Cooperative State University Baden-Württemberg
Mannheim

Bachelor Thesis

Server Side Integration of Mobile Devices into SAP Sailing Analytics

Business Information Technology

Software Methodology

Author:	Fredrik Teschke
Enrolment Number:	9197281
Company:	SAP AG
Department:	Global Sponsorships
Class:	WWI 10 SWM A
Degree Course Supervisor:	Prof. Dr.-Ing. Jörg Baumgart
Academic Supervisor:	Prof. Dr.-Ing. Jörg Baumgart joerg.baumgart@dhbw-mannheim.de +49 (0)621 4105 - 1216
Company Supervisor:	Axel Uhl axel.uhl@sap.com +49 (0)6227 7 - 65950
Timeframe:	19 November 2012 - 10 February 2013

Abstract

Author: Fredrik Teschke
Class: WWI 10 SWM A
Company: SAP AG
Topic: Server Side Integration of Mobile Devices into SAP Sailing Analytics

In the last few years, the regatta sailing experience has changed for spectators and competitors alike through the increasing use of modern technology in the sport. SAP Sailing Analytics is a custom-built web application which takes on the challenge of transforming *Global Position System (GPS)* tracking data and wind data of sailing races into meaningful metrics, key figures and visualisations. Currently, SAP Sailing Analytics relies on tracking providers that equip both sailboats and the buoys marking the course layout with special GPS tracking devices. In an attempt to make the existing functionality of SAP Sailing Analytics available for sailors and sailing clubs everywhere, this thesis deals with the server side challenges of integrating of mobile devices into SAP Sailing Analytics.

The course layout is fundamental for all calculations of metrics and key figures, but in future it may not be available, if instead of relying on tracking providers sailors use personal smart-phones and similar devices to track only their own boats. Therefore, data mining techniques are applied to the domain of regatta sailing in an effort to develop a heuristic that can infer the course layout from the GPS tracks of sailboats. Different promising approaches are presented and evaluated. Also, the design decisions of the implemented software prototype, enabling a first integration of mobile devices, are presented. Both smart-phones in specific – which allow for live tracking of races – and GPS-enabled mobile devices in general are considered. As security concerns and other important aspects are not dealt with due to the complexity and time constraints, the integration can be considered only as the first step in rolling SAP Sailing Analytics out to the masses.

Contents

Indexes

List of Acronyms

AJAX	Asynchronous JavaScript and XML
ANN	Artificial Neural Network
API	Application Programming Interface
app	Smart-phone Application
DBSCAN	Density Based Spatial Clustering of Applications with Noise
DTO	Data Transfer Object
ESN	Electronic Serial Number
GPS	Global Positioning System
GSM	Global System for Mobile Communications
GWT	Google Web Toolkit
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IMEI	International Mobile Station Equipment Identity
IP	Internet Protocol
ISAF	International Sailing Federation
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
MEID	Mobile Equipment Identifier
NAT	Network Address Translation
NMEA	National Marine Electronics Association
POJO	Plain Old Java Object
POSIX	Portable Operating System Interface
QR-Code	Quick Response Code
REST	Representational State Transfer

RIA	Rich Internet Application
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SOAP	Simple Object Access Protocol
TAM	Technical Architecture Modeling
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UI	User Interface
UML	Unified Modeling Language
URI	Uniform Resource Identifier
UTC	Universal Time Coordinated
UTM	Universal Transverse Mercator
UUID	Universally Unique Identifier
VMG	Velocity Made Good
WGS84	World Geodetic System of 1984

List of Algorithms

List of Figures

List of Symbols

α	Tacking Angle
β	Heel Angle
D	Data Set
ϕ	Latitude
δ	Leeway Angle
λ	Longitude
ϵ	Maximum Distance for Epsilon Neighbourhood
κ	Activation
μ	Minimum Number of Points for a Cluster
ν	Maximum Distance for Douglas-Peucker Algorithm
o	Output Value
p	Point
q	Point
σ	Weighted Input Sum
θ	Threshold
$\vec{w}$	Input Weights
$\vec{x}$	Input Values

List of Tables

1 Introduction

> "SAP Sailing Analytics not only helps make the sport more accessible for the fans. It also helps the sailors optimise their strategies, as the teams and their coaches can use the information to evaluate their performance better than ever before." – *Marcus Baur, three-time European 49er Champion, June 2011*

> "This is still a vision, it will still take a while. But where professional tracking systems are used today, in future smart-phones 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*

Through the introduction of modern technology the regatta sailing experience has changed for spectators and competitors alike. For a long time, the medialisation of the regatta sport lagged behind that of other sports. By partnering with the Sailing Team Germany and different sailing series and events, SAP has decided to bring sailing closer to the spectator and support professional sailors. SAP Sailing Analytics is a custom-built web application which takes on the challenge of transforming *Global Positioning System* (*GPS*) tracking data and wind data of sailing races into meaningful metrics, key figures and visualisations.

Currently, SAP Sailing Analytics relies on data from tracking providers that equip both sailboats and the buoys marking the course layout with special GPS tracking devices. The data from these devices is sent wirelessly to the tracking providers, which then pre-process and enrich the data stream before forwarding it to the SAP Sailing Analytics system. In an attempt to make the existing functionality of SAP Sailing Analytics available for sailors and sailing clubs everywhere, and not only at high-profile events, this thesis deals with the server side of integrating mobile devices into SAP Sailing Analytics.

In the progression of this thesis, an introduction into the physics, rules and charac-

teristics of regatta sailing lays the foundation for presenting the current functionality of SAP Sailing Analytics and its architecture. In the following chapter, the value of integrating mobile devices for spectators, sailors and coaches, but also for SAP Sailing Analytics is discussed, from which the functional requirements in the next chapter are deduced. While differing from the current set-up with dedicated tracking providers creates new possibilities, some major draw-backs have to be dealt with.

The positions of the buoys defining the sailing course and the time points at which the sailboats pass these buoys are fundamental for the ranking and calculation of metrics and key figures. Finding and entering these mark positions and passing times by hand is possible, but tedious work that – up to now – was handled by the tracking providers, so that a heuristic to automate at least the first step of this process is the first requirement. This problem is addressed by first introducing, and then applying and evaluating different data mining techniques, utilizing the previously presented information on regatta sailing.

Another result of SAP Sailing Analytics stepping in as its own tracking provider is that the management of mobile devices, race and competitor data, as well as the handling of device-to-server communication moves into the field of responsibility. To achieve a first cross-cut prototypical solution that demonstrates the potential of tracking races with commodity mobile devices, the architectural integration of mobile devices is developed and presented. The focus thereby lies both on smartphones – which are pre-destined candidates for replacing the professional tracking devices due to their integrated sensors and mobile network access – for live tracking of races, as well as the import of GPS tracks in standard formats, created by any kind of GPS-enabled mobile device.

The results of this thesis provide the first step in making SAP Sailing Analytics publicly available as a personal tracking and analytics solution for the sailing sport and so enabling all sailors, coaches and spectators to benefit from the provided analytical insight wherever and whenever they want.

2 Terms, Principles, and Rules of Regatta Sailing

2.1 Introduction

Originally invented for practical purposes, sailing has undergone a continuous transformation while itself transforming civilization. Today, combustion engines dominate the world's oceans, making vessels of all sizes independent of the unreliable source of propulsion unique to sailing craft – the wind – reducing sailing to a recreational activity. For the sake of simplicity, the term *sailboat* is used from now on to represent all kinds of sailing craft, even though of course there are more types of sailing craft than sailboats.

Progression in the principles of boat and sail design has gone so far as to enable sailboats to travel to destinations that lie upwind, and even to sail at speeds *faster* than the speed of the wind. Both these facts may seem counter-intuitive and are addressed in this chapter. By illustrating the restrictions a sailboat underlies in travelling, the following description of regatta course layouts is substantiated. Combined with the terminology that is conveyed in the course of this chapter, the domain knowledge for regatta sailing necessary for later chapters is laid out.

2.2 Sailing Physics and Manoeuvres

2.2.1 Forward Movement through Wind

When reflecting on the motion of a sailboat, the most natural translation that comes to mind may be that of a sailboat heading straight downwind. While this is certainly a valid direction of travel for a sailboat, it seems surprising that it is *not* the direction

in which a modern sailboat travels at its fastest. The explanation for this and other interesting aspects is found by briefly delving into the underlying laws of physics.

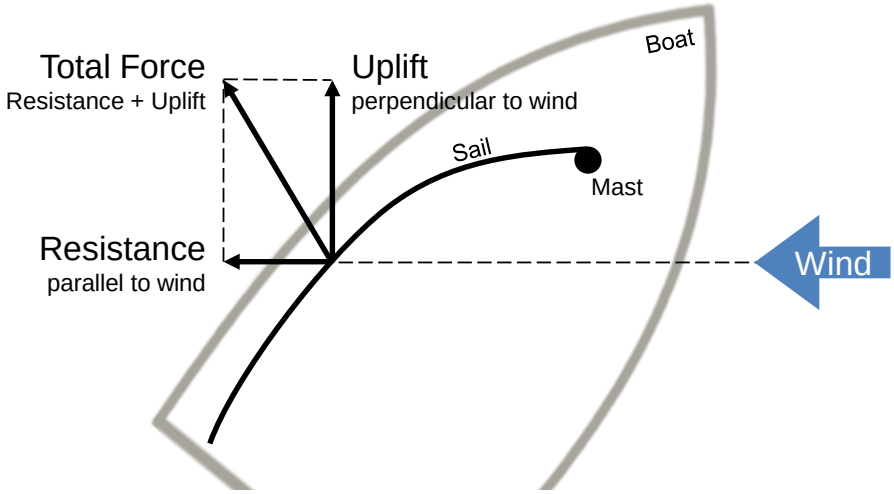


Figure 1: Forces acting on a Sail[1]

The sail of a sailboat functions in the same way as the wing of an aeroplane. It uses the wind[2] to create an *uplift* in addition to the *resisting force*. Different laws come into play to explain how this *uplift* is created (Newton's 3rd law, Bernoulli's principle, and more) which shall not further be discussed here. However, the *uplifting force* is created only if the wind can flow along both sides of the sail. The *total force* is the vector sum of both components, graphically shown through the force parallelogram in figure 1. In itself, it would mainly move the sailboat sideways, with only a small amount of forward movement. Therefore sailboats have a rudder and a daggerboard, which are submerged in the water, thus reducing sideways slip and instead directing the boat forward.

However the *total force* is not completely converted to forward movement, partially causing the boat to slip sideways, and also causing the boat to heel (tilt sideways), as depicted in figure 2 on the following page. The greater the heel angle β is, the less efficient the wind energy will be converted into forward movement. The size of β depends on many factors, among them wind speed and trim (setup of the boat and sail).[3]

[1] *Whidden/Levitt*, Das Segel, 1992, p. 105

[2] The *apparent* wind, to be precise. The exact meaning of this is discussed later on, but it should be noted at this point that this is what the wind arrow in figure 1 represents.

[3] cf. *Goodison*, Laser handbook, 2008, p. 69.

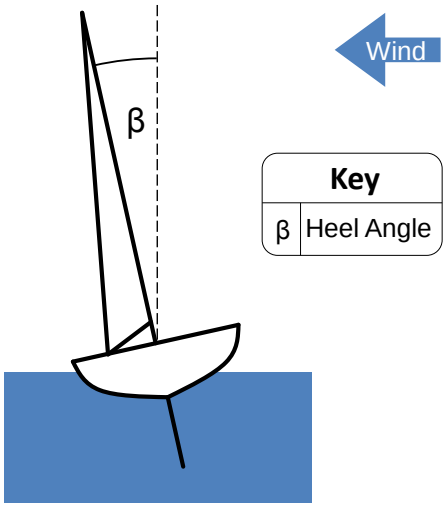


Figure 2: Heel Angle

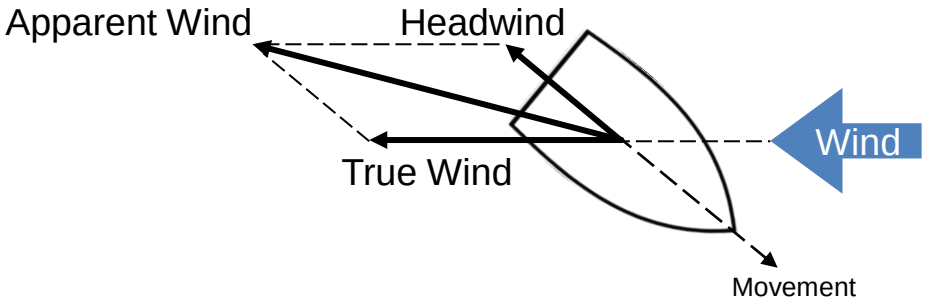


Figure 3: Apparent Wind[4]

Up to this point, the term *wind* has been presented as rather monolithic. However, as figure 3 shows, this was not entirely correct and done only for the sake of simplicity. The following terms should be distinguished:

True Wind: This is the wind actually blowing, and can be directly observed from a motionless reference point.

Headwind: The cause of this wind is the movement of the object (sailboat) itself through the air, and is the same as experienced when riding a bicycle on a windless day.

Apparent Wind: This is the combination, or vector sum, of both the *true wind* and *headwind.* For any observer on the object in motion this is the only directly

[4] *Curry*, Regatta-Segeln, 1949, p. 61

observable wind.

Taking these definitions into account, the observed wind on a sailboat is always the apparent wind. Thus, it is also the speed and direction of the apparent wind which act upon the sail of the sailboat, causing the resisting and uplifting force. As can be gleaned from the force parallelogram in figure 3 on the preceding page, the apparent wind may be greater than the true wind, depending on the direction the sailboat is moving in. This is also the reason, why sailboats may travel at speeds above that of the wind: They create additional headwind on top of the true wind by moving.

2.2.2 Heading and Bearing

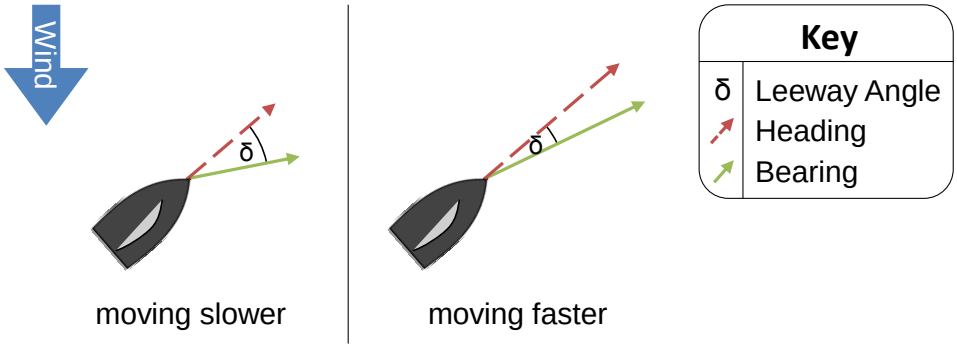


Figure 4: Heading, Bearing and Leeway Angle[5]

Another important distinction shall be drawn between *heading* and *bearing*. These two terms have different meanings in different contexts, and may also be used differently in the context of sailing. The following definition of these terms therefore serves to clarify their meaning in this thesis.

Heading: The direction in which the boat's prow (or more precisely its long axis) is pointing.

Bearing: The direction of the boat's forward movement.

As figure 4 emphasizes, these two do not necessarily have to be the same, although the difference between the two may be unnoticeably small. A sailboat that is headed in the general direction of the true wind always bears slightly further downwind. This is true, because no matter how well the *total force* on the sail is translated into forward movement, a small amount always causes a sideways slippage. This effect

[5] cf. *Goodison*, Laser handbook, 2008, p. 71

becomes less relevant with higher boat speeds relative to the wind speed, resulting in a smaller *leeway angle* δ.[6]

2.2.3 Movement relative to the Wind

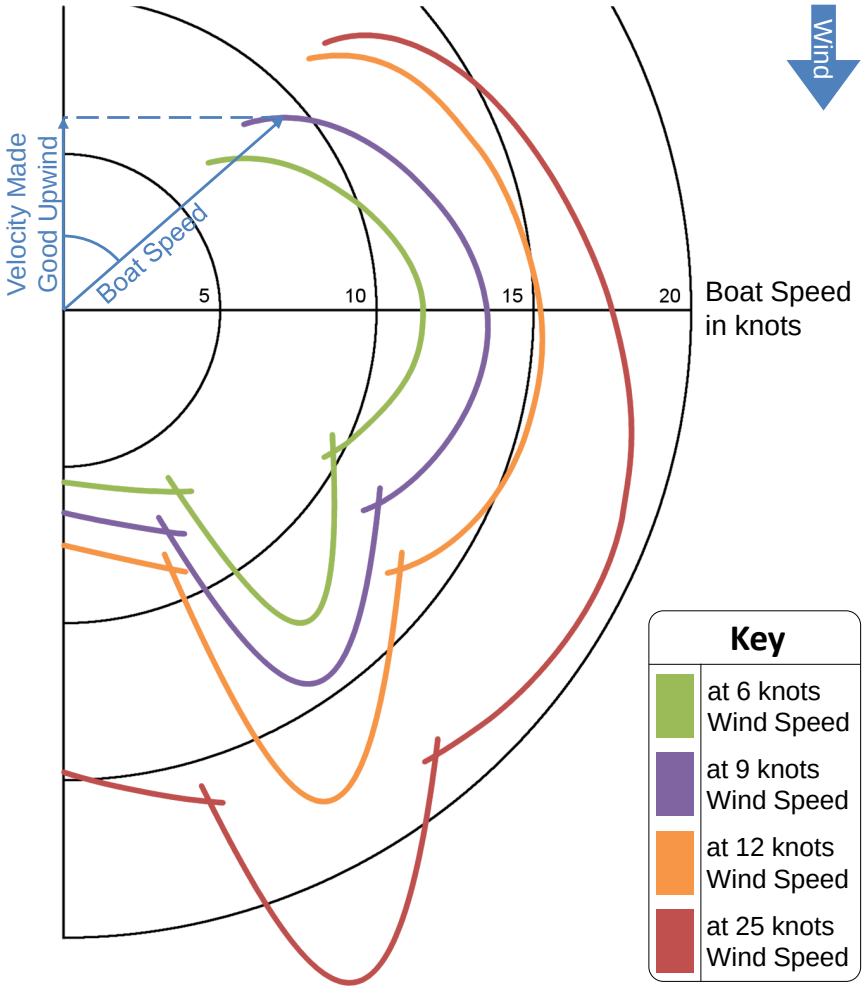


Figure 5: Polar Diagram for 49er Boat Class[7]

Polar diagrams as the one depicted in figure 5 show the highest possible speed at which a sailboat can move, depending on its heading and the true wind speed.[8]

[6] cf. *Goodison*, Laser handbook, 2008, p. 71.

[7] *Bethwaite*, 49er Polars, 2012

[8] To generate such a polar diagram from measured data would require exact measurements of the current *true* wind speed at the current position of the sailboat, and also the perfect trim for that wind speed and angle to the wind. Therefore a measured polar diagram is always inaccurate,

Each polar diagram is distinctive of the boat for which it was created, or of its boat class in case of a one-design boat classes such as the *49er dinghy boat class*[9]. The syntax of a polar diagram is the following:

- The true wind blows from top to bottom.
- A point in the diagram represents both a boat speed and heading relative to the true wind.
  - The boat speed represented by a point is given by the distance between that point and the origin, so that concentric circles connect points representing the same boat speed.
  - A sailboat can be imagined to be resting in the origin, able to rotate about its yaw axis. All points lying on the straight line in the direction of its heading represent that heading.
- All black lines in figure 5 on the preceding page are auxiliary lines.
- The coloured lines can be thought of as measurement curves for true wind speeds. On these curves lie the points that represent the highest possible boat speed for the heading which they represent.
- As a polar diagram should be symmetrical to the wind axis, only 180° need to be shown.

A polar diagram has two advantages when compared to a regular x/y function plot with the heading as the independent variable on the abscissa. On the one hand the diagram can be mentally superimposed by simply turning it to match the true wind direction. On the other hand, and even more importantly, it is very easy to find the heading with the highest upwind (or downwind) *Velocity Made Good*$^{\mathrm{GL}}$ (*VMG*). This is as simple as finding the point on the curve for the given true wind speed that lies furthest to the top (or bottom) of the diagram, as is shown exemplary for the $9\,knots$ curve. If the boat travels at the indicated angle, it travels at the highest possible upwind VMG.[10] Disregarding the bulges at roughly 150° to the wind – which are due to the deployment of a *spinnaker*, a special kind of sail only deployable at certain headings to the wind – many conclusions can be drawn from this polar diagram for modern sailboats in general, some of which validate earlier statements:

as these requirements cannot be fulfilled (at least not continuously). To avoid these problems, polar diagrams can be generated by simulating the behaviour of the sailboat. Provided that there are no errors in the simulation, this gives the highest physically possible boat speeds, which can then also serve as a benchmark for sailors.

[9] This boat class is named after its hull length of roughly 4.99 meters and was designed 1999 by Julian Bethwaite, who also provided the data for the polar diagram in figure 5 on the previous page.

[10] cf. *US Sailing*, IMS Performance Package, 1997.

- A sailboat cannot travel directly upwind, due to the functional principles of sails. However, moving at angles significantly less than 90º to the wind is possible.
- The higher the wind speed, the higher the possible boat speed in any direction is.
- The boat speed is not at its largest when moving directly downwind, as the airflow around the sail is then stalled, and no uplift is generated.
- A boat speed higher than the true wind speed is achievable. For example, a 49er dinghy can move up to twice as fast as the true wind speed.

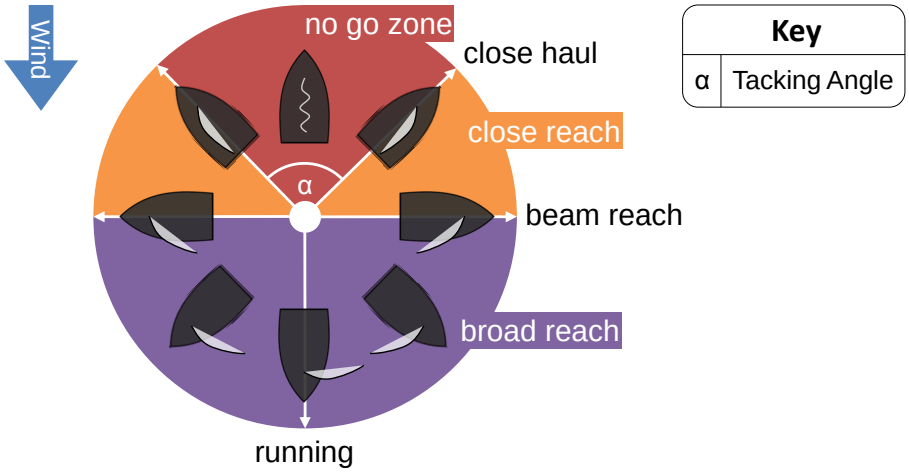


Figure 6: Points of Sail[11]

Figure 6 depicts the so-called *points of sail* – the terms for the directions in which a sailboat can travel relative to the direction of the wind. Only a rather small portion of all possible headings does not result in a forward movement of the sailboat. This was already visible in figure 5 on page 7, the according segment is named *no go zone*. The size of the tacking angle α – the angle between the two highest headings relative to the wind at which the sailboat can travel – depends on the type of the sailboat, and may be as small as 80 degrees for the Laser boat class[12,13]. As soon as the sailboat is headed far enough out of the wind to pick up speed, it is on a *close haul*, followed by several gradations of reaching: *close reach, beam reach* and *broad reach*. Finally, a sailboat can *run* straight downwind.

In relation to the wind, sailboats can move on one of two different *tacks*, as shown in figure 7 on the following page: *starboard tack* and *port tack*. The naming of these

[11] *Mühlbauer*, Richtig Segeln, 2011, p. 24
[12] cf. *Goodison*, Laser handbook, 2008, p. 122.
[13] The Laser is another Olympic one-design boat class. Further details is explained later on in chapter 2.3 on page 11.

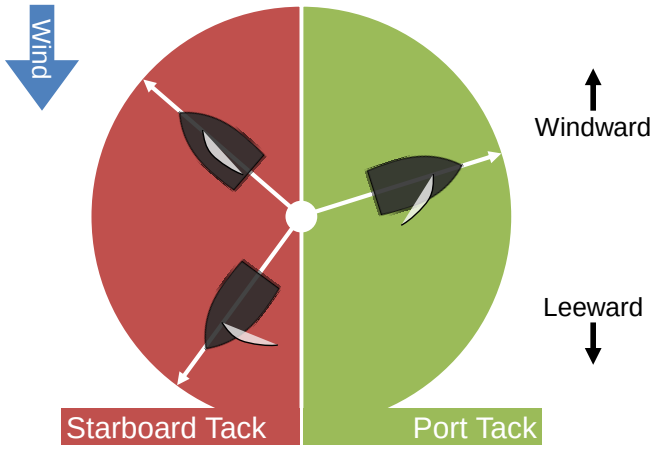


Figure 7: Starboard and Port Tack[14]

two is derived from the nautical terms for right and left, which are *starboard* and *port*, respectively. The nautical term for the windward side of the boat – the side of the boat over which the wind is blowing – determines that *tack's* name. For example, if the wind is coming from *starboard* (from the right side of the boat), then the boat is on *starboard tack*.

2.2.4 Manoeuvres

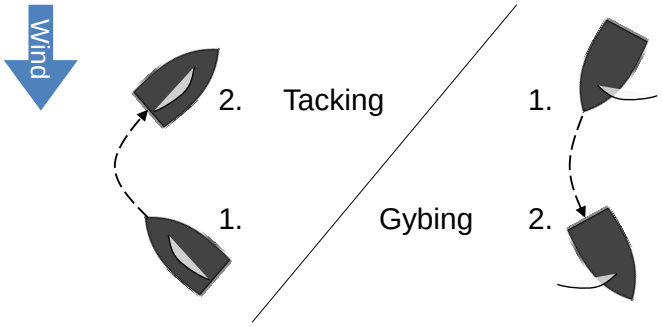


Figure 8: Tacking and Gybing[15]

The term *tack* is overloaded, having one meaning as a noun (as just explained), and another as a verb. The verb form is also where *tacking angle* is derived from, as *tacking* is one of the two basic sailing manoeuvres, the other one being *gybing*. By

[14] *Mühlbauer*, Richtig Segeln, 2011, p. 24
[15] cf. *Goodison*, Laser handbook, 2008, pp. 42 ff.

performing one of these manoeuvres a sailboat changes from one tack to the other. They are displayed in figure 8 on the previous page. When *tacking* from the highest possible heading to the wind on starboard tack to the highest possible heading on port tack, the boat turns over an angle which is the *tacking angle.* Two other basic manoeuvres are *heading up* and *bearing away*, meaning a turn of the boat further into the wind or further out of the wind, respectively.

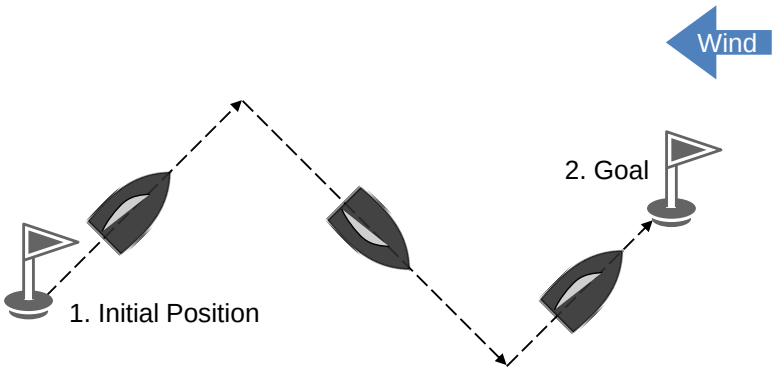


Figure 9: Beating to upwind Destination

As travelling directly upwind is not possible, sailboats have to be moved to destinations which lie within the tacking angle, as seen from their initial position, by employing a series tacks, called *beating.* To reach the goal as fast as possible, the sailboat has to sail at close haul, with the heading providing the highest upwind VMG, which can be found as described earlier in a polar diagram. The goal in figure 9 could certainly have been reached with only one tack, however straying to far from the direct course might be unwise in case of changes in wind, which is revisited in chapter 2.3.

2.3 Regatta Sailing

2.3.1 Match and Fleet Racing

Now that the principles of sailing are understood, a brief introduction into the rules governing regatta sailing ensues[16]. These rules, including the course layouts, make up most of the domain knowledge necessary for chapter 7 on page 52. The set

[16] A *regatta* is what one would call a race in other forms of sport. The term may also be used to described a set of sailing races, taking place at the same venue in parallel or consecutively.

of rules taken into consideration is that maintained by the *International Sailing Federation*[GL] (*ISAF*), as it develops the *Racing Rules of Sailing* relevant for all major sailing competitions, such as the Olympic Games.[17]

In general, there are two kinds of regattas in which sailboats can compete:

Match Racing: Here only two boats compete against one another, the winner being the boat to first cross the finish line.

Fleet Racing: This is the more common type of regatta, with more than two sailboats competing at once. Depending on the kind of boat class, two different types of fleet racing exist:

One-Design: All distinguishing factors except for the sailor's ability are eliminated by using the same design, same sail area and other identical technical details for all boats in one class.

Handicap: The finish times of the boats are adjusted depending on a predetermined *handicap* for each boat, so as to compensate for technical advantages some boats may have since they are *not* one-design.[18]

2.3.2 Course Layouts

Regatta courses are laid out through certain waypoints, called *marks*. The *Racing Rules of Sailing* give a very tangible yet generic definition of how a course has to be sailed:

> "A boat shall start, sail the course described in the sailing instructions and finish. While doing so, she may leave on either side a mark that does not begin, bound or end the leg she is sailing. After finishing she need not cross the finishing line completely. A string representing a boat's track from the time she begins to approach the starting line from its pre-start side to start until she finishes shall, when drawn taut
>
> 1. pass each [waypoint] on the required side and in the correct order,
> 2. touch each rounding [waypoint],
> 3. pass between the marks of a gate from the direction of the previous [waypoint].

[17] cf. *International Sailing Federation*, What is ISAF, 2012d.
[18] cf. *International Sailing Federation*, Fleet Racing, 2012a.

She may correct any errors to comply with this rule, provided she has not finished."[19,20]

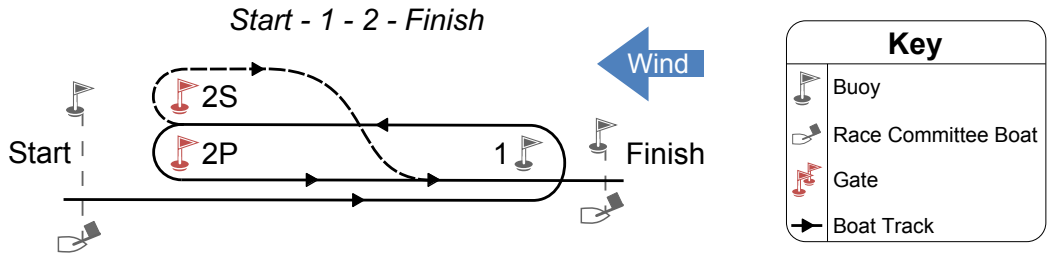


Figure 10: Windward-Leeward Course[21]

As this definition includes some new terms, it is best explained with the help of a simple, exemplary regatta course, as depicted in figure 10, using the following syntax:

- The course is defined by *marks,* which may be either
  - a *buoy*,
  - or a race committee boat.
- A single mark, or two marks together make up one *control point.* The different types of control points are
  - the simplest control point, consisting of only a single mark,
  - the *start line,* delimited by two marks,[22]
  - the *finish line,* also delimited by two marks,
  - and a *gate*, which is made up of two marks (S for the starboard, P for port mark).[23]
- The sequence in which the marks have to be visited after starting and before finishing is stated at the top of the course diagram. The specific occurrence of a control point in this sequence is referred to as a *waypoint.* As a control point may occur more than once in this sequence, several waypoints belonging to the same control point may exist.

[19] *International Sailing Federation*, Racing Rules of Sailing, 2012c, p. 19.
[20] Due to the imprecise nature of the term *mark* used by the Racing Rules of Sailing to represent different things, for this thesis the more precise terms *mark, control point or waypoint* as used in SAP Sailing Analytics are distinguished. For this reason, the term *mark* has been replaced in this quote as indicated.
[21] cf. *International Sailing Federation*, Racing Rules of Sailing, 2012c, p. 141.
[22] This is the most common type of start. A *gate start*, as described in *Royal Yachting Association*, Gate Starts, 2010, works differently.
[23] *International Sailing Federation*, Racing Rules of Sailing, 2012c, p. 8.

- The string mentioned in the ISAF definition shall visualize how this sequence has to be executed by a sailboat. Its depiction as a black line is named *boat track,* even though differs from a real boat track, as
  - it shows the passing of a single buoy from only one side, whereas they may be passed from either side,
  - a sailboat can choose only one of the two possible tracks after passing through a gate – one of which is therefore depicted as a dashed line to clarify that the choice of track is exclusive.
- A *leg* is a stretch of the course between two waypoints. It may be identified by consecutive numeration (1stleg, 2ndleg, etc.) by counting the legs from start to finish, or by its relation to the wind (upwind leg, reaching leg, etc.).[24]

Such a *windward-leeward course,* named after its legs, is always used for *match racing*, albeit with several repetitions of the two legs.[25]

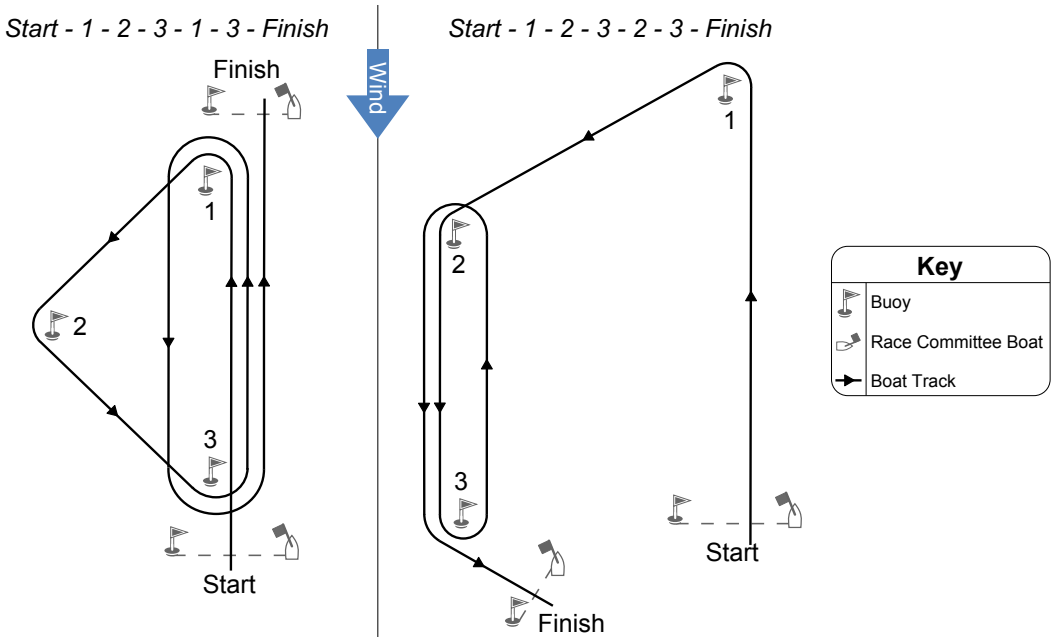


Figure 11: Triangle and Trapezoid Courses[26]

Other typical course layouts are presented in figure 11. They are also usually laid out, so that the first leg is upwind, but have additional legs at different points of sail. The *triangle course* is often used at the Olympic Games, thus also often referred

[24] cf. *Saltonstall*, Race training manual, 1990, pp. 81 ff.
[25] cf. *International Sailing Federation*, Match Racing, 2012b.
[26] cf. *International Sailing Federation*, Racing Rules of Sailing, 2012c, pp. 142 f.

to as the *Olympic Triangle.* The *trapezoid course* is also useful for large events, as the buoys 2 and 3 can be used for two courses at once – which then are mirror-inverted to each other – on which regattas can then be held in parallel, mingling in the middle.

2.3.3 Playing Board of a Regatta

The *playing board* of a regatta is the combination of basic sailing principles and course layouts. When looking at a single leg of a course, there is a virtual playing board in which all sailing action naturally takes place. As shown with the help of the polar diagram in figure 5 on page 7, there exists an ideal heading to the wind to make maximum upwind VMG for a given type of sailboat.

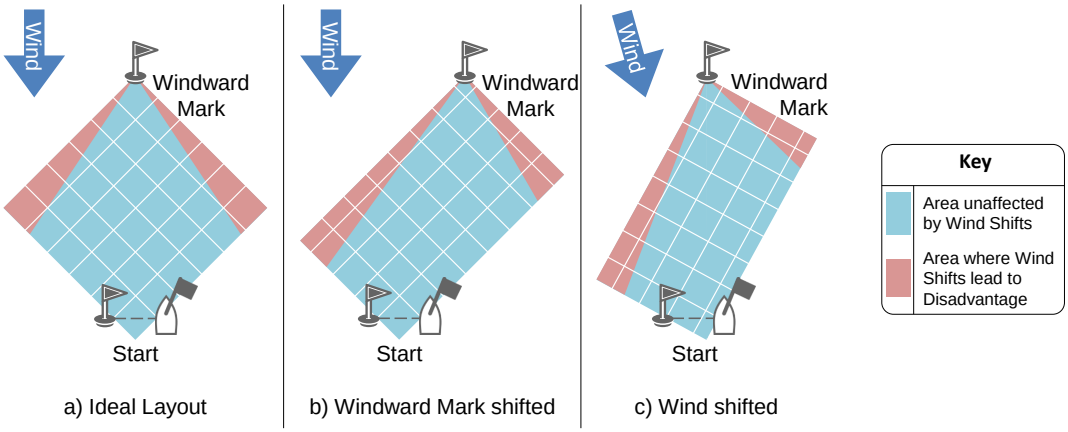


Figure 12: Playing Board Variations for an upwind Leg[27]

If this ideal angle is thought of as being at 45° to the wind (which is roughly true for the *49er dinghy boat class*), the playing board for an upwind leg might – in the simplest case – look like playing board *a)* in figure 12. The edges of the coloured area and also the white lines are all at an angle of 45° to the wind, creating a chequerboard grid, along the lines of which sailboats should move (or move parallel to), to reach the windward mark as fast as possible. The blue area indicates the area in which sailboats should move to be immune to wind shifts (at least to a certain degree). No boat should sail outside of the coloured area, as it would mean having to go on a heading with less upwind VMG at some point to pass the windward mark, thus losing time.

[27] *Baur*, Streckbug Segeln, 2012b

The two other playing boards, *b)* and *c)*, are both different in one aspect from *a)*. For *b)*, the windward mark has been moved, creating an elongated playing board – if moved further and further, it would eventually represent a reaching leg. For *c)*, the wind has shifted counter-clockwise, turning the white lines and boundaries of the coloured area.[28] Playing board *c)* can also be thought of as a special case of *b)*, with a start line not perpendicular to the wind. They are otherwise identical if one turns *c)* clockwise, so that its wind axis aligned with *b)*.[29,30]

The important thing to note is that, regardless of the specific wind situation or course layout, an imaginary bounded area – a playing board – exists in which sailboats move across the water along an imaginary grid. The question whether the sailors actually manage to steer their boats at this ideal angle to the wind is dealt with later. The playing board changes over time, as the wind may shift, and as marks drift over time. Also, the playing board looks different for different legs of the course which are not upwind legs. The effect of these changes has already been indicated in figure 12 on the previous page through *b)* and *c)*.

2.3.4 Race Committee

A race committee is similar to the set of referees or umpires in other sports. It consists of several people in charge of overseeing a sailing race from a specific time prior to the start, until all boats have finished the race. During this time, the race committee is responsible for ensuring orderly conduct and observation of the rules in place, having different measures to sanction improper behaviour, ranging from imposing a penalty turn – requiring the offender to perform a tight 360° turn – up to disqualification.

2.4 Conclusion

The manner in which sailboats can move, along with the understanding how they can move fastest was first presented, followed by a summary of the important rules and aspects of regatta sailing, concluded by the notion of an imaginary playing board.

[28] The wind strength also changes the playing board, as the angles for the highest upwind (downwind) VMG may change. The direction and strength of an existing water current also has an influence, but it usually has less impact than the wind.

[29] cf. *Baur*, Streckbug Segeln, 2012b.

[30] cf. *Baur*, Der Kreuzdiamant, 2012a.

All of this combined serves as the vocabulary and basic insight into regatta sailing necessary for comprehending the next chapters – especially chapter 7 on page 52 – as they progressively focus on the specifics of the regatta sailing domain.

3 SAP Sailing Analytics

3.1 Introduction

A spectator, standing at the Channel coast in the Weymouth bay, trying to follow the Olympic medal race taking place some hundred metres off the coast with binoculars.

A commentator, relaying what information he can gain from live camera feeds on boats and helicopters as well as from radio transmissions from helpers out on the water to spectators on-site and on television.

A sailor and his coach, trying to reconstruct the order of events of the unsuccessful race a few hours later, to identify opportunities for improvement.

These three stereotypical user stories depict the setting that SAP Sailing Analytics has evolved in, representing different problems sailing as a sport presents for competitors, coaches, commentators and spectators. This chapter first shows the purpose of SAP Sailing Analytics, and how it is put to use, which is necessary for understanding the relevance of integrating mobile devices. Afterwards, a short summary of current partners and data providers is given, highlighting problems and drawbacks of this setup. Finally, a high-level overview of the current system architecture and design decisions provides the technical details essential for the technical chapter 8 on page 76.

3.2 Purpose and Usage

3.2.1 Technical Advances in the Coverage of Regatta Sailing

> "Historically, it has been almost impossible for spectators to follow a sailing race. It is a costly challenge to get telling images of the water, and it is a challenge to make the spectators understand the dynamic nature of the game. But with the advances in GPS tracking technology, cutting edge analytics, and 3D animations, it is now possible to dive into the action of a sailing race, and better understand what makes a boat win." – *Marcus Baur, three-time European 49er Champion, June 2011*

Regatta sailing is not a popular sport compared to others such as European football. The problem is not caused by the complexity of the rules (although there are plenty more than shown so far). Rather, this is due to the difficulties inherent in a water-bound sport with widespread race courses, combined with the fact that the strategic aspect is what makes sailing regattas interesting.

Figure 13: Helicopter Footage of 2012 Olympic Race[31]

Growing budgets and modern technology have already enabled high quality aerial and water level live footage, as shown in figures 13 and 14. Both show the same scene, the rounding of the 2nd mark, at roughly the same point in time. It is plain

[31] *International Olympic Committee*, Laser Men Medal Race, 2012

Figure 14: Footage of 2012 Olympic Laser Class Medal Race at Water Level[32]

to see how the chosen angles affect the balance of high-level insight as opposed to specific detail. Aerial footage can provide an overview, giving a better idea of the relation of the current happenings to the course layout and progress of the race as a whole, while water level footage provides more detail and allows identification of individual sailboats. Through the combination of both and sufficient experience, it is possible to mentally gain a good overview of the race. However, the chosen footage represents unusually ideal circumstances of the medal race at the Olympic games, captured close to the shore in ideal weather conditions. The majority of sailing races does not have such excellent video coverage, for example during events such as the Kiel Week, where several races of less significance take place at the same time, even with overlapping course layouts.

For sailors and coaches, analysis in retrospect using video footage also proves difficult, as the footage does not display all sailboats at all times, and as navigation through such footage is time-consuming. In other sports, specific metrics and figures are used to aggregate and abstract the progression of the game or race, and make room for analysis, which can be used for commentary purposes or simply displayed alongside the video footage. In European football, a simple example is the number of goals each team has shot.

[32] *International Olympic Committee*, Laser Men Medal Race, 2012

3.2.2 Components

Race Viewer

Figure 15: Main Areas of the SAP Sailing Analytics Race Viewer[33]

Total rank	Competitor	Name	R3	L1	L2	Gap Change [s]	VMG [kts]	Rank Gain	L3
1	DEN8964	SAUGMANN+GORGE	10.00	11	9	-6	7.92	2	
2	DEN9062	BOJSEN-MOLLER+BOJSEN-MOLLER	13.00	18	15	-11	8.31	3	12
3	GER9090	KELLNER+SCHOELER	2.00	3	2	-13	9.18	1	
4	GER9054	BOGACKI+DEHNE	6.00	5	4	-12	9.05	1	

Figure 16: Regatta Leaderboard[34]

GPS trackers and additional sensors have – in recent years – provided an opportunity to smooth out the inherent disadvantages of regatta sailing. By gathering GPS tracker readings on the position of sailboats and marks, it is possible to visualize sailing races in progress in a new manner that provides a good overview and does justice to the focal point of sailing races: strategy. SAP Sailing Analytics provides commentators, spectators, sailors and coaches with a web interface to follow and analyse live or passed regattas. The default *User Interface* (*UI*), named *Race Viewer*[35], is shown in figure 15. It consists of three main areas: *Time Slider*, *Leaderboard*, and *Map*. These are briefly described in the following paragraphs.

[33] *SAP Sailing Analytics*, 505 World Championship, 2012, at 18:12:38
[34] *SAP Sailing Analytics*, 505 World Championship, 2012, at 18:16:23
[35] The application can be accessed at `http://sapsailing.com`

Located at the bottom, the *Time Slider* indicates the progression of the race and may be used to navigate through the already tracked parts.

The table on the left hand side is called *Leaderboard*, and contains one row for each sailor in the regatta and a variable number of data columns, which are either associated with the regatta as a whole or a specific leg. In figure 16 on the previous page part of such a Leaderboard is displayed with the detailed columns for the third leg expanded, among them the gap change in seconds to the leader on that leg, the average VMG along the line connecting the two marks bounding the leg, and the gain or loss in rank.

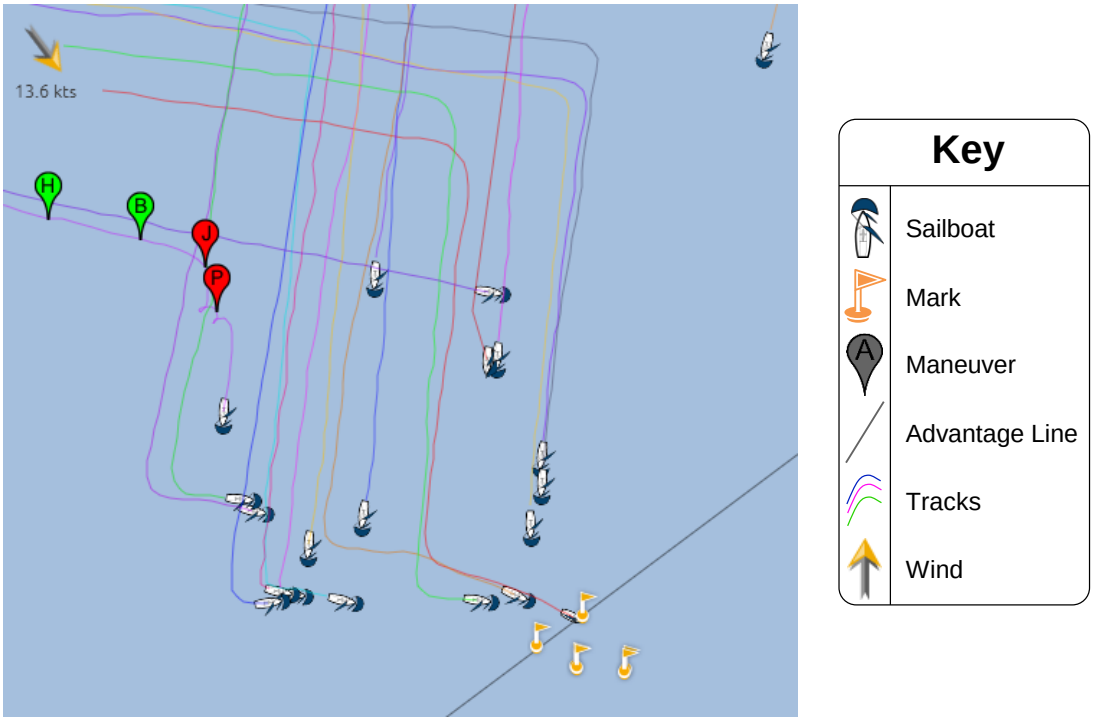


Figure 17: Map Visualization of Regatta[36]

The *Map,* depicted in figure 17, shows a moveable and zoom-able visual representation of the positions of boats and marks, as well as the wind direction and strength. Further elements on the map include the *advantage line*, a virtual line perpendicular to the wind through the position of the sailboat ranked first, which is internally also used to calculate the windward distance to that sailboat. Also, tail lines representing the most recent parts of the sailboats tracks are displayed, and can be enriched through the detected manoeuvres – depicted as red and green markers depending on the tack they were performed on, with a letter representing the manoeuvre: *J* for

[36] *SAP Sailing Analytics*, 505 World Championship, 2012, at 18:15:40

gybing[37], T for tacking, B for bearing away, H for heading up, and P for a penalty turn.

Administration

In addition to the Race Viewer, a UI for administrative purposes exists. This includes the management of configuration data, such as maintaining regatta and leaderboard configurations. By doing so, a real-world event, such as the Kiel Week, attains a representation in SAP Sailing Analytics through corresponding domain model entities. Also, the connection to the tracking providers and regatta operators, described in further detail in the next section, is dealt with in the administrative UI. Existing race definitions of the tracking provider can be mapped to events defined within SAP Sailing Analytics, triggering the import of existing data for past races, and opening an input channel for future races with data arriving whenever it is available, piece by piece.[38]

3.3 Data Providers and Regatta Operators

SAP collaborates with different partners that provide the necessary tracking data. These providers are charged with managing the physical tracking devices, matching them with the correct sailboats and marks, collecting the data coming from these devices and multiplexing it over a unified channel to SAP Sailing Analytics. The technologies employed vary from small and cheap GPS trackers with *Global System for Mobile Communications*[GL] (*GSM*) wireless units, to larger and more expensive units transmitting across amateur radio bands, requiring receiving stations to be placed in the vicinity. In all cases, the devices employed currently only transmit positional data. Handling of master data regarding sailors, sail numbers, teams, nationalities and so forth is also dealt with by the currently partnered with tracking providers. Further details regarding the technical aspects of these connections are presented in chapter 3.4.2 on page 25.

While tracking the sailboats is the major source of data, wind data certainly ranks second. Even though information on the wind can be inferred from course layouts

[37] Two different spellings exist for this manoeuvre: gybing (gybe) and jibing (jibe).

[38] Technically, importing past and live events does not differ at this point in time. When selecting a race that is already fully tracked, an input channel is also opened and the existing data is streamed in via this channel immediately.

and boat tracks – the reasons having been given in chapter 2 on page 3 – the best source is of course direct measurement. As wind conditions may vary locally, wind measurement devices are placed on vessels, which stay close to the bulk or most interesting subset of sailboats. These vessels are also tracked via GPS, so that on the one hand the position and therefore significance of the wind measurements toward a specific race can be determined. On the other hand, the true wind has to be deduced from the measured apparent wind when moving, for which information on the speed and direction of movement is needed. This data is similar to that gained from the tracking providers, but is sent directly from the various vessels to SAP Sailing Analytics, without an intermediary provider acting as broker.

Apart from sensor data, software solutions are already in place for the management and operation of many events, including competitor lists, fleet assignments, boat class to race course area assignments and of course scoring and ranking. While SAP Sailing Analytics primarily uses the figures computed from incoming sensor data, these may differ from reality. An example is a post-race disqualification of a sailboat through the race committee, changing the score calculated for that boat. As such existing systems are usually considered to be the single source of truth, import capabilities exist, so that in the scoring process the data that is considered true by the regatta operator is used.[39]

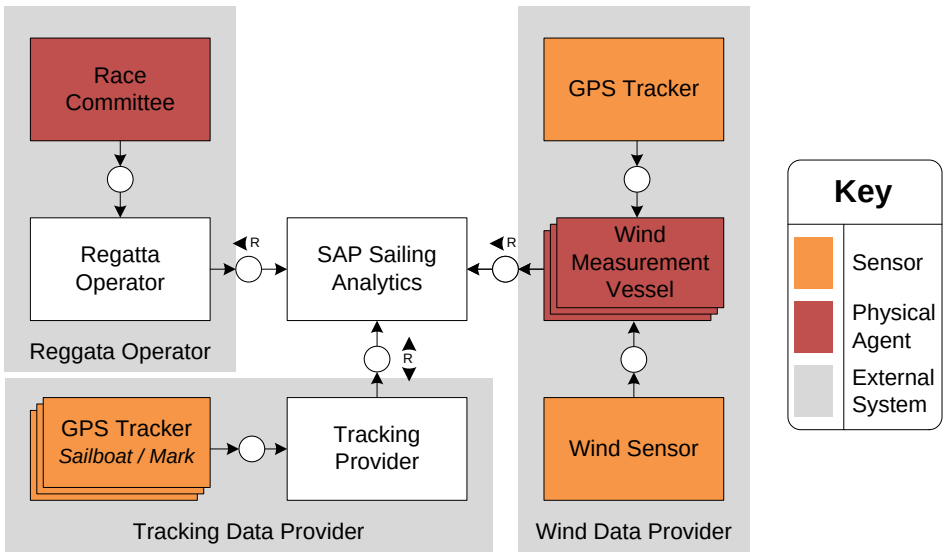


Figure 18: Data Providers and Regatta Operators

The diagram in figure 18 sums up these connections to data providers and regatta

[39] cf. *Uhl*, SAP Sailing Analytics Architecture, 2012.

operators visually. The graphical syntax is that of a Block Diagram in the SAP *Technical Architecture Modeling* (*TAM*) standard, the relevant details of which are described in appendix A on page 95. This gives an overview over the data channels SAP Sailing Analytics depends on to function, and where these come from. The diagram is modelled for one specific sailing event, which in consequence means there is only one regatta operator and tracking provider. In other words: different regatta operators and tracking providers exist, but for one event only one of each is relevant.

3.4 Current Architecture

3.4.1 Overview

SAP Sailing Analytics is built following a three tier architecture, with the front-end on the client tier responsible for the UI, the application logic in the middle tier, and a database as the back-end. Applications with a web front-end are beneficial for reaching large number of users, while the chosen technologies for middle tier and back-end are able to scale to such growing numbers of users.

Figure 19 on the next page represents the entire SAP Sailing Analytics architecture. It also adheres to the graphical syntax of the Block Diagram in the SAP TAM standard. The boundaries of what is considered part of SAP Sailing Analytics, as opposed to external systems relevant for the architecture, are indicated through a grey shading. The remainder of the diagram is explained piece by piece through the following paragraphs.

3.4.2 Server Side Architecture

All of the server side application logic resides within an *Equinox OSGi container,* which is comparable to a traditional Java application server. The concept of *OSGi*$^{\text{GL}}$ and its benefits are briefly described in the following paragraphs.

OSGi is a framework specification for modular Java applications, with the goal of mending the shortcomings of the traditional Java modularization, dependency and versioning system. This is achieved by introducing the concept of *bundles*, which are reusable modules. Essentially, a bundle is a regular *.jar* archive with

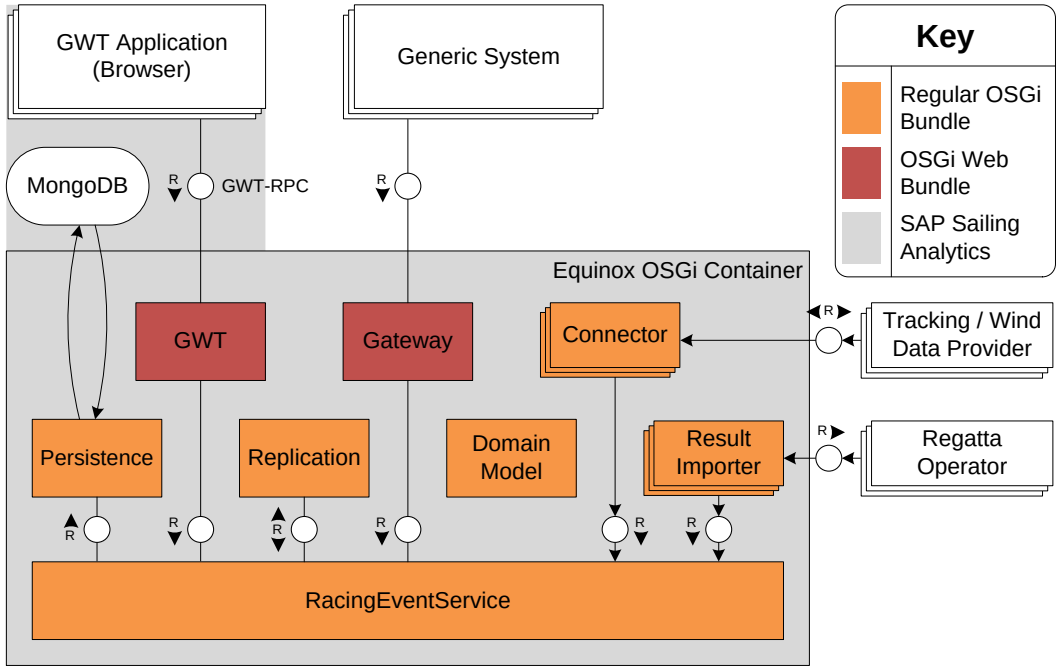


Figure 19: SAP Sailing Analytics Architecture[40]

additional metadata in its manifest file, giving the bundle a name and version, and also explicitly defining dependencies to other bundles on a package level. An OSGi framework manages the classpath individually for each bundle, making visible only the external packages that satisfy the name and version of the declared dependencies of that bundle. In the manifest, the packages to be exported are also explicitly defined. Only these can become visible to other bundles, should they define a matching dependency. This enables a clean application design, as only packages containing *interfaces* can be exported, hiding the implementing classes.

Bundles can then offer and consume functionality dynamically by using the OSGi service registry. Here, implementations for certain interfaces can be registered, offering a method for publishing and accessing *Plain Old Java Objects* (*POJOs*) as services. To complete the dynamic nature, bundles can also listen for services, being able to react to newly available or now unavailable services.[41] By coupling bundles in a loose manner and yet providing strict dependency declarations, bundles can be exchanged on-the-fly. The reusability of components, dynamic updates of

[40] The diagram is a visual representation of the architecture described textually in *Uhl*, SAP Sailing Analytics Architecture, 2012.

[41] cf. *Rubio*, Pro Spring dynamic modules, 2009, p. 2 ff.

bundles and especially reduced complexity are the major benefits of using a OSGi framework.[42]

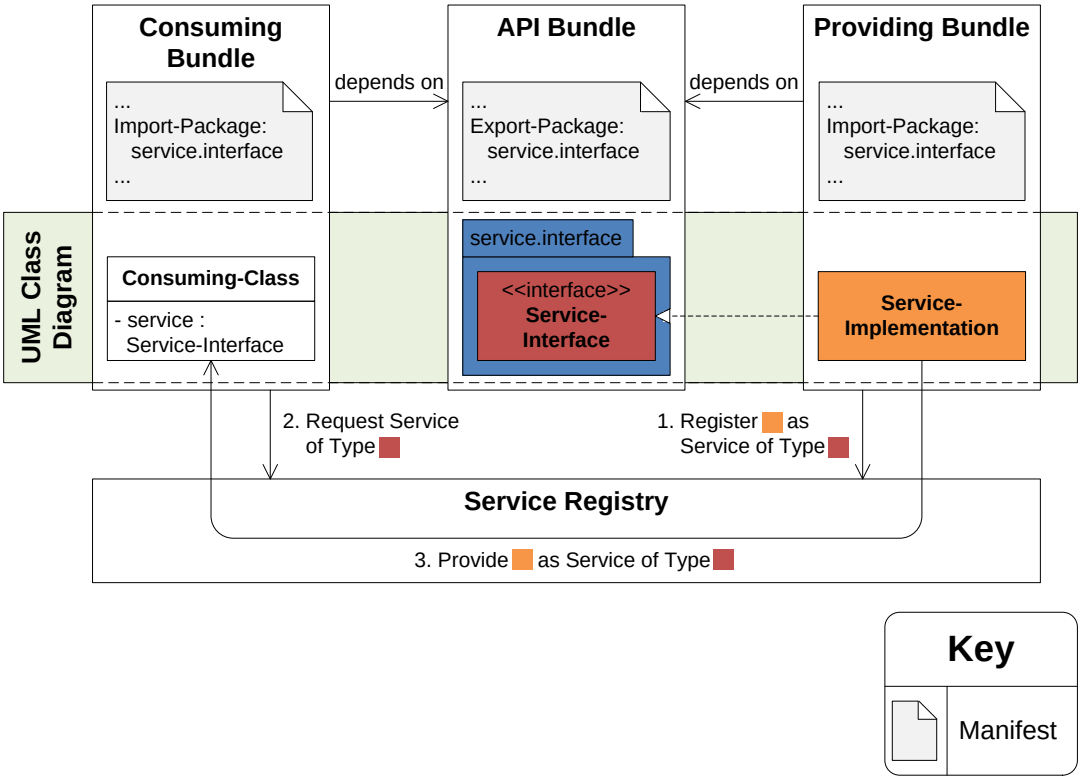


Figure 20: OSGi Service Registry

Figure 20 shows an example of how the service registry can be used. It is a diagram including several levels of abstraction, one of them being the *Unified Modeling Language* (*UML*) class diagram on the horizontal centre axis. The *Application Programming Interface (API) Bundle* defines a *Service-Interface*, which is implemented by the *Service-Implementation* class in the *Providing Bundle,* which also registers its class as an implementation of the service in the service registry. When the *Consuming Bundle* now requests a service that implements the *Service-Interface*, the service registry can supply it with the previously registered class. The indirection ensures that bundles providing and consuming services need not be aware of each other, but instead are only coupled through a common interface. In this specific example, the two outer bundles depend on the presence of the *API Bundle*, which the OSGi container knows from their manifest files. Through the automatically generated dependency graph, an OSGi container manages the life-cycles of the bundles it holds to ensure only bundles with resolved dependencies are active.

[42] cf. *OSGi Alliance*, Benefits of Using OSGi, 2012a.

An OSGi container is a host for bundles, and implements the OSGi framework specification. Therefore, bundles may make use of framework features, such as the OSGi service registry. The OSGi implementation in use for SAP Sailing Analytics is *Equinox.* To enable scalability, several SAP Sailing Analytics *Equinox OSGi* container instances exist, possibly distributed across different physical machines, with a load balancer acting as a single point of contact for clients. None of this is shown in the diagram for the sake of conciseness.

Among the bundles depicted in figure 19 on page 26 is the central bundle providing the *RacingEventService*[43], which exposes the core functionality of SAP Sailing Analytics. This functionality is accessed by the *Gateway* and *GWT* bundles, the latter of which is covered in chapter 3.4.3 on the following page. The Gateway bundle acts as an intermediary to any kind of system requiring different input or output formats, such as export and import of GPS track files, or an RSS feed of the newest race results. Both of these bundles are *OSGi web bundles*, and as such expose endpoints via web servlets managed by a servlet container within the Equinox container.

Data input channels are handled by the *Connector* and *Result Importer* bundles, which provide matching endpoints for systems of *Tracking / Wind Data Providers* and *Regatta Operators*, as discussed in chapter 3.3 on page 23. By isolating the details of connectivity to specific partners in individual bundles, the core application is independent of these data sources. As already mentioned, most master data – sailors, sail numbers, teams, nationalities and so on forth – is currently provided by tracking providers. It is mapped to the SAP Sailing Analytics domain model through so-called *Domain Factories,* which are partner-specific and retain enough knowledge to map incoming data to the correct domain objects they previously instantiated.

The *Domain Model* bundle provides no functionality via an OSGi service, but it provides the domain classes and interfaces used across the entire application. While the *Replication* bundle enables automatic propagation of race data between several SAP Sailing Analytics instances, a separate bundle exists to handle *Persistence.* The database backing this bundle is the document oriented *MongoDB*, which provides schema free access to *JavaScript Object Notation* (*JSON*) documents, and also regular files, while supporting replication across multiple instances.

Considering all channels extending over the boundaries of the Equinox Container, both channels extending from the external agents located above the container specify an incoming access direction. This means that the communication over these

[43] This is not the name of the bundle itself (which has the not-so-expressive name of *Server* bundle), but for the sake of simplicity it has been used as such in the diagram.

channels is triggered by the external agents, even though data subsequently flows in both directions. This is different for the external agents located to the right of the container. The Tracking / Wind Data Provider channel specifies a bidirectional access direction, as the communication is first initialized by SAP Sailing Analytics, but the subsequent sending of data packets for live scenarios is initiated by the provider. For the Regatta Operators, this is simpler, as the import of results is always triggered from within SAP Sailing Analytics.

3.4.3 Front-end Technology

SAP Sailing Analytics is developed in Java from end to end. For a web application, this can be considered unusual and is made possible through the usage of the *Google Web Toolkit* (*GWT*). GWT enables the development of both client and server code in Java by acting as an abstraction layer and generating the JavaScript code necessary for the client application, which runs in a browser. This is beneficial as it relieves developers from switching between different programming languages within one project, and also allows sharing code between client and server with the benefits of type safety.

As a toolkit for *Rich Internet Application* (*RIA*), GWT makes *Asynchronous JavaScript and XML* (*AJAX*) an omnipresent technique in GWT applications by simplifying its use. A proprietary *Remote Procedure Call (RPC)*[44] channel is provided, which allows the asynchronous exchange of serializable objects, often referred to as *Data Transfer Objects* (*DTOs*). Therefore, a GWT application behaves more like a desktop application than a traditional web application, as normally no page refreshs are necessary during a user session. The GWT bundle in figure 19 on page 26 – a web bundle as mentioned before – exposes the functionality of the *RacingEventService* via this proprietary RPC channel to the client side browser application.

3.4.4 Additional Design Decisions

An architectural design decision made in the early stages of SAP Sailing Analytics was to store only facts in memory – for example sensor readings provided by tracking

[44] While Java is an object oriented programming language, still the term RPC is used instead of the object-oriented counterpart *Remote Method Invocation* (*RMI*). The reason for this is that normally a single *RemoteServiceServlet* acts as the endpoint for all communication between client and server, enforcing a pseudo-procedural programming style for this one class, which then can of course further delegate, resuming object-orientation.

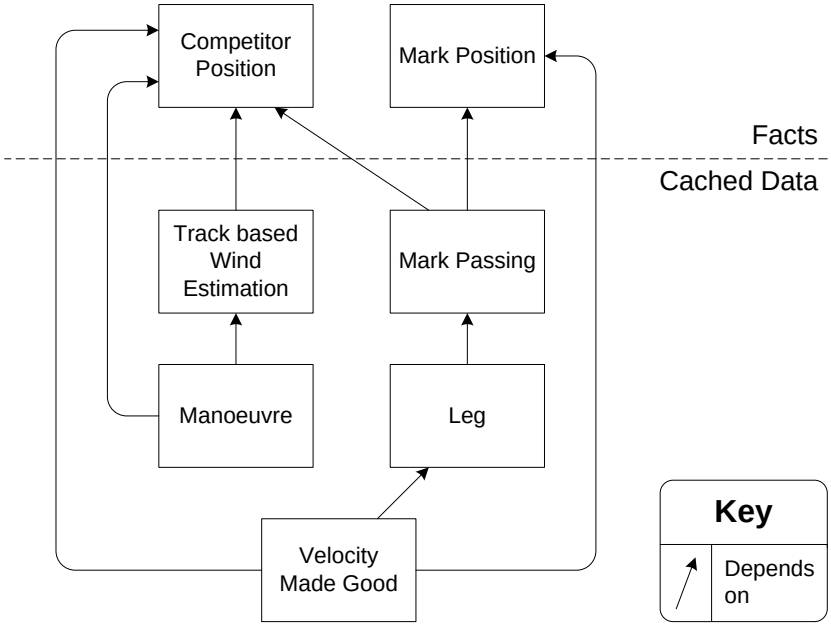


Figure 21: Dependency Graph of Facts and Cached Data

providers – and calculate all derived figures, such as wind estimation or manoeuvre detection, on the fly for every client request. The reason for this is that SAP Sailing Analytics was originally created solely for live scenarios, where facts arriving from tracking providers may arrive out of chronological order, or be rewritten at later points in time. Changes to these basic facts invalidate a majority of the calculated data. This can be visualised through a dependency graph, as figure 21 shows for a small part of the application. For example, if a GPS tracker loses connection during the race and reconnects, it may feed the missing data into the system. This causes a cascade of further changes, as both the track based wind estimation and mark passings[45] have to be re-evaluated. Other figures, in turn, depend on these now newly evaluated figures. One is the manoeuvre detection, as the heading relative to the wind before and after is important to decide on the manoeuvre type. Also, determining which leg a competitor is on depends on thee mark passings. The lowest level in the graph is VMG, which is the vectorial part of the boat speed in direction of the line connecting the two marks bounding the current leg, so that changes to both the current leg as well as the corresponding mark positions force revaluation.

As some figures are expensive to compute, caching of computed figures has been introduced in some cases whilst conceding that this introduces new problems, namely

[45] This is currently done by the tracking providers, and not implemented in SAP Sailing Analytics.

complex invalidation mechanisms. Mostly, client requests are fulfilled with the most recent calculated result for a figure, while updates of caches resulting from changes to facts are handled asynchronously in the background. To keep the caching logic as simple as possible, domain objects are kept immutable wherever possible. This has benefits when considering replication, or displaying figures in the client UI.

3.5 Conclusion

In this chapter the functional and architectural fundamentals of SAP Sailing Analytics were presented after reasoning out its relevance for regatta sailing. At certain points, explicit hints at shortcomings and problems that are covered in the following chapters were given, whereas the other information serves as background knowledge.

4 Mobile Devices

4.1 Introduction

Smart-phones, digital cameras, wristwatches, and many more commodity mobile devices are available with integrated GPS receivers – the relevance of which for SAP Sailing Analytics can already be guessed. The first example – smart-phones – also is the most interesting for this chapter. After elaborating on the question of relevance for SAP Sailing Analytics a brief introduction into GPS receivers is given, along with information on geographic coordinate systems, which is necessary for understanding the data provided by the receivers and the problems involved. Especially the discussion on map projections is important for chapter 7 on page 52, in which representation of input data for different data mining techniques is covered.

4.2 Relevance and Implications for SAP Sailing Analytics

Today's setup of SAP Sailing Analytics with dedicated tracking partners is the result of historical needs. However, this artificially restricts SAP Sailing Analytics to this setup, even though benefits could be gained by widening the horizon of SAP Sailing Analytics. International sailing events most often attain sufficient sponsoring to be able to afford professional tracking of sailboats and buoys, along with the necessary manpower dealing with maintenance of master data and on-site technical setup. Yet only a fraction of all sailing races are carried out on this level, but could still benefit from the features SAP Sailing Analytics has to offer. Acknowledging this has resulted in the decision to attempt making SAP Sailing Analytics available to the general public not only as consumers, but also as producers of race and even training data. This is justified by the following potential benefits.

Number of Tracked Races: Creating the opportunity for sailing clubs to track their

own training sessions and regattas simply through the use of smart-phones can greatly increase the amounts of data input to the system. The specific benefits for SAP Sailing Analytics are covered later on, the benefits for sailors, coaches and spectators however are obvious.

Venue-related patterns: Some parts of the world's oceans and lakes are naturally well suited for sailing regattas. At a professional level, a lot of money and time is invested into exploration and a preferably holistic understanding of recurring wind and water current patterns for these venues. An alternative approach to appointing experts to this research is to exploit and re-purpose large amounts of data gathered at these venues during training sessions and races to recognize statistically significant patterns. The boat tracks and wind data that are already recorded can help to achieve this goal, but can be augmented by further sensors. *Accelerometers*$^{\text{GL}}$ can be used to infer the heel angle β, which can be used as a measure for the wind speed at known point of sails, boat class, and sails. Digital compasses enable a direct measurement of the heading of a sailboat without having to derive this from its movement, which becomes increasingly inaccurate at a relatively lower VMG in the direction of heading. This information may then be used to infer water currents, if the measured difference between bearing and heading is corrected by the known drift due to the wind at the current point of sail and wind speed.

Boat-related patterns: The same principles applying to gathering venue data also apply to gathering information on boat classes. As shown in chapter 2.2.3 on page 7, polar diagrams represent some of the characteristics of a boat class, but are difficult to obtain. A third method, apart from physical simulation or dedicated measurements, may be to generate polar diagrams from the measured boat speeds. The data gathered from a large number of boats can be used by sailors as a benchmark for estimating their own performance. This concept can also be extended to the measurement of manoeuvre performance, so that a relative measure can be introduced to quantify the speed of execution of manoeuvres.

Regatta Management: Apart from aiding individual sailors and acting as sensors, mobile devices such as smart-phones can also assist in the management of sailing events. In 2012, a mobile application was developed to support race committees, allowing users of the *Smart-phone Application* (*app*)to send wind data entered on a mobile device to the SAP Sailing Analytics back-end, where it augments the wind-based calculations. Further use cases with more in-depth integration, supporting race officials in their communication, for example in the process of laying and moving marks, changing the course layout or detecting and announcing qualifications, can easily be imagined.[46]

Visualisation: Where live footage is difficult or too expensive to obtain, generated

[46] cf. *Uhl*, SAP Sailing Analytics Architecture, 2012.

three-dimensional graphics can complement two-dimensional visualisations such as the map offered in SAP Sailing Analytics for television and on-site commentary. Such visualisations improve their closeness to reality through the level of detail on the heading, heel, boat speed, position, and swell available, which can be drastically improved through the deployment of smart-phones and their many integrated sensors.

Smart-phones are the ideal source of data for these scenarios, as they are widely available and in use, and can act as live data sources similar to the GPS trackers employed by the tracking providers, due to their combination of wireless technology and manifold sensors. The integration of other types of mobile devices – for example through post-race file uploads – can also be considered. While the benefits have already been mentioned, some problems – which are revisited in chapter 8 on page 76 – also have to be discussed.

New Application Contexts: The functionality mentioned above implies a management of master data through SAP Sailing Analytics that goes far beyond that of today, as described in 3.3 on page 23. Entrance and maintenance of sailor, boat and other data has to be shifted to SAP Sailing Analytics to gain the independence of the current partners, as they are not necessarily involved as intermediaries if smart-phones are used as alternative tracking devices. In time, this will require a more advanced user management and new application contexts apart from race viewer and administrative UI.

Revision of Business Logic: Underlying assumptions regarding the immutability of domain objects do in some cases not hold true in this envisioned new setting, as soon as these objects begin to exist outside of the life-cycle of a tracking provider-operated sailing event. Where mistakes and errors in immutable objects were so far mainly a nuisance, this topic has to be dealt with in the future, as changes to data are inevitable, once these objects come to represent the corresponding real-world entities over longer periods of time.

Persistence and Replication: Adding persistence and enhancing replication functionality also comes into play in a much larger scale. Where currently only configuration settings are saved, in future persistence for both incoming tracking and other sensor data has to be ensured, as this is not dealt with by tracking providers any more in the scenario of directly connected smart-phones. Further, the master data already mentioned above with regards to maintenance, also has to be persisted durably.

Missing Data Sources: Removing the tracking providers and their set of offerings from the constellation has already brought forward some problems. Smart-phones can well replace the GPS trackers which are employed on the sailboats, best enclosed in a waterproof casing. Whether or not buoys, which are exposed to

more unfavourable conditions, will be equipped with smart-phones, however, is more uncertain, and increasingly unlikely at lower levels, where monetary aspects carry more weight. This additional regression from the status quo requires new concepts, which are discussed in chapter 5 on page 39.

4.3 Global Positioning System

4.3.1 Functional Principles and Terms

Comprised of several tens of satellites orbiting the earth, GPS enables users to determine their position with a precision of a few meters through the use of so-called GPS receivers.[47] The functional principals of GPS in addition also shed information on the current time and speed of the user.[48] As the size of GPS receivers has become small enough to fit into smart-phones and wrist-watches, they are becoming increasingly available and can nowadays be viewed as a commodity in smart-phones.

GPS receivers calculate the user's position at a fixed sampling rate. These individual position readings are called *fixes* and are only an approximation of the user's real position. Such a fix may also include the information on time and speed. Due to the limited precision of GPS, these fixes scatter around the user's real position when he is motionless. Larger errors, outside the usual precision range, can also occur due to different reasons – some inherent in GPS technology, some due to the receivers.

A chronologically ordered sequence of fixes is called a track, and represents the path the user travelled along. Visually, a track can be either represented by drawing the fixes as points, which can create the impression of a continuous line if the ratio of speed, sampling rate and scale is correct. Another option is to draw the lines connecting consecutive fixes.

Figure 22 on the next page shows a track of a sailboat during a regatta using the point-visualisation. Also, the positions of the start line and buoys of the race course are indicated. The figure serves to clarify a problem one may encounter when dealing with GPS tracks. Due to measurement errors, the calculated position may sometimes be wrong. The outlier shown can be identified visually through its distance to other fixes, which are densely aligned along the actual track the sailboat took. On closer examination, a gap of fixes in the track can be identified close by, during which the

[47] cf. *Wendel*, Integrierte Navigationssysteme, 2007, p. 219.
[48] cf. *Wendel*, Integrierte Navigationssysteme, 2007, p. 83.

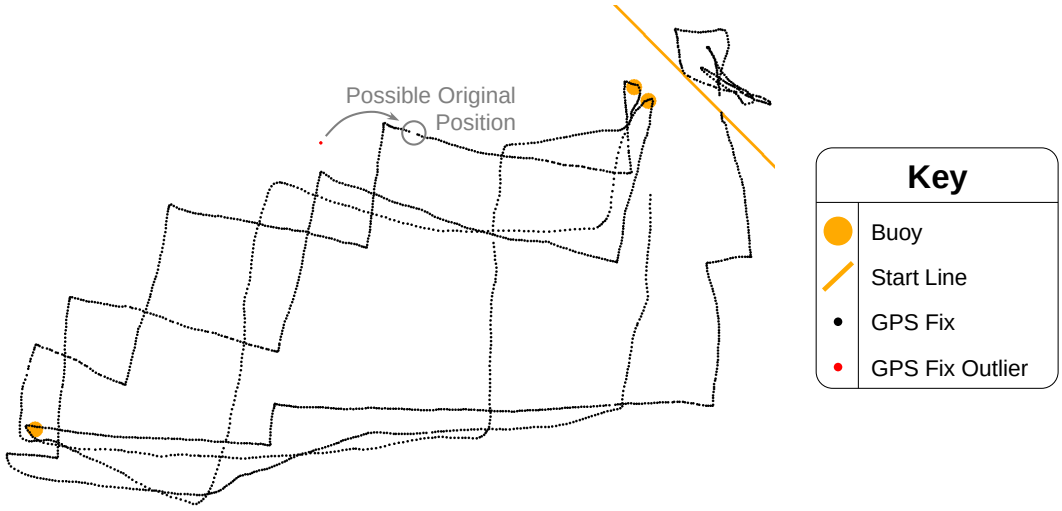


Figure 22: GPS Fixes of single Sailboat during Regatta[49]

receiver might have experienced reception issues, thus also producing an outlying fix.

This identification of outliers can easily be automated by calculating the speed required to reach its position from the chronologically previous fix's position. If this exceeds a certain threshold, for example the maximum speed a sailboat can make at the current wind speed and point of sail as described in a polar diagram, then this fix must be an outlier and can be ignored. More sophisticated methods are of course imaginable, but when analysing existing sailboat tracks this method proves effective. All tracks presented from now on are adjusted in this way, so as not to include the originally measured outliers.

4.3.2 Geographic Coordinate Systems

In reference to a three-dimensional body such as the earth, the position of a point P on the earth's surface[50] can be expressed through its geographic angular coordinates *latitude* ϕ and *longitude* λ, as depicted in figure 23 on the following page. ϕ measures the angle between the equatorial plane and the line $\overline{CP}$, with C being

[49] cf. *Ronnewinkel*, Analysis of GPS Tracks, 2012, Kiel Week 2011 – 505 Race 2 – Francois de Lisle.

[50] If a point is not directly on the surface of the reference ellipsoid, *altitude* also comes into play. For sailing this is mostly irrelevant, except for the possibility of inferring the current swell from the differences in measured altitude mentioned earlier.

the centre of the earth.[51] λ describes the angle within the equatorial plane between the line through the centre of the earth to the *prime meridian and* the line $\overline{CP}$.[52] The latitude and longitude only are meaningful when defined in terms of a spatial reference system. GPS uses the *World Geodetic System of 1984* (*WGS84*), so that all coordinates used in the further progression of this thesis can be assumed to be measured in relation to this reference system.[53]

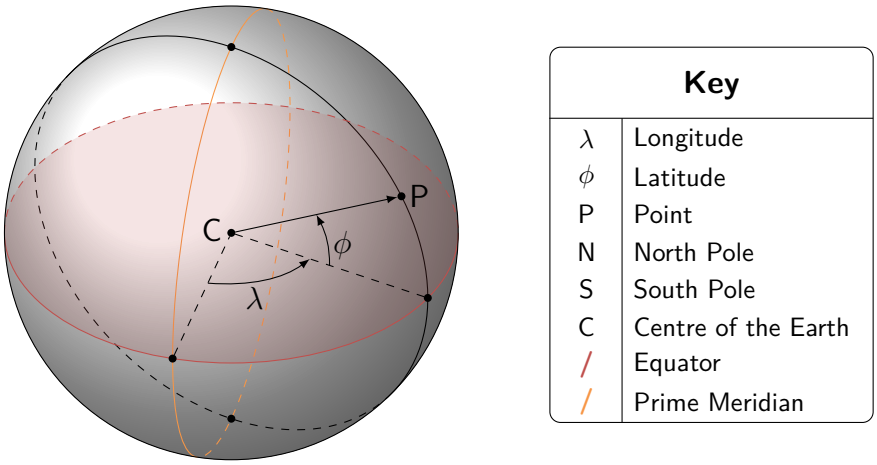


Figure 23: Geographic Coordinate System[54,55]

4.3.3 Map Projection

GPS receivers determine the user's position in such geographic coordinates. However, this form of measurement is difficult to deal with when compared to linear measures. Distances and angles have to be calculated using knowledge of the underlying spatial reference system. Instead of defining positions of a three-dimensional model, a projected coordinate system describes these positions on a two-dimensional plane, as known from maps, enabling the usage of the familiar Cartesian coordinates x and y, which are called *easting* and *northing,* respectively, in projected coordinate systems, to imply their correlation to the geographic coordinate system.[56]

[51] This is the definition of *geocentric* latitude, as opposed to *geodetic* latitude, which uses a line perpendicular to the surface of the reference ellipsoid (an abstraction for the actual form of the earth) at that point.
[52] cf. *Aitchison*, Beginning spatial with SQL server, 2009, p. 11.
[53] cf. *Aitchison*, Beginning spatial with SQL server, 2009, pp. 14 ff.
[54] cf. *Aitchison*, Beginning spatial with SQL server, 2009, p. 12.
[55] cf. *Trzeciak*, Stereographic and cylindrical map projections, 2008
[56] cf. *Aitchison*, Beginning spatial with SQL server, 2009, p. 19.

Different types of projections necessarily introduce different types of distortion: area, shape, distance, direction, and angle.[57] As a conformal projection, the *Mercator* projection preserves the shape of features, for example the route of a ship following a constant bearing is depicted as a straight line when projected. Due to this, it is the most commonly used projection in nautical navigation and also known – most likely not consciously – through its use in the Google Maps web application.[58] The *Universal Transverse Mercator (UTM)* projection alleviates the distortion the Mercator projection creates by using several local Mercator projections, in which the distortion becomes negligible. In these local projections, easting and northing are expressed in meters, and can be calculated back and forth from WGS84-based geographic coordinates.[59]

4.4 Conclusion

This chapter gave the opportunity to reflect on possibilities for improvement through the integration of mobile devices, particularly smart-phones. This integration requires additional investment into new concepts, which leads to the definition of the functional requirements in the following chapter, while the knowledge of GPS and map projections is necessary for chapter 7 on page 52.

[57] cf. *Sample/Ioup*, Tile-based Geospatial Information Systems, 2010, p. 169.
[58] cf. *Aitchison*, Beginning spatial with SQL server, 2009, p. 20.
[59] cf. *Sample/Ioup*, Tile-based Geospatial Information Systems, 2010, pp. 170 ff.

5 Functional Requirements

5.1 Introduction

After having laid the foundation in both domain-knowledge and theoretical aspects, this chapter represents the necessary steps for integrating mobile devices into SAP Sailing Analytics as functional requirements on an abstract level. The requirements are grouped content-wise into two categories: the *course layout heuristic*, and *integration of mobile devices*. These two are related, as the latter creates the need for the first, yet they have an entirely different focus. While the development of a suitable heuristic employs data mining techniques, requires more detailed knowledge of regatta sailing, and is concerned with problems of an algorithmic nature, the integration of mobile devices is a conceptual topic within the field of system analysis and design.

5.2 Course Layout Heuristic

Hitherto, tracking providers have provided not only data for the position of sailboats during a regatta, but also done the same for marks defining the course layout of the regatta. Along with the start time of the race and the mark passing times inferred by correlating the boat tracks with the mark positions, this is the data foundation for deriving further metrics and figures in SAP Sailing Analytics. By removing the tracking providers from the equation quite a few unknown quantities are introduced instead. Without the knowledge of the course layout, the tracks of the sailboats lose their value, as all figures depend on the positions of the marks. Without these positions, the concept of course legs, starting time, and all derived figures falls to pieces.

A possible, and valid option, is to provide a manual input functionality for setting the mark positions during or after the conclusion of the race. This was done for figure 24 on the following page, where the principles of sailing and the knowledge

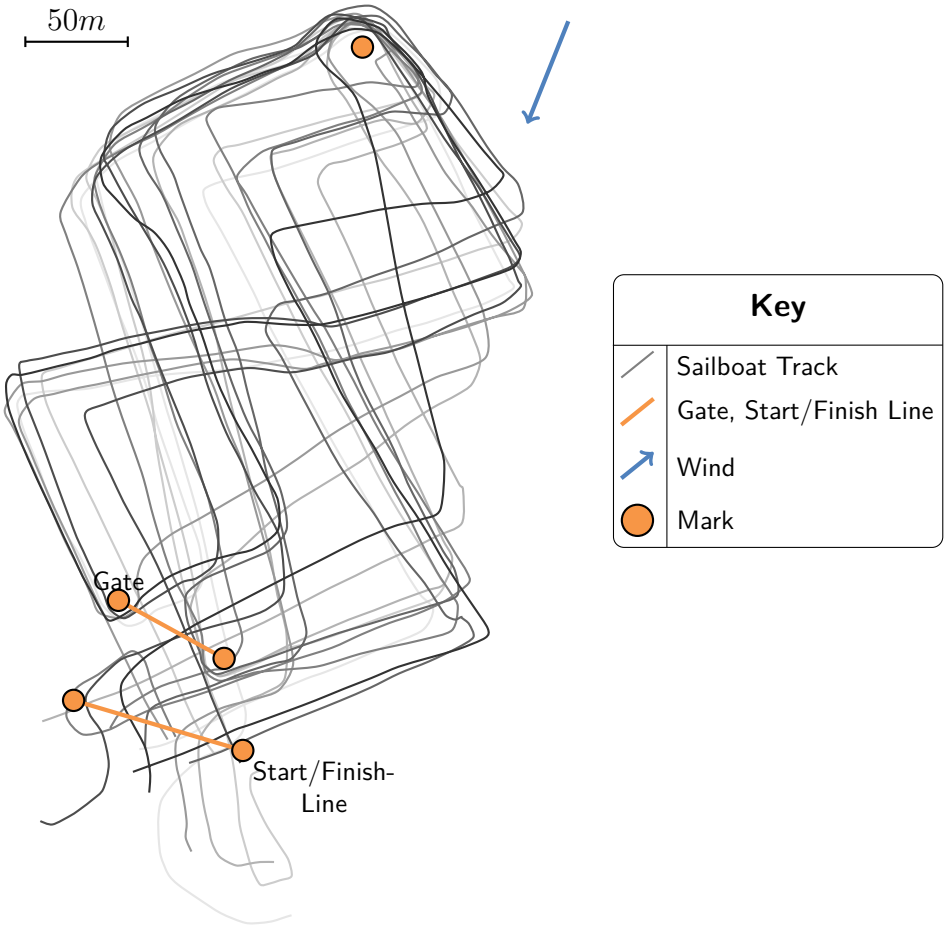


Figure 24: Mark Positions in a Tracked Race[60]

of regatta sailing were utilized to identify the mark positions and group them into a gate and a start/finish line where appropriate. The last step, which is not shown, is finding the sequence in which these marks were visited – this is a lot easier when viewing a playback of the tracks instead of a static representation as that in figure 24 – and thus gaining all information needed for SAP Sailing Analytics to calculate all further figures.

However, avoiding this step of human interaction aides the goal of making SAP Sail-

[60] The displayed sailboat tracks were tracked during the 49er European Championship 2012, Final Round 1 Race 1. They are used repeatedly in this thesis for demonstration purposes.

ing Analytics as user-friendly as possible, and thus reaching a broader user base. As mentioned before, several steps have to be taken to identify the course layout. The title *course layout heuristic* should therefore be understood as being representative of all these steps.

1. The positions of the individual marks have to be identified.
2. If multiple mark positions are identified for one single, physical mark, which drifted due to a current, these should be identified as multiple positions of the same mark.
3. Such marks that compose a control point together – a gate, start or finish line – have to be grouped.
4. The sequence in which control points were visited – the waypoints – have to be identified.

The steps listed above are ordered by increasing complexity and decreasing priority. To be able to complete a step, the information produced by all above steps is necessary. These requirements are the basis for chapter 7 on page 52, in which a heuristic shall be found for the automation of the steps.

5.3 Integration of Mobile Devices

5.3.1 Mobile Devices in General

Before delving into the smart-phone specific connectivity requirements, the general requirements for the integration of mobile devices are discussed first. Deviating from the current setup, in which tracking providers deal with most of the work concerning master data, device management and persistence, leads to the functional requirements specified below.

1. The current concept of SAP Sailing Analytics has a very rigid domain model, interspersed with immutability requirements. This makes it difficult for settings such as training races for which the focus lies solely on tracking and later viewing the race as easily as possible. The existing domain model shall therefore be adapted, so that races or training sessions with less data can be represented. Also, a means of entering this data is needed – currently done by the tracking providers.
2. Devices other than smart-phones that are not capable of sending their data

directly to SAP Sailing Analytics, are best integrated in a generic way which enables integrating a majority of such devices, the most widespread being GPS watches. File upload and import capability for the tracks recorded in standard file formats – called *trackfiles* from now on – shall provide such a generic integration.

3. Instead of having the tracking providers deal with device management, this becomes a requirement for SAP Sailing Analytics when consuming new data sources. For both file-upload and smart-phone connectivity, these data sources have to be mapped to the appropriate competitor and race. This mechanism of mapping devices to races or training sessions should be as simple as possible.

These are the requirements for an "optimum" solution. Chapter 8 on page 76 deals with their realization, however it should be noted that the focus lies on achieving a working end-to-end cross-cut rather than working out a productive solution, requiring both the analysis, design and implementation of these requirements. Dealing with the topics of replication and persistence in-depth is out of scope for this thesis. That being said, the obligation for persisting master data as well as sensor data falls to SAP Sailing Analytics as soon as there is no intermediary tracking provider, from which it can simply be reloaded.

5.3.2 Smart-phones

The trait considered unique in smart-phones for the purpose of this thesis is their connectivity. To use this to an advantage, when compared with other mobile devices, a special solution has to be developed to integrate smart-phones so that their sensor data can be fed into SAP Sailing Analytics right away.

The reason for the development of a course layout heuristic is the integration of mobile devices as additional data sources for SAP Sailing Analytics. Figure 25 on the following page shows the same diagram as in chapter 3.3 on page 23, with the addition of smart-phones – and a variety of integrated and external sensors. Smart-phones are the most important representatives of mobile devices, which is why the requirements for their integration is dealt with, before leading over to mobile devices in general.

The following requirements for the server-side are introduced by explaining their necessity on the smart-phone tier, as they are essentially two sides of the same coin.

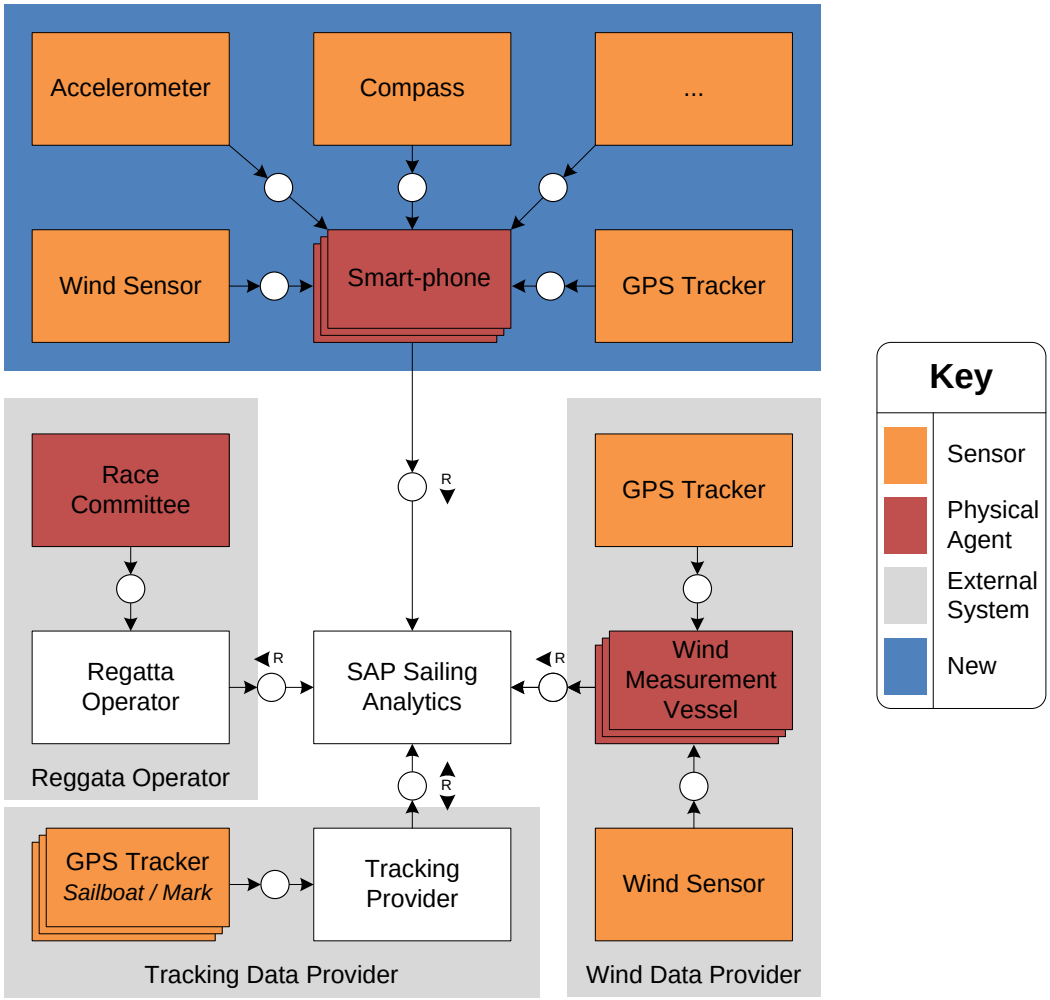


Figure 25: Smart-phones as Additional Data Providers

1. Mobile data networks such as GSM rely on local, land-based transceiver stations. In the absence of such in the vicinity of a regatta venue, and through other disruptions in reception, smart-phones may not always be able to transmit and receive data. This is a fact that has already proven true for the GSM-based GPS trackers, resulting in the caching and invalidation topic discussed in chapter 3.4.4 on page 29. Therefore, both the smart-phone and server components have to be robust to disruptions in connection, and deal with a possible state of semi- or dis-connectedness appropriately. This means that where possible, transmission of live data – with the primary goal of minimum lag in transmission – is desirable. More important, however, is to eventually

obtain continuous and valid sensor data on the server. This differentiation is borne out through the consideration of battery shortages, which may force the smart-phone to abandon the transmission of data, and instead revert to a battery-saving state in which sensor data is captured less frequently, and stored locally until after the race.

2. The syntax of the data presentation layer of the *National Marine Electronics Association* (*NMEA*) 0183 standard – which defines a multi-level protocol, originally intended for serial data buses in nautical environments – has developed into the de-facto standard for all marine-related sensor equipment.[61] As different kinds of sensors – both internal and external – may provide the data transmitted through smart-phones, this standard shall be used for encoding the sensor readings. On the one hand, this minimizes the obstacles for gathering and transmitting data of external sensors, such as wind-sensors, through smart-phones, as these most likely encode their sensor data according to NMEA 0183. On the other hand, the communication channel devised for smart-phones in particular, is more likely to be re-used by other clients apart from smart-phones with a SAP Sailing Analytics app, if it adheres to a standard.
3. Smart-phone usage for tracking gains more value if it can be performed entirely without having to resort to a regular computer for steps in-between. Therefore, apart from the possibility to transmit sensor data, a simple form of master data management shall become accessible from within the app, so that sailors can create a race, register for it, start the tracking and finally end the tracking on server-side through use of their smart-phones only.

5.4 Conclusion

Smart-phones and mobile devices in general, can help to reach out to and provide for a new user group through SAP Sailing Analytics. The steps that have to be taken to enable this integration were shown in the form of requirements in this chapter. These serve as the starting point for the chapters following the next, in which their realisation is shown.

[61] cf. *National Marine Electronics Association*, NMEA-0183, 2008.

6 Data Mining

6.1 Introduction

Data mining is an interdisciplinary field, evolved to meet the requirements of processing and interrogating rapidly growing masses of data to uncover patterns.[62] This field of data mining is first structured, so that the relevant areas for the course layout heuristic can be identified. These are then discussed by giving a brief introduction to the some fundamental data mining concepts and techniques, which are applied in chapter 7 on page 52.

6.2 Data Mining Goals and Methods

The field of data mining itself evolved to meet the needs of automatically processing ever-growing amounts of raw data. This growth is attributed on the one hand to an increasing number of data sets, on the other hand to the increasing size of each individual data set, as the number of observable attributes increases. Usually the fact that humans can no longer process the amounts of data, or are not able to discern patterns in the increasingly large data sets, is the reason for employing data mining techniques.[63] This is not the case for the applications of these techniques discussed in this thesis. Fleet sizes for sailing regattas may vary from two boats in match races to hundreds of boats in the 505 boat class, but it is still possible for humans to discern mark positions, starting time of the race and other facts when observing the progression of the race on a map. Instead, data mining techniques shall serve to simplify the process of tracking races with smart-phones, by automatically inferring the missing data. Through simplification it is hoped to reach a larger number of users, which in turn is needed for gaining some of the benefits of this integration discussed in chapter 4.2 on page 32.

[62] cf. *Abbass/Sarker/Newton*, Data mining - A heuristic approach, 2002, chp. 3.

[63] cf. *Usama Fayyad/Gregory Piatetsky-shapiro/Padhraic Smyth*, Knowledge Discovery in Databases, 1996, pp. 37 ff.

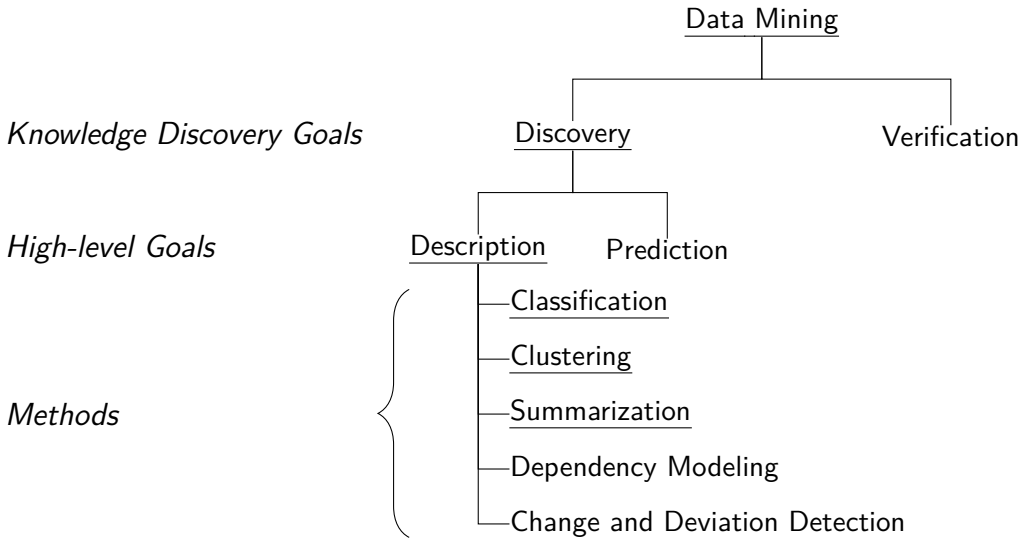


Figure 26: Data Mining Goals and Methods[64]

Figure 26 puts the goals and methods of data mining into a tree-like structure, in which the nodes relevant for this thesis are underlined. On the knowledge discovery goal level, the heuristics in chapter 7 on page 52 aim at *discovering* patterns as opposed to *verifying* hypotheses of the user. Discovery can be further split into two high-level forms, *description* and *prediction*. Only the former of these two, which deals with finding patterns to describe existing data, is relevant. On the most detailed level depicted in figure 26, the methods describe different ways of achieving the goals selected above. In the remainder of this chapter, different data mining algorithms are introduced in preparation for chapter 7 on page 52. The following criteria are used to assess, classify and compare these.

Method: Determining which of the previously mentioned data mining methods an algorithm may serve is the first criterion.

Determinism: Patterns are inferred by fitting models to the observed data. Once the model is finally fitted, it represents the inferred data. These models can be fitted through either a *statistical* or *logical* approach. While the latter has the benefit of being purely deterministic, therefore always yielding the same results on a fixed set of observed data, the statistical approach can often incorporate the uncertainty involved in gathering the original data better.[65]

Components: In general, data mining algorithms can be said to be comprised of three

[64] The figure is a visual representation of the structure described textually in *Usama Fayyad/ Gregory Piatetsky-shapiro/Padhraic Smyth*, Knowledge Discovery in Databases, 1996, p. 43.

[65] cf. *Usama Fayyad/Gregory Piatetsky-shapiro/Padhraic Smyth*, Knowledge Discovery in Databases, 1996, p. 43.

components: *model representation, model evaluation,* and *search.* The specific form these components take is distinctive for a data mining algorithm.[66]

6.3 Artificial Neural Networks

On a technical level, the human brain is a natural processing unit very different from what is conventionally considered a computer. It excels in abstract forms of processing and also creativity, where it outclasses the most elaborate super computers. *Artificial Neural Networks* (*ANNs*) try to emulate the inner workings of the human brain, which is comprised of approximately 100 billion neurons – the basic processing units – and a much higher number of synapses – the connections between neurons, at a smaller scale. This massive parallelism has been proven to be particularly adept in the task of classification. The term neuron from now on is used to refer to the artificial counterparts of the natural processing units.

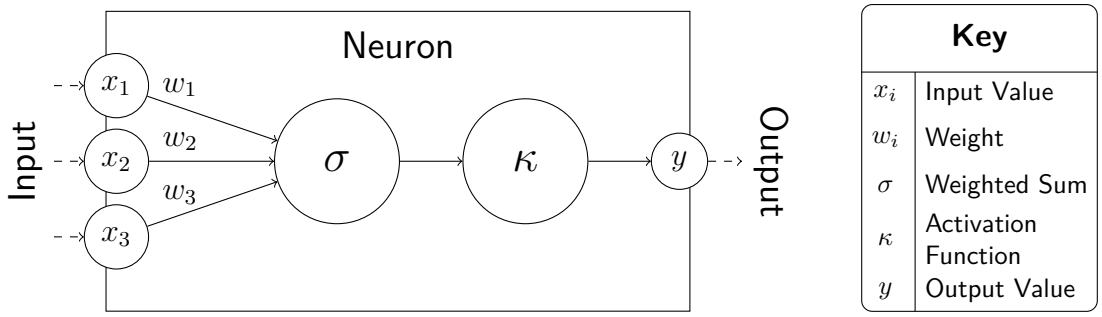


Figure 27: Activation of a Neuron[67]

Neurons are very basic, elementary units, which is why an ANN gains its potential through the way these neurons are connected and interact with each other. The neuron depicted in figure 27 has three input synapses and one output synapse. These might be connected to other neurons, or represent the input and output of a very simple ANN, called *Perceptron.* The functional principles of a neuron can be explained using this example.

[66] cf. *Usama Fayyad/Gregory Piatetsky-shapiro/Padhraic Smyth*, Knowledge Discovery in Databases, 1996, pp. 45 f.

[67] cf. *Kramer*, Computational intelligence, 2008, p. 126.

$$
\begin{aligned}
x_i &\in \{0,1\} \\
w_i &\in [0,1] \\
y &\in \{0,1\} \\
\sigma(\vec{x},\vec{w}) &= \vec{x}\cdot\vec{w} \\
y = \kappa(\vec{x},\vec{w}) &= \begin{cases} 0 & \text{if } \sigma(\vec{x},\vec{w}) < \theta \\ 1 & \text{if } \sigma(\vec{x},\vec{w}) \geq \theta \end{cases}
\end{aligned}
$$

The formulas given above represent a neuron with a simple threshold activation function κ and binary input values $\vec{x}$ and output value o. The outbound signal o is equal to one, if the weighted sum σ of the elements of $\vec{x}$ is greater or equal to a certain threshold θ. While σ is always the scalar product of $\vec{x}$ and $\vec{w}$, the activation function κ may take the form of any sigmoid function. In consequence, instead of the binary input and output values used in this example, any subset of the real numbers $\mathbb{R}$ may be applied, although for practical purposes this is usually limited to the interval of real numbers $[0,1]$ or $[-1,1]$.[68]

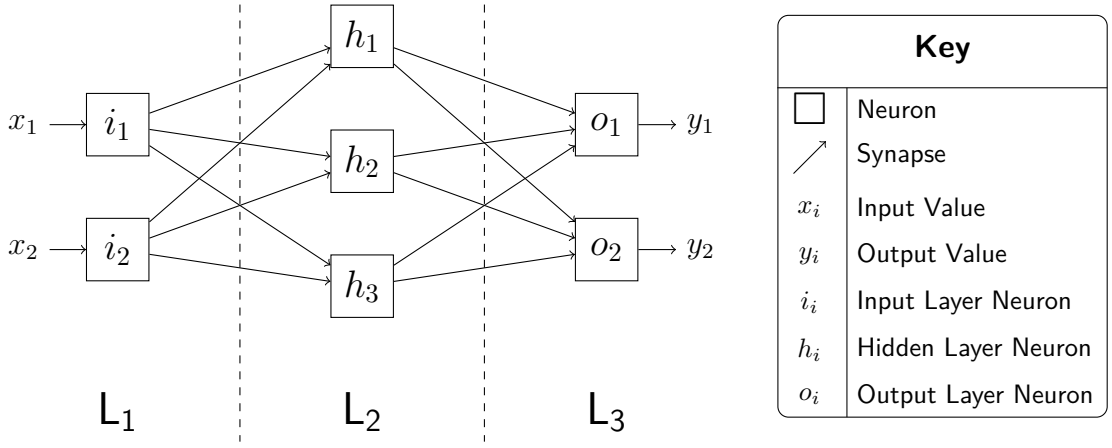


Figure 28: Feed-forward Artificial Neural Network[69]

Larger ANNs, composed of several neurons and synapses, can be trained to perform certain tasks. In a classical *feed-forward ANN*, the neurons are divided into layers, as shown in figure 28. The input values $\vec{x}$ are applied to the input layer L_1, which feeds its outbound synapses into the first of one or more hidden inner layers $L_2, \ldots, L_{n-1}$, the last of which then feeds its outbound synapses into the output layer L_n. Iterating over each layer from first to last, the activation κ of each neuron in this layer is

[68] cf. *Kramer*, Computational intelligence, 2008, pp. 125 ff.
[69] cf. *Kramer*, Computational intelligence, 2008, p. 129.

calculated. The output values of layer i then serve as input values for layer $i + 1$. The output values of layer L_n finally represent the calculated output of the ANN for the provided input.

Apart from modelling the layers and connections, the processing power of an ANN lies in the specific weight that is awarded to the value coming from each synapse. These weights are set through a training process, after initializing them randomly, by repeatedly cycling through a set of training sets consisting of input and output values. The input values for each set are applied, and the weights are then adjusted so that the output values calculated by the ANN slowly grow closer to those specified by the training sets. The training process does not require an understanding of what each individual connection represents, as the ANN is expected to align itself to the patterns present in the training sets through the training process. However, the layout of neurons and connections, the representation of input and output values, the choice of training sets, and the chosen training method have a great influence on whether the ANN is able to learn to correctly classify the patterns from the training sets. It is important to ensure that the ANN is not over-fitted to the training data, but instead classifies by recognizing the underlying patterns instead of their specific manifestations present in the training sets.[70]

During the training process, the comparison of expected and calculated output – which is the search component of ANNs – serves to evaluate the model currently represented by the ANN. This representation includes the layout of the ANN as well as the current synapses weights, so that once the training process is completed, the ANN itself represents the identified patterns. Besides the random initialization of weights, the training patterns should also be presented in different, random orders in the training cycles, to eliminate any direct dependencies the ANN might create to the training data. Due to both these facts, ANNs can be considered statistical, non-deterministic data mining techniques.

6.4 Clustering Methods

Clustering methods are named after the data mining method they serve. The concept of clustering a data set, so that similar items are grouped from dissimilar items is rather abstract when compared the detailed semantics of ANNs. The term *clustering methods* is the most vague of all the presented data mining techniques, as entirely different implementations of clustering methods exist, making it a challenge

[70] cf. *Kramer*, Computational intelligence, 2008, pp. 128 ff.

to find and choose the correct algorithm.[71] Therefore, not much can be said in general about the components and determinism of clustering methods. The only exception to this is the model representation, which always manifests itself in the created clusters.

Though the implementations may differ, clustering methods can be separated into two basic types:

Partitioning Algorithms: This type of algorithm, of which the K-means algorithm is a well-known representative, partitions an input data set into a predefined number of clusters, typically by trying to optimize an objective function such as the hill climbing heuristic. The main disadvantage of this clustering method type is that domain knowledge is necessary for specifying the number of clusters.

Hierarchical Algorithms: As the name implies, hierarchical clustering algorithms create a hierarchy of clusters within the input data set, which can be represented by a tree that splits the complete set of input data into ever smaller clusters. Depending on the algorithm this tree may be created from the bottom up by grouping individual items into larger clusters – an agglomerative approach – or by starting at the root of the tree and iteratively subdividing the complete set – a divisive approach.[72]

$$d_{Euclidean}(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \mid p,q \in \mathbb{R}^n$$

The similarity of items used for clustering is usually based on a distance measure between a cluster centre and a data item (from now on called point), or between two points. The input data needs to be defined in a metric space, as such a space defines the distance measure – also called metric – between two points. Which metric is used depends on the type of input data, which more often than is defined within the familiar Euclidean space, resulting in the usage of the Euclidean distance, shown in the equation above, as metric.[73]

[71] cf. *Panda* et al., Comparing Fuzzy-C Means and K-Means, 2012, p. 1.
[72] cf. *Ester* et al., DBSCAN, 1996, pp. 1 f.
[73] cf. *Panda* et al., Comparing Fuzzy-C Means and K-Means, 2012, p. 451.

6.5 Genetic Algorithms

> "Genetic algorithms are a family of computational models inspired by evolution. These algorithms encode a potential solution to a specific problem on a simple chromosome-like data structure, and apply recombination operators to these structures in such a way as to preserve critical information."[74]

The basic idea of genetic algorithms is summed up in the quote above. Usually, genetic algorithms are used as optimizing functions. To adapt the genetic algorithm concept to a specific solution domain (the problem), two components are needed, which happen to be the model representation and evaluation components:

1. The genetic representation specifies the encoding of the problem and describes the structure of the solution candidates, also called phenotypes. These consist of multiple chromosomes.
2. The evaluation function, also called fitness function, is used to calculate fitness values for the phenotypes.[75]

The search component of a genetic algorithm is determined by the chosen selection strategy and genetic operations such as mutation and crossover, which determine the transition between consecutive generations of phenotypes. As most selection strategies include randomized elements, and the initial population is usually randomly generated, genetic algorithms definitely have to be considered statistical data mining techniques. The data mining method they serve cannot be so easily determined, as they can be adapted to virtually any problem through the choice of the right genetic representation and evaluation function.

6.6 Conclusion

The introduction and comparison of the three data mining techniques ANNs, clustering methods and genetic algorithms covers only a small part of the field of data mining. However, this small part is the one relevant for the following chapter, in which the fundamentals described in this chapter are revisited and applied to the domain of regatta sailing in the search for a course layout heuristic.

[74] *Whitley*, A genetic algorithm tutorial, 1994, p. 65.
[75] cf. *Whitley*, A genetic algorithm tutorial, 1994, pp. 65 f.

7 Course Layout Heuristic

7.1 Introduction

In this chapter, different approaches for a heuristic that can identify a course layout based on a set of sailboat GPS tracks are presented and evaluated. Figure 24 on page 40 already visually represented the problem at hand for a simple case. The tracks of this race will also be used to visualize the results of the most promising approaches.

First, a brief overview over the difficulties of automating the process of identifying a course layout is given,. The criteria chosen for assessing the different approaches are then presented, before elaborating on some general approaches of representing the input data in a suitable fashion. The discussed representations are used in the main part of this chapter, discussing the actual approaches for finding a heuristic. These approaches focus on the first and most difficult step in identifying the course layout, which is finding the positions of the individual marks. The discussion of further steps concludes this chapter.

7.2 Course Characteristics and Assumptions

This section first summarizes the characteristics of a regatta course and sailboat movement, before then listing the assumptions on the behaviour of sailors and natural influences. These originate in part from physical principles and in part from observation and common sense.

Different from other races, sailing regattas only have a partially defined track – through the waypoint sequence. The sailors are free to chose whatever path they wish, and must only pass the marks by adhering to the string metaphor explained in chapter 2.3 on page 11. For motorized vessels, water offers 360° of freedom for moving about. If sailboats were motorized, they would therefore move from

mark to mark along straight lines, changing their direction only after passing a mark. However, sailboats are limited in their movement by the wind, leaving them much fewer options. Sailboats move along an imaginary grid, so as to travel at maximum VMG in direction of the next mark. This means that manoeuvres are performed not only at marks, but also all over the regatta course. As the angle at which the sailboats move to the wind is approximately the same for all boats in one race, sailboats perform similar manoeuvres all over the course – and not only close to control points. It is assumed that all sailboats pass all of the marks with a manoeuvre.

Therefore, similar behaviour of sailboats at a common point of the course is considered a characteristic trait of a control point. This assumption is however undermined by different influences, most of which can be traced back to the fact that – as in all races – the group of sailboats do not move as one, but at individual speeds, so that the field eventually spreads out. The influence of wind and water currents makes the fact, that sailboats in reality pass the control points at different points in time, prove problematic. Sailboats move and manoeuvre differently in varying wind conditions, so that – combined with tactical considerations – the type and and form of the manoeuvre taken to round a control point varies, depending on the geographically and temporally local conditions. Also, marks drift with the current, shifting their position over time. So even for one *waypoint*, the sailboats may pass it not only at different times, and in different manners, but also in different places.

Apart from these assumptions, a few pre-conditions are needed to limit the complexity of the problem at hand. One of the main problems is, that when trying to infer all missing data at once, many of the missing facts interleave and are co-dependent. For example, finding the position of the start line control point depends mainly on finding the time at which the race started. This is also roughly true for the finish line and end time. Finding the time and point of control point passings, which would help split up the track of a sailboat into legs, thus giving information on what part of the race a sailboat is currently in, of course also depends on first finding the positions of the control points. For this reason, the problem domain is limited to finding a course layout heuristic for all waypoints between the start and finish line.

7.3 Criteria

Many criteria can be used to evaluate the heuristics and approaches presented in this chapter. However, the chosen set of criteria now listed resides completely on

the semantic level and omits criteria of a more technical nature, such as analysis of run-time complexity, which has been sufficiently researched and documented by others for the used data mining techniques.

Generalisability: In terms of generalisability, a perfect course layout heuristic should identify any kind of course layout, regardless of the layout type, boat class and other influence factors. The approaches are assessed by how close they come to this benchmark, with the explicit goal of finding heuristics that require as little as possible prior knowledge as possible.

Accuracy: This criteria represents how close a heuristic manages to come to the actual course layout with its inferred course layout. For the sake of comparison, this criteria is assessed by applying the approach to sample data. Where possible, the theoretical limits in accuracy dictated by the chosen approach are also mentioned.

Completeness: The heuristics can be compared by how completely they manage to identify the course layout. This can be measured by the levels of information that a heuristic can provide on a course layout, as already described in 5.2: mark positions, waypoints and control points. A perfect heuristic should manage to infer all of these. They have to be distinguished, as marks may drift, so that several mark positions may belong to the same mark. An identified mark position may be used in identifying the waypoints, many of which may be identical to the individual marks, with the exception of gates which are pairs of marks. As a last step, multiple waypoints that represent occurrences of the same control point, for example the same buoy, in the course sequence can be linked.

These criteria are not applied to the data mining techniques in general, but to the specific presented approach, unless stated otherwise. This is especially important when the shortcomings of an approach are discussed, as these might stem either from a general lack of suitability of the technique, or from the chosen algorithm or representation of input data.

7.4 Representation of Input Data

7.4.1 Necessity

This representation of the input data is one of the first issues that has to be dealt with, and should not be taken lightly, as it has a major influence on the success

or failure of the introduced data mining techniques.[76] Firstly, the representation has to be processable by the used algorithm or heuristic. Secondly, it has to be a representation that preserves the important details.

ANNs can only process input data that can be represented by a fixed number of scalar input values (which are fed to the neurons of the input layer). Clustering methods, on the other hand, rely on the distance, also called metric, between elements in a metric space (for example the Euclidean space).[77] Map projection and the simplification of sailboat tracks through the detection of manoeuvres are discussed as two methods of representing the input data in the following.

7.4.2 Map Projection

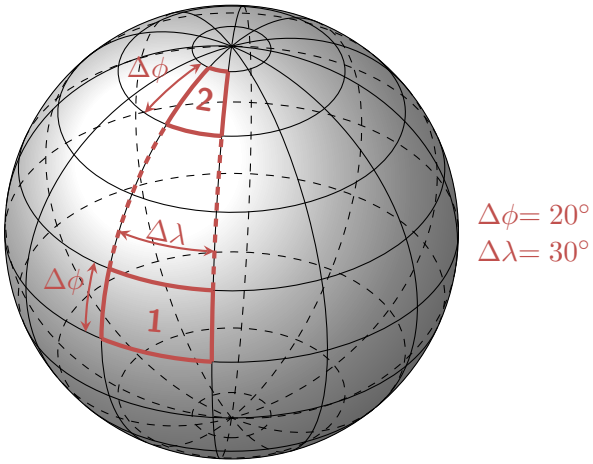


Figure 29: Distances, Angles and Spherical Coordinates[78]

A GPS track has already been defined as a chronologically ordered sequence of fixes. Each of the individual fixes is characterized through a geographic coordinate and a time point. Time points can be represented through the number of time units (for example seconds) that have passed since a reference point in time. As the latitude and longitude values of geographic coordinates are spherical and not Cartesian coordinates, they do not relate to distances and angles as one might expect. Figure 29 shows two areas marked as 1 and 2, which have equal angular distances between their latitude and longitude boundaries. However, their shapes differ greatly, as the distance between two points with the same latitude, located

[76] cf. *Gaber*, Scientific data mining and knowledge discovery, 2010, p. 9.
[77] cf. *Panda* et al., Comparing Fuzzy-C Means and K-Means, 2012, p. 451.
[78] *Trzeciak*, Stereographic and cylindrical map projections, 2008

on adjacent circles of longitude decreases the further away they move from the equator.

This is why the conformal UTM projection was introduced in chapter 4.3.1 on page 35, through which geographic coordinates can be projected onto a Cartesian coordinate system, preserving the shape of objects as they would be observed for example from an aeroplane. By projecting the sailboat tracks from their representation through GPS geographic coordinates into a Cartesian coordinate system, distances and angles can on the one hand be calculated easily (in a good approximation). On the other hand, this representation is related more directly to the original data, so that data mining techniques should have a higher success rate in identifying the patterns relevant for the course layout. All sailboat tracks shown in this thesis have been projected with the UTM projection.

7.4.3 Manoeuvres as Simplifications of Sailboat Tracks

input : L: an array of n points representing a polygonal chain
ν: max. distance from any point of L to the clostest segment of L'
output: $L' =$ `DouglasPeucker(`L,1,n`)`, L' approximates L with less points, $|L'| \leq |L|$

function `DouglasPeucker(`L,i,j`)`**:**
  $L_f \leftarrow$ point in L with $i \leq f \leq j$ that lies farthest from the line $\overline{L_i L_j}$;
  $d \leftarrow$ distance between L_f and $\overline{L_i L_j}$;
  if $d > \nu$ **then**
    $left \leftarrow$ `DouglasPeucker(`L,i,f`)`;
    $right \leftarrow$ `DouglasPeucker(`L,f,j`)`;
    return $left + right$;
  else
    return $\overline{L_i L_j}$;
  end

Algorithm 1: Douglas-Peucker Line Simplification Algorithm[79]

The Douglas-Peucker algorithm is a solution to the line simplification problem, in which a polygonal chain made up by several points is sought to be simplified by finding a different polygonal chain that has less points than the original, but represents it well.[80] A GPS track is an example for such a line that can be simplified

[79] cf. *John Hershberger/Jack Snoeyink*, Douglas-Peucker Line-Simplification, 1992, p. 2
[80] cf. *John Hershberger/Jack Snoeyink*, Douglas-Peucker Line-Simplification, 1992, p. 1.

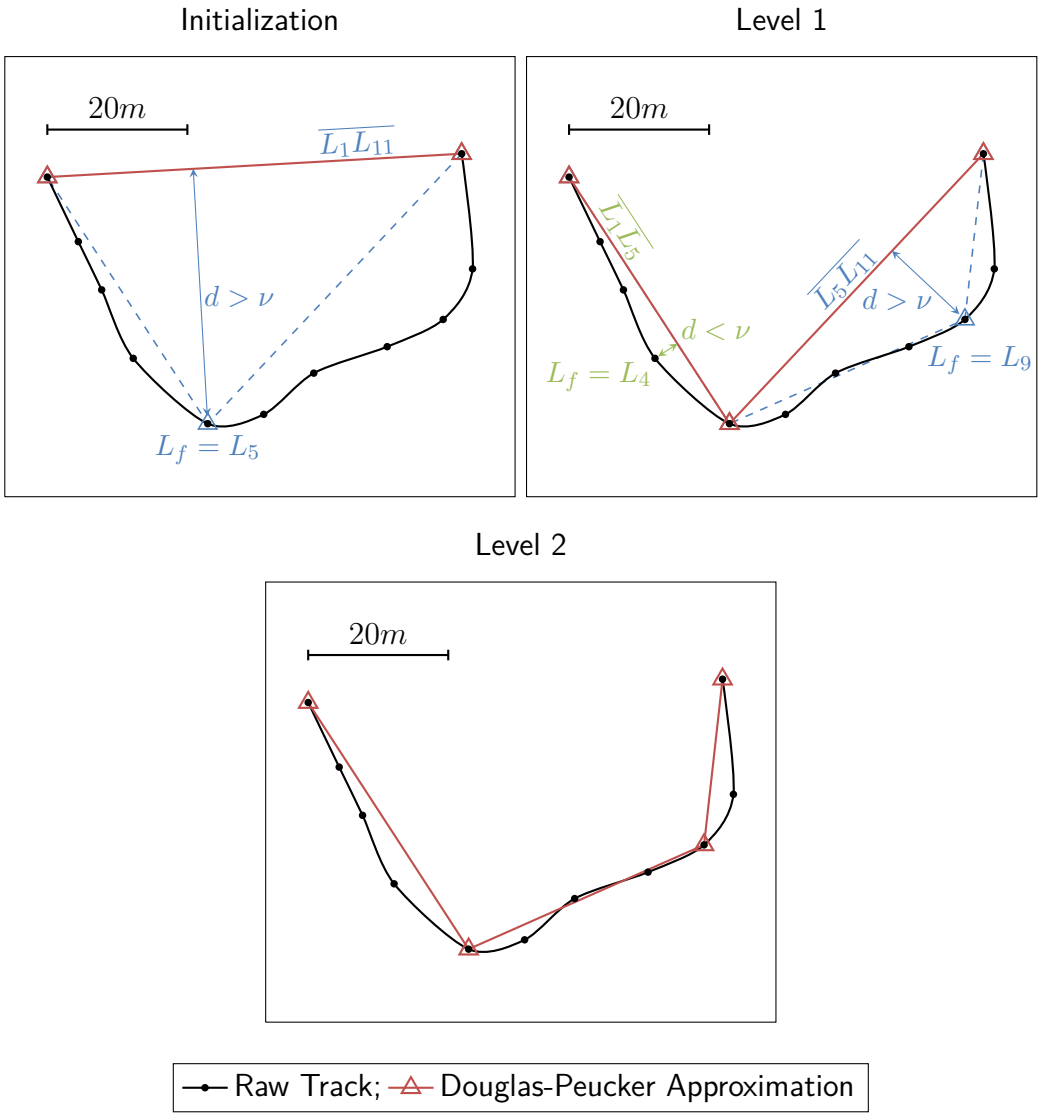


Figure 30: Douglas-Peucker Line Simplification Algorithm

with the Douglas-Peucker algorithm. Algorithm 1 on the previous page describes how this recursive algorithm works through pseudo-code, while figure 30 depicts the recursive progression. The algorithm starts with one line – the straight line between start and end-point of the polygonal chain – and breaks this up into two pieces, if the greatest distance between the straight line and any of the intermediate points of the polygonal chain is greater than some value ν. This process is repeated for both of the newly created line segments, which now represent the simplified chain after the first recursive level, until the simplified chain has been split often enough, so that no point of the original chain is further away from the simplified chain than

ν. In figure 30 on the previous page this is achieved after two recursive levels for $\nu = 10m$. In the first level the line between start and end-point was split into two parts, of which only one had to be split yet again in the second level.

This algorithm is helpful for regatta sailing because sailboats move along the imaginary grid of the playing board, with clearly defined manoeuvres to change direction. With the help of the Douglas-Peucker algorithm, the positions where the manoeuvres were performed can be approximated, if all points on the simplified polygonal chain generated by the algorithm are thought of as manoeuvre positions – but for the very first and very last, which are just the start and end point of the original chain.

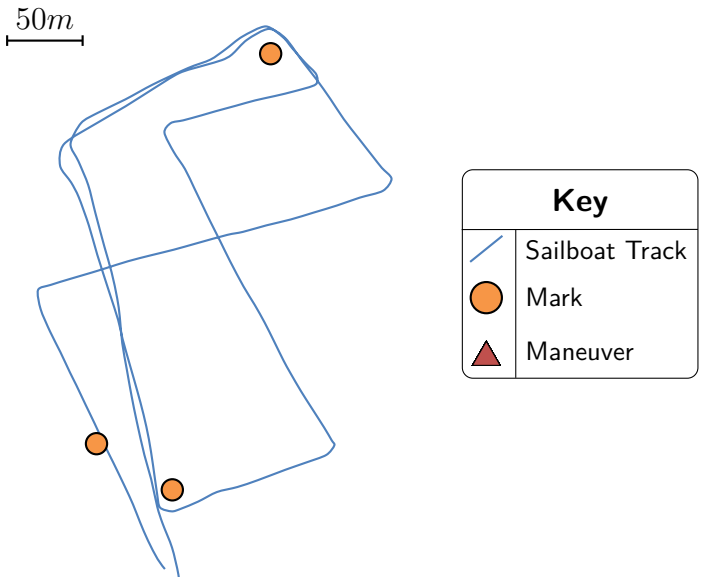


Figure 31: Representing a Sailboat Track through Manoeuvres

Figure 31 shows how a sailboat track can be represented in a simpler form through the Douglas-Peucker points which are used as manoeuvre positions. The amount of data is significantly reduced from roughly 140 GPS fixes, making up the track, to 14 manoeuvres. As the inference of wind direction and strength from sailboat tracks implemented in SAP Sailing Analytics relies on the knowledge of the course layout, the manoeuvre types (tack, gybe, etc.) – which are defined in relation the the wind direction – cannot be identified prior to inferring the course layout. The Douglas-Peucker algorithm has already been implemented in SAP Sailing Analytics for the manoeuvre detection functionality shown in 17 on page 22.

The manoeuvres of the exemplary race are shown in figure 32 on the following page. This substantiates the earlier assumption that sailboats exhibit similar behaviour

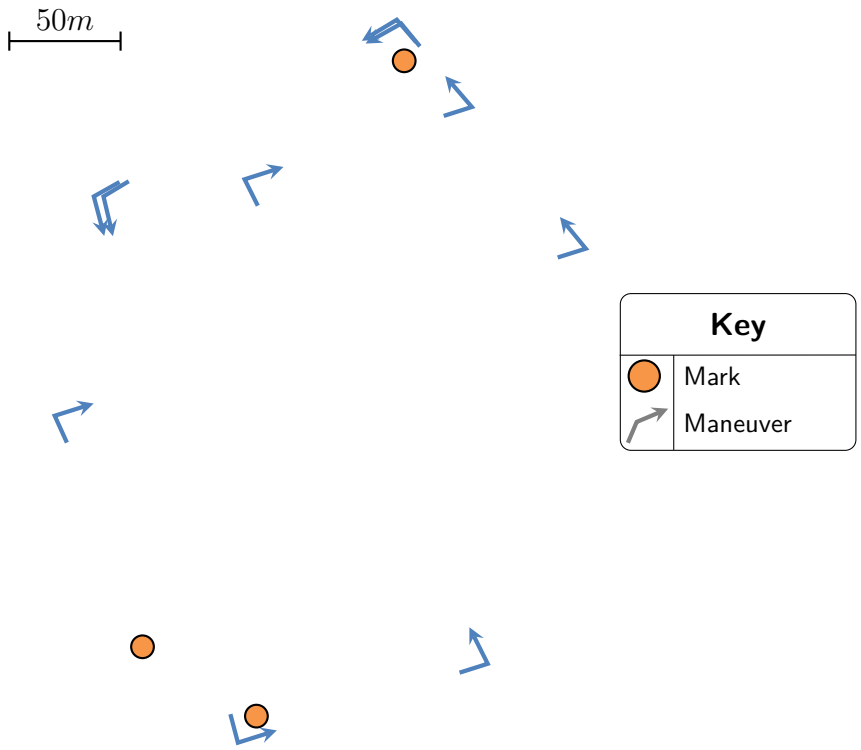


Figure 32: Manoeuvres and Control Points

near control points. The chosen visualisation can convey all information on the manoeuvres – competitor, position, heading before and after (and implicitly turn angle and turn direction) – but for the time point. While detecting manoeuvres is one way to represent the input data, this approach is not necessarily without problems. Firstly, the choice of an appropriate value for ν – in this case $13m$ – is important, as otherwise either too much noise, or not enough relevant information is contained in the reduced data set. Secondly, while the data is reduced to a form that helps a human identify mark positions, it can be argued that this is only one possible approach. Therefore, when tackling the course layout heuristic with the help of ANNs, the raw track data is used to see if the ANN can learn the characteristics of a mark position by itself, without needing pre-digested input data.

7.5 Clustering Methods

7.5.1 General Problems

The assumption that sailboats perform similar manoeuvres at certain points of the course – those points, where the control points are – indicates that it may be possible to identify those manoeuvre *clusters* through a suitable clustering method. There are a few problems with this approach, which are presented here and further discussed in the remainder of this section.

- Even if start and finish lines are ignored, gates still pose a problem. While all sailboats pass the same single mark control points, they may chose which mark of a gate to round. If one side of the gate is advantageous due to the wind direction or direction of the next leg, the sailboats may favour this side of the gate. In the extreme case, all manoeuvre around the same side of a gate, so that the control point could only be recognized as a single mark. However, any other distribution of boats between the two marks of a gate means that one has to deviate from the assumption that *all* sailboats manoeuvre around the same mark.
- Due to tactical considerations, or simply chance, all sailboats may choose to manoeuvre at roughly the same point on the course, even though no control point is there. In this case, the assumption may lead to erroneously identifying a control point where in reality there is none. This should however be considered a minor nuisance instead of a major problem, which could only be avoided by applying additional, user-provided knowledge of the course layout.
- The combination of mark drift and spread of the field destroy the close group of manoeuvres, as different competitors pass the same control point at different positions at different points in time.

7.5.2 Naive K-means Approach

The K-means clustering algorithm, which is described in algorithm 2 on the next page, partitions an input data set into a pre-defined number of clusters, based on a distance measure between points of the input data set. After randomly initializing each cluster centre, each point is assigned to the cluster centre closest to it, so that the cluster centres can be recalculated based on the assigned points. This is repeated until some criterion is satisfied, usually either a maximum number of iterations, or until the assignment of points to cluster centres has stabilized, so that the cluster

input : k: number of clusters to be found
D: data set with n points
output: A set of k clusters

for $i = 1..k$ **do**
  initialize cluster centre C_i randomly;
end
repeat
  for *each point* p *in* D **do**
    find cluster centre C_n nearest to p;
    assign p to C_n;
  end
  recalculate position of cluster centres using the mean of the assinged points;
until *stop iteration criterion satisfied*;

Algorithm 2: K-means Clustering Algorithm[81]

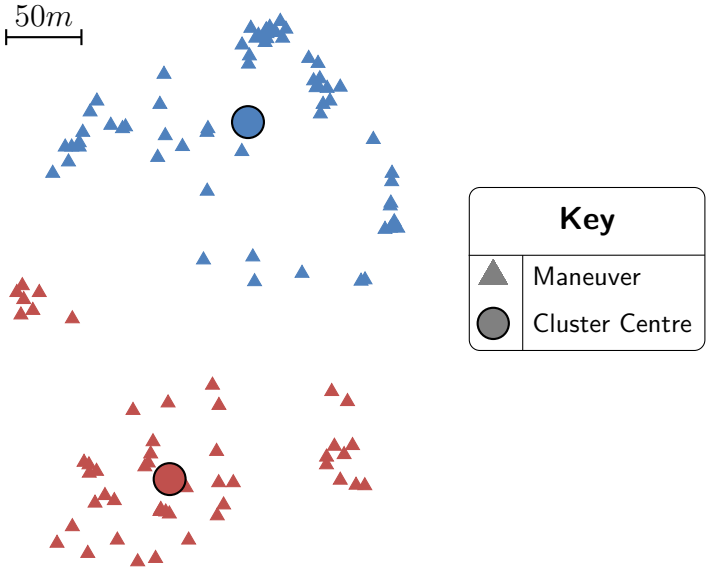


Figure 33: K-Means Manoeuvre Clustering

centres stop moving.[82]

Figure 33 presents the result of applying the K-means algorithm to the manoeuvres of a race. Only the easting and northing coordinates have been used in combination

[81] cf. *Panda* et al., Comparing Fuzzy-C Means and K-Means, 2012, p. 453.
[82] cf. *Panda* et al., Comparing Fuzzy-C Means and K-Means, 2012, p. 453.

with the Euclidean distance as a similarity measure. This approach is naive in many points, and it is only presented to ease transition into this section by identifying what is *wrong* with using this algorithm:

1. The K-means algorithm is not suited for the underlying problem. It seeks to group all input data sets into clusters, which is not the objective in identifying the course layout, where only dense groups of manoeuvres are interesting.
2. It ignores all attributes of the input data (the manoeuvres) except their coordinates.[83]
3. Apart from the input data sets, the number of clusters needs to be known beforehand. The goal however is to find a heuristic that will function without any prior knowledge of the course layout.

From these problems the requirements for a clustering method suitable for identifying the course layout can be identified:

1. Only dense groups of manoeuvres shall be detected as possible mark positions. What is considered a dense group of manoeuvres is dependent on the necessary robustness to drift and spread of field, as the combination of both has an influence on the potential density of the clusters.
2. Multiple dimensions of the input data, not only the coordinates, need to be considered.
3. The number of clusters should be detected automatically.

7.5.3 Refined Density-based Approach

The *Density Based Spatial Clustering of Applications with Noise* (*DBSCAN*) algorithm – an agglomerative hierarchical clustering algorithm – meets these requirements. The basic idea is to identify clusters by their characteristic of having a density of points which is higher than outside of the cluster. Clusters are required to consist of a minimum number of points μ that satisfy the new criterion of density-reachability. All points which are not part of a cluster are considered noise.[84] This applies to the domain of regatta sailing if mark positions can be identified as dense manoeuvre clusters, and scattered individual manoeuvres are uninteresting. Before describing the algorithm more formally a few terms have to be defined.

[83] The K-means algorithm can handle arbitrary numbers of dimensions for the input data. The problem lies only in the chosen attributes, not in the algorithm itself.

[84] cf. *Ester* et al., DBSCAN, 1996, p. 2.

Epsilon Neighbourhood: The epsilon neighbourhood of a point p, denoted by $N_\epsilon(p)$, within the data set of points D is defined by:

$$N_\epsilon(p) = \{q \in D | d(p,q) < \epsilon\}$$

This means that the epsilon neighbourhood includes all points in D for which the distance to p is less than ϵ.[85]

Directly Density-reachable: A point p is directly density-reachable from a point q ($q \rightarrowtail p$) if:

1. $p \in N_\epsilon(q)$ and
2. $|N_\epsilon(q)| \geq \mu$

The two conditions that have to be satisfied are that p must lie in the epsilon neighbourhood of q, and that the epsilon neighbourhood of q has enough points to be considered a cluster, measured by the minimum number of points needed for a cluster μ.[86]

Core and Border Points: "There are two kinds of points in a cluster, points inside of the cluster (core points) and points on the border of the cluster (border points). In general, an epsilon neighbourhood of a border point contains significantly less points than an epsilon neighbourhood of a core point."[87]

Density-reachable: A point p is density-reachable from a point q ($q \dashrightarrow p$) if:

$$\exists (p_1, \ldots, p_n) | p_1 = q \wedge p_n = p \wedge \forall p_i (p_i \rightarrowtail p_{i+1})$$

This means that p is density-reachable from q if there is a chain of points beginning with q and ending with p, in which each point is directly density-reachable from its predecessor.[88]

With these terms it is possible to understand the concept of the DBSCAN algorithm, which is now presented verbally:

> "To find a cluster, DBSCAN starts with an arbitrary point p and retrieves all points density-reachable from p with regards to ϵ and μ. If p is a core point, this procedure yields a cluster [...]. If p is a border point, no points are density-reachable from p and DBSCAN visits the next point of the [input data set]."[89,90]

[85] cf. *Ester* et al., DBSCAN, 1996, pp. 2 f.
[86] cf. *Ester* et al., DBSCAN, 1996, p. 3.
[87] *Ester* et al., DBSCAN, 1996, p. 3.
[88] cf. *Ester* et al., DBSCAN, 1996, p. 3.
[89] *Ester* et al., DBSCAN, 1996, p. 4.
[90] The algorithmic pseudo-code of DBSCAN is not presented, as many lines of pseudo-code are necessary to convey a formalized version of the textual description above, without adding any value at this point. It can be found at *Ester* et al., DBSCAN, 1996, p. 4.

Now that the functional principle of the DBSCAN algorithm has been presented, it is applied to the manoeuvres of the race. However, one of the requirements deduced from the shortcomings of the K-means example was to include more manoeuvre attributes than simply the location. The metric[91] to measure the distances between manoeuvres that is now introduced is derived from the Euclidean distance, as no suitable standard metric could be found that matches the problem at hand. Instead of limiting the space to the vector space $\mathbb{R}^n$, dimensions of any kind are allowed, where one dimension exists for each attribute of a manoeuvre. This means that it may not be possible to use subtraction to calculate the distance between manoeuvres for every dimension, so that a distance function e is needed for each dimension. The following symbol definitions lead up to the custom-built equation defining the metric calculation:

$$
\begin{aligned}
p, q &: \text{manoeuvre} \\
p_i &: \text{manoeuvre attribute} \\
w_i &: \text{attribute weight}, w_i \in [0, 1] \\
e_i\left(p_i, q_i\right) &: \text{distance between attributes} \\
d\left(p, q\right) &: \text{distance (metric) between manoeuvres} \\
d\left(p, q\right) &= \sqrt{\sum_{i=1}^{n} \left[w_i \cdot e_i\left(p_i, q_i\right)\right]^2}
\end{aligned}
$$

The manoeuvre attributes listed in table 1 on the next page each represent one dimension of the input data. For each of these attributes – position, time point and turn angle – the distance calculation between two manoeuvres within that dimension is specified. This is used by the overall metric to calculate the distance between two manoeuvres. These attributes have very different characteristics, which is why a sensible reference value range for or a specific reference value is provided for each of the attributes. The reference values are included as denominators in the distance calculations to normalise the attributes to make them comparable.

As each attribute distance is multiplied with the attribute weight in the overall metric, separating the attribute weight and reference value may be mathematically redundant, but important for comprehension. The reference value serves as a normalizing factor, whereas the attribute weight determines the influence of an attribute

[91] *Metric* is a fixed mathematical term for a function that must fulfil a variety of conditions. This conditions are not proven to hold true for the introduced distance measure, which is why naming it *metric* is not mathematically correct.

[92] The number of seconds that have passed since midnight, January 1, 1970 *Universal Time Coordinated* (*UTC*).

Name	**Description**	**Distance Calculation** $e_i\,(p_i, q_i)$
Position	The position p_c of the manoeuvre centre, assumed to be where the Douglas-Peucker point lies. Identified through easting and northing values x and y. Reference value: $r_c \in (0, \text{longest distance in course}]$	$\frac{\sqrt{(p_x - q_x)^2 + \left(p_y - q_y\right)^2}}{r_c}$
Time Point	The time point p_t at which the manoeuvre was performed, assumed to be the time point at which the fix of the Douglas-Peucker point was recorded in *Portable Operating System Interface* (*POSIX*) time.[92] Reference value: $r_t \in (0, \text{race duration in seconds}]$	$\frac{\lvert p_t - q_t \rvert}{r_t}$
Turn Angle	The turn angle p_γ of the manoeuvre, somewhere in-between $[-180°, 180°]$, so that it also includes the turn direction. In reality, the turn angle might be outside of this range, but by using only one Douglas-Peucker point to represent the manoeuvre this cannot be identified, as it could only be guessed at whether the manoeuvre covered e.g. a turn angle of -90° or 270°. Reference value: $r_\gamma = 360°$	$\frac{\left\lvert p_\gamma - q_\gamma \right\rvert}{r_\gamma}$

Table 1: Manoeuvre Attributes

towards the distance between manoeuvres. A higher attribute weight increases the manoeuvre distance if the attribute value is not zero, therefore compressing the epsilon neighbourhood in the direction of that attribute so that fewer manoeuvres are included. This is shown in figure 34 on the following page for an exemplary two-dimensional case. In general, changing the weights will stretch or compress the hyper-ellipsoid representing the epsilon neighbourhood.

Figure 35 on page 67 shows the result of applying the DBSCAN algorithm to the manoeuvres of a sailing race. The values chosen for the Douglas-Peucker algorithm, the DBSCAN algorithm and the attribute reference values and weights are also shown.

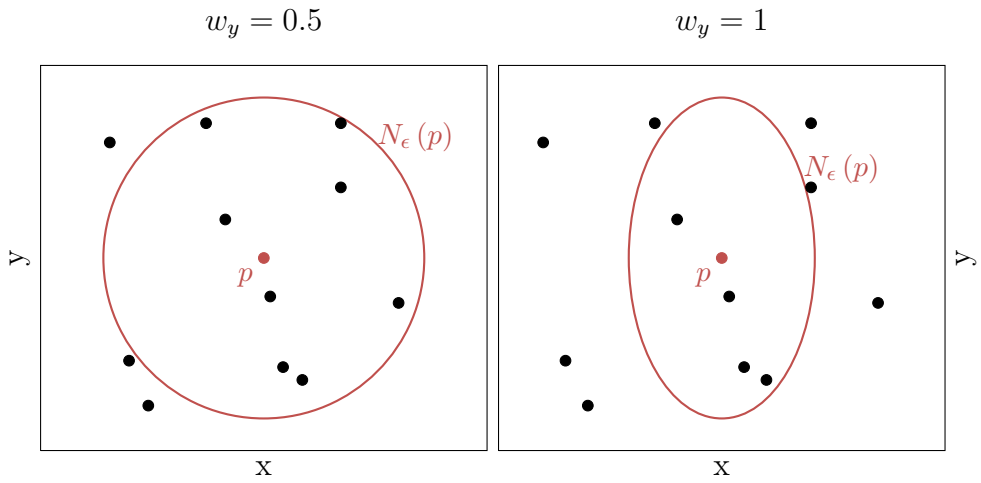


Figure 34: Influence of Weights on the Epsilon Neighbourhood

The following observations can be made:

1. The visually rather dense manoeuvre clusters on the right-hand side of the course were – correctly – not identified as clusters by the DBSCAN algorithm. The reason for this is that while they are close in position, their turn angles and especially time points differ too much.
2. Another effect of the time point attribute is that the single wind-ward buoy is represented through two clusters, as it was used twice in the waypoint sequence. This is of a positive effect, as it eases the task of later identifying the waypoint sequence.
3. In addition to the two actual control points of the course layout – the wind-ward buoy and the lee-ward gate – three more clusters on the left-hand side of the course were found. These can be disqualified by raising μ, however this means that the gate clusters are lost. A possible solution to this dilemma is presented in the next item.
4. Even though only eight sailboats competed in the race, both clusters at the wind-ward buoy contain more than eight manoeuvres. This means that for some competitors, multiple manoeuvres are counted, and influence the decision whether or not a group of manoeuvres is identified as a cluster. Therefore, an additional evaluation step should be performed, in which the number of competitors that have a manoeuvre in each cluster is checked. If there is at least one manoeuvre for every competitor in a cluster, it can be counted as a likely waypoint position. This approach can also be used to solve the gate problem: First, perform the clustering with a low value for μ. Then, in addition to finding all clusters that have manoeuvres for all competitors,

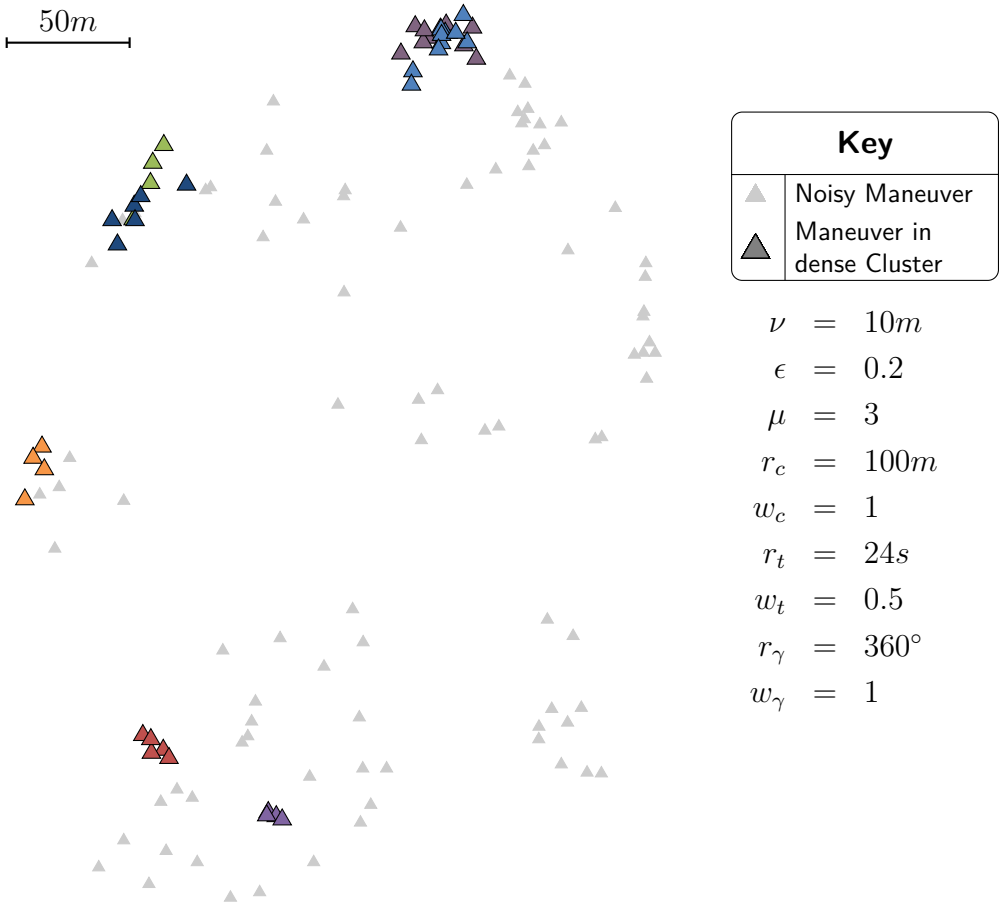


Figure 35: DBSCAN Manoeuvre Clustering

find those that contain manoeuvres for at least half of the competitors, and consider them as one buoy of a gate if another cluster can be found close by that contains manoeuvres for exactly the missing competitors.

5. The high number of parameters also listed in figure 35 presents one of the main problems of this clustering approach. Though some values, such as the reference value r_γ for the turn angle attribute, can remain constant, the quality of the clustering result depends on the correct parameter settings depending on the course type and boat class, which for example influences the boat speed, which in turn has an influence on the optimum value for ν.

7.5.4 Evaluation

Using a clustering method has great benefits, but these benefits come at a price. While clustering can yield good results for all course types and boat classes, the quality of the results depends strongly on the correct parametrisation of the clustering method. By changing these parameters it is possible to switch between the levels of detail the clustering method provides: If time, position and turn angle are included with a sufficient weight, the result is more fine-grained, providing the individual mark positions. The parametrisation also has a great influence on the ratio of false positives and false negatives, where a good balance is essential, but also difficult to obtain. As a whole, clustering methods can be deemed as promising, but too fragile to be of any real use – at least with the approaches tested so far.

7.6 Direction Changes

7.6.1 Concept

Up to now the goal of finding a course layout heuristic was tackled through data mining techniques. However this should not limit the evaluated approaches to complex solutions. As sailboats move from waypoint to waypoint and manoeuvre at each waypoint, they change their direction on reaching a waypoint – which is not a new observation. Due to the fact that sailboats do not move in straight lines in-between waypoints, these direction changes stand out less than they would for a motorboat.

As sailboats can't sail directly into the wind, they beat upwind with a series of tacks. The same can be observed for most boat classes on downwind legs, as running directly downwind does usually not result in the highest downwind VMG. Figure 36 on the next page shows how this knowledge can be used for detecting direction changes even though manoeuvres occur far more often than direction changes. The direction the sailor wishes to move in is assumed to be the angle bisector of the last two headings in-between manoeuvres (and the start point), which are labelled as *1* to *4*. The direction of movement most likely has changed at the indicated point, because the bisector angle of the heading directly before *(3)* and after *(4)* this point is not equal to the bisector angle of the two preceding headings *(2 and 3)*.

At least three different headings are necessary to decide whether the direction has changed – the first two are used to calculate the current heading of the sailboat,

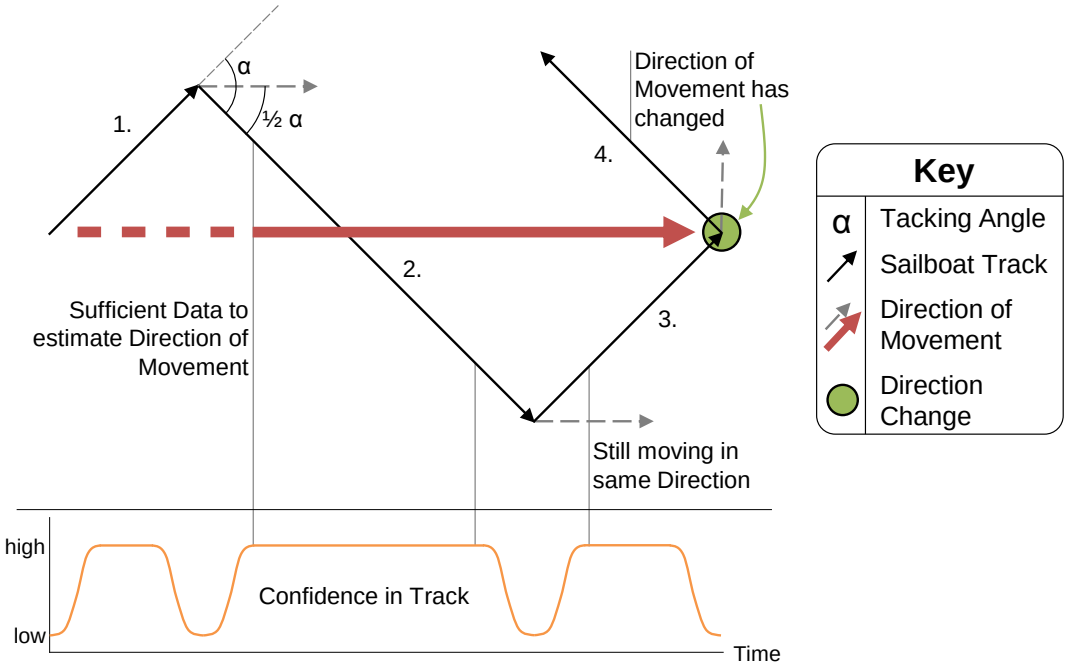


Figure 36: Change in the Direction of Movement

and the bisector angle of the second and third is compared to that heading. If the course is laid out in such a way that the first manoeuvre after the start or after any of the following detected direction changes is in reality a manoeuvre around a mark – considered here as a direction change – this cannot be identified.

To judge the heading of a boat in-between manoeuvres, the confidence in the headings for points close to the manoeuvres should be significantly lower, as in reality manoeuvres are not performed in an instant, and the values for the heading during the manoeuvre are significant. This is why a confidence curve is also shown in figure 36, relating to the headings *1* to *3*. Also, a certain margin for difference between consecutive bisector angles must be allowed for, as they will – in reality – never be exactly equal.

7.6.2 Application and Evaluation

Figure 37 on the following page shows the detected direction changes in the exemplary race. All manoeuvres that were performed at each waypoint were recognized, with a few superfluous direction changes at the wind-ward buoy.

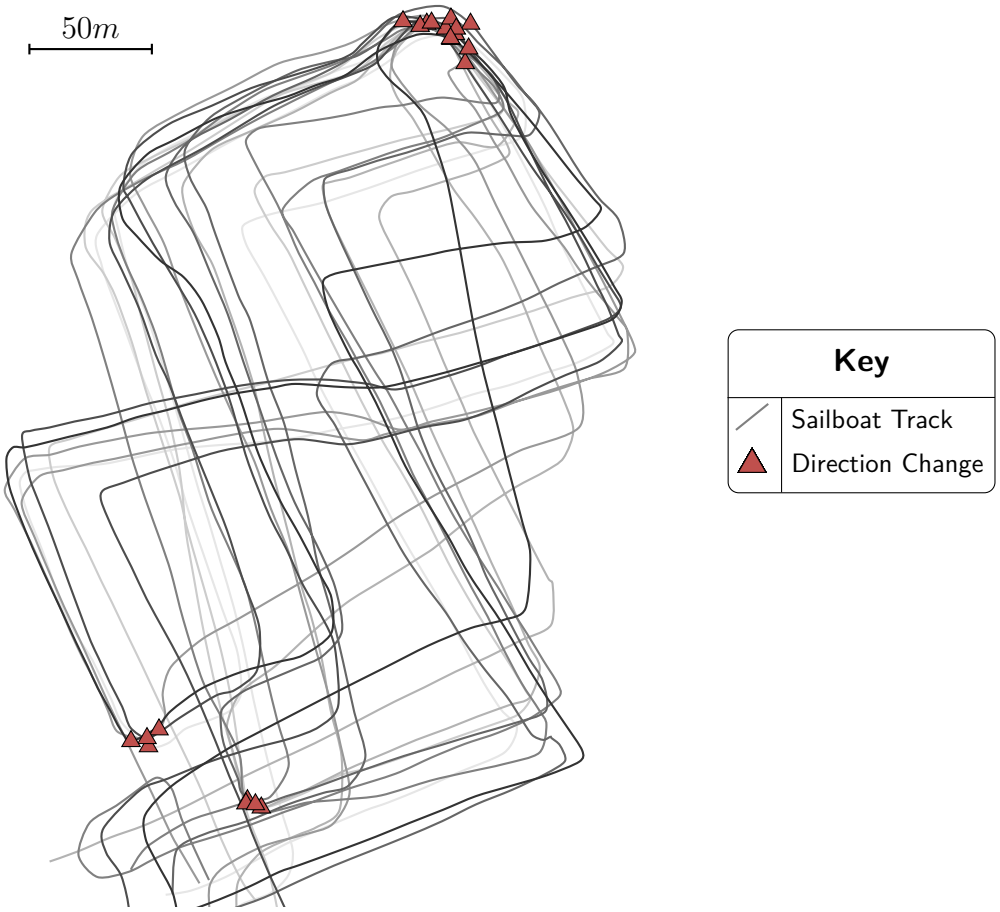


Figure 37: Direction Changes as Mark Position Indicators

Even though the approach yields good results for this special case, it is limited in its scope. The two common course layouts shown in figure 11 on page 14 both are too complex to be identified through this concept of direction changes, due to the necessity of three headings after the start and each direction change. The third waypoint of these courses – the first waypoint on the left-hand side of the course after the wind-ward buoy – is reachable by travelling only on starboard tack without manoeuvring in-between. Therefore, it is not identified as a direction change and thus not identified as a waypoint.

In general, the second control point of any leg that can be completed without a manoeuvre will go undetected. This fact limits the applicability of the direction change concept as a heuristic for finding the course layout. However, it is a fairly simple approach and yields good results for courses that are known to contain no

such legs.

7.7 Other Approaches

7.7.1 Artificial Neural Networks

ANNs find their application in the data mining method of classification. As the training process of an ANN only requires sample input values and – most often – expected output values, ANNs are the ideal technique for domains in which the underlying patterns are not understood. Though an effort has been made to formalize the characteristics of a waypoint position so that clustering methods could be applied, ANNs might achieve the same or better results with less pre-interpretation of the input data.

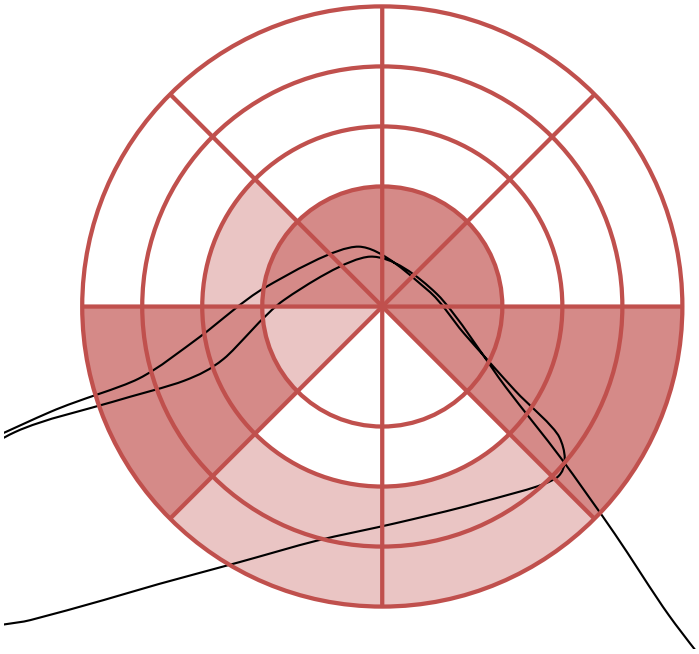


Figure 38: Annular Section Shape Descriptor[93]

A conflict arises between providing input data in a pure fashion, and finding a representation of this data suitable for an ANN – which only accepts input data sets that can be described by a vector $\vec{x} \in \mathbb{R}^n$ as explained in chapter 6.3 on page 47.

[93] cf. *Fathi/Krumm*, Detecting road intersections, 2010, p. 61.

The so-called shape descriptor used by *Fathi/Krumm* is a simple approach which has been used successfully to detect road intersections from GPS traces.[94] An exemplary shape descriptor composed of annular sections is shown in figure 38 on the previous page, and serves to explain the principle of shape descriptors in general:

- A shape descriptor is composed of various *histogram bins* – in this case in the form of annular sections.
- When positioned over a certain set of tracks, the number of tracks crossing through each of these bins is counted. A higher number of crossing tracks is visually indicated through a darker fill colour for that bin.
- The state of a shape descriptor when positioned over a certain set of tracks can be represented through a vector $\vec{x}$ by iterating over the bins and using the number of tracks crossing through each bin as the value of one row in $\vec{x}$. Optionally, the rows can be normalised by dividing $\vec{x}$ by the highest number of tracks in a bin, so that all values lie in $[0, 1]$.[95]

An ANN can then be used to classify the state of a shape descriptor. While the classification in the work of *Fathi/Krumm* distinguished between *road intersection* and *no road intersection*, for this scenario the classification should distinguish between *waypoint* and *no waypoint*. The training data is obtained by providing positive and negative examples of shape descriptor states placed at waypoint positions and not at waypoint positions. After training the ANN, the shape descriptor can be moved from point to point of a grid superimposed on the course layout, classifying each of the grid points.[96]

However, using the back-propagation training algorithm combined with a feed-forward ANN with two hidden layers, no stabilisation of the error rates could be achieved, so that no reliable classification of waypoint positions is possible. The possible reasons could include the following:

- *Fathi/Krumm* rotate the shape descriptor to a computed canonical orientation of the intersection, so as to make it rotation invariant for differently oriented road intersections. However, a set of sailboat tracks does not follow the same pattern as a set of car tracks on two roads[97], so that no simple canonical orientation of a set of sailboat tracks can be found. If the wind could be assumed static, this might be possible. However, the wind direction can change throughout a race, and the tracks of the entire race have to be used because no information exists

[94] cf. *Fathi/Krumm*, Detecting road intersections, 2010, pp. 56 ff.
[95] cf. *Fathi/Krumm*, Detecting road intersections, 2010, p. 60.
[96] Analogous to the approach in *Fathi/Krumm*, Detecting road intersections, 2010, pp. 60 ff.
[97] cf. *Fathi/Krumm*, Detecting road intersections, 2010, pp. 60 f.

yet on how to subdivide it in a sensible fashion.

- Also, *Fathi/Krumm* trained the ANN using the Adaboost algorithm, which might also provide better results for the classification of waypoint positions.[98]
- In comparison to the road intersection classification, the waypoint classification may simply be too complex for either the shape descriptor representation, the chosen ANN layout, or the training algorithm. Changing any or several of these factors may lead to better results.

The suitability of ANNs could not be conclusively determined due to time considerations, which is why the reasons above are simply stated and not evaluated in terms of their truthfulness.

7.7.2 Genetic Algorithms

Even though the applicability of evolutionary algorithms could not be verified due to time considerations, a possible approach shall be briefly outlined by discussing the key features of the problem representation: the genetic representation (phenotype structure) and evaluation (fitness function) of the solution domain. For this approach, the input data is again represented through the Douglas-Peucker based manoeuvres.

As phenotypes represent possible solutions that are evolved to find the best solution, the structure of a phenotype should represent a course layout. To keep the approach simple, only the mark positions, but neither the waypoint sequence nor the grouping of marks into control points shall be encoded by a phenotype. However, the number of marks shall not be predefined but remain variable. Therefore, a phenotype consists of a variable number of chromosomes, each of which represent one mark by its position.

The fitness function is not presented here as a function, but should incorporate the considerations that are now drawn up verbally.

- Re-using the assumption of a high manoeuvre density at mark positions, the fitness function should be based on the number of manoeuvres in the vicinity of each mark position (chromosome) of a phenotype. As the number of chromosomes is variable, each manoeuvre should count only once at the most to prevent ever-increasing phenotype sizes. This also prevents phenotypes from developing chromosomes that lie too close together.

[98] cf. *Fathi/Krumm*, Detecting road intersections, 2010, p. 62.

- A lower number of chromosomes should lead to a higher fitness. This prevents phenotypes from developing enough chromosomes to simply include all manoeuvres, no matter whether or not these are in dense clusters. This part the centrepiece of the whole approach, as it most be balanced well, so that it does not lead to phenotypes with too few or too many chromosomes.
- In addition to cluster sizes, the fitness function should also assess the quality of each cluster. The number of competitors with a manoeuvre included in the cluster should be rewarded separately as a result of the previously discussed inequality of cluster size and the number of competitors represented by the cluster. Also, the variance of manoeuvre attributes such as the turn angles or time points provide a good indicator of the likelihood of a mark position, so that a lower variance should result in a higher fitness.

The genetic representation and fitness function discussed above form a solution domain that can be considered similar to the presented clustering methods. While this is not necessarily bad, it means that this approach also suffers from the same problems – especially the complexity and number of the parameters that will become necessary when detailing the fitness function.

7.8 Further Steps

Of the identified criteria, completeness is the factor that all presented approaches lack the most. Although a basic idea for identifying gates was outlined in the context of clustering methods, more work has to be put into the identification of the other factors defining a course layout apart from the individual mark positions. Even the identification of mark positions still contains unresolved issues, as the identification of a manoeuvre cluster cannot be equated with finding the position of the mark. Using the geographic midpoint of the manoeuvre positions may be a first, approach, but as all competitors can be expected to have rounded a mark this can only provide a rough yet flawed estimate, as it places the mark in-between the sailboat tracks, instead of inward of all tracks.

The problem of adding additional levels of completeness, as well as the problem of parameter complexity might be solved by the renewed application of data mining techniques. Control points could be identified as clusters of similar waypoints, and a good parametrisation of a clustering method for a race might be achieved not through perfect default values, but by comparing and developing good parametrisations on a race-by-race basis through a genetic algorithm.

On a more general note, all presented approaches were shown to have different strengths and weaknesses, so that combining different approaches is one of the many possible further steps towards obtaining a complete, accurate, and generalisable course layout heuristic. At this point, all approaches remain true to their name – they are still approaches, and not ready for operation. If both the combination of approaches, as well as further research into the field of genetic algorithms should prove unsuccessful in future, the aspiration to identify course layouts without any prior knowledge may be to set too high. In this case, a variety of options exists. On is to evaluate the possibility of specialized heuristics for special course types, another is providing input capabilities for an approximate course layout which is then automatically refined using the sailboat tracks.

There are many more related steps and factors that need consideration, such as the detection of start and finish time and line positions, mark passings times, and especially the robustness of all these heuristics to confounding influences such as rule infringements.

8 Integration of Mobile Devices

8.1 Introduction

This chapter gives insight into the phases of analysis, design and implementation of the integration of mobile devices into SAP Sailing Analytics. The requirements for this integration were presented in chapter 5.3 on page 41. A use case diagram is first used to analyse those requirements which are considered non-technical. After this, a high level overview of the components within SAP Sailing Analytics that have to be altered or extended is given, the most important of which are then discussed in detail.

8.2 Analysis of Requirements

The functional requirements defined in chapter 5.3 on page 41 for the integration of mobile devices are analysed in this section. This lays the foundation for discussing the design and implementation of the some of the important extensions. As a first step, these requirements are abstracted through a UML use case diagram.

The original requirements point into two directions: On the one hand smart-phones or pre-recorded data from other devices shall provide an additional source of tracking data next to the existing tracking providers, without touching the concept of SAP Sailing Analytics as a centrally administered tool. On the other hand, coaches and sailors shall be enabled to create, register and track training sessions and races on their own as easily as possible. The two use cases *create closed race* and *create open race* in figure 39 on the following page address these different requirements. While a coach creates *open* or *closed races*, and then *starts* or *stops* these, a sailor only accesses the use case of *registering for an open race*. The two *create* use cases connected to the coach both extend the same abstract *create race* use case, which groups common functionality. It includes entering basic race data such as *boat class*, *start time*, and the *race name*, as well as the definition of the *course layout*.

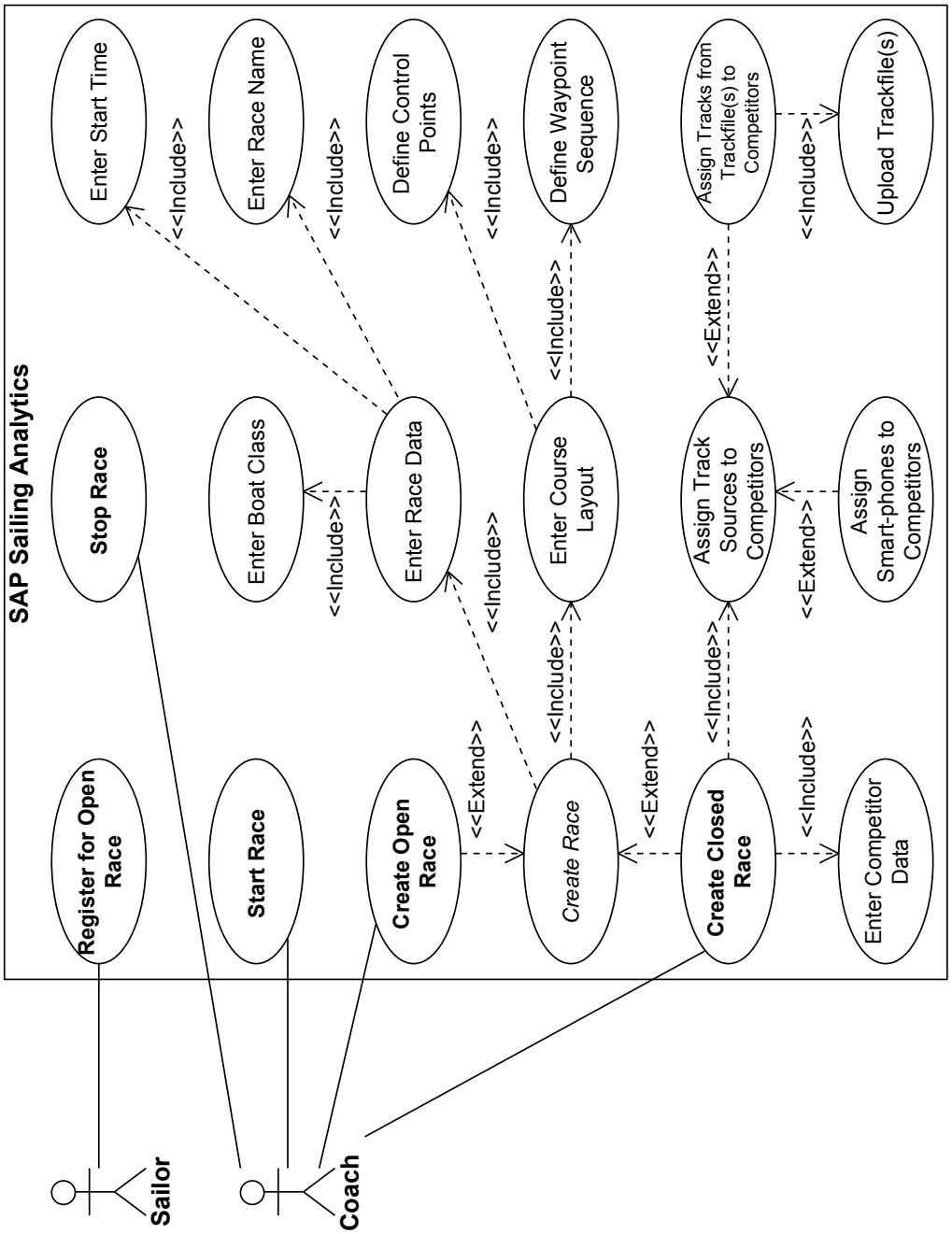


Figure 39: Mobile Devices Integration Use Case Diagram

The course layout is entered by *defining control points*, and then creating a *waypoint sequence* from these control points. As soon as the course layout heuristic is sufficiently developed, this step can be omitted, or merely be included to check and refine the automatically identified course layout. Until then the course definition

has to be entered by hand by viewing the sailboat tracks. This can only be done once the sailboat tracks are available, which is after the race.

The two major use cases are now discussed in more detail:

Create closed race: This is very similar to the current functionality of SAP Sailing Analytics, as all necessary data, including the competitors, is known beforehand. One actor – a coach – deals with assembling all of this data beforehand, so that it is never "open" for any sailor to join directly. In addition to entering the competitor data, he also specifies where the tracking data for these competitors comes from: either from trackfiles which he may upload (in this case the race has already taken place), or from smart-phones (in which case the race may still take place). This adds a new source of tracking data, in which no intermediary tracking provider is involved, but which is otherwise similar to the current setup.

Create open race: This use case aims at enabling a simpler form of tracking training sessions or small races, an "open" race, for which sailors then can simply register – all through the use of smart-phones. By defining an open race, the race data is entered as a first step. After this, however, the session is in a state in which sailors can register for it. When the coach starts the session, the sailors that have registered are the competitors for this session. From this point onward, the procedure is identical to the last steps of creating a closed race. Even though it has been named open race, this use case also applies to training sessions.

8.3 Integration into the current Architecture

Figure 40 on the next page is derived from the architecture overview in chapter 3.4.1 on page 25 and shows a high-level overview of the the changes and improvements that are necessary to integrate mobile devices into SAP Sailing Analytics. The app is an entirely new component, which is being developed in parallel in the context of a different bachelor thesis, representing the *client* side integration of mobile devices into SAP Sailing Analytics. The communication between the app has been designed to use two channels: the *Representational State Transfer*[GL] (*REST*) API and *User Datagram Protocol* (*UDP*). The reasons for these design decisions, and a detailed specification of these communication channels is included in this chapter. The extensions to the domain model, the MongoDB, which is used to store the trackfiles and thereby ensure server side statelessness, and the *RacingEventService* and *GWT* bundles are not further explained, as they deal only with small parts of the big picture. Replication and Persistence (apart from the trackfiles) could not be dealt with due to the complexity and time consuming nature of the other requirements, and

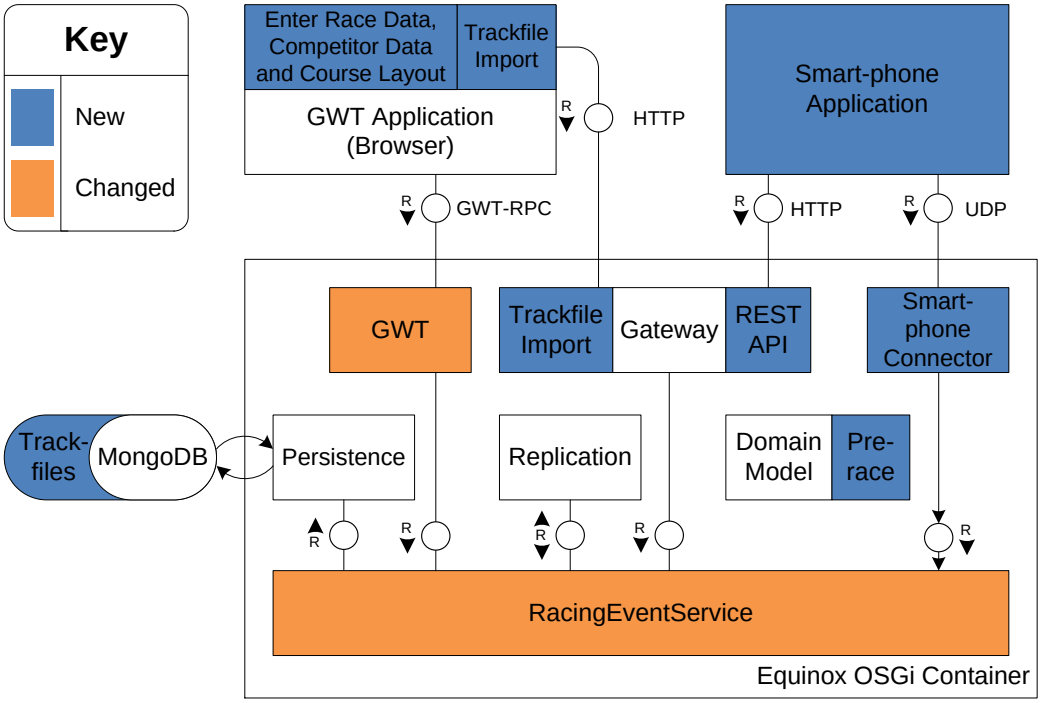


Figure 40: Mobile Devices Integration Architecture

are therefore not included in the highlighted changes in figure 40. Appendix B on page 97 includes some screen-shots and short explanations documenting the additions to the GWT browser application that provide input functionality for basic race data, competitor mapping, trackfile uploads and course layouts – the latter of which is necessary as the course layout heuristic could not yet be developed sufficiently to render this unnecessary.

8.4 Pre-race Domain Model Addition

The process of creating, starting and stopping an open race is not that different from a closed race. The new term *pre-race* is derived from the additional state an open race has when compared to a closed race, as depicted in figure 41 on the next page. In this state, the race is not fully defined, and awaiting the registration of competitors. Once it is started, the race becomes active, which transforms the pre-race into the actual race, which is more static, and contains references to mostly immutable objects and collections. For example, the list of competitors in a race

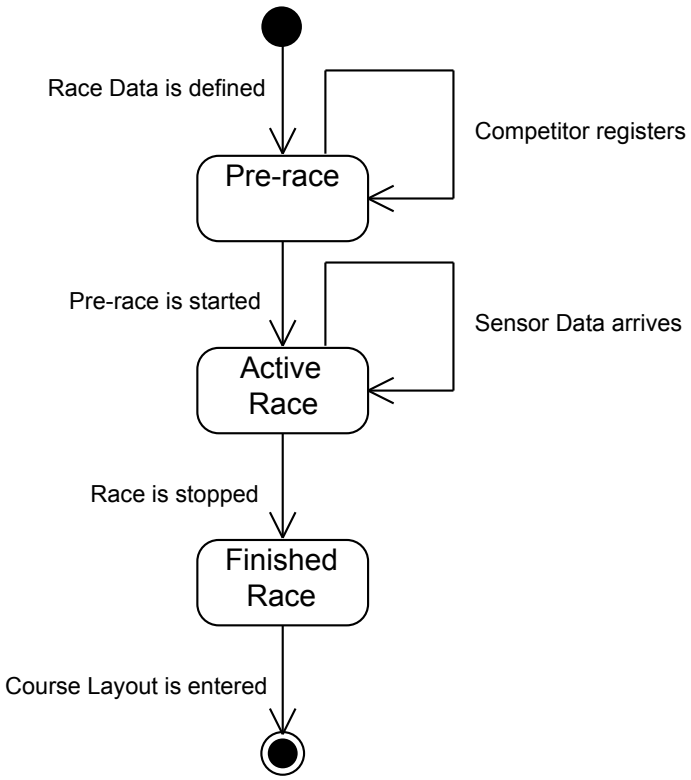


Figure 41: UML State-chart of an Open Race

is immutable, which is the reason for introducing the state of pre-race in the first place.

8.5 Linking Smart-phones, Competitors and Races

To enable smart-phones to identify races and competitors when communicating with SAP Sailing Analytics, *Universally Unique Identifiers*[GL] (*UUIDs*) have been introduced. Each smart-phone is also identified through a device identifier, either the *International Mobile Station Equipment Identity* (*IMEI*), *Mobile Equipment Identifier* (*MEID*), or *Electronic Serial Number* (*ESN*) depending on the mobile network infrastructure the smart-phone uses. Two different strategies for linking smart-phones to a competitor and race, based on the two main use cases of race creation, are proposed and have been implemented for this thesis:

Open Race: In an open race, the sailor himself decides to register for a race in its pre-race state. When registering, the smart-phone also transmits its device identifier

in addition the information on the sailor wishing to register, so that all sensor data that it sends from now on can be linked to that competitor, for that race.

Figure 42: QR-Code used for assigning a Smart-phone to a Competitor

Closed Race: In a closed race, the coach enters the competitor data, and may assign a smart-phone to a competitor. Instead of registering all smart-phones that are going to be used before-hand, and then selecting one of these, a simpler process of registering the physical smart-phone to the competitor is proposed. A *Quick Response Code* (*QR-Code*) – a two-dimensional bar-code, that can encode text-based data – is generated for each competitor that shall be tracked by a smart-phone, and graphically encodes the UUID identifying the competitor. The QR-Code can be scanned with a smart-phone, which then lets the link between the physical device and pre-defined competitor and race snap into place by automatically registering its device identifier for this closed race and competitor.

8.6 Smart-phone Race Management Interface

8.6.1 Base Protocol Choice

The focus of an open race is simplicity, so that it can be created, registered for, and started by using smart-phones only. As the app cannot use the proprietary GWT-RPC communication channel utilized by the GWT browser application, a different communication channel has to be used. To limit the effort and ensure easy integration, this channel shall use existing standards for web services, with the *Hypertext Transfer Protocol* (*HTTP*) as the application layer protocol on top of the *Transmission Control Protocol* (*TCP*)/*Internet Protocol* (*IP*) protocol stack. Essentially, two options present themselves here: *Simple Object Access Protocol*[GL] (*SOAP*) and REST web services. The discussion over which of these paradigms to follow shall not be drawn out at length here: the choice is in favour of REST-ful web services, as they require less definition and transmission overhead, but instead rely on a well-defined resource model and clean semantic usage of the standard

HTTP request methods. The structure of REST-ful web services and the definition and evaluation of the smart-phone API enabling coaches and sailors to create and register for open races is discussed in the remainder of this section.

8.6.2 REST-ful Web-services

The REST API is one of the two communication channels through which smart-phones can communicate with SAP Sailing Analytics. This API effectively is a group of web-services exposing data and functionality residing on the server tier to smart-phones acting as clients. As it conforms to the REST architectural style, it is described as a REST API. A brief introduction into the implications of this architectural style for web-services is given and then applied by presenting an excerpt of the defined API.

The REST architectural style describes a client-server architecture that is centred on the notion of resources. These resources can be uniquely identified and accessed through a set of well-defined operations by sending self-descriptive messages. The client must present all relevant information in the message, thus relieving the server of having to retain any client-related state. The web-standard HTTP implements this architectural style, and it is through the significance of HTTP that the term REST API has become synonymous to a group of HTTP-based web-services that are defined and implemented in a REST-ful way – that is, following the REST principles.

Resources – identified through their *Uniform Resource Identifiers* (*URIs*) – are hierarchically structured and can be accessed through four main operations, the HTTP request methods: `GET`, `POST`, `PUT`, and `DELETE`.[99,100] The HTTP standard defines that `GET` should be used to retrieve information only, without any side-effects that the client can be held accountable for. That is why it is one of the three methods that are defined as idempotent: `GET`, `PUT`, and `DELETE`. This means that the side-effects for any number of requests should be the same as for a single request.[101]

By applying the further definitions of the request methods in the HTTP standard to the idea of REST-ful web-services, rules for the structure of a REST API can be derived. One of the structural elements categorizes a resource as one of four resource types:

[99] cf. *Wilde/Pautasso*, REST, 2011, pp. 5 ff.
[100] There are three more methods: *OPTIONS, TRACE,* and *HEAD*.
[101] cf. *Fielding* et al., Hypertext Transfer Protocol 1.1, 1999.

Document: A singular component, comparable to an object instance or a database record.

Collection: A server-managed directory of resources. Clients may only propose creating, altering or deleting resources, but whether these actions are performed, and what URI each resource should be identified through, is decided by the server.

Store: A client-managed resource repository. The URI of the stored resources, and anything action creating, altering or deleting them are solely initiated by the client.[102]

Controller: Similar to an executable function in a procedural concept, optionally expecting input and return output values. This type of resource is used whenever an application-specific action cannot be logically linked to one of the four standard methods.

With these four resource types in mind, the standard usage for the four HTTP methods for these resource types can be defined.

	Document	**Collection**	**Store**	**Controller**
`GET`	Retrieve the state of a resource, in some representational form.			
`POST`	-	Create a new, contained resource	-	Execute the function
`PUT`	Insert a new resource into a store or update an existing, mutable resource			
`DELETE`	Remove the resource from its parent			

Table 2: Standard Usage of HTTP Request Methods[103]

The two combinations of method and resource type that have no defined semantics generate errors. The HTTP standard defines numbered status codes with well-defined semantics. As a status code is included in every server response to a client request, a client may interpret the success or error resulting from his request accordingly. These status codes are split into five groups by the first digit in their respective number.

1. Informational (1xx)
2. Success (2xx)
3. Redirection (3xx)
4. Client Error (4xx)

[102] cf. *Wilde/Pautasso*, REST, 2011, pp. 32 ff..
[103] *Wilde/Pautasso*, REST, 2011, p. 33.

5. Server Error (5xx)[104]

8.6.3 Smart-phone API Definition

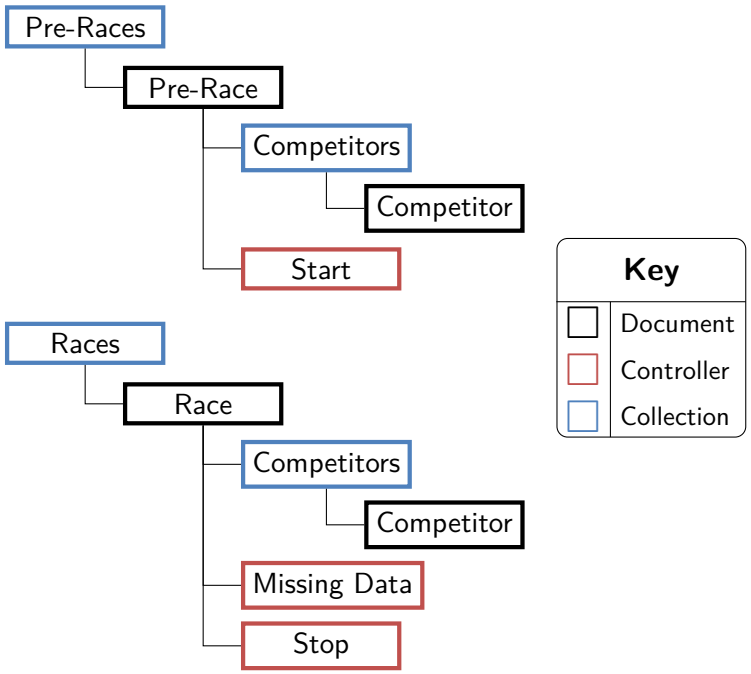


Figure 43: Hierarchical Resource Model of REST API

The smart-phone REST API for SAP Sailing Analytics can be assembled using these semantics, resulting in an API that is very expressive of the underlying resource model. This underlying resource model is represented in figure 43. It consists of two top-level collections, *pre-races* and *races,* that store documents of their respective kind. Among the data associated with each of these documents is another collection for the registered *competitors.* In addition, each pre-race has a controller to *start* it, thus transforming it into a race, allowing tracking data to be received and recorded for this race from now on. Similarly, the race can then also be *stopped*, which ends the tracking and cleans up the occupied resources for this race. Also, a smart-phone may inquire as to which data packets have not been received by the server, and therefore *missing,* so that they can be re-sent.

Table 3 on the next page is an excerpt from the full API definition shown in appendix C on page 102. Its purpose is to briefly explain the process of translating the

[104] *Wilde/Pautasso*, REST, 2011, p. 28.

Resource	Method	Details
`/preraces`	`GET`	List all pre-races *Out* `200 ok` `{pre-races[]}`
	`POST`	Add new pre-race *In* `name, start_time` *Out* `201 created` `{uuid, name, start-time}` *Errors* `400 bad input` *Effects* Adds the pre-race, generating a new `uuid`, only if `name` contains non-white-space characters and `start-time` lies in the future.

Table 3: Pre-races Collection in the Smart-phone REST API

resource model into a REST API and also explain the structure chosen to represent the API. The resource in question – the pre-races collection – has two defined methods through which it can be accessed. This is due to the fact that it is a collection, for which only the semantics of the methods `GET` and `POST` are defined. The URI of the resource is shown only in part, as the domain and servlet context parts of the URI are irrelevant, and the protocol is always HTTP. For each of the methods defined for a resource, the details inform on the semantics, the expected input values, the output values, the possible errors and side-effects of the method. If one of these is insignificant or non-existent, it is omitted.

For example, when performing a `GET` request on the pre-races collection, no input values are expected. The server simply responds with the status code *200*, which implies that the request is valid and could be processed, and inserts a JSON document containing an array of all current pre-races in the response body. For this method, no errors are to be expected, as the worst case is an empty JSON array in the response. This is a little different for the POST request, as here either an invalid name or a start-time lying in the future can result in the response status code *400*, meaning that the supplied data is not accepted. If this is not the case, a new pre-race is created and its JSON representation is returned in the response. The reason a POST request is used to create a new pre-race, is that for collection resources the server decides whether to and under what resource identifier to create this pre-race. By applying the same rules to the other resources the complete REST API in appendix C on page 102 is derived. However, it is not exhaustive, as not all methods semantically possible for all resources are defined, as they have been

deemed less important.

Resource	Method	Details
`/preraces/<uuid>/competitors`	`POST`	Register competitor for pre-race `<uuid>` *In* `name, sail-id, device-id, geo-location` *Out* `201` *created* `{uuid, name, sail-id, device-id, geo-location}` *Errors* `400` *bad input* `404` *pre-race not found* *Effects* If no competitor with the same device_id was previously registered for pre-race `<uuid>`, and all input data is valid, a registration is created.

Table 4: Competitor Registration in the Smart-phone REST API

Table 4 shows the registration process for a competitor wishing to join a race. Among the necessary information he must provide is the *device-id*, which is used later to identify any data transmitted from this device and link it to the correct competitor. Currently, this is provided by the device. As an alternative approach, the server might also generate an identifier and transmit this back to the smart-phone in the response body.

8.6.4 Evaluation

An analysis of the defined REST API reveals some shortcomings. It is incomplete and in some places focuses on enabling a first end-to-end showcase, rather than representing its final version. For example, the competitors exist currently only as resources nested within pre-races and races, and do not have a life-cycle of their own. Instead, a competitor is both registered and created in the context of a pre-race at the same time, and is then migrated to the corresponding race once it is started. One benefit of having a smart-phone based solution is the fact that every sailor with a smart-phone already possesses a personal tracking device. In the SAP Sailing Analytics app, in the future the sailor could already be logged in, thus linking the device to a competitor profile. When registering for a race or training session, this existing profile should merely be linked to the race, decoupling their life-cycles.

Also, security concerns are not dealt with at this point. A wide variety of steps that can be taken is imaginable. Firstly, a user-based authentication would allow only registered sailors to access the functionality, and could be achieved through different standards, such as *Hypertext Transfer Protocol Secure* (*HTTPS*) client authentication, or *OAuth*[GL]. However, this again requires permanent competitor profiles, which do not exist at the moment. Once authenticated, different measures should be taken to enable sailors to limit the group of other registered sailors who can register for the pre-race they have created. A first step is to limit this to sailors physically close, as identified by the geo-location parameter provided on registration, and possibly also the boat class, and only allow the creator of a pre-race to further control the life-cycle of this pre-race. A further step might allow a pass-phrase to be specified by the creator of a pre-race, that has to be provided by anyone wishing to register for the pre-race. In case the solution is used for high-profile events, the control flow should be reversed, so that the sailors would not register themselves, but be added to the race, or receive an invitation to join the race, such as the presented QR-Code approach. Here the security should also be ensured through other means than relying on the small likelihood of guessing a correct UUID. Apart from this, the security measures should also ensure the integrity, and in some cases also the confidentiality of the transmitted data.

8.7 Transmission of Sensor Data

8.7.1 Base Protocol Choice

Apart from the REST API defined for the management of open races, the requirements necessitate a communication channel between smart-phones and the server for the transmission of sensor data. For sensor data, a minimum latency and overhead is required, so that data transmitted during races with potentially bad reception is as up-to-date as possible. For the transmission of the sensor data, UDP was identified as a suitable transport layer protocol. As UDP is connection-less, it has both advantages and disadvantages when compared to a connection-oriented transport layer protocol such as TCP:

> "[UDP] provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism. The protocol is transaction oriented, and delivery and duplicate protection are not guaranteed. Applications requiring ordered reliable delivery of

streams of data should use TCP."[105]

One of the key requirements for the transmission of sensor data during a race is that it be as light-weight and simple as possible. Even with a low-bandwidth and frequently dis-connected wireless data connection, an update on the current position and other sensor readings should be sent to the server as soon as possible during the race. It is less important, that each and every one of the sensor readings reaches the server, as for example the movement can be interpolated. It is far more important for anybody viewing the race while it is still running via SAP Sailing Analytics that the data used to render the map and leaderboard is as up-to-date as possible, so that the progress viewed on screen coincides as closely as possible to the actual progress that may be observed directly in parallel.

8.7.2 Application Layer Protocol

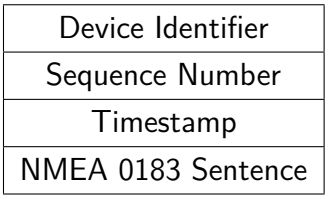


Figure 44: UDP Datagram Data Structure

UDP provides reduced latency when compared to TCP through its minimum protocol mechanism, which matches the functional requirements well. However, the disadvantages of UDP have to be dealt with at application layer. This can be ensured by reusing some of the data that has to be transmitted anyway, and also adding some additional data fields. Figure 44 shows the four fields that have been identified as necessary for the data section in each UDP datagram, which shall be separated by line-breaks.

Transmitting a unique *device identifier* becomes necessary, as other mechanisms of identifying the smart-phone and therefore the competitor sending the data are not resilient enough. Using the IP-address for this purpose is ruled out, as it is prone to change. This may not happen during the race itself, if the mobile data connection ensures a constant IP-address. But wireless local area networks using *Network Address Translation* (*NAT*) make it impossible to identify the specific device merely through its IP-address. If the registration is completed on shore, possibly while connected to a wireless local area network, the connection then changes to a mobile

[105] *Postel*, User Datagram Protocol, 1980.

data network for the duration of the race, and the missing data is later again transferred from within a wireless local area network, the IP address does not remain constant. This unique device identifier is therefore negotiated between smart-phone and server during the registration for the race.

As neither delivery nor duplicate protection are guaranteed by UDP, the *sequence number* is included in every datagram sent by a smart-phone. In combination with the device identifier, this uniquely identifies every datagram. The size of the sequence number is limited to the range $[0, 2^{63} - 1]$, which corresponds to all positive values of the Java primitive *Long* type.

Resource	Method	Details
`/races/<ruuid>/competitors/<cuuid>/missing`	`GET`	Get missing data ranges for the race `<ruuid>` and competitor `<cuuid>` *In* `up_to` *Out* `200 ok` `{missing[{<from>,<to>},...]}` *Errors* `404 race or competitor not found` *Effects* Returns an array with all ranges of missing data packets, identified by their unique consecutive numbers, up to datagram sequence number `up_to`.

Table 5: Querying missing Datagrams in the Smart-phone REST API

During the race, the datagrams are simply sent out, hoping that some or even most reach the server. Whenever the resources are sufficient during or after the race, it is possible to retransmit the datagrams that have not reached the server without transmission errors, which can be verified through the datagram checksum, that also covers the data.[106] A smart-phone can request what datagrams the server has not received (at all, or just not with a valid checksum) through the REST API controller shown in table 5. This returns an array of missing datagram ranges, identified by their sequence numbers. The UDP receiving mechanism is expected to function idempotent, so that duplicates with the same device identifier and sequence number cause no side-effects, but overwrite each other.

The last two data fields in every datagram are the *timestamp* and *NMEA 0183 sentence.* The NMEA 0183 standard defines different sentence types that have a well-defined structure. One of these, for example is the *GGA* sentence type, con-

[106] cf. *Postel*, User Datagram Protocol, 1980.

```
$GP GGA ,113928.67 ,4917.6438 ,N ,00838.6972 ,E , ... ,*23<CR><LF>
  1  2       3          4          5          6
```

Field	Name	Description
1	Talker ID	Identifies the type of device sending the sentence.
2	Sentence Type	Every sentence type has a defined number of fields with defined lengths.
3	Time	UTC timestamp in the format *hhmmss.SS* (h hours, m minutes, s seconds, S milliseconds).
4	Latitude	Latitude in the format *ddmm.mmmm,h* (d degrees, m minutes, h hemisphere).
5	Longitude	Longitude in the format *ddmm.mmmm,h* (d degrees, m minutes, h hemisphere).
6	Checksum	Checksum over the sentence up to this point.

Figure 45: NMEA 0183 Position Sentence[107]

taining the data for a GPS fix. The most important data fields of a typical GGA sentence and their semantics are shown in figure 45. The standard NMEA sentence types suffice for transmitting most data that can be expected from smart-phones. For anything going beyond these sentence types, the concept of a proprietary message exists, which can be used for own sentence types. However, the timestamp contained in most NMEA sentence types is insufficient, as it merely records the time of day. If a smart-phone were to re-send all missing datagrams on a later day than the race, the time point would be unclear. Apart from this, some sentence types do not even include even this kind of rudimentary timestamp. Therefore, the datagram structure includes a fully qualified timestamp, indicating the number of milliseconds that have passed since midnight, January 1, 1970 UTC.

[107] cf. *SiRF Technology*, NMEA Reference Manual, 2005

8.7.3 Evaluation

Just as for the REST API, reviewing this UDP-based protocol reveals that no aspects of security are covered. Integrity, confidentiality and authenticity of the datagrams are not ensured. While confidentiality is less important for the transmission of sensor data, especially the authenticity and integrity will be ensured in future. A lightweight mechanism for doing so might use the registration process to exchange a common secret, so that each datagram could be digitally signed by the client.

8.7.4 Sensor Data Store and Forward

Even though the topic of persistence has not been dealt so far, a store and forward mechanism for the sensor data shall be briefly introduced as a first step in this direction. Different from the current set-up in which the tracking providers deal with the persistence of tracking data, and SAP Sailing Analytics only uses this data in main memory, persistence functionality is needed for the sensor data of smart-phones, as there simply is no intermediary tracking provider. As the raw tracking data is not recoverable except the smart-phones themselves durably store all sensor data, it is important to persist all incoming data as soon as possible, so that it is stored durably and safe in the case of server problems and crashes.

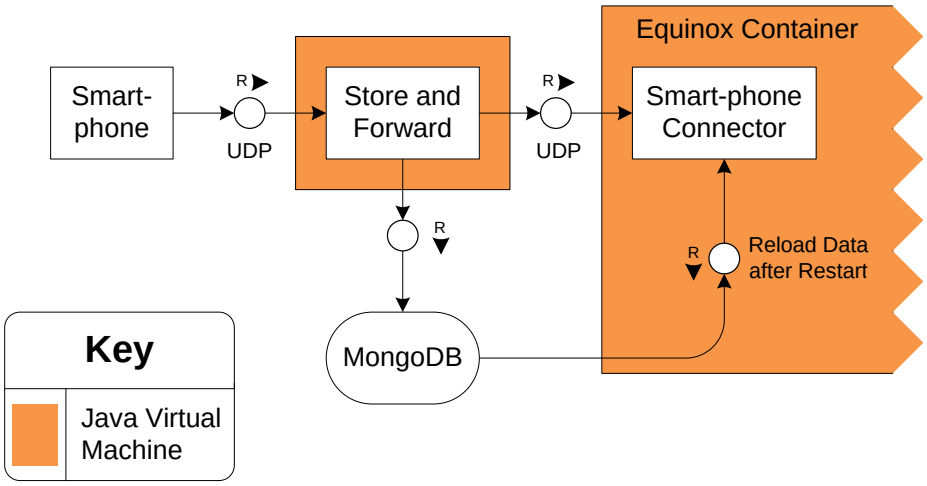


Figure 46: Sensor Data Store and Forward Mechanism

For this purpose, the store and forward mechanism depicted in figure 46 was de-

veloped following an existing concept already existing in SAP Sailing Analytics due to the quirks of a specific tracking provider. It simply stores all incoming UDP datagrams and forwards them to the main application. This small program resides outside of the OSGi infrastructure and runs in its own *Java Virtual Machine* (*JVM*). Even though OSGi is a modular system that decouples the individual components as completely as possible, this additional security measure was taken to make the persistence mechanism as simple and robust as possible. The mechanism is content-agnostic, so that it persists duplicate datagrams multiple times. This increases the complexity of the reload mechanism that should be invoked after a server is restarted, but serves to keep the store and forward mechanism as simple and robust as possible.

8.7.5 Multiplexing Mechanism

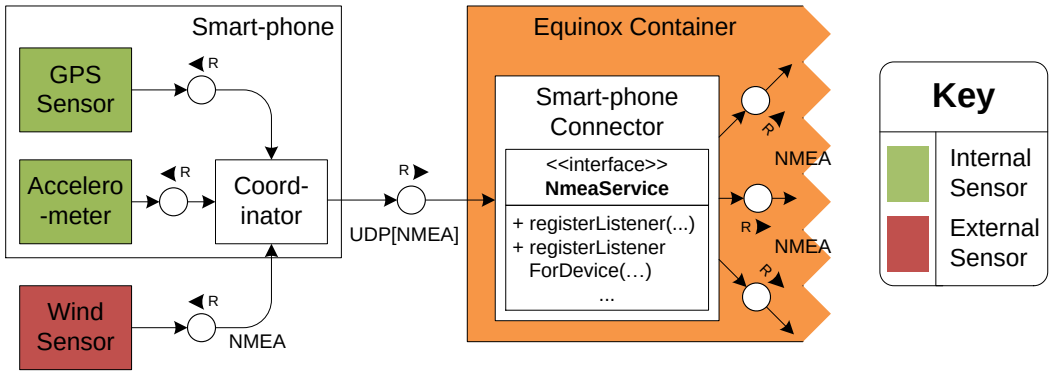


Figure 47: Multiplexing Mechanism

For now, only the transmission of positional data has been dealt with. The potential benefits of additional sensor data, that were highlighted in chapter 4.2 on page 32, were considered in the definition of the sensor data transmission protocol. On server side, the implemented smart-phone connector provides the *NmeaService*, which is displayed in a rudimentary form in figure 47, via the OSGi service registry. Following the observer pattern, any agent on server side wishing to receive sensor data can register a listener implementation through the service, which is notified whenever a new NMEA sentence arrives – or, depending on the mode of registration, only if an NMEA sentence for the specified device identifier arrives.

Figure 47 includes an external wind sensor which may be tethered to the smart-phone via wireless local area network or blue-tooth as an example. Neither the

wind sensor, nor any consumer for wind data on server side need to have an understanding of, or deal with the communication protocol between the smart-phone and the smart-phone connector. A coordinator agent in the app transparently handles the multiplexing of all NMEA data via the custom UDP-based protocol, which is then de-multiplexed by the smart-phone connector, and forwarded to any registered listeners.

The data of the wind sensor may either be linked directly to the smart-phone by using the same device identifier, or could be linked to a different device identifier that is unique for the wind sensor - which would necessitate an addition to the REST API.

8.8 Conclusion

Even though some requirements could not be met in the limited timespan, this chapter has presented all the necessary steps of a first integration of mobile devices into SAP Sailing Analytics. Smart-phones can be used both as a replacement for tracking providers, while retaining the idea of a centralized race definition, as well as for enabling a new type of access to SAP Sailing Analytics, in which a race is open for registering sailors. With the fitting app, this can help expand the usage scenarios of SAP Sailing Analytics, thus providing larger amounts of data, as well as new kinds of data. Also, the demand for importing races through files exported from other mobile devices, such as GPS watches, was met. By enabling manual entry of the course layout, the analytical capabilities of SAP Sailing Analytics can be applied to such an imported or smart-phone tracked race.

9 Conclusion

The results that were presented in this thesis are only the first step to a fully-fledged integration of mobile devices into SAP Sailing Analytics. The definition of communication protocols and the implementation of server side logic has already opened up SAP Sailing Analytics for smart-phone based tracking with a first version of the app, which is still under development. In addition, the import capability for trackfiles meets the requests of sailors to import and analyse personal tracked training sessions or races in retrospect. The same code base was also used to provide an export functionality, so that competitor and buoy tracks can now be downloaded as trackfiles, which helped greatly for the analysis of course layout heuristics in chapter 7 on page 52. The presented approaches for a course layout heuristic are not by any means complete or ready for deployment, but provide a good starting point for further research into this topic, so that hopefully, in the future, the need for manually entering a course layout will cease to exist.

Some requirements with lower priority could not yet be dealt with due to time limitations, and some – such as the security considerations of the smart-phone connectivity – could only be mentioned so far. However, the persistence of races is currently being approached in general, so races tracked by commodity mobile devices will most likely not need a custom persistence functionality. The management of competitors and devices on the other hand still has vast potential for improvement, and it is through this functionality that the desired benefits of making SAP Sailing Analytics available for personal tracking and analysis can be expected to become reality.

A Technical Architecture Modeling Syntax

The following definition of the *Component/Block Diagram* is copied from the TAM reference.[108]

Purpose: The new Component/Block diagram as described in this document is based on UML component diagrams and integrates aspects of the FMC Block diagram. It intends to describe a static structure of a (software) system and, thus, provides a conceptual view on the architecture of this particular system. An extension to the UML meta model is defined within the scope of this standardized modeling in order to formally extend the basic UML component diagrams. Basically, the Component/Block diagram describes a system using active (agents) and passive (storages, channels) visual elements. The connection is established by arcs identifying how agents can access passive elements (read, write, and modify). On conceptual level only the block diagram elements agents, storages, channels, and accesses are used to describe a system. On design level it is possible to, additionally, integrate UML component elements (component, interfaces, and connectors) describing specific regions of the system in more detail. Characteristically, these regions are of central interest for implementation.

Abstraction Levels: Component/Block diagrams are available on conceptual level and design level.

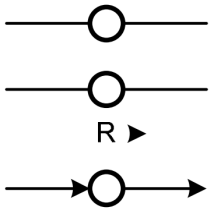
Diagram Elements:

Agent: Active elements, which are capable of doing a certain action. Agents can contain agents, storages, subsystems, components and classes (nesting).

Channels: Passive elements that are used for communication between agents. All transferred information is volatile. Optionally, the request direction can be

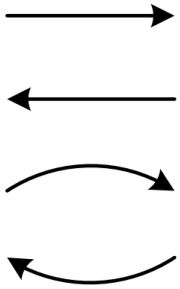
[108] pp. 8 ff *Lünzmann/Pleyer/Schreiter*, Technical Architecture Modeling, 2007.

denoted by adding a "R" with a black triangle pointing in the same direction as the requests are transmitted. It is also possible that requests are sent in both directions. In this case, one "R" and two triangles are drawn. For channels, a graphical simplification similar as for accesses is allowed. Additionally, the dataflow direction can be shown by adding arrow heads pointing into the same direction.



Storage: Passive elements on which agents can act upon. Usually, storages contain data of any kind. Storages can be nested within agents, storages and components/subsystems.

Accesses: Read, Write and Modify. Access arcs define the way, how agents can access a passive element. Accesses are drawn as directed edges. Their direction determines what kind of access is being used. An edge pointing from an active component (e.g. agent, component, subsystem, class) to a storage is a write access. An edge pointing from a storage to an active component is a read access. Two edges pointing in one direction each (usually in shape of a semicircle) is a modifying access.



B GWT Race Data Input Masks

B.1 Competitors and Track Sources

This section describes the process and UI designed and implemented for creating a closed race. This means that all data for the race and competitors is centrally defined by a coach. The tracks for the competitors can either stem from uploaded trackfiles, or be transmitted through smart-phones, that have to be linked to the competitors. The input mask for the basic race data shall not be shown, instead focussing on the process of defining competitors and linking them to track sources. Most of the screen-shots have been cropped and condensed to fit the format of this thesis, which is visually indicated through ragged black lines.

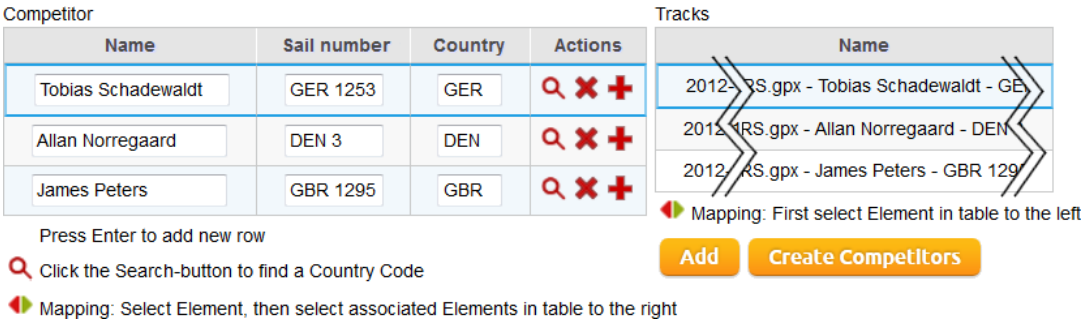


Figure 48: Competitor and Track Sources Screen-shot

The screen-shot displayed in figure 48 shows the two tables in which, on the left hand, the competitor data can be entered, and on the right hand, the track source for a competitor can be selected. This track source can either be a track already present in a trackfile previously uploaded to the server, or a place-holder for the track of a smart-phone.

The mechanism of linking a competitor to a track source graphically involves selection a competitor in the left hand table, and then selecting the track source to be mapped to it in the right hand table. Figure 49 on the next page represents this through a class diagram of the DTOs involved. The abstract *TrackSourceDTO* class

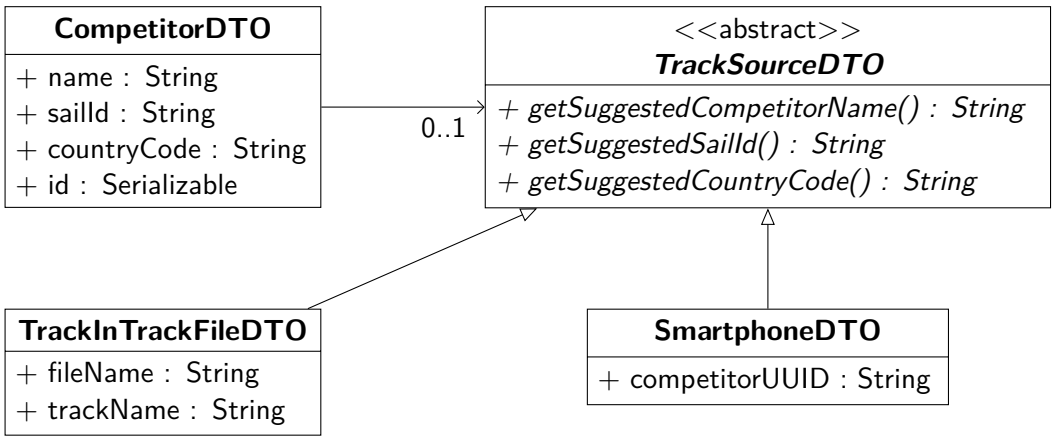


Figure 49: Race Data Entry Class Diagram

extended by the *TrackInTrackFileDTO* and *SmartphoneDTO* classes, which implement polymorphic methods for suggesting the data related to the competitor. The TrackInTrackFileDTO for example tries to infer the sailors name, his sail number and his country from the original file name and the name of the track within the file. Even though a relationship between the CompetitorDTO and TrackSourceDTO is displayed, in reality this relationship is handled by the UI logic through a *Map* data structure, as DTOs, as the mapping is not directly related to the competitor, which can exist independently of it.

B.2 Importing Trackfiles

Trackfiles, which were defined as files in a standard file format containing GPS tracks, may contain one or several tracks per file. If a whole race or training session was tracked with mobile devices such as GPS watches, several raw files most likely exist, that have to be uploaded and then imported into the appropriate race.

Figure 50 on the following page shows the communication between the GWT application, running in the browser, the Equinox container in which the main server side part of the SAP Sailing Analytics runs, and the MongoDB. All communication, but for the very first request from the browser application to the Equinox container, is handled via GWT-RPC. The actual upload of the trackfile has to be performed through a HTTP `POST` request, as web-sites cannot directly access local files, but have to rely on the browser to perform the upload via a POST request.[109] This

[109] The W3C has published a working draft of the *File API*, which will enable web applications also

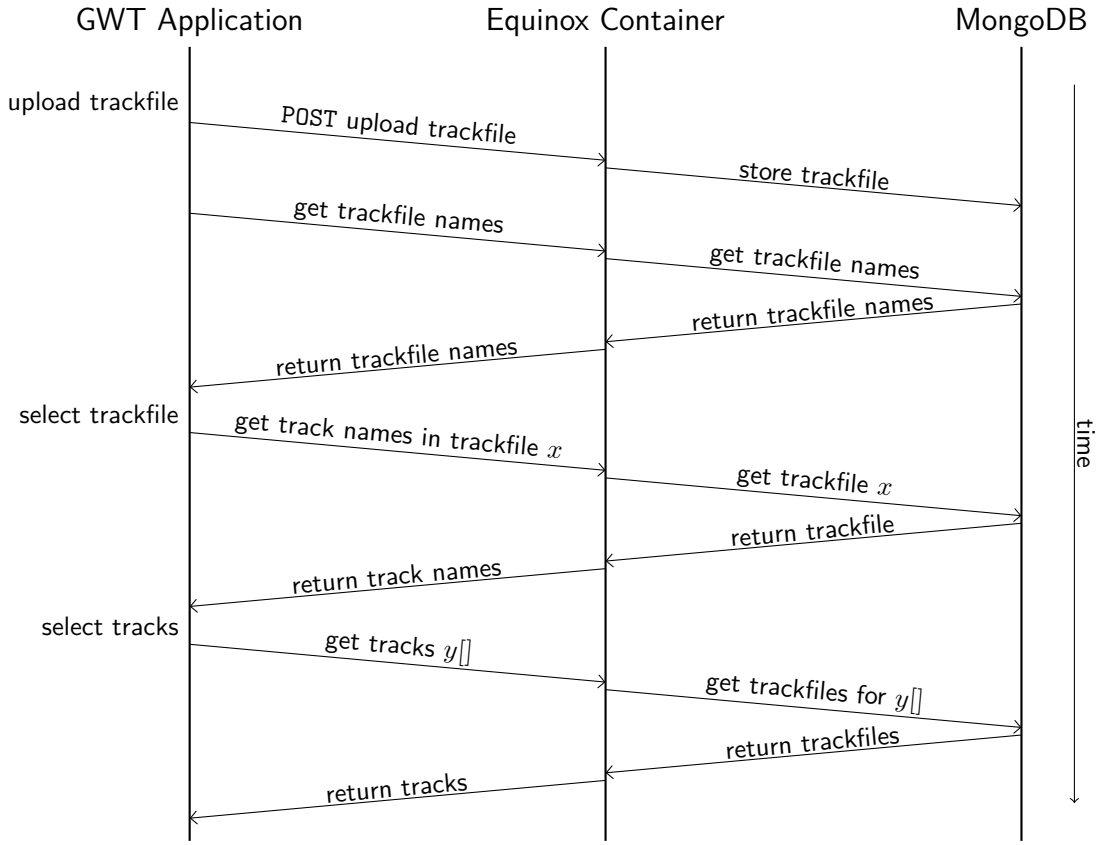


Figure 50: Communication between Components for Trackfile Import

file is then directly stored in the MongoDB GridFS – which allows storing binary data within the MongoDB. By using MongoDB to persist the trackfile on server side, it is automatically replicated, so that all instances of SAP Sailing Analytics have access to it. Therefore, no matter to what instance client requests from the browser application are directed from now on, this instance has access to the uploaded file (as soon as it was replicated). This is important to keep the server side stateless, and not make the application setup dependent on client requests always being dispatched to the same server instance.

51 on the next page shows the UI for uploading trackfiles, and then accessing the tracks contained within these. By pressing the *Upload* button, the communication in figure 50 up to *select trackfile* is triggered. After this, the user can select a trackfile in the left table, which in turn triggers the further communication up to *select tracks*, in which the names of the tracks contained in this trackfile are added to the right

to directly interact with local files in the future, so that a file upload may be fully integrated into the GWT-RPC communication.

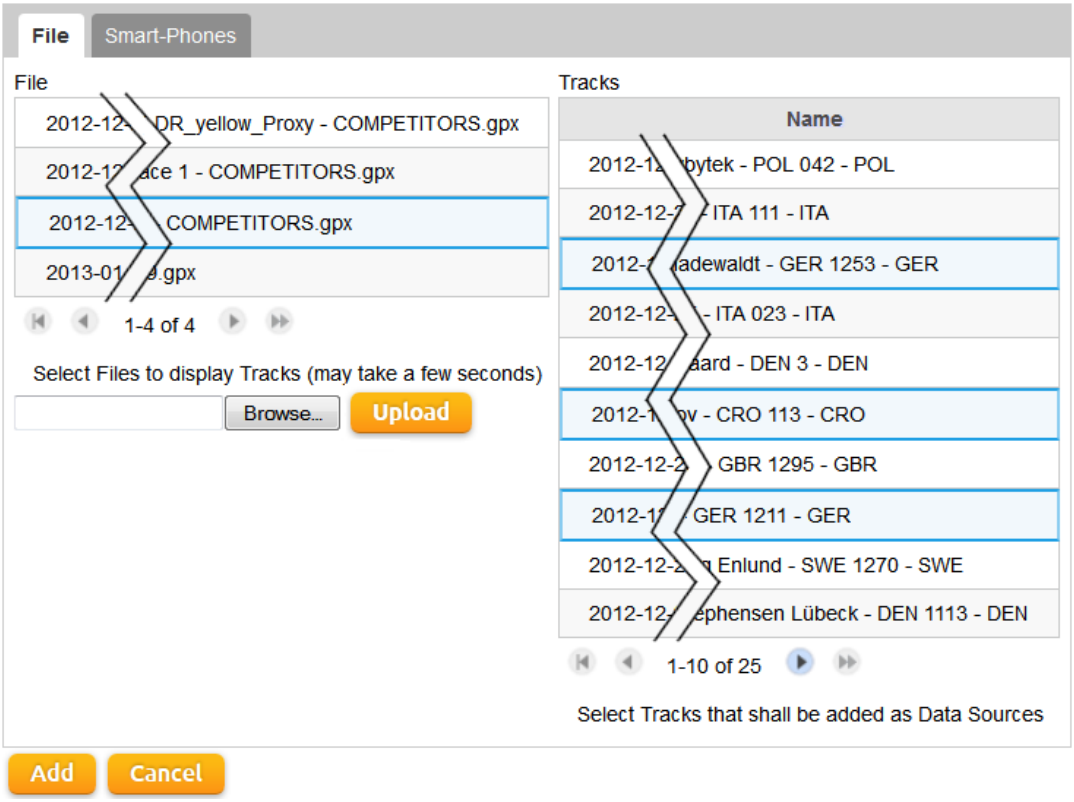


Figure 51: Trackfile Upload and Track Selection Screen-shot

hand table. Finally, by selecting tracks on the right hand side and pressing the *Add* button, these tracks are added to the track sources for the race currently being defined. Also, the actual tracks is requested, depicted through the communication after *select tracks* in figure 50 on the previous page. The server sends an approximate representation of these tracks, generated through the Douglas-Peucker algorithm, to limit the amount of glsgps fixes that have to be transmitted back to the client. These can then be displayed in a map, to assist the user in defining the course layout, which is now presented.

B.3 Entering the Course Layout

After receiving the Douglas-Peucker approximation of the tracks, these are displayed in a map, which can be seen in figure 52 on the following page. The user can then place control points onto the map, which are automatically added to the left table, containing the marks and gates. From this collection of control points, the sequence

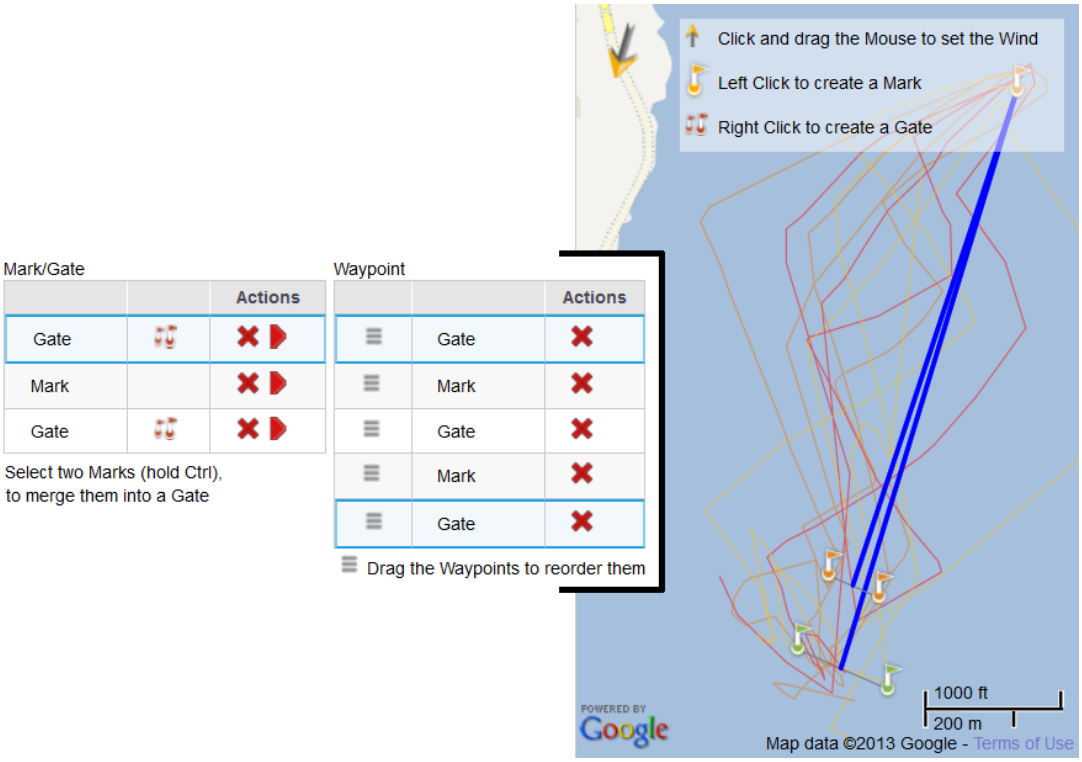


Figure 52: Course Layout Definition Screen-shot

in which they have to be passed can then be defined by adding control points to the right waypoint table. The creation, selection and deletion of the control points and waypoints is synchronized between the two tables and the map through the use of the observer pattern. Each of these three elements observes the events generated by the other elements, and acts on any events that are relevant for the data it contains. For example, by selecting the lower gate in the map, its colour turns green to indicate its selection. At the same time, the according gate in the control point table is selected, as are all occurrences of this control point in the waypoint table.

This input mask for the course layout is not only used to enter the course layout for closed races created with tracking data from trackfiles, but also when entering the course layout after the conclusion of an open or closed race, for which the tracking data was received directly from smart-phones.

C Smart-phone REST API

Resource	Method	Details
`/preraces`	`GET`	List all pre-races
		Out `200 ok` `{pre-races[]}`
	`POST`	Add new pre-race
		In `name, start_time`
		Out `201 created` `{uuid, name, start-time}`
		Errors `400 bad input`
		Effects Adds the pre-race, generating a new `uuid`, only if `name` contains non-white-space characters and `start-time` lies in the future.
`/preraces/<uuid>`	`GET`	Get data for pre-race `<uuid>`
		Out `200 ok` `{uuid, name, start-time}`
		Errors `404 pre-race not found`
	`PUT`	Modify the `start_time` of the existing pre-race `<uuid>`
		In `start_time`
		Out `200 ok` `{uuid, name, start-time}`
		Errors `400 bad input` `404 pre-race not found`
		Effects The start_time is set to the new value, if it lies in the future.
	`DELETE`	Delete pre-race `<uuid>`
		Out `200 ok`
		Errors `404 pre-race not found`

Resource	Method	Details
		Effects If the pre-race `<uuid>` existed and was not yet started, the race and all associated data is deleted.
`/preraces/<uuid>/competitors`	`GET`	List registered competitors for pre-race `<uuid>` *Out* `200 ok` `{competitors[]}` *Errors* `404` *pre-race not found*
	`POST`	Register competitor for pre-race `<uuid>` *In* `name, sail-id, device-id, geo-location` *Out* `201` *created* `{uuid, name, sail-id, device-id, geo-location}` *Errors* `400` *bad input*, `404` *pre-race not found* *Effects* If no competitor with the same device_id was previously registered for pre-race `<uuid>`, and all input data is valid, a registration is created.
`/preraces/<puuid>/competitors/<cuuid>`	`DELETE`	De-register competitor `<cuuid>` from pre-race `<puuid>` *Out* `200` *ok* *Errors* `404` *pre-race or competitor not found* *Effects* If pre-race `<puuid>` contained a registration for competitor `<cuuid>`, this registration is deleted.
`/preraces/<uuid>/start`	`POST`	Start pre-race `<uuid>` *Out* `202` *accepted* *Errors* `404` *pre-race not found* *Effects* Pre-race `<uuid>` is converted into a race with the same `<uuid>`, so that UDP messages for registered competitors can now be processed. This removes it from the pre-races.
`/races/<ruuid>/competitors/<cuuid>/missing`	`GET`	Get missing data ranges for the race `<ruuid>` and competitor `<cuuid>` *In* `up_to`

Resource	Method		Details
		Out	`200` *ok* `{missing[{<from>,<to>},...]}`
		Errors	`404` *race or competitor not found*
		Effects	Returns an array with all ranges of missing data packets, identified by their unique consecutive numbers, up to packet number `up_to`.
	`POST`	Stops the the race `<ruuid>`	
`/races/<ruuid>/stop`		*Out*	`202` *accepted*
		Errors	`404` *race not found*
		Effects	Any incoming data for this race is from now on ignored. All resources acquired for tracking this race are released.

Glossary

Accelerometer

An acceleromenter can determine the physical orientation of a device in space relative to gravity's pull.[110] → pp. 32

Global System for Mobile Communications (GSM)

The Global System for Mobile Communications (GSM) is an open, digital cellular technology used for transmitting mobile voice and data services. GSM supports voice calls and data transfer speeds of up to 9.6 kbps, together with the transmission of SMS (Short Message Service).[111] → pp. 22, 41, 42

International Sailing Federation (ISAF)

The International Sailing Federation (ISAF) is the world governing body for the sport of sailing, officially recognized by the International Olympic Committee. The International Sailing Federation is responsible for: the promotion of the sport internationally; managing sailing at the Olympic Games; developing the Racing Rules of Sailing and regulations for all sailing competitions; the training of judges, umpires and other administrators; the development of the sport around the world; and representing the sailors in all matters concerning the sport.[112] → pp. 11, 13

OAuth

OAuth provides a method for clients to access server resources on behalf of a resource owner (such as a different client or an end-user). It also provides a process for end-users to authorize third-party access to their server resources without sharing their credentials (typically, a username and password pair), using user-agent redirections.[113] → pp. 85

OSGi

The OSGi technology is a set of specifications that define a dynamic component system for Java. These specifications enable a development model where applications are (dynamically) composed of many different (reusable) components. The OSGi specifications enable components to hide their implementations from other

[110] cf. *Komatineni/MacLean*, Pro Android 4, 2012, p. 843 f.
[111] *GSM Association*, GSM, 2012.
[112] *International Sailing Federation*, What is ISAF, 2012d.
[113] *Hammer-Lahav*, OAuth 1.0 Protocol, 2010.

components while communicating through services, which are objects that are specifically shared between components. This surprisingly simple model has far reaching effects for almost any aspect of the software development process.[114] → pp. 24–27, 90, 91

Representational State Transfer (REST)

Roy Fielding, in his Doctor dissertation, coined the term Representational State Transfer (REST), and codified it under four principles:

1. Identification of resources.
2. Manipulation of resources through representations.
3. Self-descriptive messages.
4. Hypermedia as the engine of application state.

These principles describe the architecture of systems and interactions that make up the Web. The building blocks of the Web are called resources. A resource is anything that can be named as a target of hypertext (e.g., a file, a script, a collection of resources). In response to a request for a resource, the client receives a representation of that resource, which may have a different format than the resource owned by the server. Resources are manipulated via messages that have standard meanings; on the Web, these messages are the Hypertext Transfer Protocol methods. The fourth principle means that the state of any client-server interaction is kept in the hypermedia they exchange, i.e., links, or URIs. Any state information is passed between the client and the server in each message, thus keeping them both stateless.[115] → pp. 77, 80, 81, 83–86, 88, 89, 91

Universally Unique Identifier (UUID)

A Universally Unique Identifier (UUID) can be used for multiple purposes, from tagging objects with an extremely short lifetime, to reliably identifying very persistent objects across a network. UUIDs are an octet string of 16 octets (128 bits). The 16 octets can be interpreted as an unsigned integer encoding. It is usual to represent them with hexadecimal digits with a hyphen separating the different fields within the 16-octet UUID. Example: `f81d4fae-7dec-11d0-a765-00a0c91e6bf6` is the hexadecimal notation that denotes the same UUID as `329800735698586629295641978511506172918` in decimal notation.[116] → pp. 79, 80, 86

Velocity Made Good (VMG)

Velocity Made Good (VMG) is the velocity (or speed) along the straight line between starting position and goal. If travelling directly along this line, the boat speed is equal to the Velocity Made Good. The further one deviates from this

[114] *OSGi Alliance*, The OSGi Architecture, 2012b.
[115] *Wilde/Pautasso*, REST, 2011, p. 37.
[116] *International Telecommunications Union*, Universally Unique Identifiers, 2012.

line, the lower the Velocity Made Good gets, until finally reaching zero at a bearing perendicular to the line.[117] → pp. 8, 11, 15, 21, 29, 32, 52, 67

[117] *Köhne/ Wößner*, Navigation mit GPS, 2007.

Bibliography

Abbass, Hussein A./Sarker, Ruhul A./Newton, Charles S. [Data mining - A heuristic approach, 2002]: Data mining: A heuristic approach. Hershey: Idea Group, 2002, ISBN 978–1–930708–25–9

Aitchison, Alastair [Beginning spatial with SQL server, 2009]: Beginning spatial with SQL server 2008. Berkeley: Apress, 2009, ISBN 978–1–4302–1830–2

Baur, Marcus [Der Kreuzdiamant, 2012a]: Der Kreuzdiamant. ⟨URL: `http://www.youtube.com/watch?v=9w8oQl6Xxtc`⟩ – visited on 29.12.2012

Baur, Marcus [Streckbug Segeln, 2012b]: Streckbug Segeln. ⟨URL: `http://www.youtube.com/watch?v=PdJ1N8Kmqc0`⟩ – visited on 06.01.2013

Bethwaite, Julian [49er Polars, 2012]: 49er Polars. Sydney, 2012

Curry, Manfred [Regatta-Segeln, 1949]: Regatta-Segeln: Die Aerodynamik der Segel. 5th edition. Zürich: Schweizer Druck- und Verlagshaus, 1949

Ester, Martin et al. [DBSCAN, 1996]: A density-based algorithm for discovering clusters in large spatial databases with noise. AAAI Press, 1996 ⟨URL: `http://dns2.icar.cnr.it/manco/Teaching/2005/datamining/articoli/KDD-96.final.frame.pdf`⟩, 226–231

Fathi, Alireza/Krumm, John [Detecting road intersections, 2010]: Detecting road intersections from GPS traces. In Proceedings of the 6th international conference on Geographic information science. Berlin and Heidelberg: Springer-Verlag, 2010 ⟨URL: `http://dl.acm.org/citation.cfm?id=1887961.1887966`⟩, ISBN 3–642–15299–6, 56–69

Fielding, Roy et al. [Hypertext Transfer Protocol 1.1, 1999]: RFC2616: Hypertext Transfer Protocol – HTTP/1.1. ⟨URL: `http://tools.ietf.org/html/rfc2616`⟩ – visited on 02.02.2013

Gaber, Mohamed Medhat [Scientific data mining and knowledge discovery, 2010]: Scientific data mining and knowledge discovery: Principles and foundations. Heidelberg: Springer, 2010, ISBN 978–3–642–02788–8

Goodison, Paul [Laser handbook, 2008]: RYA Laser handbook. Southampton: Royal Yachting Association, 2008, ISBN 978–1–905104–65–9

GSM Association [GSM, 2012]: GSM. ⟨URL: `http://www.gsmworld.com/aboutus/gsm-technology/gsm`⟩ – visited on 16.01.2013

Gudgin, Martin et al. [SOAP, 2007]: SOAP Version 1.2 Part 1: Messaging Framework. ⟨URL: `http://www.w3.org/TR/soap12-part1/`⟩ – visited on 05.02.2013

Hammer-Lahav, E. [OAuth 1.0 Protocol, 2010]: RFC5849: The OAuth 1.0 Protocol. ⟨URL: `http://tools.ietf.org/html/rfc5849`⟩ – visited on 02.02.2013

International Olympic Committee [Laser Men Medal Race, 2012]: Sailing Laser Men Medal Race Full Replay: London 2012 Olympic Games. ⟨URL: `http://www.youtube.com/watch?v=MOFWQbZYK7s`⟩ – visited on 13.01.2013

International Sailing Federation [Fleet Racing, 2012a]: Fleet Racing. ⟨URL: `http://www.sailing.org/newtosailing/fleet_racing.php`⟩ – visited on 06.01.2013

International Sailing Federation [Match Racing, 2012b]: Match Racing. ⟨URL: `http://www.sailing.org/newtosailing/match_racing.php`⟩ – visited on 06.01.2013

International Sailing Federation [Racing Rules of Sailing, 2012c]: Racing Rules of Sailing 2013-2016. ⟨URL: `http://www.sailing.org/tools/documents/ISAFRRS20132016Final-[13376].pdf`⟩ – visited on 29.12.2012

International Sailing Federation [What is ISAF, 2012d]: What is ISAF? ⟨URL: `http://www.sailing.org/about/isaf/index.php`⟩ – visited on 31.12.2012

International Telecommunications Union [Universally Unique Identifiers, 2012]: Universally Unique Identifiers. ⟨URL: `http://www.itu.int/ITU-T/asn1/uuid.html`⟩ – visited on 04.02.2013

John Hershberger/Jack Snoeyink [Douglas-Peucker Line-Simplification, 1992]: Speeding Up the Douglas-Peucker Line-Simplification Algorithm. In Proceedings of the 5th Inernational Symposium on Spatial Data Handling.

1992 ⟨URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.17.6932\&rep=rep1\&type=pdf⟩, 134–143

Köhne, Anja/Wößner, Michael [Navigation mit GPS, 2007]: Navigation mit GPS. ⟨URL: http://www.kowoma.de/gps/Navigation.htm⟩ – visited on 29.12.2012

Komatineni, Satya/MacLean, Dave [Pro Android 4, 2012]: Pro Android 4. New York: Apress, 2012, ISBN 978–1–4302–3930–7

Kramer, Oliver [Computational intelligence, 2008]: Computational intelligence: Grundlagen und Konzepte. Berlin: Springer, 2008, ISBN 978–3–540–79738–8

Lünzmann, Martin/Pleyer, Adolf/Schreiter, Torben [Technical Architecture Modeling, 2007]: Standardized Technical Architecture Modeling. ⟨URL: https://wiki.wdf.sap.corp/wiki/download/attachments/859550719/ModelingStandard_V11.pdf?version=1\&modificationDate=1294398200010⟩ – visited on 07.01.2013

Mühlbauer, Hans [Richtig Segeln, 2011]: Richtig Segeln: Ausrüstung, Boote, Technik, Manöver. München: blv, 2011, ISBN 978–3–8354–0762–6

National Marine Electronics Association [NMEA-0183, 2008]: NMEA 0183 Standard. ⟨URL: http://www.nmea.org/content/nmea_standards/nmea_0183_v_410.asp⟩ – visited on 19.01.2013

OSGi Alliance [Benefits of Using OSGi, 2012a]: Benefits of Using OSGi. ⟨URL: http://www.osgi.org/Technology/WhyOSGi⟩ – visited on 07.01.2013

OSGi Alliance [The OSGi Architecture, 2012b]: The OSGi Architecture. ⟨URL: http://www.osgi.org/Technology/WhatIsOSGi⟩ – visited on 16.01.2013

Panda, Sandeep et al. [Comparing Fuzzy-C Means and K-Means, 2012]: Comparing Fuzzy-C Means and K-Means Clustering Techniques: A Comprehensive Study. In *Wyld, David C./Zizka, Jan/Nagamalai, Dhinaharan, editors:* Advances in Computer Science, Engineering & Applications. Volume 166, Advances in Intelligent and Soft Computing, Springer Berlin Heidelberg, 2012 ⟨URL: http://link.springer.com.ezproxy.dhbw-mannheim.de/chapter/10.1007/978-3-642-30157-5_45⟩, ISBN 978–3–642–30156–8, 451–460

Postel, J. [User Datagram Protocol, 1980]: RFC768: User Datagram Protocol. ⟨URL: http://tools.ietf.org/html/rfc768⟩ – visited on 03.02.2013

Ronnewinkel, Christopher [Analysis of GPS Tracks, 2012]: Analysis of GPS Tracks: Diagrams. 2012

Royal Yachting Association [Gate Starts, 2010]: Gate Starts - Best Practice. ⟨URL: http://www.rya.org.uk/SiteCollectionDocuments/Racing/RacingInformation/RaceOfficials/Resource\%20Centre/Best\%20Practice\%20Guidelines\%20Policies/Gate\%20Starts.pdf⟩ – visited on 06.01.2013

Rubio, Daniel [Pro Spring dynamic modules, 2009]: Pro Spring dynamic modules for OSGi service platforms. Berkeley and New York: Apress, 2009, ISBN 978–1–4302–1613–1

Saltonstall, Jim [Race training manual, 1990]: Royal Yachting Association race training manual. 2nd edition. London: Macmillan, 1990, ISBN 978–0–333–54217–0

Sample, John T./Ioup, Elias [Tile-based Geospatial Information Systems, 2010]: Tile-based Geospatial Information Systems: Principles and practices. New York: Springer, 2010, ISBN 978–1–4419–7630–7

SAP Sailing Analytics [505 World Championship, 2012]: 505 World Championship 2012. ⟨URL: http://505worlds2012.sapsailing.com/gwt/RaceBoard.html?leaderboardName=505\%20Worlds\%202012\&raceName=Worlds\%20Race\%203\&root=leaderboardGroupPanelWorlds\%20Race\%203\&regattaName=SAP\%20505\%20World\%20Championship\%202012\%20\%28505\%29\&leaderboardGroupName=505\%20Worlds\%202012⟩ – visited on 07.01.2013

SiRF Technology [NMEA Reference Manual, 2005]: NMEA Reference Manual. 2005 ⟨URL: http://www.sparkfun.com/datasheets/GPS/NMEA\%20Reference\%20Manual1.pdf⟩ – visited on 03.02.2013

Trzeciak, Tomasz M. [Stereographic and cylindrical map projections, 2008]: Example: Stereographic and cylindrical map projections. ⟨URL: http://www.texample.net/tikz/examples/map-projections/⟩ – visited on 16.01.2013

Uhl, Axel [SAP Sailing Analytics Architecture, 2012]: SAP Sailing Analytics: Architecture Overview. 2012

US Sailing [IMS Performance Package, 1997]: IMS Performance Package. ⟨URL: http://desiresailing.org/PolarsC\&C32.pdf⟩

Usama Fayyad/Gregory Piatetsky-shapiro/Padhraic Smyth [Knowledge Discovery in Databases, 1996]: From Data Mining to Knowledge Discovery in Databases. AI Magazine, 17 [1996], 37–54

Wendel, Jan [Integrierte Navigationssysteme, 2007]: Integrierte Navigationssysteme: Sensordatenfusion, GPS und Inertiale Navigation. München: Oldenbourg, 2007, ISBN 978–3–486–58160–7

Whidden, Tom/Levitt, Michael [Das Segel, 1992]: Das Segel: Material, Konstruktion, Aerodynamik, Praxis. Bielefeld: Delius Klasing, 1992, ISBN 978–3–7688–0766–1

Whitley, Darrell [A genetic algorithm tutorial, 1994]: A genetic algorithm tutorial. Statistics and Computing, 4 [1994], Nr. 2, 65–85 ⟨URL: `http://link.springer.com.ezproxy.dhbw-mannheim.de/article/10.1007/BF00175354`⟩, ISSN 0960–3174

Wilde, Erik/Pautasso, Cesare [REST, 2011]: REST: From research to practice. New York: Springer, 2011, ISBN 978–1–4419–8302–2

Statutory Declaration

“This is to solemnly declare:

1. that I have produced this thesis on the topic

   Server Side Integration of Mobile Devices into SAP Sailing Analytics

   by myself;
2. that all ideas taken directly or indirectly from other sources have been marked as such;
3. that this thesis has not yet been shown to any other board of examiners;
4. that the digital version is completely identical to the printed version.

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

Place, Date Signature