



Baden-Wuerttemberg Corporate State University  
Engineering Faculty - Applied Computer Science IBC

Project Thesis

**Buoy Position Modeling and Correction  
in the SAP Sailing Analytics Suite**

Author	Jonas Dann
Matriculation Number	3346893
Company	SAP SE Dietmar-Hopp-Allee 16 D-69190 Walldorf
Department	GM Sponsorship
Class	TINF13AIBC
Company Supervisor	Dr. Axel Uhl Chief Development Architect axel.uhl@sap.com
Time Frame	2015-11-23 - 2015-02-12

# Statement of Authorship

“This is to solemnly declare:

1. that I have produced this project thesis on the topic

## **Buoy Position Modeling and Correction in the SAP Sailing Analytics Suite**

by myself;

2. that all ideas taken directly or indirectly from other sources have been marked as such;
3. that the digital version is identical to the printed version.

I am fully aware of the legal consequences of making a false declaration.”

Walldorf, March 5, 2016

\_\_\_\_\_ Jonas Dann

# Abstract

Especially with privately tracked sailing events it can be important to have an administrative interface to correct the buoys' positions on the sailing course. This currently is already possible in the SAP Sailing Analytics suite, but very complicated and error prone. The goal of this thesis is to make the process of buoy position modeling and correction as pleasant for the user as possible. Thus, an approach to user interaction design called Goal-Directed design is applied to the problem. The design produced by this process was then implemented. The implementation successfully meets the user goals but has some problems that are discussed in the final part of the thesis.

# Contents

<b>Indexes</b>	<b>v</b>
List of Figures . . . . .	v
List of Tables . . . . .	vi
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation and Goal . . . . .	1
1.2 Thesis Structure . . . . .	2
<b>2 SAP Sailing Analytics</b>	<b>3</b>
2.1 Overview . . . . .	3
2.2 Race Map . . . . .	5
2.3 Current Method of Buoy Position Editing . . . . .	6
<b>3 User Interaction Design</b>	<b>10</b>
3.1 Overview . . . . .	10
3.2 Goal-Directed Design . . . . .	11
<b>4 Requirements Analysis</b>	<b>14</b>
4.1 Persona . . . . .	14
4.2 Scenarios . . . . .	16
4.3 Requirements . . . . .	18
<b>5 Buoy Position Editing View</b>	<b>19</b>
5.1 Mockups . . . . .	19
5.2 Backend Service . . . . .	26

5.3	Implementation . . . . .	28
5.4	Result . . . . .	29
<b>6</b>	<b>Discussion and Outlook</b>	<b>31</b>
	<b>Bibliography</b>	<b>32</b>

# Indexes

## List of Figures

2.1	Course Layout Model . . . . .	4
2.2	Race Map . . . . .	5
2.3	Current Method of Buoy Position Editing . . . . .	7
2.4	Current Method of Buoy Position Editing Step 1 . . . . .	7
2.5	Current Method of Buoy Position Editing Step 2 . . . . .	8
2.6	Current Method of Buoy Position Editing Step 3 . . . . .	9
3.1	Phases of Goal-Directed Design . . . . .	12
5.1	Mockup 1.1 Buoy Position Editing . . . . .	20
5.2	Mockup 1.2 Buoy Position Editing . . . . .	21
5.3	Mockup 2 Buoy Position Editing . . . . .	22
5.4	Mockup 3 Buoy Position Editing . . . . .	23
5.5	Mockup 4 Buoy Position Editing . . . . .	24
5.6	Table with Button to Add a Mark . . . . .	25
5.7	Fix Position Overlay . . . . .	26
5.8	Mark Position Service . . . . .	27
5.9	Result . . . . .	30

## List of Tables

4.1	Requirements . . . . .	18
-----	------------------------	----

# 1 Introduction

## 1.1 Motivation and Goal

*“This is still a vision, it will still take a while. But where professional tracking systems are used today, in future smartphones could be used. One can well imagine that at some point in the future, a club regatta will be tracked by registering smart-phones with SAP Sailing Analytics, so that after going out onto the water and racing, the sailors can watch their own race back in the club-house.”*

– Stefan Lacher, Head of Technology SAP Sponsorships, July 2012

Since Stefan Lacher said that in 2012, SAP Sailing Analytics was extended by an option to track sailing races with mobile devices.<sup>1</sup> Even sailors only equipped with a smartphone and the respective app can now track their races and watch it back later. Before the race starts a race setup process has to be undergone, including pinging the position of the buoys, marking the course. Alternatively to pinging the buoys manually with a smartphone, a smartphone can be attached to a buoy to track its position frequently. Though this is possible the pinging app is used in most cases, because normally the sailors don't have multiple phones to attach to buoys and the phones could get lost.

While pinging multiple things can go wrong. Maybe the location tracking of the smartphone currently is inaccurate, the wrong buoy is pinged or the sailors forgot to ping the buoy, to name only a few possible problems. Thus, there should be a convenient way to edit the buoy positions after the race ended.

---

<sup>1</sup>cf. Broß, “Tracking sailing regattas with mobile devices”.



It currently is possible to edit buoy positions, but it is complex and error prone. The goal of this project thesis is to make the process of editing buoy positions afterwards visual and pleasant to use. It will explore the current development of user experience design and apply it to the new way of buoy position editing.

## **1.2 Thesis Structure**

After the introduction this thesis begins with an outline of the SAP Sailing Analytics suite, a more detailed look at the most important component of the suite, the race map, and a description of the current method of buoy position editing. This is followed by an overview of user interaction design and more detailed look at a specific implementation of user interaction design, Goal-Directed design. Section 3.2 also illustrates the process of Goal-Directed design that is executed in the subsequent chapters. The thesis is concluded with some implementation details, a look at the end result and an discussion of the results combined with an outlook.

## 2 SAP Sailing Analytics

### 2.1 Overview

SAP Sailing Analytics is a sailing analytics suite that evolved around the idea of making sailing more watchable and analyzable for spectators, commentators, sailors and coaches.

It was really hard for spectators to follow the happenings on the water. In the best case, the boats only sail around a hundred meters away from the spectator and even this distance does not allow for a close following of single boats when the competitor field has more than a few sailors. This makes it especially hard for new spectators to get into the sport. Commentators had to rely on helpers on the water, watching the boats closely and maybe camera images to comment on sailing races. Sailors and coaches used their intuition and reconstructed races from their memory to “analyze” races and improve their skills.

SAP Sailing Analytics can help solve these problems. Races are tracked with GPS trackers attached to the boats and the buoys, marking the course on the water. The tracking data is shown on a map, further explained in section 2.2. The map is only one of many components of the analytics suite. Other components include a landing page, an administrator console (admin console) and a race simulator.

The landing pages shows all the sailing events, tracked with the analytics suite, and lets the user navigate and view the regattas, races and leaderboards of the event. Most of the administrative tasks are done in the admin console. It combines views for creating sailing events, races, competitors, course layouts, etc. Another important task, done in the admin console, is the linking of races to external race data providers that track races and provide services to access the race data.

Races tracked with smartphones are too maintained in the admin console. Particularly the course layout and mark data editing is interesting for this thesis. Figure 2.1 shows a simplified course layout model. Attributes and classes unimportant in this context were left out.

A course layout consists of waypoints, that each are a combination of a controlpoint and a passing instruction that describes how boats have to sail around the buoy or through the waypoint to pass it. Controlpoints are a aggregation of zero to multiple marks. “2-Mark Controlpoints” are controlpoints explicitly consisting of only two marks. As “Mark” and “2-Mark Controlpoints” extend “Controlpoint” they can be waypoints too and mostly are. The marks are associated with GPS fixes that serve as the track that the mark has traveled over time. As mentioned earlier, one goal of the thesis is to be able to edit the GPS fixes graphically.

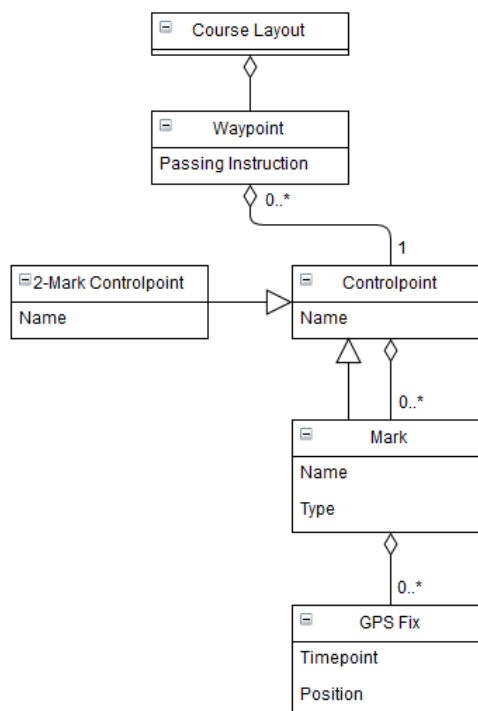


Figure 2.1: Course Layout Model <sup>2</sup>

<sup>2</sup>own illustration

## 2.2 Race Map

The race map is one of the most important components of the sailing analytics suite. It can stream sailing races live as they are happening and replay races, sailed in the past. Figure 2.2 shows the layout of the race map with an example race. The race map is split up in three main parts. A map (1) showing how the boats sail around the course, a leaderboard (2) where the competitors of the race are listed and a time slider (3) that can be adjusted by the user.

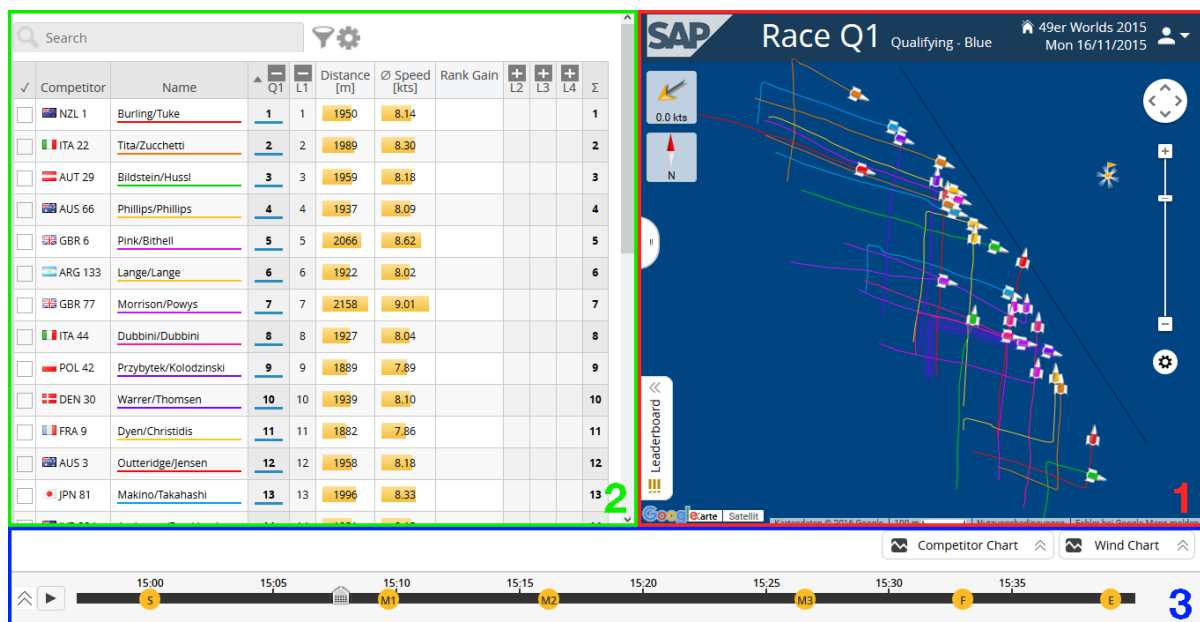


Figure 2.2: Race Map <sup>3</sup>

To show the competitors on a map, boat images are drawn onto a Google map. Besides the sailing boats, the map contains several helplines, including a tail where the boats sailed along, an advantage line that shows the advantage, to the wind, of the first competitor towards the others and several others. Furthermore, the map has a header that shows some meta data about the race being watched.

<sup>3</sup>own illustration

The leaderboard contains the main analytics capabilities of the race map. The competitors are shown in a table and for each leg, the routes sailed between the course marks, various calculated values are displayed. The columns shown in the table can be configured. Only two of many different values are shown in figure 2.2. Competitors in the table can be selected and selectively hidden from the map to gain a better overview.

The time slider is used to control the time of the state that is shown on the map and the leaderboard. The yellow markers symbolize the start, end and mark passings of the first boat. Above the slider on the right there are two charts that can be expanded. The competitor chart is used to compare the performance of competitors chosen in the leaderboard over the period of the race. The wind chart shows wind data collected for the race.

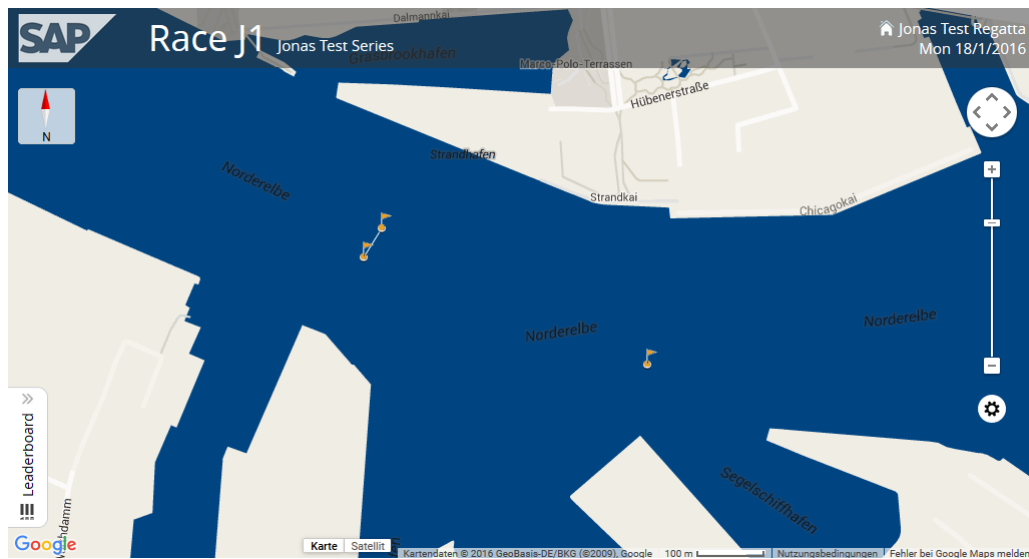
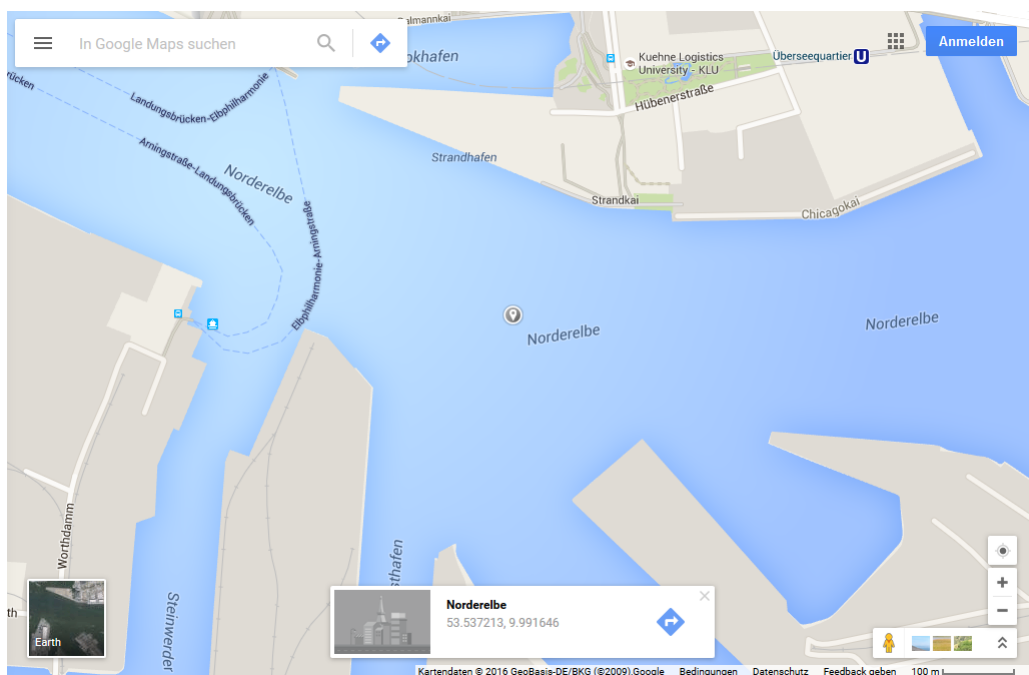
## 2.3 Current Method of Buoy Position Editing

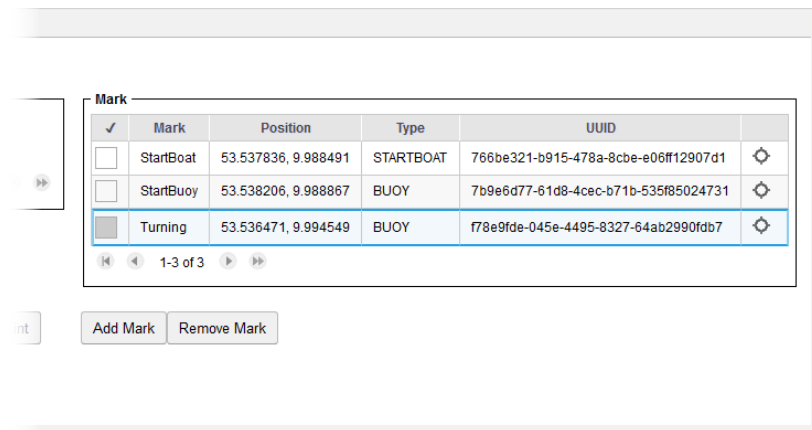
It is already possible to edit buoy positions. But the current method of buoy position editing is time consuming and error prone. How an administrator can move a mark from one position to another will be shown by the following screenshots and explanations.

Figure 2.3 shows the current state of an example race. There are three markers, one for the start boat and a buoy next to it, forming the start and finish line, and one buoy, the sailors have to maneuver around. For simplicity the boats have been left out. An administrator now wants to move the single buoy marker towards the other two markers.

The first step he has to do is to open another browser window with a Google maps tab, move one window behind the other and switch between them repeatedly to align the two maps to each other. Landmarks can help with this.

Figure 2.4 shows the almost perfectly zoomed and aligned map of the harbor in Hamburg that fits the map in figure 2.3. The administrator can now focus his eyes the point on the screen where he wants to move the marker on the Race Map and then switch to the Google maps window, left clicking where he focused. Following, Google maps shows such a marker as in figure 2.4 with a popup at the bottom, displaying the latitude and the longitude of the position.

Figure 2.3: Current Method of Buoy Position Editing <sup>4</sup>Figure 2.4: Current Method of Buoy Position Editing Step 1 <sup>5</sup><sup>4</sup>screenshot<sup>5</sup>screenshot

Figure 2.5: Current Method of Buoy Position Editing Step 2 <sup>6</sup>

After opening the administration console, the administrator has to enter the “Connectors” menu item where he clicks on the “Smartphone Tracking” tab. A list of leaderboards is presented to him and after choosing the respective leaderboard, he can open a course layout popup partly shown in figure 2.5.

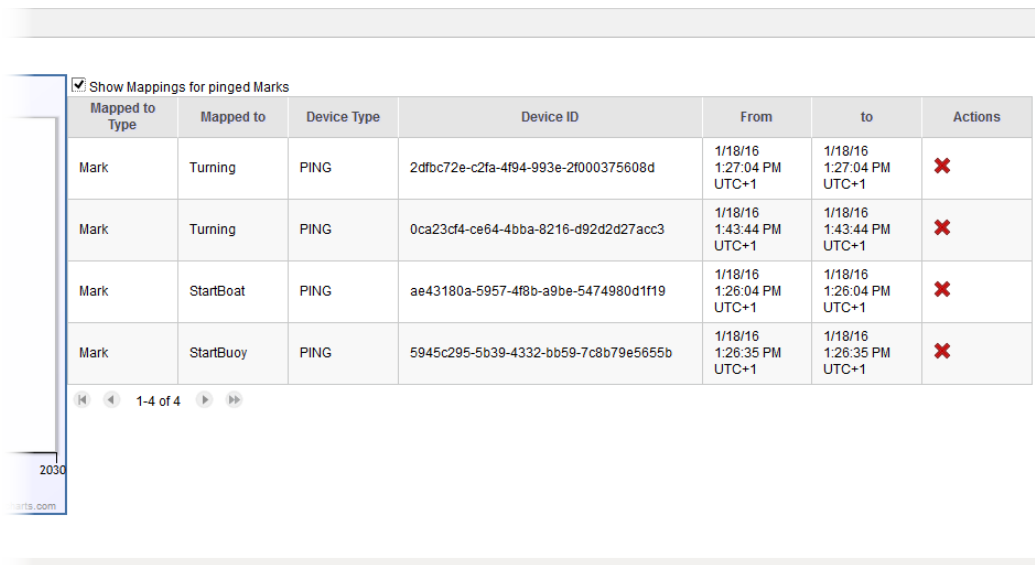
The popup contains a table of marks with a clickable icon in the last column that allows the user to enter a new latitude and longitude for the mark. But this position entered is set for the current time point determined by the system clock. Thus, the administrator has to change the system clock to the time point during the race that he wants to set the position for. Finally he can copy the latitude and longitude from the Google maps window and paste them to the administration console. However, the old mark position is still there.

To remove the old mark position the administrator has to close the course layout popup and open the device mapping popup, partly shown in figure 2.6, in the same “Smartphone Tracking” tab. There he has to find the correct device mapping, by looking at the timestamp and remove it by pressing the red cross in the “Actions” column.

Multiple problems arise from this process. The information, lists, tables and forms, needed to move a mark, are scattered in the system and buried in unnecessary technical terms and information, requiring the user to have opened multiple tabs and browser windows and

---

<sup>6</sup>screenshot



Mapped to Type	Mapped to	Device Type	Device ID	From	to	Actions
Mark	Turning	PING	2dfbc72e-c2fa-4f94-993e-2f000375608d	1/18/16 1:27:04 PM UTC+1	1/18/16 1:27:04 PM UTC+1	✗
Mark	Turning	PING	0ca23cf4-ce64-4bba-8216-d92d2d27acc3	1/18/16 1:43:44 PM UTC+1	1/18/16 1:43:44 PM UTC+1	✗
Mark	StartBoat	PING	ae43180a-5957-4f8b-a9be-5474980d1f19	1/18/16 1:26:04 PM UTC+1	1/18/16 1:26:04 PM UTC+1	✗
Mark	StartBuoy	PING	5945c295-5b39-4332-bb59-7c8b79e5655b	1/18/16 1:26:35 PM UTC+1	1/18/16 1:26:35 PM UTC+1	✗

1-4 of 4

Figure 2.6: Current Method of Buoy Position Editing Step 3 <sup>7</sup>

moving data between them. Even an aggregation of these elements would make moving a mark easier. Resulting from this, errors can happen easily. For example, wrong device mappings could be removed, latitude and longitude could be interchanged or the wrong system time could be entered.

---

<sup>7</sup>screenshot



# 3 User Interaction Design

## 3.1 Overview

*“Most digital products today emerge from the development process like a creature emerging from a bubbling tank. Developers, instead of planning and executing with a mind towards satisfying the needs of the people who purchase and use their products, end up creating technologically focused solutions that are difficult to use and control.”*

– About Face 3: The Essentials of Interaction Design<sup>8</sup>

The current solution is incredibly technology focused and more of a side product of the admin console than a wanted feature. As it is already possible to edit mark positions in the admin console, the main goal of this thesis is to make the process of mark position editing as pleasant as possible for the user. The project tries to dodge exactly the pitfalls described in the quote above. The principle of user interaction design is applied to mark position editing.

User interaction design is “the practice of designing interactive digital products, environments, systems, and services”<sup>9</sup>. The focus of interaction design lies on the design of behavior. Software and software enabled products often behave a lot more abstract and complex than purely mechanical products. Thus, a firm understanding of the user, how

---

<sup>8</sup>Cooper, Reimann, and Cronin, *About face 3: The essentials of interaction design* / Alan Cooper and Robert Reimann, p. 3.

<sup>9</sup>Cooper, Reimann, and Cronin, *About face 3: The essentials of interaction design* / Alan Cooper and Robert Reimann, p. xxvii.

the user thinks, what he wants and how he uses the software or software enabled product is required. This information is gathered in user research, reconditioned and used in the design process to build a product optimally tailored to the user.

There were multiple approaches to user interaction design considered for this thesis. These include the systems design approach, the genius design approach and Goal-Directed design.<sup>1011</sup> The approach to interaction design used in this thesis is Goal-Directed design. It was chosen over systems and genius design because it is centered heavily on the user and the goals of the user. Systems design is more centered on the design of components of a system and as said before the goal of this thesis is to create a solution that is as pleasant as possible for the user. The genius design approach is focused on the skill and wisdom of the designer, but the writer of this thesis is no expert in design and has not had several years of job experience.

## 3.2 Goal-Directed Design

“Goal-Directed Design is the approach to product and service design developed at Cooper, a leading design consultancy. Its fundamental premise is that the best way to design a successful product is to focus on achieving goals.”<sup>12</sup> It combines many different techniques, interaction principles and patterns to be able to reliably work towards achieving goals. The process of Goal-Directed Design is located after the initiation of a project and the construction. The design should be completed before the project is built.<sup>13</sup> Goal-Directed Design is divided into six main phases, research, modeling, requirements definition, design framework, design framework, design refinement and design support. The phases are depicted in figure 3.1.

To be able to work towards the users goals means to gain a deep understanding of the user in the first place. In the research phase the scope and general direction of the project

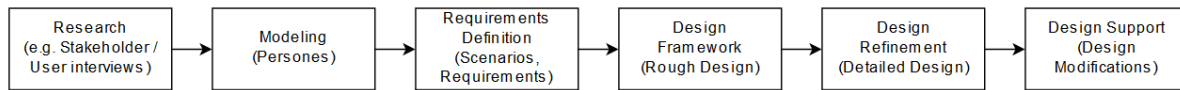
---

<sup>10</sup>cf. Saffer, *Designing for interaction: Creating innovative applications and devices*, p. 33.

<sup>11</sup>cf. Cooper, Reimann, and Cronin, *About face 3: The essentials of interaction design* / Alan Cooper and Robert Reimann.

<sup>12</sup>cf. Goodwin, *Designing for the digital age: How to create human-centered products and services* / Kim Goodwin.

<sup>13</sup>cf. Cox, *You Can't Detect A Touchscreen — Blog — Stu Cox*.

Figure 3.1: Phases of Goal-Directed Design <sup>14</sup>

is defined. Thereafter, stakeholders, that includes internal managers, customers, etc., are interviewed. Customers means in this context not the end users of the software, but the people buying the project. Stakeholder interviews can define a more detailed scope of the project before interviewing and analyzing end users of the projects result. Good user research is the foundation for a successful design process.

The modeling phase is about setting down the insights of the user research in models later used for constructing the design of the product. The modeling tool used in this thesis is personas that describe exemplary users. Personas capture the most important aspects of each potential user group and define the user groups' goals, ideas and background. Personas are a good medium to argue about design decisions, because they are relatable and one can put oneself into the persona to get away from their own model of thinking.

After the personas are defined they can be used to describe key scenarios that convey how users would use the product. The definition of key scenarios is also closely connected to user research and specifically when there is an existing process that should be mapped with the new product, it is possible to find out how users would imagine a better tool. The scenarios can then be compiled into requirements that have to be implemented by the design.

The fourth phase establishes a design framework to work from on more detailed designs. In huge applications spanning a multitude of views, a design framework, especially the design structure and flow of the product, is very important, but as the mark position editing is a rather small, confined problem that does not need a real flow of views, the thesis will skip this step and directly go over into a more detailed design phase.

---

<sup>14</sup>own illustration

After the design framework is laid down, the design is further detailed. The appearance, widgets, behavior, information, visualization, etc. are defined and the views of the system are worked out in detail. For larger projects it is essential to have a good design framework for a coherent design to come together in this phase. At the end of this phase the design of the product should be completed.

Finally, the design support phase is the transition into the next big phase of the project, the construction of the product. Requirements, needs and goals of the customer will change and the developers should be supported if changes to the design have to be made.

## 4 Requirements Analysis

### 4.1 Persona

“Personas are archetypes that describe the various goals and observed behavior patterns among your potential users and customers.”<sup>15</sup> They aggregate the most important behaviors and aspects of each user group. The following personas are extracted from a bug in the projects bug tracker and a conversation with the company supervisor of this project thesis, Axel Uhl. He has already spent time with several potential users of the solution and verified the personas after they were created and modified after consultation with him.

#### **Peter Müller**

Peter Müller is an enthusiastic sailor and member of a sailing club. He loves to race with his friends in the club and do training weekends every one or two months. The races are very competitive and Peter seeks to improve his sailing skills every race and training session. When he was at the Extreme Sailing Series Event in Hamburg, he was impressed by SAP Sailing Analytics and wants to use it to improve his own sailing skills. Thus, he convinced the sailing club chairman to introduce SAP Sailing Analytics for the club. The club chairman in return asked him if he would take the role of the race administrator that sets up and maintains races in the new system, at least for his own races and trainings. He sees himself capable of fulfilling this role, because he knows a lot about sailing, and agrees to do it.

---

<sup>15</sup>cf. Goodwin, *Designing for the digital age: How to create human-centered products and services* / Kim Goodwin.

**Peter's goals:**

- Get precise replays when tracking a race: The system was introduced to improve the club members' sailing. Thus, it should provide precise replays, to be able to analyze the race afterwards and improve upon it.
- Focus only on the necessary administrative tasks: Sailing is exhaustive and time consuming. Therefore, Peter does not want to spend a lot of time configuring races before sailing and maintaining the course data afterwards.

**Michael Wagner**

Michael Wagner is a functionary in a sailing club and wants to differentiate his club from other clubs in the area. The club has weekly training sessions and to attract more members he wants to use SAP Sailing Analytics in the training. To make it most pleasant for the club members training he wants to operate the system and just present the results to the trainees.

**Michaels goals:**

- Get precise replays when tracking a race: The system was introduced to improve the club members' sailing. Thus, it should provide precise replays, to be able to analyze the race afterwards and improve upon it.
- Edit the race configuration while the race is in process: Michael wants to be able to edit the race data on his tablet while the race is in process.
- Be able to perfect the race data: The system should be easy to use and focus on the important features, but Michael has the motivation to spend more time than Peter on maintaining the race data.

The main persona of this thesis is Peter Müller. He describes the general potential user of the mark position editing tool. Michael Wagner is more of a specialization of Peter. He represents the subset of potential users that are more invested in the tool and really want to get the best results from using it. He is a power user so to say.

## 4.2 Scenarios

Scenarios are a plausible description of how the product will be used by the users in the future. Scenarios are already used in a wide variety of fields, such as military tactics and urban planning, to ensure that by imagining all possible and probable happenings the solution to a problem covers all possibilities. Each scenario can involve one or several personas and describe what they would do. The scenarios are a good foundation to argue upon about design decisions, because they give the designers concrete examples to test their design against and detect flaws in their design.

The scenarios for this project were too created in collaboration with Axel Uhl.

### **Scenarion 1 (Peter Müller): Correcting mark fixes**

Peter has a free weekend and decides to go train his sailing skills. He gets to his club and sets up a race with him as the only competitor. His position on the water should be tracked with his smartphone, therefore he maps his phone to the competitor. Peter puts his phone away safely, makes a boat ready and launches it onto the quarry pond that is in close proximity to the clubs location. With a race course in mind he starts the tracking app on his phone and sets off to sail it as fast as possible. During his sailing session the app tracks the course he went for and as he gets off of the water he wants to know how he did. After putting away the boat, he opens up the Race Map of the race he previously created in the administration console on his tablet, but cannot get useful information about his training, because no race course layout and mark positions are defined. Peter creates the fitting amount of markers and defines the course layout. Thereafter he watches his race back, selects each marker and positions it on the map, when he sees where he sailed and where its position should be. It is not necessary for him to have perfect marker positions, but only with a rough course layout and marker positions he can use the analytical capabilities of SAP Sailing Analytics.

**Scenarion 2 (Peter Müller): Editing wrong mark fix**

Peter sailed a smartphone tracked race with his friends. Before racing they created the race in the administration console, layed out the buoys on the water and pinged them with their smartphones. After the race they arrive in the club house and want to watch their race back. They open the Race Map on their tablets, but the position of a mark on the course is wrong by 100 meters. Probably the location determination of one of their smartphones was wrong. This wrong position hinders them in having useful information in the leaderboard. Thus, Peter moves this one mark on the map to the right position and the sailors can finally watch and analyze their race.

**Scenarion 3 (Michael Wagner): Add new mark fix while race is going on**

Michael has organized a sailing weekend for the teenagers in his sailing club. He has prepared every race in SAP Sailing Analytics prior to the weekend. Furthermore he layed out the buoys and pinged them early in the morning before the first training races started. All the boat of the teenagers are equipped with smartphones that track their position and the first race started already. Now he stands on the start boat with his tablet and watches what is happening in the distance on the Race Map. He sees that all the boats are turning at a different position as the buoy marker is on the map. The buoy must have been moved by the wind. He moves the time slider to just before the first boats arrived at the buoy and adds a new marker position at where the boats turn. He then goes back into live mode and continues to watch the race.

**Scenarion 4 (Michael Wagner): Editing wrong mark fixes**

Michael got a few really cheap smartphones to attach them to the buoys marking the course. This enables a much better position tracking of the buoys. He layed down the buoys and attached the smartphones to them. Furthermore he already created a race with the course layout and device mappings. Some club members are now sailing the course and when they finish, Michael wants to analyze the race with them. It seems that some of the positions tracked by the smartphones are wrong. This is a typical outlier scenario



and can even happen with dedicated GPS trackers. Michael deletes them, to clean up the fixes. He can then show the race to the race competitors.

## 4.3 Requirements

Besides being used to evaluate designs, scenarios are too used to construct a list of requirements for the product. Requirements have the advantage against scenarios of being shorter and easier to overview. But the requirement list generated here is not fixed and does not take an as prominent spot in the project process as in many other projects. “Most requirements processes are fraught with problems. Marketers or product managers spend prodigious amounts of time developing requirements and are disappointed when a stellar product does not result.”<sup>16</sup> The following requirements were not gathered or developed in the open, but taken from the scenarios. Only new requirements that emerge in a scenario are listed for each scenario.

Scenario	Requirement
1: Correcting mark fixes	1. View all the marks of the course 2. Add a new position for a selected mark
2: Editing wrong mark fix	3. View all positions of a mark 4. Move a position of a selected mark
3: Add new mark fix	5. View mark positions that are outside of the race time frame 6. Be able to use touch and mouse
4: Editing wrong mark fixes	7. Delete a position of a selected mark

Table 4.1: Requirements

Another stakeholder requirement not documented in the table is the ability to reuse the mark position editing tool in a course layout editor that should be created in the future.

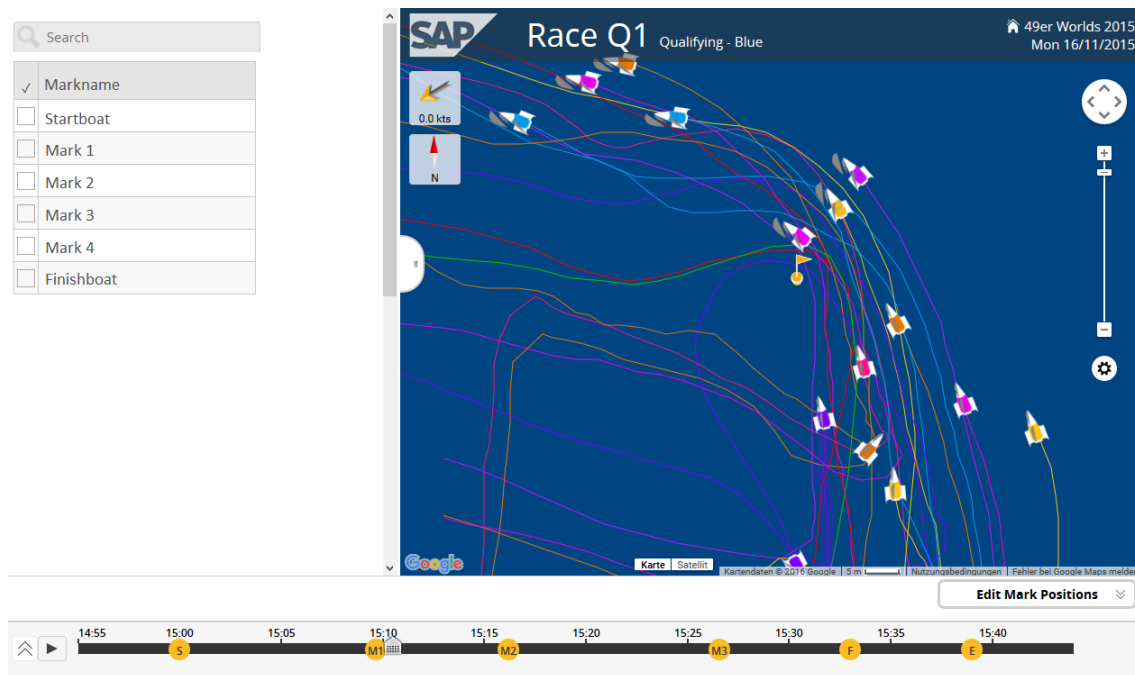
<sup>16</sup>cf. Goodwin, *Designing for the digital age: How to create human-centered products and services* / Kim Goodwin.

# 5 Buoy Position Editing View

## 5.1 Mockups

From the requirements established above multiple possible mockups were developed and are now compared to each other. Because of the scope of the thesis and the mark position editing tool only having one view, the mockups are already detailed and not focused on high level flow of the application. To be able to edit the GPS fixes visually the editing tool is located in the race map. The race map already implements a lot of the needs for a graphical editing tool and can be easily expanded. The basic idea that every possible mockup is based upon is shown in figure 5.1.

There already is the established way of navigating the race with the Google map and the time slider. These two elements should stay, because similarities with the surrounding application are important for users to easily understand certain areas of the application. Moreover they work well as they are and specifically the time slider is already established in other areas like video players. As explained in section 2.2 the bar above the time slider is used for expandable charts. This fits the button to enable the mark position editing tool really well, because there may be a chart or something similar to display the GPS fixes in that area. The main change in the stage of the mark position editing tool, where no mark is selected, is the table of marks on the left hand side of the screen. It contains all the marks that are defined by the race. There has to be a complete list of marks, because if they have no position on the map they are not shown on the map. Thus, a purely map based selection of marks to be edited would not be possible. Another idea was to have a palette of mark icons that the user can select one from, but the only identifying attribute the marks have is their name. The table is located on the left upper side of

Figure 5.1: Mockup 1.1 Buoy Position Editing <sup>17</sup>

the screen, because it fits the master detail view pattern that is already established. The master view is the table where the user is able to select a mark that he want to edit the “details” (fixes) of. The positional details should then be displayed in the map and the time details in close proximity to the time slider, both together acting as the detail view. This placement of the elements furthermore supports the reading direction at least of western cultures where people read from left to right and from top to bottom.<sup>18</sup> Thus, one expects the “beginning”, the starting point, of the tool to be in the upper left corner.

There are now multiple options to display the time points and the order of the GPS fixes. It is not only important to be able to edit the location of the selected mark, but to edit the location of the selected mark at a certain time point. There are four promising options that are subsequently discussed on the basis of the next four mockups.

<sup>17</sup>own illustration

<sup>18</sup>cf. Goodwin, *Designing for the digital age: How to create human-centered products and services* / Kim Goodwin, p. 574.

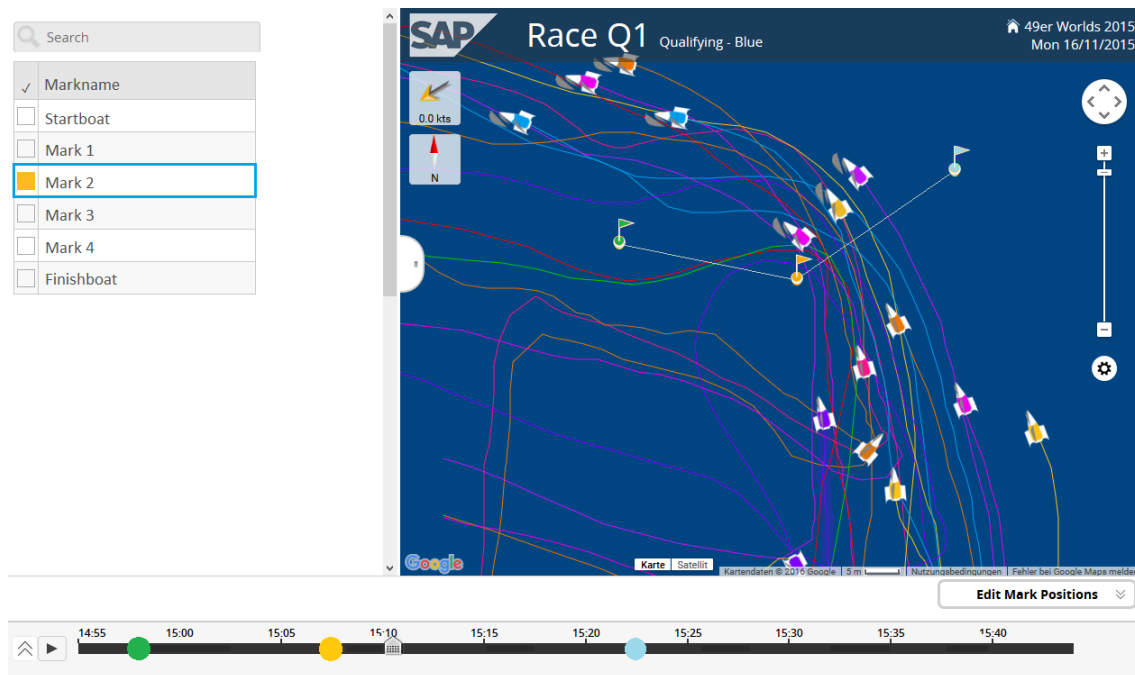
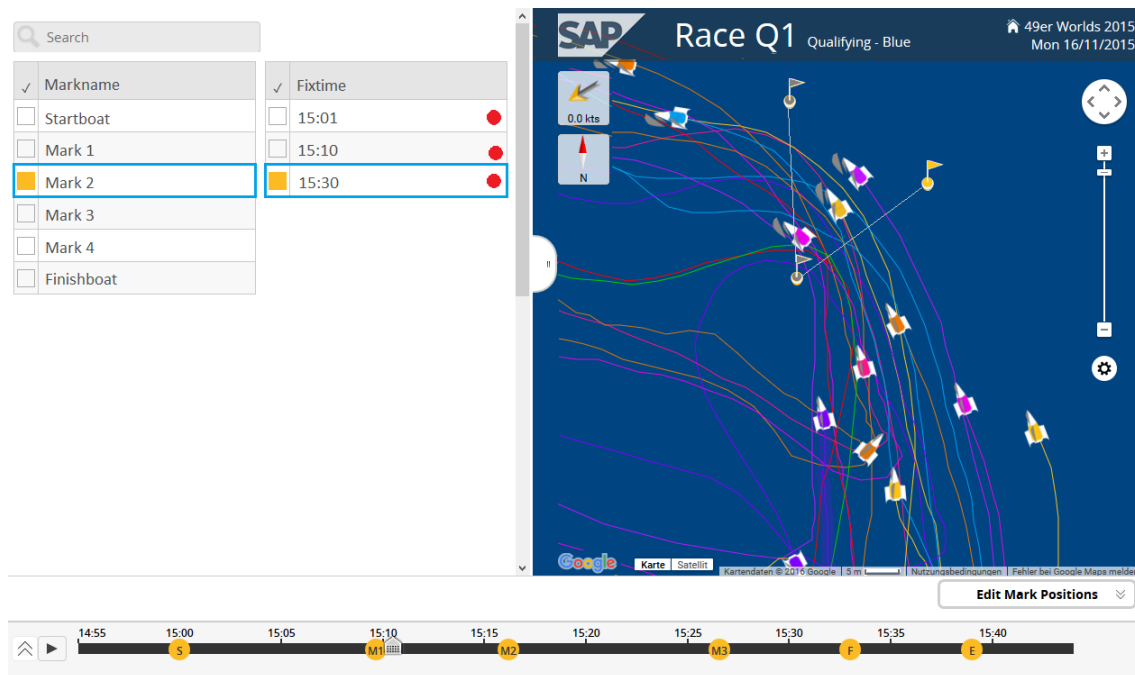
Figure 5.2: Mockup 1.2 Buoy Position Editing <sup>19</sup>

Figure 5.2 shows the first of the four options. The time points of the different GPS fixes are directly embedded into the existing time slider. This is similar to the mark passings displayed as circles on the time slider. The advantages of this design is its compactness. This is especially important for touch devices as they generally have less screen real estate. But this design also has negative aspects. It is really hard to display a lot of fixes this way. Though this could be solved by implementing a zoom on the time slider. But as the zoom directly on the slider is implemented nowhere else in the application it would lead to inconsistencies. It is impossible to instantly detect outlier fixes whose position differ hugely from the rest of the fixes from just looking at the time slider. Probably the biggest problem with this design is that the context of mark passings and the start and end of the race is lost. The mark passing is especially important for the mark that is currently edited. As around the time of passing the mark the tails of the boats curve around the mark and give a good indication where the mark should have been.

---

<sup>19</sup>own illustration

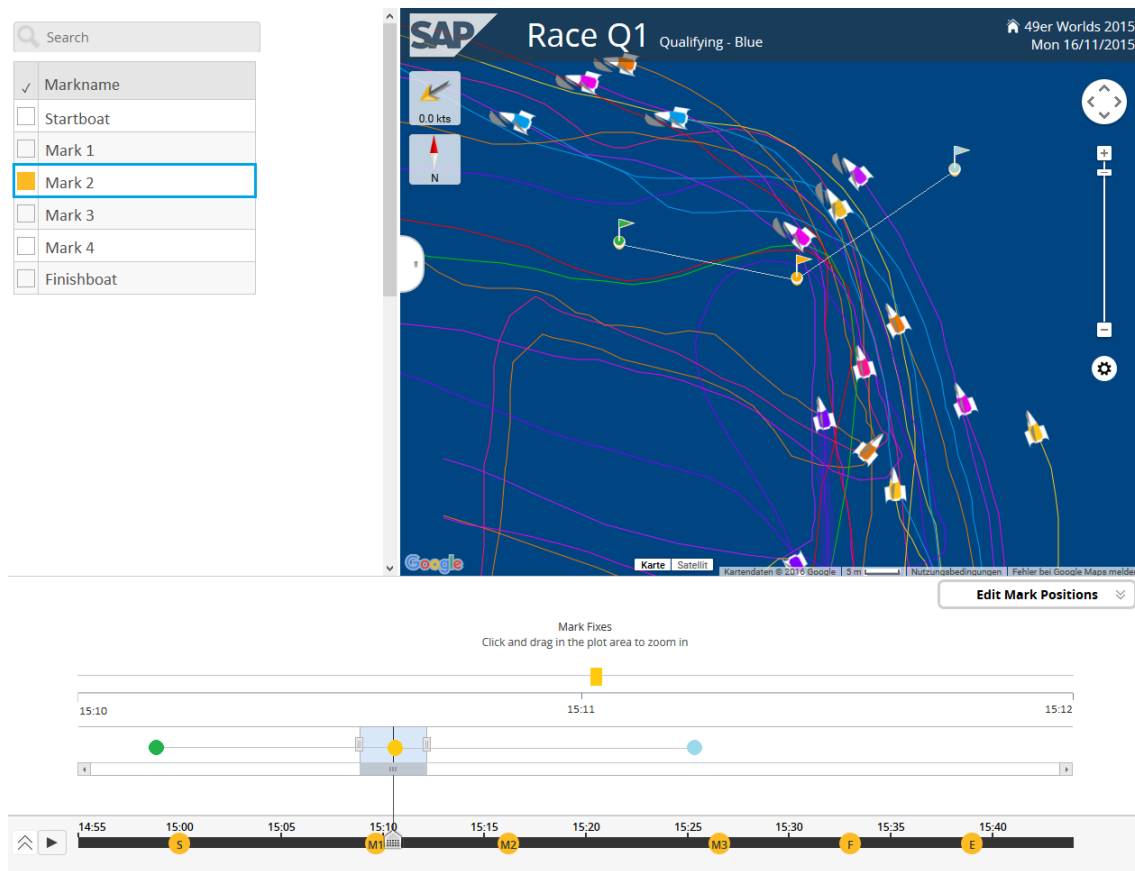
Figure 5.3: Mockup 2 Buoy Position Editing <sup>20</sup>

A second idea (figure 5.3) is to display the fixes for the selected mark in a table directly alongside the mark table. This effectively solves the problem of displaying a huge amount of fixes. The table would be scrollable. Furthermore the deletion of fixes could be elegantly embedded into the table by adding a column with a delete button (in figure 5.3 the red circles).

Coming with the ability to display huge amounts of fixes comes the problem of missing association with the time slider. In the worst case the user would have to scroll through a huge table of fixes to find the one he wants to edit. The time slider could scroll the table and make the searching process easier, but it non the less is a problem more elegantly solved by solutions in close proximity to the time slider. A positive aspect of the previous design was that is it very compact. This design is the opposite in this respect. It needs a lot of vertical screen space even if the tables are smaller than shown in figure 5.3.

Figure 5.4 shows the third possible idea. This design has a chart with two sections. The lower section shows all the fixes in the same time scale as the time slider and the upper

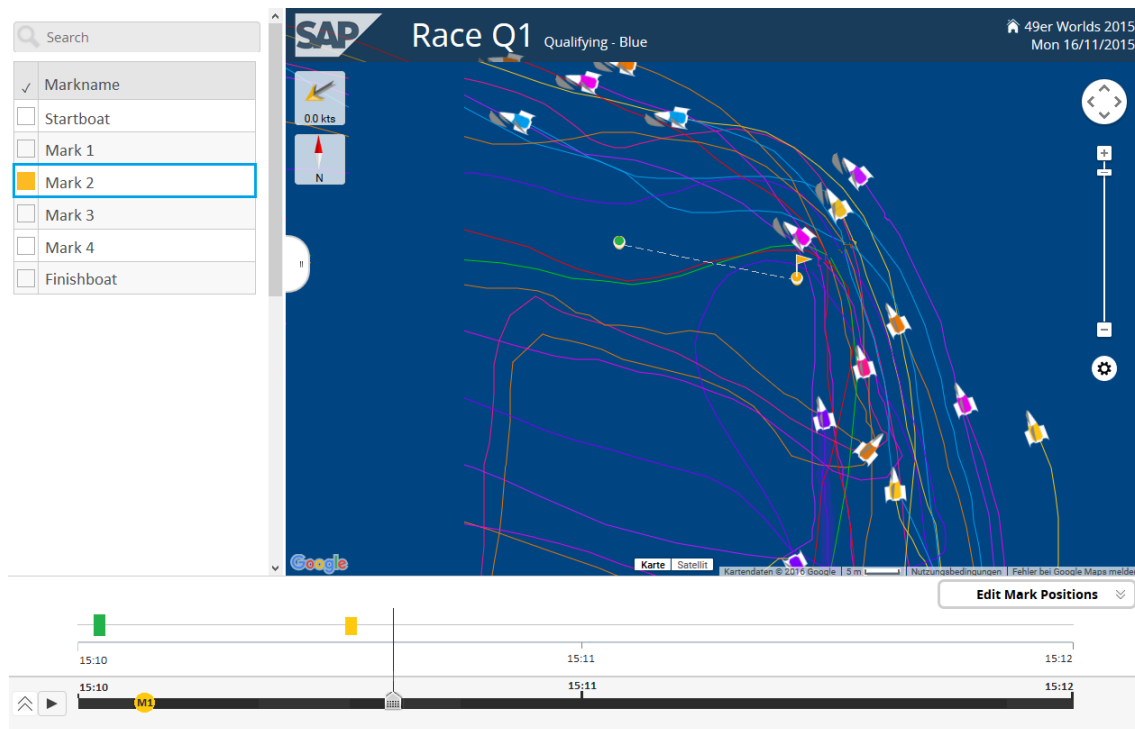
<sup>20</sup>own illustration

Figure 5.4: Mockup 3 Buoy Position Editing <sup>21</sup>

section shows a selected part of the lower section. This allows the lower part to display a huge amount of fixes while still being able to navigate the fixes precisely in the upper part. This design is really practical on big screens, because it is very clear, associates the fixes well to the time scale and does not lose the context of mark passings and the race start and end.

But this solution takes up a lot of space and especially with a low number of fixes the zoom function is not needed. The space problem would make the application harder to use on wide screen displays and smartphones. Furthermore the three time scales of the time slider, the lower section and the upper section of the chart take up space.

<sup>21</sup>own illustration

Figure 5.5: Mockup 4 Buoy Position Editing <sup>22</sup>

Finally the design shown in figure 5.5 was chosen. It has a chart like the third design, but spares the lower section. Thus, the ability to zoom in and inspect the fixes in more detail is still given, but some unnecessary space is saved. Moreover it keeps the other advantages of the third design.

The following paragraphs will play through the scenarios established in section 4.2. The interaction between Peter and the mark position editing tool in scenario 1 is very limited. He clicks on the button to open the mark position editing tool and wants to add a fix for each mark. He sees all the marks in the table on the left. For the purpose of adding a mark we modify the design by adding another column to the table with a button to add a mark. This is shown in figure 5.6.

Peter has to click on the button in each row to add a fix to each mark with the timepoint currently selected with the time slider. A button in the table was chosen, because it is similar to the action bars in other tables in the application and it only requires one click

<sup>22</sup>own illustration

✓	Marks	
<input checked="" type="checkbox"/>	Five Fixes	<button>Add new fix</button>
<input type="checkbox"/>	No Fix	<button>Add new fix</button>

Figure 5.6: Table with Button to Add a Mark <sup>23</sup>

instead of e.g. selecting a mark and clicking on the map to get a context menu where there is an “add mark” option. A labeled button was chosen instead of an icon, because for e.g. a plus icon it is not clear what this does. It could add a mark too. The labeled button of course has the tradeoff of being large and taking up a lot of screen space, but should be clearer.

After clicking on the button an overlay over the map has to appear to choose the position of the fix on the map. Peter could use a mouse or touch, so an overlay has to be chosen which fits both input styles. For usage with a mouse dragging the fix over the map is the best option. After the position is determined there should be two buttons to confirm or cancel the action. For touch devices, dragging the fix over the map is no option, because dragging is really inaccurate on touch devices as you cannot see what happens under your finger, currently touching the screen. The solution for this, inspired by the Google maps distance measure tool on smartphones, is to drag the map under the fix instead of the fix over the map. The fix is fixed in the middle of the screen and the map is draggable. Additionally to the fixed fix in the middle there should be two buttons that allow the user to confirm on this position or cancel the addition of the fix. The overlay is shown in figure 5.7

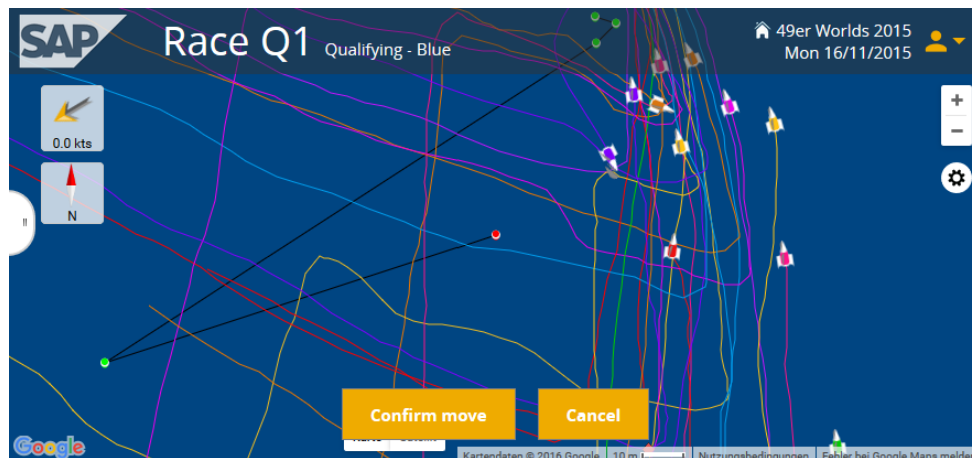
Newly introduced with scenario 2 is the editing of fix positions. Peter again opens the mark position editing tool and chooses the mark which has the wrong fix from the table. He sees the wrong fix and wants to move it. On touch devices he should be able to click on it and select to move the mark from a context menu. This should again open the overlay shown in figure 5.7. With a mouse he should drag the fix over the map.

In scenario 3, after Michael navigated to right before the first boats arrive at the buoy, he clicks on the button shown in figure 5.6 for the mark he wants to add the fix for. Again

---

<sup>23</sup>own illustration



Figure 5.7: Fix Position Overlay <sup>24</sup>

the popups described above to add a new fix open and the moves the fix to the correct position. Finally he confirms the new fix and leaves the mark position editing tool.

In scenario 4, Michael opens the mark position editing tool with the button every time he sees a mark with a wrong position. He selects the mark and if he uses a mouse, he right clicks on the wrong positioned fix and clicks the “delete fix” option in the newly opened context menu. If he uses a touch device he touches on the fix and clicks the “delete fix” option in the newly opened context menu.

## 5.2 Backend Service

Resulting from the requirements worked out in section 4.3 there are four main features the backend service, connecting the server and the frontend, should have.

The frontend has to be able to add fixes to the race via the service, it has to be able to delete fixes and to be able to move fixes it has to edit the position of a fix. These three features are straight forward and trivial to define in a backend service interface. The fourth main feature is the retrieval of the data displayed in the frontend. This includes marks and their fixes. The difficulty in this feature is with huge amounts of fixes. With certain races, e.g. the races at the Kieler Woche where races can realistically be 2 hours

<sup>24</sup>own illustration

long and buoys could be tracked with trackers that create a GPS fix every 1 second, depending on your internet speed it could be multiple minutes until fixes appear in the tool. Thus, a mechanism has to be built into the backend service that allows the frontend to only retrieve or the backend to only send a subset of all the fixes.

Furthermore the frontend has to check if the user is allowed to edit the mark fixes for this race and if he can remove certain fixes from the race.

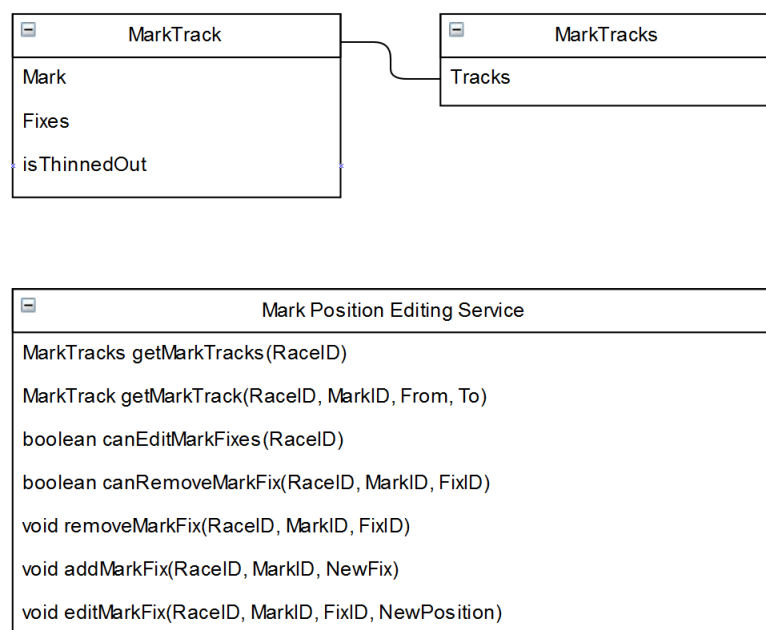


Figure 5.8: Mark Position Service <sup>25</sup>

Figure 5.8 shows the backend service, called mark position editing service. As the data elements like marks and GPS fixes are already defined and not further detailed in this thesis the specific data types were left out in the figure. The features of removing, adding and editing mark fixes and the both checks are trivial. The letters “ID” in the arguments names stand for identifier.

<sup>25</sup>own illustration

There are two methods to get marks and their fixes. A set of fixes for a mark is called its track. The first method “getMarkTracks” tries to get all the data, but if it is too much data to send it all at once it only sends a subset. There were several options to handle the case of too many fixes.

An exception for this case would be possible, but the frontend would have to decide what to do when there are too many fixes and this logic belongs into the backend. Another approach would be to aggregate fixes, but then there would have to be logic in place to handle the interaction with aggregated fixes. But, this is intransparent for the user.

The solution chosen here is to send a thinned out version of the fixes. If the track is thinned out is flagged in the “MarkTrack” data type. The frontend can load more fixes, a more detailed version of the track, if the chart showing the fixes is zoomed in to. This is done with the method “getMarkTrack” which takes a time span what part of the track the frontend expects. This can too thin out the track if it is still too much data.

A problem of the chosen solution is that the thinned out fixes may not contain the fix the user explicitly wants to edit. As this is probably a fix that hugely differs from the position of its surrounding fixes, this should be incorporated into the implementation of the service.

## 5.3 Implementation

In this implementation section only the modifications to the initial design and interesting problems during the implementation will be discussed. Furthermore it is to mention that the backend service of the mark position editing tool was not implemented because the time ran out.

There were several little modifications to the initial design. The search bar shown in the mockups above was not added, because there are normally not that many marks in a race that a search bar is needed and the mark names are at least currently more arbitrary than rational. The fixes in the chart and on the map are in the current implementation not colored, because a red colored fix symbol in the chart and on the map are used to

illustrate which fix is edited at the moment. Multiple other colors would weaken this highlight. No dashed line is used between the fixes on the map, because the Google map does not support dashed lines. Furthermore, the fix' y-position on the chart is determined by the distance from the average position of all the fixes on the map. This makes it easier to spot outlier fixes, because they do a spike in the chart.

The biggest problem with the implementation was the recognition of touch and mouse users. With the current landscape of different devices, supporting either one of the both input methods or allowing the user to switch between them and the browsers not reliably differentiating between touch and mouse clicks, it is impossible to limit the user interface to one input method or switch between them arbitrarily without making the user experience for one of the user group worse.<sup>2627</sup> Though this problem exists at the moment, it might be solvable in the future or there might be another solution.

Thus, input methods optimized for touch devices were generally favored over methods optimized for mouse users, because it is much easier to use a touch interface with a mouse than the other way around and where possible there were shortcuts added for the mouse users. The exact implementation of the mixture of these input methods is further detailed in section 5.4.

It was tested how many fixes can be displayed on the map and in the chart. The map supports well above 1000 fixes, but the chart can only display around 200 fixes. With as much as 20000 fixes the whole browser tab crashed. The implementation of these limitations and the usage of the “getMarkTrack” method (see section 5.2) was not achieved in the given time.

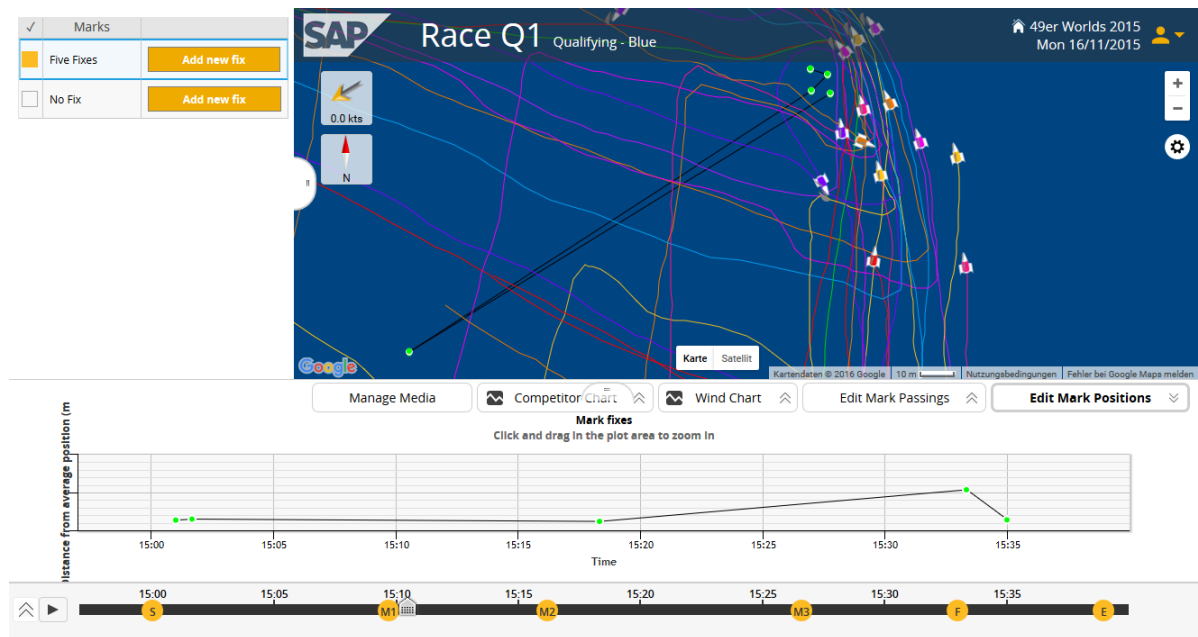
## 5.4 Result

Figure 5.9 shows the final mark position editing tool. Above the time slider there is the button that enables and disables the tool. Most of the behavior was already described

---

<sup>26</sup>cf. Cox, *You Can't Detect A Touchscreen — Blog — Stu Cox*.

<sup>27</sup>cf. Lauke, *Detecting touch: it's the 'why', not the 'how' - Mozilla Hacks – the Web developer blog*.

Figure 5.9: Result <sup>28</sup>

in the design process and the minor modifications to the design were mentioned in the implementation. The most significant thing that was not already described is how the touch and mouse input method work together in the tool.

When adding a fix with the button in the table, the overlay shown in figure 5.7 opens. When the user left clicks a fix on the map, a context menu opens that gives the option of deleting the fix that just removes it and an editing option that again opens the overlay shown in figure 5.7. As a shortcut for mouse users the fixes can too be dragged over the screen. When right clicking a fix, a context menu open which gives the option of deleting the fix. This is for mouse users and if there is ever a reliable method to detect how the user is interacting with the application at the moment the left click option should be disabled for mouse users.

The frontend shown in figure 5.9 is fully functional, but it is only connected to a dummy implementation of the backend service. The backend service was not implemented because the time ran out.

<sup>28</sup>screenshot

## 6 Discussion and Outlook

As mentioned in section 5.4 the mark position editing tool was not fully implemented with an implementation of the backend service missing. To really understand the existing backend and to integrate the mark position editing backend service takes a lot of time for somebody that does not know the existing backend already. For somebody with a good knowledge about that part of the SAP Sailing Analytics suite it should be easy. The interface is already defined it just has to be implemented.

The handling of huge amounts of fixes is currently a problem with the implemented tool. The backend service specification has the capabilities to handle this problem, but the frontend currently has no way of loading more detailed data if the chart is zoomed.

In the future this project should be taken as the foundation of a course manager that allows the user to not only edit mark positions but create marks and course layouts. This could benefit from predefined course layouts, something like templates for courses, that can be positioned on the water. The course manager is an important component on the way of delivering the SAP Sailing Analytics suite to private clubs and a graphical course manager would be a huge improvement to the current situation.

# Bibliography

- Broß, Jan. “Tracking sailing regattas with mobile devices”. In: (2013). URL: [http://wiki.sapsailing.com/doc/theses/20130913\\_Bross\\_Tracking\\_Sailing\\_Races\\_with\\_Mobile\\_Devices.pdf](http://wiki.sapsailing.com/doc/theses/20130913_Bross_Tracking_Sailing_Races_with_Mobile_Devices.pdf).
- Cooper, Alan, Robert Reimann, and Dave Cronin. *About face 3: The essentials of interaction design / Alan Cooper and Robert Reimann*. rev ed. Hoboken, N.J.: Wiley and Chichester : John Wiley [distributor], 2007. ISBN: 978-0-470-08411-3. URL: <http://proquest.tech.safaribooksonline.de.ezproxy.dhbw-mannheim.de/9780470084113>.
- Cox, Stu. *You Can't Detect A Touchscreen — Blog — Stu Cox*. URL: <http://www.stucox.com/blog/you-cant-detect-a-touchscreen/>.
- Goodwin, Kim. *Designing for the digital age: How to create human-centered products and services / Kim Goodwin*. Hoboken, N.J.: Wiley and Chichester : John Wiley [distributor], 2009. ISBN: 978-0-470-22910-1. URL: <http://proquest.tech.safaribooksonline.de.ezproxy.dhbw-mannheim.de/book/graphic-design/9780470229101>.
- Lauke, Patrick H. *Detecting touch: it's the 'why', not the 'how' - Mozilla Hacks – the Web developer blog*. URL: <https://hacks.mozilla.org/2013/04/detecting-touch-its-the-why-not-the-how/>.
- Saffer, Dan. *Designing for interaction: Creating innovative applications and devices*. 2nd ed. Voices that matter. Berkeley, CA and London: New Riders and Pearson Education [distributor], 2010. ISBN: 0321643399.