TÉCNICO LISBOA

Track It!

Enhancing GPS-Enabled Applications for Outdoor Sports and Activities

Henrique Guimarães Malheiro

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. João Manuel Brisson Lopes

Examination Committee

Chairperson: Prof. Luís Eduardo Teixeira Rodrigues

Supervisor: Prof. João Manuel Brisson Lopes

Member of the Committee: Prof. Manuel João Caneira Monteiro da Fonseca

May 2014

Plans are nothing; planning is everything.

Dwight D. Eisenhower

Acknowledgments

In the first place, I would like to thank Professor João Brisson Lopes for his endless patience and encouragment, and for believing in me since the first moment. Without him this project would hardly become a reality. I also thank him for the fruitful and enthusiastic discussions, and for sharing the same passion for the theme.

I want also to thank my coleagues Marcos Simões for the amazing and insightfull discussions and all the support and encouragment; and Ronaldo Oliveira for his helpful tips and for making everything so much more simple.

I would like to thank all my co-workers who never stopped encouraging me and always believed in my skills.

I heartily thank my friends and family for the words of wisdom, for sharing their experiences, and for never letting me give up.

I am imensily happy and profoundly grateful to have found Ana Luísa, my soul mate. She always makes me dream higher and helps me aim at the intangible. This thesis is undoubtly hers too!
I will always love you. . . and the little ones too!

Abstract

GPS devices are ubiquitous and widely used in outdoor sports and activities. There are also plenty GPS-enabled applications which allow users to get the most out of positioning and fitness data. Nevertheless, a survey on state of the art applications shows that none of the existing applications support all the essential features required for a planning session.

In this thesis we developed a flexible, customisable, and extensible GPS-enabled application for visualising and planning outdoor sports and activities. We also devoted our attention to digital mapping support, including the Portuguese Military Survey Maps, and studied useful algorithms for planning, such as grade calculation and its applications to climb detection and classification, variable pacing or track simplification. The system's architecture was carefully planned and implementation details closely followed accepted standards and well-known design patterns.

The proposed application was subject to usability tests in order to verify its effectiveness. Results have shown that even inexperienced users are able to execute complex planning tasks, provided some help is available. The core algorithms were also evaluated regarding their effectiveness, accuracy and performance. The climb detection and classification algorithms demonstrated to be flexible and accurate, and produce reliable outcomes when compared with official professional cycling data. The track simplification algorithm proved to be effective in preserving track shape and maintaining an higher track point density on climb segments, where details make the difference. This work also gives suggestions for enhancements such as side-by-side activity comparisons, course catalogues with searching capabilities or social support.

Keywords

GPS Tracks, planning, digital mapping, grades, climbs, pacing, track simplification.

Resumo

Os equipamentos com sensores GPS são cada vez mais usados em atividades e desportos de ar-livre. Existe também uma grande variedade de aplicações que permitem aos utilizadores tirarem partido de informações sobre posicionamento e desempenho. No entanto, como revela o estudo sobre o estado da arte, não existe atualmente uma aplicação que satisfaça em simultâneo todos os requisitos necessários ao planeamento de uma atividade.

Neste trabalho desenvolvemos uma aplicação para visualização e planeamento de percursos GPS, simultaneamente flexível e capaz de acomodar novas funcionalidades. A solução proposta permite apresentar mapas digitais na visualização de percursos e inclui suporte para os mapas militares portugueses. Estudámos também algoritmos de planeamento como o cálculo de declives, a deteção e classificação de subidas, o estabelecimento de ritmos de refêrencia ou a simplificação de percursos. A arquitetura do sistema foi cuidadosamente planeada e os detalhes de implementação incorporaram padrões de desenho e metodologias de referência.

A eficácia da solução proposta foi avaliada com recurso a testes de usabilidade e os resultados mostraram que mesmo utilizadores inexperientes conseguem executar complexas tarefas de planeamento. Os algoritmos de deteção e classificação de subidas demostraram ser flexíveis e precisos, para além de produzirem resultados fiáveis quando comparados com dados oficiais de provas de ciclismo. O algoritmo de simplificação de percursos provou ser eficaz, quer na preservação da forma do percurso, quer na manutenção de uma maior densidade de pontos nos troços em subida. Este trabalho apresenta também sugestões para melhorias, incluindo comparação avançada de atividades ou catálogos de percursos.

Palavras Chave

Percursos GPS, planeamento, mapas digitais, declives, subidas, ritmo, simplificação percursos.

Contents

Acknowledgments i

Abstract ii

Resumo iii

Contents iv

List of Figures vi

List of Tables viii

List of Acronyms ix

1 Introduction 1
- 1.1 Motivation . . . 2
- 1.2 Goals and Research Topics . . . 3
- 1.3 Original Contributions . . . 4
- 1.4 Thesis Outline . . . 5

2 Survey on Existing Solutions 6
- 2.1 Introduction . . . 7
- 2.2 Bike Route Toaster . . . 8
  - 2.2.1 Main Features . . . 8
  - 2.2.2 Discussion . . . 9
- 2.3 Map My Ride . . . 10
  - 2.3.1 Main Features . . . 10
  - 2.3.2 Discussion . . . 11
- 2.4 GPSies . . . 11
  - 2.4.1 Main Features . . . 11
  - 2.4.2 Discussion . . . 12
- 2.5 Strava . . . 12
  - 2.5.1 Main Features . . . 12
  - 2.5.2 Discussion . . . 13
- 2.6 Sport Tracks . . . 14
  - 2.6.1 Main Features . . . 14
  - 2.6.2 Discussion . . . 14
- 2.7 Adze . . . 16
  - 2.7.1 Main Features . . . 16
  - 2.7.2 Discussion . . . 16
- 2.8 Survey Summary . . . 17

3 Architecture **20**
3.1 System's Architecture Overview . . . 21
3.2 Domain Model . . . 21
3.3 Application's Layers . . . 25
3.3.1 Presentation . . . 25
3.3.1.1 Views . . . 26
3.3.1.2 Task Manager . . . 27
3.3.1.3 Event Manager . . . 28
3.3.2 Business . . . 28
3.3.2.1 Document Manager . . . 29
3.3.2.2 Operations . . . 29
3.3.3 Data Source . . . 30
3.4 Extension Points . . . 30
3.5 Summary . . . 30

4 Implementation **31**
4.1 Introduction . . . 32
4.2 Overview . . . 32
4.2.1 Development Process . . . 32
4.2.1.1 Technologies and Tools . . . 32
4.2.1.2 Libraries . . . 34
4.2.1.3 Executing the Application . . . 34
4.2.1.4 Licensing . . . 35
4.2.2 Application . . . 35
4.2.3 Document View . . . 37
4.2.4 Summary View . . . 38
4.2.5 Data View . . . 39
4.2.6 Log View . . . 39
4.2.7 Chart View . . . 40
4.2.8 Map View . . . 42
4.2.8.1 Drawing Engine . . . 42
4.2.8.2 Map Providers . . . 44
4.2.8.3 Map Types . . . 45
4.2.8.4 Map Layers . . . 46
4.2.8.5 Map Modes . . . 46
4.2.9 Course Editing Capabilities . . . 46
4.3 Operations . . . 47
4.3.1 Consolidation . . . 47
4.3.2 Grade Calculation . . . 48
4.3.3 Climb and Descent Detection, Classification and Annotation . . . 50
4.3.3.1 Detection . . . 51
4.3.3.2 Classification . . . 53
4.3.3.3 Annotation . . . 55
4.3.3.4 Visualisation . . . 55
4.3.4 Pacing . . . 56
4.3.5 Track Simplification . . . 58
4.3.6 Removing Pauses . . . 60

4.4 Persistence . . . 60
4.5 Summary . . . 61

5 Evaluation **62**
5.1 Introduction . . . 63
5.2 Usability Tests . . . 63
5.2.1 Scenario 1 - Planning an activity based on a past activity . . . 64
5.2.2 Scenario 2 - Planning an activity from scratch . . . 65
5.2.3 Scenario 3 - Planning an activity by combining past activities . . . 65
5.2.4 Usability Tests Results . . . 66
5.3 Algorithms Tests . . . 67
5.3.1 Climb Detection and Classification Algorithms . . . 67
5.3.2 Tour de France 2013 19th Stage . . . 68
5.3.3 Tour de France 2013 20th Stage . . . 70
5.3.4 Liège-Bastogne-Liège 2014 . . . 73
5.3.5 Track Simplification Algorithm . . . 75
5.4 Conclusion . . . 76

6 Conclusions and Future Work **77**
6.1 Conclusions . . . 78
6.2 Suggestions for Future Developments . . . 80

References **81**

A Track It! Domain Model **83**

List of Figures

2.1 Bike Route Toaster user interface during a planning session. 9
2.2 Map My Ride user interface and climb details for a performed activity. 10
2.3 GPSies user interface and the available map providers. 11
2.4 Strava user interface showing performance data for an activity. 13
2.5 Sport tracks user interface. 15
2.6 Adze user interface with track editing in action. 17

3.1 Domain model elements that constitute an activity. 22
3.2 Domain model elements that comprise a course. 23
3.3 The domain model's Waypoint element. 23
3.4 Collection and workspace grouping elements, along with the document element, the application's unit of work. Documents are composed of activities, courses and waypoints. . . 24
3.5 The track point. 24
3.6 The course point. 24
3.7 The waypoint. 24
3.8 System's three layer architecture (presentation, business and data source) and their main components. 26

4.1 GNU General Public Licence (v3.0) header. 35
4.2 Track It! main window displaying a commute bike ride, with the following active views: document, summary, map and chart. 36
4.3 Entities that implement the observer pattern in order to keep the application's views synchronized. 36
4.4 Hierarchical view of an activity. 38
4.5 Hierarchical view of a course. 38
4.6 Summary view for the selected activity. 39
4.7 Data view for the selected activity. 40
4.8 Log view displaying an excerpt of a log in trace level. 40
4.9 Chart view for an activity displaying the elevation profile, speed and heart rate, in distance mode. 41
4.10 Segment zooming in chart view. 41
4.11 Snapshot of the map view. 42
4.12 Military maps in action. 45
4.13 Consolidation operation dialog. 48
4.14 Serra da Estrela climb profile (Portugal), from Seia to Torre. 51
4.15 Tour de France climb ratings for climbs included in the 1999–2007 editions. 54
4.16 Annotation dialog. 55
4.17 Document, summary, map and chart views of the selected segment. 56

4.18 Dialogue for setting the pace of a course. 58
4.19 Chart view with the results of applying the smart pacing method to a course. 58
4.20 Track simplification dialogue. 60

5.1 Track It! elevation profile for the 19th stage of the 100th Tour de France, with climbs identified and classified. 69
5.2 Stage 19 elevation profile and categorised climbs from 100th Tour de France official road book. 69
5.3 Track It! elevation profile for the 20th stage of the 100th Tour de France, with categorised climbs information. 71
5.4 Stage 20 elevation profile and categorised climbs from the 100th Tour de France road book. 71
5.5 Liège-Bastogne-Liège elevation profile and categorised climbs for the 2014 edition. 73
5.6 Effects on track shape from applying the simplification algorithm three times in succession. The original track is presented in blue while the simplified track is depicted in yellow. 75

A.1 Track It! main domain model elements, with their attributes and most relevant associations. 84

List of Tables

2.1 Summary of the surveyed solutions, categorized according to functionality, target platform and availability. 18

4.1 Summary of the libraries used in the development of the Track It! application. 34
4.2 Summary of supported events. 37
4.3 Map provider interface methods, grouped by functionality. 43

5.1 Summary of the results obtained with the usability tests. Tests were conducted with five users. Tasks were evaluated with the following scale: 1. the user completed the task autonomously, without interacting with the evaluator; 2. the user completed the task while asking some questions to the evaluator; 3. the user completed the task with the help of the evaluator; 4. the user could not complete the task, even with the help of the evaluator. 67
5.2 Parameters for the climb and descent detection algorithm. 68
5.3 Summary of the results obtained by comparing the official Tour de France 2013 information for stage 19 categorised climbs with the climbs detected and classified by Track It! 70
5.4 Summary of the results obtained by comparing the official Tour de France 2013 information for stage 20 categorised climbs with the climbs detected and classified by Track It! 72
5.5 Summary of the results obtained by comparing the information from the Liège-Bastogne-Liège official website with the climbs detected and classified by Track It! 74
5.6 Effects on track point density and distance as a function of the number track points and segment type. Density is expressed in track points per meter. 76

List of Acronyms

GPS Global Positioning System

API Application Programming Interface

1

Introduction

Contents

1.1 Motivation 2
1.2 Goals and Research Topics 3
1.3 Original Contributions 4
1.4 Thesis Outline 5

1.1 Motivation

In recent years a number of GPS-enabled applications for outdoor sports and activities have emerged, becoming popular among athletes, coaches and outdoor activities enthusiasts [1, 2]. Two main reasons have contributed to this sudden interest in this type of applications.

On the one hand, the availability to the general public of affordable, sturdy, precise, and full-fledged GPS devices made possible the collection of large quantities of raw data and opened up a whole new range of possibilities, but demanded, at the same time, the existence of additional tools to extract meaning from that data. The gathered information covers not only basic GPS data like location, time and elevation, but also calculated data like speed, distance, elevation gain, grade, just to name a few. Moreover, the development of third party devices [3, 4, 5] and the increasing interoperability [6] have made it possible to capture additional and extremely useful measurements on the same device, namely heart rate, cadence, power or temperature.

Over the last couple of years, dedicated GPS devices are slowly stepping aside to give place to mobile phones and smartphones, equipped with GPS chips that made the use of GPS data even more ubiquitous. Besides, these devices rely on other features that give them an edge over dedicated devices, in particular processing power, internet connection, assisted GPS, bluetooth connectivity, accelerometers, integrated cameras, as well as support for commercial and community applications, which opens up the path for improvements, opportunities and innovation.

On the other hand, developments on digital maps were remarkable in the past decade and, more importantly, they have been made public and accessible for everyone to explore and interact. Maps exist in different flavours, from road maps to raster maps; from satellite imagery, birds eye view [7] and street view [8] to digital elevation maps with shaded relief; from commercial products to community driven efforts like OpenStreetMaps[1] or TopoLusitania[2]. Maps are enhanced with routing, traffic and weather information and encompass a huge collection of points of interest, properly categorised. Maps allow the integration of user data and provide, most of the time, features customization. Maps have become an essential tool to visualize collected GPS data in context, analyse performance, and plan future activities.

Driven by those two trends, in particular GPS devices and digital maps, GPS-enabled applications have evolved from platform-specific desktop applications to web applications and, more recently to mobile applications. They started to be distributed, often freely, by vendors whenever a customer bought a GPS device, but soon several commercial solutions and community projects began to emerge.

Amongst the most popular features of GPS-enabled applications for outdoor sports and activities are 1) storing, searching and visualizing activities; 2) analysing performance data; 3) planning workouts and activities; 4) sharing activities with others, including coaches; 5) sharing achievements and results on social networks like Facebook[3] or Twitter[4]; 6) playing games like Geocaching[5]; 7) competing in virtual communities for a standing in a particular workout or activity.

[1] Open Street Maps (www.openstreetmaps.org)
[2] Topolusitania (topolusitania.blogspot.com)
[3] Facebook (www.facebook.com)
[4] Twitter (www.twitter.com)
[5] Geocaching (www.geocaching.com)

Activity planning assumes particular relevance nowadays. By planning we mean all the tasks involved in defining a route, calculating summary data, setting a pace, and annotating the track with all the relevant information for performing the activity.

Firstly, it is important to get the most out of our busy schedules. To achieve this we must be able to carefully plan an activity based on the available time, the past performance and the activity profile and features. Secondly, with planned activities we have the possibility to explore new places while avoiding being lost. Thirdly, having the relevant information at hand while performing an activity contributes to a safer experience, to more accurate estimates of the remaining time and/or distance, and to anticipate obstacles or sections of interest. Lastly, planned activities provide extremely useful information about our probable whereabouts which not only plays a role in our security as well as gives our family and friends a piece of mind.

Despite the large number of GPS-enabled applications available on the market and increased sophistication, there are still several issues and gaps to bridge on the topic of planning.

On the one hand, the available software generally lack flexibility as it is often platform-dependent, require a working internet connection, and seldom provide extensibility mechanisms to enhance current features and/or add new features. Besides, this kind of applications often rely, at most, in one or two of the most popular digital map providers, loosing the ability to combine the strengths and information of different sources. Additionally, there is not, to our knowledge, a GPS planning application capable of displaying the Portuguese Military Maps, available as large image tiles and containing a rich set of information not present elsewhere.

On the other hand, some of the algorithms which are successfully applied to visualisation and performance analysis could be very useful to assist the planning task. Grade calculation, for example, may be of great value to detect and classify climbs and descents and annotate activities with detailed information about them. Grade data may also be used to establish a variable pace which improves estimates on the activity time, not only while planning but also during the activity, besides providing useful information to real time assessment of performance. Track simplification is another key feature to reduce the required resources and increase the performance of GPS devices while ensuring the information of interest is delivered.

1.2 Goals and Research Topics

Driven by the motivations just depicted, we aim to achieve the following goals:

1. Develop a flexible, customizable and extensible GPS-enabled application for visualizing and planning outdoor sports and activities;

2. Give support to the most popular mapping providers available online and seamlessly integrate the Portuguese Military Maps;

3. Study useful algorithms for planning activities, namely grade calculation and its applications to annotating climbs and variable pacing, as well as track simplification.

Developing a GPS-enabled application for visualizing and planning activities which is simultaneously flexible, customizable and extensible is in itself quite a demanding task. Several issues have to be addressed namely 1) choosing an adequate data model; 2) carefully designing an architecture which allows multiple synchronized views over the same data, as well as the addition of new modules; 3) including import and export capabilities for the most popular data formats and allowing the addition of new formats later on; 4) supporting several non-functional requirements like platform independence, off-line operation, localization and preferences .

Supporting several of the most popular map providers and map types available online without the ability to use the readily available API's (only accessible to web applications, which are not an option in face of the above requirements) requires a framework for managing and displaying geospatial information, namely geographic coordinates, map tiles and geographic features like tracks or waypoints. Additionally, it is necessary to interact with tile servers to obtain the needed tiles and stitch them together in a comprehensive map.

Integrating with the Portuguese Military Maps imposes additional challenges, such as transforming coordinates, designing custom tiling schemes and dealing with tiles of considerable size which easily drains the system resources.

Grade calculation is clearly affected by the quality of the collected data, in particular location and elevation accuracy, as well as the sampling rate of the collected data. These calculations also depend on the parameters used like the considered segment length. It is our purpose to study the factors that affect grade calculation and devise a method to deal with multiple data sources of different accuracies. Given the calculated grades, we intend to study ways to successfully apply that information to climbs detection and classification and setting a variable pace to the activity. Finally, we will devote some time thinking of an algorithm to simplify a track which gracefully degrades its quality while preserving the overall shape as the number of points is reduced.

1.3 Original Contributions

In spite of the relevance of all the proposed goals and research topics described in detail in the previous section, we would like to emphasise, from where we stand, the three main contributions of the current work.

1. Support for the Portuguese Military Maps, in particular the proposed solution to overcome the system resources limitation based on a secondary, on-demand tiling scheme; and its seamless integration with the popular digital maps available online;

2. Use of grade calculation to detect and classify climbs and descents, and the employment of that information to annotate activities and provide useful information while performing an activity;

3. Use of grade calculation to set a variable pace which improves estimates over the duration of an activity and gives hints about performance, not only during the planning stage but also while performing the activity.

1.4 Thesis Outline

This dissertation is composed of six chapters that share a common structure. Each chapter begins with a general introduction, then discusses in detail the subject matter, and concludes with a summary of the presented topics.

Chapter 2 presents a survey and critical analysis of the existing solutions, taking into account the research topics defined in Chapter 1.

Chapter 3 outlines the proposed architecture, emphasising its flexibility and extensibility. It also discusses the chosen data model.

Chapter 4 gives an in depth overview about the application's design and implementation details. It devotes special attention to digital mapping support and algorithms, carefully defining the challenges we faced, describing how we tackled them, and sharing their shortcomings.

Chapter 5 evaluates the proposed solution, providing a detailed description of the test scenarios, presenting the achieved results and discussing its limitations.

Finally, chapter 6 summarises the contributions and shortcomings of the dissertation, and suggests directions for future work.

2

Survey on Existing Solutions

Contents

2.1 Introduction 7
2.2 Bike Route Toaster 8
2.3 Map My Ride 10
2.4 GPSies 11
2.5 Strava 12
2.6 Sport Tracks 14
2.7 Adze 16
2.8 Survey Summary 17

2.1 Introduction

In this chapter we analyse some of the existing solutions with focus on the research questions we have identified in Chapter 1. It is our purpose to summarise what can and cannot be done with existing software and identify its limitations, so we can contextualize current work and demonstrate its relevance.

There are currently several GPS-enabled applications available for visualizing, planning and analysing activities so it was not feasible nor desirable to go over each one of them. Instead, we have chosen to select a few representative applications and for those present a critical analysis. At the end of the chapter, we provide a comparative overview of the examined solutions.

The first goal was then to define the selection criteria for the target applications of our analysis. We decided to choose software that matched all of the following criteria:

Functionality Software aiming at searching, visualising, analysing, editing and planning activities; applications whose primary focus is exclusively on mapping and/or real-time tracking were excluded; mobile and tablet applications were also not considered because they lack, at the present moment, advanced capabilities for editing and planning activities due to limited processing power and reduced screen size, although they are extensively used to collect, visualise and navigate activities; in fact, none of the mobile applications we tried offer support for editing tracks, like Ski Tracks[1], Satski[2], GPSTrack[3], Runtastic[4] or Sports Tracker[5], just to name a few;

Availability Software available to the general public as freeware, open-source, shareware or commercially at affordable prices (less than 50€);

Popularity Software with most hits on current search engines, most discussed in the community (e.g., forums, blogs) and with the largest platform of users;

Relevance Software that stands out, either positively or negatively, in one or some of the features we have selected to study.

Additionally, we tried to cover the main platforms used, namely Windows, MacOSX and Linux, and we also considered multi-platform, web and command-line applications. The selected software should then illustrate the commonly provided features and typical limitations of the set of applications they belong.

The next step was to define the criteria to analyse the selected solutions. Taking in consideration the research questions defined for the present study we have identified six categories: 1) Editing Capabilities (importing and exporting tracks; adding and removing track points; joining and splitting tracks), 2) Supported Map Providers, 3) Grade Calculation, 4) Climb Detection and Classification, 5) Pacing, and 6) Track Simplification.

With respect to editing capabilities we looked for the ability to create activities from scratch or based on performed activities; to join and split activities; to add, move, edit and remove track points, course

[1] Ski Tracks (https://play.google.com/store/apps/details?id=com.corecoders.skitracks)
[2] Satski (https://play.google.com/store/apps/details?id=com.satski.ski)
[3] GPSTrack (https://itunes.apple.com/app/gpstrack/id378708797?mt=8)
[4] Runtastic (https://play.google.com/store/apps/details?id=com.runtastic.android.pro2)
[5] Sports Tracker (https://itunes.apple.com/pt/app/sports-tracker/id426684873)

points and waypoints; to fix timestamps and remove gaps in order to correct inconsistencies from activities, specially after performing joining and splitting operations.

Regarding digital maps, we explored the supported providers and map types as well as the ability to add new providers or custom maps.

Considering grade calculation, we verified the possibility for visualising average grades for single points, track segments or the entire activity, as well as minimum and maximum grades. We also looked for different ways of visualising grades, incorrect grade values, and the ability to fetch and/or correct elevation data.

Related to climb detection and classification, we examined the possibility to automatically identify climbs and multi-part climbs, to present summary info about the climb and to automatically calculate the climb difficulty or determine its classification. We also took into consideration the ability to customize the results.

Concerning pace we examined the capability to set the pace of an activity and the provided methods.

Finally, we reviewed the applications' ability to simplify an activity in order to improve GPS performance without compromising the detail, accuracy or usefulness of the available information.

Before directing our attention to the survey, it is important to define some of the key concepts used on the evaluation. We differentiate between activities and courses, the former representing the log of a performed activity, usually including complete and detailed data like heart rate, cadence, power or temperature; and the latter intended for navigation purposes and to provide useful information while performing an activity.

Is is also worthy to distinguish between three types of points or locations. Track points are the locations or breadcrumbs that constitute the GPS track of a course or an activity. They comprise latitude, longitude, altitude, as well as additional information collected at that location like time, distance, speed, heart rate, cadence, power, or temperature.

Unlike track points, course points only apply to courses. They are always paired with a track point by timestamp and represent an important information at that location like a feed zone or a categorised climb. Course points are added to the course during the planning stage and are usually signalled by the GPS device when that particular location is reached.

Finally, waypoints are points of interest or reference points. They are not tied to a particular course or activity and are very convenient for navigation purposes.

2.2 Bike Route Toaster

Bike Route Toaster [9] is one of the most popular web applications aimed at mapping and planning activities. It is simple, intuitive and has some interesting features.

2.2.1 Main Features

Bike Route Toaster is exclusively devoted to mapping and planning activities. It supports Google Maps and Open Street Maps as map providers, allows the creation of courses from scratch with a few

unique features, and provides import and export operations. Additionally, is is possible to add course points, visualise a static elevation profile and export a cue sheet to print on paper. Figure 2.1 presents Bike Route Toaster user interface during a planning session for a bike ride.

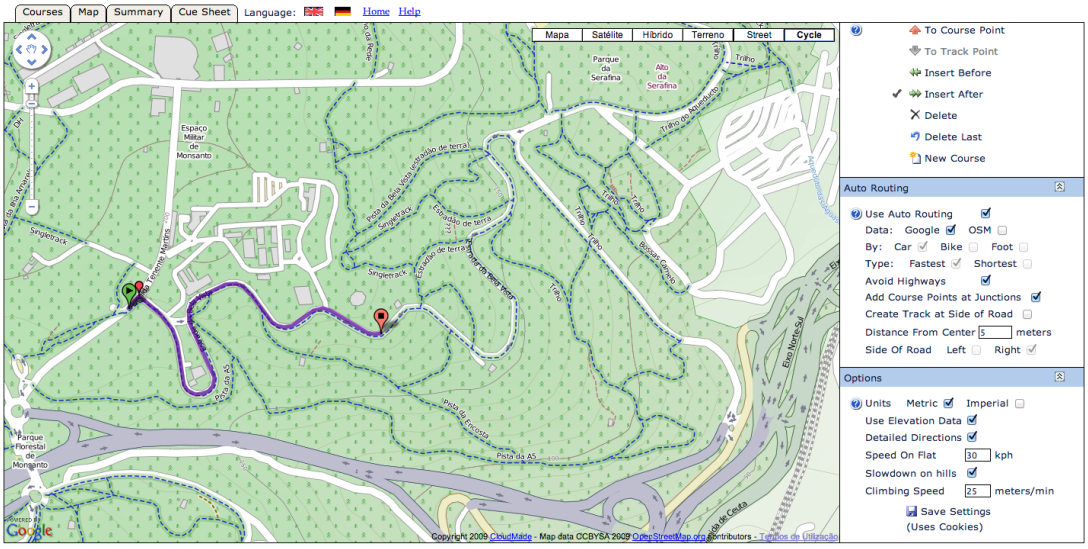

Figure 2.1: Bike Route Toaster user interface during a planning session.

2.2.2 Discussion

Bike Route Toaster is a popular mapping application because it is simple and fast, has an intuitive interface, and allows for great flexibility. It also has features closely related to those we are studying.

Firstly it has the ability to mark directions based on underlying map routing algorithms, and allows the user to choose to export the gathered information. It also provides means of manually inserting custom points of several types (e.g., climbs, danger locations, feed zones). Directions can be customised to include a previous warning point at a specified offset distance of the target point, and tracks may be drawn at a specified distance from the centre of the road, which makes it easier later to mark points on superimposed segments. Nevertheless, marking directions only work when mapping an activity using maps with routing support and do not suite off-road activities. Moreover, directions are only added during mapping and cannot be inserted automatically for loaded activities. Additionally, only crossings and junctions that are taken into account by the routing algorithm are marked on the activity, thus crossings or junctions of the track with itself, as well as loops, will not be considered.

Secondly, this web application also has the ability to set a constant or variable pace based on the predefined settings. In particular, it is possible to set a speed for flat terrain, add the option to slow down on hills, and set a climb speed to apply on hills. This three options combined can result in a variable pace which will be included in the exported data and help the user to try to match that pace. Despite the fact that this application already has a very basic way to add a variable pace based on the elevation profile, it is not very customisable, the climbing speed setting in meters per second is cumbersome, and does not take into account the past history of the user performances, requiring some trial and error to reach a reasonable pace.

Finally, we would like to emphasise that although it is possible to acquire elevation data from online services and that a static elevation profile is generated for each mapped activity, there is no way to add details about climbs and descents neither automatically nor manually.

2.3 Map My Ride

Map My Ride [10] is an all-purpose web application that covers a wide range of functionality to assist outdoor sports and activities.

2.3.1 Main Features

The Map My Ride feature set is huge, including collecting, visualising, analysing, planning, searching, exploring, sharing, and competing virtually. Despite the wide range of functionality, we will take a close look at its ability to detect and classify climbs. Figure 2.2 depicts Map My Ride user interface with climb details of a performed activity.

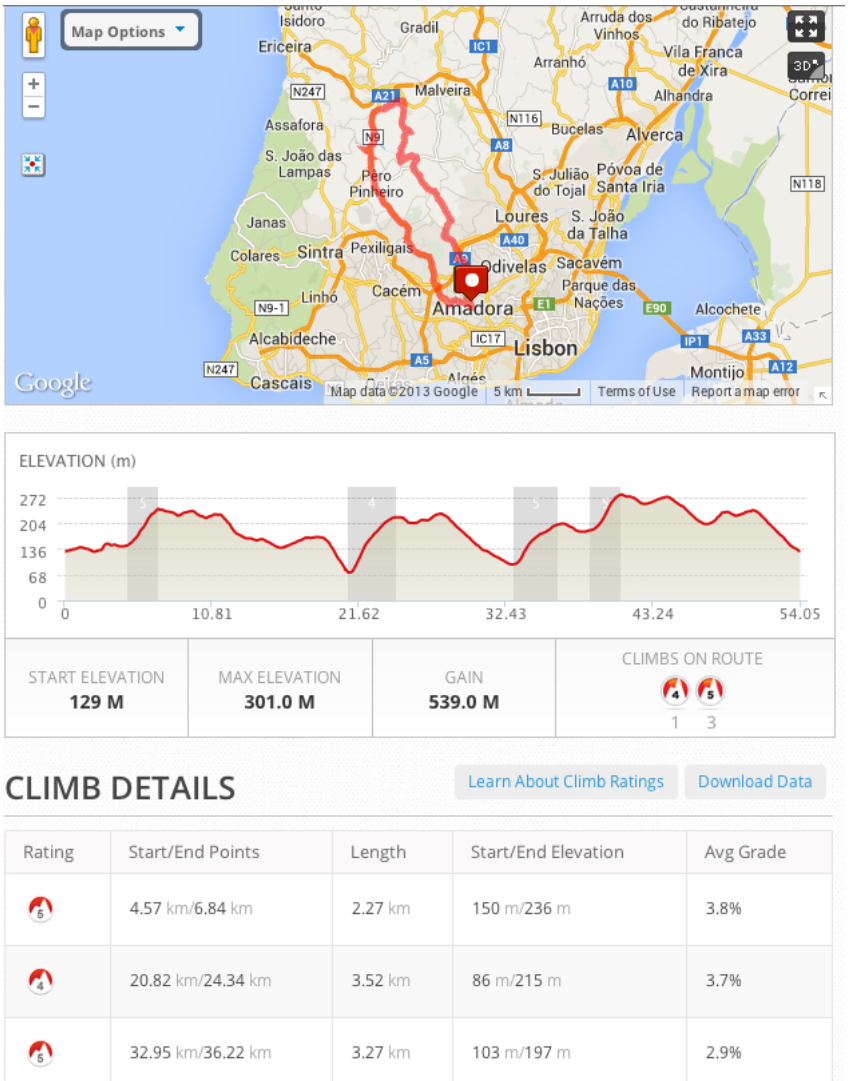

Figure 2.2: Map My Ride user interface and climb details for a performed activity.

2.3.2 Discussion

As stated in the previous section, we selected Map My Ride mainly because of its ability to automatically identify and classify climb sections within an activity.

Contrary to other applications, the employed criteria to classify and identify climbs is clearly defined, including segment length, average grade and maximum grade. Nevertheless, it is not possible to customise, fine-tune or manually tweak the ways in which climbs are detected and classified. In particular, multi-part climbs are not considered.

On the other hand, as claimed on the website, the application's goal is to achieve the classification used in professional cycling, which not always suit leisurely activities or amateur athletes.

As a final note, despite its planning capabilities, the application does not allow to export climb information as an aid while performing an activity.

2.4 GPSies

GPSies[11] is a web application whose primary focus is to manage a community driven database of activities. It allows its users to search and share activities and offers some mapping capabilities.

2.4.1 Main Features

In order to support its core business and maintain a collection of community contributed activities, GPSies provides operations for uploading, searching, exploring and exporting activities. It also allows users to create new courses from scratch; supports multiple map providers such as MaqQuest, Open Street Maps or Google Maps; and includes a set of small utilities to convert between file formats, and search for locations. Figure 2.3 presents GPSies user interface and the available map providers.

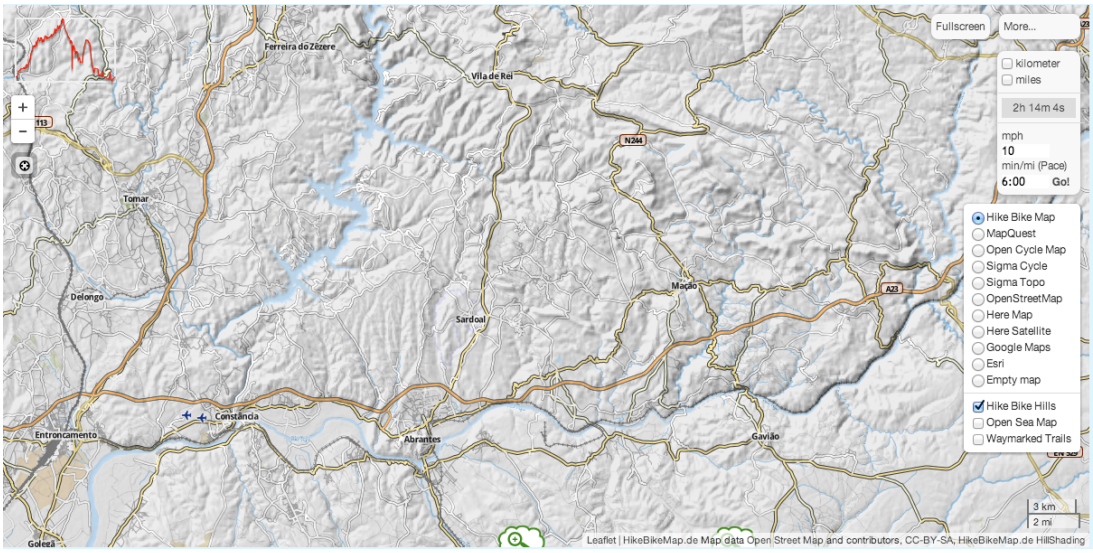

Figure 2.3: GPSies user interface and the available map providers.

2.4.2 Discussion

We considered including GPSies in our analysis for four main reasons. In the first place it supports multiple map providers beyond the commonly used Google Maps, Bing Maps or Open Street Maps. In particular, it includes maps with biking and hiking trails, custom overlays with hill shading and even an empty map that removes all the clutter and allows to concentrate on a given track shape and features. Despite the wide range of alternatives we are limited to the supported providers. Besides, being a web application, offline operation is also not supported.

The second reason we considered GPSies is its ability to set the pace of an activity during the export operation. It uses a simple pacing strategy allowing the user to set an average speed (km/h) or average pace (min/km). The downside of this approach is that a constant pace is not very helpful during the ride because it does not contemplate variations in rhythm due to the terrain elevation profile. Strangely enough, the user is limited to set a constant pace as the original timing information is removed!

The third reason for choosing GPSies relates to track simplification. In fact, it is possible to specify the maximum number of points included in an exported course in order to accommodate the limitations of some devices. On the other hand, is is also possible to simplify the exported track to maximize the device performance while rendering the track. Currently GPSies supports the Douglas Peucker [12] algorithm with three levels of simplification: low, medium and high. Although useful, we cannot control in which ways simplification affects track characteristics like the elevation profile or the maximum distance between track points.

Finally, we would like to emphasise the possibility to automatically include course points for directions based on bearing. Additionally, the user can specify the maximum number of markers as well as the distance the marker shall appear before the turn. Although a variation of 90 degrees on bearing is often a good predictor of a turn, slight variations on bearing are not necessarily turns but frequently simple bends on the road or trail. For that reason, this approach is not very useful as it generates a lot of clutter and many false positives.

2.5 Strava

Strava [13] is one of the most promising fitness applications we have reviewed and has been gaining a great momentum over the past couple of years. Its motto is “social fitness” which means users can share, compare and compete against other’s fitness data, putting workouts and races into context. It has been a pioneer in the virtual competition although some of the competing applications are now following this trend. Strava is available both as a web application and a mobile application, and features a free version as well as premium membership.

2.5.1 Main Features

Strava allows the user to keep track of his activities and visualize detailed data of his performances both on maps and on charts. It also provides advanced tools to analyse performance, specially power and heart rate related, and compare achievements against past activities or other users activities. In

particular, users may virtually compare their performances and compete with other users on the same courses. Figure 2.4 presents Strava user interface along with performance data for an activity.

The presented features revolve around the concept of segment, in other words a portion of a GPS track of interest along with performance data. Strava maintains a set of segments, either manually created or automatically identified, which are then matched against all stored activities. This way, the application can present leader boards, compare users performances on segments, and allow searching and exploring segments. Besides manual segmentation, Strava also identifies climb sections as segments, gives them a category and calculates average grade.

At the time of this writing Strava is releasing a route planner which allows the user to plan the course of a ride. Currently Google Maps are the only supported map provider and the planner features automatic routing and manual tracks. Although this planner is in its early stages, the most distinctive feature is undoubtedly the ability to colour segments on the map according to its competitive status or popularity.

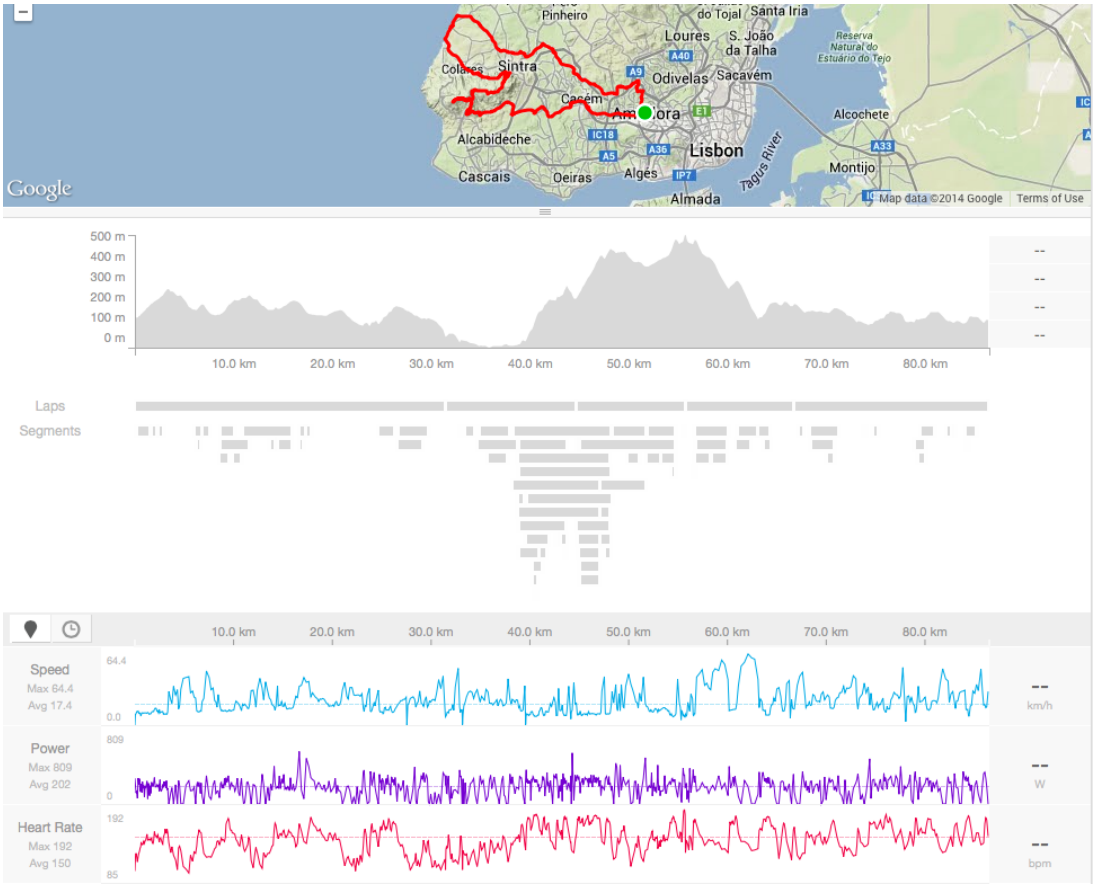

Figure 2.4: Strava user interface showing performance data for an activity.

2.5.2 Discussion

Strava's primary focus is on performance analysis and not on activity planning. For that reason, several of the proposed features are somehow tackled by Strava but with a different purpose in mind.

Firstly, this web application only offers a basic route planner, featuring Google Maps support and allowing for automatic or manual routing. Editing or combining routes is not supported, nor it is possible to use alternative map providers. In fact, the only distinctive feature that sets this planner apart is the ability to take advantage of the large data set uploaded by its users, providing valuable hints about segments of interest during the planing stage. It is worthy to note that this feature is not limited to popular segments as the user also has access to its own heat map, the most used segments on his activities. Nevertheless, the provided information is static and the planner clearly lacks edition capabilities which would allow the user to combine and split segments interactively.

In most competitions climbs receive special attention as they differentiate athletes performance. Not surprisingly, climb segments acquire particular relevance in Strava's analysis. Climbs can be identified manually or automatically based on grade, and are usually described by their length and average grade. Because the aim is at performance comparison, climbs classification criteria is clearly defined, not allowing users to fine-tune or customize climb detection and classification to multi-part climbs or to suite a particular fitness level. Moreover, some vital information is lacking like maximum grade or grade at specified intervals.

One of the most prominent shortcomings of Strava is clearly its inability to export data. Either because the application's core business relies on exploring data uploaded by its users, or due to privacy concerns, Strava is very reluctant in making data available for download. Only premium users are allowed to export GPS tracks in GPX format, stripped of all timing and performance data. Exported data is then very limited for navigation purposes and useless to monitor performance and follow a reference pace while performing an activity.

2.6 Sport Tracks

Sport Tracks[14] is a desktop application that runs on Windows systems. It allows users to keep track of activities and gear, and offers a rich set of visualisation tools.

2.6.1 Main Features

Sport Tracks is a software dedicated at collecting, storing, searching and visualising activities. It also allows to monitor gear usage and provides mapping support for planning new routes to follow on a GPS device. One of its main strengths relies on the possibility of extension through plugins developed by the community. At the time of this work there are several plugins available for download, most of them for free. Figure 2.5 presents Sport Tracks user interface.

2.6.2 Discussion

Our intention to include Sport Tracks in this survey is twofold.

On the one hand, this application was designed for extension, allowing for the addition of new functionalities and modules through the development of plugins. This way, it was ensured the application stays aligned with the community needs, greatly benefiting from the contribution of its members at the

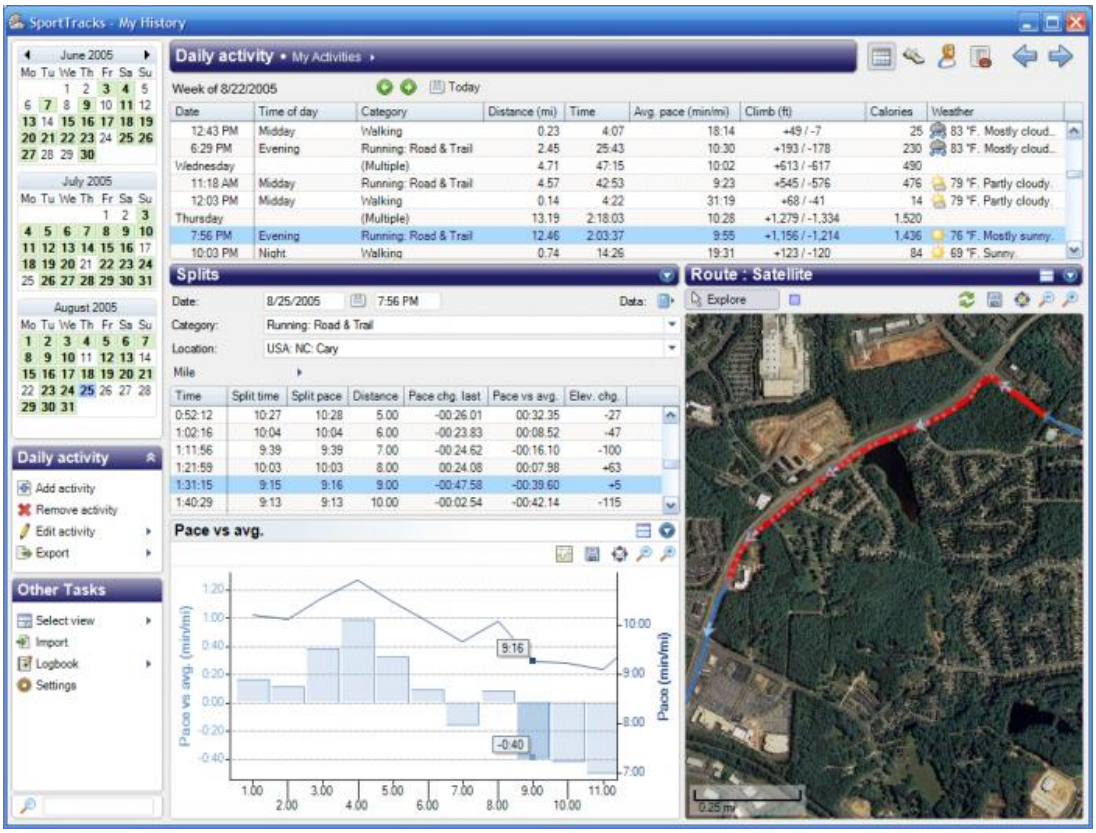

Figure 2.5: Sport tracks user interface.

same time. According to Sport Tracks developers, plugins may be designed to target two broad areas: 1) device and data integration; and 2) features customization. In particular, it is possible to support new GPS devices, sensors and file formats as they are introduced, customise existing functionalities or add entirely new features, and devise views and actions specially suited to the task at hand.

From where we stand, Sport Tracks is a reference application with respect to outdoor sports and activities, specially because of its extensibility features. Nonetheless, the fact that it is limited to run on the Windows platform constitutes its main limitation, leaving aside Linux and MacOSX users. Actually, these extensibility concerns overlap with some of the defined goals for this thesis and were accounted for on our proposal, as we shall see in Chapter 3. Moreover, it was also our motivation to design a multi-platform application in order to serve a wider range of users and overcome Sport Tracks limitation.

On the other hand, Sport Tracks features some interesting plugins with respect to the proposed goals. We shall highlight the most relevant to the present study.

“Course Score” plugin categorises activities according to its difficulty and presents information about the available climbs and difficulty, average grade, as well as the longest, steepest and most difficult climbs in the course. It is also possible to compare the performance on those hills with previous efforts. Once more, the downside is that it is not possible to include the visualised information on future courses to display in GPS devices as valuable hints. Furthermore, it is not clear how the difficulty indexes are calculated, which parameters are used on that calculation, nor the scale they comply. Still, it is missing more detailed information about the climbs like maximum grade or the relative difficulty of each of its

segments.

“Apply constant pace” and ”Pace Rechner“ plugins allow the user to set the pace for an activity, which can be used in the field for monitoring purposes. Nevertheless, the former plugin only allows to set a constant pace based in the average speed defined by the user, which as we saw earlier it is not very useful during the activity. The latter plugin additionally allows to define a target time or the finish time as references to set the pace, a goal which can be easily unrealistic. None of them provides a way to set the pace based on the elevation profile nor establish variable pace as we advocate.

Finally, the “Elevation Correction” plugin acquires elevations from online data sources like the Shuttle Radar Topography Mission (“SRTM”) or the U.S. Geological Survey (“USGS”) to replace the collected elevation data. However, this sources are not available for every location on earth at the appropriate resolution and suffer from the problems we described earlier with online elevation services. Besides, it is not possible to apply smoothing operations to the collected data without actually replacing it.

2.7 Adze

Adze[15] is a lightweight GPS data editor for the MacOSX operating system. Only a small set of features may be downloaded for free. Access to the full-featured application requires the payment of 20 USD.

2.7.1 Main Features

As announced in the application’s website, Adze focus is on GPS data manipulation. The features may be categorised in: 1) track points edition, including add, remove and move operations; 2) track segments edition, namely selecting, merging, splitting and deleting operations; 3) track simplification; 4) stop detection; and 5) timestamps manipulation. Additionally, the application is able to display summary information about the route and allows the user to edit track metadata. Figure 2.6 shows Adze user interface with track editing in action.

2.7.2 Discussion

Track editing is one of the key features when planning an activity and for that reason we have proposed to study ways to manipulate GPS data. There are not many options to consider when searching for a lightweight, useful and multi-purpose application to manipulate tracks. The reason to include Adze in the current survey is that it covers the fundamental track manipulation operations and has some unique features relevant to the present work.

Adze provides several tools to select, add, remove and move track points, as well as to split and merge tracks. Nonetheless, there are some issues we would like to emphasise, most of them user-interface related. Selections are hard to identify because their colours are bright and different from the rest of the track; tools are difficult to use and master as they frequently require pressing control keys which are not self-evident nor standard; the join operation does not let the user choose the joining order explicitly, which frequently leads to unexpected results.

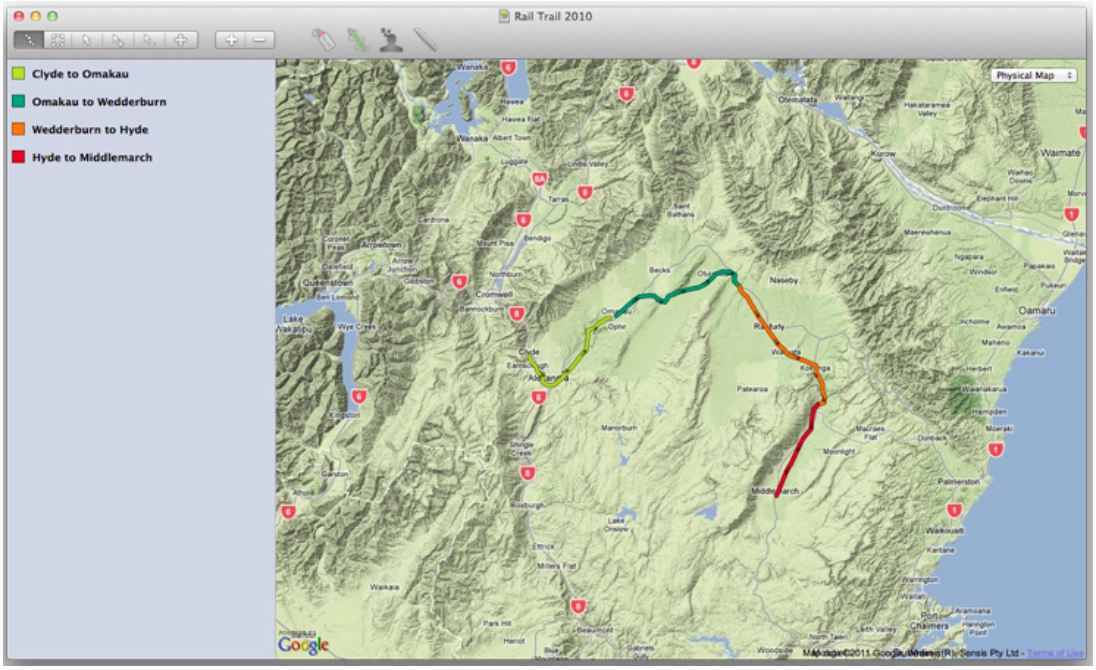

Figure 2.6: Adze user interface with track editing in action.

Timestamps manipulation is also an essential feature when editing a track, specially when we are setting a pace or combining GPS data from different sources or collected at different moments in time. Adze allows the user to clear and sort timestamps, add an offset or overwrite timestamps altogether based on a constant speed. This later feature may alternatively be seen as setting the pace for the activity, but it is also the only strategy available to set the pace.

Adze allows stops from more than one minute to be detected and removed, but for this feature to be useful smaller pauses should be considered. In fact, pauses of a few seconds are certainly more difficult to detect as they depend on the sampling rate and the accuracy of the GPS data.

Finally, being only available to the MacOSX operating system may be seen as an important limitation of this application.

2.8 Survey Summary

In this chapter we have reviewed some of the available solutions with focus on the research questions we have defined in Chapter 1. In particular, we considered in this survey six categories of problems: 1) track editing capabilities, 2) map providers, 3) grade calculation, 4) climb detection and classification, 5) pacing, and 6) track simplification. Target applications were selected based on functionality, availability, popularity, relevance and supported platforms. Table 2.1 summarizes the surveyed software.

Planning activities require most of the time creating routes from scratch or manipulating existent tracks. In this survey we found that there is good support for mapping tracks either using the routing algorithms supplied by the map providers or drawn point by point. There also some useful options to assist these tasks like elevation fetching or signalling the most popular or competitive track segments. Nevertheless, advanced track manipulation, including combining existing tracks, is seldom supported

Table 2.1: Summary of the surveyed solutions, categorized according to functionality, target platform and availability.

	Functionality[a]	Target Platform	Availability
Google Earth	1), 2), 3)	Desktop application: MS Windows, MacOSX, Linux	Freeware
Garmin Basecamp	1), 2), 6)	Desktop application: MS Windows, MacOSX	Freeware
Garmin Connect	1), 2), 5)	Web application	Freeware
Bike Route Toaster	1), 2), 5)	Web application	Freeware
Map My Ride	1), 2), 3), 4)	Web application	Freeware and Commercial (premium members)
GPSies	1), 2), 3), 5), 6)	Web application	Freeware
Strava	1), 2), 3), 4), 5)	Web application	Freeware and Commercial (premium members)
Sport Tracks	1), 2), 3), 4), 5), 6)	Desktop application: MS Windows	Freeware and Commercial
Adze	1), 2), 3)	Desktop application: MacOSX	Commercial
GPS Babel	5), 6)	Desktop and console application: MS Windows, MacOSX	Freeware

[a] 1) Editing and planning activities; 2) Digital maps support; 3) Grade calculation; 4) Climb detection and classification; 5) Pacing; 6) Track simplification.

and the available applications are frequently targeted at a specific operating system.

We also found that only a few route planning applications offer the possibility to mark tracks with relevant information to be used while performing the activity, and most of the time course points or waypoints must be manually added by the user. The exception is marking turns when using the provided routing algorithms, despite the fact that this only useful for mapping activities that follow roads. Inferring turns from bearing often result in inaccurate information.

Google Maps, Bing Maps or Open Street Maps are the most common choices when it comes to digital mapping support. Only a few applications support more than two map providers and even less add extra functionality such as hill shading or 3D views. There is some support to proprietary maps like Garmin's maps, and from the reviewed solutions only Google Maps support custom maps, which have to be calibrated and for that reason are not easily available to the general public. None of the surveyed applications allows seamless integration with the Portuguese military maps while planning an activity.

Grade calculation and climb detection and classification are frequently supported for visualising or comparing performed activities. Nevertheless, grade calculation is often affected by inaccurate GPS data resulting in unrealistic information. On the other hand, the user cannot customise climb detection and classification to fit his level and special cases are not considered like multi-part climbs. None of the

surveyed applications allow to automatically export relevant information about climbs to a GPS device.

Setting a custom pace based on a specified average speed is often supported by route planning applications. None of the reviewed applications allow to easily set a variable pace based on the terrain profile or on the user's past activities. Adjusting the pace by percentage or setting time or duration goals are also non-existent.

Track simplification is generally available as a means to overcome some GPS devices limitations on the number of track points that may be displayed. It is often not clear or self-evident the criteria used to simplify tracks.

Having reviewed state of the art software and identified its strengths and weaknesses, we will seek to incorporate this useful information while designing and developing a solution to the proposed goals.

3

Architecture

Contents

3.1 System's Architecture Overview 21
3.2 Domain Model 21
3.3 Application's Layers 25
3.4 Extension Points 30
3.5 Summary 30

3.1 System's Architecture Overview

In this chapter we describe the application's architecture, present its modules and discuss some of the design decisions.

It was our purpose to design an application for visualising, editing and planning GPS-enabled outdoor activities. Simultaneously, the application should enable us to study and present solutions to the research questions stated in Chapter 1. In particular, the system's architecture should provide the foundation for the following requisites: 1) multiple map providers support, including custom or off-line maps like the Portuguese Military Survey Maps; 2) grade calculation and automatic climb detection and classification; 3) activity pacing using multiple strategies, including variable pace based on the elevation profile and the user's past history; 4) track simplification, preserving properties other than distance or shape; 5) track editing capabilities, including joining and splitting tracks; adding, removing, and moving track points.

Additionally, the application should support some non-functional requisites, such as: 1) flexibility, in order to provide the user with different perspectives over the same information and allow him to perform different operations over the same activity; moreover, the user must be able to run the application in multiple operating systems and in off-line mode; 2) customisation, enabling the user to adapt the application's look and behaviour to his needs through preferences pages, as well as to personalise the performed operations; furthermore, a multilingual interface must be supplied to accommodate a wider range of users; 3) extensibility, allowing developers to enhance functionality and performance through extension mechanisms over multiple application's components, namely domain model, views, operations and file types.

In the following sections we tackle the system's components in greater detail.

3.2 Domain Model

The domain model assumes a key role when designing an architecture as it determines the available information to work with, as well as the operations that may performed over that information. Thus, our main concern was to design a domain model that is both flexible to allow viewing, editing and planning activities; and extensible to accommodate additional features. In the following paragraphs we will describe the domain model's most important facets. For the complete reference to the major domain model elements see Appendix A.

Based on existing file formats, in particular GPX[1], TCX[2] and FIT[3], we have identified three core elements to our model: activity, course and waypoint.

Activities allow us to represent a performed activity, thereby they contain general information like the activity's name or its starting time, aggregate data such as the duration or distance, and detailed information collected at each of the locations that constitute the breadcrumb of the activity, including positioning data (longitude, latitude, altitude), timing information (timestamp), performance data (e.g.,

[1]GPX file format (http://www.topografix.com/GPX/1/1/gpx.xsd)
[2]TCX file format (http://developer.garmin.com/schemas/tcx/v2/)
[3]FIT file format (http://www.thisisant.com)

heart rate, cadence, power) and environment measures (temperature). Activities are static and may only be viewed and analysed, although corrections may be applied to the data and missing summary data may be calculated.

Activities are composed of several elements, including the activity element itself, as depicted in Figure 3.1. Activities are thereby organised as follows: 1) the activity element comprises a name, the start time and notes; 2) the associated session element contains aggregate information about the whole activity; we have considered the session element in order to latter support activities that span over multiple days, each day consisting in an individual session within the activity; 3) each session is divided in laps, which represent a certain duration of the activity along with summary data for that duration; laps may be manually created by the user or automatically added by the device; 4) each lap may have one or more tracks to group the collected data, for example to separate track segments between pauses; 5) finally, each track consists of a list of track points which actually store the collected information at every location determined by the sampling rate and/or sampling strategy; 6) events associated with an activity identify start and finishing times, automatically detected pauses, route resuming after an off-course section, among other device-generated events; 7) device elements represent the gadgets which where connected to the GPS-device during the activity gathering information such as heart rate straps, footpods, speed/cadence sensors or power meters; 8) segments consist of user or application defined lists of track points that represent a particular section in the activity's route, along with the summary data for that section; segments are particularly useful to represent sections of interest like climbs, descents or dirt trails, for example; moreover, segments stand for distance as laps stand for duration.

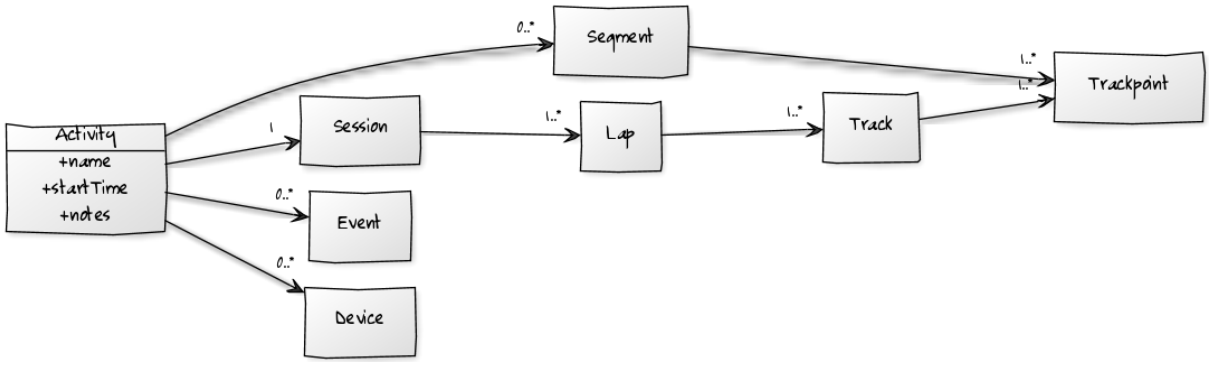

Figure 3.1: Domain model elements that constitute an activity.

Courses express a planned activity to be loaded in a GPS device in order to provide useful information while performing that activity. Additionally, a course allows the user to edit the route of the activity, define course points with key locations like feed zones or summits, set a reference pace or simplify the track.

Like activities, courses are composed of several elements, including the course element itself, as showed in Figure 3.2. Courses share many elements from the activity's structure, although there are some interesting differences. Courses are then arranged in the following manner: 1) the course element contains the name of the course as well as all the summary information about that course; this is an important difference to the activity's structure, which contains summary data in the session element; in

fact, the course duration may be known in advance and planned accordingly, contrary to activities which may span over multiple days, with the user disconnecting the device in between, and the session item being important to separate logical sessions within the activity; 2) laps represent spans of time within the course and incorporate the matching aggregate information for that duration like distance or average speed; 3) laps are subdivided into tracks, which group related track points like sections between pauses; 4) the course's track points represent the reference information at each of the locations that constitute the route of the course; 5) the segment element represents a section of interest within the course and comprises the list of associated track points along with summary data; 6) finally, a course may contain course points, which provide additional information when following the course outdoors like feed zones, and which are usually signalled by GPS devices when that particular location is reached.

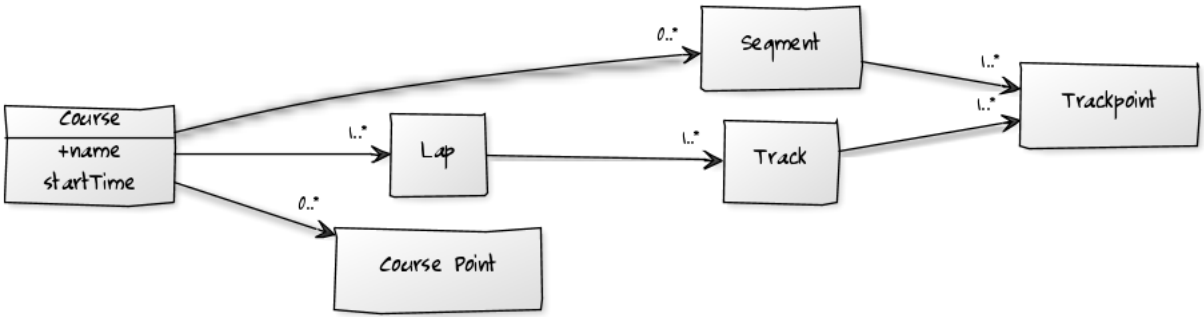

Figure 3.2: Domain model elements that comprise a course.

Finally, waypoints are not tied to any particular activity or course but may be added as reference points to aid navigation and provide useful information about landmarks, emergency services or sites of interest. Hence, waypoints are valuable at the planning stage and effective at aiding navigation while outdoors. Waypoints contain positioning information, name, type and description, as depicted in Figure 3.3.

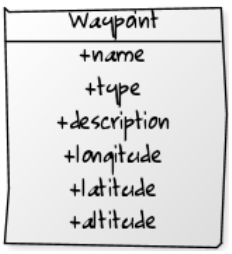

Figure 3.3: The domain model's Waypoint element.

Planning activities often involve 1) loading several performed activities; 2) splitting, joining and editing tracks; 3) creating courses based on the edited tracks; 4) annotating courses with course points, setting the reference pace and simplifying the track; 5) adding waypoints as reference locations. To make it easier to the user to perform these different tasks, we have created the concept of document: the unit of work that manages the required activities, courses and waypoints associated with a single planning session (see Figure 3.4).

Furthermore, we have gathered documents in two main groups according to their purpose: 1) the collection, a persistent storage for the user's documents; and 2) the workspace, a staging area for loading and viewing documents on the fly, as well as for planning activities. We think that the workspace's transient quality is advantageous because it allows the user to work on or explore activities without the need to persist them or affect the stored collection. Workspace and collection elements are outlined in Figure 3.4.

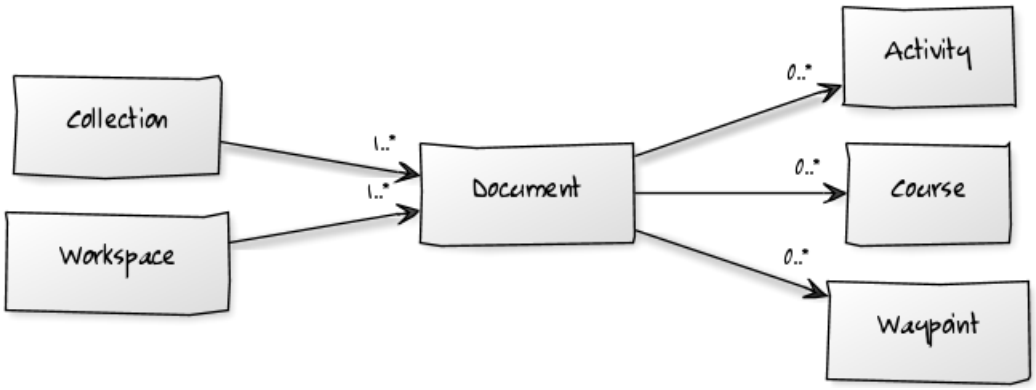

Figure 3.4: Collection and workspace grouping elements, along with the document element, the application's unit of work. Documents are composed of activities, courses and waypoints.

We would also like to emphasize the three types of points that constitute the building blocks of any activity, course or waypoint:

- track points, the breadcrumbs that make up the track of a course or an activity, containing positioning, performance and environment data (see Figure 3.5 for details);
- course points, representing special locations within a course like feed zones or categorised climbs (see Figure 3.6 for details);
- waypoints, reference points or locations of interest (see Figure 3.7 for details).

Figure 3.5: The track point.

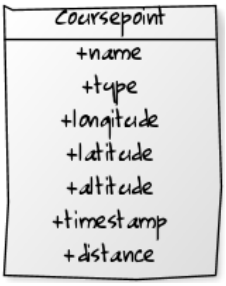

Figure 3.6: The course point.

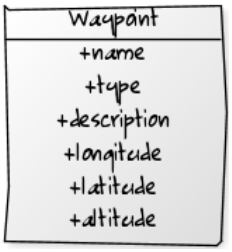

Figure 3.7: The waypoint.

In order to make the model more flexible and customisable, we have defined metadata for each of the attributes that constitute a domain model's element: 1) the full qualified name; 2) the data type; 3) the

unit category (e.g., length); 4) the unit (e.g., meter); 5) the groups the attribute belong to (e.g., timing, elevation); and finally 6) the localised name for the attribute. This way, new attributes may be added to the model and the application will be able to accommodate them in its views and operations, besides displaying proper labels and consistently formatted data.

3.3 Application's Layers

By using the layering technique when designing the system's architecture we can reduce complexity and achieve better code organisation, as layers logically separate software components and provide isolation and independence between them. Moreover, layers allow us to substitute different parts of the system with alternative implementations, as well as run system's components in different platforms, thus contributing to increased flexibility and extensibility [16].

Taking into consideration both the prevailing best practices and the application's requisites, we have chosen to divide the application in three layers:

1. The **presentation layer** is responsible for displaying information to the user and dealing with user interaction; user's requests to perform operations are dispatched to the business layer;
2. The **business layer** deals with the application's business logic; it essentially performs operations over the domain model;
3. The **data source layer** is responsible for permanently storing the user's data in a database, while allowing the application to communicate with other systems and/or service providers.

The three layers and their main components are depicted in Figure 3.8. In the following sections we are going to explore these layers in greater detail.

3.3.1 Presentation

In our design, the presentation layer's responsibilities are twofold: 1) to present information to the user, providing different perspectives over the same data; 2) to allow the user to interact with the application, forwarding user requests to the business layer for processing and presenting the results.

Despite the fact that it is good practice to separate the user's interface logic from the rest of the application, one of the main reasons that led us to consider using this layer was to allow the user interface to be extended in the future to support different platforms. In fact, we have stated in the goals section that it was our purpose to design a multi-platform desktop application, with a rich client interface capable of assisting the user in complex tasks, and with off-line support. In order to meet those requirements, he had to leave aside prevailing platforms like the web platform, which requires a working internet connection; the mobile platform, which offers a display too small to support complex planning operations and has limited processing power; or the tablet platform, which is inherently vendor specific and doesn't allow us to run an application on different devices without considerable effort. Nevertheless, by including the presentation layer in our design, we ensured it will be easier in the future to provide a subset of the application's functionalities in any of the excluded platforms.

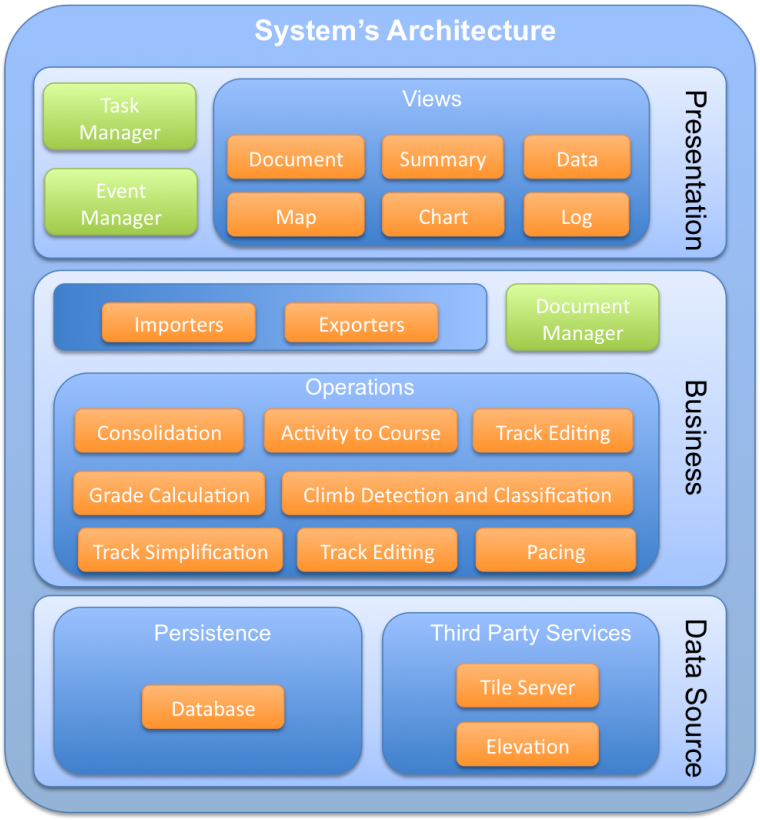

Figure 3.8: System's three layer architecture (presentation, business and data source) and their main components.

3.3.1.1 Views

Views convey a domain model's perspective to the user and allows him to interact in some particular fashion. Different tasks require the user to focus on specific aspects of the model and, consequently, irrelevant or superfluous information should be hidden or removed from the view and the available set of operations limited to the task at hand. For instance, summary data may be useful to the user so he can acknowledge the achievements of his performance; the elevation profile may come in handy when analysing the challenges imposed by the terrain; and an aerial photography view may prove effective when marking a reference point or planning an activity's route.

Views may be created independently and attached to the user's interface. The view interface requires the developer to specify the position the view occupies within the application's layout, and optionally supports a custom preference page. Views can be toggled on and off dynamically and, as we shall see briefly, interact with each other through events.

In order to cover a wider range of tasks and demonstrate the application's user interface flexibility, we decided to include the following views:

Document View This view should give the user an overall perspective over the working documents, and the activities collection as well. Thus, there must be support to visualise the collection and workspace grouping elements, documents, and the three most important elements of the domain model: activities, courses and waypoints. Additionally, the user must be able to browse and select

any of those elements and perform some general operations such as adding and removing elements, importing and exporting, as well some context specific actions, for instance converting an activity to a course when an activity is selected.

Summary View This view conveys aggregate information to the user about an activity, course or waypoint, providing him a quick glance over his achievements. It includes general data, positioning data, performance data and environment data. Relevant information must be logically grouped together and the user may be able to select which data should be displayed at each moment. This view must be able to seamlessly adapt to extensions applied over the domain model, in particular knowing how to format and group new elements.

Map View This view is particularly useful and versatile as it provides graphical representations of a wide range of geographical data and allows a myriad of operations to be performed by the user. The map view is capable of displaying the user's activities in context by supporting multiple map providers and map types, and allowing several layers of geographical information from different sources to be displayed. Most of the track editing operations are only possible within this view. Additionally, the user may select individual track points, course points or waypoints, perform zooming and panning operations, and visualise animations of his activities.

Chart View This view offers a chart representation over performance data, namely speed, heart rate, cadence and power, along with the activity's elevation profile. This view must be able to display data over time or distance, accommodate multiple units and scales, and allow the user to select segments of interest.

Data View This view offers a very detailed (and numeric) approach to the user's data, covering from documents to track points, and all the elements in between. It allows the user to browse documents, activities, courses, waypoints, laps, tracks, track points, course points, segments, events and devices, and thoroughly inspect all of the available data. Additionally, this view allows context operations to be performed over a single element or a range of elements.

Log View This view shows detailed information about errors that occurred as well as actions and steps performed by the application, and it is mostly useful for troubleshooting and development purposes. Moreover, it must provide several levels of detail in order to better suit the user's needs at any particular situation.

3.3.1.2 Task Manager

The task manager has the responsibility to supervise and execute tasks generated at the user interface level, specially those that are intensive and time-consuming. More specifically, the task manager has to accomplish the following functions:

- To ensure the user's interface responsiveness, allowing longer tasks to be performed without freezing the interface;

- To monitor tasks as they are executed, allowing a progress monitor to be displayed whenever the execution time is over a predefined threshold, informing about the percentage to the task's completion, as well as the particular step being executed; this responsibility also applies when reading from an input stream or consuming an external service;
- To allow the task to be gracefully cancelled, either because it is taking too long or the user so decides;
- To provide task scheduling, allowing multiple related tasks to be executed while respecting defined priorities; for example, fetching multiple map tiles for a given region, with tiles closer to the centre of the screen taking priority over the corner ones.

3.3.1.3 Event Manager

Within the presentation layer, views communicate with each other through events. This way all the views remain synchronised despite the fact that actions often occur in a specific view. The event manager is the entity in the system responsible for managing event subscriptions and publishing events to the interested parties.

Views, and other entities alike, may subscribe their interest on a particular event by registering themselves to the entity manager. They also must be prepared to be notified (implement an interface) in case an event occurs. On the other hand, views may register themselves as event publishers to the event manager and publish events as appropriate. Publishers may choose to be notified of the events they publish, and publishing events may apply to a single element as well as multiple elements.

Events essentially fall in one of the following categories:1) selection events, when the user selects one or multiple elements through the user interface; 2) highlight events, when elements are emphasised in the users interface, for instance when the user hovers the mouse over an element; 3) update events, when elements are added, removed, moved or updated; 4) miscellaneous events, for example when an animation step as occurred or an element has been zoomed in to fit the screen.

3.3.2 Business

The business layer holds the application's business logic. It comprehends the domain model and all the operations that may be performed over it, including importing and exporting data into multiple data formats.

As any good layering scheme suggests, the business layer must have very little or no dependency from the presentation layer, only offering services the latter. Moreover, the business layer must hide the data source layer from the presentation layer whenever possible. In particular, it may receive requests to persist entities or consume external services from the presentation layer and forward them to the data source layer. Similarly, it may collect the results of the above requests and dispatch them to the presentation layer.

By including the business layer in the application's design we have isolated the system's core business, and encapsulated the domain model and its operations in a single module. This way we were

able to fulfil two important goals. On the one hand, it is possible to update or replace the reference implementation without affecting the adjacent layers, namely the presentation and data source layers. On the other hand, it is feasible to execute this layer in a different platform in the future if so required. For example, services in this layer may be made available as webservices to be consumed by a web or mobile client application.

3.3.2.1 Document Manager

The document manager is responsible for mediating the execution of operations over the domain model, making the bridge between the presentation and business layers. The fact that documents are the working units of the domain model inspired us to name this entity.

Among document manager's main functions are:

- To provide services to be consumed by other entities, specifically the presentation layer;
- To execute operations over the domain model in order to fulfil service requests; simple operations may be performed promptly, while more complex operations may be delegated in specialised entities, called operations; one special group of operations import and export data into multiple data formats;
- To dispatch requests to persist data or consume third party services to the data source layer, and forward the appropriate results to the requester.

3.3.2.2 Operations

Operations are specialised entities that perform complex actions over the domain model. They share a common interface which allows us to chain them together and execute them in sequence. This feature may be particularly useful to create templates for some of the most common planning sessions. Furthermore, operations may be customised by specifying options. Some of the most relevant operations are listed below:

- The **consolidation** operation applies to activities and courses, correcting structural problems, filling and/or fetching missing data, calculating aggregate information and recalculating original data if needed;
- The **activity to course conversion** operation allows us to use activity data in course creation during a planning session;
- The **track editing** operation manipulates the track of a course by adding, updating, removing, and moving track points; joining and splitting tracks;
- The **grade calculation** operation employs studied algorithms in the present work to calculate grades; grade information may be directly displayed to the user or serve as input to other operations such as climbs detection or pacing;

- The **climbs detection and classification** operation deals with the proposed algorithms to detect climbs based on grade and classify them according to their difficulty;
- The **pacing** operation sets the rhythm or pace of a course to be followed while performing an activity, based in one of the many methods available;
- The **track simplification** operation reduces the number of track points of a course in order to improve performance while preserving qualities of the track like the shape or elevation profile;
- **Import and export** operations allows data to be read or written in different file formats.

3.3.3 Data Source

The data source layer is responsible for persisting data, for instance in a database, and to communicate with external systems in order to fetch relevant data. Tile servers, required by on-line map providers, and elevation services, required to fetch missing or incorrect elevation data, are amongst the most important third party services used.

This layer is important as it isolates dependencies from external sources in a single module, allows us to change the underlying database provider as well as any of the external service providers used, and allows module execution in a dedicated server to achieve better performance, availability or security.

3.4 Extension Points

The proposed architecture provides several extension points which can be summarised as follows:

- Each of the elements that compose the domain model may be extended with new attributes; metadata must be supplied so that new attributes may be accommodated by existing views and operations;
- Views may be created in order to provide different perspectives and contextualise existing data;
- Map providers and map layers may be integrated as needed;
- New operations may be added to extend the application's functionality;
- Additional third party services may be included in order to support new features.

3.5 Summary

In this chapter we presented the system's three layered architecture, composed by presentation, business and data source layers. We discussed design issues and considered the main components of each layer, along with their responsibilities.

We also explored the application's domain model and reviewed the main design decisions involved in its definition.

Finally, we examined the extension points provided by the current architecture.

In Chapter 4 we will describe implementation details for all of the components just described.

4

Implementation

Contents

4.1 Introduction 32
4.2 Overview 32
4.3 Operations 47
4.4 Persistence 60
4.5 Summary 61

4.1 Introduction

In this chapter we discuss the implementation details, which closely follow the architecture guidelines presented in chapter 3.

We begin with an overview of the development process and we describe the technologies, tools and libraries which were selected. We then present the application, explain its structure and review its main components, namely the views, readers and writers, and editing capabilities.

The remainder of the chapter discusses in greater detail some of the application's core components, in particular map providers and operations. We summarise the issues to be solved, explain the design decisions and describe the algorithms employed.

The chapter finishes with a summary.

4.2 Overview

In this section we address the development process and give an overview on the application's structure and its main components. Far from being exhaustive, we will focus on the key features and contributions from the present work. Core functionalities provided by the application's framework will be introduced as needed.

4.2.1 Development Process

When the application's development began, three main requirements were taken in consideration. In particular, the application had to be:

1. Portable, in order to be executed on the three main operating systems, namely Windows, MacOS and Linux;
2. Lightweight, not only on performance, but also on packaging and distribution;
3. Self-Contained, which relates to the previous two requirements, and implies reduced dependencies on third-party products and services, as well as on target system prerequisites.

4.2.1.1 Technologies and Tools

We chose the Oracle's Java Platform, Standard Edition (SE), for the development process as it is portable and provides the performance and versatility we needed, besides allowing the development of rich user interfaces and applying cross-platform look and feels[1]. The minimum version required is Java 7, build 1.7.0_10-b18. Although we could use one of the previous versions thus reducing system requirements, Java 7 provides several features which significantly enhance the development process and provides a richer experience to the end user [17][2], while being freely available for all major platforms and operating systems.

[1] Oracle's Java Platform SE, `http://www.oracle.com/technetwork/java/javase/overview/index.html`

[2] For example, strings in switch statements, catching multiple exception types, type inference for generic instance creation, the Nimbus Look and Feel, or the JLayer Component, among other features

Several of the Standard Edition technologies were used: 1) the Java language itself, along with core libraries provided by the runtime environment; 2) the Swing User Interface Toolkit for the development of the user interface; 3) the Image I/O to render and manipulate all kinds of images, such as map tiles and icons; 4) Java 2D for rendering custom components like maps or elevation charts; 5) Internationalization API; 6) Preferences API for storing and retrieving user preferences by using the host operating system preferences mechanism; 7) JDBC library for database connectivity; 8) Java DB for database support.

In order to keep dependencies and package size to a minimum, common tasks were performed with standard Java technologies only, such as collections, string manipulation, regular expressions, date/time manipulation, concurrency control, task execution, connectivity, and marshaling/unmarshaling objects on webservice invocation. For that reason, some of the popular Java Libraries available that would eventually simplify and streamline the development process where excluded, for example, the Guava Libraries[3], the Apache Commons Collections[4], the Trove Java Collections[5], the Joda Time[6], or the Apache Axis[7], just to name a few.

On the other hand, the graphical user interface was entirely created programmatically, without the need of any GUI Builder, essentially by using a combination of some powerful layout managers, in particular the Group Layout, the Grid Bag Layout, the Border Layout, and Layered Panes. We chose the Nimbus Look and Feel so that we could achieve a consistent user interface across platforms.

The application's code was written in the Eclipse IDE for Java Developers[8], Kepler's version, on MacOS X (Mavericks).

The building process was implemented with the Apache Maven tool[9], which according to the documentation is a "software project management and comprehension tool". Besides assisting the building process, Maven also allows to manage dependencies, such as required libraries, as well as packaging the application. In order to follow the standard, the project's directory structure agrees to the one described on the documentation. Because some of the libraries we used are not in the Maven Central Repository, we had to manually install them on the local repository, for instance:

```
mvn install:install-file -Dfile=<project_base_dir>/src/main/lib/fit.jar \
-DgroupId=com.garmin -DartifactId=fit -Dversion=10.10.0 -Dpackaging=jar
```

Versioning and source control management was supported by GIT[10], a lightweight and open-source Source Control Management, very popular and widely used by several open-source projects. The fact that it is distributed implies that we use a clone of the repository, which makes backups and synchronization easier. On the other hand, local branches are cheap and easy to create and merge, which simplifies the development workflow. Those were the main reasons that led us to adopt this tool.

[3]Guava: Google Core Libraries for Java 1.6+, `https://code.google.com/p/guava-libraries`
[4]Apache Commons Collections, `https://commons.apache.org/proper/commons-collections`
[5]Trove: High Performance Collections for Java, `http://trove.starlight-systems.com`
[6]Joda-Time: Java date and time API, `http://www.joda.org/joda-time`
[7]Axis, `http://axis.apache.org/axis/java`
[8]Eclipse IDE, `http://www.eclipse.org/downloads`
[9]Apache Maven, `http://maven.apache.org`
[10]GIT, `http://git-scm.com`

4.2.1.2 Libraries

In this project we used the libraries presented in Table 4.1. We kept them to the minimum in order to achieve the performance and packaging requirements previously stated.

Table 4.1: Summary of the libraries used in the development of the Track It! application.

Name	Version	Description
HTTP Client[11]	4.2.3	This library is part of the The Apache HttpComponents project and provides components to establish HTTP connections. It is used to fetch map tiles and access remote APIs such as routing or elevation services.
Derby & Derby Client[12]	10.10.1.1	This library provides an embedded database to use within a Java Application. It has been used in some persistence related experiments.
Log4J[13]	1.2.17	This library provides logging capabilities to the application.
JUnit[14]	4.11	This library provides a testing framework for the application.
Orange Extensions[15]	1.3.0	This library contains Apple's official Java extension stubs which allows compilation on other platforms other than MacOs. Track It! uses Apple Java Extensions to enhance user experience when on a Mac, by providing the applications menu on the system's menu bar, the preferences option on the usual location (e.g., Track It! ¿ Preferences. . .), a standard about dialog and the quit operation.
Fit SDK[16]	10.0.0	This library provides a development kit for handling the fit file format (Flexible and Interoperable Data Transfer), used to import and export operations within the application.
Geohash Java[17]	1.0.6	This Library provides an implementation of geohashes in java. It is used in the geohash map layer which will be further described.

4.2.1.3 Executing the Application

The application may be executed in several ways. The simplest, in case the Java Runtime is correctly installed on the system, is to double click the application jar provided in the package (e.g., TrackIt-0.0.10.jar). The application may also be executed from the command line, provided the Java executable is available in the path, as follows:

```
java -jar TrackIt-0.0.10.jar
```

Finally, scripts are also provided for executing the application in different environments:

- TrackIt.bat batch file for Windows systems;
- TrackIt.sh shell script for Linux and MacOS platforms.

[11] Apache HttpComponents Project, `http://hc.apache.org`
[12] Oracle's Java DB, `http://www.oracle.com/technetwork/java/javadb/overview/index.html`
[13] Apache Logging Services, `http://logging.apache.org`
[14] JUnit, `http://junit.org`
[15] OrangeExtensions, a pluggable jar containing stubs for the Apple Java Extensions, `http://ymasory.github.io/OrangeExtensions`
[16] FIT SDK, `http://www.thisisant.com/resources/fit`
[17] Geoahash Java, `https://github.com/kungfoo/geohash-java`

The source code also contains a README file with information about the building process, libraries, dependencies and instructions how to execute the application, along with a NEWS file the new features within each release.

4.2.1.4 Licensing

Track It! is released under the GNU Public License (GPL) [18], version 3.0, in order to allow other students and developers to extend its functionality, while ensuring that it will continue to be freely available to the community.

Despite the fact that the license is distributed with the source code (COPYING file), every code file (.java) contains the following header (Figure 4.1):

```
/*
* This file is part of Track It!.
* Copyright (C) 2013 Henrique Malheiro
*
* TrackIt! is free software: you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation, either version 3 of the License, or
* (at your option) any later version.
*
* Track It! is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with Track It!. If not, see <http://www.gnu.org/licenses/>.
*/
```

Figure 4.1: GNU General Public Licence (v3.0) header.

The source code also contains a AUTHORS file to list and give credit to future contributors to the project.

4.2.2 Application

The application's window has three areas. The centre pane contains the map view where most of the user interaction takes place. Because activities and courses are essentially visual and since route planning tasks are easier when displayed in context, it makes sense to devote this privileged space to mapping components. The left pane supports a navigation three of opened documents and allows the user to easily select a course or an activity, as well as one of its segments, for instance. Some of the important operations are also available on this tree view. The bottom of the left pane is dedicated to presenting summary information about the selected item. Finally, the bottom pane provides auxiliary views over the selected items, namely the chart view which graphically represents elevation, heart rate, cadence, among other measurements; the data view, which displays detailed information about the selected item in a tabular layout; and the log view which allows the user to collect information about

exceptions that occur or inspect the parameters or output of an operation. Figure 4.2 gives an overview of the application's main window, displaying a commute bike ride as an example.

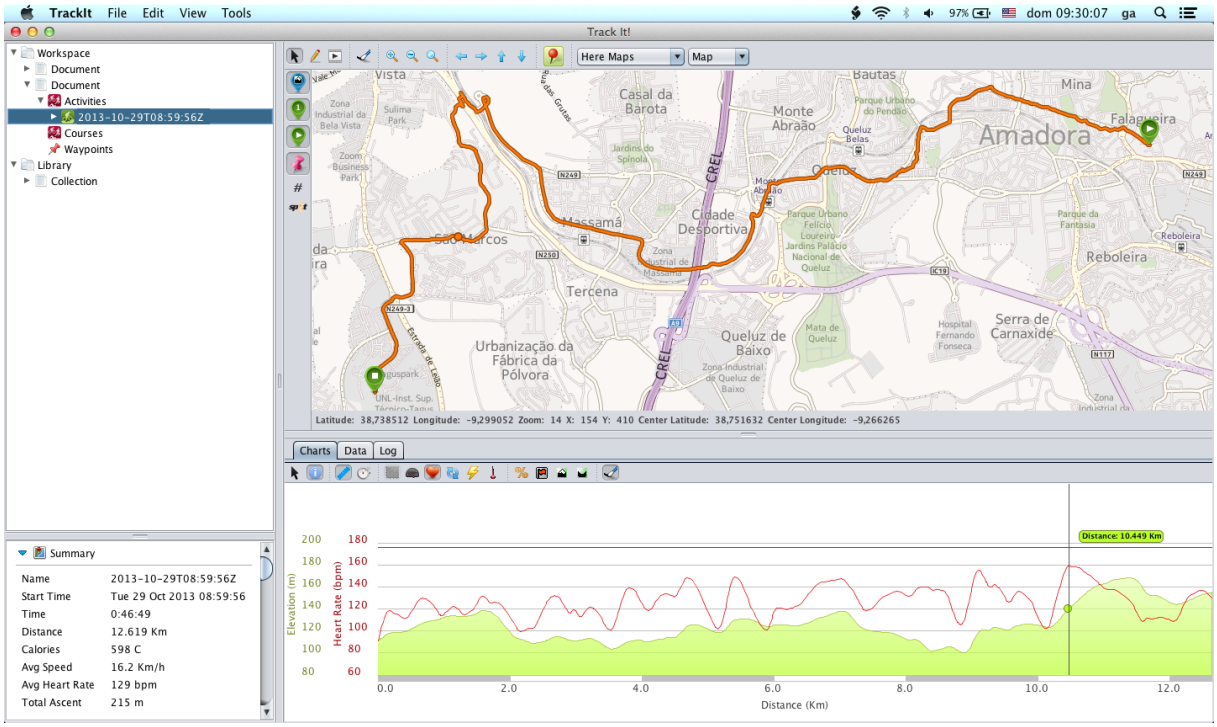

Figure 4.2: Track It! main window displaying a commute bike ride, with the following active views: document, summary, map and chart.

In order to ensure that all the views remain synchronized we implemented the observer pattern [19]. The event manager plays the role of subject as it registers observers interested in being notified of events (e.g., the views), and notifies observers whenever an event occurs (e.g., a course has been selected). Observers implement the `EventListener` interface, providing the method `process()` to handle the event. Observers are essentially (but not limited to) views. Entities implementing the `EventPublisher` interface keep the event manager informed of the occurring events so that it may notify the interested parties. Figure 4.3 presents the entities just described and their interactions.

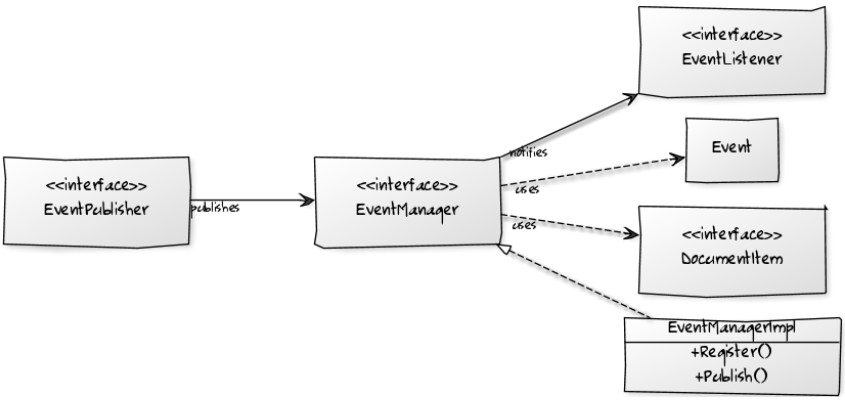

Figure 4.3: Entities that implement the observer pattern in order to keep the application's views synchronized.

In short, the process may be described as follows: 1) when the application starts up, the available views register themselves with the event manager, communicating their interest in events; 2) whenever an event occurs (e.g., the user selects a course in the document view), the component where the event took place informs the event manager of what happened; 3) the event manager then notifies the subscribing views of the occurring event, providing information about the relevant domain model elements; 4) the view updates its state accordingly.

The types of events currently supported are summarised in Table 4.2.

Table 4.2: Summary of supported events.

DOCUMENT UPDATED	COURSES SELECTED	DOCUMENT DISCARDED
NOTHING SELECTED	DOCUMENTS SELECTED	DOCUMENT ADDED
DOCUMENT SELECTED	DOCUMENT CHANGED	ACTIVITIES SELECTED
ACTIVITY ADDED	ACTIVITY SELECTED	ACTIVITY REMOVED
COURSE SELECTED	COURSE REMOVED	SESSIONS SELECTED
SESSION SELECTED	LAPS SELECTED	LAP SELECTED
LENGTH SELECTED	TRACK SELECTED	COURSE POINTS SELECTED
COURSE POINT SELECTED	EVENTS SELECTED	EVENT SELECTED
DEVICES SELECTED	DEVICE SELECTED	ANIMATION MOVE
WAYPOINTS SELECTED	WAYPOINT SELECTED	TRACKPOINT SELECTED
TRACKPOINT HIGHLIGHTED	ZOOM TO ITEM	TRACKPOINTS REMOVED
SEGMENTS SELECTED	SEGMENT SELECTED	COURSE UPDATED
FOLDER SELECTED	MISCELANEOUS SELECTION	

It is worthy to note that all of the relevant domain model elements implement the `DocumentItem` interface, which allows them to circulate within the application seamlessly. For instance, whenever an `EventListener` object handles an event, it also has access to the list of applicable items, all implementing the `DocumentItem` interface. In this particular situation the type of the domain object may be inferred through the event, but it is generally the case that the desired behaviour is delegated to the domain object itself, providing some level of independence between the entity which needs to handle the domain object and the object itself. It is often the case that the object type may not even need to be known.

4.2.3 Document View

Document view allows the user to browse performed activities, planned courses and pinned waypoints. Activities, courses and waypoints are grouped into documents, which in turn are organised in two main folders: workspace and collection. The former contains documents the user is working on, the latter represents stored documents.

Elements are presented in a hierarchically fashion[18] because this conveys better the information's nature and allows the user to unveil detailed information as needed, making the browsing experience smoother. Figures 4.4 and 4.5 show an activity and a course in document view, respectively.

Each branch allows selection as the main operation (left click). Double click performs a zoom to the item and right click displays a context menu with available operations on the selected item.

[18] JTree Component, `http://docs.oracle.com/javase/7/docs/api/javax/swing/JTree.html`

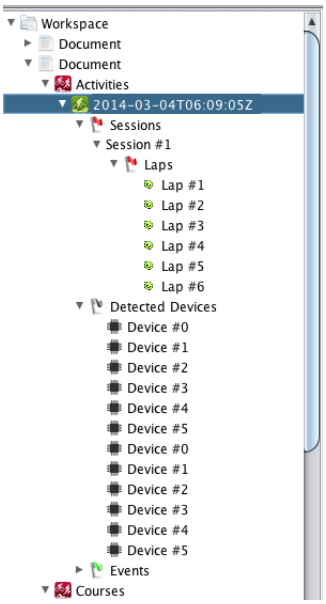

Figure 4.4: Hierarchical view of an activity.

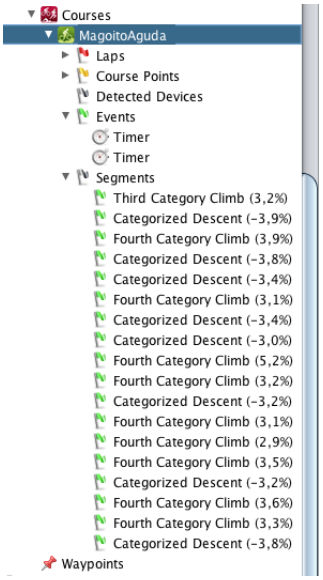

Figure 4.5: Hierarchical view of a course.

The list of available operations that may be performed over an item is obtained through an implementation of the factory design pattern [19]. The operation factory queries the item for the supported operations and then creates units of work[19] which become associated with a context menu entry and are executed at user demand.

Although not currently supported, drag and drop operations would be a convenient enhancement to user interaction, allowing the user to easily move or copy items around.

4.2.4 Summary View

The summary view allows the user to check the selected item's data at a glance. Information is logically grouped into categories to be easily identified and categories are presented in collapsible panels so that they may be shown/hidden according to user's needs. Data is properly formatted according to its type. Figure 4.6 shows the summary view for a selected activity.

The collapsible panel is a custom component that was specifically designed for this view. It extends the standard `JPanel`[20] component providing additional features such as an icon to expand/collapse the panel, which controls the panel's visibility; the collapse direction of the panel (north, south, east, west); or a custom header, with a title and an icon.

Groups and formatting take advantage of the fact that the system holds metadata information for each of the elements that compose the domain model. In particular, a specification must be provided for each of those elements in a file called `field.def`, located in the resources folder. This file is loaded at the application's start up and includes information about the element's name and data type, measurement units if applicable, the groups it belongs, and pointers to localization info. Metadata is particularly useful to automate tasks and easily accommodate changes to the system and its domain model.

By inspecting metadata information, the system is capable of 1) automatically creating collapsible

[19] Runnable interface, `http://docs.oracle.com/javase/7/docs/api/java/lang/Runnable.html`
[20] JPanel, `http://docs.oracle.com/javase/7/docs/api/javax/swing/JPanel.html`

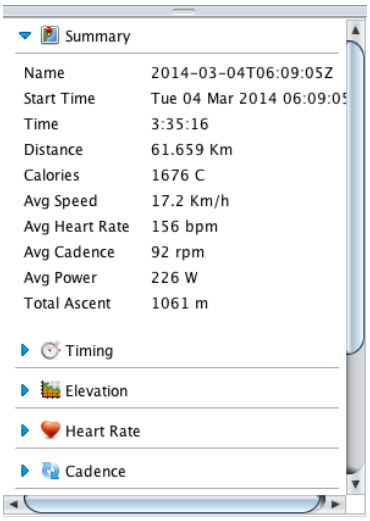

Figure 4.6: Summary view for the selected activity.

panels to shown on the summary view, once again with the aid of the factory design pattern [19]; 2) group and order the data to be presented; 3) apply adequate formatting and present the appropriate measurement unit.

Presently, data shown in the summary category is generally adequate to cycling, running or hiking activities. However, different activities emphasise different data. For instance, the elevation loss may not be relevant to a hike but it is certainly crucial to downhill cycling or skiing. This way, some degree of customisation would be desirable, perhaps as a function of the sport of the selected course or activity.

4.2.5 Data View

Data view presents the complete set of information about a domain model item in a tabular format. It also allows the user to select one of its descents in the hierarchy tree. For instance, for an activity item the user may also select the list of laps or track points contained in the activity. This view is particular useful for analysing the available information, scrolling data, and selecting ranges. It is also valuable to execute operations over a set of data, although at the present time operations are not available in this view. Figure 4.7 shows track point data for an activity as well as the available activity descendants for selection.

The data view component is backed up by the standard JTable component[21] and extended with additional features, namely data formatting and automatic table model creation. These features are supported by the domain model specification previously presented.

4.2.6 Log View

The log view is a convenient way to acquire information about application errors, operation outputs or algorithm parameters. It is specially useful for developers but it is also valuable for end users. This view currently supports six log levels (trace, debug, info, warning, error, fatal error) according to the level

[21] JTable, `http://docs.oracle.com/javase/7/docs/api/javax/swing/JTable.html`

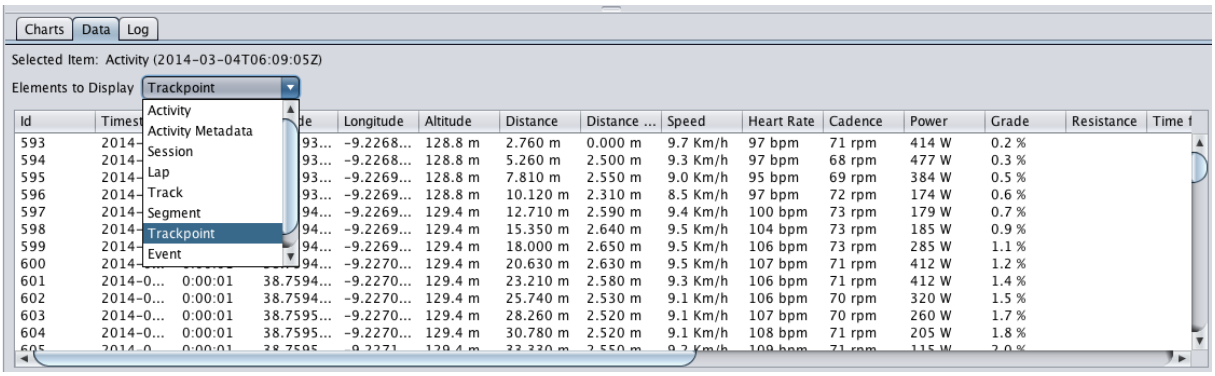

Figure 4.7: Data view for the selected activity.

of detail required by the task at hand. Additionally, all the levels may be displayed at once or logging may be disabled altogether. Figure 4.8 shows an excerpt of a log in trace level.

At the present time, logging is supported by the Log4J library (see 4.2.1.2 for details), but a custom swing component had to be designed in order to capture log events and present them to the user in a readable format. The main challenges in developing this component were to capture log events and display them in the proper formatting, ensuring that updates to the component only occur on the Swing's Event Dispath Thread, at the same time

It is worthy to note that log levels closely follow those provided by the logging framework. Some sort of refactoring would be advisable in order to create more user-friendly log levels.

Figure 4.8: Log view displaying an excerpt of a log in trace level.

4.2.7 Chart View

The chart view is one of the most important components in the application as it allows the elevation profile and performance data to be graphically displayed in a comprehensive and insightful way. In particular, the chart component is capable of displaying the following data (y-axis): elevation, speed, heart-rate, cadence, power, and temperature. This data may be displayed as a function of distance in Km, or time in the format hh:mm:ss (x-axis).

The chart component is mainly composed of: 1) the horizontal axis, which may be in Km (distance mode) or time in the format hh:mm:ss (time mode); 2) one or more vertical axis, one for each of the

displayed magnitudes; 3) the content area, where data series are displayed; and 4) a toolbar where the user can select the information to display. Figure 4.9 shows an activity chart, displaying the elevation profile, speed and heart rate. The chart is in distance mode and only the relevant vertical axis are shown.

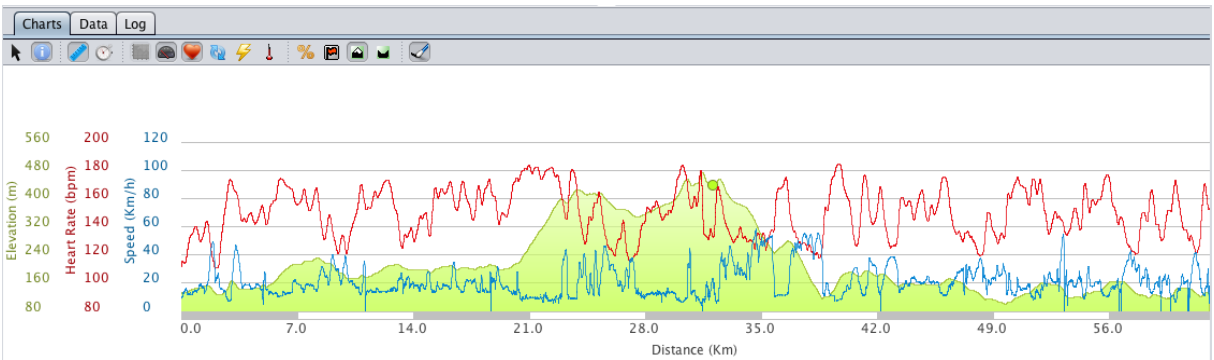

Figure 4.9: Chart view for an activity displaying the elevation profile, speed and heart rate, in distance mode.

Axis's scales are calculated according to the minimum and maximum values, as well as the number of required steps. For smaller steps, decimal values are considered.

Data may also be subject to a smoothing procedure, which basically applies a forward filter, a backward filter and a centred filter in succession, in order to remove high frequency fluctuations on data caused by GPS errors and inaccuracies [20]. The process is is conducted with a relaxation time τ and repeated a second time with a smaller relaxation time $\frac{\tau}{4}$ so that subsisting high frequency variations may be removed.

The chart component is able to display climbing segments and course points and provides zooming capabilities to inspect track segments in greater detail (see Figure 4.10).

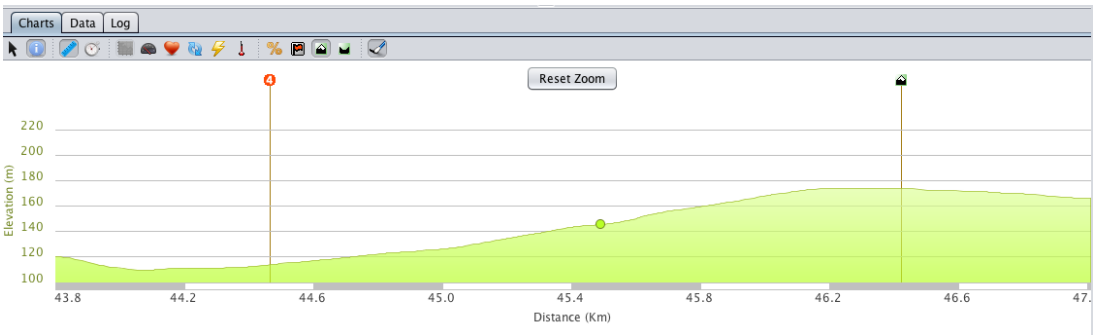

Figure 4.10: Segment zooming in chart view.

4.2.8 Map View

The map view offers a contextualised look over a course or activity. This view is capable of displaying background maps of several types and sources, supports multiple layers that extends and enhances functionality, and provides several editing and visualising tools for the user to interact. Figure 4.11 shows the map view in action.

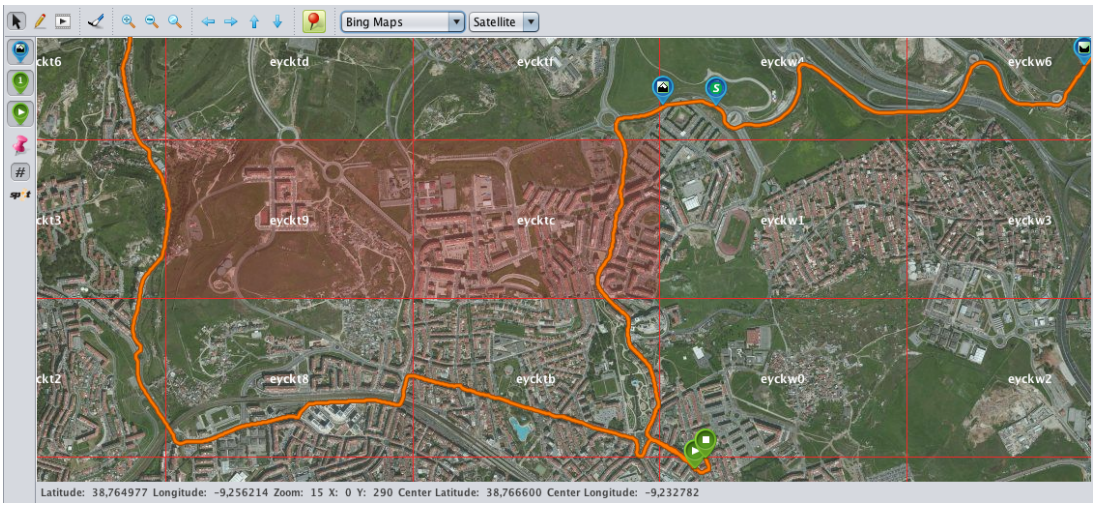

Figure 4.11: Snapshot of the map view.

Most of the functionality of the map view is supported by a pluggable map provider. The map provider interface is clearly defined and additional map providers may be developed and plugged-in in the future. The map provider's interface groups functionalities into four categories:

1. Positional functionalities, which involves displaying, converting and manipulating locations and their coordinates (longitude, latitude and altitude); zooming and panning the map view;

2. Tiling functionalities, which comprises seeking map tiles for a particular type, zoom level and location, as well as fetching map tiles from external sources;

3. Elevation functionalities, in particular fetching elevation data for a list of track points or waypoints;

4. Routing functionalities, which includes calculating routes for a given set of points, providing directions and supporting multiple options like travel mode, avoidable features or off-road incursions.

Table 4.3 summarises the map provider interface and groups the required methods into the categories just described.

4.2.8.1 Drawing Engine

The map view includes a map component which is responsible for drawing map tiles and map features on the screen.

In order to fill in the map view with the appropriate tiles, the map component asks the map provider several questions:

Table 4.3: Map provider interface methods, grouped by functionality.

Positional	Tiling	Routing	Miscellaneous
getCenterLocation ()	getMapTileGrid (width, height)	hasRoutingSupport ()	getName ()
getLocation (x, y, width, height)	fetchTileImage (mapTile)	getRoute (startLocation, endLocation, options)	getMapType ()
getCenterOffsetInPixels (location)	getTileWidth ()	hasGeocodingSupport ()	setMapType (mapType)
getCenterOffsetInPixels (longitude, latitude)	getTileHeight ()	getLocation (address)	moveCenterLocation (xOffset, yOffset, width, height)
getGroundResolution (width)	flushCache ()		moveCenterLocation (location, width, height)
getBoundingBox (x, y, width, height)			getZoom ()
			setZoom (zoom)
			setZoom (groundResolution, width)
			setZoom (boundingBox, width, height)
			getMinZoom ()
			getMaxZoom ()
			increaseZoom (width, height)
			decreaseZoom (width, height)
			setZoom (x, y, selectionWidth, selectionHeight, width, height)
			pan (direction, numberOfPixels, width, height)

- What are the longitude and latitude coordinates for the centre point on the screen, given the current map type and zoom level?
- What is the current ground resolution (meters per pixel) and how many tiles are necessary to completely fill the screen?
- What are the coordinates of the bounding box that constitutes the display area of the map view, given the current centre point, zoom level, ground resolution, and screen size?
- Can you fill the grid with the necessary map tiles for a given bounding box?

Panning and zooming operations are performed by the map provider, always having the centre point of the display as reference, as well as the amount of pan or zoom to perform.

Additionally, the map component supports a map tile cache to enhance performance and display visual cues whenever waiting for the map provider to obtain the missing tiles.

With respect of map features, the map component supports a queue of items to display. Map layers and user actions cause items to be added or removed to/from the queue of features to paint. Whenever the framework detects the need to repaint the screen, the affected features in the queue are repainted by the map component.

Each map feature supports the paintable interface. This way, rather than the map component knowing how to paint features, a combination of the factory and visitor patterns delegates the painting to the features themselves. The process may be summarised as follows: 1) the map component selects a feature to paint from the queue; 2) the map component invokes the paint method on the feature to paint as it supports the paintable interface; 3) the feature then calls the map painter factory to provide an appropriate painter for that feature; 4) the map painter factory has a few basic painters such as point painters or polyline painters and selects the one that best suits the feature to paint; 5) the feature paints itself with the aid of the just provided map painter.

4.2.8.2 Map Providers

We decided to include an implementation for following six map providers in the current work: 1) Google Maps[22]; 2) Bing Maps[23]; 3) Here Maps[24]; 4) OpenStreet maps[25]; 5) MapQuest Maps[26]; and 6) the Portuguese Military Survey maps[27].

The first five maps share common features, only differing on map tiles, elevation and routing services. In particular, these map providers use the Mercator map projection, which is conformal and cylindrical [21]; manipulate coordinates assuming the WGS84 datum; provide the same zoom levels (1-23) and ground resolutions; and consider tiles of 256 x 256 pixels as well as grids indexed by quad keys. For that reason we developed a higher level Spherical Mercator Map Provider that deals with all

[22]Google Maps, `www.google.com/maps`
[23]Bing Maps, `www.bing.com/maps`
[24]Here Maps, `www.here.com`
[25]OpenStreet Maps, `www.openstreetmap.org`
[26]MapQuest Maps, `http://www.mapquest.com`
[27]Portuguese Military Maps, `http://www.igeoe.pt`

the common logic, leaving to each of the specific map providers the contact with tiling servers and the invocation of webservices.

The Portuguese Military Maps, on the contrary, follow a radically different approach. These maps use coordinates based on the Hayford ellipsoid, the Transverse Mercator Projection and the Lisbon datum [22]. Additionally, the origin of the reference system is located 200 km west and 300 km south of Portugal's Geodetic Centre so that all the coordinates are positive. Detailed information about coordinate conversion to and from the WGS84 datum, conversion from geographic to cartesian coordinates using the Molodensky transformations, the required parameters for coordinate conversion, and the tiling scheme may be found in [22].

Map tiles consist in images in the jpeg file format, with a typical resolution of 5336×3336 which covers an area of 16 km x 10 km, stored locally[28]. Maps are fetched according to the tiling scheme defined in [22] and are subject to a caching system. Because each of these images are approximately 10 M in size we had to develop an additional tiling scheme to overcome performance and memory issues. Each tile is loaded into memory as needed and divided into 232 x 139 tiles. The resulting tiles are stored in primary and secondary caches for later use. Moreover, in order to simulate zoom levels some image manipulation is performed over map tiles, namely to modify image resolution. Figure 4.12 shows the Portuguese Military Maps in action.

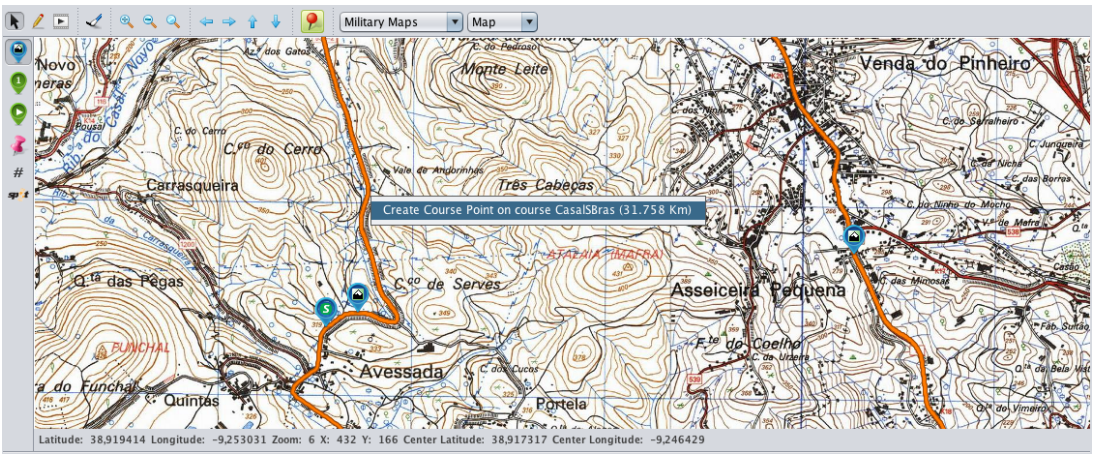

Figure 4.12: Military maps in action.

4.2.8.3 Map Types

Based on the map types available for the vast majority of map providers, we have decided to support the following map types:

- The Map type (also known as street map) is ideal for creating and manipulating routes as it displays the available roads in the routing network and provides plenty of labels for roads and locations; additionally it has a clear display and soft colours which enhances the visualisation experience;

[28] These maps are not supplied with the application. These maps are trademark of the Instituto Geográfico do Exército

- The Satellite type is specially useful for planning off-road courses or to identify buildings, landmarks or natural elements;
- The Hybrid type combines the two previous types; although the interface is sometimes cluttered, it surely provides a great amount of detail;
- The Terrain type displays topographical maps and its main strength is to display relevant data about the terrain's profile.

4.2.8.4 Map Layers

Map layers provide an extension point to add extra functionality to the map view. Map layers are pluggable and may be dynamically inserted "in front of" the map view. Two special layers are always visible. The top layer is responsible for catching events generated by user interaction and delegating them to the remaining layers. The bottom layer is the background layer and always displays a map of the selected provider and type. The order of the layers in-between is important because the higher the layer in the stack, the most priority it has to the painting engine.

Layers purpose is twofold: conveying a particular kind of information to the user and providing operations for the user to interact and execute. Several layers were developed to enhance the map view functionality and to improve the user experience, such as: map layers to display courses, activities, laps, segments, events, devices, course points or waypoints; map layers to display selected or highlighted features; map layers to provide editing capabilities; a map layer to display the geohashing grid [29]; a map layer to display waypoints collected from a spot device [30]. Other layers may be developed and added in the future.

4.2.8.5 Map Modes

Currently, the map view supports three modes. The view mode allows the user to freely pan and zoom the map to inspect features or explore map details. This mode also supports selecting features for further inspection in one of the other available views. The edit mode allows the user to modify existing features or create routes from scratch. Finally, the animation mode gives the user the opportunity to reproduce the timeline of a course or activity.

4.2.9 Course Editing Capabilities

Editing capabilities are of great significance in a course planning application. For that reason, a wide range of editing tools and operations were included.

Courses may be created from scratch using the routing support and multiple options available from any of the map providers, or they may be drawn "by hand" by the user. In any case, elevation data is fetched from remote servers and summary data for the course is calculated on the fly. Additionally, courses may also be created by splitting and stitching together (joining) existing courses and activities.

[29] GeoHash, `http://geohash.org`
[30] Spot 2 Satellite Messenger, `http://international.findmespot.com`

Course points, track points, waypoints and laps may be added or removed as needed. Map features may be zoomed in and out, selected, panned and browsed to provide the best perspective while editing.

4.3 Operations

In this section we describe some of the application's main operations in detail. We chose the presented set of operations based on its relevance, complexity and which set this solution apart from other applications. For each operation we start with the motivation and a description of the problems to be solved. We then summarise the goals to be accomplished and present the rational for the selected approach. Finally, we describe the operation's algorithm and provide an illustration for the operation in action.

4.3.1 Consolidation

In order to achieve the best results when visualising, analysing or executing operations over courses and activities, it is essential that the underlying data is valid, consistent and complete. Besides, these requirements must remain valid whenever changes are performed over that data. This is specially true when acquiring data from unreliable or incomplete sources, editing tracks, or applying operations which affect the activity's consistency.

The consolidation operation ensures the above requirements are met by executing the following actions: 1) data validation (e.g., verifying that timing information is present); 2) data correction (e.g., adjusting a lap's finish time so that it remains within the activity boundaries); 3) missing data calculation (e.g., calculating grades or fetching elevation data); and 4) summary data calculation (e.g., determining the average speed or the elevation gain).

Every element in the domain model supports consolidation. The consolidation operation orchestrates the process by calling the consolidation operation on the document's root elements, namely activities, courses and waypoints, which in turn cascades the operation to the dependant elements in the hierarchy.

The consolidation operation currently supports three levels. The consolidation levels are cumulative in which a level performs all of the preceding levels actions besides its own, and may be concisely described as follows:

Basic Ensures data consistency and timing information; corrects data if invalid and assumes a constant speed (currently parametrized as 10 km/h) if timing information is missing; future versions may enable customisation or set values according to activity type;

Summary Performs actions from basic level and fills in missing data, either by performing calculations, interpolating values or fetching data from external sources;

Recalculation Performs actions from basic level and recalculates all available information, except positional data (longitude, altitude) and altitude;

Currently, the consolidation operation is triggered automatically whenever data is imported or an operation that affects data consistency is performed. The operation is also executed on demand at the user's request. The consolidation dialog is presented in Figure 4.13.

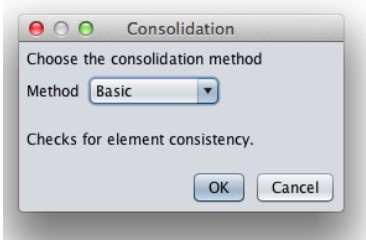

Figure 4.13: Consolidation operation dialog.

Several improvements may be introduced to this operation in the future:

- presenting summary info to the user about the required actions;
- providing fine-grained control to the user over the actions to apply;
- displaying a report with the operation results.

4.3.2 Grade Calculation

Some of the most important contributions from this work rely on the knowledge about the steepness and direction of a given segment or at specific location along a course. For instance, the user may visualise the steepness of the terrain to analyse and predict performance; climbs and descents may be detected, classified and annotated based on the steepness and direction of the terrain, a reliable predictor of the effort needed to be done in order to overcome those segments; the pace of an activity may be established as a function of the past history of performances on terrains of the same steepness and direction.

The steepness at any location within a course may be thought as the rate of change in altitude for a given distance. This quantity is known as grade or slope. The higher the rate of change in altitude in relation to distance, the steeper the terrain, and higher calculated grade or slope. The slope direction simply indicates if the terrain is ascending (climb) or descending (descent).

Mathematically, the rate of change at any point within a non-linear function is given by the derivative of that function, more specifically, it represents the slope of the line tangent to the function at that point. More formally,

$$\frac{dy}{dx} = \lim_{\delta x \to 0} \frac{\delta y}{\delta x}$$

Unfortunately, we do not know the non-linear function that represents the terrain profile for a given course. In fact, we have a collection of discrete track points that are connected together to form a polyline. So, in practice, we have a linear approximation of the function that represents the altitude profile. For that reason, it becomes trivial to calculate the slope for any given pair of consecutive points by the formula:

$$m = \frac{\Delta y}{\Delta x}$$

where m represents the slope of the line that connects two contiguous track points, δy the altitude difference and δx the travelled distance. Slopes are almost always represented by a percentage and are often expressed as:

$$\frac{rise}{run} \times 100$$

Consequently, a track point's grade could be obtained simply by:

1. calculating the grade for the line connecting the previous track point to the current track point;
2. calculating the grade for the line connecting the current track point to the next track point;
3. interpolating the previously calculated grades.

However, there are numerous factors that render the grades calculated by this method very inaccurate.

Let's begin to consider Δx or the horizontal change. To accurately calculate the horizontal distance we would have to consider a straight line between the two track points and take into account the curvature of the earth at that location. Yet, we frequently measure distances with odometers or GPS sensors, which in reality corresponds to the slope length (the actual travelled distance). In those situations, we would have to use the Pythagorean theorem or perform some trigonometric calculations in order to achieve the correct figures. Fortunately, using the slope distance to calculate grade normally does not produce errors above 3%, which are negligible in practice.

Moreover, the GPS device horizontal accuracy is somewhat limited, providing coordinates that may have measurement errors of several meters. These errors become visible when we inspect the fluctuations in the horizontal speed for a given activity. This may introduce severe errors in grade calculation, specially at greater sampling rates such as every second.

Altitude or vertical change measurement errors are nonetheless more dramatic (Δy). GPS based altitude measurements suffer from large fluctuations because of the reduced accuracy and interferences, and GPS devices equipped with barometric altimeters have to apply corrections to the collected data in order to compensate for small variations in air pressure as a result of factors other than altitude change. Once more, the chart for the vertical speed of a recorded activity will show great fluctuations.

One possible approach to improve grade calculation accuracy would be to apply a filter in order to smooth altitude and distance data. In [20] such a procedure is described. The author developed a filter that performs backward, forward and centred filtering in succession. The filter strength is controlled by the time relaxation parameter τ and the filter is usually applied two times in a row for each measurement in order to completely remove high frequency fluctuations. The filter is applied to distance and elevation data. Vertical and horizontal speeds are then calculated based on the smoothed altitude and elevation data and then subject to the filter. Slopes are finally calculated based on the smoothed vertical and horizontal speed data.

The above procedure really produces smooth variations in grade but our experiments also revealed that this attenuation also results in calculated grades lower than real grades, specially when calculating maximum and minimum grades.

Our approach consists of calculating the grade for a collection of track points or segments, instead of pairs of contiguous track points, and then interpolating grades for the track points along the line. This way we attenuate the errors described in the previous paragraphs.

However, in order to achieve good results we had to solve two remaining problems: what is the segment length that produces the best results and how to calculate segment data when the collected track points are not equally spaced.

For the first problem we tried to determine what would be the segment length that would better approximate the function of the curve that represents the terrain profile. For that we used integration, by calculating the area below the altitude chart as a function of distance for various segment lengths and analysed the results. We have noticed that the calculated area would rapidly converge till the segment length reaches 100 m, and from that point on the values would oscillate randomly. Further experiments led us to the conclusion that 70 m segments have the best performance across different conditions: data accuracy, sampling rates, data sources, among others.

For the second problem, we applied a resampling technique in order to obtain track points that are exactly 70 m apart.

Grade calculation results will be further discussed in the evaluation chapter (see Chapter 5).

4.3.3 Climb and Descent Detection, Classification and Annotation

The terrain profile is a key feature to consider when planning and performing an activity. Not only it helps choose a route that meets our expectations, but it is also crucial to analyse and predict the course's difficulty and duration, as well as manage the effort that needs to be done.

A terrain section may be generally classified as a climb, a descent or flat terrain whether the altitude increases, decreases or remains approximately constant, respectively. Climbs are often demanding and difficult obstacles to overcome and we are primarily interested in their steepness and length. It may be the spot for an attack during a cycling competition, or the reason we may conserve some energy in a leisurely ride. Descents helps us to recover from previous efforts and make up some time, but they are also frequently dangerous, requiring special care, full concentration and technical skills. Flat terrain may be ideal for a relaxing activity or a recovery run.

It is our purpose to successfully identify these three types of terrain sections, classify them according to their features and annotate them in courses in order to make this information available while performing the activity. These three goals correspond to three different operations, bounded by the concept of segment. In particular: 1) the detection operation divides a course into climb, descent and flat segments; 2) the classification operation takes each one of these segments and categorizes them; finally 3) the annotation operation includes information about the categorized segments into the course according to the user's preferences.In the following sections we describe these operations in more detail.

4.3.3.1 Detection

In order to accurately identify terrain segments, we have to solve three main problems. Firstly, our approach to detect terrain segments relies on grade. As we saw earlier in this chapter, grade calculation is very prone to errors and fluctuations. Provided that we have achieved smoother grades with the grade calculation operation, small variations prevail. Thus, the challenge is to identify segments where the grade is not strictly increasing or decreasing.

Secondly, the detection algorithm may result in a large number of segments, specially on irregular terrains and lengthy courses. However, while performing an activity, an athlete is only capable of focusing on a small number of goals. Overloading him with information about segments is not only distracting but also highly ineffective. This way, we have to filter "noisy" segments and be able to extract the relevant and useful information to present to the athlete.

Finally, there are certain situations when we will want a segment to be interleaved with segments of a different kind. This is often the case with longer climbs which may have small stretches of flat terrain or even descents and still be considered a single climb. These climbs are also known as multi-part climbs. For example, Serra da Estrela climb in Portugal is a 28.5 km climb from Seia to Torre which features two stretches of 2 km each of flat terrain / descent while being considered a sole climb (see Figure 4.14)[31].

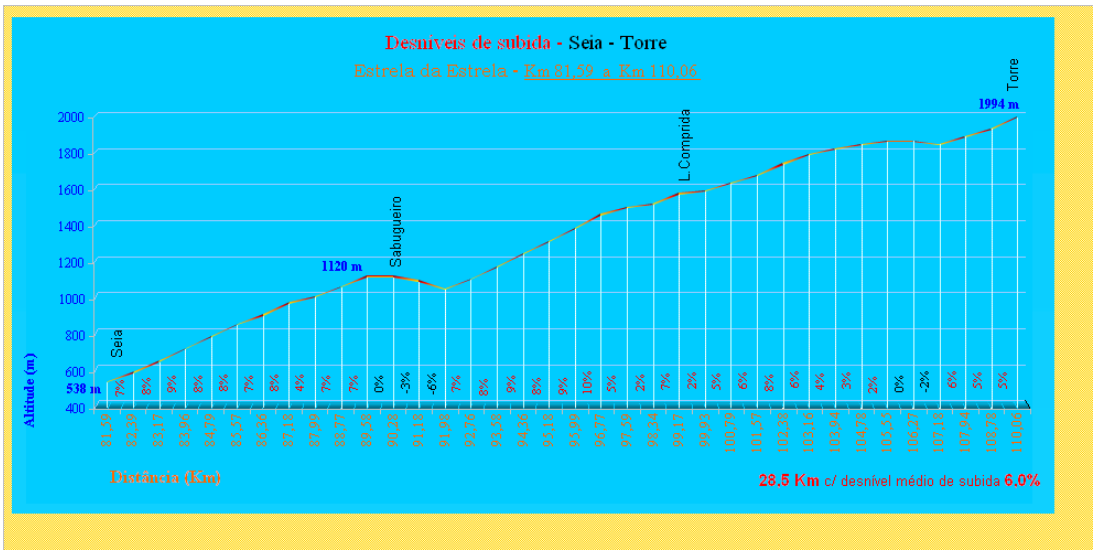

Figure 4.14: Serra da Estrela climb profile (Portugal), from Seia to Torre.

The algorithm for obtaining climb, descent and flat segments may be summarised as follows:

```
OBTAIN-SEGMENTS(course)
    create segments by grade
    merge climbs
    merge descents
    fine-tune segments
```

In the first step, we begin to classify track points according to the calculated grade. Track points with grade greater than or equal to the `climb threshold` are classified as climb track points; the ones with grade less or equal than the `descent threshold` are classified as descent track points; finally,

[31] Retrieved from `www.forumciclismo.net` on 25th April 2014

the remaining track points are classified as flat. Contiguous track points of the same type are then joined together into a segment. The `climb threshold` and the `descent threshold` parameters correspond to the minimum and maximum average grades allowed for a categorized climb or descent, respectively. At the end of this stage we have a list of small segments which strictly correspond to climbs, descents or flat terrain.

In the next two steps we merge non-contiguous climb and descent segments in order to achieve larger segments of the same type. The two steps actually share the same algorithm but with a different type parameter. The merge algorithm may be described as follows:

```
MERGE-SEGMENTS(segments, segment-type)
    type-segments ← filter segments with type = segment-type
    sort type-segments by distance in descending order

    while size(type-segments) > 0
        segment ← first(type-segments)
        adjacent-segments ← obtain adjacent segments for segment

        for each adjacent-segment in adjacent-segments
            merged-segment ← merge(segment, adjacent-segment, segments in
                between) from segments

            if merged-segment is valid then
                replace segment, adjacent-segment and segments in between by
                    merged-segment in segments
                remove adjacent-segments from type-segments
                add merged-segment to type-segments

        remove segment from type-segments
        sort type-segments by distance in descending order
```

At the core of the algorithm we make a tentative merge of a pair of segments of the same type. The merged segment has segments of the same type at both ends and segments of different types in-between. We then check if the merged-segment is acceptable by applying the following criteria:

1. The average grade for the merged segment must remain above the threshold for that particular type (`climb threshold` or `descent threshold`);

2. The percentage of track points of the same type as the merged segment type must be equal or greater than the minimum percentage specified for that type (`minimum climb percentage` and `minimum descent percentage`);

3. The category of the merged segment must be equal or greater than the maximum category of the merging segments; segment categories will be explained in detail in the next section.

In the last step, we extend both ends of climb and descent segments so that the extremes match valleys and summits. It is often the case that the start and end portions of a climb or descent have smaller slopes and thus are frequently excluded by the algorithm. This step is only performed at the user's request because sometimes we want to be notified where the categorised climb ends, and in other occasions we want to be notified of the point where the terrain really stops ascending. The same logic applies to the start of the climb and to descents.

With this approach we ensure that a manageable number of segments is achieved at the end, and that all the significant climb and descent segments are detected, even though their grades may not be monotonous increasing or decreasing. We also account for multi-part climbs and slighter slopes at the start and end of segments. The final selection of relevant segments is made at the classification stage, an operation we describe next.

4.3.3.2 Classification

Segment classification in general, and climb rating in particular, intends to quantify the segment's difficulty by providing category labels or numeric figures that enable us to compare them with other, often known, segments. Additionally, ratings are also used in races (e.g., cycling) or events to award points on a particular competition and to intensify media coverage or to increase sport fans enthusiasm.

There are several methodologies, formulas and classification systems available to rate climbs[32],[33],[34]. Nevertheless, some factors are frequently considered when rating a climb, in particular:

1. the average grade, maximum grade and altitude difference, which provides a general picture of the climb's difficulty and steepness;
2. the climb's length and position in the course;
3. the presence of downhill or flat sections within the climb;
4. the climb's altitude, as it is well documented that performance decreases as altitude raises above 1000 m, as the quantity of available oxygen also decreases;
5. the climb surface characteristics, weather conditions, etc.

In this work it is not our intention to provide a special rating system but simply to illustrate that segment classification is an important feature in course planning activities, as it helps to put difficulties in perspective. For that reason, we decided to apply the rating criteria commonly used in road cycling races, specially world tour races such as the Tour de France[35]. However, in order for this classification system to be useful to amateur athletes and leisurely activities, we relaxed the requirements a little, specially in the lowest difficulty climbs. In the future, other categorisation systems may be employed as needed.

Despite the fact that the Union Cycliste Internationale (UCI)[36] provides guidelines on how to rate climbs in races, it is generally accepted that climb ratings are frequently subjective, ambiguous and often used to the sport event's organization advantage. Nevertheless, for a climb to be rated it must have at least 500 m in length and a 3% average grade. Figure 4.15 shows various Tour de France climb ratings from the 1999 to the 2007 editions, along with the average grade and length, as an example.

There are commonly five categories for rating climbs:

[32]Climbbybike Index, `http://www.climbbybike.com`
[33]How are Strava climbs categorized?, `https://strava.zendesk.com/entries/20420292`
[34]FIETS-index, `http://www.fiets.nl`
[35]Tour de France (TDF), `www.letour.fr`
[36]UCI - Cycling's International Federation, `http://www.uci.ch`

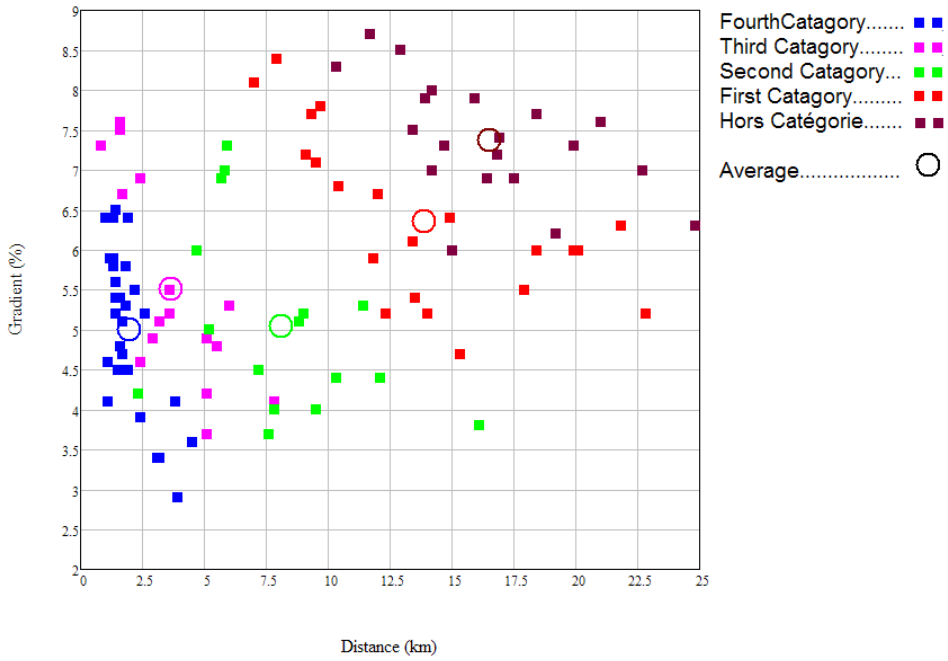

Figure 4.15: Tour de France climb ratings for climbs included in the 1999–2007 editions.

4th category Those are the easiest of the climbs; they may be short and steep, typically less than 2km at 5% average grade; or longer and less steep, usually between 2-5 km long and with a 3% to 4% average grade;

3th category These climbs may be less than 2 km long if the average grade raises above 6%, or range between 2-7 km at a 5% average grade;

2th category These climbs are usually 5-12 km long and have an average grade of 4% to 7%;

1th category These climbs are typically longer than 8 km and frequently touch the 20 km mark; average grade falls between 5% and 8%;

hors category Those are the most difficult climbs and often coincide with the end of the stage; they are no less than 10 km long, but their length usually ranges between 15 km and 25 km, at an average grade of 6% to 9%.

The rating system adopted in the current work, contrary to UCI races ratings, is completely objective. We also ensured that all the cases in the matrix of possible lengths versus average grades for categorised climbs were covered.

It is worthy to note that some of the categories' lower bounds were decreased and that the maximum grade was also considered in the climb rating system. Moreover, we extended categorisation to descents, although we only considered the distinction between categorised and uncategorised descents. In particular, descents may be short and steep, typically less than 1 km in length and a maximum grade equal or greater than 15%; or longer than 1 km at an average grade of 3% or more.

4.3.3.3 Annotation

After segments are successfully identified and properly classified, the user may choose to include information about the categorised climbs and descents in the course. This way, he may have access to useful information while performing the activity whenever the GPS device offers support for course points or waypoints, namely 1) a visual and/or audible alert to the climb or descent start and finish points, as well as visual cues on the GPS map; 2) time and distance estimates to the starting and finishing points of a climb or descent at any instant; 3) length, average grade and maximum grade of each of the categorised climbs and descents; 4) the locations where the climb or descent is most steep.

Figure 4.16 shows the annotation dialog (marking operation). The user may choose to add marks to the start and finish points of climbs and/or descents, as well as maximum and minimum grades locations. Optionally, the user may choose to substitute previous markers. In addition, the user may complement course marking with manual course points or waypoints.

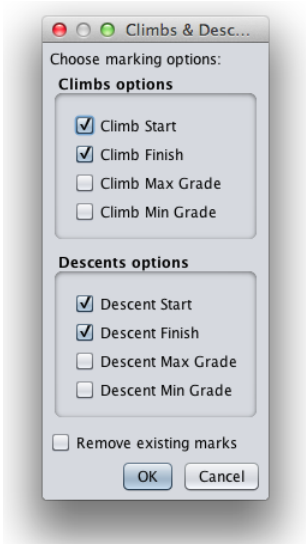

Figure 4.16: Annotation dialog.

4.3.3.4 Visualisation

We would like to conclude this section devoted to segment detection, classification and annotation with a small description of the available tools to visualise segments. Figure 4.17 shows an application snapshot displaying the main views for a segment. Segments are displayed in a different colour at the map view, and icons are provided for course points, if available. Chart view displays categorised climbs in a darker shade. Course points are also represented, if available. The document view allows segment browsing and the summary view displays useful information about the segment, such as average, minimum and maximum grade; length and estimated time to complete.

It is important to note that only categorised climbs and descents are portrayed in the above views. For a detailed analysis of all of the segments the data view may used, which presents the complete information about the available segments in a tabular layout.

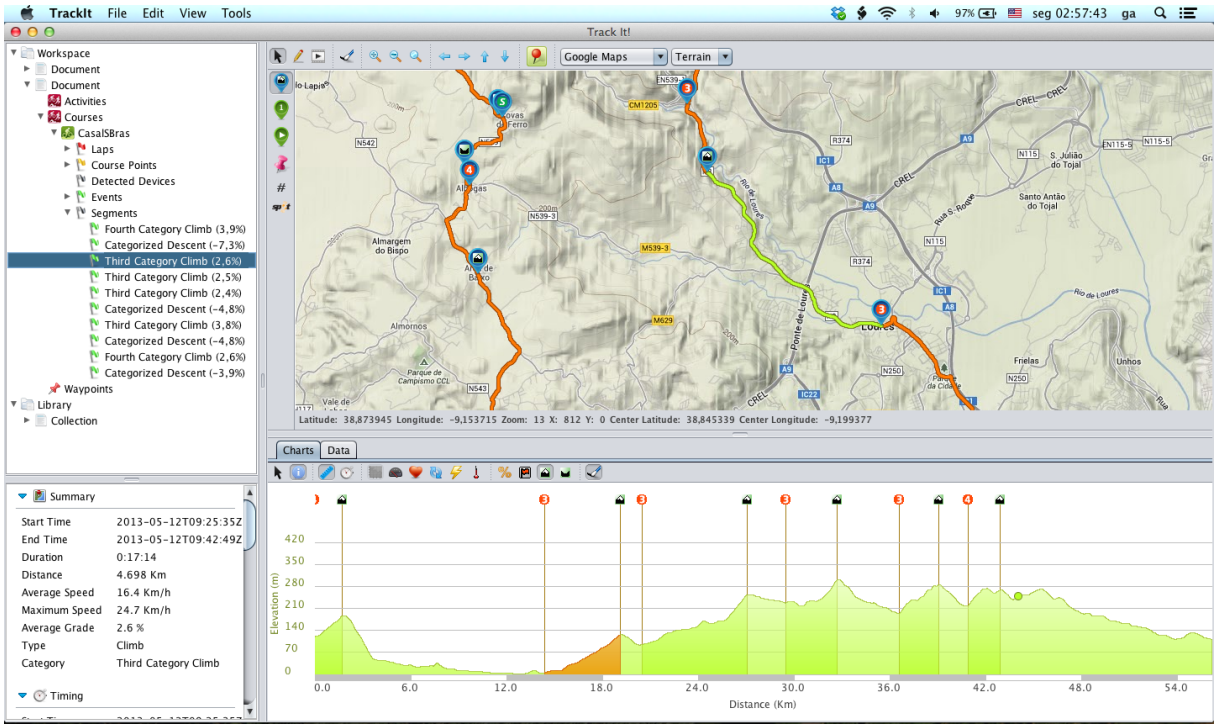

Figure 4.17: Document, summary, map and chart views of the selected segment.

4.3.4 Pacing

The pace of an activity consists on the rhythm or speed with which its route is traversed. Setting or modifying the pace of a course during the planning stage is important because it allows us to achieve the following goals during the activity: 1) to have better predictions about the remaining time to a particular location or to the end of the activity; 2) to have consistent and constant feedback about our performance in relation to what we have planned; in particular we may be informed about the time and distance ahead or behind the proposed pace.

The pace of an activity may be summarised by its average speed. Nonetheless, this figure provides little information as speed constantly varies along the route. For that reason, GPS devices often use the instantaneous speed at each location to make predictions about the remaining time and how closely we are following the established pace.

The speed recorded at every track point is a function of the travelled distance from the previous point and the time spent between points. More formally, $v = \frac{\Delta d}{\Delta t}$ that is, the rate at which distance is travelled as a function of time. Because distance is calculated from the latitude and longitude of each track point, thus fixed, modifying the pace usually means to set a new timestamp for every point.

In this work we explored six different methods to set the pace for an activity:

Target Time This method allows us to set the target duration of a particular course. It begins to determine the difference between the current duration and the target duration. Then, for each track point its contribution to the activity's time is calculated and the weighed difference is applied to the track point's timestamp. It is important to note that speed variations between track points are preserved

by this method. More formally, the speed function is shifted along the y-axis by a constant value.

Target Speed This method is useful when we intend to achieve a certain average speed at the end of the activity. In practice, this method is very similar to the previous, except for the initial step which calculates the time needed to travel the activity's distance at the desired target speed. Once more, speed variations between track points are preserved by this method.

Constant Target Speed This method simply calculates each track point's timestamp by determining the time needed to travel the distance from the previous location at the desired speed. Contrary to the previous method, speed along the route and speed variations between track points remain constant. This method may be the only possible method to apply when there is not timing information about a course, although its usefulness is very limited as it does not take into consideration speed variations caused by the terrain profile, for example.

Time Percentage This method is handy whenever we want to improve a previous performance or reproduce a previous performance at a lighter pace. For instance, we may want to cut 5% from the time needed to travel the activity's distance in relation to a previous performance. This method is very similar to the target time method, except that it initially calculates the target time by applying the percentage indicated by the user to the current duration.

Speed Percentage This method follows the same approach of the previous method applied to speed. For instance, we may want to improve our average speed by a specified percentage. The new average speed is calculated by applying the specified percentage the current speed. Then the target speed method is executed in order to achieve the desired result. Once more, speed variations between track points are preserved.

Smart Pace This method may well be the most useful and also one of the most original contribution from this work. The goal is to estimate the speed at each track point location based on the past history of rides from the user. We expect that speed variations in real performances are largely caused by the terrain's profile. Climbs take longer to overcome and descents are quicker to travel. On the other hand, the terrain's profile is expressed by the calculated grade at each track point. This method compiles the average speed at each grade based on previous activities. Then, for each track point, it calculates the time needed to travel the distance from the previous track point at the speed indicated by the current grade. This way, real performances may be estimated in advance with greater accuracy and higher quality information may be delivered to the user during the ride. Contrary to the previous methods, this approach also works even when no timing information is available. Additionally, this method may be combined with the target time/speed and the time/speed percentage methods for further improvements.

In the implementation of the pacing operation we used the factory pattern [19] to produce the appropriate pace maker for the chosen method. Moreover, the presented methods share common algorithms, which were conveniently refactored into higher level pace makers, which implement the template method. Only specific inputs to the algorithms are provided by the lower level pace makers.

The user may choose one of the previous methods in the dialogue presented in Figure 4.18. The required input parameters varies for each method. It may be a percentage, a speed value, a time value or no parameters at all.

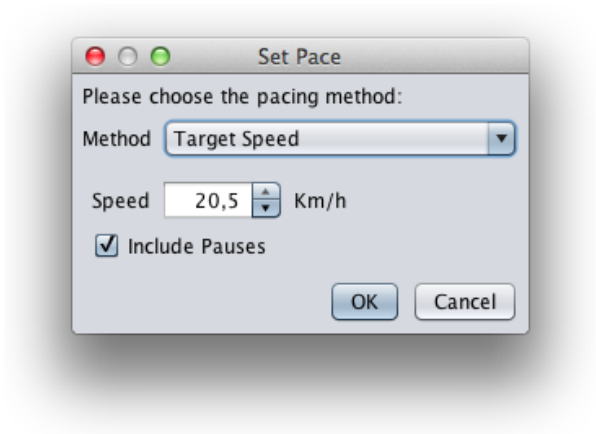

Figure 4.18: Dialogue for setting the pace of a course.

Figure 4.19 shows the result of applying the smart pace method to a course where no timing information was previously available. As we can see, speed decreases during climbs and the rate of decrease varies as a function of grade. Similarly, speed in descents increases.

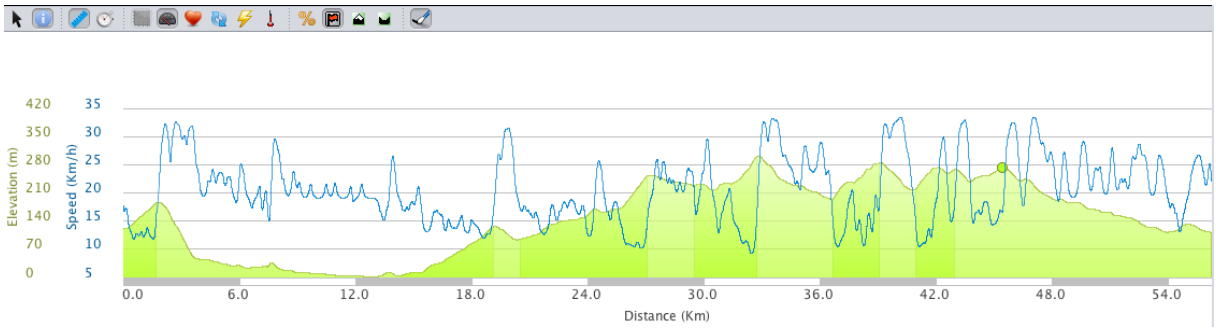

Figure 4.19: Chart view with the results of applying the smart pacing method to a course.

4.3.5 Track Simplification

The track simplification operation allows us to reduce the number of track points that constitute the track of a course while maintaining the relevant information. Simplification has a significant impact on performance, not only when a course is displayed on a GPS device, but also when rendering and processing the course in the application. Besides, file size is considerably reduced which improves transfer times and file manipulation.

The main goals for the track simplification operation are: 1) to maintain the track shape, an important feature where navigation is concerned; 2) to stop the track from being simplified when the maximum distance between two consecutive track points has been reached; 3) to preserve start and finishing points, as well as reference points like course points or lap boundaries; 4) to achieve a greater density

of points in climbs, segments where accuracy is most important and speed is greatly reduced.

A track is mainly constituted of track points and its shape is derived from the polyline that results from connecting contiguous track points. Therefore, when we started thinking of simplifying tracks we found an excellent review of polyline reduction algorithms in [23].

Nevertheless, the most accepted and widely used algorithm for simplifying GPS tracks is the Doublas-Peucker Algorithm[12]. It starts by connecting the start and end points by a direct line. Then the farthest point from that line is obtained and its distance is checked against a predefined threshold. If the distance is greater, the original line is split in two by using the farthest point as reference, and the algorithm is then applied recursively to the resulting lines. Otherwise, the points in-between the start and end points of the line being considered are removed.

Taking in consideration the goals we proposed for this algorithm, he decided to follow a different approach. We started off with the implementation described in [24], which may seen as a variation of the Douglas-Peucker algorithm, and we made further adjustments in order to meet our requirements. In short, the algorithm takes each triad of contiguous track points and draws a line between the two farthest apart. Then the distance from the middle point to the line connecting the other two is calculated (cross-track distance) and then stored in a list of cross-track distances. The list is sorted by cross-track distance, descending order, and the middle point associated with the first cross-track distance is removed. The process is repeated until the desired number of track points is reached.

Our modified version of this algorithm includes the following add-ons: a stop condition is triggered whenever removing the point associated with the minimum cross-track distance results in a distance between the two remaining points that is greater than the maximum distance allowed between two consecutive track points, as defined by the parameter `maximum trackpoint distance`; the sorting algorithm takes a weight parameter in addition to the cross-track distance in order to prioritise track points removal. The weight parameter has the following effects:

- start and finish points, as well as reference points (e.g., course points, lap boundaries), are given the least priority;
- track points belonging to climb segments (positive grade) are given a lower priority than track points within descending or flat segments.

The algorithm may be synthesised as follows:

```
TRACK-SIMPLIFICATION(course, number-of-points)
    cross-track-list → initialise cross track list with course←trackpoints
    compute cross-track distances for cross-track-list
    compute cross-track weights for cross-track-list
    sort cross-track-list in descending order, by distance and weight

    while size(course←trackpoints) > number-of-points and
        cross-track → first(cross-track-list)
        remove cross-track associated track point from course←trackpoints

        adjust adjacent cross tracks from cross-track-list
        sort cross-track-list in descending order, by distance and weight
```

The track simplification dialogue presented in Figure 4.20 allows the user to select the number of points to be removed. He also has the option to physically remove the candidate track points or simply make them invisible.

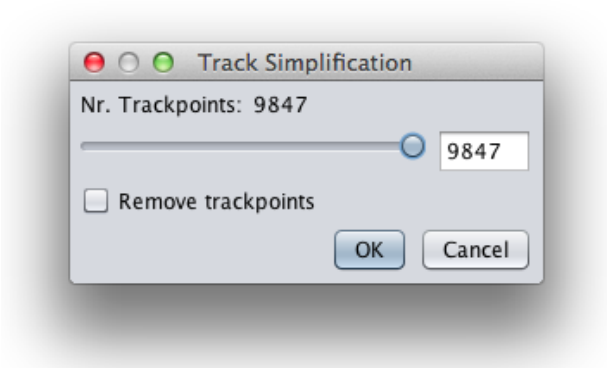

Figure 4.20: Track simplification dialogue.

4.3.6 Removing Pauses

The pauses removal operation is a simple feature that is most useful when following a reference pace which contains pauses and the GPS device supports auto-pause, that is, the timer is automatically stopped whenever the speed is 0.0 km/h or drops below a specified threshold. In this situations, the reference timer stops whenever a stop from the past performance is found, but both timers are stopped when the user manually stops the activity timer or it is automatically stopped by the auto-pause feature. This results in an "unfair" reference pace and incongruence in timing predictions.

Removing pauses simply calculates the speed between contiguous track points. If speed drops below the `pause threshold` parameter, the current offset is updated with the time difference between the two track points, and second track point acquires the first track point timestamp. At each new step track points' timestamps are subtracted from the current offset. In practice, subsequent track points will be shifted left as new pauses are found.The algorithm may be described as follows:

```
REMOVE-PAUSES(course)
    offset → 0
    for each pair (track-point-1, track-point-2)
        track-point-1←time → track-point-1←time + offset
        track-point-2 ←time → track-point-2←time + offset
        speed → calculate-speed(track-point-1, track-point-2)

        if speed < speed-threshold
            time → track-point-2←time - track-point-1←time
            offset → offset + time
            track-point-2←time → track-point-1←time
```

4.4 Persistence

Some experiments were conducted in order to test database support for a catalogue of activities and courses. In particular, Derby Embedded Database[37] was used to store activities with some degree of

[37] Apache Derby DB, `http://db.apache.org/derby`

success. Nevertheless, because this was not the focus of our work, and also because file import and export operations are available, database related functionalities were left for future work.

4.5 Summary

In this chapter we have reviewed some of the most important implementation details. We started with an overview of the development process. We then proceed to describe several aspects of the application, including its many views and features, map providers and editing capabilities. An in-depth explanation of the most relevant operations was introduced. The chapter ended with some notes about persistence.

In the next chapter we will devote our attention to the evaluation of some of the most important aspects of the proposed solution.

5

Evaluation

Contents

5.1 Introduction 63
5.2 Usability Tests 63
5.3 Algorithms Tests 67
5.4 Conclusion 76

5.1 Introduction

In this chapter we present the solution's evaluation.

Evaluation is an essential step in every system's design as it allows us to measure its effectiveness, efficiency and usability [25]. Moreover, evaluating a system often involves answering three fundamental questions: 1) what is the purpose of the evaluation (why); 2) what should be measured by the evaluation procedure and which tools and/or methodologies should be selected (how); 3) in which moment should the evaluation take place (when).

In the case of the present study, evaluating the solution allowed us to acknowledge if the application meets the requirements defined in Chapter 1, if the user is able to perform common planning tasks as advertised, and if the implemented algorithms are correct, efficient and produce the desired results. On the other hand, specific problems and limitations were identified, which are not only important for this work's conclusions, but also for proposing future research.

Therefore, for the solution's evaluation we have selected three different approaches:

1. To perform usability tests in order to verify that an ordinary user can complete typical use case scenarios;

2. To undertake comparison tests between a) what has been planned during a planning session; and b) what has been collected and measured while performing the activity, by following the pre-established plan.

3. To conduct experiments in order to validate the correctness and accuracy of the application's core algorithms, and compare the results with accepted standards whenever possible.

Finally, evaluation was accomplished only once at the end of the implementation process in spite of user interaction being on the top concerns of the current thesis, mainly because of time constraints. Nevertheless, traversal test sessions were conducted several times at the end of implementation cycles. In this, typical test data was applied.

In the following sections we will describe in greater detail the selected evaluation approaches and their results.

5.2 Usability Tests

Usability tests are crucial whenever we need to collect data about the completeness of the system's functionalities, as well as the effectiveness and efficiency with which an user can execute a set of proposed tasks. Moreover, this kind of testing frequently involves observing users interacting with the system, a technique which provides plenty of information about the system's strengths and limitations [26].

Our intention in performing usability tests was to validate our approach to the user interface design and determine if users were able to complete with success a series of test scenarios. For that reason,

the efficiency with which the tasks was performed were not the focus of our attention at this stage, and the number of selected users for the usability tests was reduced.

We devised three test scenarios representing typical planning sessions concerning outdoor sports and activities. Each scenario is composed of a sequence of tasks which eventually leads the user to the desired outcome. At the beginning of each scenario a small description about the goals to achieve is presented, as well as some key concepts required for performing the planning activity. Besides, small tips are provided at each task concerning necessary operations, required options or recommended procedures.

User interaction with the system while performing the proposed scenarios was carried out and each task was evaluated according to the following measuring scale:

1. the user completed the task autonomously, without interacting with the evaluator;
2. the user completed the task while asking some questions to the evaluator;
3. the user completed the task with the help of the evaluator;
4. the user could not complete the task, even with the help of the evaluator.

User selection was made taking into consideration the following concerns. On the one hand, we have decided to involve five users in the usability tests, as this number is generally adequate and sufficient to allow meaningful results when identifying issues with user interface design [27]. On the other hand, users with different degrees of expertise were considered, namely beginners, intermediate and advanced, ranging from users without experience with applications for visualising and analysing GPS-enabled activities (advanced) [28].

In the next paragraphs we present the three usability test scenarios, describe their goals and tasks and mention the functionalities at evaluation. The scenarios comprehend the three main planning approaches, namely: 1) making adjustments to a past activity; 2) creating an activity from scratch; 3) combining past activities to achieve a new activity. Furthermore, the scenarios are ordered by increasing level of complexity and difficulty.

5.2.1 Scenario 1 - Planning an activity based on a past activity

In this scenario, the user must prepare a course to follow during a bike ride based on the performance of a past activity. In particular, it is intended that the planned course has a reference pace which is 5% faster than the previous performance, a track that may be efficiently displayed and a straightforward name for easier identification.

This scenario encompasses the following tasks:

1. Opening the past activity, located at `/Users/ga/Desktop/2014-03-23T18-44-47Z.fit`;
2. Converting the activity to a course;
3. Renaming the course to `Ride in Monsanto`;

4. Removing the course's pauses;

5. Simplifying the track of the course to contain a maximum of 1000 track points;

6. Setting the pace for the course using the speed percentage method, with a value of 95% for the speed percentage setting;

7. exporting the course to a GPX file: `/Users/ga/Desktop/RideInMonsanto.gpx`;

With this scenario we targeted the following operations: 1) opening activities from files in the FIT file format; 2) creating a course from a loaded activity; 3) renaming a course; 4) removing pauses; 5) simplifying tracks; 6) setting the pace of a course with the speed percentage method; 7) saving a course to a file in the GPX file format.

5.2.2 Scenario 2 - Planning an activity from scratch

In this scenario, the user must prepare from scratch a course to follow during a bike ride in an unknown territory. The ride should be approximately 10 km long with a reference pace based on the history of previous performances. As the course's track is unknown to the user, climbs must be annotated to allow early preparation for the terrain difficulties and a reference point must be annotated at the middle of the ride (5 km).

This scenario encompasses the following tasks:

1. Creating a document and adding a course;

2. Creating the track of the course using the map edit mode and the `follow roads` routing option;

3. Detecting and annotating climbs;

4. Creating a course point at the half-way point;

5. exporting the course to a FIT file: `/Users/ga/Desktop/NewRide.fit`;

With this scenario we targeted the following operations: 1) creating a course from scratch with the map edit mode and routing options; 2) automatically calculating grade, and detecting and classifying climbs; 3) annotating climbs on the course; 4) creating waypoints in map view; 5) saving a course to a file in the FIT file format.

5.2.3 Scenario 3 - Planning an activity by combining past activities

In this scenario, the user must prepare a course to follow during a bike ride. The track of the course must be a combination of tracks from previously performed activities.

This scenario encompasses the following tasks:

1. Opening the first activity, located at `/Users/ga/Desktop/a.fit`;

2. Importing the second activity, located at `/Users/ga/Desktop/b.fit`;

3. Converting activity `a` to a course;

4. Converting activity `b` to a course;

5. Splitting both courses at the intersection point;

6. Preserving the first part of course `a` and deleting its second half;

7. Preserving the second part of course `b` and deleting its first half;

8. Joining the first part of course `a` with the second part of course `b`;

9. exporting the course to a GPX file: `/Users/ga/Desktop/ab.tcx`;

With this scenario we targeted the following operations: 1) opening and importing activities from files in the FIT file format; 2) creating courses from loaded activities; 3) splitting courses; 4) joining courses; 5) saving a course to a file in the TCX file format.

5.2.4 Usability Tests Results

Usability tests were conducted in a 15-inch MacBook Pro with a 2.4 GHz Intel Core Duo processor, 2 GB 667 MHz DDR2 SDRAM of memory and a NVIDIA GeForce 8600M GT 256 MB graphics card. This machine is running the OS X 10.9.2 operating system (Mavericks) and has the Java(TM) SE Runtime Environment (build 1.7.0_10-b18) installed. The build 0.0.11 of Track It! was used during this tests.

Table 5.1 summarises the results obtained with the performed usability tests.

There are two important conclusions to draw from the results obtained. On the one hand, all users were able to execute with success the tasks for the three proposed scenarios, although some of them needed advice from the evaluator, and in a few cases required extra help to conclude the scenario, especially users of the beginner's category. Nevertheless, it is often the case that some help is needed (e.g., information retrieved from the application's help system) whenever the subject is complex, the application is full-featured or the user is unfamiliar with the tasks at hand. In fact, some of these factors were reported informally by the users as they performed the test. It is also worthy to note that the task 7 from scenario 3 was consistently evaluated with 3, which is often an indicator of some design issue. Nonetheless, further inspection led us to conclude that the complexity of the join operation derived from the large number of segments resulting from splitted courses, which confuses the user as he decides what segments he must consider and those he must exclude.

On the other hand, it is interesting to note the learning effect. An user who struggled to convert an activity to a course in the first scenario showed almost no difficulty performing the same operation in the second scenario. This is a good indicator that once learned the application becomes easier to use.

It is also worthy to note that timing information was not considered in these tests as we were only interested in evaluating if users were able to successfully complete the proposed tasks and not how fast or efficiently those tasks were performed.

Table 5.1: Summary of the results obtained with the usability tests. Tests were conducted with five users. Tasks were evaluated with the following scale: 1. the user completed the task autonomously, without interacting with the evaluator; 2. the user completed the task while asking some questions to the evaluator; 3. the user completed the task with the help of the evaluator; 4. the user could not complete the task, even with the help of the evaluator.

Scenario / Task	User 1 (Intermediate)	User 2 (Beginner)	User 3 (Beginner)	User 4 (Intermediate)	User 5 (Advanced)
Scenario 1					
Task 1: open activity	1	2	1	1	1
Task 2: convert to course	2	3	3	2	2
Task 3: rename course	1	3	2	2	1
Task 4: remove pauses	2	2	3	3	1
Task 5: simplify track	1	3	2	2	1
Task 6: set the pace	3	3	3	3	1
Task 7: export course	2	2	2	2	2
Scenario 2					
Task 1: new course	2	3	2	2	2
Task 2: edit track	2	3	3	3	2
Task 3: annotate climbs	3	3	3	2	2
Task 4: create course point	2	3	3	2	2
Task 5: export course	1	2	1	1	1
Scenario 3					
Task 1: open activity	1	1	1	1	1
Task 2: import activity	1	2	2	1	1
Task 3: convert to course	1	2	1	1	1
Task 4: convert to course	1	1	1	1	1
Task 5: split course	2	3	3	2	2
Task 6: delete course	2	3	3	2	1
Task 7: delete course	1	1	1	1	1
Task 8: join courses	3	3	3	3	3
Task 9: export course	1	1	1	1	1

5.3 Algorithms Tests

In this section we present some of the tests we conducted in order to validate the correctness and accuracy of the core algorithms included in the proposed solution. In particular, we tested the climb detection and classification algorithms, as well as the track simplification algorithm.

5.3.1 Climb Detection and Classification Algorithms

In order to validate the climb detection and classification algorithms we decided to take the following approach: 1) select road races from the cycling world tour calendar; 2) collect GPS data for those races from one of the many on-line databases; 3) apply the algorithms to the collected data; 4) confront the results with the information provided in the official road book.

Eligible road races should provide an official road book with detailed information about climbs, and the set of chosen races should cover a wide range of terrain profiles and conditions. Thus, we selected two mountain stages from the 100th edition of the Tour de France and the 2014 Liège-Bastogne-Liège race.

One of the advantages of the proposed detection algorithm is its ability to adapt to different purposes through parameter settings. Table 5.2 shows the selected parameters used for the detection algorithm in

these tests. The selection was made empirically, with the presented parameters yielding the best results for pro tour road races.

Climb and descent grade limit parameters represent the grade threshold or minimum grade above/-below which a segment may be considered a climb or a descent, respectively. In this particular setting, a climb starts at 1.9% grade and a descent comprehends grades of -3.0% or lower. Minimum climb and descent percentage parameters specify the minimum amount of ascending/descending terrain a segment must contain (according to the previously defined parameters) in order to be classified as climb or descent, respectively. For instance, according to table 5.2, 90% of the distance of a climb segment must have a grade of 1.9% or higher. This setting allows some control on how multi-part climbs are evaluated. Finally, the climb fine-tuning parameter indicates if some adjustments may be performed over the initial or final portions of a climb or descent segment. For example, if fine-tuning is enabled, a climb segment finish may be pushed forward while grade is monotonously increasing.

Table 5.2: Parameters for the climb and descent detection algorithm.

Parameter	Value
Climb lower grade limit	1.9%
Descent lower grade limit	-3.0%
Minimum climb percentage	90%
Minimum descent percentage	90%
Climb fine-tuning	disabled

The classification algorithm settings were drawn from the Tour de France climb ratings history (from the 1999 to the 2007 edition), as discussed in Chapter 4.

5.3.2 Tour de France 2013 19th Stage

The first road race we selected for analysis was the 19th stage, from Bourg-d'Oisans to Le Grand-Bornand, with an extension of 204.5 km. This is a high-mountain stage, featuring two Hors Category climbs, two First Category Climbs and a Second Category climb. This is also a stage most dear to us as it was won by the Portuguese Rui Costa[1].

The data file in the GPX file format was collected and the detection and classification algorithms applied. Climbs were annotated for easier identification. The elevation profile with the identified climbs and its classification is presented in Figure 5.1. The elevation profile with climb competition information from the official road book [29] may be found in Figure 5.2.

From the chart analysis, the first thing to notice is the discrepancy in "heights" between the Track It! profile and the road book profile, for approximately the same horizontal distance. Those differences originate on the altitude scale (vertical scale) used by both sources. In fact, it is common practice to change the vertical scale in order to produce an altitude exaggeration effect and better convey the terrain difficulties.

It is also evident that the application detected more climbs than those contained in the road book. This can be easily explained by the fact that only climbs that contribute to the king of the mountains'

[1] Palmarés, `http://rui-costa.com`

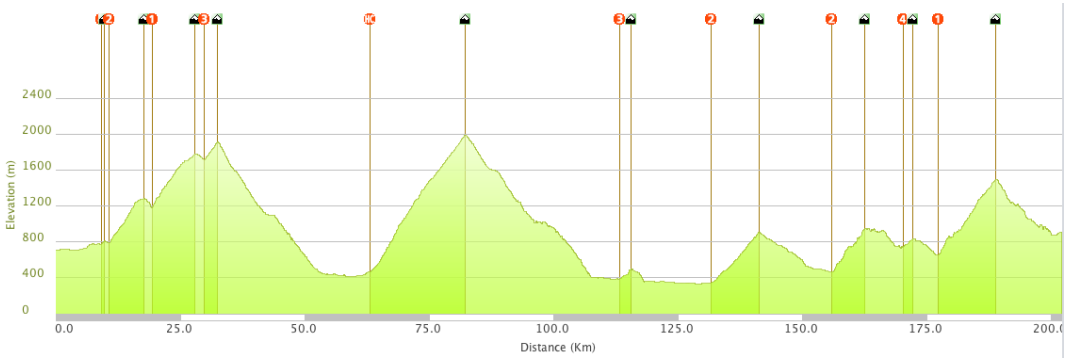

Figure 5.1: Track It! elevation profile for the 19th stage of the 100th Tour de France, with climbs identified and classified.

Figure 5.2: Stage 19 elevation profile and categorised climbs from 100th Tour de France official road book.

competition are categorised, which accounts to the difference in the number of climbs in both charts.

Table 5.3 presents summary information for the climbs included in both sources for comparison.

The presented data clearly shows that Track It!'s climb detection algorithm exhibits very accurate results concerning climb length and average grade when compared to the road book's climb information.

Official climbs 2, 3, 4, and 5 were correctly detected by the algorithm and although they are somewhat longer than those presented in the road book, length differences do not exceed 1.0 km and average grade is within 0.5% of the official value. These slightly longer climbs may be explained by the fact that start and finishing points are often adjusted by race organisers in order to accommodate race logistics, satisfy sponsorships and enhance competition. Moreover, the increased length does not contribute to a significant change in the average grade of the climb (less than 0.5%), nor does it result in a lower climb category. All these four climbs were awarded the same category as the official ones.

The only significant difference concerns the detection of climb 1. The official book presents a 21.6 km climb at 5.1% but the algorithm detected 3 climbs of different categories, interleaved by rest segments of 1.5 km and 2.0 km respectively. Experimental data have shown that setting the climb percentage

Table 5.3: Summary of the results obtained by comparing the official Tour de France 2013 information for stage 19 categorised climbs with the climbs detected and classified by Track It!

Climb Nr.	TDF 2013 Road Book	Track It!
–		Length: 0.7 km Average Grade: 6.5% Fourth Category
1.	Length: 21.6 km Average Grade: 5.1% Hors Category	Length: 6.9 km Average Grade: 7.2% Second Category
		Length: 8.6 km Average Grade: 7.1% First Category
		Length: 2.6 km Average Grade: 7.4% Third Category
2.	Length: 19.2 km Average Grade: 7.9% Hors Category	Length: 18.9 km Average Grade: 8.0% Hors Category
–		Length: 2.3 km Average Grade: 4.9% Fourth Category
3.	Length: 8.6 km Average Grade: 6.2% Second Category	Length: 9.5 km Average Grade: 5.9% Second Category
4.	Length: 6.1 km Average Grade: 7.3% First Category	Length: 6.6 km Average Grade: 7.2% First Category
–		Length: 1.9 km Average Grade: 4.8% Fourth Category
5.	Length: 11.3 km Average Grade: 7.0% First Category	Length: 11.4 km Average Grade: 7.4% First Category

parameter to 90% yielded the best results across test cases where professional races are involved. In fact, race organisers tend to favour multiple climbs at steeper grades instead of a longer climb at a lower grade, including “rest” periods. This particular case is an exception to the rule and lowering the parameter to as much as 80% would correctly identify the multi-part climb.

It is also worth mentioning that the algorithm detected three additional climbs not present in the road book. All of them are Fourth Category climbs, one a steep 700 m segment at 6.5% and the other two approximately 2 km climbs at 5%. These climbs were correctly identified and categorised but were not considered in the race mainly because enough points have been awarded to the remaining climbs, and since they are of little significance when compared to the categorised climbs or the overall stage difficulty.

5.3.3 Tour de France 2013 20th Stage

We also analysed the algorithm’s results for stage 20 of the same Tour de France, a 125 km long stage from from Annecy to Annecy-Semnoz. This is also considered a mountain stage but somewhat different in nature from the previous one. There is a mix of climbs from different lengths, average grades and categories, including 3 Third Category, one Second Category, one First Category and one Hors

Category. The application's elevation profile for the detected and categorised climbs is presented in Figure 5.3. The elevation profile with climb competition information from the official road book [29] may be found in Figure 5.4.

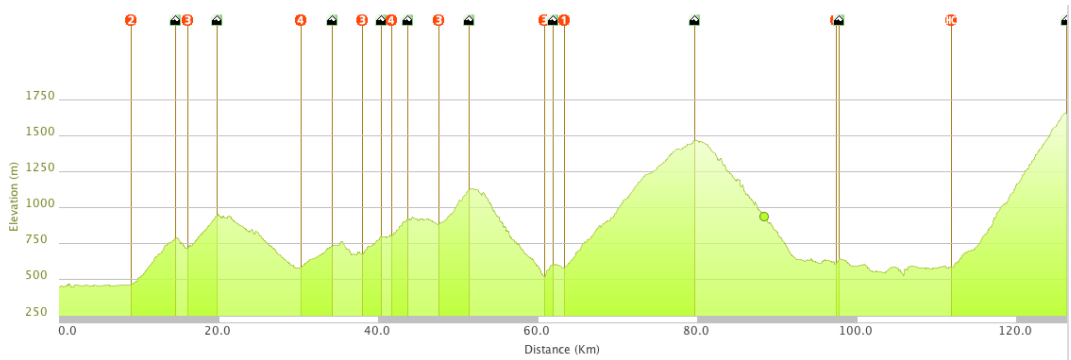

Figure 5.3: Track It! elevation profile for the 20th stage of the 100th Tour de France, with categorised climbs information.

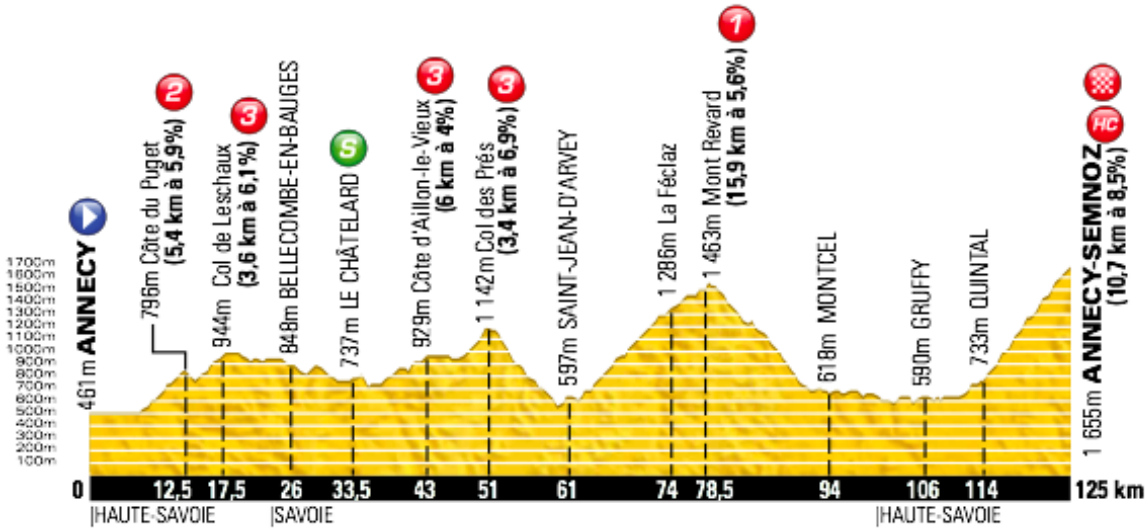

Figure 5.4: Stage 20 elevation profile and categorised climbs from the 100th Tour de France road book.

Table 5.4 presents summary information for climbs in both sources for comparison.

Similar results were found for this stage analysis when compared with those from stage 19. Four out of the six climbs detected were spot on both on length and average grade (climbs 1, 2, 4, 5). For this segments, length difference remained below 300 m and average grade did not differ more than 0.5%.

Once more, a multi-part climb 3 was not considered but two smaller climbs were detected instead. In this case, there is a 1.5 km flat segment in-between two easier climbs when compared to the remaining ones from the stage, fact that makes this longer climb less probable. To further emphasise the unlikeness of this choice from the organisation, in order to the multi-part climb to be detected it would require to lower the climb percentage parameter to 75%, value that would "merge" climbs 1 and 2, and would make climbs 5 and 6 much longer. Our hypothesis is that the position of the climb in the first third of the stage and the great amount of points at stake awarded by the other climbs justifies the organisers decision to

Table 5.4: Summary of the results obtained by comparing the official Tour de France 2013 information for stage 20 categorised climbs with the climbs detected and classified by Track It!

Climb Nr.	TDF 2013 Road Book	Track It!
1.	Length: 5.4 km Average Grade: 5.9% Second Category	Length: 5.4 km Average Grade: 5.9% Second Category
2.	Length: 3.6 km Average Grade: 6.1% Third Category	Length: 3.7 km Average Grade: 6.6% Third Category
–		Length: 3.9 km Average Grade: 3.9% Fourth Category
3.	Length: 6.0 km Average Grade: 4.0% Third Category	Length: 2.2 km Average Grade: 5.7% Third Category
		Length: 1.9 km Average Grade: 6.2% Fourth Category
4.	Length: 3.4 km Average Grade: 6.9% Third Category	Length: 3.7 km Average Grade: 6.4% Third Category
–		Length: 1.0 km Average Grade: 7.4% Third Category
5.	Length: 15.9 km Average Grade: 5.6% First Category	Length: 16.2 km Average Grade: 5.5% First Category
–		Length: 0.4 km Average Grade: 8.8% Fourth Category
6.	Length: 10.7 km Average Grade: 8.5% Hors Category	Length: 14.3 km Average Grade: 7.4% Hors Category

consider only one climb.

The case of climb 6 being shorter than the detected one also seems to be an administrative decision, taking into consideration that the average grade is only 1% lower, the category is the same and the excluded length contains segments at an average grade over 12%! Climbs at the end of the stage are frequently awarded twice the points for the climb category and there is often time bonuses at the finish. Moreover, the last part of the race is seen by millions of people around the world and is heavily commentated. Thus, we think that race organisers once more decided to make the climb shorter at a steeper grade in order to emphasise its difficulty and contribute to the illusion of a more dramatic finish.

The three additional climbs detected were not categorised in the race book probably by the same reasons presented before, that is, points in excess for the best climber's competition and for being relatively short and low category climbs when considering the overall stage profile.

With the exception of multi-part climb 3, all of the detected climbs matched the category from the road book.

5.3.4 Liège-Bastogne-Liège 2014

The last race we selected to test the climb detection and classification algorithms is the one day Spring classic Liège-Bastogne-Liège, one of the oldest and longest in the professional road cycling calendar. This race is substantially different from the two previous ones. On the one hand it is much longer, including more than 263 km. On the other hand, it features a great number of short, sharp, steep and low category climbs. The race profile obtained from the official website[2] is presented in Figure 5.5.

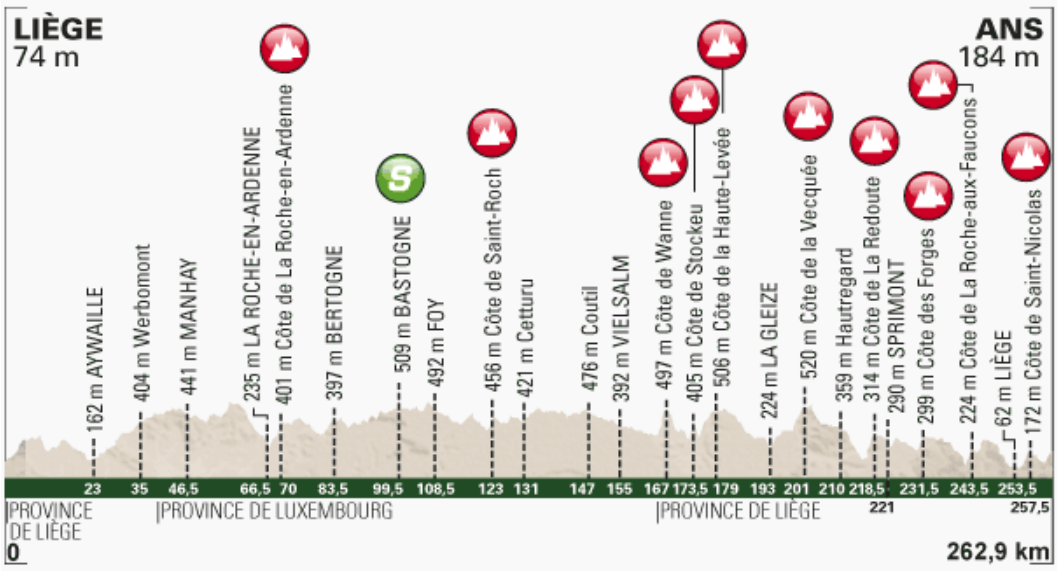

Figure 5.5: Liège-Bastogne-Liège elevation profile and categorised climbs for the 2014 edition.

Table 5.5 presents summary information for climbs in both sources for comparison.

The results for this test show that the detection algorithm performs well whenever longer races and short steep climbs are considered. Seven of the detected climbs were close matches, falling within the 1.0 km and 0.6% margins for both length and average grade, respectively. The remaining three of the "categorised" climbs were also detected and their length is within the expected values from the road book, but the calculated average grade is significantly lower than the officially stated. These climbs have in common the fact that they are very short and have a rather steep average grade (from 8.6% to 11.1%). Our hypothesis is that the observed differences may be attributed to one of the following causes: 1) miscalculated average grades from the race organisation; we have noticed that the value presented as the average grade was found to be the calculated maximum grade by the application; moreover, occasional grade overstatements have been reported in the past and the altitude difference does not account for such high average grades; 2) GPS data errors; data was downloaded from an on-line database and additional information about its quality is not available; furthermore, data was collected by an amateur cyclist with an unknown device.

Another remarkable difference from the previous tests is the number of non categorised climbs. In particular, seven climbs were detected but not included in the official road book. We thing that this a reasonable decision from the race organisation as too many climbs would not provide useful information

[2]Liège-Bastogne-Liège Official Website, `http://www.letour.fr/liege-bastogne-liege/2014/us/race.html`

Table 5.5: Summary of the results obtained by comparing the information from the Liège-Bastogne-Liège official website with the climbs detected and classified by Track It!

Climb Nr.	Liège-Bastogne-Liège Official Info	Track It!
–		Length: 1.2 km Average Grade: 5.0% Fourth Category
–		Length: 3.0 km Average Grade: 4.1% Fourth Category
–		Length: 2.8 km Average Grade: 4.1% Fourth Category
–		Length: 1.8 km Average Grade: 4.8% Fourth Category
1.	Length: 2.8 km Average Grade: 6.2% n/a	Length: 2.4 km Average Grade: 6.5% Third Category
2.	Length: 1.0 km Average Grade: 11.1% n/a	Length: 1.0 km Average Grade: 5.3% Third Category
–		Length: 1.0 km Average Grade: 4.7% Fourth Category
3.	Length: 2.8 km Average Grade: 7.2% n/a	Length: 2.4 km Average Grade: 7.8% Second Category
4.	Length: 1.0 km Average Grade: 12.4% n/a	Length: 1.0 km Average Grade: 11.9% Third Category
5.	Length: 3.6 km Average Grade: 5.6% n/a	Length: 3.6 km Average Grade: 5.7% Third Category
6.	Length: 3.1 km Average Grade: 6.4% n/a	Length: 4.0 km Average Grade: 6.1% Second Category
7.	Length: 2.0 km Average Grade: 8.9% n/a	Length: 2.6 km Average Grade: 5.1% Third Category
–		Length: 1.7 km Average Grade: 6.6% Third Category
8.	Length: 2.0 km Average Grade: 8.9% n/a	Length: 1.9 km Average Grade: 8.8% Third Category
9.	Length: 1.5 km Average Grade: 9.3% n/a	Length: 1.4 km Average Grade: 9.6% Third Category
–		Length: 1.7 km Average Grade: 5.5% Fourth Category
10.	Length: 1.2 km Average Grade: 8.6% n/a	Length: 1.2 km Average Grade: 7.6% Third Category

about the race difficulties nor add value to the competition.

Finally, we would like to emphasise that for one day races climb categorisation is not important and is seldom included in road books. For that reason, climb classification was not the target of the specific test. Nevertheless, the calculated climb categories were presented for reference.

5.3.5 Track Simplification Algorithm

The track simplification algorithm was tested against track shape preservation and higher track point density in climbs when compared to flat terrain and/or descents.

The test case consisted in applying the simplification algorithm three times in succession, and evaluating shape preservation and track point density at each step. The original track contained 6280 track points and featured sharp bends, in order to exercise shape preservation, as well as several variations on the elevation profile, so that the effects on track point density could be better evaluated on different terrain types.

Figure 5.6 shows the effect on the shape of the original track as the number of track points is reduced. From visual inspection, we may conclude that despite the inevitable precision loss resulting from track point removal, as it is evident from the straight lines and angled bends, the algorithm managed to capture the essence of the track's shape even in extreme simplification conditions.

(a) 3003 track points **(b)** 996 track points **(c)** 204 track points

Figure 5.6: Effects on track shape from applying the simplification algorithm three times in succession. The original track is presented in blue while the simplified track is depicted in yellow.

For the track point density test, a segment of each of the existing types was selected, and track point density was compared between the three simplification conditions. Table 5.6 summarises the obtained results.

The results show that track point density is higher on climb segments when compared to flat or descent segments. This demonstrates that the simplification algorithm is more conservative whenever climb segments are concerned, as expected. We have also noticed that in extreme conditions track shape on descents tends to be oversimplified. Some sort of mechanism should be implemented in the future so that, under extreme simplification conditions, the density preservation requirement on climbs may be relaxed.

The presented results also show that distance is only slightly affected by track simplification, even

Table 5.6: Effects on track point density and distance as a function of the number track points and segment type. Density is expressed in track points per meter.

Track points Nr.	Distance km	Overall Density (tp/m)	Climb Density (tp/m)	Descent Density (tp/m)
6280	30.765	.20330	.02905	.12512
3003	30.719	.09821	.24382	.03815
996	30.625	.03281	.09919	.02487
204	30.340	.00674	.01590	.00837

when the number of points removed is high. We suspect that preserving track shape, specially on twisting segments, contributes to this attenuation on the effects of simplification on the overall track distance.

It is worthy to note that the simplification algorithm was parametrised and tested having road cycling as reference. In order for the algorithm to be useful for other sports, different parameters would have to be applied. For instance, in giant slalom, an alpine skiing discipline, achieving higher track point density on descents would be desirable.

5.4 Conclusion

In this chapter we presented the proposed solution evaluation. We conducted usability tests by observing users in action, in order to measure the completeness, effectiveness and ease of use of the application. We also validated some of the application's core algorithms against other sources of information with respect to correctness and accuracy, specifically the climb detection, climb classification and track simplification algorithms. Results have shown that the implemented algorithms meet the stated requirements and produce very accurate results.

6

Conclusions and Future Work

Contents

6.1 Conclusions 78
6.2 Suggestions for Future Developments 80

6.1 Conclusions

Nowadays, GPS devices are ubiquitous and widely used in outdoor sports and activities, capturing huge amounts of positioning and fitness data. On the other hand, there are plenty GPS-enabled applications that allow users to get the most out of the collected data. In particular, it is possible to visualise and analyse performance data, use GPS data to aid navigation, perform real time tracking or manipulate data in various ways, for instance to create custom maps or assemble track networks. But, as our survey on state of the of art GPS-enabled applications has shown, applications often lack planning capabilities and the set of available operations for planning sessions is frequently disperse, limited, inadequate or incomplete. Actually, none of the surveyed applications shows support for all the features we consider essential for a planning session, despite all of them providing planning features.

In order to bridge this gap, we have created a dedicated GPS-enabled application which provides planning capabilities for outdoor sports and activities. We have also demonstrated that this application is not only useful for planning a running session or a leisurely bike ride but it is also accurate, versatile and extensible to fulfil every user's needs. Notwithstanding planning being our primary goal, tools for visualising and analysing GPS data were also included.

Several requirements were taken into consideration when conceiving the application.

Firstly, the proposed solution was designed to be multi-platform, flexible, customisable, extensible, small and to support off-line operation. This way, we have ensured that a planning session may be accomplished in a wide range of situations or circumstances. There are also several views available to provide the best perspective for the task at hand, all the time.

Secondly, the application supports a myriad of editing tools for manipulating track points, course points, laps and segments, besides providing flexibility in ways courses may be created. Specifically, the user may choose:

1. to create a course from scratch, either by using routing support or designing the route with the aid of the maps provided;
2. to modify a performed activity in order to meet new requirements; or
3. to create an entirely new course by combining several courses and activities, namely by joining and splitting tracks and/or track segments.

Thirdly, special attention was devoted to mapping support, specifically by providing a great variety of map providers and types, and by allowing flexibility and extensibility in order to support additional maps in the future. Moreover, we also achieved seamless integration of the Portuguese Military Survey Maps, a specially useful tool whenever off-road activities or activities in remote places are concerned. The list of supported map providers includes 1) Google Maps; 2) Bing Maps; 3) Here Maps; 4) MapQuest Maps; 5) OpenStreet Maps; and 6) Portuguese Military Maps. Maps may also be displayed with the following types 1) map; 2) satellite; 3) hybrid (satellite and map combined); or 4) terrain or topographic.

Finally, we provide a collection of unique operations which deliver useful information while performing an activity. In particular, we would like to emphasise the following operations:

- **Climb detection, classification and annotation**: this operation automatically detects climbs based on the terrain profile, classifies them according to the specified criteria and annotates climbs start/finish points and maximum/minimum grades on courses to provide useful information while performing an activity, such as the estimated time to reach climb base, climb difficulty, or the distance to the summit; this operation also includes an improved grade calculation algorithm and supports special features such as multi-part climbs or extending climbs when maximum/minimum locals are reached;

- **Pacing**: this operation allows the user to set the pace to follow during an activity; pace may be specified as a percentage of a previous pace, manipulated to match a target time or speed, or automatically adjusted to a variable pace based on the terrain's profile and the user's past history of activities; the operation also applies a smoothing filter to the calculated speed in order to remove spikes and produce smoother transitions between track segments;

- **Pauses removal**: this operation allows the user to remove pauses from a course, providing more accurate predictions regarding the activity's moving time; a smoothing filter is performed over the horizontal speed before the stop detection algorithm is applied so that outliers may be removed and spurious pauses ignored; the speed threshold may be specified to suit the user's needs;

- **Track simplification**: this operation simplifies the track of a course in order to improve performance without sacrificing the quality of the displayed data; it is worthy to note the ability of the algorithm to preserve the track's shape and to adjust track points density as a function of grade.

Furthermore, an extensibility mechanism was included, enabling the development and integration of new operations as needed.

The proposed application was subject to usability tests in order to verify the effectiveness of the available operations and detect problems for future development. Results have shown that even inexperienced users are able to execute common planning tasks, provided some help is available on the operations purpose and/or settings.

Some of the core operations were also evaluated regarding their effectiveness, accuracy and performance, and the achieved results were markedly positive. Some of the most important conclusions are summarised below, while detailed information about test results may be reviewed in Chapter 5.

- Climb detection algorithm: tests with professional cycling races' data have shown that the algorithm exhibits very accurate results concerning climb length and average grade when compared to the official information about the climbs; on all of the performed tests climb length was within the 1.0 km margin while the average grade did not fluctuate more than 0.6%; small variations on climb starting points or multi-part climb detection were justified based on race organisation decisions and support for those hypothesis was provided; the provided parameters also proved to be sufficient in supporting the algorithm's flexibility;

- Climb classification algorithm: all of the tested climbs were successfully categorised according to official road books, which not only demonstrates that using statistical data on climb categorisation

from previous races is an adequate predictor of the climb's difficulty, but also that the algorithm is flexible enough to adapt to specific categorisation requirements;

- Track simplification algorithm: test results demonstrated that the shape of a track is preserved during the simplification process, and that higher track point density is maintained in climb segments when compared to flat or descent segments; extreme simplification was employed to verify the algorithm's correctness and suggestions for future development were proposed, specifically on adapting the algorithm to extreme simplification conditions.

6.2 Suggestions for Future Developments

Despite the fact that the presented solution in this dissertation is fully functional and supports typical planning sessions' tasks, there are is still plenty of opportunities for improvements in the future. Specifically, we propose the following enhancements:

- the user interface could be more flexible by allowing repositioning views, easy view closing, full-screen support, or undoing operations; the user interface could also be simplified and streamlined in order to improve user experience and increase productivity;
- operations should be also applicable to segments and not exclusively to whole courses;
- profiles could be considered to provide customisation on the available operations, and to allow adapting the application's behaviour as a function of the selected sport/activity;
- database support could be implemented, allowing the maintenance of a catalogue of activities and courses as well as providing searching capabilities;
- some visualisation enhancements could be included allowing performance comparisons on tracks and segments by the same user and/or different users; side by side comparisons should be provided for chart and data views;
- social support could be included, allowing the user to share and search for courses and activities, or to challenge other users on a specific course.
- the application could offer support to attach multimedia in order to enrich courses and activities.

References

[1] J. M. B. Lopes, “Interacting with outdoor sports and performance data,” in **SIACG 2011, V Simpósio Ibero-Americano de Computação Gráfica**, June 2011.

[2] M. A. F. da Costa, “Sistema de apoio à mobilidade de utilizadores da bicicleta,” Master’s thesis, Universidade do Porto, Faculdade de Engenharia, Porto, 2011.

[3] CycleOps, “CycleOps: Products, ANT+ wireless power meters.” [Online]. Available: http://www.cycleops.com/en/products/power-meters.html

[4] Tanita, “Tanita: ANT+ wireless produts.” [Online]. Available: http://www.tanita.com/en/faq/wireless-products/

[5] Wahoo, “Wahoo fitness: ANT+ wireless sensors.” [Online]. Available: http://www.wahoofitness.com/Products/Products/Sensors-130-CL.aspx

[6] ANT+, “This is ANT: The wireless sensor network solution.” [Online]. Available: http://www.thisisant.com

[7] Microsoft, “Bing maps.” [Online]. Available: http://www.bing.com/maps

[8] Google, “Google maps.” [Online]. Available: http://www.google.com/maps

[9] B. R. Toaster, “Bike route toaster.” [Online]. Available: http://www.bikeroutetoaster.com/

[10] MapMyRide, “Map my ride: Cycling social network.” [Online]. Available: http://www.mapmyride.com/

[11] GPSies, “GPSies — GPS, tracks, trails, tours, converter.” [Online]. Available: http://www.gpsies.com/

[12] D. H. Douglas and T. K. Peucker, “Algorithms for the reduction of the number of points required to represent a digitized line for its caricature,” **Cartographica: The International Journal for Geographic Information and Geovisualization**, vol. 10, no. 2, pp. 112–122, 1973.

[13] Strava, “Strava — running and cycling gps tracker, performance analytics, maps, clubs and competition.” [Online]. Available: http://www.strava.com/

[14] S. Tracks, “Sport tracks: GPS-integrated workout logging software for windows PC.” [Online]. Available: http://www.zonefivesoftware.com/sporttracks/

[15] Adze, “Use adze to edit gps data like a champ.” [Online]. Available: http://www.http://kobotsw.com/

[16] M. Fowler, **Patterns of Enterprise Application Architecture**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[17] Oracle, “Java se 7 features and enhancements: Release notes.” [Online]. Available: http://www.oracle.com/technetwork/java/javase/jdk7-relnotes-418459.html

[18] “gnu general public license,” free software foundation. [Online]. Available: http://www.gnu.org/licenses/gpl.html

[19] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, **Design Patterns: Elements of Reusable Object-oriented Software**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995.

[20] MyTourBook, “Smoothing out gps and heart rate monitor data.” [Online]. Available: http://mytourbook.sourceforge.net/mytourbook/files/mytourbook_smooth_11.8.2.pdf

[21] I. G. do Exército, **Noções Gerais de Geodesia**. Instituto Geográfico do Exército, Lisboa, 2000.

[22] ——, **Sistemas de Referenciação**. Instituto Geográfico do Exército, Lisboa, 2004.

[23] P. S. Heckbert and M. Garland, “Survey of polygonal surface simplification algorithms,” 1995.

[24] G. Babel, “Gps babel: Free software for gps data conversion and transfer.” [Online]. Available: http://www.gpsbabel.org/

[25] H. Sharp, Y. Rogers, and J. Preece, **Interaction Design: Beyond Human-Computer Interaction**, 2nd ed. Wiley, Mar. 2007.

[26] A. Dix, J. Finlay, G. Abowd, and R. Beale, **Human-computer Interaction**. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1997.

[27] J. Nielsen and T. K. Landauer, “A mathematical model of the finding of usability problems,” in **Proceedings of the INTERACT ’93 and CHI ’93 Conference on Human Factors in Computing Systems**, ser. CHI ’93. New York, NY, USA: ACM, 1993, pp. 206–213. [Online]. Available: http://doi.acm.org/10.1145/169059.169166

[28] M. J. Fonseca, P. Campos, and D. Gonçalves, **Introdução ao Design de Interfaces**, tecnologias de informação ed., 2012. [Online]. Available: http://www.fca.pt/cgi-bin/fca_main.cgi/?op=2&isbn=978-972-722-738-9

[29] A. S. Organization, **Livre de Route: Entrez dans la légend du Tour (100e Édition)**. Issy-Les-Moulineaux Cedex: A.S.O., 2013.

A

Track It! Domain Model

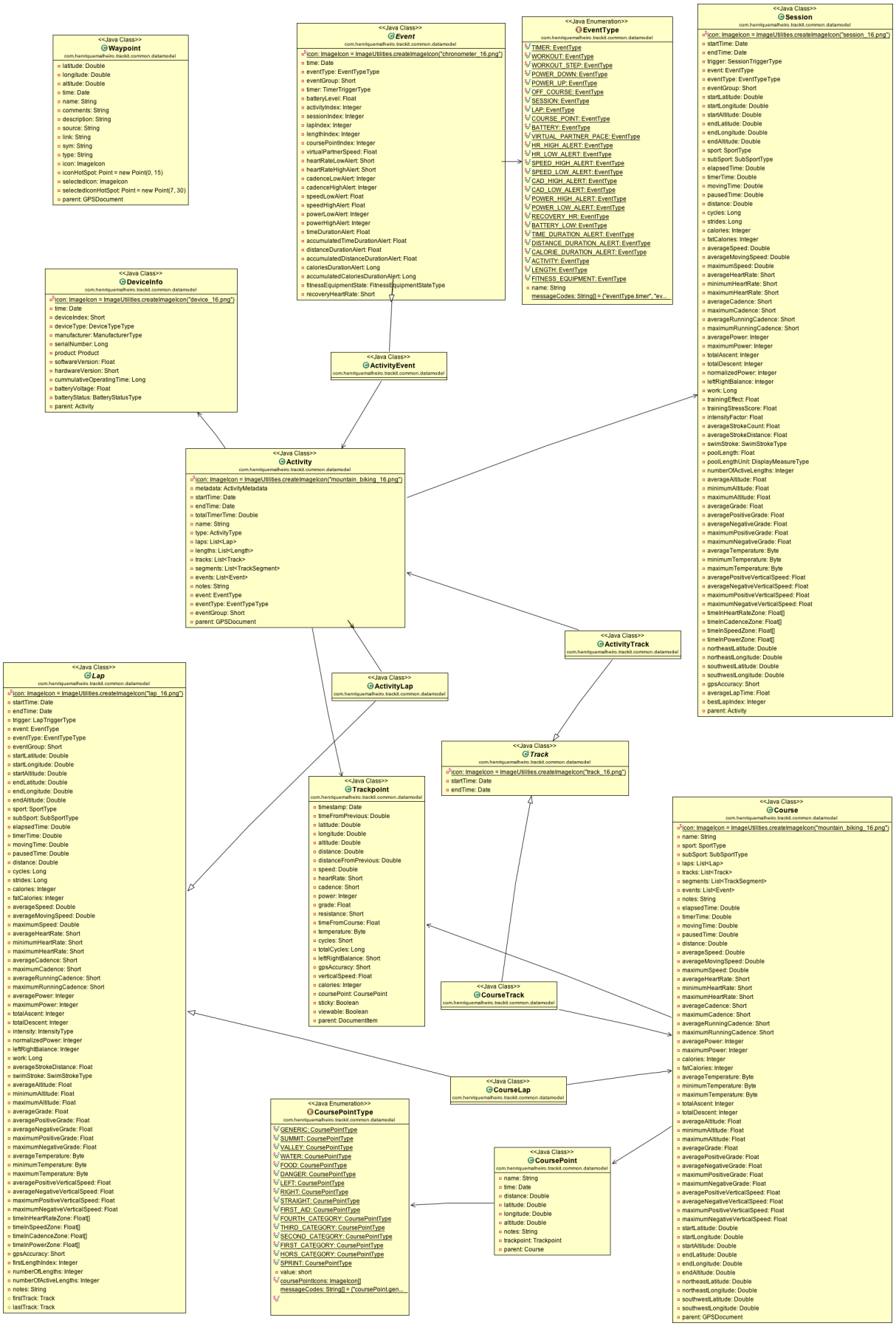

Figure A.1: Track It! main domain model elements, with their attributes and most relevant associations.