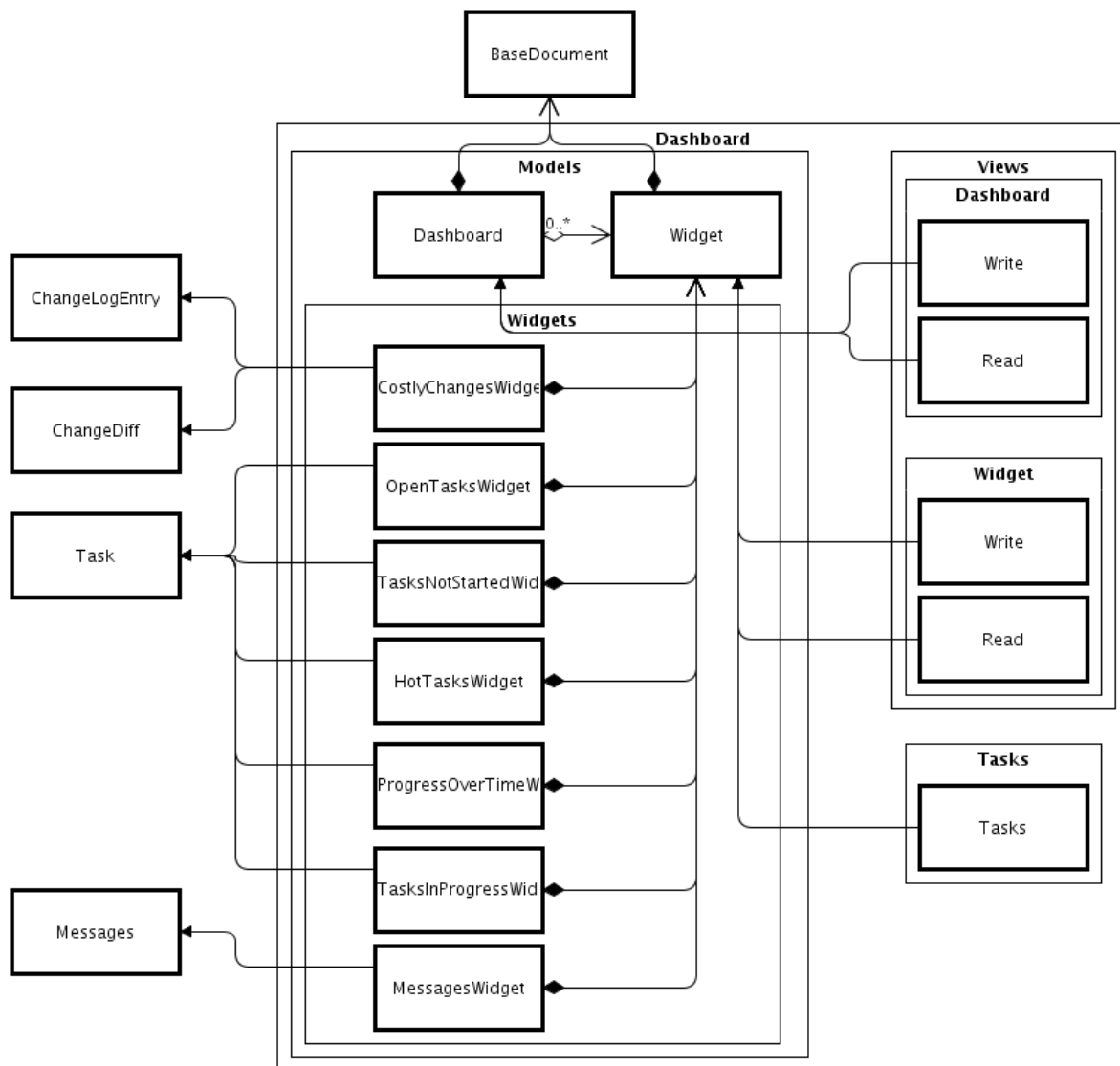


Figure 5.3: Back end class diagram

## DATABASE DESIGN

In this project we use a document-oriented database called MongoDB. This is quite different from a relational database. The main difference regarding modeling between the two types is that in a document-oriented database you are less concerned with data duplication. Since in MongoDB you can't make joins, you generally set up your documents so they can be retrieved by an indexed attribute in one go. If you need to couple two models that reside in two different documents, that means you have to make another separate call to the database, which impacts the performance in a negative way, since a round-trip to the database is a time-consuming process.

In our project we deal with two main models: dashboards and widgets. Since we work with a non-relational database we could choose for two options. We could either include widgets into the dashboard document or create a separate document for both the dashboards and the widgets. Because of the fact that a widget is always part of a dashboard, we have chosen to not create separate documents for the widgets. One of the arguments against this choice could be that data-sharing would become difficult or that it would cause data duplication. However, since we have chosen to include

user-specific widgets in chapter 4, we think widgets with identical data will be rare.

Below you find the structure of an example dashboard document including one widget.

```
{
  "_id" : ObjectId("53a01d53e7fd26341ad23b02"),
  "last_updated" : ISODate("2014-05-19T11:40:30.723Z"),
  "global_counter" : 172629,
  "created" : ISODate("2014-05-17T10:49:55.134Z"),
  "version" : 2880,
  "role" : "engineer",
  "for_user" : "admin",
  "model" : "Dashboard",
  "columns" : 2
  "widgets" : [
    {
      "model" : "HotTasksWidget",
      "_id" : ObjectId("53a0315fe7fd264ed0a933ff"),
      "col" : 0,
      "height" : 0,
      "display_name" : "Hot tasks",
      "is_generated" : false,
      "description" : "A table with tasks ...",
      "data" : [],
      "properties" : {
        "projectId" : {
          "value" : ObjectId("531979d2649bfb2439a7178c"),
          "propertyModel" : "hidden"
        },
        "number_of_days" : {
          "fieldLabel" : "Number of days",
          "description" : "The number of ...",
          "value" : 7,
          "propertyModel" : "numberfield"
        }
      }
    }
  ]
}
```

#### 5.4.2. FRONT-END

We want to use a system where we are free to add, remove and move around widgets in a screen. The front-end JavaScript framework Ext JS [14] that KE-chain is built with provides a tool for this called a *portal*. This is essentially a panel to which you can add little windows called portlets. These portlets will be our widgets. In addition, this tool supports dragging around the portlets to arrange them. In figure 5.4 you can see how a panel is built up out of different windows that can be dragged around in a very intuitive manner.

In KE-chain it is possible to open multiple pages at the same time. These pages are placed in different

“tabs” on the main screen. Our dashboard will be opened as the default page when clicking on a project from the navigation tree. This means we will implement our portal in a panel, that will be opened as a tab. Then, the individual widgets will call the server for the information they need, and the server will reply with the data so the widgets can be rendered.

The content of the widget can change from the view where it shows its contents to a small property window when the property button is clicked. Then the widget shows a small form with its properties so that the user can change them. These properties will then be saved to the server.

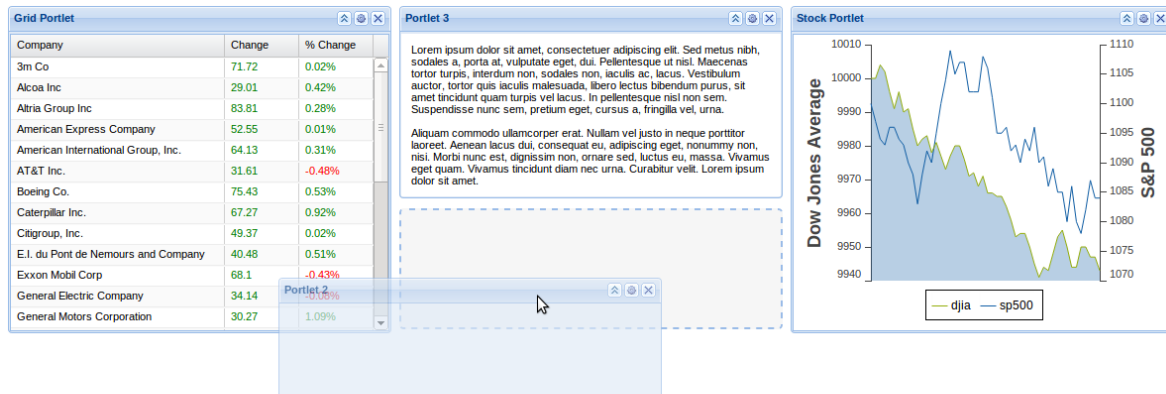


Figure 5.4: An example of a portal with a couple of portlets.

One of the challenges in the technical design lies in the loading time of the dashboard. As stated before, loading all KPI widgets synchronously will result in very high loading times, which are not acceptable to fulfil our requirements. Celery can generate the data for the widgets at any moment, or every interval we specify, but the front-end also has to be kept up to date with the latest data.

A couple of options are possible to show the newly generated data as soon as possible. For example we could think of an option in which we use some form of Reversed Ajax or COMET [15] in combination with Ext JS [16]. Another option is to use websockets as a plug-in for Ext JS [17]. When using one of these options, we can request the data-generation, and get it loaded in the widget as soon as they are complete, without having the users actively waiting for it. After trying some of these options, we found that these options require a lot of effort to implement. On top of that, from the client interviews we have derived that it will suffice if the dashboard refreshes its information with a time span of minutes or hours, since change rate of the information is not that high and there is no need to have the information available with absolute minimum latency. By calculating and refreshing our widget data automatically at fixed intervals, we will not surpass the loading time constraints, the KPI widgets will always be up-to-date according to the requirements of each widget.

In figure 5.5 the diagram of the front-end classes is being displayed. From this diagram, the *Dashboard* class is the class that is included as a tab inside KE-chain. The *Dashboard* in its turn contains a *PortletPanel*, which holds *PortalColumns*. These columns contain the *Portlets*. Each portlet holds one widget. You can see that the individual widgets are all subclassed from the *Widget* class. Each widget has their own store, and each store is linked to a model, that defines how the data should be interpreted. The controller classes control behaviour of the *Widgets* and *Portlets*, and provide functionality for the window that allows new widgets to be added. Only the relevant Ext JS classes are being included in this diagram.

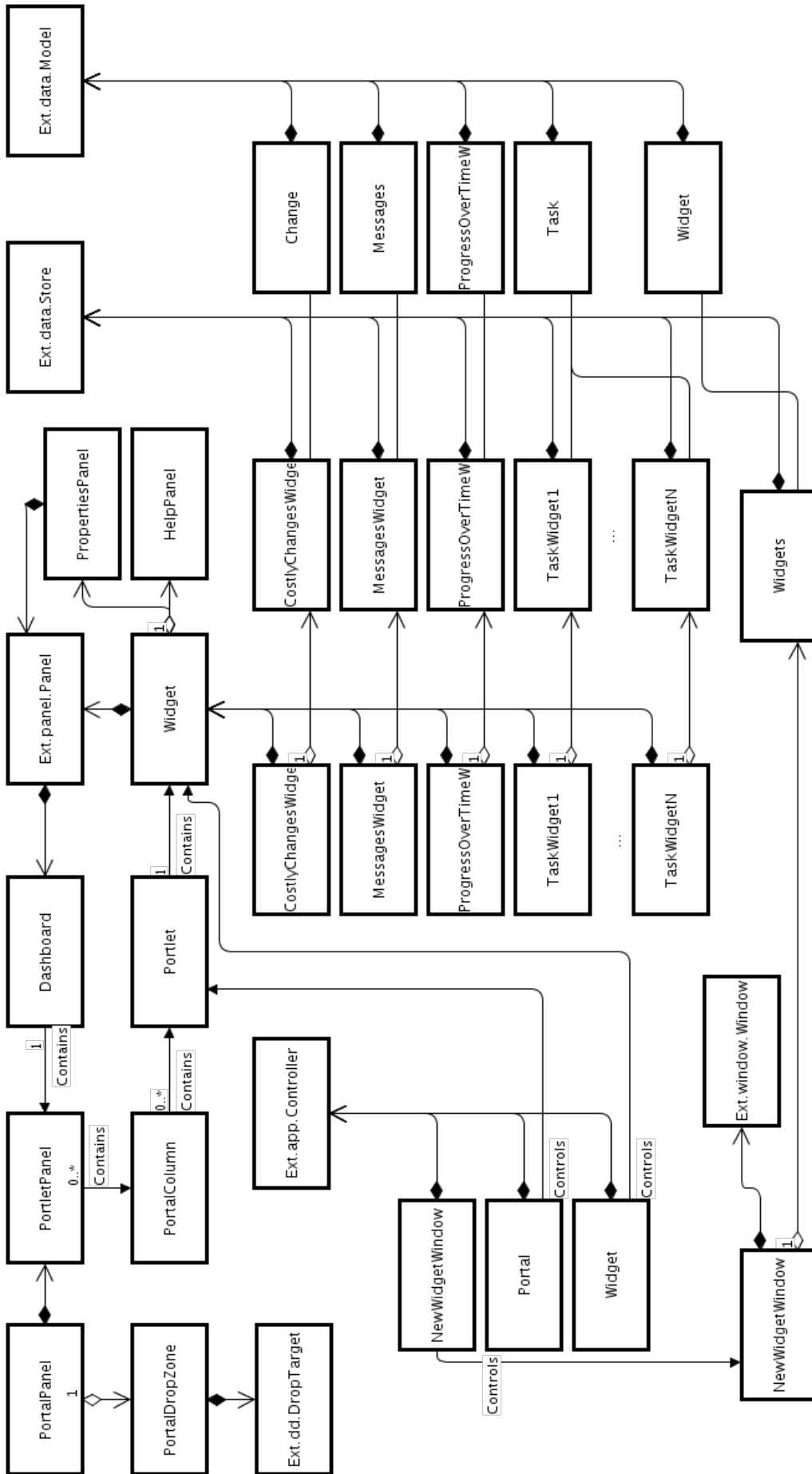


Figure 5.5: Front end class diagram.

## 5.5. TESTING

The testing aspect of this project can be divided into two main parts: the back-end and the front-end. These two parts do not only differ in programming language, but also have a completely different function. The back-end responds to calls from the front-end, queries the database and performs calculations. The front-end builds the application, listens to browser events (such as clicks) and queries the server for data. The two frameworks we use for these ends are not suited for a similar testing approach. That is why we handle testing for these parts in separate ways.

For the back-end we use both unit testing and integration testing. We use the Django testing suit to write unit tests that not only cover the functionality of our models, but also test the offered calls to the front-end to make sure that proper error-handling is perceived. Furthermore, for the data generation of the widgets we will create proper integration tests. Together this covers the testing needs for the back-end.

For the front-end a different approach is used. With the Ext JS framework it is difficult to test the front-end using unit tests because of the structure of the framework. Furthermore there is currently no possibility for automated testing of the user-interface at KE-works. Therefore we have decided to test the front-end manually. After each sprint we will test the front-end according to the use cases which can be found in appendix D. When testing the front-end we check two things: we verify that the application looks the way it is designed to and we interact with the front-end to verify that it behaves the way it is designed to for those actions.

At the end of the project we do an acceptance test in which we let the company use our application to make sure all requirements are met. We do not only test ourselves, but also use the sprint demonstration at the end of each sprint to gather feedback from the whole development team. Furthermore, in our weekly meeting with our supervisor at KE-works, we show a demo of our application and ask if it meets the company's requirements. For more information about the development process, please see section 6.1.

With that, the basis for the implementation of KE-board is now established. In the next chapter we will discuss how the implementation has taken place, including the reflexion on the implementation process and feedback on our product.

# 6

## IMPLEMENTATION OF KE-BOARD

### 6.1. METHODS AND TOOLS

At KE-works, the SCRUM [18] framework is used for software development. We joined the development team, although we only worked on our own project. This means we participated in the sprints, the sprint kickoffs, the daily SCRUM meeting and the sprint reviews. The sprints lasted for two weeks and were concluded with a demonstration by the developers. By participating in the development team, we experienced professional software development hands-on. We quickly learned all facets of the SCRUM framework when putting it into practice. Before each sprint kick-off, we defined what stories would be implemented in the upcoming sprint. When the sprint was ongoing, we divided the stories into sub-tasks and worked on those tasks. With help of JIRA [19] used at KE-works we could collaborate easily and log our work by assigning sub-tasks (issues), indicating the status (To-do, In progress, Requires feedback and Done) of those tasks and place comments. One of the key advantages of this, was that it was always clear what each member of the team was working on and what had to be done to complete a story.

To document our work so that it can be referenced by KE-works at any moment, we used Confluence [20] deployed by KE-works. By using this, we made sure that all the information generated and gathered during the project would be available to the employees at KE-works during and after the project was finished.

When working on our code, we used Subversion [21] (SVN) as a version control system. At the start of the project, we branched the latest stable version of KE-chain off into our own branch. During the project, we worked together on that branch. When there was an update available on the latest stable version of KE-chain, we merged those changes into our own branch. That way we kept our branch up to date, but did not affect ongoing development. Moreover, when KE-board is complete, it can be easily merged back into the existing application, since the periodic merging made sure that the difference between our and the KE-chain branch was as small as possible at all times.

To make sure our branch is always working well, the Jenkins CI [22] (Continuous Integration) suite was used. This system allowed us to keep an eye on any failed tests, code coverage of the tests and style warnings from pylint [23] (Python), pep-8 [24] (Python) and JSLint [25] (JavaScript). In the KE-works office there is a computer screen always showing the latest status of our branch. If any tests failed, the screen turned red and we immediately knew that we had to solve a problem we did not foresee.

## 6.2. SPRINT EVALUATIONS

In this section we will discuss the problems we had during the implementation sprints and the way we handled them. As you can see in Figure 6.1 we implemented over 7500 lines of code in approximately 5 weeks. The first sprint took from May 12 to 16 in which we started a bit late. The second sprint lasted from May 19 to 30. The last sprint took from June 2 to June 13. After the last sprint we had to complete some of the last stories and performed some code formatting improvements.

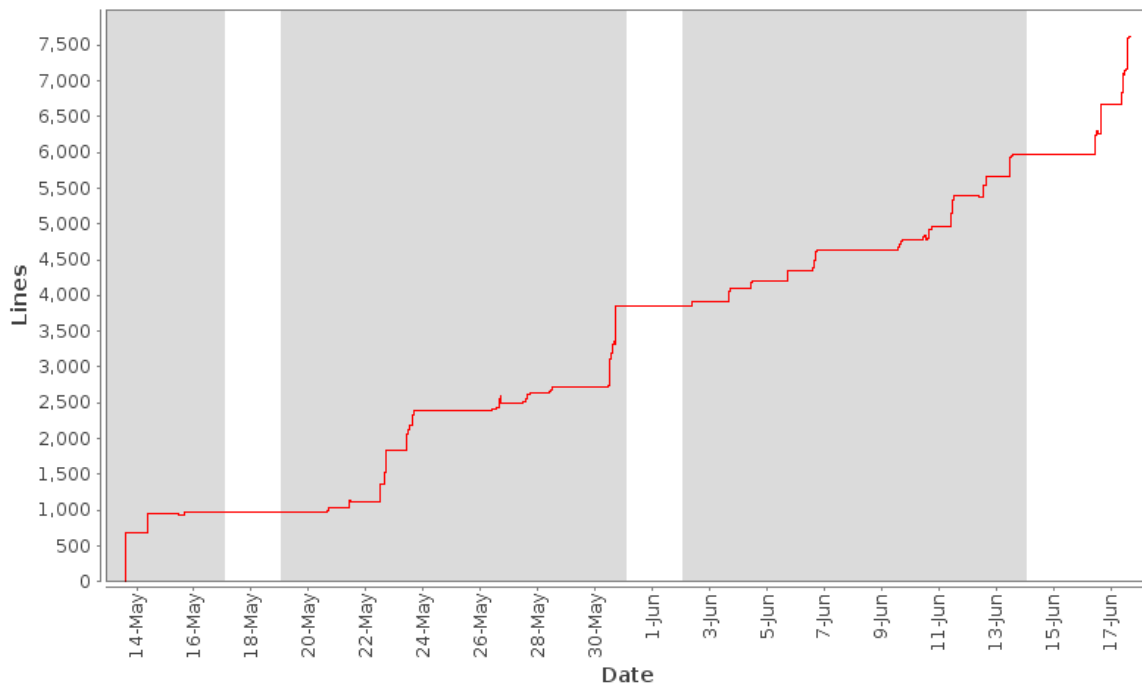


Figure 6.1: Number of lines over time

### 6.2.1. SPRINT 1

This sprint was only one week for us, as we joined the KE-chain development team mid-sprint. This sprint went quite well, considering that setting up the development environment and application can be a tedious process. Fortunately integrating the Celery framework into KE-chain went well and did not require as much troubleshooting as we (and the developers at KE-works) thought it would. Originally it was the plan to build the first widget, but there was not enough time to do that this sprint, also the priority of widgets to implement wasn't determined yet.

#### COMPLETED STORIES

**Create Requirements** We created the requirements according to the ideas we had from the conversations with KE-works. After our supervisor at KE-works reviewed the requirements we made some minor changes.

**Create technical design** Documentation of our decisions regarding asynchronous loading of data, placement of the dashboard, high-level description of our front-end and back-end approach and testing.

**Design user experience** High-level description of the experience we want to give the users of KE-board.



**Dashboard set-up** First set-up of the dashboard. Placement in front-end, creating back-end and front-end default classes. Installation of Celery, testing set-up.

### 6.2.2. SPRINT 2

During the first week of this sprint we discovered that a dashboard-list was not a good idea, and it was removed from the requirements. We had very ambitious goals for this sprint, and not all stories were completed. However that is not a direct problem, as we expected that we might not be able to finish all stories. Near the end of the sprint, a lot of testing was done and the code was sent to SIG (Software Improvement Group) for evaluation.

#### COMPLETED STORIES

**Determine & design 4 high-priority KPI-widgets per dashboard** Drawing conclusions from the interviews and questionnaire. Decide which 4 widgets are going to be implemented.

**Set up role specific dashboards** Create role-specific loading of the dashboard.

**Create default KPI widget module** Creation of default KPI widget module. Methods for adding, editing and removing properties in widgets. Methods for adding, editing and removing widgets from the dashboard.

**Hot Tasks Widget (#2 Engineer)** Creation of the data generation function and front-end, visualization in front-end.

**Progress over time widget (#1 Engineer / Manager)** Creation of the data generation function and front-end, visualization in front-end. Problems occurred regarding the generation of progress over time. The progress was not saved well in the archives and the calculation is very tough since some projects have over 500 tasks of which the average has to be calculated over time. We have chosen an approach in which the progress over time is only calculated once and then only new dates get added. This gave a major performance improvement.

#### UNCOMPLETED STORIES

Since the progress over time widget took a lot more time than expected, we chose to not implement the following stories.

**Create Open Tasks KPI Widget (#3 Engineer)**

**Create Tasks in progress KPI widget (#2 Manager)**

### 6.2.3. SPRINT 3

At the start of this sprint, most of the ground work was done. So this sprint mainly consisted of adding all the widgets we needed. Also we had to make sure that the dashboard would always be saved (persistent). Doing our development we have held the feedback on our code from SIG. We managed to complete all functionality we set out to include. After this sprint, we spent around one day to do some minor bug-fixes and increase the amount of tests.

#### COMPLETED STORIES

**Create Open Tasks KPI Widget (#3 Engineer)** Completing the widget including calculation, visuals and testing.

**Create Tasks in progress KPI widget (#2 Manager)** Completing the widget including calculation, visuals and testing.

**Create Tasks not started KPI Widget (#3 Manager)** Completing the widget including calculation, visuals and testing.

**Finalize draft report** Finish of the report as far as possible at this point and submit for review.

**Performance optimization** Some calculations that are done now every time can be cached and reused. Create this data-caching structure where applicable.

**Create Messages Widget** Port over an existing panel from the KE-chain home view to its own widget.

**Create Hot Tasks KPI Widget (#2 Engineer)** Completing the widget including calculation, visuals and testing.

**User-editable persistent dashboard** Make sure the dashboard is saved when widgets are moved, added or removed. Include the ability to add new widgets and refresh the dashboard. Create a way to receive additional information about the widget. Create a nice header for the dashboard including the project name. Lettings widgets have variable height.

**Create Costly Changes KPI Widget (#4 Manager)** Completing the widget including calculation, visuals and testing.

## UNCOMPLETED STORIES

We found that the critical task widget did not add enough value to implement it, as we concluded that we had not enough time to implement every widget. Therefore we chose to add a messages widget as a proof of concept to show KE-works that existing content in KE-chain can be easily placed in a widget as this could be implemented within a small timeframe. In hindsight this was a bad decision, since in the end we concluded that the Engineer's KE-board could use the critical tasks widget for a full overview.

**Critical Tasks Widget (#4 Engineer)**

## 6.3. FEEDBACK FROM SIG

Our code was submitted to SIG at the end of Sprint 2, so that we could receive the feedback halfway through sprint 3. This gave us the opportunity to improve our code in the final sprint based on the recommendations by SIG and re-submit it for re-measurement.

The feedback received from SIG can be read in appendix F (Dutch). SIG graded our code with a rating of almost four stars. The main two reasons for not getting a higher score was due to a lower rating for both code duplication and unit size. We have put a lot of time and effort into following the recommendations from SIG. To prevent code duplication we have moved as much code as possible to superclasses and have separated individual pieces of code as for example AJAX-requests in their own methods or configurations. To decrease unit size we have had a critical look at the front-end and back-end code. At the back-end we have successfully decreased the unit size, but in the front-end this was more difficult to achieve. This was mainly because of the large number of elements that is used in some functions which makes it hard to split them up in separate functions because of the large amount of function parameters it would require. Furthermore we sought to keep our python test suite up to date and made sure all back-end code was thoroughly tested. After the final sprint the code was submitted to SIG again for a final review of which the feedback can be read in appendix F as well.

Unfortunately the final feedback from SIG was not positive. They found that the code duplication had increased and the unit size had not decreased. In the feedback it is mentioned that there are still some places that have possibilities for unit size reduction and that we have an above average amount of long methods. Therefore they concluded that we had not taken the recommendations from the first measurement into account.

As mentioned earlier, we attempted to reduce unit size where possible. For instance in the example mentioned by SIG, it would in our opinion be unpractical to substitute parts of a function into separate methods, because of the high number of parameters that would be involved. Therefore we think it would not have contributed to the maintainability of our code. Moreover we feel that due to the use of the Ext JS framework, methods are generally quite long because they contain the configuration objects of components. However, because these configurations are highly structured we do not think this decreases the maintainability of the system.

We have communicated our opinion to SIG on which they indicated that we should have communicated our opinion along with the code in documentation. However, taken our reaction into account, SIG concluded that, their recommendations had been partly followed.

## 6.4. TEST RESULTS

To make sure the implementation is properly tested, we tested in two methods as specified earlier in section 5.5. We did this by testing each individual story by hand for the front-end, and used unit tests to cover the back end. As for the unit tests, we aimed to have all our features properly tested. We kept an eye on the status of our code coverage using the Jenkins continuous integration suite. Figure 6.2 shows the progress of the back-end test coverage by build number.

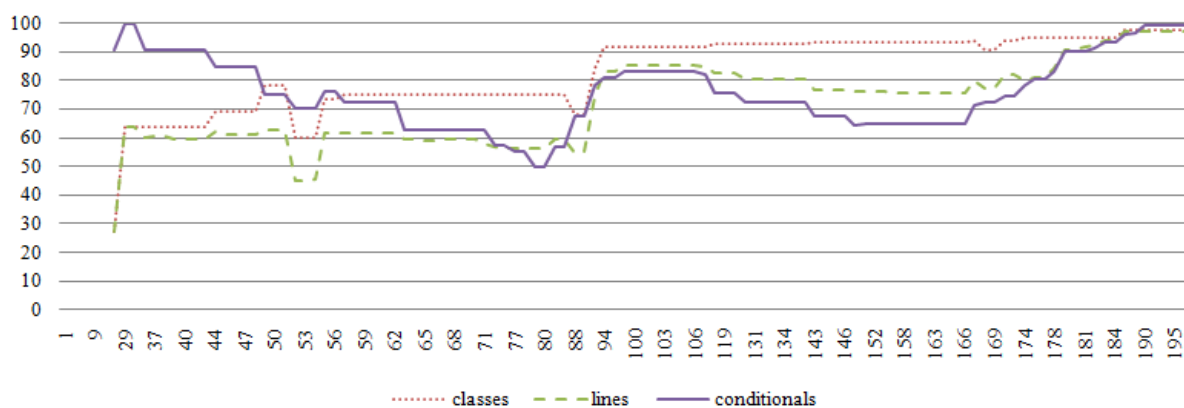


Figure 6.2: Back-end Code Coverage

As can be seen in the figure, initially there was very little code, and therefore very little conditional statements, which lead to the low line and class coverage, and the relatively high conditional coverage. As implementation progressed, the coverage goes down until the moment we built a lot of tests (build 88-100). Then afterwards the code coverage slightly dropped again, up until the final phase of the implementation where more attention was payed to testing. The code coverage was increased to about 100%. The testing of our project has occurred as we set out to do in section 5.5. For the back-end, we have not only created unit-tests for all our functionality. Additionally we have created integration tests in order to test our data-generation functions. These tests create an environment that should trigger certain results, and test if those results are right.

For the front-end, at the end of each story we tested it by hand to make sure all the functionality was implemented and working as expected. Stories were only marked as completed after all remaining issues were resolved. We quickly noticed that testing manually was far from ideal. Therefore, in the second sprint we considered to install Selenium[26] to run automated front-end tests. Unfortunately we quickly noticed that it would take too much time to integrate Selenium into the architecture of KE-chain. Therefore we have decided to continue with testing by hand.

## 6.5. MEETING THE REQUIREMENTS

In section 5.1 we described the requirements for our project, and here we discuss whether all our requirements are met. For the requirements that were not met, we discuss why and how this was handled.

Firstly, all our requirements marked as *must have* are met. This means our product can be marked as successful. Secondly, all our *should have* requirements are met except requirement 13: “The system updates the widgets automatically in real time.” We haven’t met this requirement because we chose to calculate and update widgets in a timeframe that is reasonable for each individual widget, instead of having the widgets update in real time. We chose this because it costs less time to implement, decreases complexity of the system and does not require any external libraries. Moreover, we have not created any KPI widgets that would require real-time updates. Lastly, our *could have* and *won’t have* requirements are not met. For the *could have* requirement, this was mostly because it would cost a lot of time to implement, whilst adding very little value.

We can conclude that the requirements are met, and that from that perspective, the product is implemented successfully. However, we will seek to find the opinion of the consultants at KE-works, and ask them if they think we fulfilled the requirements and whether they see all the measures they indicated as important. Altogether, this will show the success of the KE-board implementation.

## 6.6. FUNCTIONALITY

In this section we will review the finished product and show what it is capable of. First we show the dashboard as it is presented to a user when he sees it for the first time. Then we discuss the individual widgets in section 7.3.4. In figure 6.3 the dashboard is displayed. In this view we see four widgets placed on the dashboard, and there are an ‘add widget’ and ‘refresh’ button on the top left and top right corners respectively.

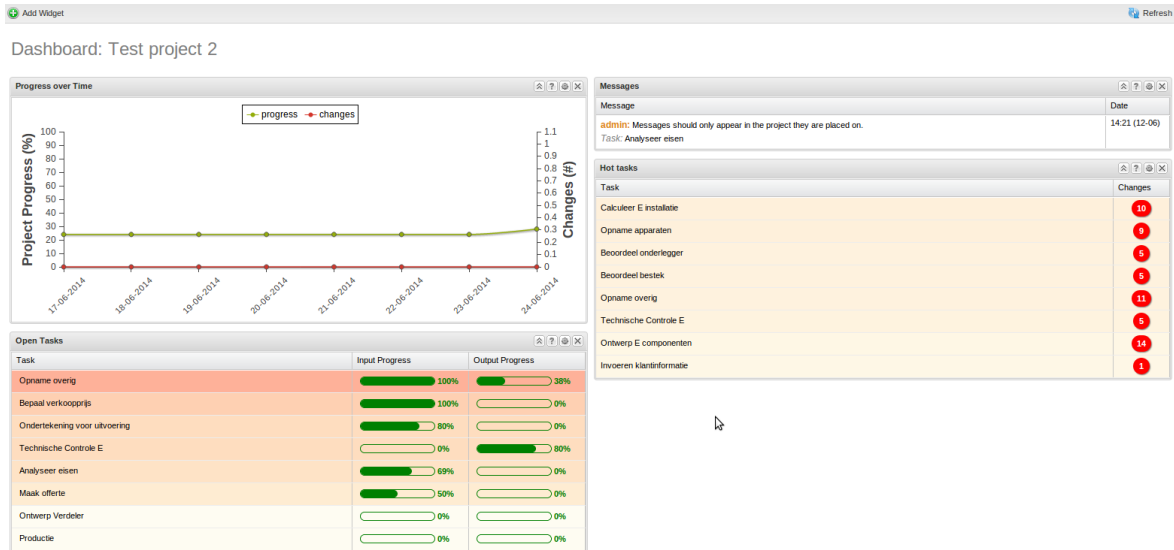


Figure 6.3: KE-board default engineer dashboard

Users are free to customize their dashboard by dragging around the widgets to change their location on the dashboard. While dragging a widget, a grid appears on the spot where the widget will be placed when the user lets go the mouse button. This is shown in figure 6.4.

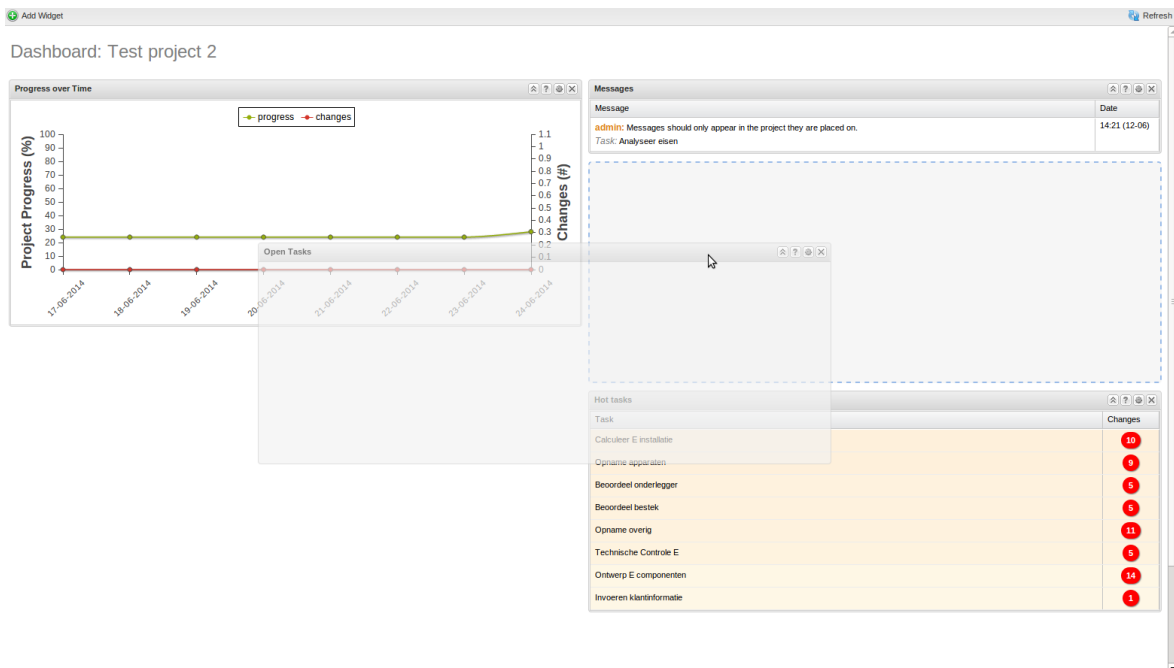


Figure 6.4: A widget is being dragged to another location

To change the properties of an individual widget, a user can click the gear icon to reveal the properties. Then the user can change the properties, and finally the user can save them. When the new properties are saved, the widget is reloaded and the view returns to its previous state. This can be seen in figure 6.5 and 6.6.

Task	Input Progress	Output Progress
Opname overig	100%	38%
Bepaal verkoopprijs	100%	0%
Ondertekening voor uitvoering	80%	0%
Analyseer eisen	69%	0%
Maak offerte	50%	0%
Technische Controle E	0%	80%
Ontwerp Verdeler	0%	0%
Productie	0%	0%

Figure 6.5: Properties of a widget can be opened.

**Open Tasks**

*Maximum number of tasks:*

7

The maximum number of tasks to display in the widget.

Save Cancel

Figure 6.6: Properties of a widget can be changed and saved.

Widgets can be added by using the 'add widget' button in the toolbar at the top of the dashboard. When this button is clicked, a window opens up with the list of available widgets. From this list, the user can select a widget and then add it to the dashboard by clicking 'add widget'. This procedure is displayed in figure 6.7.

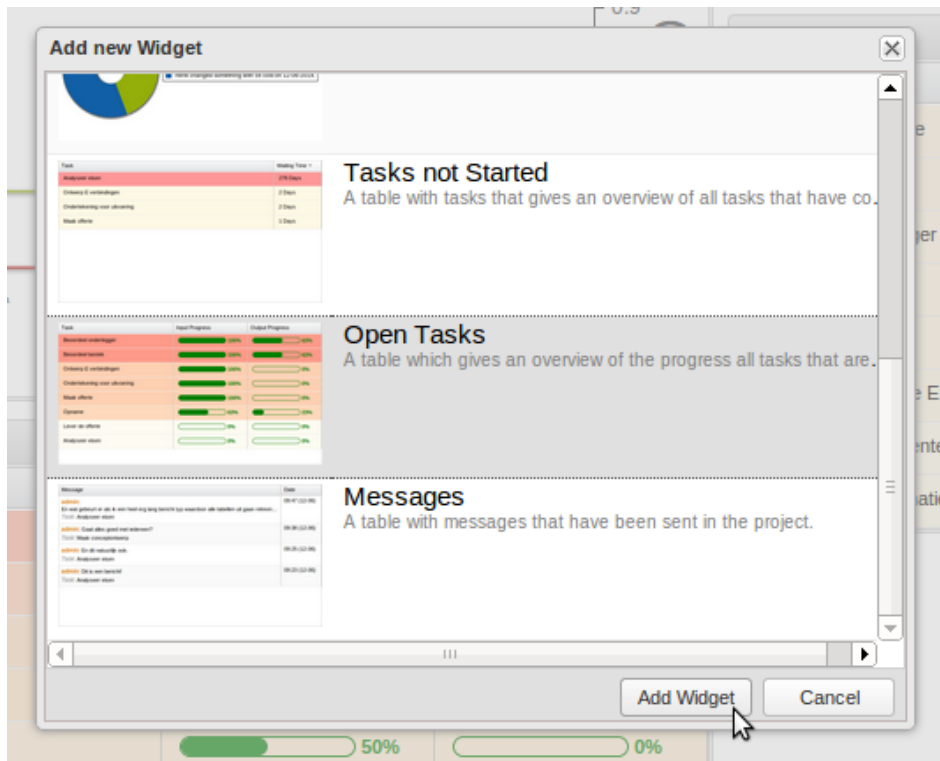


Figure 6.7: Select a widget from the list and click 'add widget' to add it.

Another type of view that can be opened on a widget is the help view. This works similar to the properties, and displays a text to the user to inform him of what is shown in the widget. This functionality is shown in figure 6.8 and 6.9.

Hot tasks	
Task	Changes
Calculeer E installatie	10
Opname apparaten	9
Beoordeel onderlegger	5
Beoordeel bestek	5
Opname overig	11
Technische Controle E	5
Ontwerp E componenten	14
Invoeren klantinformatie	1

Figure 6.8: Clicking the help icon on a widget.

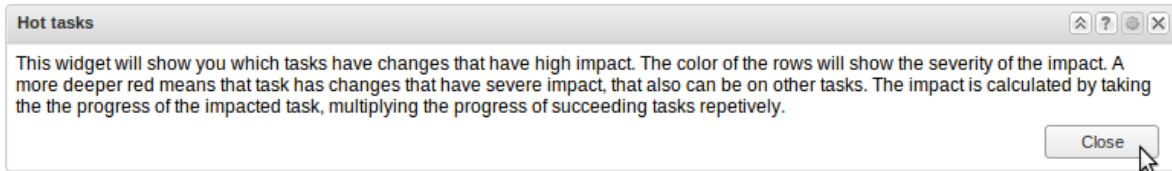


Figure 6.9: The help text is being displayed on a widget.

To remove a widget from the dashboard, the user can press the close (cross) button on the widget. When a user does this, he is prompted with a window where he needs to confirm the decision. When the user confirms, the widget is removed from the dashboard. This can be seen in figure 6.10 and 6.11. Of course, the widget can be added again through the 'add widget' button.

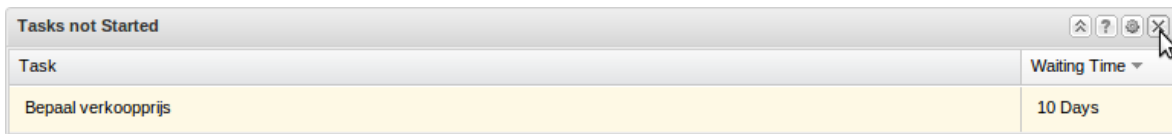


Figure 6.10: A widget can be removed by pressing the cross button

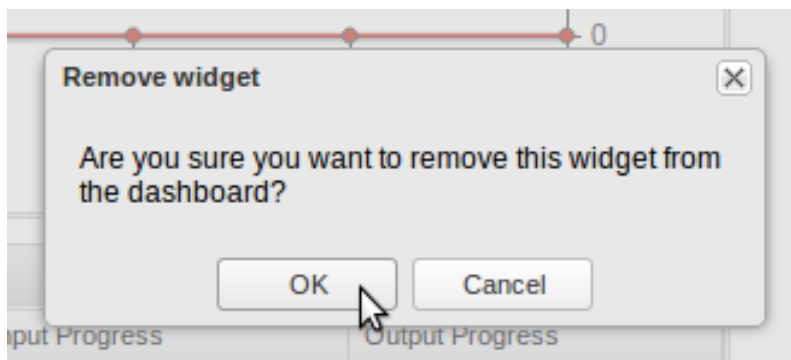


Figure 6.11: The removal of a widget need to be confirmed.

This concludes the functionality of KE-board. In the next chapter, we evaluate KE-board by reviewing the individual widgets, discuss feedback on KE-board and provide recommendations for future work and for the development process at KE-works in general.



# 7

## EVALUATION

For the evaluation of KE-board we have conducted three interviews with consultants at KE-works. Furthermore we have sent a questionnaire to all consultants at KE-works to evaluate the functionality of the widgets to see if the created KPI's really achieve their goals. The results of both are discussed in this chapter after the evaluation of our project development in general. Finally we give a general conclusion of our project.

### 7.1. DEVELOPMENT METHODOLOGY

As specified in section 6.1, the SCRUM methodology was used as development framework. In this section we reflect on the use of that development framework. The SCRUM system was already in use at KE-works when we joined, and we joined their development team for the duration of our project when we started developing although we worked on a separate project. Everyday during the daily scrum we could get insight in all the problems that were occurring in the entire team and give our opinion and feedback, as well as get feedback and help from the other developers. This was a good learning opportunity, was very helpful and only cost a little bit of time.

At the start of the sprint, there was a kick-off meeting, in which the team decided what stories to implement that sprint, as well as giving those stories a score on how much work we would expect them to be. These meetings are a very important part of the scrum framework, but they were not useful for us, because we have little knowledge about KE-chain. Also, we planned our sprints on our own and simply presented them in the kickoff, as no other developers were involved. This meetings were often very lengthy, because different members of the team did not agree on what was most important to include in the sprint. Furthermore there often was a lot of debating on the exact content of a story. On top of that, the prioritization of the stories does not matter to most developers.

### 7.2. PLANNING

In section 2.3 we defined our intended planning. In reality, we stumbled upon some problems which caused some of the phases to take longer. The actual timeline of our project can be found in Table 7.1. In the research phase the planning of the interviews with the users of KE-chain was difficult and took more time than expected. Drawing the final conclusions therefore overlapped with the design and the start of the implementation phase. This has caused the design phase to take a week longer as well. We have continuously adapted our planning and chose to start with a new phase on the intended

dates. We managed to finish the implementation phase in the intended five weeks and only had to do some minor changes at the beginning of the evaluation phase.

Week #	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10
Research										
Design										
Implementation										
Evaluation										

Table 7.1: Actual Project Timeline

## 7.3. PRODUCT

In this section we evaluate the final product as we delivered it to KE-works.

### 7.3.1. INTERVIEW

To evaluate whether KE-board is the product KE-works expected, we have conducted three interviews with consultants at KE-works. The interviews took 45 to 60 minutes each and were held by the two team members Sjoerd en Michiel and one consultant using the KE-chain application with KE-board integrated. The interviewees were using KE-board with minimum prior knowledge, as they have only seen it demonstrated to them in the development stage. The results are discussed in section 7.3.3 and 7.3.4.

### 7.3.2. QUESTIONNAIRE

The look and feel of the widgets have been discussed in the interviews with the consultants. However, to get to know if the KPI's actually contribute to the initial goal of KE-board, we have conducted a questionnaire to all consultants of KE-chain in which we ask them to only look at the *functionality* of the widget. The set-up and results of the questionnaire can be found in appendix H. In short we have simulated a couple of cases for each widget. For all these cases we have asked the consultants to explain which conclusions they would draw. Furthermore we have asked them to grade each KPI from 1 to 5 on whether it contributes to its purpose and helps to achieve its goal. The questionnaire was completed by 6 consultants. The results are summarized and discussed in section 7.3.4.

### 7.3.3. KE-BOARD

One issue that arises when users first see the widgets, is that it is not always immediately clear what they are seeing within them. For instance, they don't know that the information in a widget is calculated based on a certain start date, and is limited to some factor. We have tried to solve this issue by adding the help button on the widget to provide more information. However, it seems to be necessary to provide clear information on what is seen in a widget on its main view.

Another suggestion that has risen from the interviews is that there are a number of widgets that consist of a list of tasks, but a common factor between these lists is missing. That makes it harder to keep track of the complete picture. For instance, the due date of a task could be displayed in all tasks.

The moment users open the dashboard they are impressed with the amount of information presented. Generally they find it very easy and intuitive to customize their dashboard. Moving, adding and removing widgets are actions that are very easy to do. They find that it works very well and

presents them with a very nice overview of the project. A lot of information that is present in KE-chain is represented by the dashboards. It is generally felt that KE-board can be implemented in KE-chain on a short term.

### 7.3.4. WIDGETS

In this section we discuss the results of both the interview and the questionnaire of each widget that has been implemented and show the final looks of the widgets. First of all in table 7.2 you can find the average score of each widget according to the 6 respondents at the question wether they think that the widget helps to achieve its goal.

Widget	$\mu_{score}$
Progress over time widget	4.66
Tasks in progress widget	4.16
Open tasks widget	4.00
Tasks not started widget	4.00
Hot tasks widget	3.66
Costly changes widget	3.16

Table 7.2: Average score per widget

#### PROGRESS OVER TIME WIDGET

**Interview** The right information is presented in this widget as the relation between progress and the amount of changes is clear. It is suggested to use a different type of graph for the changes, to indicate the difference better. Also it is suggested that the changes are split up into tree separate kind of changes: additions, modifications and removals.

**Questionnaire** The respondents all draw nearly perfect conclusions from the widget. However, there is still a lot of information open for interpretation. It is not clear what is happening when a lot of changes occur but the progress does not rise. It could be of value to split changes up in to additions, modifications and deletions to get more insight. The widget is graded with an average of 4.66, which makes this a successful widget.

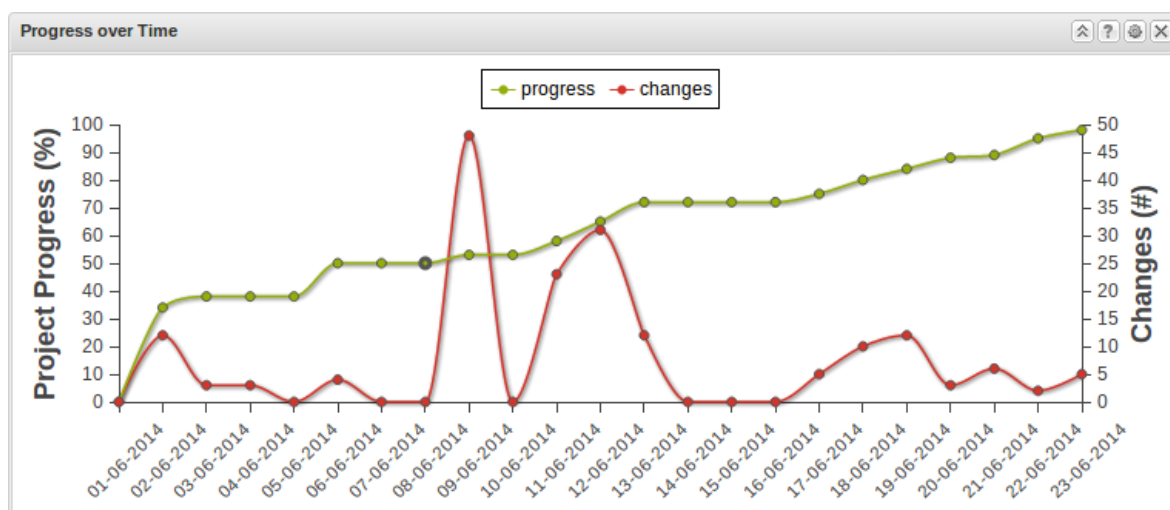
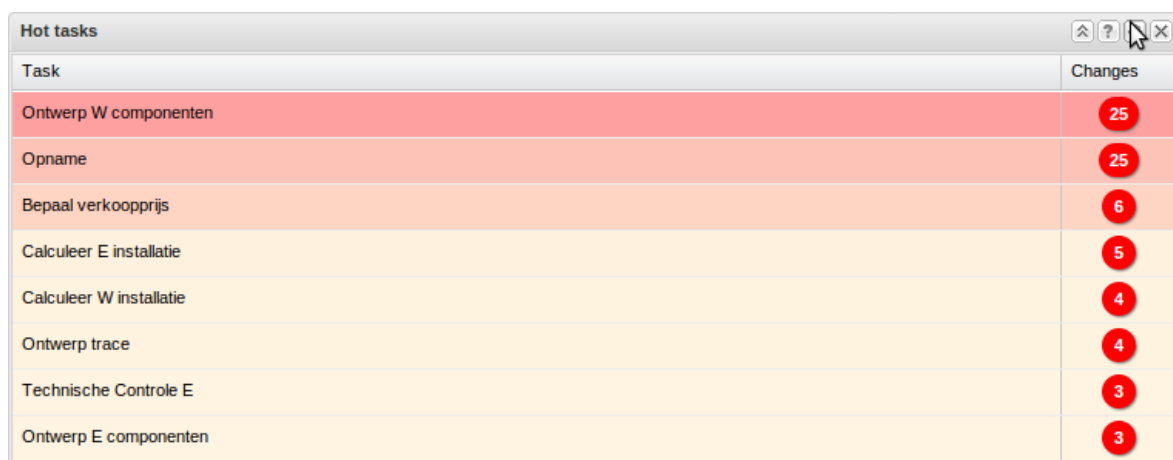


Figure 7.1: Progress over time widget

## HOT TASKS WIDGET

**Interview** A lot of interpretation is needed to extract the essence of the widget. It is not immediately clear what the coloring of the rows means. Also more information about the tasks could be shown, such as the due date or the assignees.

**Questionnaire** In general the respondents draw the right conclusions. They are able to see which tasks have been highly impacted, but nearly all of them wonder how the list is sorted. The list is sorted by impact, but there should be a way of actually seeing this in the widget. It is clear to the respondents that the item that is on top is most impacted and that they should not only look at the number of changes. The goal of this widget is to minimize outdated information. Since all respondents are able to identify the hot tasks and grade this widget with an average of 3.66, we may state that this widget achieves its goal, but could be improved by visualizing the impact in a better way.



Task	Changes
Ontwerp W componenten	25
Opname	25
Bepaal verkoopprijs	6
Calculeer E installatie	5
Calculeer W installatie	4
Ontwerp trace	4
Technische Controle E	3
Ontwerp E componenten	3

Figure 7.2: Hot tasks widget

## OPEN TASKS WIDGET

**Interview** It isn't clear on what data the tasks are sorted, and it could be made more clear that these are the tasks that the engineer should be working on. Also it is suggested to give the input progress bar a different color than the output progress bar.

**Questionnaire** The respondents are able to get a good overview of which tasks are important and draw the right conclusions. However, there is a lot of discussion about which tasks should be started. In this case the goal is to finish tasks that are almost finished as fast as possible to reduce waste (empty information elements). Unfortunately, this is not understood by all respondents and it might be good to clarify this by communicating the goal of the widget more properly. On the other hands, when grading the achievement of the goal of the widget, the respondents grade an average of 4.

Open Tasks		
Task	Input Progress	Output Progress
Maak offerte	100%	52%
Ondertekening voor uitvoering	100%	41%
Ontwerp E verbindingen	100%	20%
Beoordeel onderlegger	100%	5%
Opname	63%	23%
Beoordeel bestek	40%	18%

Figure 7.3: Open tasks widget

### TASKS NOT STARTED WIDGET

**Interview** Expanding this widget with more columns such as ‘assigned users’ and ‘due date’ can increase the use for this widget. It is seen as a very useful widget though.

**Questionnaire** All respondents draw the right conclusion and see that tasks which have high waiting time are colored more red and should be started. Suggestions are being made that automatic reminders should be sent when tasks that can be started are located. It is not clear to some respondents that all tasks in this list have a 100% input progress, which makes the widget unclear. The respondents grade the widgets contribution to its goal with an average of 4.

Tasks not Started	
Task	Waiting Time
Maak offerte	17 Days
Ontwerp E verbindingen	14 Days
Ondertekening voor uitvoering	1 Days

Figure 7.4: Tasks not started widget

### COSTLY CHANGES WIDGET

**Interview** There has to be given more detailed information what information changed *exactly* to make this widget more useful. Currently it is not clear what is the use of this widget. Presenting a tooltip with this more detailed information and allowing the users to act upon the information presented can improve this widget.

**Questionnaire** Although the correspondents seem to draw to right conclusions, the widget is not very helpful. The correspondents see which changes are the most costly, but they do not know how costly or where to find how costly they were and how to prevent this in the future. This widget is graded an average score of 3.16 which makes this the least successful KPI. The main reason for this is that it does not give the ability to perform an action, so it is difficult to draw conclusions or go deeper.

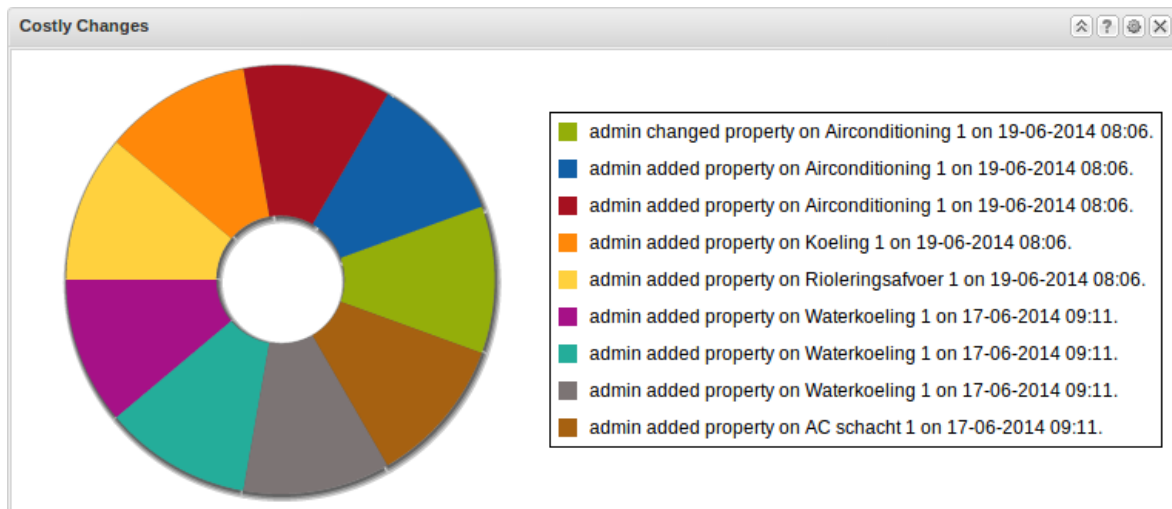


Figure 7.5: Costly changes widget

### TASKS IN PROGRESS WIDGET

**Interview** The column names: 'lead time' and 'working time' are a bit technical and not very clear. Suggested is that when there is no due date specified on a task, that this is displayed instead of '0 Days'. Furthermore this is perceived as a very nice and useful widget.

**Questionnaire** It is clear to the respondents that the color coding tells them which tasks are critical, but again it is not acceptable to sort on something else than the color coding. However, most respondents are able to draw the right conclusions apart from the sorting. The color coding is a very good way of making clear what is critical, but the respondents want to see what is being calculated and what exactly the widgets are sorted on. The goal of this widget is to prevent overdue tasks and the correspondents think it contributes to this goal with an average fo 4.16. The parameters like lead time and working time should be clarified somewhere.

Task	Progress	Lead time	Working time	Remaining Time
Opname	65%	10 Days	50 Days	-25 Days
Maak offerte	52%	13 Days	26 Days	1 Days
Ontwerp E-verbindingen	20%	42 Days	3 Days	3 Days
Beoordeel onderlegger	71%	15 Days	2 Days	5 Days
Ondertekening voor uitvoering	41%	17 Days	3 Days	5 Days

Figure 7.6: Tasks in progress widget

### 7.3.5. CONCLUSION

Despite the fact that there are suggestions for improvements for most widgets, most of them have been graded as useful and contributing to achievement of their target. Widgets as *Progress over time widget* and the *Tasks not started* have been graded high and are therefore expected to improve the product development process in a positive manner. Widgets as for example the *Costly changes widget* are in need for further improvement before they can be considered useful.

Overall the respondents are able to draw the conclusions we expected them to draw from the different cases of the widgets. This means that we have managed to create a certain insight into the data of KE-chain which was the initial goal of KE-board. It is great to see that the reaction of most respondents is that they even want to dive deeper into the data that KE-board shows. This indicates that there is room for more KPI widgets and a future for KE-board.

Furthermore we may conclude that the consultants of KE-chain consider KE-board to be an intuitive and pleasant tool with a great user experience.

## 7.4. FEEDBACK FROM KE-WORKS

As part of the acceptance of KE-board to KE-works. We have asked the CTO (Chief Technical Officer) Stefan van der Elst and our coach Tijn Hendrich to evaluate our delivered product and to give their opinion on our participation within the development team.

“The KE-board module provides our workflow management platform KE-chain a competitive advantage by making advanced principles of Lean Engineering available to its users. KE-board is a project based dashboard with an overview of the progress and points of interest within a project. The dashboard provides users with a set of smart widgets that provide insight into important metrics within a project. These so-called Key Performance Indicators are based on (advanced) principles of the Lean Engineering. This allows the the users of KE-chain to lift the Operational Excellence of projects to a higher level. Within the manufacturing / engineering industry, there are currently no comparable software packages available that allow steering of projects based on the metrics in that KE-board offers.

The approach of the development of KE-board was of such a high level that the results can be deployed almost instantly within KE-chain. In particular, the high code quality, good coverage of unit tests on the back-end code and the flexibility of the layout of the dashboard are important lessons for KE-chain in general.”

*Stefan van der Elst, CTO at KE-works*

“KE-board has become starting point for incorporating business intelligence within KE-chain, tailored to individual users. In the future, KE-works will integrate KE-board into the main KE-chain application to make it commercially available and use KE-board as an additional selling argument. Besides streamlining product development processes and decreasing their lead time, KE-chain generates data that gives insight on the KPIs of product development processes in real time during use of the application. In the current engineering practice, this level of insight is not achieved, or is achieved infrequently (e.g. using monthly reports). KE-board has helped to unlock this data allowing KE-chain users to plan their next steps and to incorporate the principles of lean manufacturing also to product engineering. To enhance this ability further, KE-works will extend number of available widgets besides the “must have” ones implemented for the current KE-board project.

In general, Michiel and Sjoerd set-up and maintained a well-structured approach and planning. During the execution of their project it was clear to KE-works which deliverables they were going to hand in at what time. At times, Michiel and Sjoerd had a tendency to invest too much effort into a single, contained technical issue, but this occurred rarely and did not keep them from producing good results. They were able to quickly define and grasp functional concepts involved in KE-board, while accounting for the re-

quirements they had identified. Michiel and Sjoerd worked autonomously, and took the initiative for asking questions and feedback whenever they needed in order to get ahead or resolve issues. This also holds for the (preparation of) the interviews they held with KE-works customers to identify the needs for the KE-board dashboard. All of the above led to a dashboard of high quality that KE-works will be able to integrate in operational practice! Finally, apart from the functional and technical results of the projects, Michiel and Sjoerd managed to become valued and appreciated development team members in a short time.”

*Tijl Hendrich, Knowledge Engineer at KE-works, Supervisor of KE-board*

## 7.5. CONCLUSION

For this bachelor project an integrated KPI dashboard called KE-board has been designed and implemented into the existing KE-chain application of KE-works. The users of KE-chain and the consultants at KE-works have been interviewed and questioned to get to know which measures in KE-chain are most important. From these measures several Key Performance Indicators have been composed which are considered to contribute to the optimization of the product development process. After that KE-board and seven widgets have been completely designed and implemented.

The initial goal of KE-board was to give the users of KE-chain a real-time overview of the status of a project within KE-chain to eventually be able to optimize the product development process. The KPI's have been composed of measures which have been created and graded according to the Lean methodology and the experience of the users of KE-chain and the consultants of KE-works. Therefore, we believe the KPI's with the corresponding purposes and goals are contributing to the optimization of the product development process. To make sure the KPI's achieve the goals we've set for them, they have been reviewed by the consultants of KE-works. In general they managed to draw the right conclusions from the KPI's that were presented to them, which indicates that the KPI's give insight in the available data of KE-chain and achieve their goals.

Furthermore KE-board has been completely reviewed by KE-works and has shown to be an intuitive tool which allows its users to get an overview of the projects in a glance. In addition, it provides numerous ways of acting upon the data presented.

Both KE-board and the KPI's have been reviewed positively, our research goal has been achieved and the requirements for KE-board have been met. Consequently, we may conclude that the goal of KE-board has been achieved and the project has been completed successfully.



# 8

## RECOMMENDATIONS

In this chapter we discuss the recommendations we have for the people that are going to work on KE-board in the future and KE-works in general. First we discuss the recommendations concerning the further implementation of KE-board after which we continue with the recommendations regarding the development methodology at KE-works.

### 8.1. KE-BOARD

Previously in section 7.3.3 we have discussed the evaluation of KE-board as a result from the interviews. In this section we have formulated a number of recommendations to solve some of the issues that have been brought up.

On the issue of that it is unclear what information is displayed in a particular widget, we recommend to update the titles of the widgets to include this information. For instance the 'Hot tasks' widget may then carry the title: 'Hot tasks: displaying highly-impacted tasks, incorporating changes from the past 7 days.' This way the user immediately knows what is being displayed. On top of this, when a user has multiple widget of the same type on his dashboard, but with different properties that would become instantly clear from the title.

To solve the problem of the missing link between the different widgets that show a list of tasks in some way, we recommend to include such a common field. The due date of a task is a great candidate for such a field because it is a great indication of whether a task should be completed in the near future or not. By adding this field, it can be easier to keep an eye on the project as a whole.

Other smaller suggestions we have include simple UI improvements such as:

- Give KE-board a more easy to find location in KE-chain, by adding it to available links and context-menu's.
- Re-clicking the help or gear icon should bring the user back to the original view of the widget.
- Insert newly added widgets on the columns in an alternating fashion, so the widgets are spread out evenly.
- Add the ability to add and remove columns to the dashboard to increase customizability.
- Underlining text that can be clicked as a link when a user hovers their mouse over them indicates that a user can click them to go somewhere. Also a tooltip can be shown to hint a user of

this kind of behaviour.

## 8.2. DEVELOPMENT

In this section we shortly explain some of the aspects of the development process at KE-works that can be improved, although we can state that the approach is very structured and professional, these are our remarks.

Firstly, we have a recommendation to attempt to reduce the length of the sprint-kick off. This may be done by having the different stakeholders decide beforehand in a meeting what stories must be included, then the developers can give their opinion on how much time they think those stories will take. This way the developers do not have to concern themselves with decisions that are out of their hands, letting them focus on estimating the weight of the stories whilst reducing the length of this meeting.

Secondly, another crucial part of the scrum sprint is the review; the members of the team attempt to identify the things that went well and didn't went well during the sprint. When we started our project, these sprint reviews were not documented in any way. However, during the project KE-works decided that is was better to document these reviews, and let every member write down their opinions. We think this is a great move to prevent the same mistakes from happening again and to keep improving the process.

Finally, one of the issues that we struggled with during the KE-board project, is that it was very hard to test the front-end in an automated way. By automatically testing (just like in the back-end) you can more easily test components of the system, but also test if certain behaviour interacts with the system as expected. Using a front-end testing framework can really help to create the tests and execute them. Our recommendation is to setup a front-end framework and start building tests for the components that are most used. This way bugs can be found more easily and by integrating the front-end tests into the already used continuous integration platform, bugs can be prevented from slipping into the system. Altogether this provides a way to increase the quality and stability of KE-chain.

# BIBLIOGRAPHY

- [1] T. Ohno, *The Toyota Production System*, Ph.D. thesis, Monterey Institute of International Studies (1987).
- [2] M. Simms, *The seven wastes in engineering design*, (2007).
- [3] A. Neely, *The evolution of performance measurement research: developments in the last decade and a research agenda for the next*, International Journal of Operations & Production Management **25**, 1264 (2005).
- [4] K. F. Cross and R. L. Lynch, *The “smart” way to define and sustain success*, National Productivity Review **8**, 23 (1988).
- [5] D. P. Keegan, R. G. Eiler, and C. R. Jones, *Are your performance measures obsolete?* Management accounting **70**, 45 (1989).
- [6] R. S. Kaplan and D. P. Norton, *The balanced scorecard*, Vol. 6 (Harvard Business School Press Boston, 1996).
- [7] B. List and K. Machaczek, *Towards a corporate performance measurement system*, in *Proceedings of the 2004 ACM symposium on Applied computing* (ACM, 2004) pp. 1344–1350.
- [8] C. Lohman, L. Fortuin, and M. Wouters, *Designing a performance measurement system: a case study*, European Journal of Operational Research **156**, 267 (2004).
- [9] A. Neely, H. Richards, J. Mills, K. Platts, and M. Bourne, *Designing performance measures: a structured approach*, International journal of operations & Production management **17**, 1131 (1997).
- [10] Balsamiq, rapid effective and fun wireframing software. <http://balsamiq.com/>.
- [11] The Python programming language. <http://jenkins-ci.org/>.
- [12] The Django framework. <https://www.djangoproject.com/>.
- [13] Celery Project, <http://www.celeryproject.org/>.
- [14] JavaScript framework for building rich desktop web applications. <http://www.sencha.com/products/extjs/>.
- [15] D. McCarthy and C. Crane, *Comet and Reverse Ajax: The Next-Generation Ajax 2.0* (Apress, 2008).
- [16] Example of Ext JS with websync. <http://www.frozenmountain.com/websync/learn/demos>.
- [17] Ext JS WebSocket is an extension to handle and use the HTML5 WebSocket with ExtJS. <https://github.com/wilk/ExtJS-WebSocket>.
- [18] Scrum is a framework for developing and sustaining complex products. <http://www.scrum.org/>.

- [19] JIRA issue tracker. <https://www.atlassian.com/software/jira>.
- [20] Confluence information sharing and collaboration. <https://www.atlassian.com/software/confluence>.
- [21] Subversion open source version control system (SVN). <http://subversion.apache.org/>.
- [22] Jenkins CI, An extendable open source continuous integration server. <https://www.python.org/>.
- [23] Pylint, Code analysis for Python. <http://www.pylint.org/>.
- [24] PEP8, Style guide for python code. <http://legacy.python.org/dev/peps/pep-0008/>.
- [25] JSLint, the Javascript code quality tool. <http://www.jshint.com/>.
- [26] Selenium, Automated web user interface testing. <http://docs.seleniumhq.org/>.



## KE-CHAIN NON-FUNCTIONAL REQUIREMENTS

1. Usability
  - (a) KE-chain users should not require specialized knowledge in IT to work with the system.
  - (b) Naming should be free of IT jargon.
  - (c) A manual shall be supplied along with the application.
2. Accessibility
3. Availability
  - (a) KE-chain shall have an uptime of 99%, assuming that all deployment requirements have been met.
4. Backup
  - (a) The application shall automatically make a backup of the database at least every 24 hours.
5. Capacity
  - (a) The application shall hold at once a maximum of:
    - i. 10 projects
    - ii. 10 unique users simultaneously
    - iii. 100.000 data records in a work sheet
    - iv. 100 attributes (columns) per record in a work sheet
    - v. A maximum total of 10.000.000 fields in a work sheet
6. Compliance
  - (a) Compliance to functional requirements shall be proven with predefined tests as specified in the Master Test Plan.
  - (b) Compliance to non-functional requirements shall be asserted with the following tests:
    - i. Stress test

- ii. Performance test
  - iii. Load test
- (c) These tests are specified in section 5.1
- 7. Efficiency (resource consumption for given load)
- 8. Effectiveness (resulting performance in relation to effort)
- 9. Extensibility (adding features, compatibility)
- 10. Portability
  - (a) KE-chain shall be accessible through a VPN from any computer.
- 11. Mean Time To Recovery (MTTR)
  - (a) The MTTR shall be covered in the SLA between the Customer and KE-works.
- 12. System requirements
  - (a) Client side:
    - i. Operating system: Windows 7 32 / 64 bits
    - ii. Browser: IE8+/Firefox 3.6+/Chrome
    - iii. Minimum screen resolution of 1280x720
    - iv. RAM: 1GB
    - v. CPU: Dualcore 1.6GHz
    - vi. Network connection to host server
  - (b) Server side: The host server configuration and layout shall be as described in "KE-chain on RHEL6.x dedicated server grade hardware"
- 13. Security
  - (a) The security of the application shall be covered by the Customer ICT organisation.

# B

## FOLLOW-UP QUESTIONS

### B.1. PARTICIPANT 1

You scored the following measures as important where others did not, could you briefly explain why?

- Completion of project (over time) - *Answer: this simply what you want to look at when you are at the highest level of monitoring. Do we make progress, how fast and when are we done?*
- Completion of project per department - *Answer: this is not useful information for employees, because it would create competition and that should never be the case. However, for a project manager or at even higher management levels this could be a valuable KPI.*
- Lead time of project - *Answer: again as a project manager this is very valuable information.*
- Standards most used per task - *Answer: could be very useful, but only over multiple projects.*
- All KPI's regarding to number of clicks - *Answer: not per se the number of clicks, but mainly the number of mutations or work that has been done by users of the system. Nowadays it is difficult for companies to see how to manage and especially on which part of the processes, this could be helpful to see where most work or mutations seem to happen.*

You did not score the following measures as important where others did, could you briefly explain why?

- Number of tasks in progress (over time) - *Answer: in my opinion there are always many tasks in progress, so this won't give much value.*
- Number of unchanged fields per task - *Answer: this could be of value when I am reviewing my current system, but not for me as a user of the system. It is no actual status information of a task, project.*

### B.2. PARTICIPANT 2

You scored the following measures as important where others did not, could you briefly explain why?

- Number of tasks in progress (over time) - *Answer: gives you an idea how the process is approached: head on, all at once or in a more linear fashion.*

- Number of unchanged fields per task - *Answer: if a field does not change in a task, this is a possible indication that the field doesn't belong there or is even redundant.*
- Standards most used per user - *Answer: what standards are "popular"? Enables you to see whether a "custom value" is used by just the user who created it or by multiple users, giving rise to the question: should this become a standard?*

You did not score the following measures as important where others did, could you briefly explain why?

- Working time per task (so the estimated time that someone has actually worked on a task) - *Answer: working time of a human user can become deceptive once automation comes into play.*
- Number of changes per project (over time) - *Answer: for a statistic which is nice to put on your office wall, I'd take progress over time. This one is nice for sales though.*
- Standards most used per task - *Answer: actually, this one is quite alright (together with standards per user). Better would even be use of standards across the whole product.*
- Task with most succeeding tasks waiting - *Answer: KE-chain configuration does.*
- Succeeding tasks that have not been started yet - *Answer: In KE-chain, going through the process linearly while finishing all preceding tasks before continuing is not a prerequisite. Furthermore, you can already deduce this KPI by checking for 0% progress in the workflow. I think you can do more with due dates.*
- All KPI's regarding to number of clicks - *Answer: although this is interesting from UX perspective, for specific "click paths", to a user number of clicks holds no specific information on what he has done or what he should be doing.*
- Relative number of changes per task wrt other tasks - *Comparing tasks this way tricks you into thinking that this is a "fair comparison" while possibly it isn't. Showing absolute numbers on changes gives you more information (which tasks are bristling with change) and is easier to interpret.*

### B.3. PARTICIPANT 3

You scored the following measures as important where others did not, could you briefly explain why?

- Number of changes per project (over time) - *Answer: gives a nice indication of the maturity of the project.*
- Relative number of changes per task w.r.t. other tasks - *Answer: gives an indication of which tasks are most changing, which could be of value.*
- Number of tasks in progress (over time) - *Answer: keep the number of tasks low, this could be an important indicator. also gives an idea of the maturity of the process.*
- Working time per task (so the estimated time that someone has actually worked on a task) - *Answer: very important. if you know that some tasks cost a lot of working time you could choose for more resources or automation.*
- Task with most succeeding tasks waiting - *Answer: good indication to see which tasks need to be worked on.*



- Succeeding tasks that have not been started yet - *Answer: you could see it in the workflow, but an overview could be of value on the dashboard as well.*
- All KPI's regarding to number of clicks - *Answer: same as working time, could give an indication of the amount of effort/work that has been needed for this tasks, could lead to task improvement or other use of resources.*

You did not score the following measures as important where others did, could you briefly explain why?

- Completion of project (over time) - *Answer: completion of a project is somewhat important, but in my opinion less important than completion of a task, since then you can see where has to be worked in more detail.*

# C

## GENERAL PLANNING PER WEEK

### **Week 1 - Research**

- Planning
- Start research phase
- Establish project requirements
- Find relevant literature
- Familiarize with company tools
- Deliverables: Plan of action

### **Week 2 - Research**

- Interview client
- Finalize requirements and constraints
- Summarize literature
- Deliverables: Research report

### **Week 3 - Implementation - Design**

- Design functional architecture
- Design user experience
- Deliverables: Design of KE-board

### **Week 4 - Implementation Sprint 1**

- Start of Sprint 1
- End of Sprint 1

### **Week 5 - Implementation Sprint 2**

- Start of Sprint 2
- Demo
- Deliverables: Sprint Evaluation Report

### **Week 6 - Implementation Sprint 2**

- End of Sprint 2
- First code check at SIG

### **Week 7 - Implementation Sprint 3**

- Start of Sprint 3
- Deliverable: Draft Report
- Demo

### **Week 8 - Implementation Sprint 3**

- End of Sprint 3
- Process code check feedback
- Deliverable: Sprint Evaluation Report

### **Week 9 - Evaluation**

- Demo
- Final code check at SIG
- Deliverable: Sprint Evaluation Report

### **Week 10 - Evaluation**

- Finish report
- Optional: implement SIG feedback
- Prepare presentation
- Deliverables: Evaluation Report, Final Report

### **Week 11 - Presentation**

- Presentation
- Deliverables: Presentation

# D

## USE CASES

### VIEW DASHBOARD

<b>Actors</b>	Project Manager / Engineer
<b>Entry conditions</b>	None
<b>Activity diagram</b>	User navigates to the dashboard.
<b>Exit conditions</b>	Dashboard has been displayed.
<b>Exceptions</b>	If it is the first time the user visits the dashboard, a dashboard lay-out according to the role of the user is generated.

Table D.1: Use case: view dashboard

### ADD KPI WIDGET TO DASHBOARD

<b>Actors</b>	Project Manager / Engineer
<b>Entry conditions</b>	The user has navigated to the dashboard.
<b>Activity diagram</b>	The user: <ol style="list-style-type: none"><li>1. Clicks the "add KPI widget" button.</li><li>2. Selects the desired KPI widget.</li><li>3. Clicks "save".</li></ol>
<b>Exit conditions</b>	The selected KPI widget is added to the dashboard and if needed minor properties have to be set in the widget before it functions as described in Table D.4. The state of the dashboard has been saved. The KPI widget can be moved across the board by the user to arrange.
<b>Exceptions</b>	If the user clicks "Cancel" the system returns to the dashboard..

Table D.2: Use case: add KPI widget to dashboard

## REMOVE KPI WIDGET FROM DASHBOARD

<b>Actors</b>	Project Manager / Engineer
<b>Entry conditions</b>	The user has navigated to the dashboard and wants to remove KPI widget X.
<b>Activity diagram</b>	The user: <ol style="list-style-type: none"> <li>1. Clicks the "remove" button on KPI widget X.</li> <li>2. Clicks "yes" on the "are you sure" alert.</li> </ol>
<b>Exit conditions</b>	KPI widget X is removed from the dashboard and the surrounding KPI widgets are arranged so that it looks like KPI widget X was never there. The state of the dashboard has been saved.
<b>Exceptions</b>	When the user clicks "no" on the "are you sure" alert, KPI widget X is not removed.

Table D.3: Use case: remove KPI widget from dashboard

## EDIT PROPERTIES OF KPI WIDGET

<b>Actors</b>	Project Manager / Engineer
<b>Entry conditions</b>	The user has navigated to the dashboard and wants to edit the properties of KPI widget X.
<b>Activity diagram</b>	The user: <ol style="list-style-type: none"> <li>1. Clicks the "properties" button on KPI widget X.</li> <li>2. Adjusts the properties of KPI widget X.</li> <li>3. Clicks "save".</li> </ol>
<b>Exit conditions</b>	The properties of KPI widget X are saved and the adjusted KPI widget is shown on the dashboard. The state of the dashboard has been saved.
<b>Exceptions</b>	When the user enter wrong information, an error message is presented. When the user clicks "cancel", the properties window is closed and the system returns to the widget.

Table D.4: Use case: edit properties of KPI widget

## DRAG KPI WIDGETS TO ARRANGE DASHBOARD

<b>Actors</b>	Project Manager / Engineer
<b>Entry conditions</b>	The user has navigated to the dashboard.
<b>Activity diagram</b>	The user: <ol style="list-style-type: none"> <li>1. Clicks and holds the top bar of any widget in the dashboard.</li> <li>2. Moves the KPI widget to the desired location. (column and position in column)</li> <li>3. While moving, the KPI widget snaps into each position it could be placed in.</li> <li>4. Releases to place the widget into the desired place.</li> </ol>
<b>Exit conditions</b>	The selected KPI widget is moved to another location in the dashboard. The state of the dashboard has been saved.
<b>Exceptions</b>	The user moved the KPI widget back to its original place. In this case the widget is not moved.

Table D.5: Use case: Drag KPI widgets to arrange dashboard

# E

## SPECIFICATION OF KPI's

<b>Title</b>	Project progress over time
<b>Purpose</b>	Inform of current state of project and amount of changes being introduced in relation to the completion.
<b>Role</b>	Engineer
<b>Relates to</b>	1.1, 1.2, 1.3, 1.4, 1.5, 2.1, 2.3, 2.4, 2.5
<b>Target</b>	Improve project development.
<b>User-defined properties</b>	[unit] Day, week, month
<b>Formula</b>	Calculate completion of project per [unit], calculate number of changes per [unit] and show it over time from project start date until now
<b>Refresh rate</b>	Half hour
<b>Who acts on the data</b>	Project Manager or Engineer
<b>What do they do</b>	See if the project is on track and look for problem sources if problems have occurred.
<b>Visualization method</b>	Bar chart and line graph

Table E.1: Project progress over time

<b>Title</b>	Critical tasks
<b>Purpose</b>	Give insight in tasks that are critical and should be worked on.
<b>Role</b>	Engineer
<b>Relates to</b>	1.1, 3.1, 5.3
<b>Target</b>	Reduce lead time of tasks and project.
<b>User-defined properties</b>	[num] Maximum number of displayed tasks.
<b>Formula (<i>F</i>)</b>	Calculate for all tasks that are not finished yet: their completion, lead time. Select tasks that are almost due/overdue and limit to [num].
<b>Refresh rate</b>	Short (half hourly)
<b>Who acts on the data</b>	Engineer
<b>What do they do</b>	Work on the critical tasks and make sure they get completed.
<b>Visualization method</b>	Grid with tasks and specified properties. Color coding of the lines in the grid indicates the criticality.

Table E.2: Critical tasks

<b>Title</b>	Hot Tasks
<b>Purpose</b>	Give insight in changes that have been made.
<b>Role</b>	Engineer
<b>Relates to</b>	2.1, 2.6, 2.9
<b>Target</b>	Minimize outdated information.
<b>User-defined properties</b>	[num] Maximum number of displayed tasks.
<b>Formula</b>	Calculate the number of changes for all tasks of the user. Calculate the impact of those changes for all users. Sort tasks according to these measures and limit to [num]
<b>Refresh rate</b>	5 minutes
<b>Who acts on the data</b>	Engineers
<b>What do they do</b>	Make sure the data entered is still valid.
<b>Visualization method</b>	Grid with tasks with number of changes and the impact of those changes specified with some color indication.

Table E.3: Hot Tasks

<b>Title</b>	Open Tasks
<b>Purpose</b>	Give insight in tasks that could or are being worked on.
<b>Role</b>	Engineer
<b>Relates to</b>	1.1, 3.2
<b>Target</b>	Reduce lead time of tasks.
<b>User-defined properties</b>	[num] maximum number of displayed tasks.
<b>Formula</b>	Calculate for all tasks that have been worked on? their completion of input values. Select tasks that have the highest completion of inputs and limit to [num].
<b>Refresh rate</b>	Short (half hour)
<b>Who acts on the data</b>	Engineer
<b>What do they do</b>	Start working on tasks that have complete input values.
<b>Visualization method</b>	Grid with tasks and specified the completion of their inputs.

Table E.4: Open Tasks

<b>Title</b>	Tasks in progress
<b>Purpose</b>	To identify which tasks are in progress and are critical.
<b>Relates to</b>	1.1, 3.1, 3.3
<b>Target</b>	Prevention of overdue tasks.
<b>User-defined properties</b>	[num] maximum number of displayed tasks
<b>Formula</b>	Calculate of all tasks: input and output completion, working time, lead time and float (due date minus now). Sort the tasks according to these measures and limit by [num].
<b>Refresh rate</b>	Half hour
<b>Who acts on the data</b>	Project Manager
<b>What do they do</b>	Tell Engineers to focus on certain tasks.
<b>Visualization method</b>	Grid with tasks and specified the input and output completion, working time, lead time and float with color indications.

Table E.5: Tasks in progress

<b>Title</b>	Tasks not started
<b>Purpose</b>	Identify tasks that could be, but are not worked on.
<b>Relates to</b>	1.1, 1.6, 3.5, 3.6
<b>Target</b>	Reduce waiting time of tasks.
<b>User-defined properties</b>	[num] maximum number of displayed tasks
<b>Formula</b>	Calculate all tasks that have not been worked on yet, but have complete inputs, sorted by waiting time, limited by [num].
<b>Refresh rate</b>	Short (half hour)
<b>Who acts on the data</b>	Engineer
<b>What do they do</b>	Start work on tasks that can be started.
<b>Visualization method</b>	Grid with tasks and specified the completion of their inputs and the waiting time

Table E.6: Tasks not started

<b>Title</b>	Costly changes
<b>Purpose</b>	Identify changes that have high impact on project.
<b>Relates to</b>	2.9
<b>Target</b>	Prevent changes that have high impact in the future.
<b>User-defined properties</b>	[threshold] Minimum impact factor of changes to display. [time] The amount of time to go back to.
<b>Formula</b>	For all changes that occurred later then [time], calculate the number of times an element e occurs as input for other tasks, for each task multiplied by the progress of that task and its succeeding tasks. Then sort them by impact and limit to the changes that are above [threshold]
<b>Refresh rate</b>	Medium (daily)
<b>Who acts on the data</b>	Project manager
<b>What do they do</b>	Investigate why the changes happened and make sure it will not happen again.
<b>Visualization method</b>	Donut graph where each change represents a chunk, highlightable. Or column chart over time.

Table E.7: Costly changes



# F

## SIG FEEDBACK (DUTCH)

### F.1. AANBEVELINGEN

De code van het systeem scoort bijna 4 sterren op ons onderhoudbaarheidsmodel, wat betekent dat de code bovengemiddeld onderhoudbaar is. De hoogste score is niet behaald door een lagere score voor Duplicatie en Unit Size.

Voor Duplicatie wordt er gekeken naar het percentage van de code welke redundant is, oftewel de code die meerdere keren in het systeem voorkomt en in principe verwijderd zou kunnen worden. Vanuit het oogpunt van onderhoudbaarheid is het wenselijk om een laag percentage redundantie te hebben omdat aanpassingen aan deze stukken code doorgaans op meerdere plaatsen moet gebeuren. In dit systeem is er bijvoorbeeld duplicatie te vinden in het genereren van een 'span' met juiste kleur style binnen de verschillende portlets. Het is aan te raden om dit soort duplicaten op te sporen en te verwijderen.

Voor Unit Size wordt er gekeken naar het percentage code dat bovengemiddeld lang is. Het opsplitsen van dit soort methodes in kleinere stukken zorgt ervoor dat elk onderdeel makkelijker te begrijpen, te testen en daardoor eenvoudiger te onderhouden wordt. Binnen de langere methodes in dit systeem, zoals bijvoorbeeld de 'notifyOver'-methode binnen 'PortalDropZone.js', zijn aparte stukken functionaliteit te vinden welke ge-refactored kunnen worden naar aparte methodes. Commentaarregels zoals bijvoorbeeld '// determine column' en '// find insert position' zijn een goede indicatie dat er een autonoom stuk functionaliteit te ontdekken is. Het is aan te raden kritisch te kijken naar de langere methodes binnen dit systeem en deze waar mogelijk op te splitsen.

Over het algemeen scoort de code bovengemiddeld, hopelijk lukt het om dit niveau te behouden tijdens de rest van de ontwikkelfase. De aanwezigheid van test-code voor het Python gedeelte is in ieder geval veelbelovend, hopelijk zal het volume van de test-code ook groeien op het moment dat er nieuwe functionaliteit toegevoegd wordt.

### F.2. HERMETING

In de tweede upload zien we dat het codevolume is gegroeid, en dat de score voor onderhoudbaarheid ongeveer gelijk is gebleven.

De duplicatie is in de tweede upload verder toegenomen, met name in de JavaScript-code. De configuratie van de verschillende widgets die jullie in de user interface gebruiken bevat een grote ho-

eveelheid herhaling, als je de gedeelde configuratie wilt aanpassen zul je onnodig veel werk moeten doen (omdat dezelfde parameter 4 keer ingesteld wordt).

Bij Unit Size zien we dat in PortalDropZone.js nog steeds gedeelten van methodes met commentaar aangegeven worden, in plaats van er aparte methodes van te maken. Dit maakt hergebruik moeilijker, en het schrijven van unit tests wordt hierdoor moeilijker. Dit voorbeeld staat niet op zichzelf, jullie hebben bovengemiddeld veel lange methodes.

De testcode voor het Python-gedeelte ziet er nog steeds goed uit, het is ook positief om te zien dat de hoeveelheid testcode is toegenomen. De verhouding productiecode/testcode zit nu dicht in de buurt van het ideaal van 1:1. Dit is een groot contrast met het JavaScript-gedeelte, waar helemaal geen tests voor zijn. Hoe controleren jullie dan of deze code het doet? Je kunt met de hand testen, maar dat is veel meer werk en moeilijk herhaalbaar.

Uit deze observaties kunnen we concluderen dat de aanbevelingen van de vorige evaluatie nauwelijks zijn meegenomen in het ontwikkeltraject.

# G

## ORIGINAL PROJECT DESCRIPTION

Design and develop a dashboard in KE-Chain that provides out-of-the box analytic's for workflows in KE-chain, our existing web-application. The Key Performance Indicators in this dashboard should support the measurement and validation of the SOL state of the development process as defined in KE-chain per project. Similar KPIs and KPI dashboard widgets are applicable to analyze the application and use of engineering library components, such as design solutions, rules and requirements. The different end-users of KE-chain should be facilitated with a role-based dashboard: A manager would like to see process related KPI's; an engineer would like to monitor the tasks that are assigned to him. The students are tasked with the job to analyze the system and its user and determine what KPI's to include. The implementation of KE-board should visualize these features in an optimal way and be suited to the different users.

# H

## WIDGET EVALUATION QUESTIONNAIRE

In this Appendix the set-up and the results of the questionnaire are included.

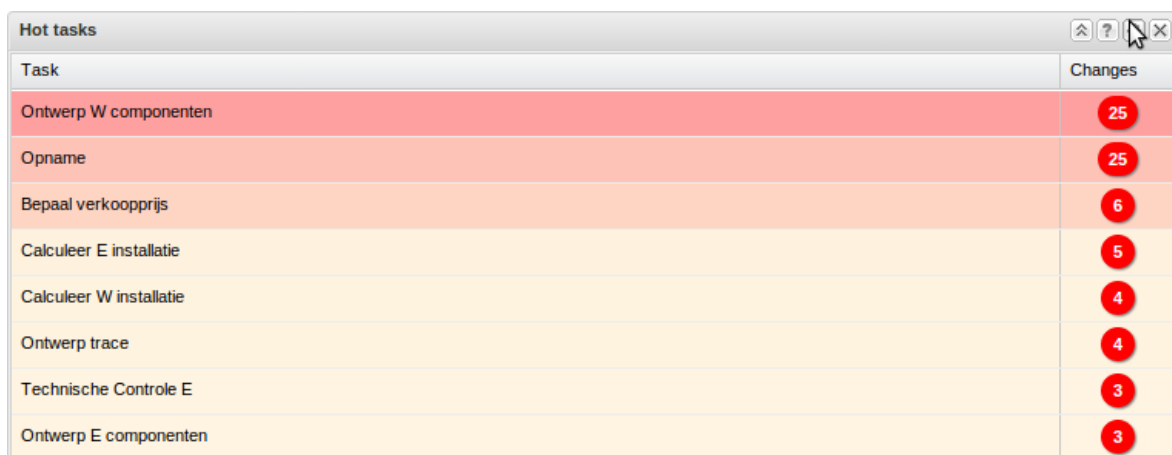
### H.1. HOT TASKS WIDGET

The widget shows you which tasks are "hot", this means that they either have a lot of changes or have high impact changes. The tasks are sorted by the impact of the changes. Impact is calculated from the number of highly progressed tasks that are begin affected by the changes made.

Perspective: engineer

Purpose: give insight in changes that have been made.

Target: minimize outdated information.



Task	Changes
Ontwerp W componenten	25
Opname	25
Bepaal verkoopprijs	6
Calculeer E installatie	5
Calculeer W installatie	4
Ontwerp trace	4
Technische Controle E	3
Ontwerp E componenten	3

### WHAT ARE THE CONCLUSIONS YOU CAN DRAW FROM THE WIDGET ABOVE?

- I'm not sure. Red is telling me to do ontwerp W componenten, but other tasks have much more changes, which indicates that they are "hot".
- I understand the first task is key, but I dont understand why.

I would like to differentiate between; 1) number of successive tasks 2) minimum impact, maximum impact & estimated impact

For me this is now a black box, I don't understand the cause of the order, which is fundamental for understanding the need.

Second: I could see a case in which the HOTNESS is adapted manually for instance by a project manager

- The sorting in the widget is broken. The only information I see is the # of changes; is somehow seems to use other information for the sorting. I would like to see this, by adding a column with for example the # of tasks impacted
- Ontwerp W componenten is a task which has much impact on other tasks in the process and thereby eventhough there is only one single change this task is probably more important then the tasks with a large nr of changes
- ontwerp w componenten has few changes, but they have enormous impact, contrary to the changes in calculeer W installatie.
- "Ontwerp W componenten has 1 change. Opname has 1 change. Opname and bepaal verkoopprijs have the same impact. The ontwerp W componenten change has a very high impact The impact of a change in ontwerp trace has a lower impact

Task	Changes
Ontwerp W componenten	1
Opname	1
Bepaal verkoopprijs	6
Calculeer E installatie	5
Calculeer W installatie	25
Ontwerp trace	25
Technische Controle E	3
Ontwerp E componenten	3

#### WHAT ARE THE CONCLUSIONS YOU CAN DRAW FROM THE WIDGET ABOVE?

- Go "Ontwerp W componenten"!
- No prioritisation other then the amount of changes. I would expect that the first task in the flow is also on top of this list. I would understand this better if the dependencies would be somehow visible.
- Again, this widget is broken, this time its the colorcoding. Is this widget perhaps not properly covered by unit tests? ;)
- Changes in all tasks have the same impact, therefor the task with the most changes is most important
- the high number of changes don't have a lot of impact: -> the impact per change is lower for ontwerp W componenten, so I have to look at Ontwerp E componenten.
- Ontwerp W componenten has 25 changes, opname too, etc. etc.
- They all have the same impact. the tasks are sorted on the number of changes

Hot tasks	
Task	Changes
Ontwerp W componenten	25
Opname	25
Bepaal verkoopprijs	6
Calculeer E installatie	5
Calculeer W installatie	4
Ontwerp trace	4
Technische Controle E	3
Ontwerp E componenten	3

### WHAT ARE THE CONCLUSIONS YOU CAN DRAW FROM THE WIDGET ABOVE?

- Go "Ontwerp W componenten"! Clearly there is some serious work to be done there.
- Combination of the above.
- Correct! sorted on # of changes. However, if the widget should sort on another parameter, this is not correct. I would strongly advise to add the column with missing info
- Ontwerp W componenten has both a large nr of changes as well as a large impact on the process
- the changes in ontwerp w componenten have a lot of impact, but i cannot deduce if this is due to the high number of changes of their impact on following tasks
- the tasks are sorted on amount of changes Ontwerp W componenten has a high impact, after that opname, subsequently bepaal verkoopprijs and after that the impact is the same.

### DO YOU THINK THIS WIDGET CONTRIBUTES TO ITS PURPOSE AND HELPS TO ACHIEVE ITS GOAL? (RATE FROM 1 TO 5)

Average score: 3.66

## H.2. OPEN TASKS WIDGET

The widget shows all open tasks assigned to the engineer, which means they have either input or output progress which is higher than 0. Furthermore the widget is sorted and colored by input progress + output progress.

Perspective: engineer

Purpose: give insight in tasks that could or is being worked on.

Target: reduce lead time of tasks.

Task	Input Progress	Output Progress
Maak offerte	100%	52%
Ondertekening voor uitvoering	100%	41%
Ontwerp E verbindingen	100%	20%
Beoordeel onderlegger	100%	5%
Opname	63%	23%
Beoordeel bestek	40%	18%

WHAT ARE THE CONCLUSIONS YOU CAN DRAW FROM THE WIDGET ABOVE?

- I need to work on "beoordeel onderlegger", but I'd rather work on beoordeel bestek because of its low progress value.
  - This one is clear! Now I also see the cause of the colors inside the same widget
  - I would expect tasks that are nearly completed to be color coded green. Red = bad.
- Perhaps a little indication in the title of the widget can help to identify the color coding scheme. I would think that (without reading the text above) the tasks that have less progress might need more attention.
- Working on Beoordeel onderlegger is probably the best action. Opname is strange because the input progress is not even 2/3 but the output progress is almost 100%
  - beoordeel onderlegger is nearly completed, finish the last bits. opname is nearly completed , but still needs a lot of input maak offerte en E verbindingen have all inputs but output is lagging and i should work on this much more. others: I do not have all input available to start work
  - Beoordeel onderlegger has all the input but not the output. Opname lacks some input and probably that's the reason why not all output is generated. It is unclear based on what the colours are given. I don't know what the list is sorted on.

Task	Input Progress	Output Progress
Beoordeel onderlegger	100%	89%
Opname	63%	96%
Maak offerte	100%	52%
Ontwerp E verbindingen	98%	50%
Ondertekening voor uitvoering	20%	41%
Beoordeel bestek	40%	18%

WHAT ARE THE CONCLUSIONS YOU CAN DRAW FROM THE WIDGET ABOVE?

- There are 4 tasks for which input is complete and output could be completed as well, but isn't. Apparently Maak offerte is the most urgent one.
- Clear, although also here I would like to see the dependencies if they are there.

- This sorting is more obvious and seems correct! Lots of fields that can be filled straight away. I would expect however that the "beoordeling onderlegger" task would come first, since here there is most ground to cover
- The first four tasks are all ready to start, maak offerte has to most progress so probably can be finished first
- i can immediately start working on the top four tasks, for the others i should wait a little
- The list is sorted on input progress Probably the input progress relates to the colour of the item A lot of tasks have all the input but not all output. So someone needs to get to work.

DO YOU THINK THIS WIDGET CONTRIBUTES TO ITS PURPOSE AND HELPS TO ACHIEVE ITS GOAL? (RATE FROM 1 TO 5)

Average score: 4.00

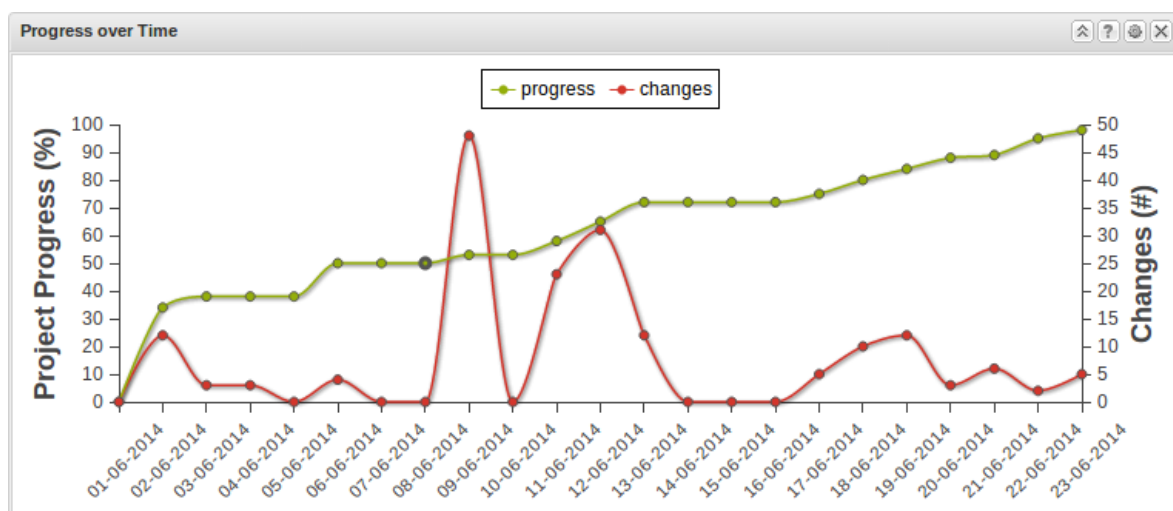
### H.3. PROGRESS OVER TIME WIDGET

This widget shows the progress and the number of changes of project over time per day.

Perspective: engineer / project manager

Purpose: inform of current state of project and amount of changes being introduced in relation to progress.

Target: improve project development.



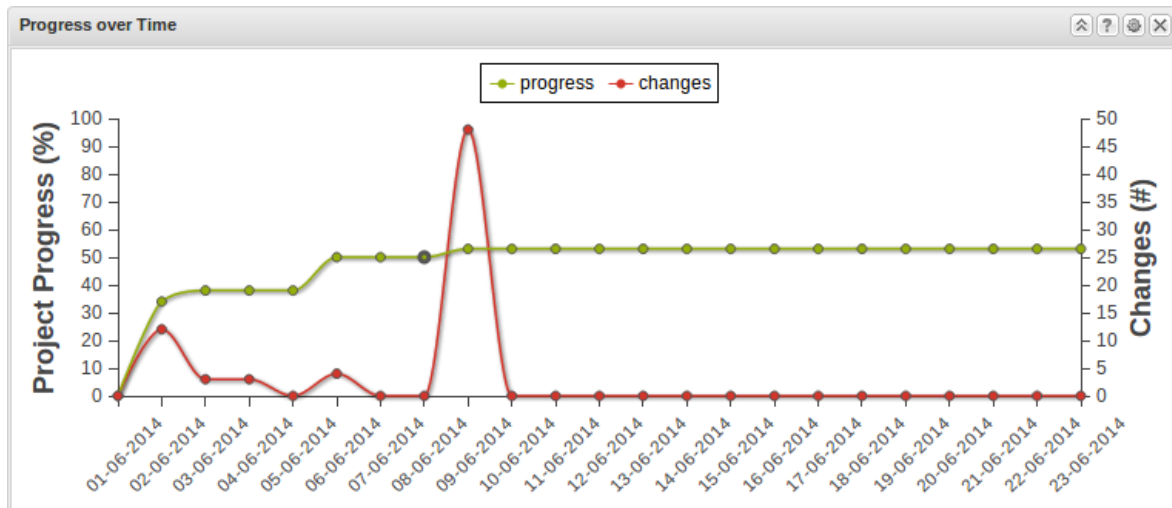
WHAT ARE THE CONCLUSIONS YOU CAN DRAW FROM THE WIDGET ABOVE?

- Something useless happened 8-6-2014. The project had a good start, but is near to death right now.
- One big change loop. No impact on progress: That means that the changes do not impact the defined information.
- Snap. I have a lazy team of engineers. That have not worked over the past 14 weeks. Holiday?



Besides, I have not so smart engineers in my team. Over 45 changes only lead to a 5% (ROM) increase in progress. that is not very efficient.

- Emergency!!! There has not been any progress since 09-06-2014
- very little changes, only on 8/6/2014. progress is flat since that date... something happened?
- I had a lot of changes at the 08-06-2014 but the changes did not lead to more progress. the first of june was a very productive day. Someone fell asleep at the 8th of june and probably banged his head on the keyboard and made a lot of changes.



WHAT ARE THE CONCLUSIONS YOU CAN DRAW FROM THE WIDGET ABOVE?

- In this project the progress is good, but work on it is highly irregular (changes graph). Maybe some optimization is possible here (i.e. resource allocation, less waiting time between change "peaks")
- I would expect that changes impact also the progress .. here all the changes do not impact the progress, but are actually not changes?!
- I would argue that changes per definition imply a change in defined information, and therefore should always impact progress." Time to grab a beer, job done! Also we have climbed up the learning curve and have become more efficient, less changes leading to more progress.
- The project progress is nicely progressing over time, but there are still some changes so the data is not frozen yet
- yes! work is being performed, upward progress. quite some activity based on the amount of changes. number of changes is reducing, which is ok.
- Changes are made frequently. Changes are made with peaks, so there is no steady amount of changes (e.g. every day one change). the progress keeps going over time.

DO YOU THINK THIS WIDGET CONTRIBUTES TO ITS PURPOSE AND HELPS TO ACHIEVE ITS GOAL? (RATE FROM 1 TO 5)

Average score: 4.66



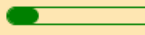


## H.4. TASKS IN PROGRESS WIDGET

This widget shows the project manager which tasks are currently in progress. Furthermore it shows the lead time (days in progress since first change), working time (number of days worked on this task) and the remaining time (number of days until due date). Tasks are sorted on remaining time and colored by the expected possibility to finish the task in time.

Perspective: project manager

Purpose: give insight in critical tasks

Target: prevent overdue tasks

Task	Progress	Lead time	Working time	Remaining Time ^
Opname	 65%	10 Days	50 Days	-25 Days
Maak offerte	 52%	13 Days	26 Days	1 Days
Ontwerp E-verbindingen	 20%	42 Days	3 Days	3 Days
Beoordeel onderlegger	 71%	15 Days	2 Days	5 Days
Ondertekening voor uitvoering	 41%	17 Days	3 Days	5 Days

### WHAT ARE THE CONCLUSIONS YOU CAN DRAW FROM THE WIDGET ABOVE?

- I need to work on opname. I find it strange that the lead time is smaller than the working time. I know lead time as the total time for a task incl. waiting.
- Remaining time is ok, but also think in dates! Need to be finished ""Tomorrow"" is clearer than ""1 Days"" and then 15 days are mentioned, maybe it is better to mention the date: ""Wednesday 9 July"" or ""two weeks"" I don't see directly the column that is related to the colors, is unclear what the colors are. By closer inspection I guess it means lead time & working time, I would expect these columns to be clearly indicated if they define the colors
- Most urgent tasks are not sorted. I think my brain always expects a widget to color code to the applied sorting. This does not match currently.


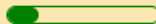



Either apply color coding to the used sorting or vice versa, sort on the parameter which has been used for color coding (this seems to be most important anyway)

- Data of Opname seems to be incorrect, working time cannot exceed lead time, right? Furthermore ontwerp maak offerte and E-verbindingen needs attention, low progress and limited remaining time
- Q: how is the possibility calculated? don't understand.

opname is critical, based on colour. but maak offerte as well, due to the low amount of remaining time. a lot to do in ontwerp e verbindingen still..

i cannot sort on expected possibility..

- Opname has a high possibility to finish the task in time. Maak offerte has only one day to go. beoordeel onderlegger has 71% progress and a lot of remaining time so probably it will finish in time. People already spend 50 days on opname. I don't get it why the lead time is 10 days and the working time is 50 days for opname.

Task	Progress	Lead time	Working time	Remaining Time ^
Maak offerte	 52%	13 Days	2 Days	1 Days
Ontwerp E-verbindingen	 20%	42 Days	3 Days	3 Days
Ondertekening voor uitvoering	 41%	17 Days	3 Days	5 Days
Opname	 65%	10 Days	50 Days	10 Days
Beoordeel onderlegger	 71%	15 Days	2 Days	15 Days

### WHAT ARE THE CONCLUSIONS YOU CAN DRAW FROM THE WIDGET ABOVE?

- Opname clearly has priority now. Still, lead time > working time, which should not be possible. Definitions for each time component need to be clear w.r.t. one another.
- Task over time.
- Houston we have a problem. I will not be able to finish my tasks in time.
- All hands on deck!! we need to finish opname right away.
- opname should be done ASAP!  
maak offerte should be done next.  
dont care about the rest
- Opname is over time. Maak offerte is a critical task, only one day to go and already spent 26 days on the task and now at 52% of the task.

### DO YOU THINK THIS WIDGET CONTRIBUTES TO ITS PURPOSE AND HELPS TO ACHIEVE ITS GOAL? (RATE FROM 1 TO 5)

Average score: 4.16

## H.5. TASKS NOT STARTED WIDGET

The tasks not started widget shows project managers which tasks have completed input, but have not been started yet. The tasks are colored and sorted by the waiting time (time not started since input progress is complete).

Perspective: project manager

Purpose: identify tasks that should be started.

Target: reduce waiting time of tasks.

Task	Waiting Time v
Maak offerte	17 Days
Ontwerp E-verbindingen	14 Days
Ondertekening voor uitvoering	1 Days

### WHAT ARE THE CONCLUSIONS YOU CAN DRAW FROM THE WIDGET ABOVE?

- Maak offerte has been idle for a long time. I should work on it.
- Be clearer for the user: ""Input ready"" is clearer then ""waiting time"".

And here the project manager would like to see who to chase. Add a link to the roles (and people) responsible for the task.

- Easy. Simple. Late.

Better late.

- Maak offerte is ready to start for 17 days now, why is no one taking this task up?
- somebody is sleeping... why is maak offerte not done yet?!
- Different colours are used. It is not clear that these tasks have 100% input already. (maybe another title). Someone is already waiting 17 days to start with maak offerte.

Tasks not Started	
Task	Waiting Time ▾
Maak offerte	2 Days
Ontwerp E verbindingen	2 Days
Ondertekening voor uitvoering	1 Days

### WHAT ARE THE CONCLUSIONS YOU CAN DRAW FROM THE WIDGET ABOVE?

- My tasks are ok. Waiting time is not excessive. I'll go and have lunch.
- No intervention action required by PM.

Here I would also think the PM would like to send reminders to the task responsible that the task should be started (think of email or at least an automatic generated message behind a button?!)

- Why does my color coding not work? All is well?
- All tasks which can start have been ready for a limited amount of time, apparently everyone is doing a good job in taking up tasks
- everything ok.
- No severe waiting times. 3 tasks can already be executed.

### DO YOU THINK THIS WIDGET CONTRIBUTES TO ITS PURPOSE AND HELPS TO ACHIEVE ITS GOAL? (RATE FROM 1 TO 5)

Average score: 4.00

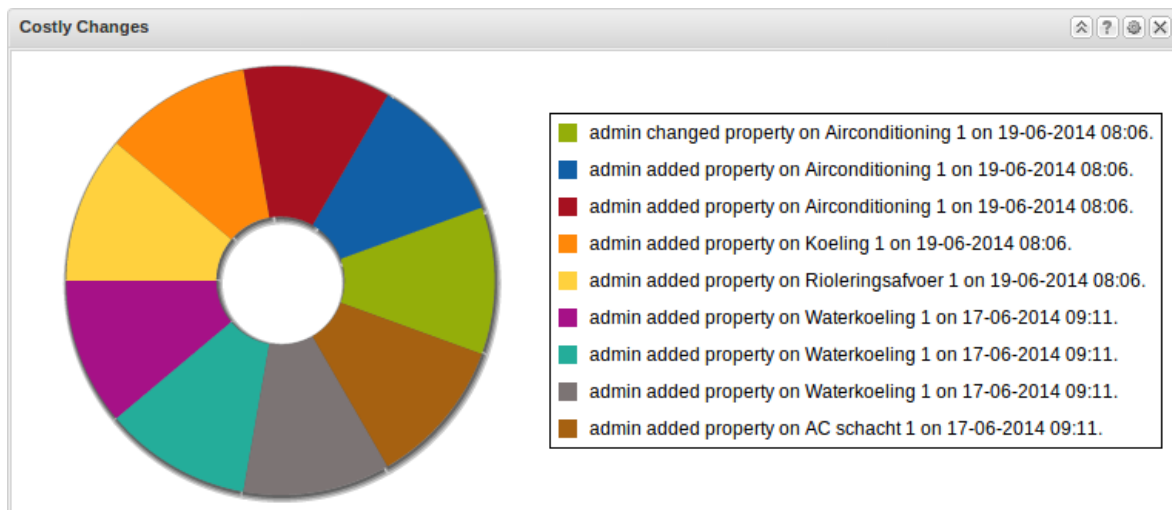
## H.6. COSTLY CHANGES WIDGET

The costly changes widget shows you what have been the most costly changes in the past seven days. In the pie chart you see the impact the change has had. In the legend you can see what changes have happened and when.

Perspective: project manager

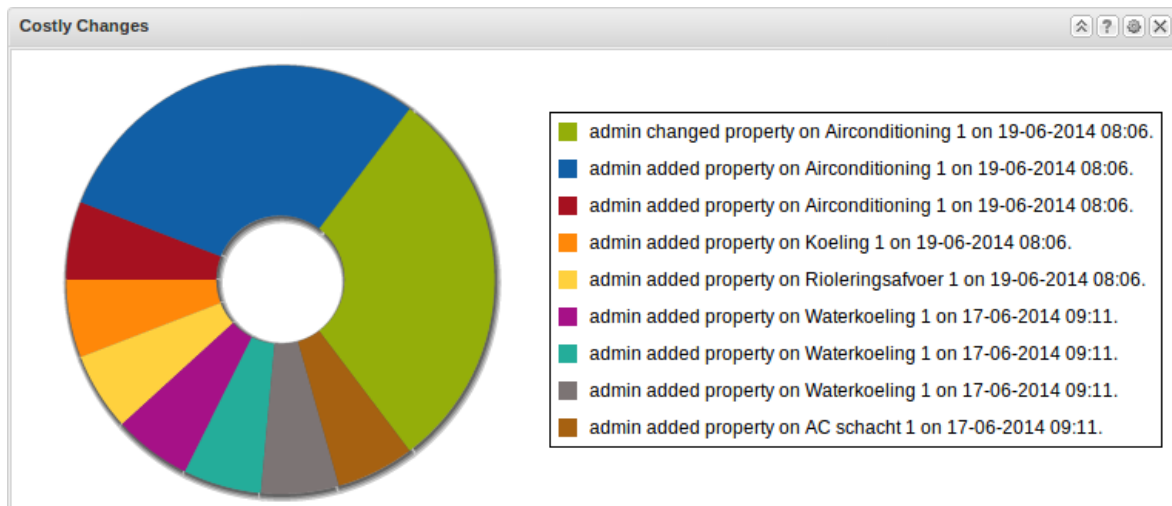
Purpose: Identify changes that have high impact on project

Target: prevent changes that have high impact in the future



#### WHAT ARE THE CONCLUSIONS YOU CAN DRAW FROM THE WIDGET ABOVE?

- None of the changes is really standing out. I don't know what to do with this information.
- Hard to grasp. Do you identify single changes or do you relate changes to each other, such that you can see change patterns through the tasks.
- I would expect a # to indicate the number of tasks downstream that are impacted, or the # of people I bother with my change. I cannot discriminate between the tasks currently
- There are some property changes that have had impact on the process, all changes have the same impact
- nothing fancy. everything is equally expensive....  
but, WAIT a second! how much does it cost in absolute numbers?! i should chekc this out!
- That i see a lot of info. That all changes have the same impact. That i cannot see perfectly the percentage. That i have to look for a long time to see what change belongs to what colour.



### WHAT ARE THE CONCLUSIONS YOU CAN DRAW FROM THE WIDGET ABOVE?

- The change in airconditioning is a critical one impacting lots of other areas of my product model. Still, I'm not sure what to do. I can't revert the change, I'm sure my colleagues had a good reason for it.
- Blue and green are costly, but I cannot go to somewhere to inspect the change. Second, I would now also expect that you could estimate the impact of a new change based on this data
- Some changes are apparently very costly; I cannot judge why unfortunately.
- There are some property changes that have had impact on the process, two changes in Airconditioning 1 have the most impact. let's check this out!
- ooh, i should definitely focuses on blue and green changes and see what happened.
- Adding property to airconditioning costs a lot of money, the same holds for adding a property to airconditioning. I cannot see the difference between the 3 top changes.

### DO YOU THINK THIS WIDGET CONTRIBUTES TO ITS PURPOSE AND HELPS TO ACHIEVE ITS GOAL? (RATE FROM 1 TO 5)

Average score: 3.16