Master Thesis

im Rahmen des
Universitätslehrganges „Geographical Information Science & Systems“
(UNIGIS MSc) am Interfakultären Fachbereich für GeoInformatik (Z_GIS)
der Paris Lodron-Universität Salzburg

zum Thema

„Einsatz von Geoinformatik im Triathlon Sport“

leistungsparametrische Modellierung und GeoSimulation der dynamisch segmentierten Radstrecke des Ironman Austria

vorgelegt von

Mario Kollegger

102762, UNIGIS MSc Jahrgang 2013

Zur Erlangung des Grades
„Master of Science (Geographical Information Science & Systems) – MSc(GIS)”

Gutachter:
Ao. Univ. Prof. Dr. Josef Strobl

Klagenfurt, 30.04.2016

Danksagung und Vorwort

Die folgenden Zeilen sind jenen Personen gewidmet, die einen wesentlichen Anteil zur Fertigstellung dieser Arbeit beigetragen haben. Ein sportlicher Dank gilt den 107 Athletinnen und Athleten des Ironman Austria, die durch Bereitstellung ihrer aufgezeichneten GPX Tracks als empirische Felddaten die Forschungsgrundlage dieser Arbeit geschaffen haben.

In zweifacher Weise möchte ich mich bei Peter Kollegger bedanken, der mir als treuer Studienkollege viele GIS spezifische Inputs lieferte und gleichzeitig als mein Vater für moralische Unterstützung sorgte. Ebenso sehr erwähnenswert ist mein Bruder Fabian, der mit seiner enorm hilfreichen Fachexpertise ein separates Kapitel zur Danksagung verdient hätte. Für kontinuierliche Motivation sorgten zudem meine Mutter Silvia und meine Oma Ingeborg Frey-Schlegl, die mit großem Interesse den Prozess zur Fertigstellung dieser Arbeit verfolgten und denen ich an dieser Stelle meinen Dank aussprechen möchte.

Für eine perfekte Unterstützung während des Studiums sorgte das UNIGIS Lehrgangsteam, welches ein erkenntnis- und zugleich abwechslungsreiches Studienangebot zusammengestellt hat. Ein besonderer Dank geht an Prof. Dr. Strobl, der es mir ermöglichte, mich einem Thema widmen zu können, das losgelöst aus dem beruflichen Kontext seit Jahren von persönlichem Interesse ist und in das viel Zeit und Mühe investiert wurde.

Erklärung der eigenständigen Abfassung der Arbeit

"Ich versichere, diese Master Thesis ohne fremde Hilfe und ohne Verwendung anderer als der angeführten Quellen angefertigt zu haben, und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat. Alle Ausführungen der Arbeit die wörtlich oder sinngemäß übernommen wurden sind entsprechend gekennzeichnet."

Klagenfurt, 30.04.2016 *eigenhändige Unterschrift*

Ort, Datum

Kurzfassung

Nach sportwissenschaftlichen Erkenntnissen soll die Rennstrategie bei der Raddisziplin von Langdistanz Triathlon Wettkämpfen so gewählt werden, dass ein möglichst gleichmäßiges Tempo über die gesamte Distanz zur optimalsten Leistung führt. Beim Auswahlgebiet des Ironman Austria wird diese Even-Pacing-Strategy jedoch nicht in die Praxis umgesetzt, da die 500 letztplatzierten Athleten in der zweiten Radrunde einen Zeitverlust von durchschnittlich 11%, also 21 Minuten, zur ersten Runde aufweisen. Diese Diskrepanz zwischen sportwissenschaftlicher Theorie und realer Umsetzung im Wettkampf soll durch den Einsatz von GIS-Methoden und -Technologien gelöst werden. Dazu wurden Marschrouten generiert, die auf GPS Systemen aufrufbar sind, wodurch der Sportler eine ständige Anweisung und Rückmeldung zur Einhaltung der Rennstrategie erhält. Als empirische Grundlage konnten im Zuge dieser Arbeit 1,6 Millionen Punktgeometrien als raum-zeitliche Bewegungsdaten eingeholt werden, die durch 107 Athleten während des Wettkampfes aufgezeichnet wurden. Über eine induktive geostatistische Analyse konnte ein Modell erzeugt werden, um je nach gewünschter Zielzeit eine entsprechende Marschroute zu generieren. Unter Verwendung der Methodik zur dynamischen Segmentierung wurden Leistungsparameter ausgeforscht, die einen entscheidenden Einfluss auf die Athleten nehmen. Diese Einflussgrößen wurden mittels der Programmiersprache C# in einem ArcMap Add-in eingearbeitet und können als graphische Benutzeroberfläche, je nach Vorgaben, individuell an die Fähigkeiten des Athleten adaptiert werden. Dieser Workflow zur Modellierung und GeoSimulation generiert schlussendlich einen GPX Track als Marschroute, der am GPS Gerät angewandt werden kann. Die empirischen Felddaten dienen im Zuge dieser Masterarbeit nicht nur zur Bildung einer Hypothese, sondern werden ebenfalls zur Überprüfung der Modellierung und der GeoSimulation herangezogen. Durch Deduktionsschlüsse soll die Beantwortung der interdisziplinären wissenschaftlichen Fragestellung in Form von Zahlenvergleichen zwischen simulierten und realen GPX Dateien überprüft werden.

Abstract

According to sports-scientific findings, the preferred bicycle racing strategy in long-distance triathlon tournaments should be keeping a consistent speed the entire distance in order to achieve the best possible result. However, this so-called even pacing strategy is not actually applied during the Ironman Austria race, which is the object of this thesis. The 500 athletes who came in last showed an average time loss of 11 % during the second round, which is 21 minutes compared to the first round. This thesis seeks to solve this discrepancy between sports-scientific theory and the real practice during races by using GIS methods and technologies. In doing so, routes will be generated that can be accessed with GPS systems to deliver permanent instructions and feedback on maintaining the racing strategy to the athlete. The empirical basis of this thesis are 1.6 million point geometries in the form of spatiotemporal movement data. These were recorded by 107 athletes during tournaments. By means of an inductive geostatic analysis, a model was created to generate an adequate route for each desired target time. Using the methodology of dynamic segmentation, performance parameters with an essential influence on the athletes were identified. These influencing factors were inplemented in an ArcMap add-in using the programming language C#. According to the specific requirements, they can be adapted to the individual skills of the athlete using a graphical user interface. Finally, this workflow for modelling and simulation generates a GPX track as a route that can be loaded on a GPS device. The empirical field data is not only used for developing a hypothesis for this master thesis, but also for monitoring the modelling and geosimulation process. Deductive reasoning will be used to answer the interdisciplinary scientific question by comparing the numbers of simulated and real GPX files.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

Abkürzung	Erläuterung	Kapitel
ALS	Airborne Laserscanning	2.4.3
App	Applikation	4.3.4
ASCII	American Standard Code for Information Interchange	2.4.3
DBF	Data Base File	5.1
DGM	Digitales Geländemodell	2.4.3
DGPS	Differential Global Positioning System	2.4.1
DHM	Digitales Höhenmodell	2.4.3
ebd.	ebenda	
Esri	Environmental Systems Research Institute	2.4.3 & 5.2
FME	Feature Manipulation Engine	3.1.3.3
GIP	Graphenintegrations-Plattform	3.1.2
GNSS	Global Navigation Satellite System	2.4.1
GPS	Global Positioning System	2.4.1
GPX	GPS eXchange Format	2.4.2
GUI	Graphical User Interface	5.1
LiDAR	Light detection and ranging	2.4.3
MVVM	Model View ViewModel	5.2
OGC	Open Geospatial Consortium	2.4.2
ÖK	Österreich Karte	3.1.3.5
OSM	OpenStreetMap	3.1.2
SHP	Shapefile Datenformat	3.3.1 & 3.3.3
TIN	Triangulated Irregular Network	2.4.3
W3C	World Wide Web Consortium	2.4.2
WPF	Windows Presentation Foundation	5.1
XBAP	XAML Browser Applications	5.1
XSD	XML Schema Definition	2.4.2 & 9

1. Einleitung

1.1 Motivation und Relevanz

Auf Grund von mehreren Teilnahmen an Langdistanz Triathlon Bewerben entstand die Motivation, über dieses Themengebiet eine Master Thesis zu verfassen, um durch GIS-Methodiken den Athleten eine Unterstützung im Wettkampf zu ermöglichen.
Der Triathlon Sport im Allgemeinen, somit auch der Ironman Austria im Speziellen, hat im Verlauf der letzten Jahre immer mehr Anklang innerhalb der Gesellschaft gefunden. Dies zeigt etwa die Abbildung 1-1, in der ein Anstieg der Teilnehmerzahlen beim Ironman Austria vom Jahr 1998 mit 124 teilnehmenden Personen bis ins Jahr 2014 mit 3.000 Athleten ersichtlich ist. Seit dem Jahr 2013 stehen 3.000 Startplätze zur Verfügung, die jedoch auf Grund der starken Nachfrage bereits innerhalb von wenigen Stunden nach Öffnung der Anmeldung vergriffen sind (Kleine Zeitung 2013b).

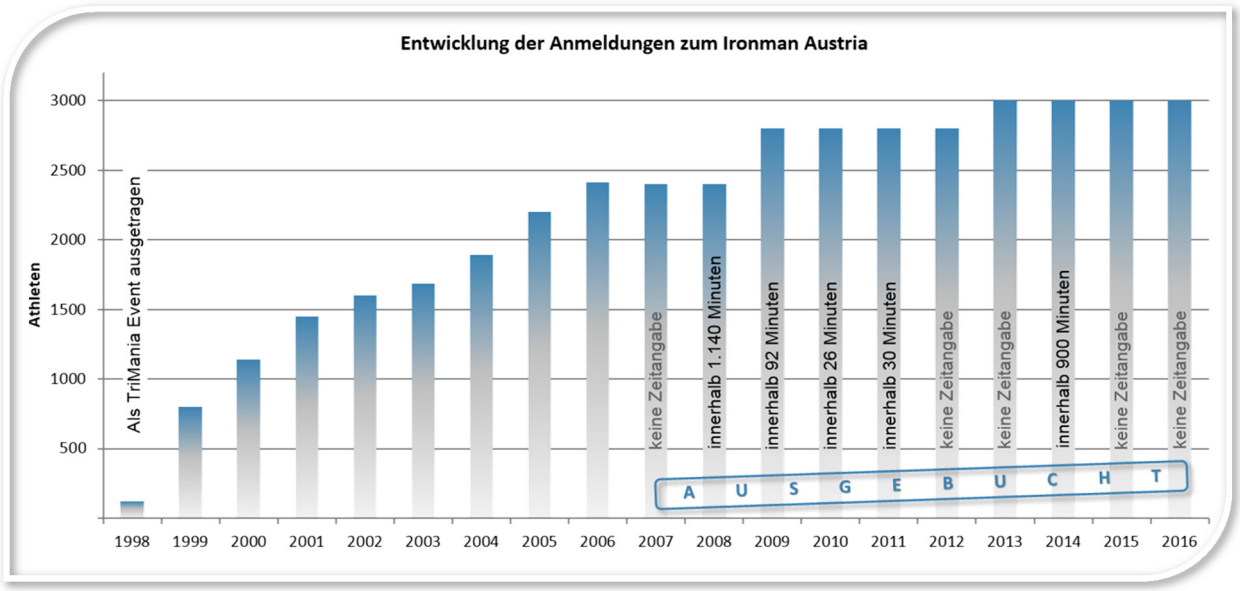


Abbildung 1-1: Entwicklung der Anmeldung zum Ironman Austria
Eigene Bearbeitung. Datenquellen: (Petschnig 2007, S. 20–148) , (Österreichischer Rundfunk 2011), (Kleine Zeitung 2013a) und (Kleine Zeitung 2013b)

Des Weiteren hat der Ironman Austria durch die Wertschöpfung von 7 Millionen Euro auch eine enorme wirtschaftliche Bedeutung für den Zentralraum Kärntens gewonnen (Zeitlinger 2013). Auch die Verwendung von GPS Geräten während des Wettkampfes, die für die vorliegende Arbeit bzw. Forschungsfrage unerlässlich sind, haben sich in den letzten Jahren immer größerer Beliebtheit erfreut und werden nach Dolan et al. (2011, S. 1022) von 41% der Triathleten verwendet.
Doch der wichtigste Aspekt, der für eine Relevanz der Master Thesis spricht, ist die nach Lepers (2008, S. 1828–1831) vorhandene Tendenz der Athleten zu immer besseren und

schnelleren Rennleistungen. Nach einer Studie von Lehmann (2016, S. 5) sind Ironman Sportler als sehr leistungsorientiert zu klassifizieren und bei 80% aller Athleten ist die Zielzeit ein wichtiges Motiv für die Teilnahme. Diese Arbeit soll dazu beitragen, die Vorstellungen und Zielvorgaben der Athleten durch den Einsatz von Geoinformatik zu erleichtern.

1.2 Darstellung des realweltlichen Problems

Die Leistungen und Zielzeiten wurden seit Bestehen der Sportart kontinuierlich verbessert und führten somit zu immer schnelleren Rennzeiten. Zurückzuführen ist dies auf den Einsatz von neuem Material sowie neuen Ernährungs- und durch Pacing-Strategien (Lepers 2008, S. 1828–1831), was ein zentraler Bestandteil dieser Arbeit ist. Zum Aspekt der Renntaktik im Triathlon Sport (engl. pacing-strategy) hat sich in den letzten Jahren, speziell im Bereich der Sportmedizin und Sportwissenschaft, ein großes Forschungsfeld aufgetan. Das Ziel ist es dabei, die Pacing-Strategie derart zu adaptieren, dass ein möglichst hohes Leistungsergebnis zu erwarten ist (Abbiss und Laursen 2008, S. 239). Nach Friel (2012) bzw. Abbiss und Laursen (2008, S. 239) stellt die Slow-Starting-Strategy eine weit verbreitete Theorie dar, wonach eine anfänglich geringere Durchschnittsgeschwindigkeit zu einer höheren Leistung führt. Ebenso stark vertreten ist die Even-Pacing-Strategy, bei welcher für ein hohes Leistungsergebnis ein möglichst gleichmäßiges Tempo nahe an der Trainingsleistung empfohlen wird (Laursen 2011, S. 258) und (Boswell 2012, S. 651).

Eine Analyse der Rad-Rundenzeiten von 2.179 Athleten, die das Ziel des Ironman Austria 2010 erreichten, spiegelt in der Praxis jedoch ein gänzlich anderes Bild wider.

Platzierung der Athleten nach Gesamtradzeit	Gesamtradzeit in Gruppierung	Zeitunterschied der beiden Ironman Runden (je 90km)	
		Relativ (in%)	Absolut
1 bis 10	4h:18m bis 4h:35m	4,5%	05m:57s
11 bis 20	4h:36m bis 4h:39m	6,2%	08m:22s
11 bis 50	4h:36m bis 4h:49m	5,4%	07m:29s
21 bis 50	4h:40m bis 4h:49m	5,2%	07m:11s
51 bis 100	4h:50m bis 4h:53m	5,0%	07m:19s
101 bis 250	4h:53m bis 5h:01m	5,1%	07m:46s
251 bis 500	5h:01m bis 5h:13m	6,2%	09m:18s
501 bis 1000	5h:13m bis 5h:34m	7,2%	11m:23s
1001 bis 1500	5h:34m bis 6h:01m	8,6%	14m:21s
1500 bis 2179	6h:01m bis 8h:25m	10,9%	20m:43s

Tabelle 1-1: Radzeiten des Ironman Austria nach Platzierung und Differenz pro Runde
Eigene Bearbeitung. Datenquelle: (Pentek Timing 2010, S. 1–23)

Die detaillierte Ergebnisliste von Pentek Timing (2010, S. 1–23) wurde nach den Resultaten der Raddisziplin gefiltert (Tabelle 1-1). Von den insgesamt 2224 Starten beim Ironman Austria haben 2.179 Athleten die Radstrecke in der vorgegebenen Cut-off Time von 10h:10m absolviert. Dies entspricht somit einer Ausfallquote von 2% der Starter. Die Ergebnisse in der Raddisziplin wurden anschließend nach den zwei identen Radrunden und der Platzierung der Athleten klassifiziert. Die Tabelle 1-1 weist für die ersten zehn Athleten im durchschnittlichen Vergleich der beiden Radrunden einen Zeitunterschied von 4,47% auf. In absoluten Werten bedeutet dies einen Zeitverlust zwischen der ersten und zweiten Radrunde von 5 Minuten und 57 Sekunden. Die Zeitdifferenz wird hierbei mit abnehmender Platzierung immer größer. So weisen die letzten 500 Athleten des Wettkampfes einen durchschnittlichen Zeitunterschied von 11% bzw. 21 Minuten von Runde 1 zu Runde 2 auf.

Die Auswertung der Radzeiten des Ironman Austria deckt sich mit jenen der Studie von Abbiss et al. (2006, S. 726–734), die mit gut trainierten Langdistanz-Triathleten durchgeführt wurde. Dabei konnte eine repräsentative Auswahl an Athleten beim Ironman Western Australia mit Herzfrequenz- und Watt-Leistungsmessungen in jeder der drei Radrunden analysiert werden. Auch unter dem Gesichtspunkt eines natürlichen Leistungsabfalls, durch Kraft- und Konditionsabnahme während der physischen Anstrengung, war der Geschwindigkeitsverlust von Runde zu Runde maßgeblich durch eine falsche Pacing-Strategie geprägt (Abbiss et al. 2006, S. 726–734) .

Demnach gibt es eine Diskrepanz zwischen der in der Sportwissenschaft geforderten Even-Pacing-Strategy und der in der Praxis umgesetzten Rennleistung.

Diese Beobachtung einer falschen Renntaktik, speziell von unerfahrenen bzw. weniger trainierten Athleten, stellt ein realweltliches Problem dar, welches als Ausgangspunkt der vorliegenden Masterarbeit herangezogen wird.

1.3 Hypothese, Zielsetzung und Forschungsfrage

Der realweltlichen Problemdarstellung zur Widersprüchlichkeit von sportwissenschaftlicher Theorie und Praxis bei der Rennstrategie des Ironman Austria soll in dieser Arbeit nachgegangen werden. Die Lösung dieses Sachverhaltes basiert auf folgender **Hypothese**: Ein Athlet, der eine ständige Anweisung und Rückmeldung zur Einhaltung der Rennstrategie bekommt, wird durch optimale Pacing-Strategien aus sportwissenschaftlichen Erkenntnissen eine konstantere und demnach schnellere Radleistung im Wettkampf erbringen können.

Ziel ist es, durch den Einsatz von Methodiken und Technologien der Geoinformatik während des Wettkampfes eine Marschroute vorzugeben, um den Geschwindigkeitsrückgang durch eine falsche Renneinteilung zu verhindern. Die Leistungsfähigkeit des Athleten kann somit reguliert und möglichst gleichmäßig abgerufen werden. Eine weitere Zielsetzung ist dabei als GeoSimulation die individuelle Anpassung der Marschroute an die Fähigkeiten und Leistungsparameter des Athleten.

Im Zuge der Master Thesis wird dieser Zielsetzung mit folgender wissenschaftlicher Forschungsfrage nachgegangen:

Ist es möglich, auf Basis der in der Sportwissenschaft vorherrschenden Pacing-Strategien eine für jeden Athleten spezifische und nach seinen Leistungsparametern definierte Modellierung und Geosimulation der Radstrecke, am Beispiel des Ironman Austria, als Marschroute für GPS Geräte umzusetzen?

1.4 Lösungsansatz der wiss. Fragestellung

Das in Kapitel 1.2 erläuterte realweltliche Problem und die in Kapitel 1.3 formulierte wissenschaftliche Fragestellung sollen durch den Einsatz von Methodiken geographischer Informationssysteme (GIS) gelöst werden. Dabei wird als konzeptioneller Ansatz eine Marschroute herangezogen, durch die der Athlet während des Rennens seine, zuvor individuell erstellte, Rennstrategie umsetzen kann. Um eine solche Hilfestellung zu ermöglichen, wird die GPS Technologie (Kapitel 2.4.1) eingesetzt, welche nach Friel

und Vance (2013, S. 153, 143) ein hervorragendes Werkzeug zur Trainings- und Wettkampfgestaltung darstellt. GPS Geräte zeigen dem Athleten in Echtzeit, neben biometrischen Daten, vor allem Geschwindigkeitsinformationen an, welche das aktuelle Pacing, nach Leistungszustand des Sportlers, validieren (ebd.). Eine Studie von Dolan et al. (2011, S. 1022) zeigt den breiten Einsatz von GPS Geräten bei Triathleten mit 41% und dies wird nur noch durch den Einsatz von Pulsmessgeräten mit 84% übertroffen. Es ist noch darauf hinzuweisen, dass aktuelle GPS Geräte oftmals die Funktion zur Messung der Herzfrequenz bereits implementiert haben. Auf solchen, speziell für Athleten entworfenen, GPS Devices kann eine Marschroute zur Lösung des realweltlichen Problems gespeichert und abgespielt werden. Dabei kann dem Athleten während der sportlichen Aktivität die Position bzw. Differenz zur optimalen Marschroute visuell und optional auch akustisch ausgegeben werden (Abbildung 1-2).

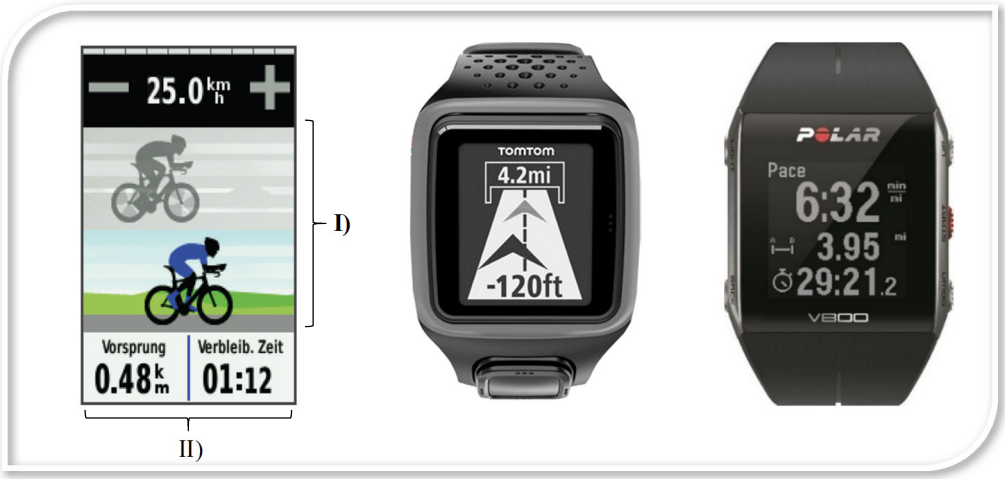


Abbildung 1-2: Visualisierungen von Marschrouten auf GPS Geräten
Quellen: (Garmin 2013, S. 5) (TomTom 2016, S. 27) (Polar 2016, S. 58)

Die Abbildung 1-2 vergleicht die Visualisierungen von Marschrouten der bekanntesten Hersteller von GPS Geräten für Triathleten. Bei der Firma Garmin (links) wird der aktuelle Abstand zur vorgegebenen Marschroute in Form zweier Radfahrer schematisch veranschaulicht (Abbildung 1-2, I). Weiters stellt das Gerät konkrete Zahlen über den Vorsprung bzw. Rückstand zur vorgegebenen Marschroute in Kilometer und Minuten dar (Abbildung 1-2, II).

Als Marschroute kann an den GPS-Geräten werkseitig nur eine lineare Vorgabe mit einer Durchschnittsgeschwindigkeit erstellt werden. Es besteht nach aktuellem Stand der Technik keine Möglichkeit, auf die Leistungsparameter der Athleten und somit auf die Topographie, oder auf Pacing-Strategien Einfluss zu nehmen. Somit wird dem Athleten eine Fahrgeschwindigkeit vorgegeben, die keine Rücksicht auf Steigungen oder Gefälle nimmt. In der Praxis bedeutet dies, dass etwa bei einer definierten Durchschnittsgeschwindigkeit von 32 km/h dieser Schnitt stets eingehalten werden muss, egal ob der Athlet gerade einen Berg mit 12% Steigung bewältigt oder ob er sich in einer rasanten Abfahrt befindet.

Im Zuge dieser Masterarbeit sollen die Leistungsparameter, Rennstrategien und das Gelände in die Modellierung der Marschroute einfließen. Um dies zu ermöglichen wird ein, mit GPS Geräten kompatibles, Dateiformat namens GPX eingesetzt (Kapitel 2.4.2). Der Fokus dieser Arbeit liegt in der Manipulation des Timestamps zur GPX Datei (Abbildung 1-3), um die individuelle Durchschnittsgeschwindigkeit bzw. Gesamtzeit des Athleten simulieren zu können.

```
<time>2016-01-09T19:43:22.838Z</time>
```

Abbildung 1-3: Beispiel eines Timestamp – Tags einer GPX Datei

Diese Timestamps werden so verändert, dass der Athlet nach Eingabe von Leistungsparametern und unter Berücksichtigung der Topographie, eine simulierte Streckenführung für seinen Wettkampf erhält. Diese GPS gestützte Marschroute gibt während des Rennens eine zu fahrende Geschwindigkeit vor, die je nach Steigung, Gefälle, Pacing-Strategie usw. ständig variiert. Durch diesen dynamischen Aspekt der Simulation können die in der Sportwissenschaft vorherrschenden Theorien eingesetzt werden, um eine schnellere und dabei kraftsparendere Radleistung abzurufen.

1.5 Struktur der Arbeit

Nach bisherigen Literaturrecherchen gibt es keine konkreten Arbeiten zur simulativen Manipulation der Timestamps von GPX Dateien, um diese als Marschroute zu verwenden. Daher sollen mittels der vorliegenden Master Thesis und durch den Einsatz von GIS Methoden und GIS Technologien neue Erkenntnisse bei der Umsetzung der wissenschaftlichen Fragestellung angestrebt werden. Auf Basis von fachspezifischer Literatur der Sportmedizin, Sportwissenschaft, Fernerkundung, räumlichen Analysemethoden, Modellierung, GeoSimulation und Global Positioning Systemen soll

die interdisziplinäre wissenschaftliche Fragestellung beantwortet werden. Dieser aktuelle Forschungsstand aus den jeweiligen Disziplinen wurde in Kapitel 2 eingearbeitet, um die theoretischen Grundlagen und eingesetzten Technologien zu erläutern.

Die Grundlage dieser Arbeit erfolgt auf einer empirischen, operativen Ebene mittels einer Umfrage. Dabei wurden bei 622 Triathleten empirische Felddaten angefragt, die tatsächlich beim Wettkampf des Ironman Austria als GPX Datenformat aufgezeichnet wurden. Insgesamt konnten 107 raum-zeitliche Bewegungsdaten mit insgesamt über 1,6 Millionen Messpunkten eingeholt und in Form einer deskriptiven Statistik analysiert werden (Kapitel 3.2). Auf Basis dieser empirischen Felddaten werden die Leistungsparameter der Athleten nach unterschiedlichen Leistungskategorien der Zielzeiten klassifiziert (Kapitel 3.3.2). Diese durchschnittlichen Radzeiten wurden über eine räumliche Verknüpfungsmethode mit den reinen Objektgeometrien zusammengeführt. Diese leeren Punkt Geometrien werden nach spezifischen Charakteristika erstellt, um einerseits eine geordnete Datenanalyse durchführen zu können und, andererseits, um einen späteren Export in eine GPX Datei zu ermöglichen. Für die Modellierung wird des Weiteren eine Terrain - und Radstreckenanalyse durchgeführt, um mittels hochauflösender Airborne Laserscan Daten sehr genaue Höheninformationen abbilden zu können.

Ein zusätzlicher Schwerpunkt dieser Arbeit liegt in der Anwendung einer Methodik namens „dynamische Segmentierung“, um alle identifizierten Leistungsparameter in die modellierte Radstrecke aufzunehmen (Kapitel 4.2). So werden in diesem Kapitel jene Einflussfaktoren ermittelt und analysiert, die dazu führen, dass der Sportler keine lineare und gleichmäßige Geschwindigkeit auf der Radstrecke umsetzen kann. Im Zuge dessen erfolgt eine Gewichtung der Leistungsparameter, um Induktionsschlüsse aus den empirischen Datensätzen zu ziehen und diese in einem Theoriekonzept abbilden zu können. Dabei soll die Frage beantwortet werden, wie stark der Einfluss jedes Leistungsparameters auf die Radleistung ist, um ein möglichst realistisches GIS Modell zu erzeugen.

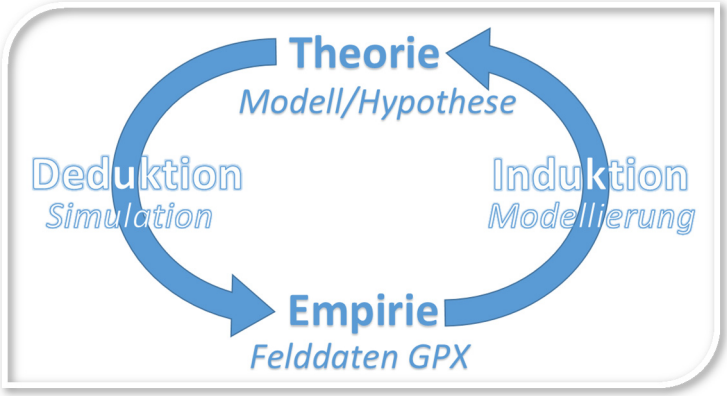


Abbildung 1-4: Ablaufdiagramm zur Modellierung und Simulation
Eigene Bearbeitung. Grundidee: (Balzert et al. 2008, S. 49)

Die Abbildung 1-4 veranschaulicht die Vorgehensweise in der Masterarbeit, um durch Induktionsschlüsse aus der Empirie bzw. den erhobenen Felddaten eine möglichst wirklichkeitsgetreue Modellierung umzusetzen.

Die Erkenntnisse der statistischen Analysen wurden in einem Workflow zur Modellierung und GeoSimulation eingearbeitet, um die leeren GPX Track Geometrien über die entsprechenden Geschwindigkeitsangaben mit Timestamps befüllen zu können (Kapitel 5). Die Programmierung dieser Funktionen erfolgte in der C# Programmiersprache und kann über ein ArcMap Add-in als graphische Benutzeroberfläche ausgeführt werden. Der Benutzer soll, auf Basis der empirischen Felddaten, GPX Tracks modellieren können, die jeweils den Durchschnitt einer Leistungsklasse repräsentieren. Über die GeoSimulation ist es möglich, die Leistungsparameter individuell und Athleten spezifisch zu adaptieren und nach den jeweiligen Bedürfnissen und Zielzeiten anzupassen.

Die erfassten Felddaten dienen im Zuge dieser Masterarbeit nicht nur der Bildung einer Hypothese, sondern werden auch zur Überprüfung der Modellierung und der GeoSimulationen herangezogen. Durch Deduktionsschlüsse soll die Beantwortung der wissenschaftlichen Fragestellung, welche als konkrete Zielsetzung formuliert wurde, in Form von Zahlenvergleichen überprüft werden. Dies bedeutet in der Umsetzung eine Validierung der simulierten GPX Dateien mit den empirischen Felddaten, um die Funktionsfähigkeit der GIS Modelle und der GIS Simulationen zu garantieren (Kapitel 6.2).

1.6 Konzeptuelles Modell der Methodiken als Überblick

konzeptueller Ansatz: in Richtung der Erfüllung des langfristigen Zieles

Typisierung der Ziele	Methodik	Ergebnis / Überprüfbarkeit
(konkretes) Ziel I: geometrische Datenerfassung Welche Route wird innerhalb der gesperreten Straße verwendet? Erstellung der GPX Trackpoints auf Basis der GIP Daten mit Koordinatenangaben (ohne timestamps).	**Modellierung** durch Tracing	**Kapitel 3.1.2** □ Geometrie der GPX Datei mit Koordinaten der Trackpoints
(konkretes) Ziel II: Höheninformationen Verschneidung der GPX Trackpoints mit DGM, um Höheninformationen zwischen den Points zu erlangen.	**Räumliche Analysemethoden:** Verschneidung, Interpolation	**Kapitel 3.1.3.2 & 3.1.3.4** □ GPX Track mit Z Information und & Steigung in %
(konkretes) Ziel III: Semantische Verknüpfung Verknüpfung der GPX Trackpoints mit den Grenzen der Katastralgemeinden und der Anstiege laut ÖK 50.000.	**Join & Verschneidung**	**Kapitel 3.1.3.1 & 3.1.3.5** □ GPX Track mit Attribut zur Katastralgemeinde und zum Namen des Anstieges
(operatives) Ziel IV: deskriptive Statistik zu empirischen Felddaten Fakten über Datenqualität und Repräsentativität der empirischen Felddaten	**statistische Berechnung und Beschreibung**	**Kapitel 3.2** □ Kennzahlen zu Range, Median, Modus, Standardabweichung
(konkretes) Ziel V: Aggregieren der empirischen Felddaten Bildung von 12 Leistungsklassen der 107 empirischen Felddaten. Aggregieren dieser Daten zu den GPX Trackpoints	**Klassifikation, Spatial Join, Puffer**	**Kapitel 3.3** □ GPX Tracks mit empirischen Felddaten nach Klassen
(konkretes) Ziel VI: Datenaufbereitung zur Analyse Geometrie als Kilometrierung entlang der Strecke zu den Themen: Gemeindegrenze, Topographie, Labestation, Kurvenradien, Steigung in % und Fahrbahnbeschaffenheit.	**dynamische Segmentierung & Klassifizieren**	**Kapitel 4** □ Route mit Event Tabellen
(operatives) Ziel VII: geostatistische Analysen Erhebung von Leistungsparametern und Interpretation dieser auf die Fahrgeschwindigkeit nach Leistungskategorien.	**GeoStatistik**	**Kapitel 4** □ Diagramme zu Auswirkungen der Leistungsparameter auf Fahrgeschwindigkeit. □ Klare Rückschlüsse zur Einarbeitung in GeoSimulation
(konkretes) Ziel VIII: Workflow zur Modellierung & GeoSimulation Automatisierter Worklfow zur Generierung von GPX Tracks nach bestehenden empirischen Felddaten (Modellierung) oder GeoSimulation nach Eingabe von individuellen Leistungsparametern.	**Programmierung der Modellierung und GeoSimulation**	**Kapitel 4** □ GUI zur Modellierung und GeoSimulation
strategisches Ziel: **Beantwortung der wiss. Fragestellung:** Ist es möglich, auf Basis der in der Sportwissenschaft vorherrschenden Pacing-Strategien eine für jeden Athleten spezifische und nach seinen Leistungsparametern definierte Modellierung und Geosimulation der Radstrecke, am Beispiel des Ironman Austria, als Marschroute für GPS Geräte umzusetzen?	**Summe aller verwendeten Methoden.**	GPS konforme GPX Datei, die nach Eingabe individueller Leistungswerte der Athleten und der *pacing strategy* , als Marschroute während des Wettkampfes dienen soll. Überprüfbarkeit: □ GPX Track nach Modellierung □ GPX Track nach GeoSimulation

Überprüfung

(konkretes) Ziel IX: deduktive Datenüberprüfung Validierung der Funktionsfähigkeit der GeoSimulation durch Deduktionsschlüsse vom Modell zu den empirischen Felddaten.	**Datenvergleich**	**Kapitel 6.2** □ GeoSimulation von zwei Athleten □ Vergleich der Ergenisse mit GPX Tracks des Wettkampfes

2. Theoretische Grundlagen und eingesetzte Technologien

In diesem Kapitel sollen die grundlegenden sportwissenschaftlichen Erkenntnisse sowie die eingesetzten GIS Technologien erläutert werden, die zur Beantwortung der wissenschaftlichen Fragestellung notwendig sind. Dabei erfolgt eine Definition des Triathlon Sports und es wird beschrieben, weshalb die Forschungsfrage nicht für alle Distanzen übertragbar ist. Neben den Charakteristiken zum Untersuchungsgebiet des Ironman Austria sind die sportwissenschaftlichen Pacing-Strategien wichtige Unterkapitel, um die späteren Parameter der GeoSimulation verstehen zu können.

2.1 Triathlon Sport

Die Vision eines Multisportlers, der mehrere aufeinander folgende Sportarten kombiniert, galt schon in der Antike als erstrebenswert (Neumann et al. 2004, S. 13). Aus dieser Vorstellung heraus entwickelte sich etwa der Dekathlon, ein moderner Zehnkampf in der Leichtathletik (ebd.). Der Begriff Triathlon bezeichnet einen Dreikampf aus den Sportarten Schwimmen, Radfahren und Laufen ohne zeitlicher Unterbrechung. „Ein erster echter Triathlon, der den heutigen Vorstellungen entsprach, wurde bereits am 25.09.1974 in der Mission Bay (USA) mit 805 Meter Schwimmen, 8 Kilometern Radfahren und 8 Kilometern Laufen gestartet“ (Neumann et al. 2004, S. 14). Obwohl Neumann et al. (2004, S. 14) in Bezug auf diese Erstaustragung im Jahr 1974 das Adverb: „bereits“ verwendet, beschreiben Burke (2007, S. 71) und Lund (1996, S. 7) den Triathlon Sport als eine relativ junge Sportart, welche erstmals im Jahr 2000 als Disziplin bei den Olympischen Spielen vertreten war (Burke 2007, S. 71). Die dabei absolvierten Entfernungen der drei Sportarten sind klar definiert, weshalb eine solche Veranstaltung als Wettkampf über die olympische Distanz bezeichnet wird. Ein Triathlon kann über mehrere unterschiedliche Streckenlängen ausgetragen werden (Tabelle 2-1).

Bezeichnung	Disziplinen (in km)		
	Schwimmen	Radfahren	Laufen
Sprint	0,75	20	5,0
Olympische Distanz – Kurzdistanz	1,50	40	10,0
Mittel – Half Ironman	1,90	90	20,0
Lang - Ironman	3,80	180	42,195
Dreifach	11,40	540	126,6
Zehnfach	38,00	1800	422,0

Tabelle 2-1: Streckenlängen im Triathlon Sport
Eigene Bearbeitung. Datenquelle: (Neumann et al. 2004, S. 21)

Die Ironman Wettkämpfe zählen zu den weltweit beliebtesten und bekanntesten Triathlon Bewerben über die Langdistanz. Seit der Erstaustragung des Ironman Hawaii im Jahr 1978 ist in dieser Serie mit über 20 Rennen weltweit ein starkes Ansteigen an Teilnehmern zu verzeichnen (Rüst et al. 2012, S. 114–115). Dieser Anstieg wurde in Kapitel 1.1 am Beispiel des Ironman Austria dargestellt und die Entwicklung seiner Teilnehmerzahlen tabellarisch aufgelistet. Im Zuge dieser Arbeit wurde der Fokus auf das Langdistanz Ironman Rennen gelegt, da diese Streckenlänge und der entsprechende Rennverlauf für eine GIS Modellierung am optimalsten sind. Im Gegensatz zu Triathlons über die olympische Distanz herrscht beim Ironman keine dynamische Rennsituation vor, sondern eine relativ kontinuierliche Geschwindigkeitseinteilung, da von den Athleten eine Even-Pacing-Strategy (Kapitel 2.3) präferiert wird. Die Begründung für eine sich ständig verändernde Tempogestaltung auf der olympischen Distanz liegt sowohl in der Freigabe des Windschattenfahrens bei der Raddisziplin (kein Mindestabstand zwischen den Athleten) als auch in den vermehrten Attacken und Tempoverschärfungen (Wu et al. 2014, S. 226–227).

2.2 Das Untersuchungsgebiet Ironman Austria

Der, seit dem Jahr 1998 existierende Ironman Austria als Langdistanz Triathlon zählt zu den größten Sportveranstaltungen Kärntens (World Triathlon Corporation 2016). Derzeit gibt es weltweit 39 Langdistanz Ironman Sportveranstaltungen (ebd.). Diese sind hinsichtlich der Topographie, der klimatischen Verhältnisse und der Streckenführung höchst unterschiedlich (ebd.). Die Laufstrecken sind im Durchschnitt mit 150 Höhenmetern relativ flach (ebd.) und daher für eine Modellierung im Zuge dieser Masterarbeit nicht sonderlich interessant. Aktuelle Sport GPS Geräte können für die wissenschaftliche Fragestellung (Kapitel 1.3) bereits eine Marschroute ohne Hinblick auf die Topographie vorgeben, weshalb der Fokus bei der Modellierung und Simulation einer Marschroute explizit auf der Radstrecke lag. Dabei gibt es eine große Diversität in Bezug auf die bergauf zu überwindenden Höhenmeter. Diese reichen von 300 Höhenmeter beim Ironman Florida bis hin zu 2.600 Höhenmeter beim Ironman Lanzarote (World Triathlon Corporation 2016). Der Ironman Austria zählt dahingehend mit den offiziell ausgewiesenen 1.700 Höhenmetern bergauf (ebd.), zu den topographisch anspruchsvolleren Radstrecken. Nicht nur auf Grund der Topographie und des persönlichen Bezuges durch mehrere Teilnahmen ist dies ein interessantes Bearbeitungsbeispiel, sondern auch angesichts der geringen äußeren Störeinflüsse.

So gibt es beim Ironman Austria kaum nennenswerte meteorologische Einflüsse, wie etwa hohe Windgeschwindigkeiten, welche die Simulation einer Marschroute enorm erschweren würden. So liegt nach Research Studios Austria iSPACE (2016) die mittlere Jahreswindgeschwindigkeit bei weniger als 3,1 m/s, was im Vergleich zur Ironman Weltmeisterschaft auf Hawaii mit 6,5 m/s (National Weather Service Weather Forecast Office 2016) eher gering ausfällt. Dies ist sicherlich in der Beckenlage begründet, in der sich die Radstrecke des Ironman Austria befindet, welches auch die Abbildung 2-1 verdeutlicht.

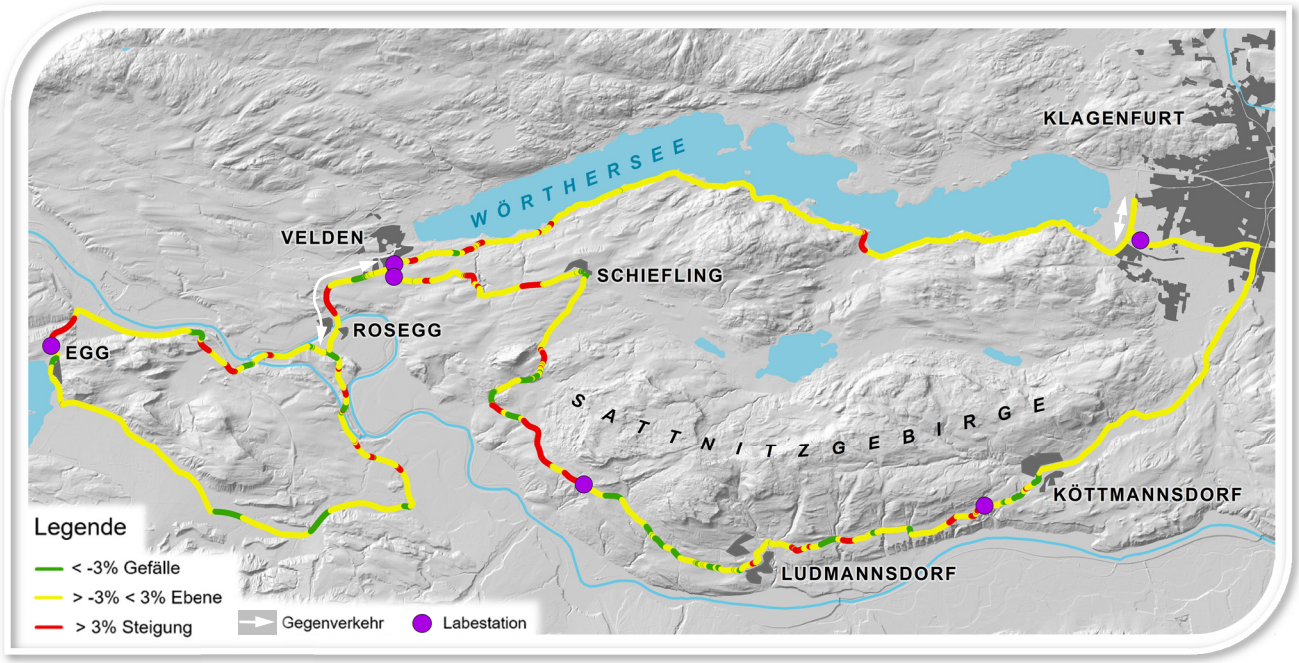


Abbildung 2-1: Streckenverlauf des Ironman Austria auf Hillshade Kartengrundlage
Eigene Bearbeitung. Datenquelle Airborne Laserscan Daten: (Amt der Kärntner Landesregierung 2016)

Diese Abbildung 2-1 visualisiert als Kartengrundlage eine Hillshade Darstellung, welche durch Airborne Laserscan Daten (Kapitel 2.4.3) generiert wurde. Dabei handelt es sich um eine geschummerte Reliefkarte, welche unter Einbeziehung von Beleuchtung und Schattenwurf einen guten topographischen Überblick verschafft. Dieser optische Eindruck des Hillshade wird durch die Einfärbung der Fahrstrecke nach Steigung, Gefälle und Ebene zusätzlich verstärkt. Die dabei eingesetzten Klassifizierungsgrenzen erfolgten in der vorliegenden Arbeit nach Friel und Vance (2013, S. 125). Eine erste Analyse zeigt, dass 74,7 % der Radstrecke als Ebene zu definieren sind, 13,1% als Steigung und 12,2 % als Gefälle. Die Fahrstrecke des jährlich stattfindenden Ironman Austria führt über zwei nahezu idente Runden von der Wechselzone über die Süduferstraße des Wörthersees nach Velden. Die vorhandenen Ausgleichskilometer der ersten Runde, welche die 180km Distanz ermöglichen, wurden für die Analysen und die Vergleichbarkeit herausgerechnet.

Nach dem Passieren der ersten von sechs Labestationen, an denen die Athleten mit Getränken und Nahrung versorgt werden, erfolgt ein Abschnitt mit Gegenverkehr. Dieser Sektor ist hinsichtlich der Modellierung unter besondere Berücksichtigung zu stellen, da hier die empirischen Felddaten der richtigen Fahrspur und somit der entsprechenden topographischen Gegebenheiten (bergauf oder bergab) zugeordnet werden müssen (Kapitel 3.3.3). Die bereits angesprochenen Labestationen stellen einen weiteren Aspekt dar, der bei der Simulation zu beachten ist, da die Verweildauer je nach Leistungskategorie höchst unterschiedlich ist.

2.3 Sportwissenschaftliche Aspekte

Der Fokus dieser Arbeit liegt eindeutig im Forschungsfeld der Geographischen Informationssysteme. Trotzdem soll, nun folgend, kurz auf die sportwissenschaftlichen Aspekte eingegangen werden, da diese den Grundstock zur Beantwortung der Forschungsfrage bilden. Durch die Erkenntnisse aus der Sportwissenschaft kann das realweltliche Problem der suboptimalen Renn- und Tempoeinteilung in der Raddisziplin bearbeitet werden. Als Pacing-Strategie bezeichnet Laursen (2011, S. 258) die Tempogestaltung der Athleten während eines Ausdauerwettkampfes zum Erzielen einer möglichst hohen Leistung. Dieser optimale Leistungsoutput korreliert mit einer niedrigen Renndauer und es ergeben sich somit hohe Wettkampfplatzierungen der Sportler. Auch unter dem Gesichtspunkt eines natürlichen Leistungsabfalls durch Kraft- und Konditionsabnahme während der physischen Anstrengung soll der Geschwindigkeitsverlust von Runde zu Runde maßgeblich durch eine falsche Pacing-Strategie geprägt sein (Abbiss et al. 2006, S. 726–734). Die Forschungen in der Sportwissenschaft zu Thema Pacing-Strategien konzentrieren sich hauptsächlich auf Felduntersuchungen, um die optimalste Tempoeinteilung, bei möglichst ökonomischem Energieverbrauch der Athleten, zu finden. Speziell, auf den Triathlon Sport hin zugeschnittene Untersuchungen sind äußerst rar und dabei liegt der Fokus insbesondere auf der Sprint- und olympischen Distanz (Kapitel 2.1). Nach intensiven Recherchen wurde eine Handvoll an Publikationen ausfindig gemacht, die sich explizit der Renneinteilung bei längeren Triathlon Wettkämpfen, wie der Mittel- und Langdistanz, widmet (Abbiss und Laursen 2008, S. 239). Boswell (2012, S. 651) schreibt mit der Dynamic-Pacing-Strategy von einer weit verbreiteten Tempogestaltung, die intuitiv und spontan erfolgt. Diese variable Tempogestaltung ist sehr stark vom Rennverlauf geprägt, der auf Grund von gegnerischen Ausreißversuchen, zu einer spontanen

Tempoverschärfung führen kann (Abbiss und Laursen 2008, S. 240). Diese Taktik verfolgen nach Boswell (2012, S. 651) hauptsächlich Profisportler bei Weltmeisterschaften bzw. olympischen Spielen. Renntaktiken, die auch bei Amateur Athleten zum Einsatz kommen, klassifiziert Koning et al. (2011, S. 1–2) nach der Even-Pacing-Strategy, der Positive-Pacing-Strategy und der Negative-Pacing-Strategy. Bei der Even-Pacing-Strategy bleibt der Leistungsoutput über die gesamte Renndauer konstant hoch und führt bei Strecken mit mehreren Runden zu gleichen Durchgangszeiten (Abbiss und Laursen 2008, S. 244). In der Praxis wird die Positive-Pacing-Strategy am häufigsten von den Athleten umgesetzt (Abbiss et al. 2006, S. 726). Dabei erfolgt über die gesamte Wettkampfdistanz eine sukzessive Geschwindigkeitsreduktion (ebd.). Diese Renneinteilung wird vom Athleten nicht bewusst umgesetzt, sondern intuitiv gewählt (ebd.). Abbiss und Laursen (2008, S. 243) bezeichnet diese Renneinteilung gar als unrealistische Zielsetzung von überambitionierten Athleten. Eine bewusst eingesetzte Renntaktik, die in der Sportwissenschaft großen Anklang findet, ist jene der Negative-Pacing-Strategy (Abbiss und Laursen 2008, S. 241). Hierbei erfolgt ein stetiger Geschwindigkeitszuwachs über die gesamte Wettkampfdauer, welcher eine anfänglich übermäßige Sauerstoffaufnahme (VO_2) verhindern soll (ebd.). Untersuchungen zufolge können Temporeduktionen von 15% über ⅕ der Renndauer zu einer signifikanten Verbesserung der gesamten Wettkampfleistung führen (ebd.). Aisbett et al. (2009, S. 1201) bezeichnet diese zurückhaltende Renntaktik am Beginn eines Wettkampfes als Slow-Starting-Strategy. Feldforschungsergebnisse aus der Sportwissenschaft zur Renntaktik bei Triathlon Mittel- und Langdistanzen geben keine eindeutige Pacing-Strategie vor. Abbiss et al. (2006, S. 730) und Laursen (2011, S. 258) kommen in ihren Forschungen zum Ergebnis, dass die Even-Pacing-Strategy zum größten Wettkampferfolg führt. In der Arbeit von Aisbett et al. (2009, S. 1201) ist hingegen nachzulesen, dass eine Temporeduktion zu Beginn des Wettkampfes zur erfolgreichsten Rennleistung zählt. Diese inhomogene Meinung in der Sportwissenschaft führt dazu, dass in der Simulation der Marschroute als GPX Track beide vorherrschenden Pacing-Strategien in der GUI und Berechnung umgesetzt werden (Kapitel 5). Bei fast allen Studien wird auf die externen Leistungsparameter hingewiesen, welche einen großen Einfluss auf die Tempogestaltung haben. So wurde im Zuge dieser Arbeit durch geostatistische Analysen ein spezielles Augenmerk auf die Topographie, Kurvenradien und Fahrbahnbeschaffenheit gelegt (Kapitel 4.3).

2.4 Eingesetzte GIS Technologien

2.4.1 Globale Navigationssatellitensysteme (GNSS)

Nach Maddison und Ni Mhurchu (2009, S. 76–77) findet die GPS Technologie bereits sehr breiten Einsatz im Sportsektor. Während der sportlichen Aktivität wir die GPS Methode eingesetzt, um den Athleten durch die Navigationsfunktion und aktuelle Geschwindigkeitsangaben zu unterstützen. Die GPS Technik wird aber auch sehr stark für raum-zeitliche Analysen nach der sportlichen Einheit verwendet, um den Athleten Rückschlüsse auf Leistungsintensität oder Rennstrategie geben zu können (Maddison und Ni Mhurchu 2009, S. 76–77). Da der Begriff Global Positioning System (GPS), welcher hauptsächlich die Standortbestimmung implementiert, für die moderne Inwertsetzung durch Navigationsabfragen nicht mehr passend ist, ändert sich die Begrifflichkeit hin zu globalen Navigationssatellitensystemen (GNSS) (de Lange 2013, S. 187–200). Diese Systeme haben einen hohen kommerziellen Erfolg als Navigationshilfen im Individualverkehr, im Tourismus oder im Sportsektor (ebd.). Die Anfänge in der satellitengestützten Navigation liegen im Jahr 1978, als das US Militär mit diesem Projekt startete (ebd.). Für die Standortbestimmung werden zeitgleich mindestens drei verschiedene Satelliten benötigt, die in einer Höhe von ca. 30.200 km auf sechs unterschiedlichen Erdumlaufbahnen kreisen (ebd.). Es handelt sich um ein passives System, da die Daten vom GPS Gerät nur empfangen werden können (ebd.). Bei den übermittelten Informationen handelt es sich um die Signallaufzeiten sowie um die Position der Satelliten, durch welche die geographische Lage am Empfangsgerät berechnet werden kann (ebd.). Die Genauigkeit von GPS Empfängern hängt einerseits von der Geometrie ab, wie die Satelliten zueinander stehen (ebd.). Ein weiterer Faktor für nicht exakte Standortbestimmungen kann eine mögliche Abschattung durch Häuserfluchten oder Geländekanten sein (ebd.). Die Refraktion, also die unterschiedliche Brechung der Radiowellen in der Ionosphäre, verursacht durch die Wetterlage, ist noch ein Faktor zur Exaktheit der GPS Technologie (ebd.). Bei der Standortbestimmung mit Consumer Geräten führen diese angesprochenen Kriterien zu einer Unschärfe von etwa +/- 10 Meter (ebd.). Beim Differential Global Positioning System (DGPS), welches auch bei der Methode zum Airborne Laserscanning eingesetzt wird (Kapitel 2.4.3), kann die Positionsabweichung auf 1-2 cm reduziert werden (ebd.).

2.4.2 GPS Exchange Format (GPX)

Die 107 empirischen Felddaten mit realen Wettkampfaufzeichnungen der Athleten beim Ironman Austria wurden im GPX Format übermittelt. Dieses Dateiformat wird ebenfalls bei der Modellierung und Geosimulation zur Erstellung einer Marschroute eingesetzt, um schlussendlich die wissenschaftliche Fragestellung beantworten zu können. Auf Grund der breiten Anwendung dieses Datentyps in der Masterarbeit soll, nun folgend, die Grundlage zur Bearbeitung dieser Dateien vermittelt werden. Das GPS eXchange Format wird verwendet, um den Datenaustausch zwischen GPS Apps und Web Services zu ermöglichen (Topografix 2016). Dabei handelt es sich um einen auf XML Sprache basierenden Standard (ebd.). Der Begriff Standard wird verwendet, wenn sich Formate, durch eine sehr breite Nutzung in der Community einen Namen gemacht haben (de Lange 2013, S. 234). Bei solchen standardisierten Datenformaten kann es sich noch immer um proprietäre und von Softwareherstellern spezifische Dateitypen handeln (ebd.). Erst durch den Prozess der Normierung wird eine Loslösung von den Herstellern ermöglicht, um ein technisches Regelwerk, welches auch De-jure-Standard bezeichnet wird, zu schaffen (ebd.). Eine solche Normierung wird durch eigene Institute, wie dem Open Geospatial Consortium (OGC), durchgeführt (ebd.). Bei der angesprochenen XML Sprache handelt es sich um einen OGC konformen Standard (Open Geospatial Consortium 2016). Die gespeicherten Daten in der GPX Datei, welche durch Global Positioning Systems aufgezeichnet wurden, können somit lizenzfrei verwendet, verändert und weitergegeben werden (Topografix 2016). Die Datenstruktur dieses Formates wird in der Schema Datei definiert, welche nach Vorschlag des W3C strukturiert ist (ebd.). Beim W3C (World Wide Web Consortium) handelt es sich um eine Organisation zur Standardisierung von Methodiken aus dem World Wide Web (World Wide Web Consortium 2016). Die XSD Schema Datei des W3C für das GPS eXchange Format ist am 09. August 2004 in der Version 1.1 veröffentlicht worden (Topografix 2016). Ein Auszug dieser XSD Datei wurde im Kapitel 9 beigefügt. An dieser Stelle sollen die elementaren Bestandteile erläutert werden, die zur Analyse, Modellierung und Geosimulation in dieser Arbeit verwendet wurden. Je nach Anforderung des Gerätes und des Benutzers werden die Koordinaten der GPS Position in einem Waypoint, einer Route oder in einem Track gespeichert. Eine einzelne Positionsmessung wird im Waypoint Element <wpt> gespeichert. Hingegen kann das Route Element <rte> zum Navigieren verwendet werden, da mehrere Positionen als Ziel und Zwischenziel verortet sind. Das Track Element <trk> wird im Zuge dieser Arbeit bei den empirischen Felddaten herangezogen, um mehrere aufeinander abgelegte Punkte als Trackpoints zu speichern. Im Tag der Points <trkpt>

müssen bzw. können weitere Attribute ausgewiesen werden. Die geographischen Koordinaten werden als Grad mit Dezimalgrad in der Nachkommastelle, basierend auf dem WGS84-Referenzellipsoid, angegeben (Abbildung 2-2). Es können unter den Extensions aber auch noch Athleteninformationen, wie die Herzfrequenz <gpxtpx:hr>, Trittfrequenz <gpxtpx:cad> oder Außentemperatur <gpxtpx:atemp> an der aktuellen Position, gespeichert werden.

```
<trkpt lon="14.26149945706129" lat="46.61632848903537">
 <ele>455.0</ele>
 <time>2014-06-29T05:44:43.000Z</time>
 <extensions>
  <gpxtpx:TrackPointExtension>
   <gpxtpx:hr>147</gpxtpx:hr>
   <gpxtpx:atemp>19.0</gpxtpx:atemp>
   <gpxtpx:cad>85</gpxtpx:cad>
  </gpxtpx:TrackPointExtension>
 </extensions>
</trkpt>
```

Abbildung 2-2: Beispiel eines GPX Trackpoint Elementes

Durch die Speicherung der Seehöhe <ele> und der Timestamps <time> kann die Geschwindigkeit über die 3D Strecke zwischen zwei Trackpoints berechnet werden. Die Manipulation dieser Timestamps <time> (Abbildung 2-2) ist zentraler Bestandteil der vorliegenden Arbeit, um die Geschwindigkeit in den Segmenten, je nach Leistungsparametern der Athleten zu modellieren bzw. zu simulieren.

2.4.3 Airborne Laserscan Daten (ALS)

Die Höheninformationen zur Radstrecke des Ironman Austria sind ein essentieller Bestandteil dieser Arbeit und somit erfolgt ein kurzer Überblick zur Erfassung von dreidimensionalen Daten. Airborn Laserscanning (ALS) ist eine Methode aus der Fernerkundung, welche mittels Laserstrahlen, die aus einem Flugzeug gesendet werden, die Erdoberfläche erfasst (de Lange 2013, S. 201–202). Das konkrete Verfahren zur Erzeugung eines digitalen Geländemodells (DGM) heißt Light detection and ranging (LiDAR) und funktioniert über die Zeitmessung vom Absenden des Lichtstrahles bis hin zum Eintreffen von diesem (ebd.). Dieses Echo des Lichtstrahles kann als first Pulse erfolgen, um etwa Baumkronen zu reflektieren und somit ein digitales Höhenmodell (DHM) mit Objekten auf der Erdoberfläche zu generieren (ebd.). Beim last Pulse Echo wird hingegen die reine Geländeoberfläche abgetastet, um ein DGM zu erzeugen (ebd.). Damit nun die Lage- und Höheninformationen berechnet werden können, muss das Flugzeug im dreidimensionalen Raum eindeutig über die Differential Global Positioning

System (DGPS) Methode erfasst werden (ebd.). Diese topographische Oberflächenbeschreibung wird bei den verwendeten DGM-Datensätzen (Amt der Kärntner Landesregierung 2016) in ASCII Zeichenkodierung gespeichert. Das Esri ASCII Raster Datenformat (ASC) besitzt, über den Header, die Informationen zu den Zellen und anschließend die eigentlichen Höheninformationen jeder Zelle (Esri 2016b) (Abbildung 2-3). Die Auflösung der Laserscan Daten beträgt laut Header Cellsize einen Meter. Dies bedeutet, dass Höhenangaben in diesem regelmäßigen Abstand entlang der Radstrecke verfügbar sind.

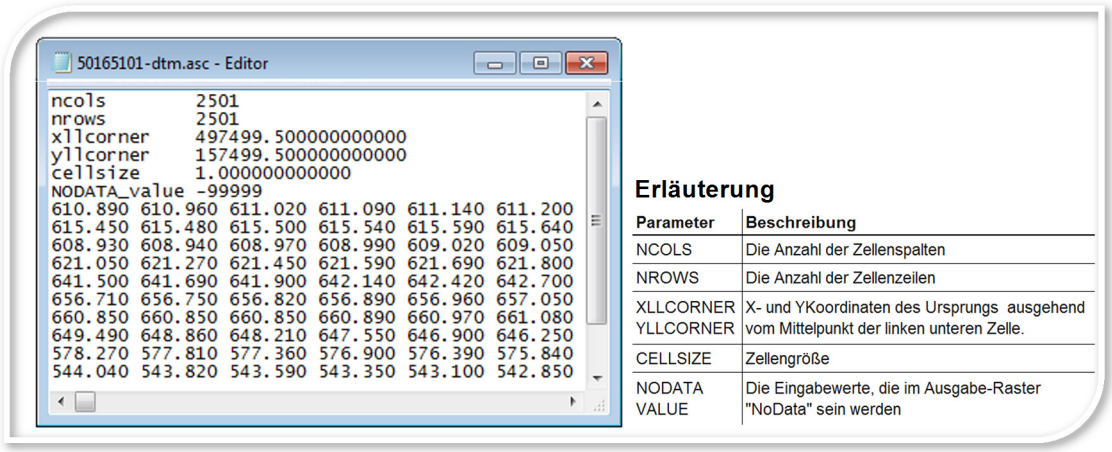


Abbildung 2-3: Höheninformationen des Auswahlgebietes im ASCII Datenformat
Eigene Bearbeitung. Quelle der Erläuterung: (Esri 2016b) Datenquelle: (Amt der Kärntner Landesregierung 2016)

Die Esri ASCII Raster Rohdaten wurden so detailliert beschrieben, um zu verdeutlichen, dass eine geostatistische Analyse damit noch nicht möglich war. Es folgte eine Konvertierung in Terrain- und TIN-Datensätze, um Geländesteigungen in Prozent zu errechnen bzw. um die Höheninformationen mit den GPX Felddaten zu verschneiden.

3. Modellbildung auf Basis von empirischen Wettkampfaufzeichnungen

Dieses Kapitel verfolgt den Zweck, auf Basis der empirischen Felddaten und über induktive Methoden ein Modell zum typischen Fahrverhalten von Athleten zu bilden. Dazu erfolgt die strukturierte Erstellung von reinen Objektgeometrien nach spezifischen Charakteristika, um für die spätere Modellierung und GeoSimulation einen GPX Export zu ermöglichen (Kapitel 3.1). Danach wurde die deskriptive Statistik über die eingeholten empirischen Felddaten verfasst und es folgte eine kritische Beurteilung zur Qualität dieser 107 GPX Tracks (Kapitel 3.2). Die Methoden zur Verknüpfung der Wettkampfaufzeichnungen mit den Objektgeometrien für die Induktion eines Modells wurden im Kapitel 3.3 ausführlich beschrieben.

3.1 Geometrische Modellierung der Radstrecke

3.1.1 Theoretische Vorüberlegungen zur Modellierung

Der Begriff des Modells bzw. der Prozess der Modellierung beschreibt nach Bartelme (2005, S. 43–44) das wissenschaftliche Vorgehen mit dem Ziel, die Wirklichkeit zu abstrahieren. Diese Modelle dienen als Hilfskonstrukt, um in einer virtuellen Umgebung realweltliche Probleme abbilden zu können und mittels Analysen bzw. Simulationen zu vereinfachen oder gar zu lösen (ebd.). Im Zuge dieser Arbeit soll dem Athleten die optimale Rennstrategie über eine Marschroute als Datenmodell vorgegeben werden. Dazu dient der nach Bartelme (2005, S. 8–11) zweckorientierte Prozess der Abstraktion des Raumes als Vorüberlegung. Dieser Prozess beschreibt als ersten Schritt die Identifikation von natürlichen Phänomenen (ebd.). In der praktischen Umsetzung bedeutet dies für die Master Thesis, dass nicht das gesamte Bearbeitungsgebiet mit allen Straßen bzw. topographischen Informationen abgebildet wird, sondern dass nur jene Bereiche einbezogen werden, die für die Beantwortung des realweltlichen Problems von Relevanz sind. So wird nach dem Prinzip der Generalisierung weder die Wechselzone (vom Schwimmen zum Radfahren bzw. vom Radfahren zum Laufen), noch die Penalty Box (Einrichtung entlang der Radstrecke zum Absitzen der Strafe des Wettkampfrichters) modelliert. Nach Bartelme (2005, S. 8–11) werden beim Prozess der Modellierung die realweltlichen Phänomene thematisch, räumlich und zeitlich klassifiziert. Für die Beantwortung der wissenschaftlichen Fragestellung bedeutet dies, dass als Thema die

Fahrstrecke im Untersuchungsraum des Ironman Austria und die Fahrgeschwindigkeiten der Athleten als zeitliches Phänomen abstrahiert werden. Eine weitere Anforderung ist es, ein dynamisches Modell zu erzeugen, welches im Vergleich zu einem statischen Modell keine einzelne Momentaufnahme abbildet, sondern vielmehr den Aspekt der zeitlichen Veränderung von Geodaten berücksichtigt. Durch die dynamische Adaptierung des Modells können die sich ständig verändernden Geschwindigkeiten des Athleten über unterschiedlichste Positionen im Raum repräsentiert werden. De Lange (2013, S. 354–357) stellt einen Vergleich von Vektor- und Rasterdaten dar, welcher für die Wahl des Datenmodells zur Beantwortung der wissenschaftlichen Fragestellung herangezogen wurde. Demnach zeichnen sich Rasterdatenmodelle durch eine trivialere Datenstruktur aus und sind bei der Erstellung hinsichtlich der Geometrie und Topologie einfach zu handhaben (ebd.). Neben dem Nachteil der weniger ansprechenden kartographischen Präsentation von Rasterdaten waren es vor allem Gründe der hohen geometrischen Exaktheit, der besseren Speichereffizienz und eindeutigeren Objektbeschreibung (ebd.), die für die Verwendung des Vektordatenmodells sprachen. Die geometrischen Informationen werden bei Vektordaten in einem eindeutigen Koordinatensystem definiert und in einer eigenständigen Tabellenstruktur abgespeichert (de Lange 2013, S. 346–351). Als Grundelement der Geometrie gilt dabei der Punkt, welcher als Vektor zu interpretieren ist, da der Anfangspunkt im Koordinatenursprung des Bezugskoordinatensystems liegt (ebd.). Der Punkt wird hinsichtlich der Objektdimensionalität als 0-dimensional klassifiziert, obwohl diese Dimension in der realen Welt nicht existiert und daher als Objekt mit geringer räumlicher Ausdehnung oder als Stichprobe entlang einer Linie angesehen wird (ebd.).

Im Zuge der Masterarbeit soll ein Vektordatenmodell entstehen, welches exakt diese Datenpunkte entlang einer Linie als repräsentative Stichprobe enthält. Als Linie ist dabei die Fahrstrecke anzusehen, an der in regelmäßigen Abständen die Stichproben als Punktwerte aggregiert werden. Zur Beantwortung der wissenschaftlichen Fragestellung wird eine Marschroute herangezogen, die in einem GPX Datenformat gespeichert ist und als Geometrie 0 dimensionale Punktobjekte besitzt. Nach Booth und Mitchell (2001, S. 55–56) kann die zu modellierende Objektgeometrie noch exakter als Mult-Point Geometrie klassifiziert werden. Dies ist die Bezeichnung für thematisch zusammenhängende Punktmengen, die pro Datensatz mehrere Koordinaten Tupel speichern können (ebd.).

Die Aneinanderreihung von gerichteten Vektoren wird in der Objektmodellierung als Linienzug definiert (de Lange 2013, S. 347–349). Als Ausgangsbasis ist ebenfalls die Punkt Geometrie eine Voraussetzung, da diese den Anfangs- und Endpunkt als Knoten (engl.: node) der Linie darstellt (de Lange 2013, S. 350). Der Streckenverlauf des Linienzuges, auch Polylinie genannt, wird durch den Einsatz von Punkten namens Vertices geregelt und diese sind wiederum durch Kanten (engl.: arc) miteinander verbunden (ebd.). Eine Polylinie ist hinsichtlich der Objektdimensionalität als eindimensional einzustufen, da die Länge der Segmente gemessen werden kann (Bartelme 2005, S. 68,98). Die 2,5 Dimensionalität wird durch das Verschneiden der Polylinie mit Geländeinformationen erreicht und die Dreidimensionalität wird erst durch die Modellierung von Volumskörpern möglich (ebd.).

Diese theoretischen Grundlagen führen zu einer weiteren Objektgeometrie mit der Kategorie Linienzug, die im Zuge der Masterarbeit modelliert werden soll. Diese Polylinie dient als Geometrievorlage, um die Fahrstrecke der Athleten aus der Wirklichkeit in ein Modell zu abstrahieren. Der Linienzug wird erstellt, um Analysen hinsichtlich von Kurvenradien der Radstrecke und dessen Einflüssen auf das GeoModell durchführen zu können. Dabei soll eine 2,5 Dimensionalität angestrebt werden, um neben den XY Koordinaten auch noch eine Höheninformation Z zu erlangen. Für die praktische Umsetzung der Marschroute bedeutet dies zuerst eine Erstellung der Polylinie, um im Anschluss, auf Basis von topologischen Beziehungen, ein Multi-Point Feature als GPX Datei zu erzeugen.

3.1.2 Geometrische Datenerfassung

Im Kapitel 3.1.1 wurden theoretische Vorüberlegungen zum Datenmodell angestellt und dabei die erforderliche Objektgeometrie definiert, um die wissenschaftliche Fragestellung zu beantworten. Als nächster Schritt erfolgt die Datenerfassung, also die praktische Umsetzung des Geometriemodells mit der Abstrahierung der Fahrstrecke als Polylinie. Dabei ist nach de Lange (2013, S. 177,180,181) zwischen der Erfassung von Primär- und Sekundärdaten zu differenzieren, da diese beiden Typen unterschiedliche Methoden erfordern. Unter dem Begriff Primärdaten werden Datenerfassungen bzw. Datenmessungen verstanden, die noch nicht vom Computerbenutzer weiter verarbeitet wurden (ebd.). Eine Methode zur Erfassung von Primärdaten ist die manuelle On-Screen Digitalisierung, welche auf Basis eines Rasterbildes erfolgt (ebd.). Für die Erstellung der Fahrstrecke der Athleten beim Ironman Austria würde diese manuelle On-Screen

Digitalisierung auf der Kartengrundlage eines Orthophotos geschehen. Bei dieser klassischen Methode der Bildschirmdigitalisierung würde im Orthophoto der Straßenverlauf als Polylinien Geometrieobjekt nachgezeichnet werden. Gegen diese Methode spricht hingegen ein gewisser Ungenauigkeitsfaktor, der durch die manuelle Tätigkeit des Benutzers entsteht. De Lange (2013, S. 182) spricht bei dieser Methode sogar von einer groben und eckigen Datenerfassung, welche erst durch die weitere Verarbeitung mittels Glättungsalgorithmen über Grainwerte, sprich Körnungen, modellhafter wird. Aus den Gründen der Unschärfe wurde daher diese Datenerfassungsmethode verworfen und der Fokus auf die Verwendung von Sekundärdaten gelegt. Nach de Lange (2013, S. 177,178,179) werden Sekundärdaten erst durch Ableitungen und Veränderungen auf Grundlage von Primärdaten gewonnen. Ziel ist es, durch geometrische Konstruktionsmethodiken auf Basis der Primärdaten neue Objektgeometrien zu generieren (ebd.). Eine Datenerhebungsmethode für Sekundärdaten ist jene der zeitlichen- und räumlichen Diskretisierung, bei der laufend Werte erhoben und in Objektgeometrien gespeichert werden (ebd.).

Für die Datenerhebung der Fahrstrecke sind solche diskreten Punktgeometrien in Form von empirischen Felddaten von 107 Athleten des Ironman Austria Wettkampfes bereits vorhanden. Die weitere Überlegung ist dahingehend ausgerichtet, dass aus jedem dieser Datensätze in Form von Multipoint Features, eine Polylinie erzeugt wird. So werden aus über 1,6 Millionen diskreten Datenpunkten 107 Polylinien erstellt, welche jeweils die Fahrstrecke eines Athleten darstellen. Nun soll nach Vorlage der wissenschaftlichen Arbeiten von Brunsdon (2007, S. 1–9) und Thieler et al. (2013, S. 1–52) aus diesen verschiedenen Punkt Datensätzen eine einzelne Polylinie entstehen, die dem Mittelwert aller Einzelpunkte entlang der Radstrecke entspricht und somit eine „durchschnittliche“ Fahrstrecke widerspiegeln soll. Die Machbarkeitsstudie von Brunsdon (2007, S. 1–9) wurde herangezogen, da wie in der Master Thesis auch hier GPS Bewegungsdaten im GPX Format verwendet wurden. Bei den aufgezeichneten Daten handelt es sich um diskrete Punktdaten vieler Wanderungen der gleichen Route. Auf Basis des „Basic Principal Curve“ Algorithmus mit einigen Modifikationen, um systematische GPS Messfehler zu ignorieren wurde ein durchschnittlicher GPS Track aus einer Punktwolke generiert. Nach Angaben des Autors ist die generierte Mittelwerts Linie, verglichen mit einer tatsächlichen Referenzlinie, durchaus valide und brauchbar. Diese Überprüfung der Ergebnisse wird nach Berechnung der Mittellinie zur Fahrstrecke des Ironman Austria im Zuge dieser Masterarbeit ebenfalls mit einer Referenzlinie verglichen (Abbildung 3-1). Eine weitere Studie, die sich mit dem Erstellen von Mittellinien zur sekundären

Datenerfassung beschäftigt, ist jene von Thieler et al. (2013, S. 1–52). Diese wissenschaftliche Publikation wurde herangezogen, da die Organisation USGS (United States Geological Survey), aus einer Big Data Studie von über 26.000 Küstenlinien vom Jahr 1844 bis 2013, eine durchschnittliche Küstenlinie erzeugt hat. Unter der Einbeziehung von GIS Methodiken gibt diese Arbeit einen Aufschluss über die Veränderungen der Küstenverläufe, welche als Polyline Geometrien modelliert wurden, um schlussendlich Erosionsprozesse erklären zu können. Diese Verwendung von enormen Datenmengen deutet auf eine brauchbare Nutzung dieser Methode hin, um die durchschnittliche Fahrbahn von über 100 empirischen Felddaten der Athleten zu generieren. Die Durchführung dieser Datenerfassungsmethode, auf Basis der GPS Bewegungsdaten beziehungsweise eine visuelle Validierung dieser Daten wird in der Abbildung 3-1 diskutiert.

Eine weitere Methode zur Erfassung von Sekundärdaten ist jene der Digitalisierung auf Basis von bereits bestehenden Primärdaten. Ein für die Erhebung der Radstrecke relevanter Primärdatensatz ist das Straßenfeature des OpenStreetMap (OSM) Projektes. Nach de Lange (2013, S. 228–229) werden die Geodaten zu diesem Projekt, durch unentgeltliche Arbeitsleistungen von freiwilligen Nutzern erstellt bzw. modifiziert und aktuell gehalten. Die Datenerfassung der User geschieht über die Verwendung von GPS Geräten, beziehungsweise über die Auslesung dieser Daten und den anschließenden Upload auf einer Web Plattform (ebd.). Die Daten werden schlussendlich in eine OpenStreetMap-Datenbank eingespielt, können alternativ aber auch durch manuelle Digitalisierung auf Kartengrundlagen erhoben werden (ebd.). De Lange (2013, S. 230) weist jedoch zudem auf eine inhomogene Datenqualität hin, welche in Hinblick auf auf Datenverfügbarkeit und Lagegenauigkeit, je nach zweckmäßiger Verwendung, unterschiedlich brauchbar sein kann. Die Polyline Geometrien der OpenStreetMap Datenbank wurden als Shape Format für die Bildung von Sekundärdaten herangezogen. In der Abbildung 3-1 wurden alle Datenerhebungsmethoden visualisiert und im Anschluss auch kritisch beschrieben, um die optimalste Geometriemodellierung zu verwenden. Eine weitere Grundlage zur Erstellung von Sekundärdaten sind die Geobasisdaten der Graphenintegrations-Plattform, kurz GIP. Diese Verkehrsdaten werden durch die Behörden eingespielt und miteinander auf einer gemeinsamen Plattform geometrisch und semantisch vernetzt (Heimbuchner 2014, S. 467). Das seit dem Jahr 2001 laufende Projekt bietet sehr detaillierte Verkehrsgraphen an, welche sogar dem Anspruch rechtsverbindlicher Verwaltungsauskünfte gerecht werden (ebd.). Genau diese Exaktheit hinsichtlich der geometrischen Lage der Polylinien wird in der vorliegenden

Masterarbeit aufgegriffen. Alle in diesem Kapitel angesprochenen Methoden zur Datenerfassung wurden umgesetzt und dargestellt (Abbildung 3-1).

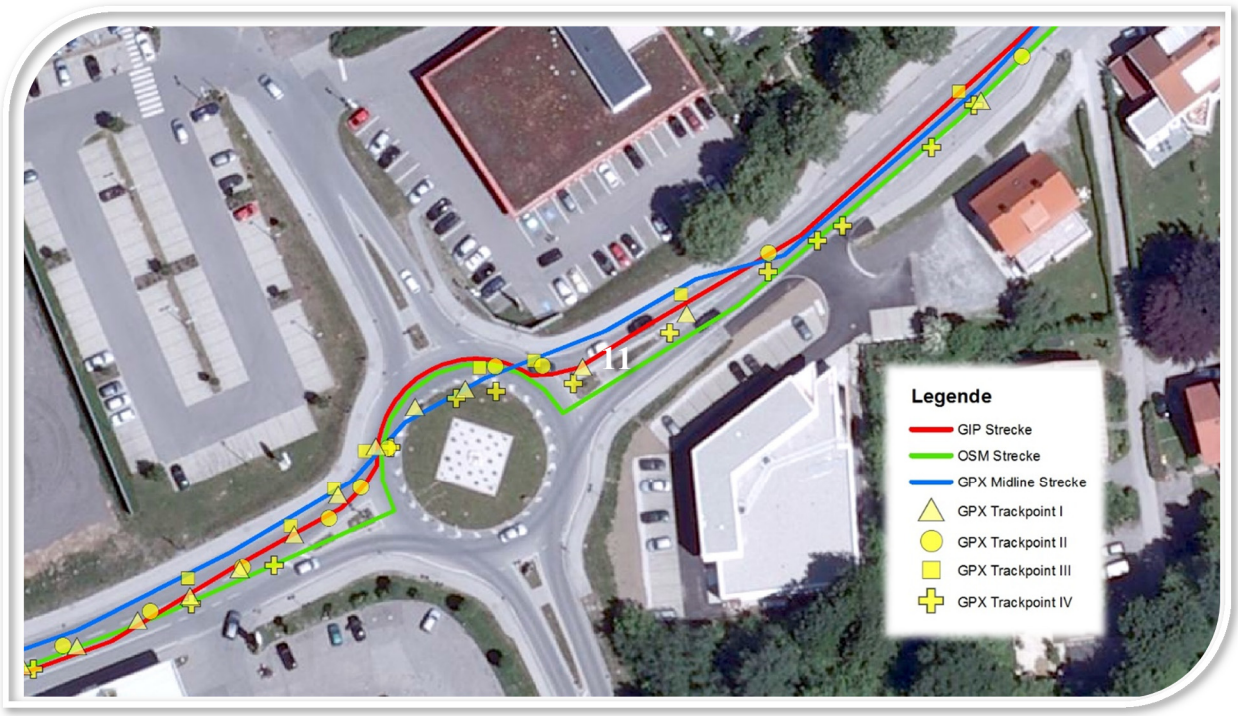


Abbildung 3-1: Visuelle Überprüfung der Methoden zur geometrischen Datenerfassung
Eigene Bearbeitung. Datenquellen: OpenStreetMap Daten:(Geofabrik 2016)
GIP Daten: (Amt der Kärntner Landesregierung 2016)

Die obige Abbildung 3-1 zeigt einen Vergleich der in diesem Kapitel angesprochenen Geometrie-Erfassungsmethoden. Auf Basis des Orthophotos (Quelle in Arbeit: Amt der Kärntner Landesregierung 2016) soll nun eine visuelle Validierung der Daten geschehen. Dabei wurde mit dem Ausschnitt des Kreisverkehrs in Selpritsch ein anspruchsvolles Geometrieobjekt gewählt, welches aus der Realität abstrahiert werden soll. Die symbolisierten GPX Trackpoints I bis IV sind eine visuelle Auswahl von den empirischen Felddaten der GPS Dateien des Ironman Austria Wettkampfes. Jede Point Geometrie steht für eine diskrete Messung an der jeweiligen Koordinate. Diese Punkte wurden nach Athleten zu Polylinien verbunden, um somit reale Fahrstrecken zu erhalten. Auf Grund der in Kapitel 2.4.1 angesprochenen Messungsungenauigkeit von GPS Geräten wurde nun aus diesen Polyline Datensätzen eine mittlere Fahrstrecke erzeugt. Für die Berechnung dieser „Average Line“ wurde das Software Produkt Topofusion (Morris 2016) mit dem Network Feature verwendet und als blaue GPX Midline Strecke auf dem Orthophoto visualisiert. Diese Fahrstrecke ist nach visuellen Überprüfungen mit dem Orthophoto die inexakteste Datenerhebungsmethode. Speziell im Bereich des Kreisverkehres, welcher sogar „durchfahren“ wird, ist die durchschnittlich generierte Fahrlinie aus den GPX Tracks für eine weitere Verwendung im Zuge der Masterarbeit

nicht brauchbar. Die Straßen Features des OpenstreetMap Datensatzes wurden als grüne Linie in der Abbildung 3-1 dargestellt und weisen ebenfalls im Bereich vor dem Kreisverkehres Durchschneidungen des Fahrbahnteilers auf. Der Datensatz der Graphenintegrations-Plattform (rote Linie) modelliert die Kreisverkehrsituation unter Berücksichtigung von Fahrspuren aus rein visuellen Bewertungskriterien am exaktesten. Nun gilt es, aus diesem großen GIP Basisdatensatz des Verkehrsnetzes von Österreich lediglich die Geometrie der Radstrecke des Ironman Austria zu extrahieren. Dieser Prozess der Sekundärdatenerfassung wurde mittels Verfolgungsmethodik (engl. Tracing) umgesetzt. Beim Digitalisierungsprozess kann somit auf Basis des GIP Datensatzes eine Linienverfolgung verwendet werden, um nur die Geometrie der Fahrstrecke zu erhalten. Bei der Route des Ironman Austria handelt es sich um einen geschlossenen Rundkurs, der über 3,6 km hinweg Gegenverkehrsabschnitte aufweist (Abbildung 2-1). Diese Abschnitte sind für die Athleten speziell gekennzeichnet (Abbildung 3-2 rechts), da hier aus Sicherheitsgründen ein striktes Überholverbot gilt, dessen Nichteinhaltung sogar mit Rennausschluss geahndet wird. Bei der Modellierung der Geometrie zur Radstrecke wird auf solche Segmente ebenfalls besondere Rücksicht genommen, da durch Verschiebungen (Offsets) eine zweite Gegenfahrspur, auf Basis der GIP Daten, modelliert werden muss (Abbildung 3-2).

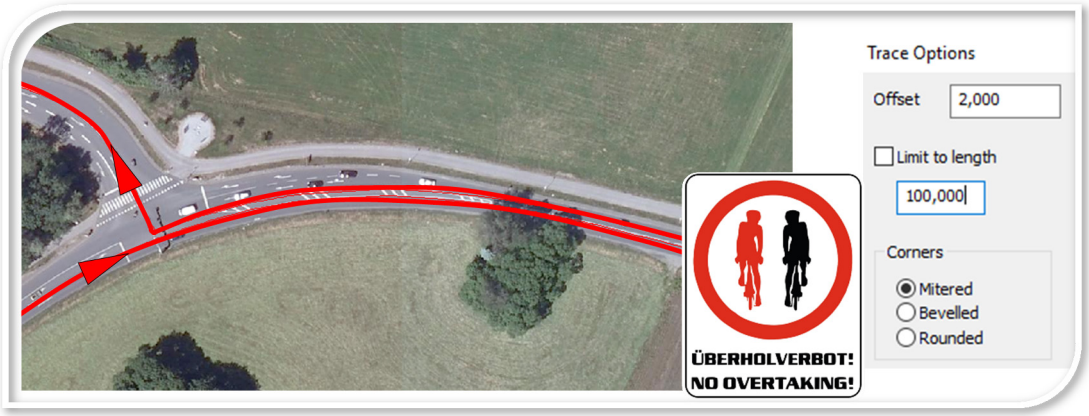


Abbildung 3-2: Modellierungsmethodik der Linienverfolgung im Gegenverkehr
Eigene Bearbeitung. Bildquelle (rechts): (World Triathlon Corporation 2016)

3.1.3 Erstellung von semantischen 2,5 dimensionalen Polylines und GPX Points

Bei dem Begriff Semantik handelt es sich nach (Bartelme 2005, S. 179–180) um eine von zwei Säulen, auf die das georationale Modell aufsetzt. Einer dieser Bestandteile ist die in Kapitel 3.1.1 angesprochene und umgesetzte Geometrie von Punkten, Linien und Flächen (ebd.). Bei der semantischen Modellierung wird der Geometrie eine inhaltliche Bedeutung beigemessen, um dem Benutzer, speziell bei der thematischen, kartographischen Visualisierung, ein besseres Raumverständnis vermitteln zu können (ebd.).

3.1.3.1 – Verschneidung mit Gemeindeflächen

Die Radstrecke des Ironman Austria (Abbildung 3-4, I a) soll mit Raumeinheiten verschnitten werden, um für kartographische Darstellungen bzw. Segmentbeschreibungen dem Leser eine bessere Verortung geben zu können. Die Semantik wird auch bei der Marschroute des GPX Tracks Verwendung finden, um dem Athleten durch topographische und örtliche Hinweise eine bessere Orientierung und Zuordnung der rein geometrischen Segmente zu ermöglichen. Für eine räumlich-territoriale Zuweisung wurden die Polygonflächen der Katastralgemeinden mit der bereits erzeugten Fahrstrecke verschnitten. Der Begriff „Verschneiden“ bezeichnet eine topologische Methodik, um Liniensegmente mit einer Fläche zu überlagern und im Anschluss die topologische Abfrage „ist beinhaltet“ (engl.: is contained) durchzuführen (Esri 2016e, S. 1). Das Ergebnis dieser topologischen Verschneidung ist eine nach Gemeindegrenzen segmentierte Polylinie mit dem Ortsnamen in der Attributtabelle (Abbildung 3-4, I b).

3.1.3.2 – Interpolation mit DGM

Bei der Streckenbeschreibung sind für den Athleten, neben der angrenzenden Orte, aber auch zusätzlich topographische Informationen zu Anstiegen und Abfahrten von Relevanz. Auch für die geostatistischen Analysen muss aus der 2 dimensionalen Polylinie der Fahrstrecke erst eine 2,5 dimensionale Geometrie entstehen, um die Höheninformationen zu erlangen. Umgangssprachlich werden Daten, welche Höheninformationen beinhalten, automatisch mit dreidimensionalen Daten gleichgesetzt. Nach Turner (1997, S. 54) werden bei der Terrain Modellierung entweder Vektor - oder Rasteroberflächenmodelle erzeugt, welche jedoch keine Volumensberechnungsfunktionalität ermöglichen (ebd.). Tatsächliche dreidimensionale Daten können eine Analyse bezüglich des Volumens

umsetzen (ebd.). Deshalb wird die Fahrstrecke als 2,5 dimensional definiert und das im Anschluss zu erstellende GPX Multipoint Feature ebenfalls als 2,5 dimensionale Geometrie mit der Höhe als Attributwert beschrieben.
Die Höheninformationen für die Erstellung einer 2,5 dimensionalen Polylinie werden aus dem Digitalen Geländemodell (DGM) entnommen. Das DGM als Modell der natürlichen Geländeoberfläche beinhaltet flächendeckende Höheninformationen (Kapitel 2.4.3).
Die Integration von Höheninformationen in die erstellte Fahrstrecke des Ironman Austria erfolgt über die Methodik der Interpolation von Z-Werten (Abbildung 3-4, II a). Dabei wird eine Veränderung des Geometrie Typus von einer 2D-Polylinie zu einer Z-Polylinie durchgeführt, um im Anschluss diesen Track auch in einem 3D Viewer, z.B. ArcScene, mit unterschiedlichsten Grundlagedaten, wie Orthophoto, visualisieren zu können. Solch eine perspektivische Darstellung wurde am Türkeihügel, dem Anstieg zum Faakersee, in der Abbildung 3-4, II b veranschaulicht.

3.1.3.3 – Erstellung von GPX Punkt Geometrien entlang der Polylinie

Im Kapitel 3.1.1 wurde darauf hingewiesen, dass einerseits eine Linien-Geometrie der Fahrstrecke und andererseits ein Punkt Geometriemodell erzeugt werden soll. Diese zu erstellenden 3D Points sollen als geometrisches „Füllgefäß“ des endgültigen GPX Files der Marschroute verwendet werden, um in diesen Geoobjekten die Modellierung und die Simulation der Fahrgeschwindigkeiten als Attribut zu verankern. Diese GPX Trackpoints sollen exakt auf der zuvor erstellten Fahrstrecke basieren. Die dazu verwendete topologische Methode ist jene der „Punkte auf Linie“ (engl.: Point on Line) mit der Regel, dass sich die zu erstellenden Punkt Geometrien auf einer Polylinie befinden müssen (Esri 2016e, S. 1). Dabei sollen die Punkte in regelmäßigen Distanzen zueinander liegen, um für die spätere Simulation der Fahrgeschwindigkeiten gleichbleibende räumliche Parameter zu garantieren (Abbildung 3-4, III). Die räumliche Operation wurde mittels FME Transformationswerkzeug „Line Divider“ umgesetzt (Abbildung 3-4, IV), um alle 10 Meter entlang der 2,5 dimensionalen Polylinie der Fahrbahn einen Punkt zu setzen.

3.1.3.4 – Berechnung der Fahrbahnsteigung in Prozent

Im Anschluss wurde eine weitere räumliche Analysemethodik (add surface information) verwendet, um topographische Informationen in die Attributtabelle des Features zu schreiben. Auf Basis des digitalen Geländemodells sollen über Interpolationsverfahren mit den jeweiligen 10 Meter Segmenten statistische Werte, wie absolute, mittlere, minimale, maximale Höhe und die Steigung in Grad ermittelt werden. Diese Höheninformationen werden durch diese Methode in die Attributtabelle geschrieben, um eine Klassifizierung und im Anschluss eine Simulation nach Fahrbahnsteigung zu ermöglichen. Nach Friel und Vance (2013, S. 125) wird die Hangneigung im Radsport üblicherweise in Prozent der Fahrbahnneigung angegeben. Bei der Funktion: „add surface information“ in ArcGIS wird dieses Gefälle jedoch nur in Grad ausgegeben. Um dies zu realisieren, erfolgte der Einsatz folgender mathematischer Formel:

Sei *S* die Steigung in Prozent, *H* die Höhendifferenz in Meter und *F* die Fahrstrecken in Meter.

$$S = 100 \times \tan(\arcsin\frac{H}{F})$$ Quelle: (Esri 2016a)

Diese wurde in einem Script angewandt, um automatisiert über alle Segmente hinweg die Steigung in Prozent zu errechnen (Abbildung 3-4, IV b).

An dieser Stelle wird der Distanzunterschied bei zweidimensionaler Berechnung und dreidimensionaler Strecke über das DGM anhand eines Beispiels verglichen. Das Segment von Kilometer 31,03 bis 31,04 der 2D Polylinie mit den BMN Koordinatenpunkten 496.542,885/161.832,675 Meter und 496.539,194/161.826,801 Meter hat als 2D Strecke eine Distanz von 10 Meter. Dasselbe Segment (nach Koordinaten gemessen) als 2,5D Strecke über dem DGM, besitzt bei 13% Steigung und 1,3 Meter Anstieg, eine Oberflächendistanz von 10,08 Meter (Abbildung 3-4, IV a und IV b).

3.1.3.5 – Topographische Semantik

Wie bereits in Kapitel 3.1.3.1 angesprochen ist die Semantik, also die Rauminformation, für den Athleten von Wichtigkeit, um charakteristische Streckensegmente räumlich verorten zu können. Dies erfolgte in einem ersten Schritt mit einer Verschneidung der Gemeindeflächen und der Fahrstrecke. Durch die Interpolation mit dem DGM und der dadurch ermittelten Höhendaten können nun auch topographische Informationen für die Semantik verwendet werden. Um zu ermitteln, in welchen Segmenten ein signifikanter Anstieg vorhanden ist, wurde ein Höhenprofil der Radstrecke des Ironman Austria generiert (Abbildung 3-3).

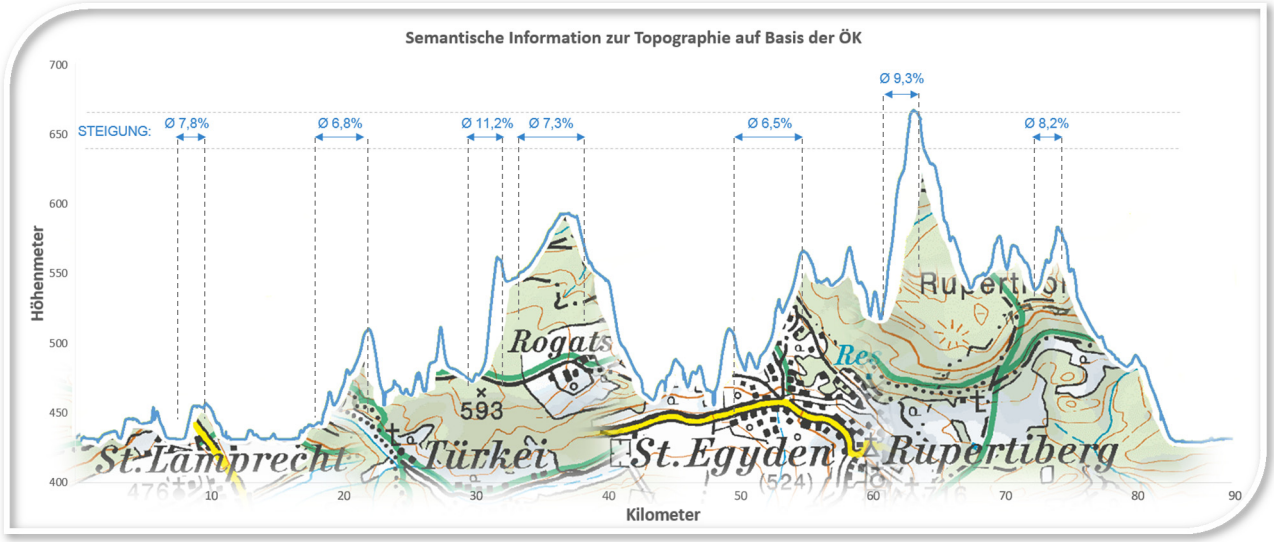


Abbildung 3-3: Semantische Information zur Topographie
Eigene Bearbeitung. Quelle ÖK: (Bundesamt für Eich- und Vermessungswesen 2016)

Aus der Kombination von Erfahrungswerten zur Radstrecke und der visuellen Überprüfung mit dem Höhenprofil wurden insgesamt 11,2km lange Segmente gebildet, die eine Klassifizierung als Berganstieg erhielten. Diese Steigungsabschnitte wurden mit einer kartographischen Österreich Karte (ÖK) als 2,5 dimensionales Oberflächenmodell in einem Viewer (ArcScene) dargestellt (Abbildung 3-4, V). Somit konnte eine klare topographische Semantik aus den Segmenten der Anstiege und der kartographischen Darstellung der ÖK 50.000 gewonnen werden. Diese Informationen wurden in einem eigenen Attributfeld „Semantik“ gespeichert und die entsprechenden 10 Meter Abschnitte z:B. als „Anstieg nach Drobollach“ bezeichnet.

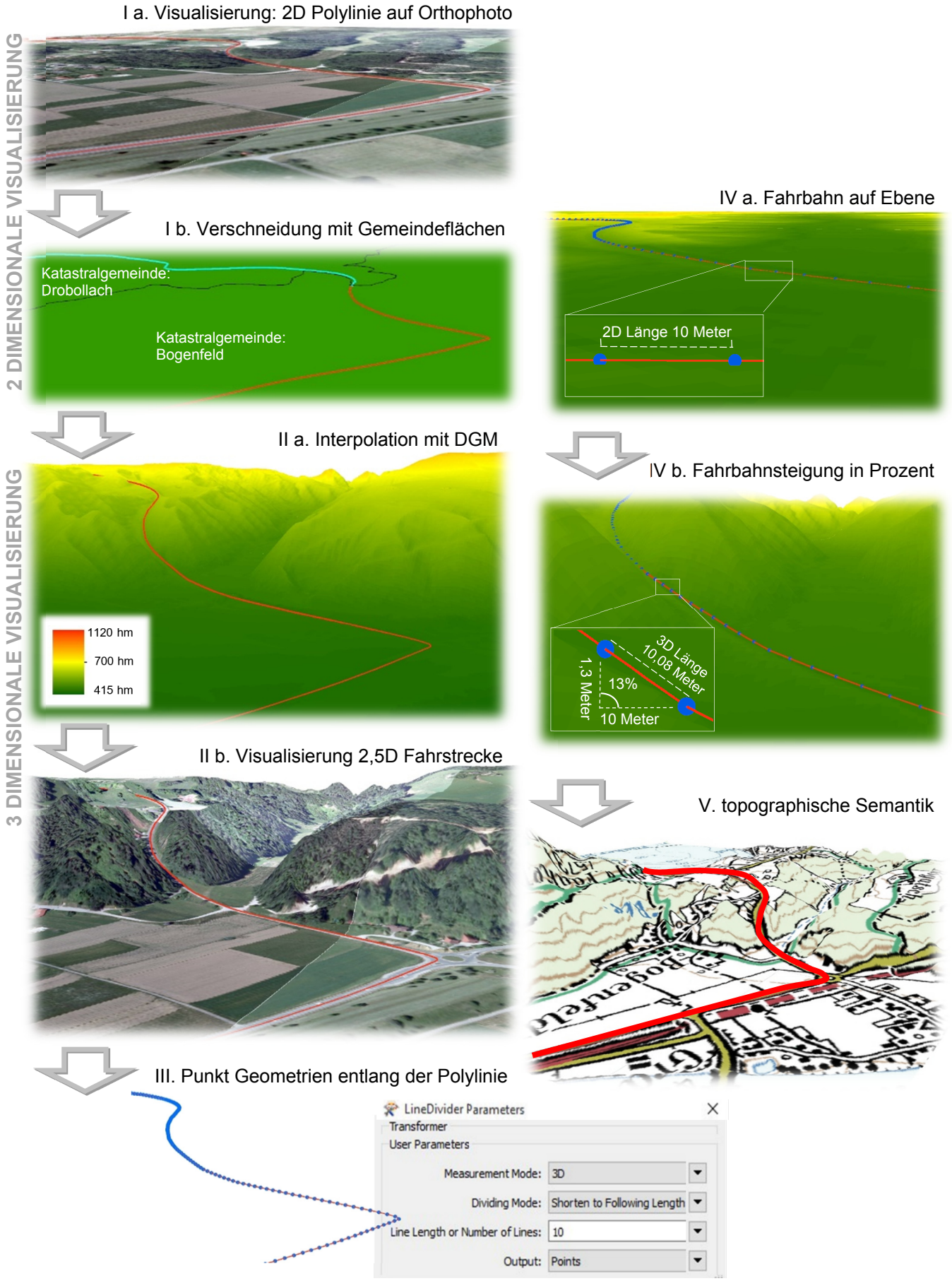


Abbildung 3-4: Geometrischer Modellierungsprozess

3.1.4 Koordinative Bezugssysteme und Transformationen

Im Zuge dieser Arbeit wird auf zwei unterschiedliche Koordinatensysteme zurückgegriffen. Bei der Visualisierung bzw. bei der Verarbeitung von GeoProzessen traten des Öfteren Lageungenauigkeiten auf (Abbildung 3-5), die mit den falschen Transformationsparametern zusammenhängen. Auf diese Spezifika soll nun eingegangen werden, um auf die in Kapitel 5 verwendeten Geoverarbeitungswerkzeuge hinzuweisen.

Bei Koordinatensystemen handelt es sich um modellhafte Konstrukte, welche mithilfe der Angabe von Zahlenwerten Geoobjekte in einem Raum verorten. Bei den Koordinaten kann in einem ersten Ansatz hinsichtlich ihrer Dimension differenziert werden. Bei eindimensionalen Koordinaten geschieht die Positionierung lediglich über die Entfernung entlang von Geoobjekten. So kann etwa entlang einer Straße eine Kilometrierung aufgetragen werden, um Geoobjekte zu verorten. Diese Technik wird beispielsweise bei der Methode zur linearen Referenzierung angewandt, welche im Kapitel 4.1 näher beschrieben wird und auch im Zuge dieser Masterarbeit Einsatz findet.

Nach de Lange (2013, S. 141–142) kann weiters zwischen kartesischen Koordinaten und Polarkoordinaten differenziert werden. Bei den kartesischen Koordinaten, welche eher für kleinräumige Bearbeitungsräume geeignet sind, ist für jede Dimension eine Achse reserviert (ebd.). Bei der Verwendung von Geographischen Informationssystemen bilden die kartesischen Koordinaten eine Grundlage bei der Visualisierung von Punkt, Linien, Polygon-Geometrien sowie bei den Rastermodellen (ebd.). Sobald nun ein spezifischer Punkt auf der Erdkugel bestimmt werden soll, kommt das Konzept der Polarkoordinaten zu tragen de Lange (2013, S. 143–144). Dieses basiert auf der Verwendung eines gekrümmten Bezugsmodells, welches die Erde als stark abstrahierte Kugel ansieht (ebd.). Die Position eines Punktes auf dem Bezugsmodell kann nun über die Angabe von zwei Winkeln erfolgen, welche als geographische Breite und Länge bestimmt werden (ebd.). Die folgende Koordinatenangabe bezeichnet den Startpunkt der Radstrecke: Breite: 46°36´41,3´´ Länge: 14°15´37,8´´.

Bei dem GPX Datenformat der empirischen Felddaten wird bei der Koordinatenangabe nach der XSD Schema Datei das World Geodetic System (WGS) aus dem Jahre 1984 verwendet:

```
<xsd:attribute name="lat" type="latitudeType" use="required">
<xsd:annotation>
<xsd:documentation>
The latitude of the point. This is always in decimal degrees, and always in WGS84 datum.
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
```

Dabei erfolgt die Koordinatenangabe nicht in Grad, Minuten und Sekunden, sondern in Dezimalgrad. Für das zuvor genannte Beispiel mit dem Startpunkt der Radstrecke, entspricht dies folgendem GPX Tag:

<trkpt lon="14.260490108281374" lat="46.61146338097751">

Das WGS 84 Datum zeichnet sich weiters dadurch aus, dass nicht die Erdkugel als Bezugsmodell verwendet wird, was lediglich für kleinmaßstäbige Darstellungen brauchbar ist (de Lange 2013, S. 145). Beim WGS 84 als globales geodätisches Datum wird der Bessel-Ellipsoid verwendet und es ist somit auch bei größeren Maßstäben durchaus brauchbar (de Lange 2013, S. 146).

Um in großmaßstäbigen Darstellungen möglichst exakte Ergebnisse und Darstellungen zu erhalten, wird ein lokales Datum verwendet, welches auch bei Landesvermessungen zum Einsatz kommt (ebd.). Die verwendeten Geodaten der Graphenintegrations-Plattform für die Modellierung der Fahrstrecke des Ironman Austria, befinden sich in einem geodätischen Koordinatensystem namens „Österreichisches Bundesmeldenetz“ (BMN). Da dieses Koordinatensystem auf Grund der starken Verwendung in Schulatlanten und Tourismus Karten von Kärnten sehr geläufig ist, wurde dieses auch in den kartographischen Abbildungen dieser Masterarbeit verwendet. Hingegen wirkt das WGS 84 als geodätisches Referenzsystem bei Kartendarstellungen des Wörthersee Raumes sehr verzerrt und unnatürlich.

Zusammenfassend bedeutet dies, dass bei der Induktion, also der Abstrahierung der empirischen Felddaten, das WGS 84 Koordinatensystem verwendet wird. Bei der Visualisierung und bei der geostatistischen Analyse kommt das BMN 31 als geodätisches Koordinatensystem zum Einsatz. Bei den Deduktionsprozessen vom Modell zur Praxis, sprich bei der Geosimulation und der Erstellung neuer GPX Marschrouten, wird wieder das WGS 84 als Bezugssystem herangezogen. Geographische Informationssysteme können diese unterschiedlichen Bezugssysteme der einzelnen Features erkennen und in einer „On The Fly“ Methode, ohne tatsächlich umgerechnet zu werden, parallel

darstellen. Im Fall der beiden verwendeten Bezugssysteme WGS 84 und BMN 31 ist solch eine On The Fly Darstellung mit einer Abweichung von etwa 80 Meter verbunden (Abbildung 3-5). Um die empirischen Felddaten lagegenau mit dem Straßennetz zu überlagern, muss eine spezifische richtungsgebundene Datumstransformation gewählt werden. Der Begriff Transformation beschreibt nach Bartelme (2005, S. 109–113) den Prozess der Rotation, Verschiebung (Translation) und Größenveränderung von Geometrien in unterschiedliche Bezugssysteme. Eine Transformationsmethode ist jene der 3-Paramter Transformation (ebd.), welche auf Basis der geozentrischen Translation aufsetzt (Esri 2016d). Nach Umwandlung der Koordinaten in ein geozentrisches Koordinatensystem erfolgen Verschiebungen in die X-,Y- und Z- Richtung, um die Differenzen der beiden Bezugssysteme auszugleichen (Esri 2016d). Dies wird durch die Translation des Ursprungs beim Basiskoordinatensystem ermöglicht (ebd.). Am Beispiel der Masterarbeit bedeutet dies, dass eine 3-Paramter Transformation in Richtung des Datums von BMN namens „MGI-AT“ zum „WGS84“ Datum mit folgenden Parametern zu erfolgen hat:

-dx=682,000000 dy=-203,000000 und dz=480,000000

Diese 3-Parameter Transformation muss auch beim Export der simulierten Marschroute zu einem GPX File über die Data Interoperability Extension angewandt werden (Kapitel 5.2).

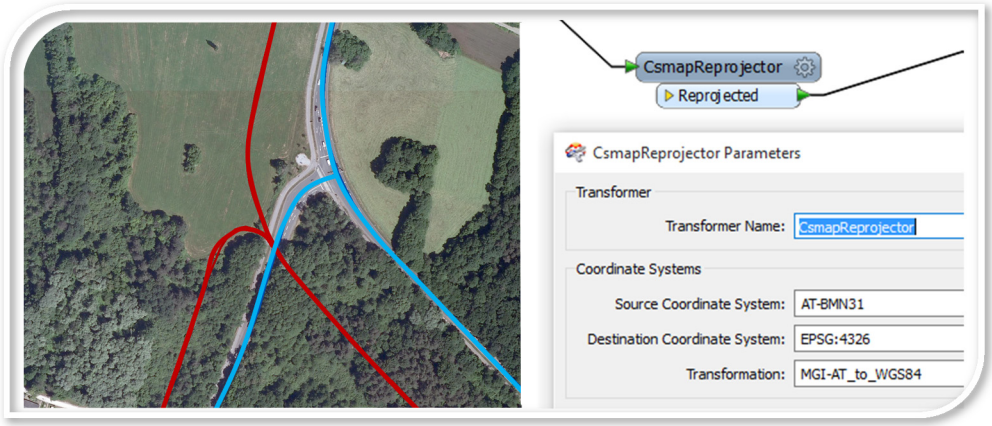


Abbildung 3-5: Lageungenauigkeit WGS 84 und MGI ohne Transformation

Doch auch zur geostatistischen Analyse der GPX Felddaten muss zuvor eine richtungsgebundene Koordinatentransformation namens „WGS84_to_MGI-AT“ verwendet werden, um eine Lageungenauigkeit wie in Abbildung 3-5 auszuschließen.

3.2 Deskriptive Statistik zu den empirischen Felddaten von Athleten

Für den wissenschaftlichen Forschungsprozess der Induktion zu einem Modell ist es notwendig, auf die Empirie aufzusetzen. Kromrey (2002, S. 33–34) bezeichnet als empirische Forschung das Vorgehen nach wissenschaftlichen Regeln, welches das methodische Ansammeln von Daten mit Erfahrungswerten als Ziel hat. Diese Erfahrungswerte wurden anhand von Wettkampfaufzeichnungen beim Ironman Austria in Form von GPX Felddaten eingeholt. Von den insgesamt 2.179 Athleten, welche die Radstrecke in der vorgegebenen Cut-off Time von 10h10min absolviert haben, wurden 622 Athleten kontaktiert und nach GPX Files angefragt. Dies erfolgte über soziale Netzwerke wie Facebook oder über Plattformen wie Garmin Connect, wo Athleten ihre GPS Aufzeichnungen sammeln und teilen können. Insgesamt wurden 107 raum-zeitliche Bewegungsdaten des Ironman Austria übermittelt, was einer Rücklaufquote von 17,2% entspricht wodurch empirische Daten von 4,9% aller Athleten des Wettkampfes verfügbar sind. Bei dieser Erhebung handelt es sich um ein willkürliches Auswahlverfahren, da die empirischen Datensätze ohne Systematik eingeholt wurden. Somit besteht hinsichtlich der Verteilung und Verhältnisse der Felddaten normalerweise kein Zusammenhang mehr zur Grundgesamtheit der Ergebnisse aller Ironman Teilnehmer. Die folgende Abbildung 3-6 und die Lagemaße bzw. Streuungsparameter zeigen jedoch ein anderes Bild, welches der Erhebung einer geschichteten Stichprobenauswahl sehr nahe kommt, da die grundlegenden Verteilungscharakteristika der Wettkampfergebnisse nachgebildet wurden.

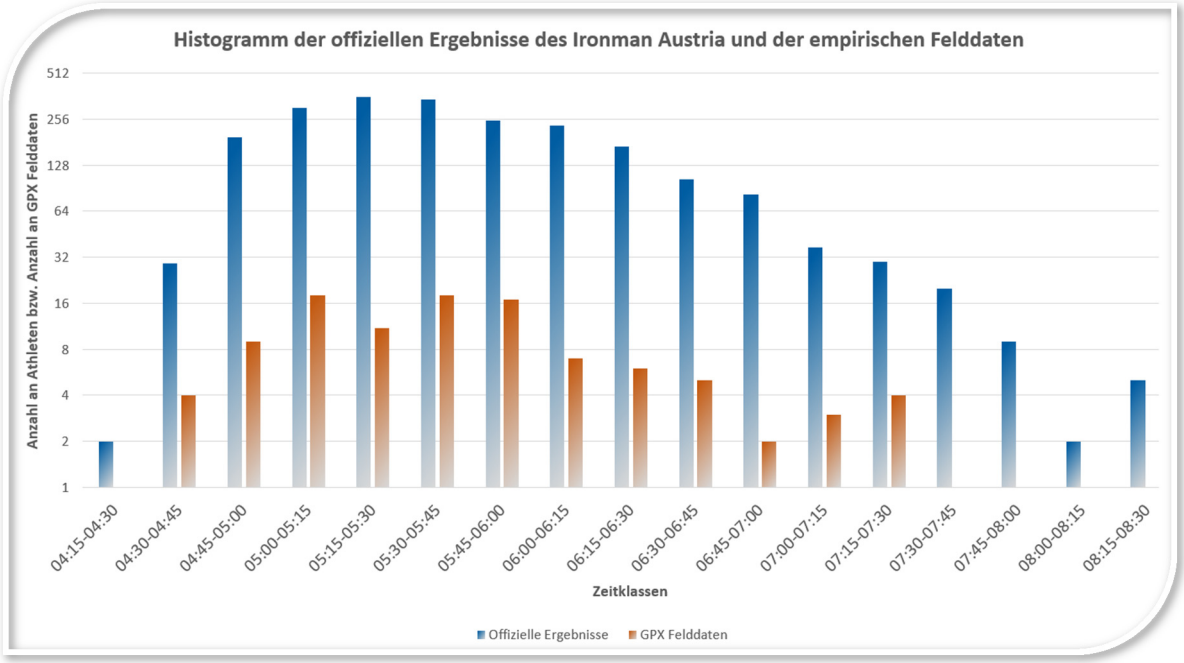


Abbildung 3-6: Histogramm der offiziellen Ergebnisse und der empirischen Felddaten

In der Abbildung 3-6 wurde die Häufigkeitsverteilung aller Athleten des Wettkampfes als logarithmische Skalierung, gruppiert nach Leistungskategorien der Radzeiten in 15 Minuten Abständen, aufgetragen. Im Vergleich dazu sind zudem die 107 erhaltenen GPX Felddaten mit den gleichen Leistungsklassen visualisiert. Bei beiden Datensätzen ist eine rechtsschiefe Normalverteilung (engl. positive skew) vorhanden, was für eine überwiegende Mehrzahl der Datensätze auf der linken Seite (höhere Leistungskategorien) spricht. Auch die meiste Anzahl an Datensätzen in einer Leistungskategorie (offizielle Ergebnisse: Klasse 05h:15m-05h:30m n=356 / empirische Felddaten: Klasse 05h:30m-05h:45m n=18) ist somit links des Mittel- bzw. Medianwertes zu finden.

Kennzahlen der deskriptiven Statistik

	Alle Ironman Ergebnisse	empirische Felddaten
Range	04h:07m:07s	03h:49m:00s
arithmetisches Mittel	05h:45m:22s	05h:46m:12s
Median	05h:37m:19s	05h:40m:00s
Modus	05h:10m:45s	05h:07m:00s
Standardabweichung	00h:39m:38s	00h:44m:10s

Tabelle 3-1: Kennzahlen der deskriptiven Statistik

Die Tabelle 3-1 zeigt einen Vergleich aller Ironman Ergebnisse und der empirischen Felddaten nach Lagemaßen wie Range, arithmetisches Mittel, Median und Modus.

Die Spannweite (engl. Range) fällt bei den erhobenen GPX Daten um 18 Minuten und 7 Sekunden geringer aus, da nicht die gesamte Bandbreite an Leistungskategorien, der

stärksten und schwächsten Athleten, abgedeckt werden konnte. Modus bzw. Modalwert beschreibt den dichtesten und am häufigsten vorkommenden Wert, welcher mit 05h:10m:45s und 05h:07m:00s angesichts der Range von 04h:07m:07s und 03h:49m:00s relativ nahe beieinander liegt. Diese ähnlichen Kennzahlen der Ironman Ergebnisse und der Felddaten sind ebenfalls beim Medianwert, auch 50-Prozent-Quantil genannt, erkennbar. So liegen beim Median von 05h:37m:19s bzw. 05h:40m:00s gleich viele Datensätze über sowie unter dem jeweiligen Wert. Des Weiteren wurde eine relativ ähnliche Standardabweichung als Streuungsparameter errechnet, welche einen Indikator für die Verteilung der Datensätze außerhalb des Zentrums darstellt.

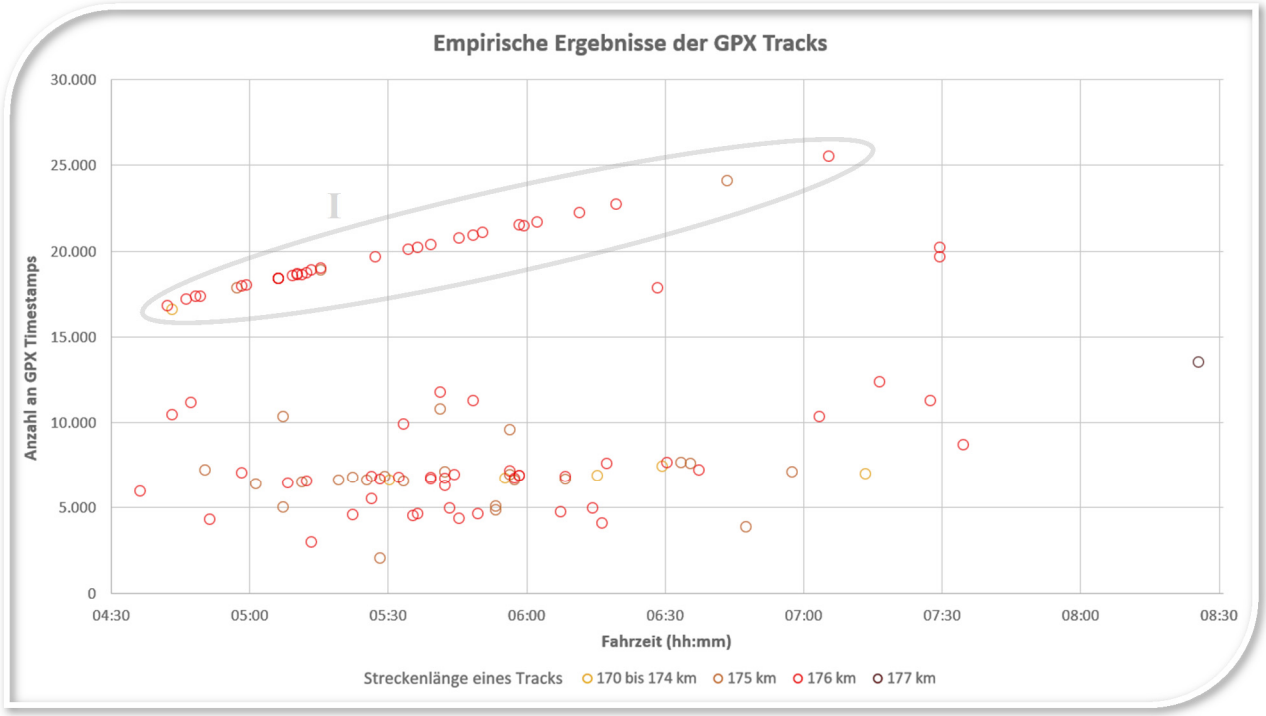


Abbildung 3-7: Streudiagramm der empirischen Felddaten nach Datenqualität

Die Abbildung 3-7 verdeutlicht die Datenqualität der eingeholten GPX Bewegungsdaten anhand der Anzahl von gesetzten Timestamps (Y Achse) und der gemessenen Streckenlänge (vier farbige Klassen) jeder mit GPS Systemen aufgezeichneten Wettkampffahrt. Die Distanz variiert zwischen 170,0 und 177,4 Kilometer. Dies ist in der unterschiedlichen Dauer zur Findung von genügend GPS Satelliten während der Fahrt begründet, sofern das Gerät nicht bereits vor dem Betreten der Wechselzone eingeschaltet wurde. Ein weiteres Merkmal zur Validierung der Datenqualität ist die unterschiedliche Anzahl an aufgezeichneten Datenpunkten (Timestamps), welche durch den jeweils verwendeten Algorithmus zur Datenaufzeichnung zu erklären ist. Die GPX Daten, welche in der Abbildung 3-7 durch ein Oval mit der Nummer I markiert wurden, sind mit jenem Modus aufgezeichnet worden, dass automatisch jede Sekunde ein Datenpunkt gesetzt

wird. Dies begründet auch die lineare Form der Felddaten im Streudiagramm, da umso mehr Datenpunkte aufgezeichnet wurden, je langsamer die Fahrzeit des Athleten war. Die restlichen GPX Files wurden mit einer intelligenten und auch stromsparenden Methode aufgenommen, wo nur bei signifikanter Geschwindigkeitsveränderung mehr Datenpunkte gesetzt werden. Dieser Algorithmus führt somit zu durchwegs weniger Datenpunkten als bei der Aufzeichnungsmethode im Ein Sekunden Intervall.

3.3 Aggregieren der empirischen Felddaten zu einem Modell

3.3.1 Dateninteroperabilität und Datentransformation der GPS Bewegungsdaten

Um eine geostatistische Auswertung der GPX Files von insgesamt 107 Athleten beim Ironman Austria durchführen zu können, müssen diese Daten erst analyse-konform aufbereitet werden. Dies bedeutet neben der Koordinatentransformation, welche in Kapitel 3.1.4 erläutert wurde, auch eine Anpassung hinsichtlich der gespeicherten Attribute, da die Dateninteroperabilität nur eingeschränkt gegeben ist. Der Begriff Interoperabilität bezeichnet nach Bartelme (2005, S. 363) die reibungslose Kommunikation zwischen GIS Programmen und der Möglichkeit des Austausches von GeoDaten, ohne dass der Benutzer über ein spezifisches Know-how verfügen muss. Bei den Felddaten handelt es sich um einen auf XML Sprache basierenden Standard (Topografix 2016). Beim Shape Datenformat, welches von der Firma „Environmental Systems Research Institute“ (ESRI) entwickelt wurde, handelt es sich um einen Quasi-Standard (de Lange 2013, S. 234).

Die beiden Datenformate GPX und Shape sind durch Standardisierungen und Normierungen in technischer Form klar definiert. Trotzdem ist die Interoperabilität zwischen den Formaten lediglich hinsichtlich der Topologie und der Geometriemodelle einfach umzusetzen. Bei der Mehrfachnutzung zur geostatistischen Analyse der Sachdaten in den GPX Files fehlt es dahingegen an der Interoperabilität, um die Attributfelder auch im Shape Format nutzbar darstellen zu können. De Lange (2013, S. 232) begründet diese Inkompatibilität anhand von fehlender syntaktischer Interoperabilität. Darunter wird eine unterschiedliche Struktur im Aufbau der Syntax verstanden (ebd.). Bei einem Import der GPX Dateien in ArcGIS (entweder über File/Add Data oder über die „GPX to Features“ Conversion – ArcToolbox) werden diese angesprochenen Interoperabilitätsprobleme ersichtlich. Es wird zwar die Objektgeometrie als Punkte erkannt und ein Großteil der Informationen aus den Tags

korrekt ausgelesen, jedoch wird der Zeitstempel von einem Datums/Uhrzeit Format in ein String Datenformat umgewandelt. Dies begründet sich darin, dass die Datumsformatierung in der dBASE Datenbanktabelle auf „yyyyMMddHHmmss“ beschränkt ist. Nach der GPX Schema Datei wird jedoch die Angabe von Millisekunden in der Formatierung „yyyy-MM-dd HH:mm:ss,000“ benötigt, um eine exakte Geschwindigkeitsangabe zwischen den einzelnen gesetzten Punkte zu ermöglichen. Diese Angabe der Zeit zwischen den einzelnen Trackpoints ist aber eine Grundvoraussetzung für die weitere geostatistische Analyse. Daher wurde ein Datentransformationswerkzeug namens Feature Manipulation Engine (FME) eingesetzt, um eine Harmonisierung der Datenformate zu ermöglichen.

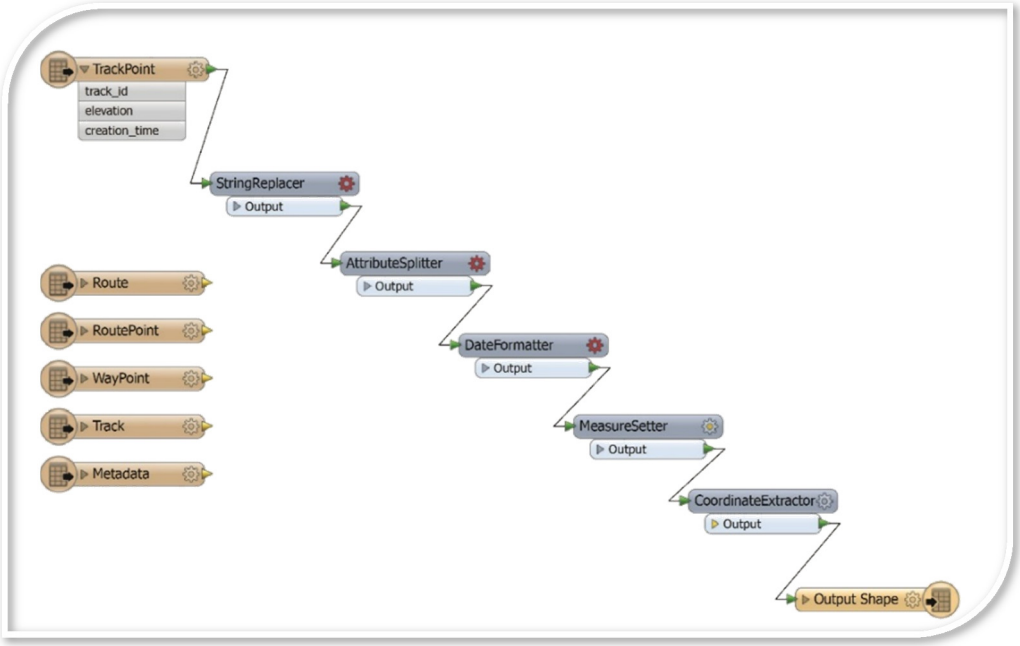


Abbildung 3-8: Workflow zur Datentransformation

Die obige Abbildung 3-8 zeigt den entwickelten Workflow in der FME Workbench Software. Zuallererst wird eine Format Translation durchgeführt, indem die Trackpoints der GPX Felddaten (links oben) zu einem Output Shape File (rechts unten) exportiert werden. Die GPX Route, Waypoints und Tracks bleiben in diesem Diagramm nicht verknüpft, da diese in der Aufzeichnungsmethode nicht verwendet wurden, aber trotzdem in der XSD Schema Datei zur GPX Spezifikation vorkommen (Kapitel 9). Die blauen Elemente, welche zwischen den Import- und Export Daten gekoppelt sind, werden als Transformer bezeichnet. So wurde etwa das ursprüngliche GPX Timestamp Format von „yyyy-MM-dd HH:mm:ss,000“ in ein reines Uhrzeit Format mit „HH:mm:ss,000“ gesplittet. Weiters erfolgte die Übergabe der WGS 84 Koordinaten mit dem Coordinate Extractor in die Attributtabelle der Shape Datei. Dieser Workflow wurde dahingehend ausgebaut, dass schlussendlich die vergangene Zeit in Kombination mit der Distanz,

zwischen den gesetzten GPX Points, als km/h in ein eigenständiges Attributfeld im SHP File geschrieben wurde.

3.3.2 Klassifikation der empirischen Felddaten

Nachdem die GPX Daten als Wettkampfaufzeichnungen des Ironman Austria inhaltlich, geometrisch und lagegetreu aufbereitet wurden, können diese klassifiziert werden. Das systematische Einteilen der Daten in einzelne Klassen wird vorgenommen, um durch die Reduktion von Felddaten einen besseren Überblick zu erzeugen (Zimmermann-Janschitz 2014, S. 91). Dieser Prozess des Zusammenführens von Daten ist jedoch auch mit einem Verlust an individuellen Detailangaben gekoppelt, welche in der Klassifizierung verschwinden (ebd.). Für die Einteilung der Untersuchungsdaten in Kategorien, sollen nach Zimmermann-Janschitz (2014, S. 102) folgende Parameter eine Rolle spielen: Wertebereich x_{min} bis x_{max}, Klassenanzahl k, Klassenbreite b und die untere bzw. obere Klassengrenze x_j^u bzw. x_j^o. Des Weiteren soll in Hinblick auf die Repräsentativität der Daten ein Mindestmaß von 30 Merkmalswerten pro Klasse angestrebt werden (Zimmermann-Janschitz 2014, S. 94). Bei der Bearbeitung der wissenschaftlichen Fragestellung zu dieser Master Thesis ist die Erstellung der Marschroute ein zentraler Aspekt. Da der Benutzer selbst diese Marschroute generieren kann, spielen die zeitlichen Motive der Athleten eine wichtige Rolle (Kapitel 1.3). So stellen gewisse Zeitschwellen für die Athleten, wie etwa die Radzeit mit 5 Stunden und 30 Minuten, eine wichtige Zielvorgabe dar. Daher wurde beim Prozess der Modellbildung versucht, diese zeitlichen Zielvorstellungen durch eine entsprechende Klassifizierung umzusetzen. Auf Grund der Prämisse der vorgegebenen Zeit des Athleten wurden die von Zimmermann-Janschitz (2014, S. 94) angesprochenen Aspekte der Repräsentativität und der Klassifizierungsschritte der Erhebungsgrundlage angepasst. Auf Basis dieser erhobenen Daten und der zeitlichen Zielvorgaben der Athleten wurden die Datensätze wie folgt klassifiziert:

Klassifikation der empirischen Felddaten

empirische Felddaten	Leistungsklassen											
	04:45	05:00	05:15	05:30	05:45	06:00	06:15	06:30	06:45	07:00	07:15	07:30
Minimalwett	04:39	04:50	05:08	05:27	05:37	05:54	06:12	06:29	06:36	06:58	07:14	07:28
Maximalwert	04:48	05:07	05:26	05:36	05:51	06:09	06:20	06:34	06:48	07:06	07:17	07:35
Datensätze in der Klasse	7 (2 Profis)	11	18	12	18	16	6	4	4	3	2	4
Mittelwert	04:45	04:59	05:14	05:31	05:43	05:59	06:16	06:31	06:41	07:02	07:15	07:30
einzelne Zeiten jedes Athleten	04:39 04:44 04:44 04:45 04:47 04:48 04:49	04:50 04:51 04:52 04:58 04:59 04:59 05:00 05:02 05:07 05:07 05:07	05:08 05:08 05:09 05:10 05:11 05:11 05:12 05:12 05:13 05:14 05:14 05:14 05:16 05:16 05:20 05:23 05:23 05:26	05:27 05:27 05:28 05:29 05:29 05:30 05:31 05:33 05:34 05:34 05:35 05:36	05:37 05:37 05:40 05:40 05:40 05:42 05:42 05:43 05:43 05:43 05:44 05:45 05:46 05:46 05:49 05:49 05:50 05:51	05:54 05:54 05:56 05:57 05:57 05:57 05:58 05:58 05:59 05:59 05:59 06:00 06:03 06:08 06:09 06:09	06:12 06:15 06:16 06:17 06:18 06:20	06:29 06:30 06:31 06:34	06:36 06:38 06:44 06:48	06:58 07:04 07:06	07:14 07:17	07:28 07:30 07:30 07:35

Tabelle 3-2: Klassifikation der empirischen Felddaten

Die oben angeführte Tabelle 3-2 zeigt die Klassifizierung der Felddaten des Ironman Wettkampfes in insgesamt 12 Gruppen nach Fahrzeiten, von 4 Stunden und 45 Minuten bis zu 7 Stunden und 30 Minuten. Durch diese Klasseneinteilung sollen typische Fahrverhalten der Athleten in verschiedenen Leistungskategorien zusammengefasst und modelliert werden. In Kapitel 4.3 werden dann Analysen durchgeführt, wie sich typische 5 Stunden Fahrer von 7 Stunden Fahrern bei Steigung und Gefälle oder bei Labestationen unterscheiden. Da bei den einzelnen Gruppen der Durchschnittswert aller enthaltenen GPX Files nicht immer exakt mit der vorgegebenen Zeit übereinstimmt, wurden Ausgleichsparameter über die gesamten Geschwindigkeitsangaben angewandt. So konnte eine gewünschte Zeitvorgabe mit beispielsweise 5 Stunden und 15 Minuten zusammengefasst bzw. modelliert werden, obwohl die Durchschnittszeit der einzelnen GPX Files in der Klasse nur 5 Stunden und 14 Minuten beträgt.

3.3.3 Räumliche Verknüpfung der GPX Daten zu einer Marschroute

Die insgesamt 12 verschiedenen Shape Dateien, klassifiziert nach Fahrzeiten der Athleten, welche die insgesamt 1,6 Millionen GPX Messpunkte der Felddaten beinhalten, sollen nun aggregiert werden. Der Begriff „aggregieren“ bezeichnet den Prozess des Ansammelns von gleichen Objektgeometrien mit ähnlichen Merkmalausprägungen in ein einzelnes, eindeutig bestimmtes Objekt (Esri 2016c). Dies bedeutet, dass die verschiedenen GPX Messpunkte, welche auf Grund von Messungsungenauigkeiten der GPS Technologie (Kapitel 2.4.1) um die erzeugte Fahrstrecke verstreut sind, nach

Fahrgeschwindigkeiten zusammengefasst werden müssen. Die Geometrie, in der die Feldmesspunkte des Wettkampfes zusammengefasst werden, ist jenes Punkt Geometriemodell, welches in Kapitel 3.1.3 erzeugt und erläutert wurde. Diese Punkte, welche in einem 10 Meter Abstand entlang der 3D Fahrstrecke erzeugt wurden, dienen als geometrisches Füllgefäß, um die empirischen Felddaten darin zusammenzufassen bzw. zu modellieren. Nach de Lange (2013, S. 138) wird die vorliegende Situation mit GPX Felddaten und der Marschroute hinsichtlich des topologischen Beziehungstypus als „Punkt zu Punkt Beziehung“ klassifiziert. So sollen die Points der empirischen Messwerte mit den Punkten der Marschroute übereinstimmen. Das Aggregieren wird durch den räumlichen Analyseprozess des „Join“ ermöglicht. Bartelme (2005, S. 327) definiert als „Join“ eine Verbindung von separaten Attributtabellen über einen Primärschlüssel bzw. eine Domäne. Dieser Spezialfall des kartesischen Produktes enthält als 1:1 Verbindung schlussendlich alle Datenfelder, deren Werte im Primärschlüssel ident sind (Bartelme 2005, S. 327). Im vorliegenden Fall der GPX Trackpoints und der Geometrie der Marschroute ist eine 1:n Verbindung notwendig. Dies ist darin begründet, dass den in einem Abstand von 10 Meter liegenden Punkten der Marschroute mehrere Messpunkte der GPX Felddaten zugewiesen werden sollen. Darüber hinaus fehlt für eine klassische „Join“ Verbindung die Existenz einer gemeinsamen Domäne. Diese könnte eventuell durch das Einführen einer Kilometrierung entlang der Fahrstrecke erfolgen, doch scheitert dies an dem unterschiedlichen räumlichen Aufzeichnungsbeginn der GPX Felddaten durch das GPS Gerät. Aus diesem Grund wurde eine Spezialform der räumlichen Analysemethoden angewandt, welche als „Spatial Join“ bezeichnet wird. Bei dieser räumlichen Verknüpfung von Geodaten handelt es sich um eine Kopplung von Attributdaten, welche über die relative Lagebeziehung der Geometriepunkte zueinander geschieht (Abbildung 3-9).

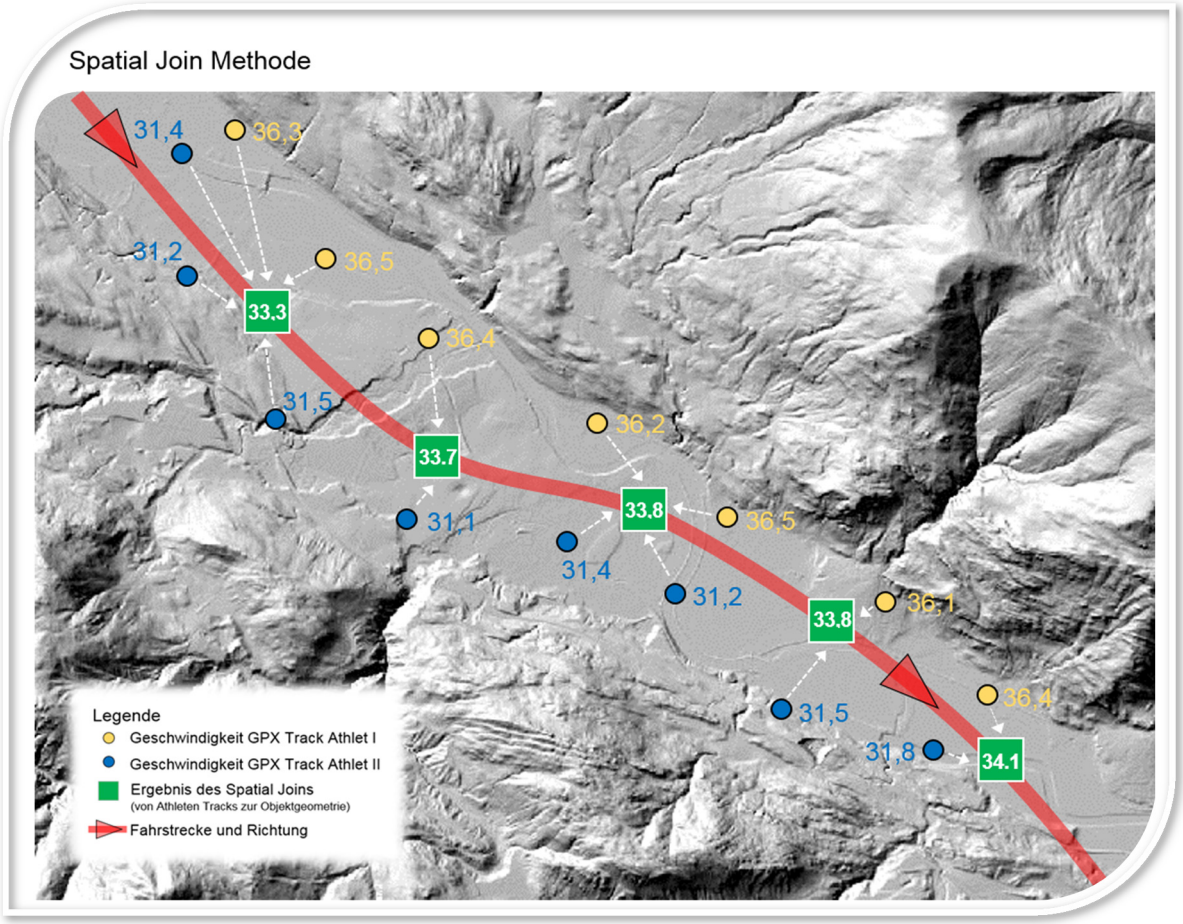


Abbildung 3-9: Spatial Join Methode

Die obige Abbildung 3-9 visualisiert den Prozess des „Spatial Join“, um schlussendlich typische, durchschnittliche Fahrzeiten für 5 Stunden Fahrer zu aggregieren. Entlang der Fahrstrecke wurden die GPX Felddaten der Wettkampfaufzeichnung von Athleten mit den klassifizierten Fahrzeiten von 4 Stunden und 50 Minuten bis 5 Stunden und 7 Minuten dargestellt. Bei der „Spatial Join“ Methode wird nun das Geschwindigkeits Attribut des GPX Messpunktes anhand der kürzesten Distanz mit dem nächstgelegenen Punkt der Marschroute verknüpft. Es wurde also eine Vielzahl an Geschwindigkeitsangaben (im Attribut der GPX Daten) nach folgenden Parametern zu einem Marschroutenpunkt aggregiert: Average, Sum, Minimum, Maximum, Standard Deviation und Variance. Für weiterführende statistische Auswertungen kann nun an jeder Position der Marschroutenpunkte die durchschnittlich gefahrene Geschwindigkeit aller Athleten einer bestimmten Leistungsklasse (z.B. 5 Stunden), analysiert werden.

Die Methodik des „Spatial Join“ funktioniert entlang der gesamten Radstrecke des Ironman Austria, ausgenommen des Abschnittes mit Gegenverkehr. Hier besteht die Problematik, dass die Messpunkte der GPX Felddaten nicht eindeutig einer Geometrie der Marschroute zugewiesen werden können. Speziell unter topographischen Gesichtspunkten ist dies eine Herausforderung wie die Abbildung 3-10 verdeutlicht.

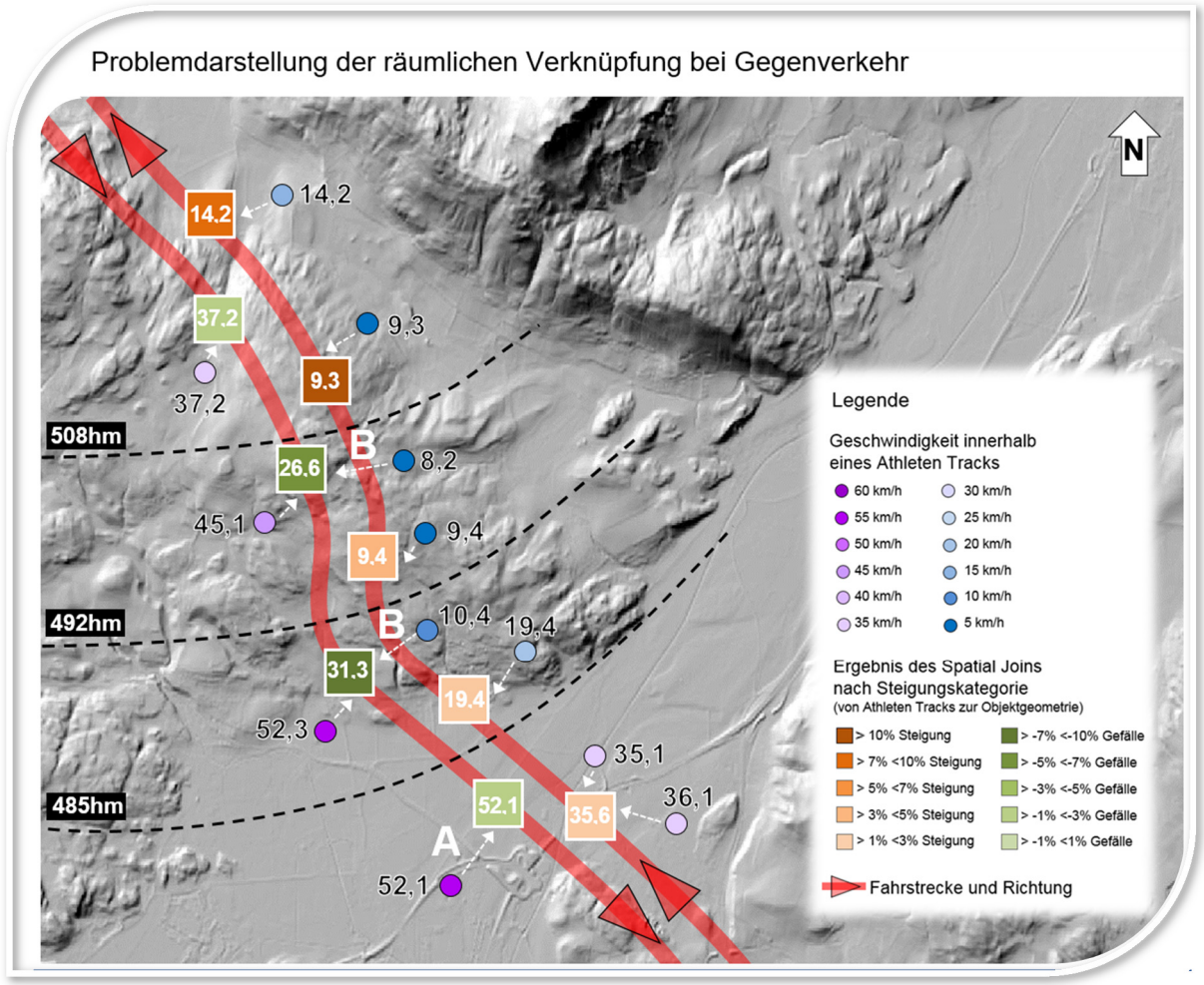


Abbildung 3-10: Problemdarstellung der räumlichen Verknüpfung bei Gegenverkehr

Die Abbildung 3-10 zeigt die „Spatial Join“ Methode in einem Abschnitt des Ironman Austria mit Gegenverkehr, was durch die beiden parallelen Fahrspuren verdeutlicht wird. Wie die Höhenschichtenlinien aufzeigen, fällt das Gelände nach Süden hin ab. Dies wird auch durch die Klassifizierung der Punktgeometrien der Marschroute nach Steigungen visualisiert. So ist ersichtlich, dass die linke Fahrspur bergab führt und die rechte Fahrspur dementsprechend bergauf führt. Die verstreuten GPX Messpunkte entlang der Strecke entstammen der Wettkampfaufzeichnung eines Athleten und wurden nach der Geschwindigkeit klassifiziert. Die „Spatial Joins“ mit der Deklaration „A“ in der Karte, sind korrekte räumliche Verknüpfungen, da die hohen Geschwindigkeitswerte den Punkten des Gefälles zugeordnet werden. Durch die Verknüpfung nach der geringsten Entfernung entstehen aber auch „Joins“ die nicht korrekt sind („B“ Kennzeichnungen). Hierbei werden Punkte, die eine niedrige Geschwindigkeit der GPX Felddaten aufweisen, fälschlicherweise in das Gefälle und nicht in den Bergauf Abschnitt der Marschroute verknüpft.

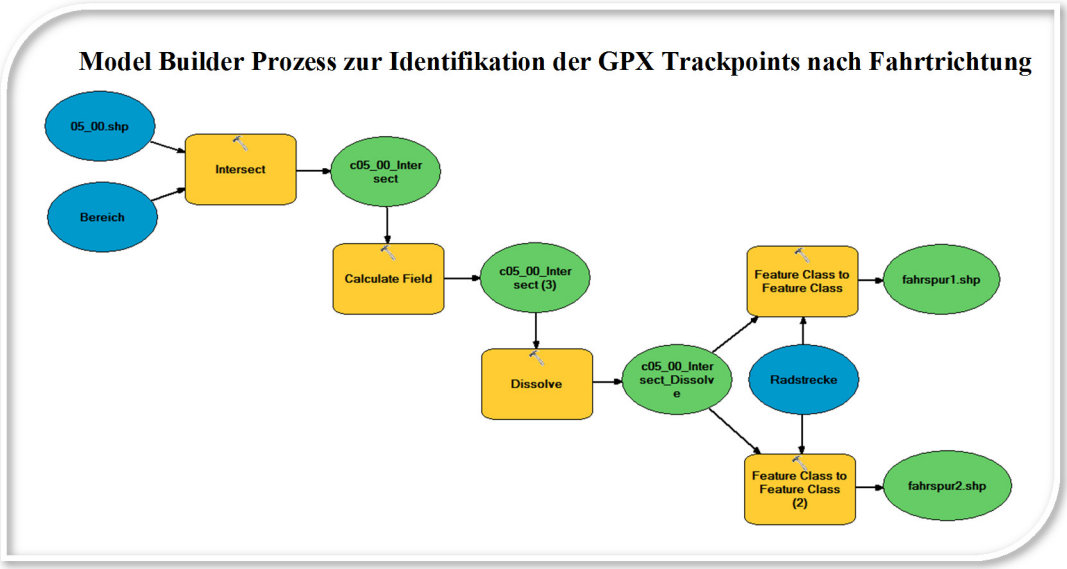


Abbildung 3-11: Model Builder Prozess zur Identifikation der GPX Trackpoints nach Fahrtrichtung

Zur Lösung dieser Problematik wurde ein Model Builder Prozess geschaffen, der es zum Ziel hat, die GPX Trackpoints der Athleten, in automatisierter Weise der entsprechenden Fahrspur zuzuweisen. Dazu mussten die jeweiligen Felddaten, die zu einem Abschnitt mit Gegenverkehr gehören, räumlich mit einem spezifischen Gebiet (Bereich.shp) über den „intersect“ Befehl verschnitten werden. Danach wurde anhand der Field ID, welche ein Maß über die Reihenfolge der gesetzten GPX Points ist, eine erste Zuordnung berechnet. So weisen etwa in einem Beispiel die Field IDs mit den Werten von 2283 bis 2564 und zugleich einer relativ hohen Geschwindigkeit von 48,1 km/h darauf hin, dass diese zur linken Fahrspur mit einer Abfahrt gehören. Diese linke Fahrbahn ist jene, die im Streckenverlauf zuerst zu absolvieren ist und somit auch mit den „niedrigen“ Field IDs gekoppelt wird. Nach einer Schleife wird der Abschnitt mit der Gegenfahrbahn nochmals befahren, doch nun wird das Gefälle zu einer Steigung. Im Anschluss wurde die nächste Gruppe an Field IDs mit den „höheren“ Werten von 4918 bis 5228 bei einer Geschwindigkeit von 18,3 km/h herangezogen und mit der rechten Fahrspur gekoppelt. Über die Bereiche der Field IDs mit den zu absolvierenden Fahrbahnabschnitten erfolgt somit eine eindeutige Zuordnung. Im Anschluss erfolgt, nach einem Zusammenführen der Attribute über den „Dissolve“ Befehl, ein Export der entsprechenden GPX Felddaten als „Fahrspur 1“ bzw. „Fahrspur 2“, um anschließend den „Spatial Join“ Prozess fortführen zu können.

4. Methodiken und Analysen als Lösungsansatz des realweltlichen Problems

Nachdem im Kapitel 3 die Geodaten mit typischen Fahrverhalten der Athleten erzeugt wurden, erfolgt nun die geostatistische Analyse ebendieser. Dazu wurde auf die Methodik der linearen Referenzierung und dynamischen Segmentierung zurückgegriffen. Diese Funktionen sind notwendig, um die unterschiedlichen Leistungsparameter effizient ermitteln und analysieren zu können, die dazu führen, dass der Sportler keine lineare und gleichmäßige Geschwindigkeit auf der Radstrecke umsetzen kann.

4.1 Lineare Referenzierung der Radstrecke

Geographische Informationssysteme stellen räumliche Informationen und damit verbundene Objektgeometrien meist in Form von X - und Y- Koordinaten dar (Esri 2001, S. 1). Eine weitere Methode zum räumlichen Definieren von Objekten ist jene der linearen Referenzierung (ebd.). Dabei wird eine relative Positionsangabe an linearen Objektgeometrien angegeben, um keine absoluten Koordinaten mehr speichern zu müssen (ebd.). Angesichts der vorliegenden wissenschaftlichen Fragestellung bietet sich die Methode der linearen Referenzierung optimal für die Fahrstrecke des Ironman Austria an. Durch die Transformation der Fahrstrecke als 2,5 dimensionale Polylinie in eine Route wird ein eindeutiger Identifikator durch das Hinzufügen eines neuen Maßsystems geschaffen. Nach Esri (2001, S. 2–3) handelt es sich beim Maßsystem entweder um Entfernungs- oder Zeiteinheiten, die losgelöst von X- und Y- Koordinaten entlang des Linien Features zur Anwendung kommen. Bei der Fahrstrecke wurde als Maßsystem die Entfernung in Meter vom Startpunkt (Wechselzone) angegeben. Bei der Methode der linearen Referenzierung kann durch die Funktion „Calibrate Routes“ dieser Startpunkt beliebig verschoben und auch die Kilometrierung entlang der Fahrstrecke neu kalibriert werden. Laut Esri (2001, S. 2–3) werden einzelne Punkte entlang der Route als M-Werte gespeichert, das heißt dass diese keine Koordinaten beinhalten, sondern vielmehr relative Distanzangaben entlang der Route, ausgehend vom Startpunkt, sind. Durch den Einsatz von Messwerten können Ereignisse (engl. events) entlang der Route klar angegeben werden (ebd.). Als Ereignisse werden Linien- oder Punktobjektgeometrien definiert, die sich topologisch mit der Route schneiden, oder in einem gewissen Suchradius zur Fahrstrecke zu finden sind (ebd.). Diese Events werden nicht als Shape Dateiformate, sondern in Form von Tabellen gespeichert.

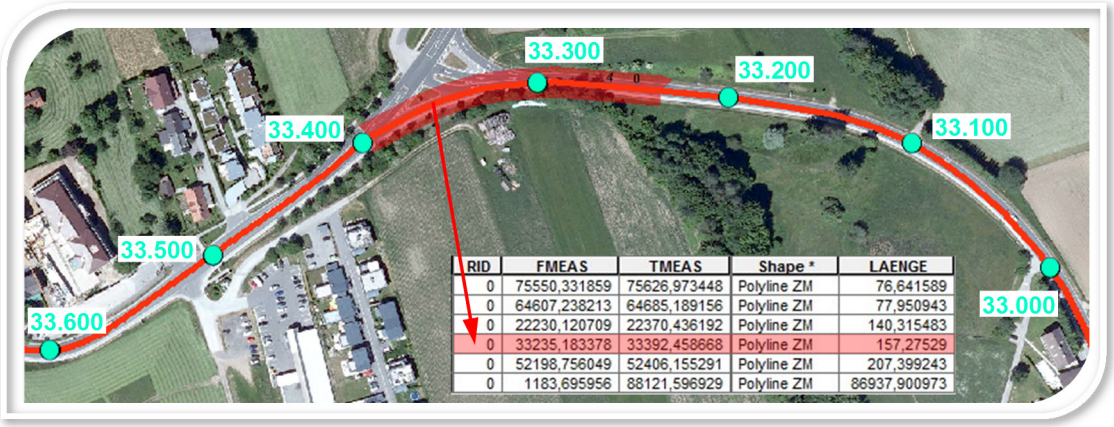


Abbildung 4-1: Routen-Feature der Fahrstrecke mit den Labestationen als Event Tabelle

Die Abbildung 4-1 links zeigt die Fahrstrecke des Ironman Austria mit dem Anstieg nach Drobollach als Route-Feature-Class. Die einzelnen Stützpunkte dieser Polylinie bestehen aus den XYZM-Angaben. Dabei bezeichnen die M Angaben die Entfernung vom Startpunkt entlang der Strecke zur jeweiligen Position in Meter. Diese lineare Referenzierung wurde, durch die Verwendung von Hatches als Skalenstriche, in einem 100 Meter Abstand, visualisiert. Die Abbildung 4-1 rechts zeigt einen Event Table, der die Labestationen als Linien Elemente verdeutlicht. Das Attribut FMEAS bezeichnet den Beginn der Labestation in Meter als lineare Entfernung zum Startpunkt der Strecke. So beginnt die erste Labestation 22.230 Meter nach dem Start der Radstrecke und endet mit dem TMEAS Attribut bei 22.370 Meter. Keine weiteren Koordinatenangaben sind notwendig, um die räumliche Position der Versorgungstationen für die Athleten zu bestimmen.

4.2 Dynamische Segmentierung zur GeoAnalyse

Erst durch die Methode der linearen Referenzierung kann darauf das Konzept der dynamischen Segmentierung aufbauen. Als Gegenkonzept gilt die statische Segmentierung, welche immer mit dem Einfluss bzw. der Veränderung der Objektgeometrie und der Koordinaten einhergeht. Nach Esri (2001, S. 1) kann dahingegen durch die Methode der dynamischen Segmentierung eine Vielzahl an Event Tabellen, sprich Ereignissen, entlang einer Route definiert werden. Die angesprochene Dynamik ist darin begründet, dass durch diese GIS Methode die Route hinsichtlich der Objektgeometrie nie segmentiert wird (ebd.). Das heißt, es erfolgt keine Zerschneidung der Polylinie nach Koordinaten, sondern es werden vielmehr die beliebig vielen Attribute durch die Kilometrierung an die Strecke angehängt (ebd.). Diese Methode bietet sich für die Modellierung der Radstrecke des Ironman Austria an, da die Fahrbahn nach vielen Aspekten dynamisch lokalisiert werden soll (Abbildung 4-2).

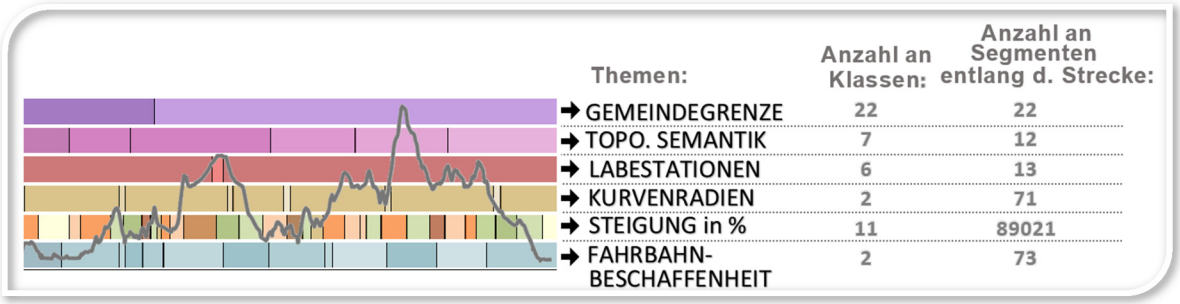


Abbildung 4-2: Schematische Darstellung der verwendeten Themen zur dynamischen Segmentierung

Die Abbildung 4-2 zeigt die Attribute und die damit gekoppelten Event Tabellen entlang der Radstrecke des Ironman Austria als schematische Darstellung. Durch die dynamische Segmentierung können diese Abschnitte im GIS visualisiert, abgefragt und bearbeitet werden. Doch das stärkste Argument für die Verwendung dieser Methode liegt in der Analysefunktion, welche etwa die Überlagerung von Attributthemen „Line-on-Line Overlay“ ermöglicht, ohne dass sich dies auf die Geometrie der Fahrstrecke auswirkt. So können für statistische Analysen beispielsweise jene Segmente visualisiert werden, die eine Steigung von mehr als 10% mit hohen Kurvenradien (≤ 178 Grad und ≥182 Grad) und einer Fahrbahnbeschaffenheit mit der Vibration von einem Beschleunigungsfaktor von ≥ +/-2 g-Kräften aufweisen.

4.3 Geostatistische Analysen nach Leistungsparametern der Felddaten zur Induktion eines Modells

Wilson et al. (2004, S. 1–247) schreibt in „Bicycling Science“ über diverse Faktoren im Radsport, welche die Leistungsfähigkeit beeinflussen. Die technischen Leistungsparameter, wie Luftdruck der Reifen oder die Aerodynamik des Fahrrades und die Sitzposition des Athleten werden auf Grund des fehlenden GIS Bezuges in dieser Arbeit nicht näher erläutert. Die physischen Leistungsparameter der Athleten wurden durch die Erhebung der empirischen Felddaten und die Klassifikation ebendieser in Kapitel 3 erläutert beziehungsweise abgedeckt. In den folgenden Unterkapiteln sollen die Leistungsparameter der äußeren Umwelteinflüsse analysiert werden, um aus diesen Induktionsschlüsse zu ziehen, welche in der programmiertechnischen Modellierung umgesetzt werden.

4.3.1 Topographie

Den Einfluss der Topographie einer Radstrecke beschreibt Wu et al. (2014, S. 224–225) als entscheidenden Faktor auf die Leistung der Athleten. Dieser Leistungsparameter kann in seiner Intensität bei den unterschiedlichen weltweiten Ironman Rennen stark variieren. Die Anzahl der zu bewältigenden Höhenmeter kann von lediglich 300 (Ironman Florida) bis 2.600 Höhenmeter (Ironman Lanzarote) differieren. Beim Ironman Austria werden offiziell 1.700 Höhenmeter bergauf ausgewiesen, womit das Rennen hinsichtlich der Topographie als durchaus anspruchsvoll klassifiziert werden kann. Dieser nach der World Triathlon Corporation (2016) angegebene Wert von 1.700 Höhenmeter kann nach GIS Berechnung mit den Laserscan Daten nicht bestätigt werden. Über die Modellierungsmethode von GPX Punktdaten im Abstand von 10 Meter und der damit verbundenen GIS Analyse des 1 Meter Laserscan Datensatzes wurden 1.653 Höhenmeter berechnet, welche bergauf über die Radstrecke zu überwinden sind. Für eine Analyse der Geschwindigkeiten über verschiedene Steigungsprozente musste eine Klassifikation von diesen durchgeführt werden. Die Klassengrenzen wurden vom renommierten Triathlon Trainer Joe Friel und seiner Publikation (Friel und Vance 2013, S. 125) übernommen (Abbildung 4-3).

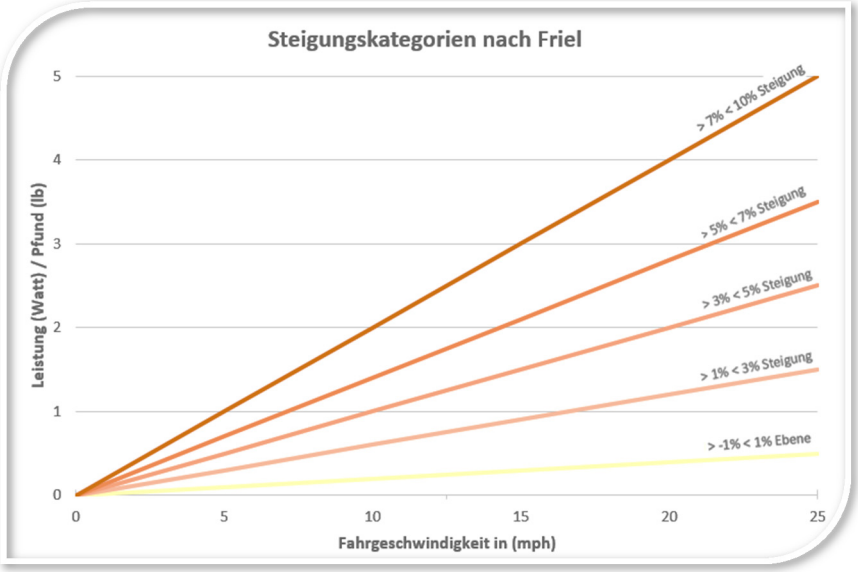


Abbildung 4-3: Steigungskategorien im Radsport nach Leistungseinsatz
Eigene Bearbeitung, Datenquelle: (Friel und Vance 2013, S. 125)

Die Abbildung 4-3 zeigt das Verhältnis der Faktoren aus Geschwindigkeit, Leistung pro Körpergewicht (Watt pro Pfund, engl.: lb) und der Steigung in Prozent. Die gewählten Klassengrenzen der Steigungsprozente haben nach Friel und Vance (2013, S. 125) signifikante Auswirkungen auf die Leistung. Als interessantes Faktum ist der lineare Anstieg der Leistung nach Geschwindigkeit und Steigung zu erwähnen. Dieses Phänomen der linearen Veränderung ist auch bei der Analyse nach Leistungsklassen der Athleten und der Topographie bei den empirischen Felddaten erkennbar (Abbildung 4-4). Da nach Wu et al. (2014, S. 224–225) die Geschwindigkeit auf Basis der Topographie einen signifikanten Leistungsparameter darstellt, wurde die gesamte Radstrecke des Ironman Austria nach den Klassengrenzen von Friel und Vance (2013, S. 125) über die Methode der dynamischen Segmentierung eingeteilt.

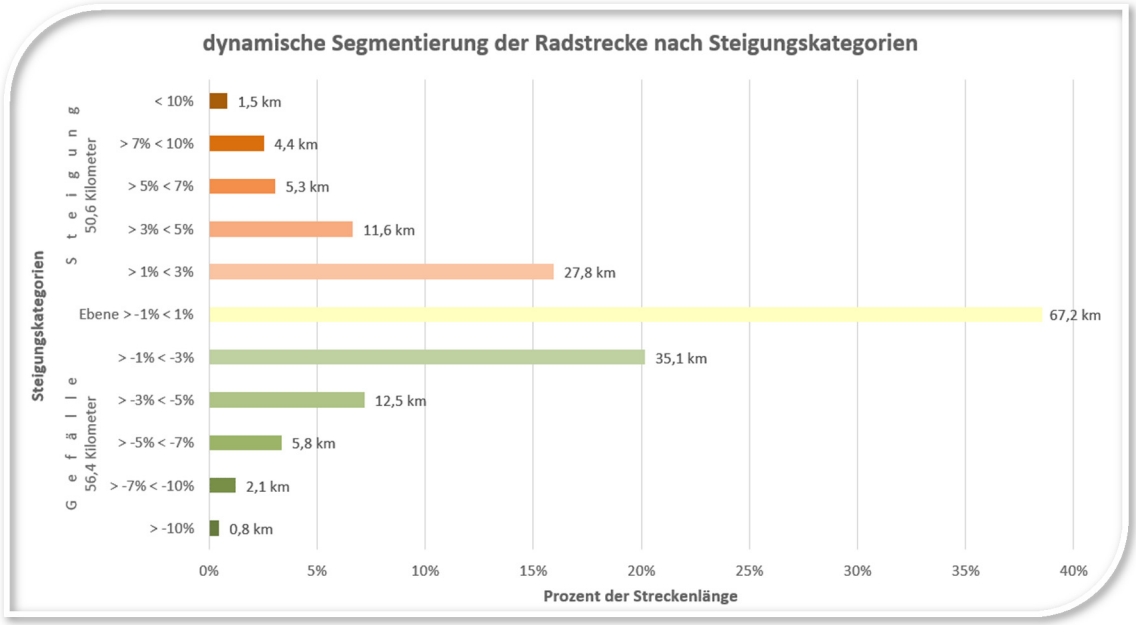


Abbildung 4-4: Dynamische Segmentierung der Radstrecke nach Steigungskategorien

Das Diagramm der Abbildung 4-4 veranschaulicht die Verteilung der dynamischen Segmentierung zur Radstrecke nach Steigungsprozenten und den entsprechenden Straßenkilometern. Die Klasse, welche als „Ebene“ bezeichnet wurde und Steigungen von >-1% und < 1% aufweist, ist mit 67,2 Kilometer die dominanteste Gruppe. Weiters fällt auf, dass alle Segmente mit Gefälle um 5,9 km länger sind als jene mit einer Steigung. In Kombination mit dem Höhenprofil kann die Topographie der Radroute des Ironman Austria als Strecke mit kürzeren und dafür steileren Anstiegen und langen Abfahrten mit geringen Prozenten des Gefälles interpretiert werden. Umso wichtiger ist es daher, nicht nur die Steigungssegmente mit starken Geschwindigkeitsveränderungen, sondern auch die Abschnitte mit Gefälle zu analysieren. In der Abbildung 4-5 wurden die Geschwindigkeiten nach Leistungskategorien aus den Erhebungen der empirischen Felddaten mit den Steigungsklassen verschnitten. Interessant ist dabei die Tatsache, dass nicht jene Segmente mit dem höchsten Gefälle (>-10% Gefälle) auch die schnellsten Abschnitte sind, sondern jene mit einer Abfahrt von <-5% und <-7%. Eine mögliche Begründung wird in Kapitel 4.3.3 erläutert, wo enge Kurvenradien einen Einfluss auf die Fahrgeschwindigkeit der Athleten haben.

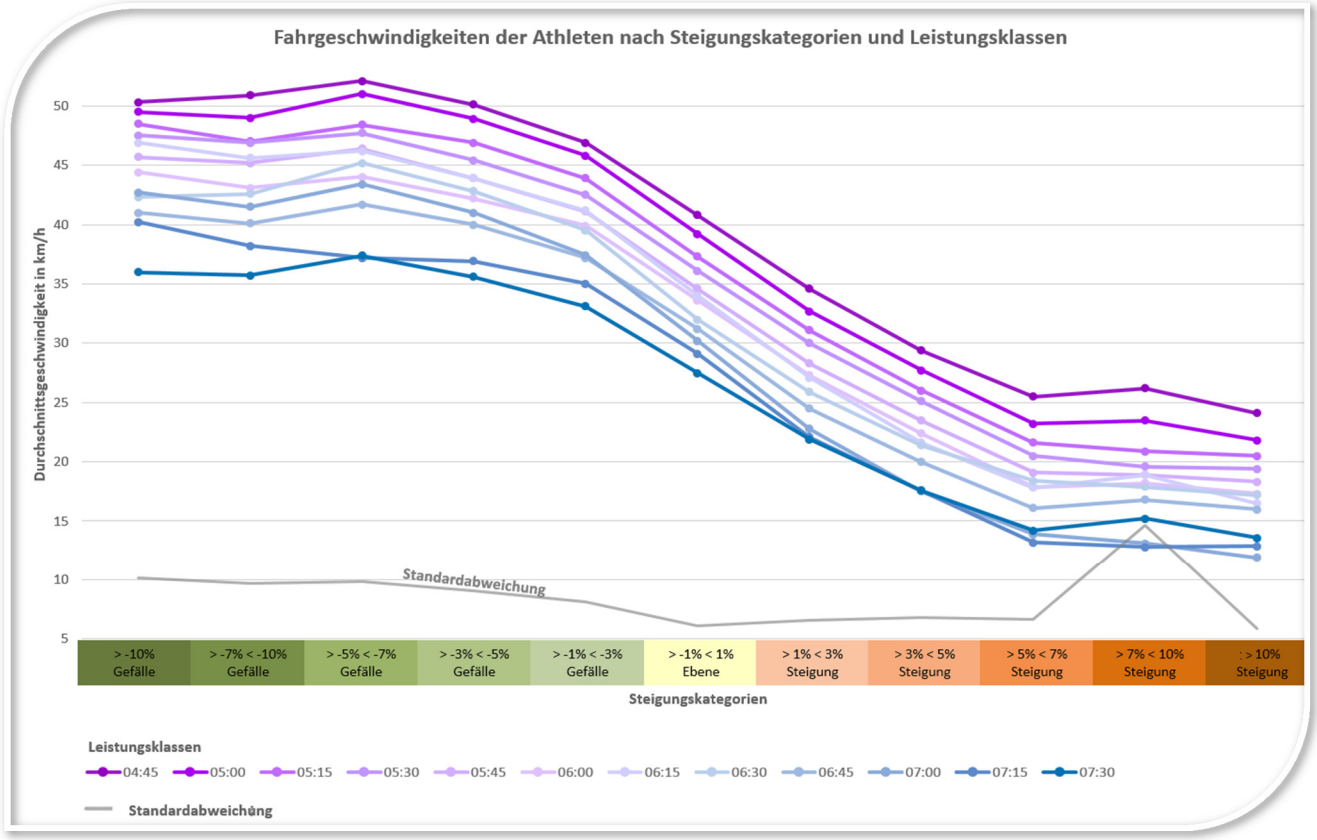


Abbildung 4-5: Fahrgeschwindigkeiten der Athleten nach Steigungskategorien und Leistungsklassen

Wie bereits in der Darstellung 4-3 von Friel und Vance (2013, S. 125) erläutert, ist auch im Diagramm 4-5 eine gewisse lineare Geschwindigkeitsveränderung speziell bei den Steigungen ersichtlich. Lediglich die Klasse mit einer Steigung von > 7% und < 10% fällt, durchgehend über alle Leistungskategorien der Athleten, aus diesem Trend. Dies verdeutlicht auch den Anstieg der Standardabweichung in dieser Steigungsklasse. Hierbei ist es wichtig anzumerken, dass sich die Standardabweichung nicht auf die Leistungskategorien der Athleten bezieht, sondern auf die Unterschiedlichkeit der Geschwindigkeiten in den 10 Meter Abschnitten. Für die Klasse mit > 7% und < 10% Steigung bedeutet die entsprechende Standardabweichung somit, dass sich die Geschwindigkeiten über alle Leistungskategorien hinweg in den entsprechenden 10 Meter Segmenten um 14,9 km/h vom Mittelwert streuen. Daraus ist zu schließen, dass obwohl alle 10 Meter Segmente die gleiche Steigungswerte von > 7% und < 10% aufweisen, eine hohe Variabilität der Geschwindigkeiten zu verzeichnen ist. Um dieses einzelne Phänomen, welches entgegen des linearen Trends der Abbildung 4-3 wirkt, näher zu analysieren, wurden alle Segmente der Klasse selektiert (Abbildung 4-6).

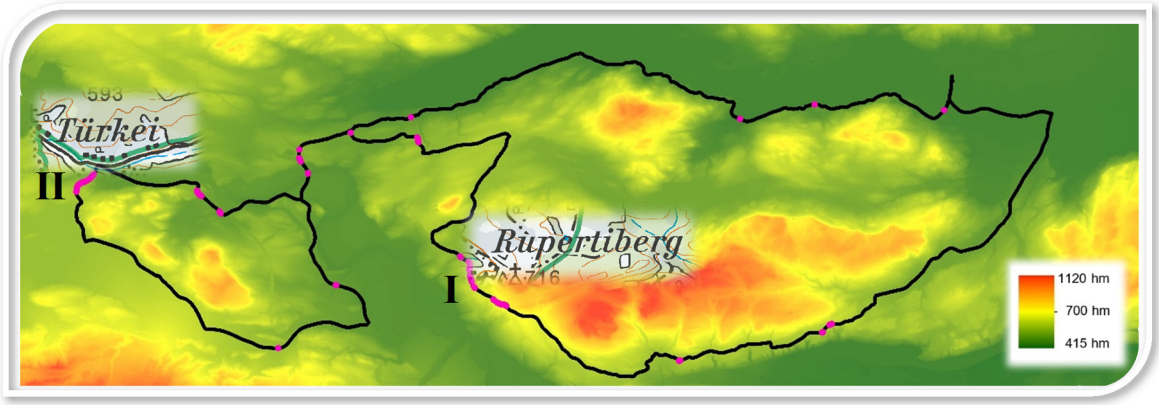


Abbildung 4-6: Übersicht der Sektoren mit Steigungen von > 7 und < 10%

Exemplarisch wurden die zwei längsten Segmente genauer analysiert, welche durch die semantische Verknüpfung mit der ÖK (Kapitel 3.1.3.5) als Rupertiberg und Türkeihügel bezeichnet sind.

Analyse des Abschnittes I: Rupertiberg

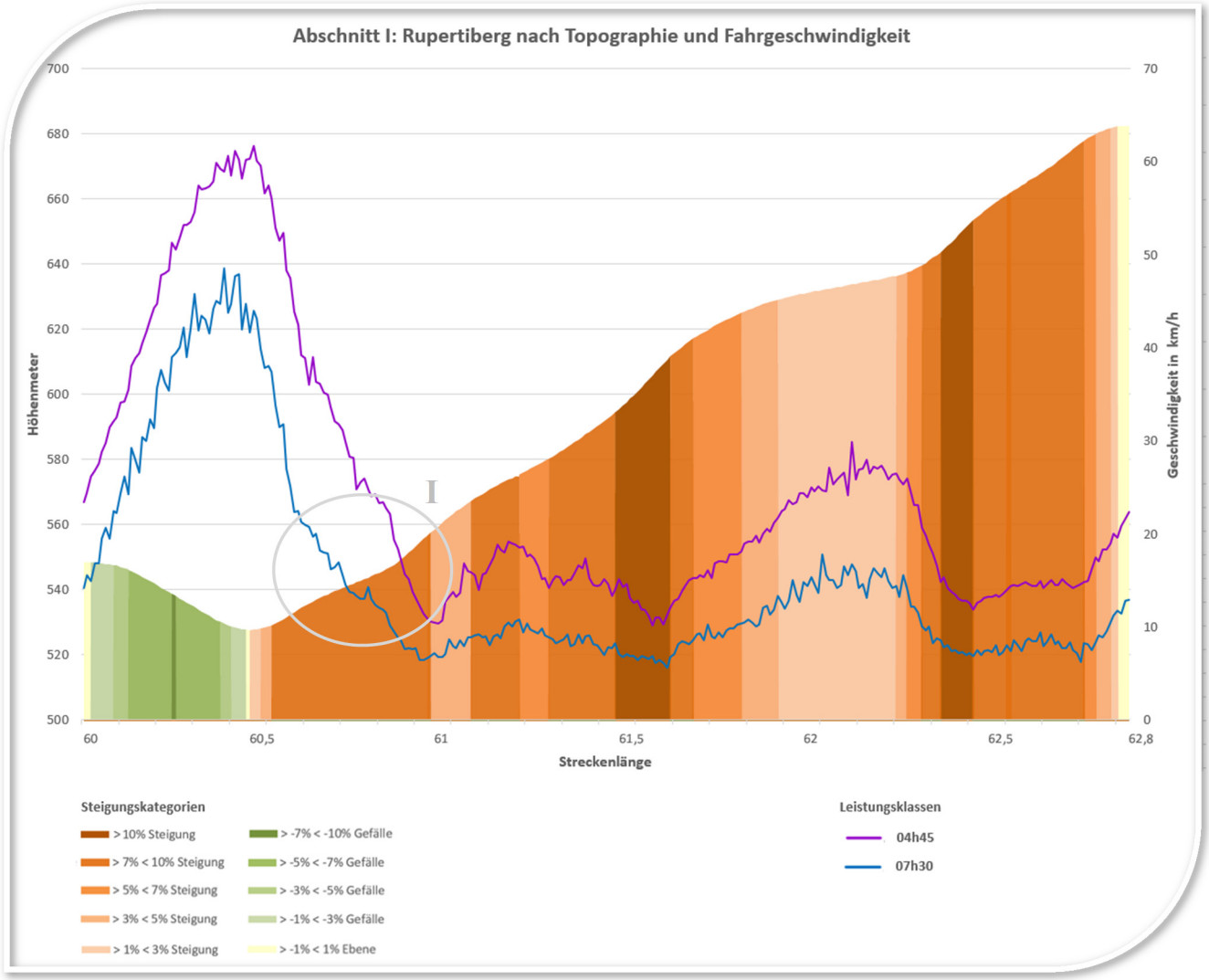


Abbildung 4-7: Analyse des Abschnittes I: Rupertiberg

Das Diagramm 4-7 visualisiert den Abschnitt des Anstieges zum Rupertiberg als Höhenprofil, klassifiziert nach den Steigungs- und Gefälleklassen. Hierbei wird ersichtlich, dass vor dem Anstieg eine kleine Abfahrt mit 18 Höhenmeter erfolgt, und dadurch die Geschwindigkeit der beiden exemplarischen Leistungskategorien von 4h:45m und 7h:30m auf 50 bzw. 60 km/h ansteigt. Wie im Diagramm ersichtlich, verpufft die Geschwindigkeit nicht gleich beim Eintreffen in die Steigung, sondern wird sukzessive abgebaut. Durch diese Situation der Senke, kann viel Geschwindigkeit in den anfänglichen Steigungsbereich von > 7% und < 10% mitgenommen werden. Dies erklärt die hohe Standardabweichung dieser Leistungsklasse, da die typische Geschwindigkeit in den 10 Meter Sektoren auf Grund der vorangegangenen Abfahrt verändert wird. Normalerweise wird in der Leistungskategorie mit 04h:45m Athleten bei einer Steigung von > 7% und < 10% eine durchschnittliche Geschwindigkeit von 22,5 km/h gefahren (Abbildung 4-7, I). In unserem Beispielsegmenten fahren die 04h:45m Athleten jedoch

28,4 km/h, was zu einer höheren Streuung der Geschwindigkeit in allen 10 Meter Segmenten mit > 7% und < 10% Steigung führt.

Analyse des Abschnittes II: Faakersee:

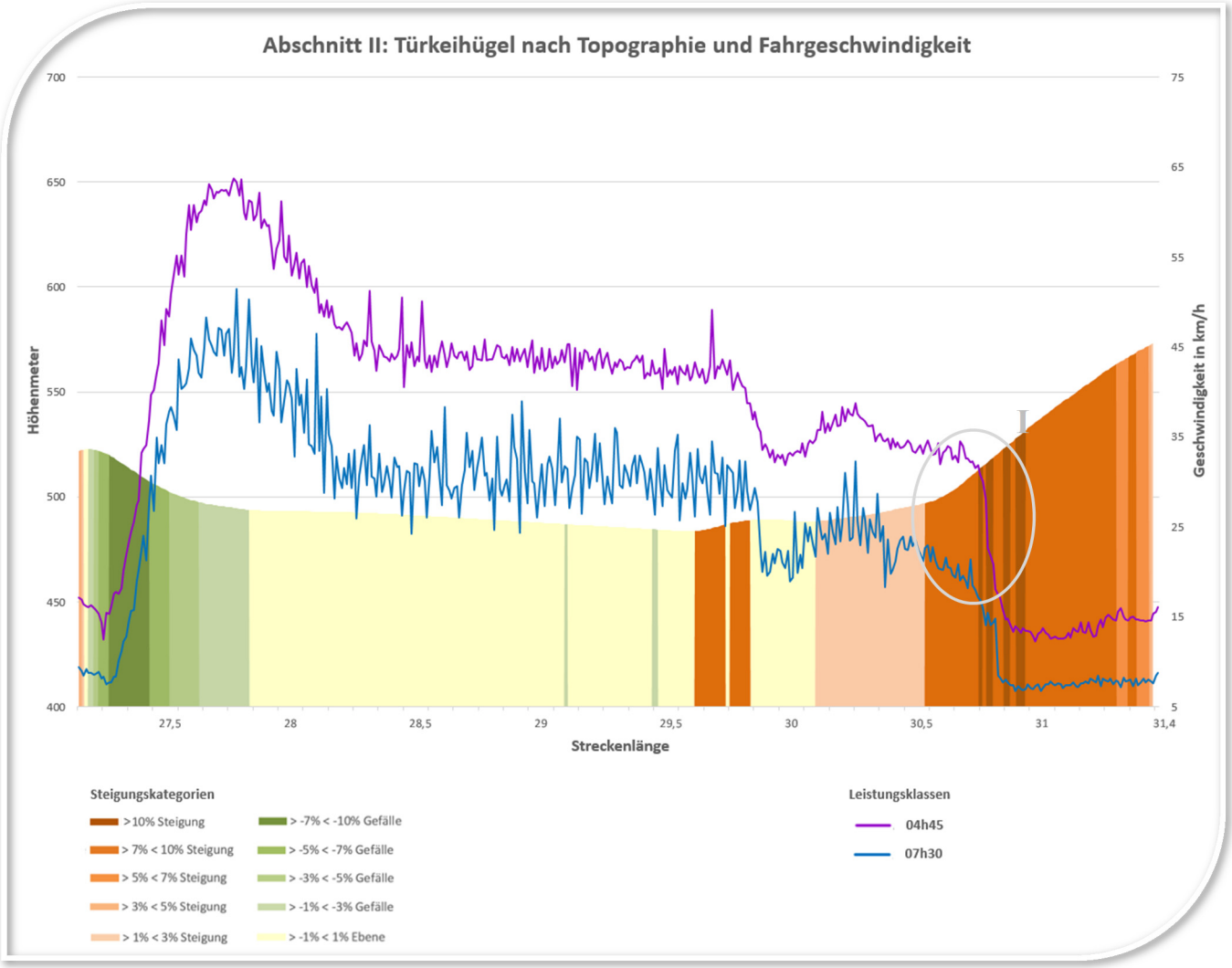


Abbildung 4-8: Analyse des Abschnittes I: Türkeihügel

Die Abbildung 4-8 mit dem Anstieg zum Faakersee veranschaulicht eine ähnliche Situation wie jene des Rupertiberges. Auch hier folgt auf eine Abfahrt ein Flachstück mit leichtem Gefälle bis zum eigentlichen Anstieg, wo die Geschwindigkeit in die Steigung mitgenommen wird (Abbildung 4-8, I). Diese beiden Beispiele sind auf Grund der vorhergehenden Segmente mit Gefälle und der so genannten Senken-Situation Ausnahmebeispiele, welche für untypische Geschwindigkeitswerte in den Klassen mit > 7% und < 10% Steigung sorgen. Die Erkenntnisse dieser Analyse werden in das Kapitel 5 zur Geosimulation eingearbeitet und es werden für die beiden oben angeführten Steigungssegmente spezielle Ausgleichsfaktoren zur Berücksichtigung der untypischen Geschwindigkeiten verwendet.

Bei der Analyse der Leistungsparameter nach Geschwindigkeit über die Topographie sind aber nicht nur die Auswertungen aller Leistungskategorien der Athleten von Interesse, sondern auch die Unterschiede innerhalb einer Leistungsklasse. So soll nun der Frage nachgegangen werden, in welcher Intensität es zu Abweichungen bei den Geschwindigkeiten in den einzelnen Steigungsklassen innerhalb der Leistungskategorien von Athleten kommen kann.

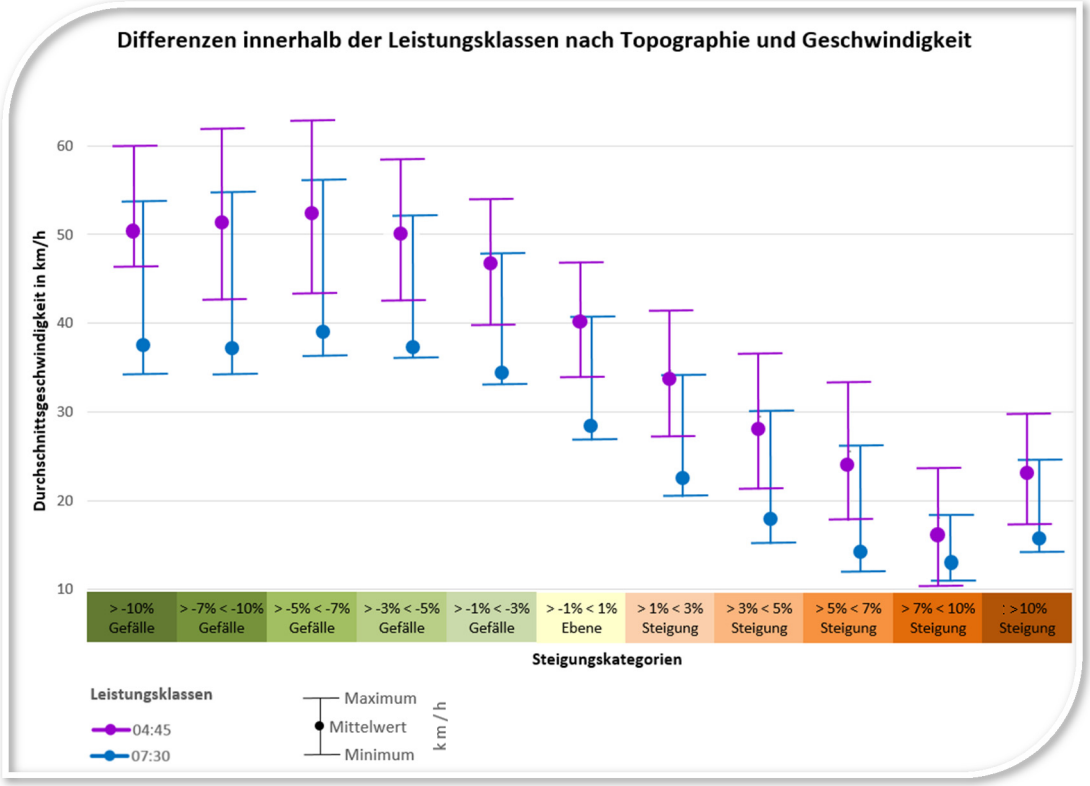


Abbildung 4-9: Differenzen innerhalb der Leistungskategorien

Ein Vergleich der Fahrverhalten bei unterschiedlichen Steigungen innerhalb der stärksten Leistungsklasse mit 04h:45m Fahrern und den schwächsten Fahrern mit 07h:30m, wird in der Abbildung 4-9 angestrebt. In beiden Leistungsklassen variiert die Geschwindigkeit in einem ähnlichen Ausmaß von 14 km/h. Generell ist festzustellen, dass die Geschwindigkeitsunterschiede von Athleten innerhalb einer Leistungsklasse bei Segmenten mit einem Gefälle höher ausfallen als bei Steigungen. Die größte Range mit 18 und 19 km/h ist in den beiden Klassen zwischen 5% und 10 % Gefälle gegeben. Doch auch bei Steigungssegmenten ist eine markante Divergenz in der Fahrgeschwindigkeit der Athleten, innerhalb einer Leistungsklasse, zu erkennen. Beispielsweise sind in der Steigung von > 5% und < 7% in der Leistungsklasse mit 04h:45m Fahrer vertreten, die durchschnittlich in diesen Sektoren 20 km/h sowie auch 34 km/h fahren. Auf Grund der verschiedenen Fahrweisen innerhalb der Leistungsklassen, fließt diese Erkenntnis der

Analyse in die Simulation der Marschroute ein. In der GUI zur Erstellung eines GPX Tracks werden die typischen Fahrgeschwindigkeiten der Leistungsklassen nach Mittelwerten in den Steigungskategorien voreingestellt. Die Klassen können jedoch beliebig variiert und die dazugehörige Geschwindigkeit, nach individueller Präferenz, adaptiert werden (Abbildung 5-2).

4.3.2 Labestationen als „Points of Interests"

Als Points of Interests (POIs) werden sehenswerte bzw. relevante räumliche Informationen für den Benutzer bezeichnet. Dies sind bei der klassischen GPS Navigation Einrichtungen wie Krankenhäuser, Hotels, Einkaufsmöglichkeiten, Restaurants usw. Für den Athleten des Ironman Austria bestehen ebenfalls Punkte an der Radstrecke, welche von Interesse sind. Bei Labestationen (engl. Aid Stations), die entlang der Radstrecke eingerichtet sind, werden durch freiwillige Helfer in organisierter Form, Wasser, Bananen, Energie Drinks und Energie Riegel ausgegeben. Die Modellierung dieser Einrichtungen ist im Zuge dieser Masterarbeit von Relevanz, um eine Hypothese des Zeitverlustes in diesen Sektoren zu überprüfen. Nach der offiziellen Streckenkarte zum Ironman Austria werden diese insgesamt sechs Labestationen als Punkt Geometrie in einem Kartenmaßstab von 80.000 dargestellt. Die exakte Position lässt sich bei einer solchen kleinmaßstäbigen Darstellung nur erahnen, soll aber in diesem Abschnitt trotzdem als Attribut in das Geometrie Feature der Radstrecke modelliert werden. Im Gegensatz zu den rein geostatistischen Verfahren und Berechnungsmethoden in Kapitel 3.2 geschieht die Positionierung der Aid Stations über visuelle Überprüfungen. Dazu wurden die GPX Bewegungsdaten der empirischen Befragung herangezogen und nach Geschwindigkeitsklassen eingeteilt. Nun kann überprüft werden, ob in den Abschnitten der vermeintlichen Segmente der Labestationen eine Geschwindigkeitsreduktion feststellbar ist. Bei den Athleten mit schnelleren Radzeiten konnte nur marginal festgestellt werden, wo sich die Labestation befindet. Im Gegensatz dazu wurde bei der visuellen Analyse der Athleten mit langsameren Radzeiten tatsächlich eine Geschwindigkeitsreduktion festgestellt. Auf Grundlage dieser visuellen Validierung wurden die entsprechenden Segmente als Labestationen definiert und mit einem Attribut versehen (Abbildung 4-10). Ebenfalls ein Indikator für Labestationen sind viele Datenpunkte, da bei einem Aufzeichnungsintervall von einer Sekunde eine Agglomeration von einer Vielzahl an GPX Points eine lange Standzeit bedeutet und somit als Aufenthalt in einer Labestation interpretierbar ist.

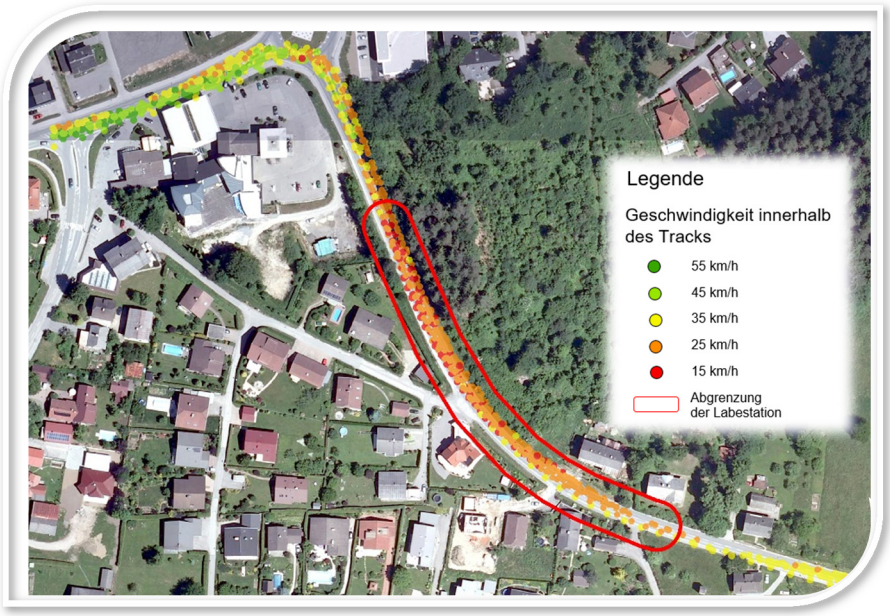


Abbildung 4-10: Identifikation der Labestationen

Nach der Definition und Abgrenzung der Lage zu den Labestationen erfolgt nun eine Analyse zum Fahrverhalten der Athleten in diesen Sektoren. Dazu wurden die durchschnittlichen Zeitverluste in den sechs Labestationen nach Leistungsklassen analysiert und in der Abbildung 4-11 dargestellt.

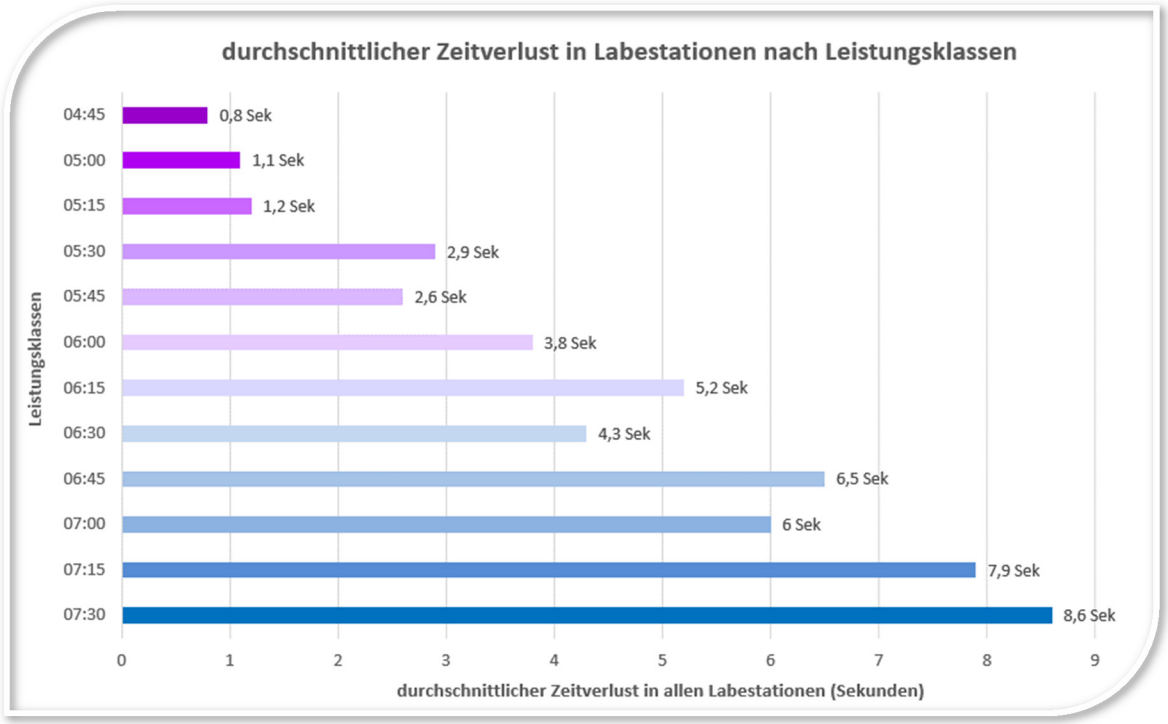


Abbildung 4-11: Durchschnittlicher Zeitverlust in Labestationen nach Leistungsklassen

Tendenziell ist festzustellen, dass der Zeitverlust in den Labestationen von den schnellen zu den langsamen Athleten hin, naturgemäß ansteigt. Die schnellsten Athleten der empirischen Felddaten mit Radzeiten um 04h:45m haben in allen Labestationen durchschnittlich einen Zeitverlust von 0,8 Sekunden im Vergleich zu den Durchschnittsgeschwindigkeiten in Segmenten mit vergleichbaren Steigungen. Der durchschnittliche Zeitverlust steigt bei den Athleten der langsamsten Leistungsklasse auf 8,6 Sekunden an. Dies bedeutet, dass der Zeitverlust in den Labestationen, im Vergleich zum generellen Geschwindigkeitsunterschied der gesamten Radstrecke, von schnellen (4h:45m) und langsamen (07h:30m) Athleten exponentiell ansteigt. Für die Simulation des GPX Tracks als Marschroute bedeuten diese Erkenntnisse, dass dieses divergierende Verhalten der Athleten in der GUI eingearbeitet wurde (Kapitel 5.1). So kann der Benutzer, nach Auswahl der gewünschten Zielzeit und des vorgeschlagenen typischen Zeitverlustes in den Labestationen, diesen Wert individuell nach seinen Leistungsklassen und Ernährungsstrategien adaptieren (Abbildung 5-2).

4.3.3 Kurvenradien

Wie in Kapitel 4.3.1 bereits angesprochen, so liegt, aus Erfahrungen als Athlet, die Vermutung nahe, dass auf Grund enger Kurvenradien, die Geschwindigkeiten bei Segmenten mit >-10 % Gefälle langsamer sind als etwa jene mit >-7% <-10% Gefälle. Um dies zu belegen, wurde eine Analyse der Fahrstrecke durchgeführt. Die Polylinie, welche alle 10 Meter einen Vertex aufweist, wurde nach unterschiedlichen Kurvenradien klassifiziert.

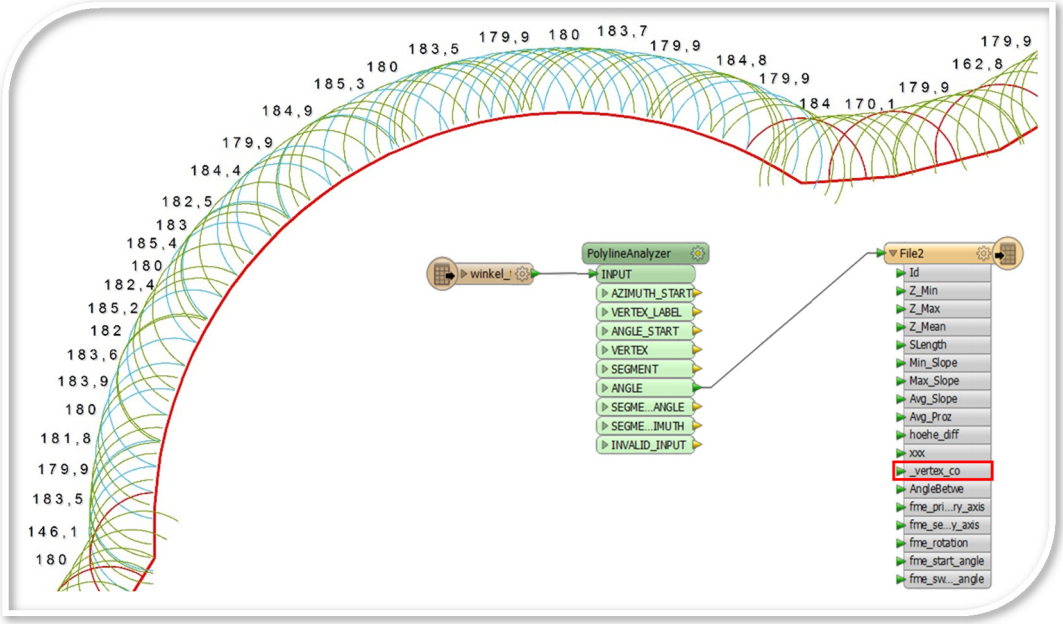


Abbildung 4-12: Ergebnisse zur Berechnung der Kurvenradien

Der oben abgebildete FME Prozess (Abbildung 4-12) berechnet über eine Analyse Methode jeden Winkel zwischen zwei aufeinandertreffenden Kanten (englisch: arc) der Polylinie. Ein Winkel von 180 Grad bedeutet ein absolut gerades Teilstück. Linkskurven aus Sicht des Athleten sind Kurvenradien von mehr als 180 Grad und Rechtskurven jene mit weniger als 180 Grad, wobei der Begriff der Kurve bei einem Winkel von 179 oder 181 Grad, sicherlich unpassend ist, da dies lediglich kleine Richtungskorrekturen sind (Abbildung 4-12). Die Klassengrenze zu hohen Kurvenradien (<178 Grad und >182 Grad) wurde auf Basis von persönlichen Erfahrungswerten zum Ironman Austria gewählt und weist folgende Kennzahlen auf (Tabelle 4-1).

Kennzahlen zu Kurvenradien

arithmetisches Mittel der Kurvenradien zur gesamten Radstrecke	180,21 Grad
Medianwert der Kurvenradien zur gesamten Radstrecke	180,00018 Grad
Distanz und Steigung an Rechtskurven mit hohen Radien (<178 Grad)	6,93 km Ø 0,03% Steigung
Distanz und Steigung an Linkskurven mit hohen Radien (>182 Grad)	7,09 km Ø 0,7% Steigung

Tabelle 4-1: Kennzahlen zu Kurvenradien

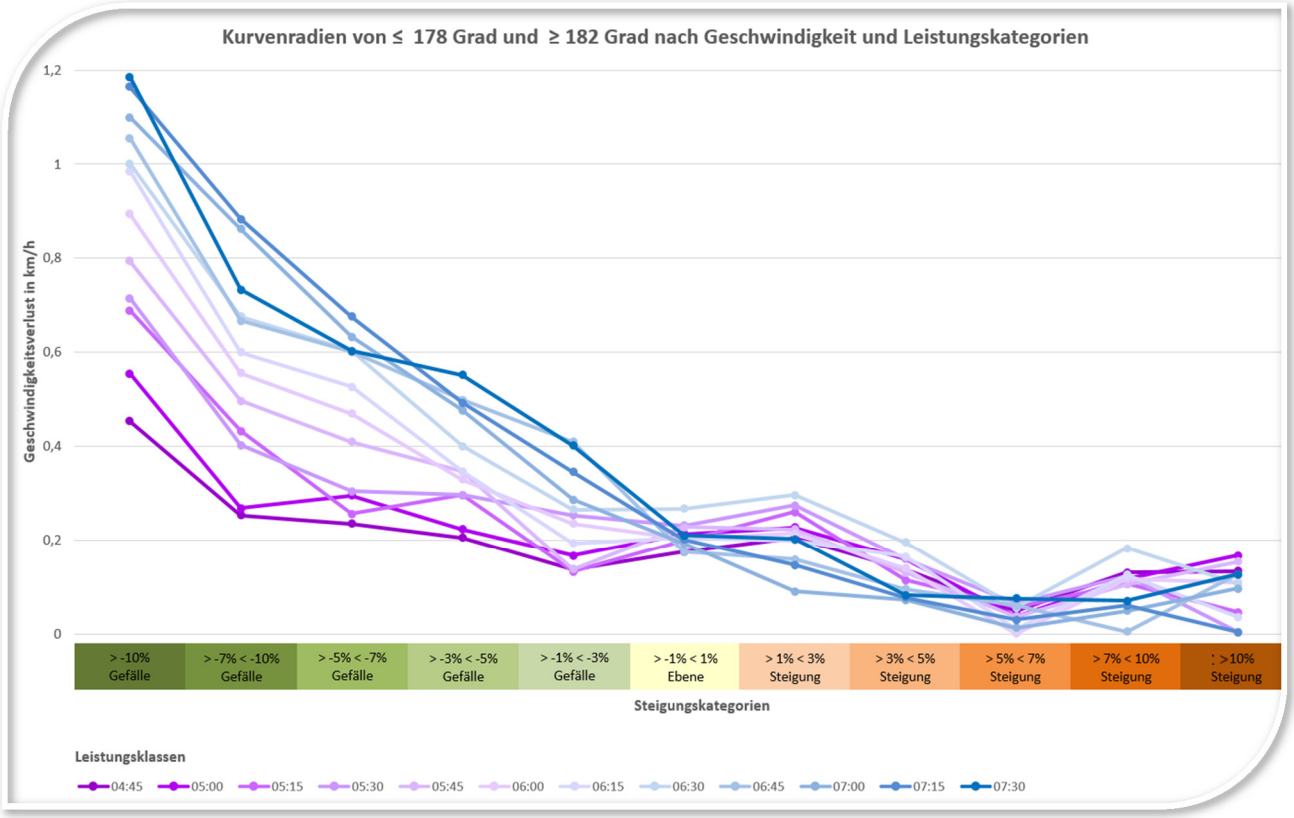


Abbildung 4-13: Analyse der Geschwindigkeit bei hohen Kurvenradien

Um den Leistungsparameter der hohen Kurvenradien auf die Radstrecke des Ironman Austria analysieren zu können, musste der Einfluss der Fahrbahnsteigung relativiert werden. Dies geschah im Diagramm der Abbildung 4-13, welches das Fahrverhalten der Athleten in Segmenten mit hohen Kurvenradien (≤ 178 Grad und ≥182 Grad) verdeutlicht. Diese Abschnitte wurden weiters nach den 11 Steigungs- und Gefälleklassen (X-Achse) differenziert. Auf der Y-Achse wurde der Geschwindigkeitsverlust (in km/h) in den kurvenreichen Segmenten mit den durchschnittlichen Geschwindigkeiten in den jeweiligen Steigungs- und Gefälleklassen visualisiert. Generell ist erkennbar, dass über alle Leistungskategorien hinweg in allen Segmenten mit hohen Kurvenradien die Durchschnittsgeschwindigkeit, um 0,4 km/h verringert wird. Dieser Wert ist so marginal, dass sich bei 6 Stunden Athleten die gesamte Radzeit um lediglich 20 Sekunden verkürzt und somit dieser Leistungsfaktor nicht in die Simulation der Marschroute aufgenommen wird. Durch die Analyse hoher Kurvenradien konnte aber trotzdem die einleitende Hypothese bestätigt werden, dass in Segmenten mit dem höchsten Gefälle (> -10%) der Geschwindigkeitsverlust signifikant am stärksten ist. In Segmenten mit dem zweit höchsten Gefälle (>-7% und <-10%) ist der Geschwindigkeitsverlust durch hohe Kurvenradien um bereits durchschnittlich 0,4 km/h über alle Leistungsklassen hinweg gesunken. In der Abbildung 4-13 ist weiters ersichtlich, dass in den Abschnitten der Ebene und der Steigungen kaum ein Geschwindigkeitsverlust durch hohe Kurvenradien zu verzeichnen ist. Bei der Analyse nach Leistungsklassen der Athleten kann in Segmenten mit Gefälle klar dargestellt werden, dass schwächere Athleten einen höheren Zeitverlust durch Kurvenradien aufweisen. Dahingegen konnte bei den Teilstücken, welche eine Steigung und hohe Kurvenradien besitzen, kein klarer Trend von schnelleren und langsameren Athleten ausgemacht werden.

4.3.4 Fahrbahnbeschaffenheit

Ein weiterer Faktor, der die Leistung des Athleten beeinflussen kann, beschreibt Wilson et al. (2004, S. 122–123) als „rough roads“ und dieser wird in diesem Kapitel als Fahrbahnbeschaffenheit geführt. Wilson erörtert den Einfluss von schlechten Fahrbahnoberflächen einerseits auf den Athleten selbst, der durch unbequeme Vibrationen nicht mehr die gewünschte Leistung bringen kann (ebd.). Andererseits haben suboptimale Straßenzustände eine negative Auswirkung auf den Reibungswiderstand, was zu einem Energieverlust und somit auch Geschwindigkeitsverlust führen kann (ebd.). Um diesen leistungsmindernden Faktor in Bezug auf das Auswahlgebiet des Ironman

Austria analysieren zu können, mussten erst die entsprechenden Daten eingeholt werden. Dies wurde durch zweimaliges Abfahren der Radstrecke mit einer speziellen Smartphone Applikation namens VibSensor umgesetzt. Da Wilson et al. (2004, S. 122–123) bei den Straßenvibrationen von der hohen Relevanz des Reifendruckes spricht, wurden eine Testfahrt mit 7 Bar und eine mit 10 Bar Reifendruck durchgeführt. Die VibSensor App misst die Vibrationen durch die eingebauten Beschleunigungssensoren im Smartphone in drei Richtungen (Abbildung 4-14). Die gemessenen Werte werden im Beschleunigungsfaktor g-Kraft angegeben, wobei der Wertebereich von -2 bis +2 g reicht. Nach Schröter (2004) wird dieser Wertebereich als noch „uneingeschränkt ertragbar“ definiert. Somit wurden lediglich die Maximalwerte von exakt -2 und +2g, als schlechte Fahrbahnbeschaffenheit, in die dynamische Segmentierung aufgenommen. Die exportierten CSV Dateien wurden analysiert und sowohl die Mittelwerte der drei Aufzeichnungsachsen als auch die Mittelwerte der beiden Fahrten mit unterschiedlichem Reifendruck berechnet. Da, die Daten noch keine räumliche Verortung haben, wurden diese mit dem Timestamp des parallel aufgezeichneten GPX Tracks verknüpft.

Abbildung 4-14: Auszug aus der VibSensor App Aufzeichnungsmodus (links) und Analysemodus (rechts). Bildquelle: VibSensor App

Um nun den äußeren Leistungsparameter der Fahrbahnbeschaffenheit mit der Vibration von einem Beschleunigungsfaktor von ≥ +/-2 g-Kräften auf die Radstrecke des Ironman Austria analysieren zu können, musste der Einfluss der Fahrbahnsteigung relativiert

werden. Dies wurde im unten angeführten Diagramm durch die Differenzierung nach Steigungskategorien visualisiert (Abbildung 4-15).

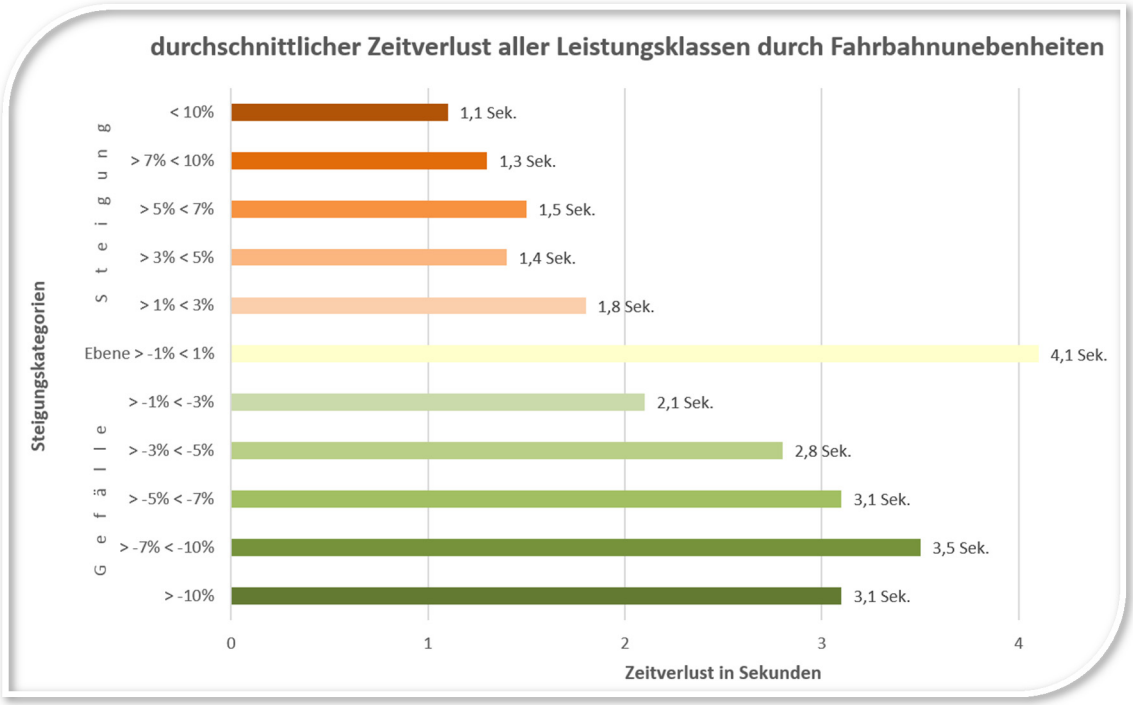


Abbildung 4-15: Zeitverlust durch Fahrbahnunebenheiten

Die geostatistische Analyse nach Fahrbahnunebenheiten zeigt, dass der absolute Zeitverlust in der Steigungskategorie von -1 bis +1% Steigung am höchsten ist. Dies liegt ganz klar darin begründet, dass diese Kategorie mit 67,2 km den größten Anteil an der Radstrecke aufweist. Der Zeitverlust von 4,1 Sekunden bei einer Fahrzeit von mindestens 4h45 ist äußerst gering. Der Gesamtverlust über alle Leistungsklassen hinweg beträgt im Durchschnitt 26 Sekunden. Als Fazit der Analyse kann gesagt werden, dass die Fahrbahnunebenheiten von ≥ +/-2 g-Kräften so partiell und kleinräumig auftreten, dass eine Simulation dieses Leistungsfaktors im Sekundenbereich liegt und somit außer Acht gelassen werden kann.

5. Workflow zur Modellierung & GeoSimulation der wissenschaftlichen Fragestellung

Die Erkenntnisse der statistischen Analysen (Kapitel 4) wurden in einem Workflow zur Modellierung und GeoSimulation eingearbeitet, um die leeren GPX Track Geometrien über die entsprechenden Geschwindigkeitsangaben mit Timestamps befüllen zu können. Die Programmierung dieser Funktionen erfolgte in der C# Programmiersprache und kann als ArcMap Add-in als graphische Benutzeroberfläche ausgeführt werden.

5.1 Graphische Benutzeroberfläche

Um eine graphische Benutzeroberfläche für das ArcMap Add-in zu entwickeln, wurde innerhalb des .NET Frameworks von Microsoft, auf die Windows Presentation Foundation (WPF) Technologie zurückgegriffen. WPF ist ein modernes graphisches Anzeigesystem für Windows und stellt einen Paradigmenwechsel von zuvor verwendeten Technologien dar (Hall 2010, S. 1–2). WPF, als gänzlich neu eingeführte Bibliothek von Klassen, wurde mit innovativen Funktionalitäten ausgestattet, wie z.B. eingebauter Hardwarebeschleunigung, Auflösungsunabhängigkeit, verbesserter Unterstützung für Audio/Video Media oder eine mächtige Animationsengine (ebd.). Um WPF zu ermöglichen, wurden nach Hall (2010, S. 1–2) folgende essentielle Subsysteme dem .Net Framework hinzugefügt, auf die im Zuge dieser Programmierung zurückgegriffen wurde:

• MIL Core (Media Integration Layer): Ein „unmanaged Wrapper“ für DirectX, welcher der CLR (Common Language Runtime = virtuelle Maschine des .Net Frameworks) die Anbindung an die DirectX Bibliothek erlaubt.

• Presentation Core: Die generischen Basistypen aus denen alle Formen und Controls der WPF abgeleitet sind. Dieses Subsystem bildet die Grundlage der WPF.

• Presentation Framework: Beinhaltet alle Komponenten der Anwenderschnittstelle, sowie die Funktionalitäten für das „Data Binding“ und andere höhere Programmierabstraktionen (wie z.B. Styles) um die Top-Level WPF Typen zur Verfügung zu stellen.

Durch die innovative WPF Technologie und der XAML Browser Anwendung (XBAP) wäre auch eine Umsetzung im World Wide Web oder als Smartphone fähige Applikation möglich. Aktuell ist die GUI und der damit verbundene Workflow über das ArcMap Add-in aufrufbar. Die folgenden Abbildungen 5-1 und 5-2 verdeutlichen den Funktionsumfang zur Modellierung und Simulation.

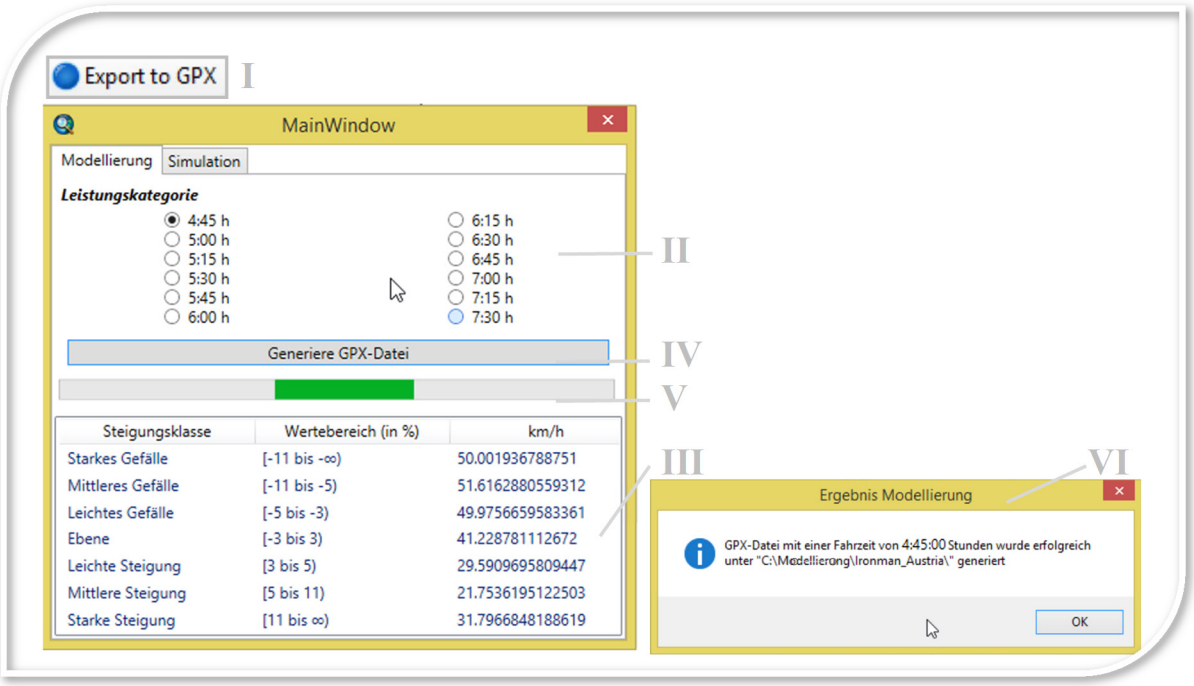


Abbildung 5-1: GUI zur Modellierung

Nach Aktivieren des ArcMap Add-in Buttons (Abbildung 5-1, I & Kapitel 5.4, I) erfolgt der Aufruf des Main Windows. Unter dem Punkt: „Leistungskategorie“ kann nun je nach gewünschter Zielzeit ein Radio Button aktiviert werden (Abbildung 5-1, II & Kapitel 5.4, II). Daraufhin erfolgt die Selektion der entsprechenden Datensätze, die in Kapitel 3.3.2 klassifiziert wurden. Es folgt eine statistische Auswertung der Mittelwerte aller gefahrenen GPX Daten einer Leistungskategorie nach Steigungsklassen. Diese Datensätze wurden innerhalb einer DBF Datei gespeichert und geben detaillierten Aufschluss über die Geschwindigkeiten und Steigungen alle 10 Meter entlang der Radstrecke. So wurde in der Abbildung 5-1, III & Kapitel 5.4, III ein Tabellenformular generiert, welches über eine Sortier- und Verschiebefunktion verfügt und die durchschnittlichen Geschwindigkeitsangaben nach Gefälle für eine Zielzeit von 04h:45m:00s ausgibt. Mit der Aktivierung des Buttons: „Generiere GPX Datei“ (Abbildung 5-1, IV & Kapitel 5.4, IV) wird der Modellierungsprozess zur Generierung eines GPX Tracks in Gang gesetzt. Während der Generierung des GPX Files ist eine Ladebalkenanimation zu sehen (Abbildung 5-1, V). Über einen Hinweis wird der Output Pfad und die generierte Zielzeit ausgewiesen (Abbildung 5-1, VI & Kapitel 5.4, VI).
Das Main Window verfügt aber nicht nur über die Funktion der Modellierung, sondern kann über das Tab Element die GUI einer GeoSimulation darstellen (Abbildung 5-2).

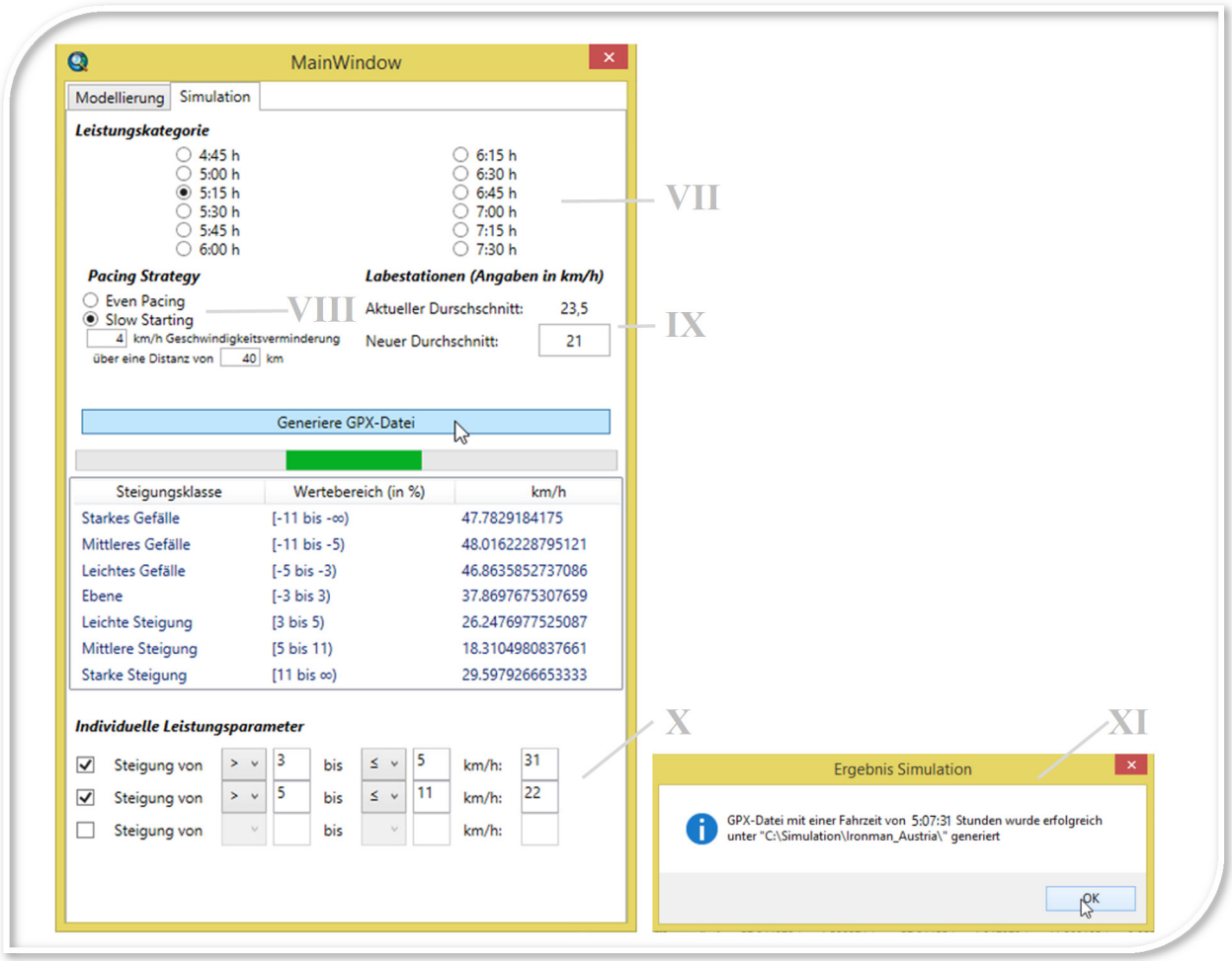


Abbildung 5-2: Funktionen des Simulations-Tabs in der GUI

Die GeoSimluation basiert auf den Basisdaten der Modellierung und somit wird als erster Schritt die gewünschte Zielzeit als Leistungskategorie ausgewählt (Abbildung 5-2, VII & Kapitel 5.4, VII). Auf Grund dieser sehr unterschiedlichen Fahrverhalten erfolgt nun die Modifikation nach sportwissenschaftlichen Aspekten wie der Pacing-Strategie (Kapitel 2.3). So kann die Even- oder Slow-Starting-Strategy durch das Ansteuern der Radio Buttons ausgewählt werden (Abbildung 5-2, VIII & Kapitel 5.4, VIII). Bei der Slow-Starting-Strategy kann über eine beliebige Distanz eine vom Benutzer definierte Geschwindigkeitsverminderung durchgeführt werden. Diese Adaptierung der Rennstrategie wird dann über die zuvor ausgewählten Fahrzeiten gerechnet. Die Analyse der empirischen Felddaten zeigte auch eine enorme Divergenz in den Fahrgeschwindigkeiten der Labestationen auf (Kapitel 4.3.2). Diese Erkenntnis wurde ebenfalls in der GUI umgesetzt, weshalb der Benutzer die berechnete Durchschnittsgeschwindigkeit in den Labestationen frei nach seinen Bedürfnissen adaptieren kann (Abbildung 5-2, IX & Kapitel 5.4, IX). Der Leistungsparameter mit dem wohl größten Einfluss auf die Radzeit des Ironman Austria ist die Topographie (Kapitel

4.3.1). Daher wurde bei der Umsetzung dieses Faktors besonders viel Zeit investiert, um eine möglichst flexible und benutzerorientierte GUI zu schaffen (Abbildung 5-2, X & Kapitel 5.4, X). Der Athlet kann über die Aktivierung von Checkboxen einen eigenen Bereich hinzufügen und diesen über Vergleichszeichen (<, ≤, >, ≥) nach Steigungen individuell definieren. Es werden alle Segmente mit diesen Angaben in den modellierten Felddaten ausgewählt und die neuen Geschwindigkeitsangaben des Benutzers prozentual verändert. Das ausgewählte Beispiel der Abbildung 5-2, liefert schlussendlich einen GPX Track mit einer Gesamtfahrzeit von 05h:07m:31s auf Basis des Fahrstiles von 05h:15m Fahrern, da entsprechende Geschwindigkeitserhöhungen in den Steigungen angewandt wurden (Abbildung 5-2, XI & Kapitel 5.4, XI).

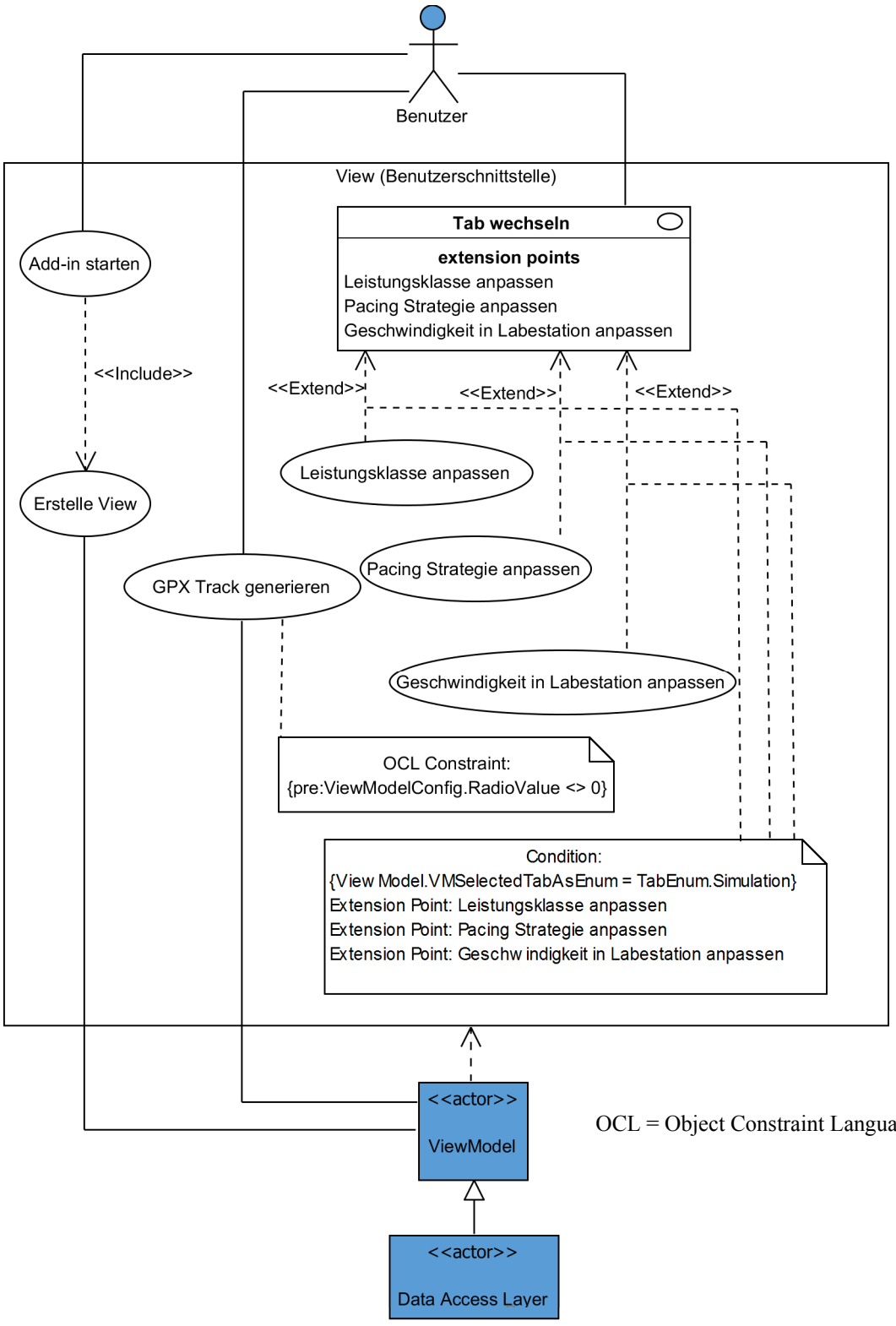
5.2 Programmierung der Modellierung und Simulation

Bei der Programmierung des Workflows zur Modellierung und Simulation als ArcMap Add-in in C# wurde auf das .NET Framework zurückgegriffen. Im Zuge dessen wurde ebenfalls das Model View ViewModel (MVVM) als Entwurfsmuster der WPF verwendet. Dieses wird eingesetzt, um die Visualisierung und Logik der GUI voneinander zu trennen. Die von Esri bereitgestellte Klassenbibliothek namens ArcObjects wurde verwendet, damit innerhalb des .NET Frameworks auf die COM Komponenten zugegriffen werden kann. Zum Export des GPX Tracks auf Basis der modifizierten Shape Daten wurde die Data Interoperability Extension von FME implementiert.

Da die Softwarelösung als ArcMap Add-in für ArcMap 10 mittels ArcObjects entwickelt wurde, und die zu lösende Problematik eine klare Erstellung eines Model Layers erschwert, wurde nicht auf eine rigorose Umsetzung des Patterns gepocht. Das Model wurde mittels eines Data Access Layers komplementiert, welcher den Zugriff und die Manipulation der Datenbank übernimmt. Aus den obigen Gründen der Komplexität wurden zeitweise auch Events anstelle von Commands appliziert.

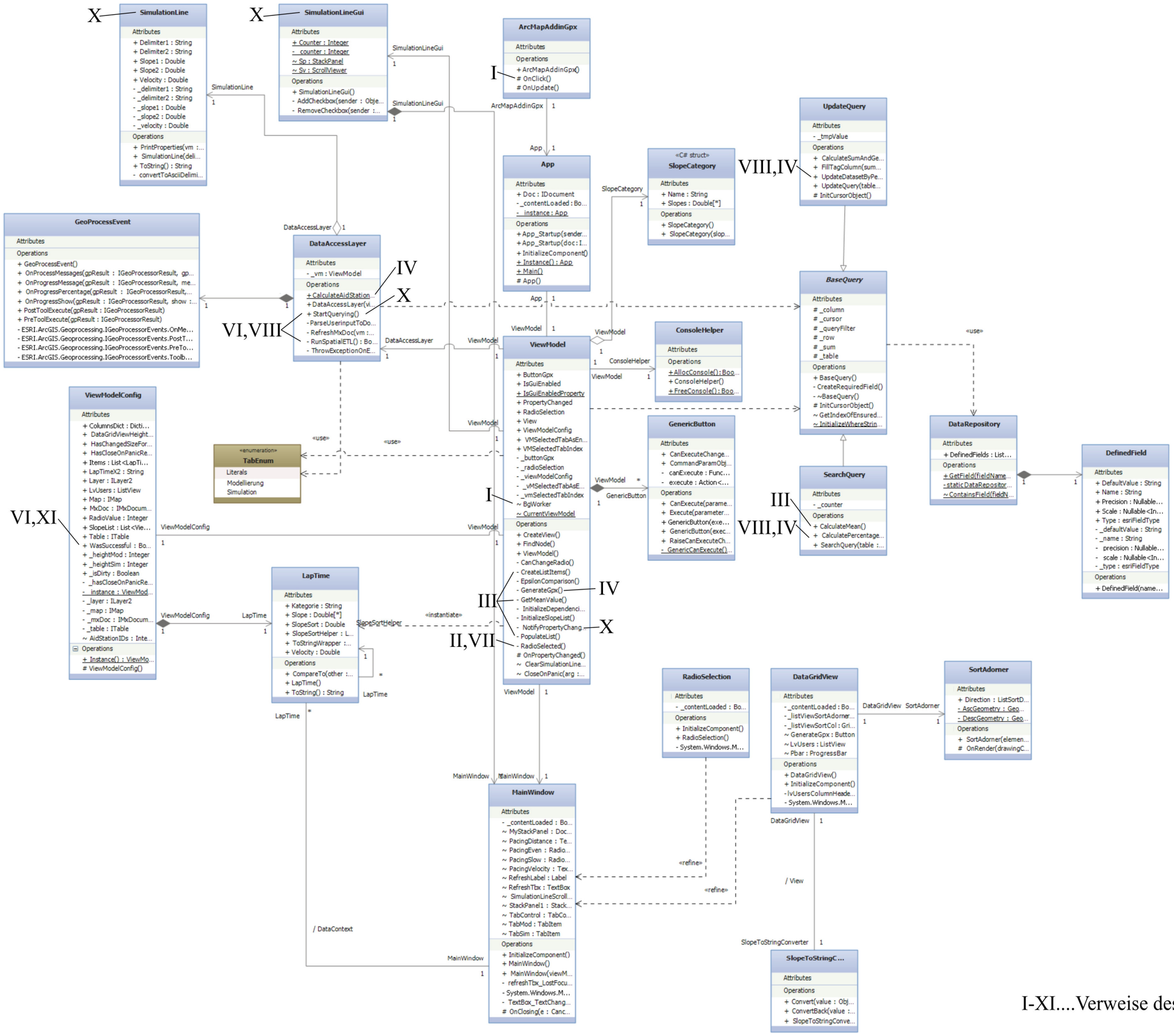
Die Umsetzung des Workflows zur Modellierung und GeoSimulation als C# Programmcode wurde in Kapitel 9.2 angehängt.

5.3 Use Case Diagramm



OCL = Object Constraint Language

5.4 UML Diagramm



I-XI....Verweise des Kapitels 5.1

6. Ergebnisse und Deduktionsschlüsse

6.1 Hypothesenüberprüfung

Die eingangs formulierte Hypothese: „Ein Athlet, der eine ständige Anweisung und Rückmeldung zur Einhaltung der Rennstrategie bekommt, wird durch optimale Pacing-Strategien aus sportwissenschaftlichen Erkenntnissen eine konstantere und demnach schnellere Radleistung im Wettkampf erbringen können“ konnte im Zuge dieser Arbeit bestätigt werden (Datenvalidierung siehe Kapitel 6.2). Durch den Einsatz von GPS Technologien und der Erstellung von GPX Tracks als Marschroute gelangt der Athlet zu einer laufenden Überprüfung seiner Rennstrategie. Die raum-zeitliche Modellierung der Radstrecke ermöglicht, über Hinweise des GPS Devices, aktuelle Rückmeldungen an jeglicher Position entlang der Strecke. Die Analysen der 107 empirischen Felddaten haben gezeigt, dass die sportwissenschaftlichen Empfehlungen durchwegs nicht eingehalten werden konnten. Wenn der Athlet die Vorgabe der Marschroute umsetzt, so sind stetige Geschwindigkeitsverminderungen durch überambitionierte Athleten (Positive-Pacing-Strategy nach Abbiss et al. 2006, S. 726) nicht mehr möglich. Das Abfahren eines generierten GPX Tracks bewirkt, dass der Athlet die erste und zweite Radrunde bis in den Sekundenbereich gleichmäßig zu absolvieren hat. Nicht nur die Hypothese konnte in dieser Masterarbeit positiv validiert werden, sondern es wurde auch die Zielsetzung der wissenschaftlichen Fragestellung erfüllt. So ist es möglich, auf Basis der in der Sportwissenschaft vorherrschenden „pacing-strategies“ eine, für jeden Athleten spezifische und nach seinen Leistungsparametern definierte, Modellierung und Geosimulation der Radstrecke am Beispiel des Ironman Austria als Marschroute für GPS Geräte umzusetzen. In der graphischen Benutzeroberfläche zur GeoSimulation wurden sowohl die Pacing-Strategien als auch die Leistungsparameter eingearbeitet, um dem Benutzer eine flexible Definition seiner Marschroute zu ermöglichen (Abbildung 5-2). Eine konkrete Überprüfung der Ergebnisse zur Modellierung bzw. Geosimulation wurde im anschließenden Kapitel 6.2 durchgeführt.

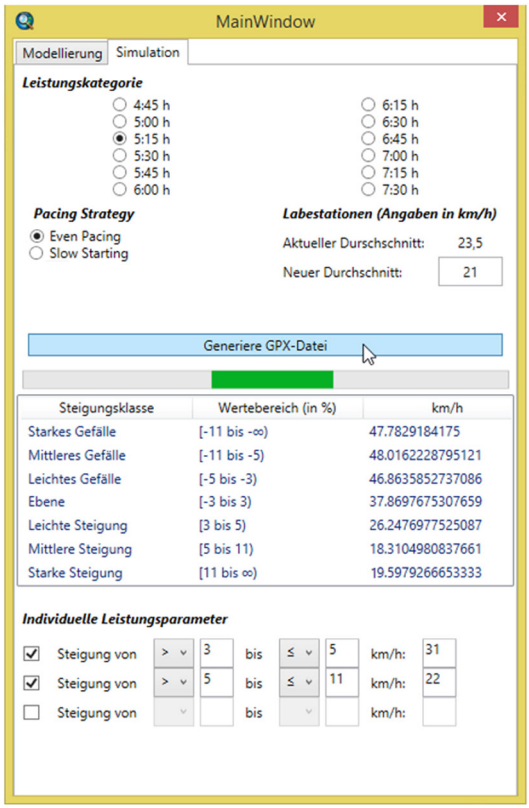
6.2 Deduktiver Datenvergleich zwischen Simulation und Wettkampfaufzeichnungen der Athleten

Der Kreislauf von der Empirie über die Induktion zur Theorie wurde in der Abbildung 1-5 dargestellt. Nun folgt der Rückschluss über die Deduktion vom Modell zur Empirie. Durch die Anwendung des programmierten Workflows zur GeoSimulation wird eine konkrete Überprüfung der Hypothese möglich. Zwei Athleten des Ironman Austria wurden nach der Einschätzung ihrer eigenen Wettkampfleistung gefragt. Sie sollten durch die Definition ihrer Leistungsparameter und der Pacing-Strategie über die programmierte GUI ihre tatsächlich absolvierte Wettkampffahrt simulieren. Es folgen nun die Beschreibungen der Athleten, ihre Eingaben in der GUI und die Analysen der generierten GPX Tracks im Vergleich zu ihrer tatsächlichen Wettkampffahrt.

Athlet I:

Befragung

Alter:	32
bisherige Langdistanz Erfahrung:	4 Wettkämpfe
Gesamtzeit des Ironman Austria:	10h::12m:31s
Radzeit des Ironman Austria:	05h::08m:10s
Durchschnitzsgeschwindigkeit:	34,9 km/h
1. Runde:	02h:24m:54s
2. Runde:	02h:43m:16s
Differenz Runde 1 und Runde 2:	00h:09m:11s
verwendete Pacing Strategie:	keine
qualitative Einschätzung der Radleistung:	Im Vergleich zu anderen Athleten in der Leistungskategorie war die Geschwindigkeit, bei leichter bis mittlerer Steigung, wesentlich höher. Hingegen kam es in den Labestationen zu einem geringen Leistungsrückgang.
Schlussfolgerung für die Simulation mit Programm:	Geschwindigkeitswerte eingeben und in den Labestationen etwas geringere.



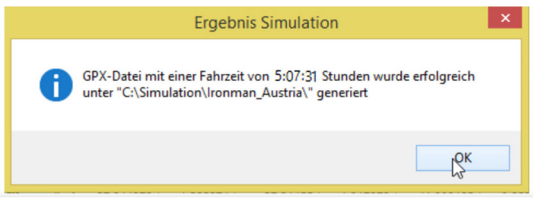


Abbildung 6-1: Qualitative Befragung und Simulation des Athleten I

Athlet I versuchte seine tatsächlich absolvierte Wettkampffahrt von 05h:08m:10s mit dem erzeugten Workflow zur GeoSimulation nachzubilden. Die Leistungsparameter wurden

nach eigener Einschätzung definiert und auf Basis der empirischen Felddaten von 05h:15m Athleten ausgewählt. Die Geschwindigkeiten dieser Leistungsklasse werden nach Steigungsklassen im Tabellenbereich der GUI visualisiert. Auf Grundlage dieser Angabe wurden zwei Veränderungen der Geschwindigkeit in der Steigung zwischen >3% und ≤5% Steigung bzw. >5% und ≤11%, durchgeführt (Abbildung 6-2).

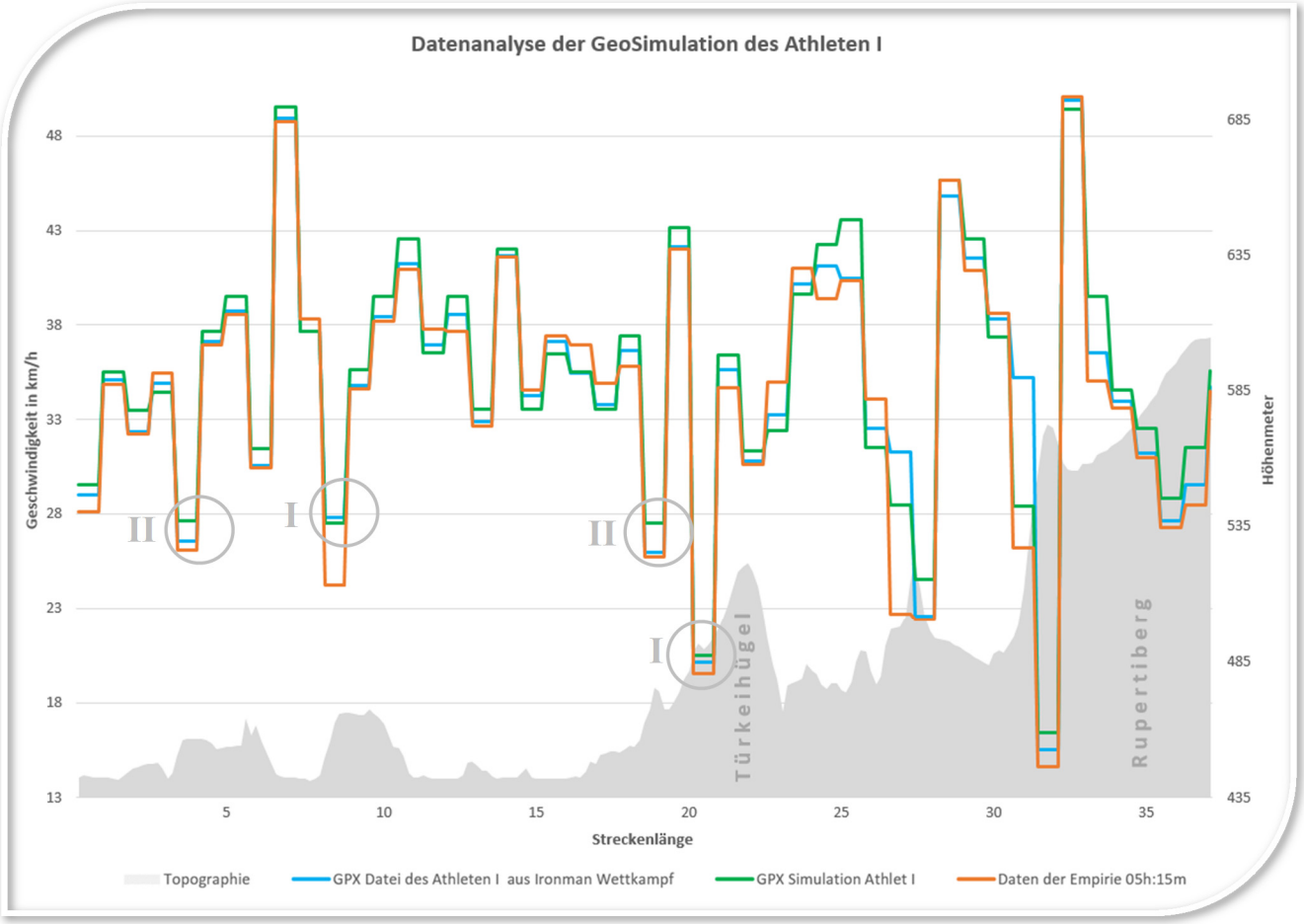


Abbildung 6-2: Datenanalyse der GeoSimulation des Athleten I

Die Ergebnisse der durchgeführten GeoSimulation des Athleten I wurden im Diagramm der Abbildung 6-2 visualisiert. Neben der simulierten GPX Tracks wurden weiters der tatsächlich gefahrene File des Wettkampfes und die Daten zur Empirie von typischen 05h:15m Fahrern auf den ersten 37 Fahrkilometer dargestellt. Durchgängig kann festgestellt werden, dass die empirischen Felddaten über alle Steigungen hinweg die niedrigste Geschwindigkeit aufweisen. Dies ist auch klar begründbar, da auf Basis dieser typischen Fahrverhalten von 05h:15m Athleten, eine Simulation mit einer Geschwindigkeitserhöhung angewandt wurde. Dieser simulierte GPX Track kommt der tatsächlichen Wettkampfaufzeichnung, speziell in leicht ansteigenden Abschnitten, schon sehr nahe (Abbildung 6-2, I). So gibt es in den Segmenten mit >3% und ≤5% Steigung eine durchschnittliche Differenz von nur 0,5 km/h. Bei den Segmenten mit >5% und

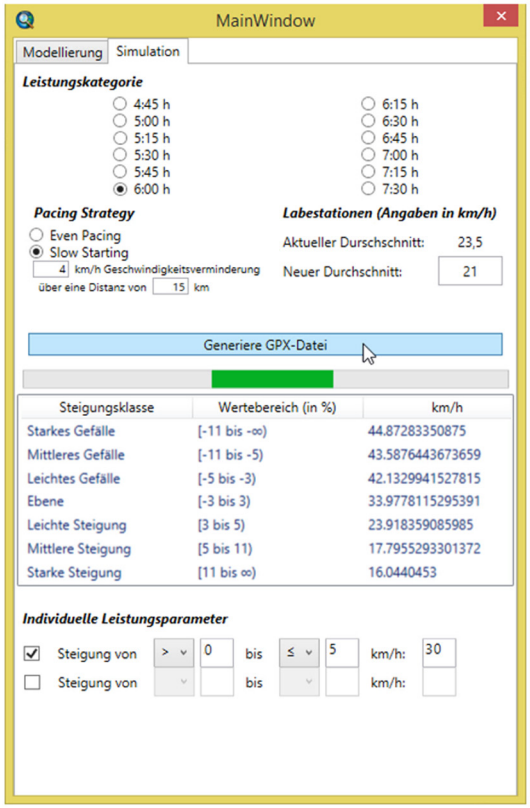
≤11% Steigung hat der Athlet den Geschwindigkeitszuwachs von 18km/h auf 22 km/h etwas zu hoch angesetzt (Abbildung 6-2, II). Eine Veränderung um 20,2 km/h im angesprochenen Leistungsparameter, hätte ein optimales Ergebnis erzielt und den tatsächlich gefahrenen GPX Track beinahe perfekt nachgebildet. Als Fazit ist der simulierte GPX Track mit einer maximalen Abweichung von 1,8 km/h in den Steigungssektoren als durchaus brauchbares Resultat zu klassifizieren.

Um die Funktionsfähigkeit der GeoSimulation auch bei niedrigeren Leistungskategorien zu testen, wurde ein weiterer Athlet um Nachbildung seiner Wettkampffahrt gebeten.

Athlet II:

Befragung

Alter:	51
bisherige Langdistanz Erfahrung:	keine Erfahrung
Gesamtzeit des Ironman Austria:	12h::48m:35s
Radzeit des Ironman Austria:	05h::49m:38s
Durchschnitzsgeschwindigkeit:	30,7 km/h
1. Runde:	02h:40m:15s
2. Runde:	03h:09m:23s
Differenz Runde 1 und Runde 2:	00h:14m:34s
verwendete Pacing Strategie:	Slow-Starting-Strategy
qualitative Einschätzung der Radleistung:	Auf Grund der nicht vorhandenen Wettkampferfahrung wurde das Tempo auf die ersten 15km bewusst reduziert. Bei leichter Steigung konnten viele andere Athleten überholt werden.
Schlussfolgerung für die Simulation mit Programm:	Die Slow-Starting-Strategy soll auf 15km simuliert werden. Weiters erfolgte eine leichte Erhöhung des Durchschnittsstempos in der Ebene und geringer Steigung.



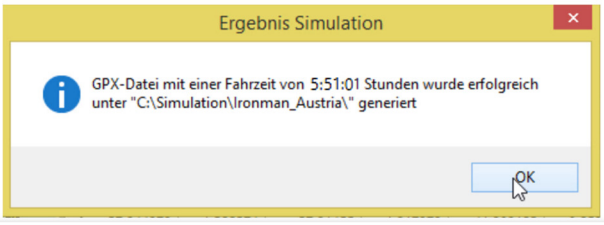


Abbildung 6-3: Qualitative Befragung und Simulation des Athleten II

Athlet II gab an, im Wettkampf bewusst eine Slow-Starting-Strategie verwendet zu haben, welche durch eine Geschwindigkeitsveränderung um 4 km/h über 15 km nachgebildet wurde (Abbildung 6-3).

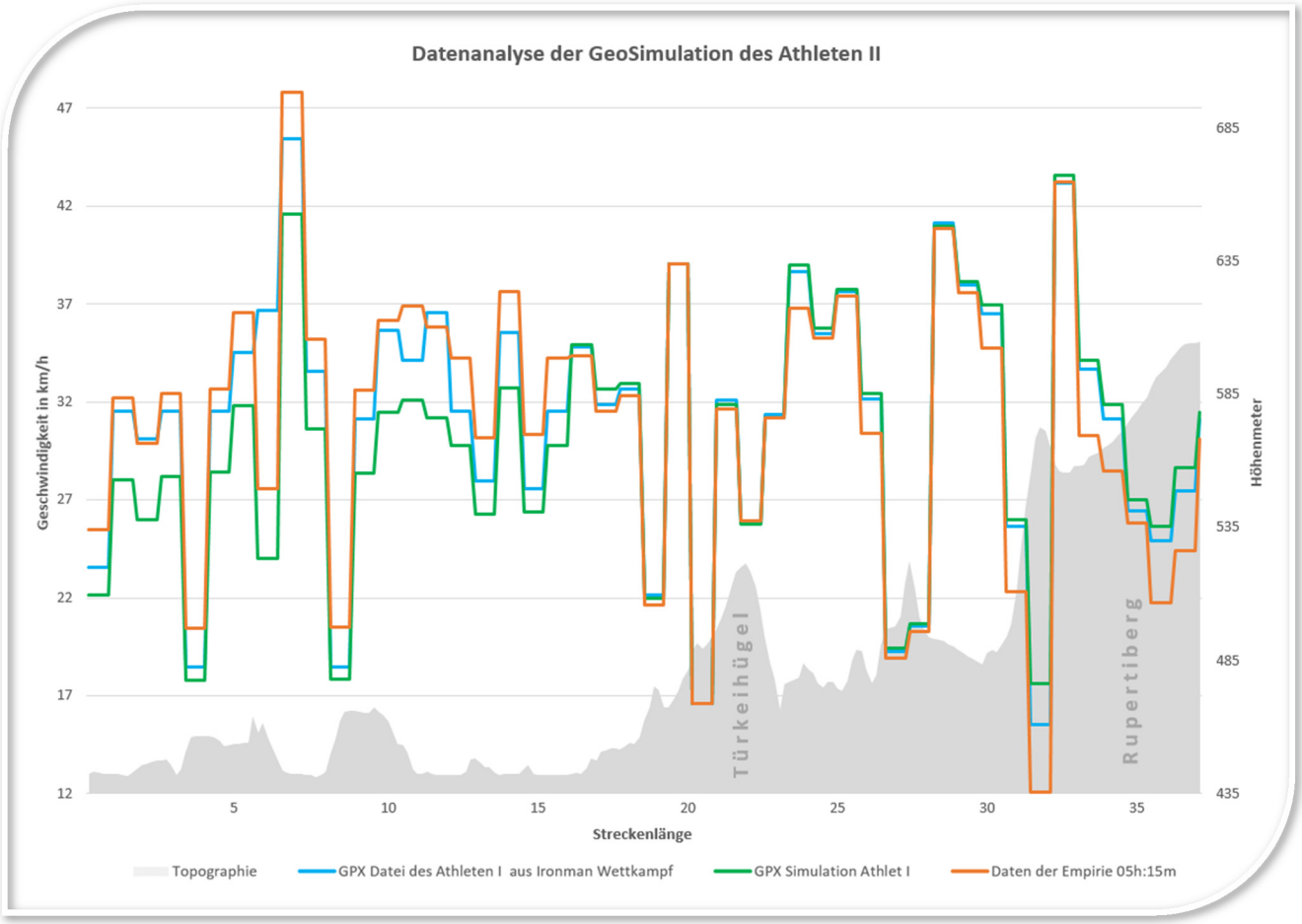


Abbildung 6-4: Datenanalyse der GeoSimulation des Athleten II

Die Analyse der GeoSimulation, des tatsächlichen Wettkampf Tracks und der empirischen Felddaten zeigt den Einfluss der Pacing-Strategie auf die Geschwindigkeit. So liegen die simulierten Daten und die Wettkampfdaten bis Kilometer 15 durchwegs unter dem Tempo der empirischen Felddaten von 06h:00m Athleten (Abbildung 6-4). Ab Kilometer 15 ändert sich dieser Trend und die die empirischen Werte liegen stets unter der GeoSimulation und des Wettkampf Tracks. Dabei wurde die angegebene Geschwindigkeitsverminderung um 4 km/h in der Praxis mit 2 km/h umgesetzt. Über die restliche Strecke hinweg wurde der Fahrstil der 06h:00m Athleten herangezogen bis auf die Sektoren mit >0% und ≤5% Steigung. In diesen Abschnitten schätzte sich der befragte Sportler korrekterweise, mit angegebenen 30km/h, schneller als der typische 06h:00m Athlet ein. Durch die definierten Eingabeparameter konnte der Benutzer den tatsächlichen GPX Track des Wettkampfes, von 05h:49m:38s, mit einer Abweichung von 01m:23s simulieren (Abbildung 6-3).

7. Diskussion und Ausblick

7.1 Nachvollziehbarkeit, Transparenz und Übertragbarkeit

Durch die Visualisierung des konzeptuellen Modells und durch das Kapitel 1.5, Struktur der Arbeit, wurde versucht eine gewisse Nachvollziehbarkeit für den Leser der Arbeit zu schaffen. Diese Prämisse wurde durch die Formulierung einer klaren wissenschaftlichen Fragestellung und durch Erläuterung der theoretischen Grundlagen weiter geführt. Ein weiterer Grundsatz dieser Arbeit war es, die Transparenz der eingesetzten Methoden und der verwendeten GeoDaten zu wahren. So wurde etwa der komplette Programmcode zum Workflow der Modellierung und GeoSimulation beigefügt (Kapitel 5.2). Des Weiteren erfolgte eine genaue Abbildung der empirischen Felddaten mit der deskriptiven Statistik und der Erläuterung zur Datenqualität. Die Übertragbarkeit des Workflows zur Modellierung und Simulation auf einen anderen Datensatz, sprich auf ein anderes Beispielgebiet und somit differierendes Rennen, ist ohne größeren Aufwand durchaus möglich. Durch die Verwendung des .NET Frameworks und mit der System .XML assembly besteht die Möglichkeit der Konvertierung von SHP in GPX, ohne an das ArcGis Backend gebunden zu sein.

7.2 Schlussfolgerungen und Vorschläge

Die Ergebnisse der deduktiven Datenanalyse zeigen, dass die GPX Dateien, welche durch die GeoSimulation der Athleten generiert wurden, leicht von den realen Wettkampfaufzeichnungen abweichen (Kapitel 6.2). Diese marginale Divergenz ist durch die leicht überschätzten Eingaben der Leistungsparameter in der GUI zu erklären. Als Fazit der GeoSimulation ist zu sagen, dass die Fahrweise der Athleten innerhalb der Leistungskategorien sehr stark variieren kann und daher eine sehr feine Abstimmung und Adaptierung der Leistungsparameter meist unerlässlich ist. Durch die verwendete Windows Presentation Foundation (WPF) Technologie wäre es denkbar, dass die Modellierung und Geosimulation mittels der XAML Browser Anwendung (XBAP) im World Wide Web publiziert wird. XAML ist eine XML basierte Sprache, die es ermöglicht, auf deklarativem Wege Objekte zu erstellen und deren Eigenschaften zu setzen. Mit diesen Technologien wäre es für den Athleten möglich, über einen Browser auf die graphische Benutzeroberfläche zu gelangen und den modellierten bzw. simulierten GPX File, losgelöst von der ArcMap Software, zu generieren. Auch eine

Smartphone fähige Applikation wäre denkbar, um den Bedienungskomfort für die Benutzer noch weiter zu erhöhen. Ein anderer Vorschlag zur Weiterentwicklung wäre es, neben dem Output als GPX Datei eine kartographische und tabellarische Visualisierung anzubieten. So könnte sich der Benutzer noch vor der Verwendung der Marschroute mit den vorgegebenen Geschwindigkeitsangaben vertraut machen. Eine tabellarische Auswertung der Sektoren mit den semantischen Rauminformationen und eine kartographische Visualisierung der Radstrecke mit den Geschwindigkeitsangaben könnte dem Benutzer eine zusätzliche Hilfestellung sein, um die gewünschten Pacing-Strategien umzusetzen.

7.3 Kritische Einschätzung

Der größte Kritikpunkt, der im Zuge dieser Arbeit zu benennen ist, hängt mit der Stichprobe zur Empirie zusammen. Durch die Rücklaufquote von 17,2 % konnten nicht alle Leistungskategorien der tatsächlich erbrachten Wettkampfergebisse in der Stichprobe abgebildet werden. So ist in den empirischen Felddaten keine Radzeit unter 4h:39m vorhanden, die tatsächlich jedoch von insgesamt 19 Athleten unterboten wurde. Ein weiterer Kritikpunkt an den empirischen Felddaten und der Stichprobe ist die teils fehlende Repräsentativität. Durch die Klassifizierung der Ergebnisse in 12 Leistungskategorien konnte das Mindestmaß von 30 Merkmalswerten pro Klasse nicht durch gängig eingehalten werden Zimmermann-Janschitz (2014, S. 94). Eine weitere Erkenntnis, die beim Verfassen dieser Masterarbeit gewonnen wurde, ist die Vielfalt an Parametern, welche einen Einfluss auf die Leistung haben. So wurden in Kapitel 4.3 die Topographie, Labestationen, Kurvenradien und Fahrbahnbeschaffenheit anhand der empirischen Felddaten analysiert. Nach Abbiss et al. (2006, S. 727) bestehen jedoch weitere externe Leistungsparameter, wie die Windrichtung, Windintensität oder das Drafting, welche Einfluss auf die Rennzeiten haben. Diese Parameter sind aber so dynamisch und nicht vorhersehbar, dass eine Modellierung kaum umsetzbar ist. So sind die Effekte des Draftings beim legalen Abstand von 10 Meter, also das Ausnützen des Raumes mit geringer Windgeschwindigkeit hinter dem vorderen Athleten, virtuell nicht abbildbar. Ein weiterer Aspekt, der an dieser Stelle erwähnt werden soll, betrifft den gleichmäßigen Leistungsabruf der Athleten nach sportwissenschaftlichen Erkenntnissen. So wird die Maßeinheit für die Umsetzung zur Pacing-Strategie hauptsächlich in Watt angegeben. Der Vorteil dabei liegt in der Unabhängigkeit gegenüber der Leistungsparameter wie Steigung oder Windverhältnisse. In dieser Arbeit wurde

hingegen der Fokus auf die Zielsetzung der Athleten gelegt, wo nach einer Studie von Lehmann (2016, S. 5) für 80% aller Triathlon Sportler die Rennzeit ein wichtiges Motiv ist. Durch den Fokus auf die Geschwindigkeit als Maßeinheit können die Vorstellungen und Zielvorgaben der Athleten auf die Sekunde genau modelliert bzw. simuliert werden.

8. Literaturverzeichnis

ABBISS, CHRIS R.; LAURSEN, PAUL B. (2008): Describing and Understanding Pacing Strategies during Athletic Competition. In: *Sports Medicine* 38 (3), S. 239–252.

ABBISS, CHRIS R.; QUOD, MARC J.; MARTIN, DAVID T.; NETTO, KEVIN J.; NOSAKA, KAZUNORI; LEE, HAMILTON; SURRIANO, ROB; BISHOP, DAVID; LAURSEN, PAUL B. (2006): Dynamic pacing strategies during the cycle phase of an Ironman triathlon. In: *Med Sci Sports Exerc* 38 (4), S. 726–734.

AISBETT, BRAD; LE ROSSIGNOL, PETER; MCCONELL, GLENN K.; ABBISS, CHRIS R.; SNOW, ROD (2009): Effects of starting strategy on 5-min cycling time-trial performance. In: *Journal of Sports Sciences* 27 (11), S. 1201–1209.

AMT DER KÄRNTNER LANDESREGIERUNG (2016): Open Data Kärnten. URL: http://data.ktn.gv.at, Zugriff: 01.10.2015.

BALZERT, HELMUT; BENDISCH, ROMAN; KERN, UWE; SCHÄFER, CHRISTIAN; SCHRÖDER, MARION; ZEPPENFELD, KLAUS (2008): Wissenschaftliches Arbeiten. Herdecke: W3L-Verlag.

BARTELME, NORBERT (2005): Geoinformatik. Modelle, Strukturen, Funktionen. Berlin: Springer.

BOOTH, BOB; MITCHELL, ANDY (2001): Getting started with ArcGIS. Redlands, CA: Environmental Systems Research Institute.

BOSWELL, GRAEME P. (2012): Power variation strategies for cycling time trials: A differential equation model. In: *Journal of Sports Sciences* 30 (7), S. 651–659.

BRUNSDON, CHRIS (2007): Path estimation from GPS tracks. In: Proceedings of the 9th International Conference on GeoComputation. National Centre for Geocomputation, Maynooth University, S. 1–9.

BUNDESAMT FÜR EICH- UND VERMESSUNGSWESEN (2016): Austrian Map. URL: http://www.austrianmap.at, Zugriff: 18.01.2016.

BURKE, LOUISE (2007): Practical sports nutrition. Champaign, IL: Human Kinetics.

DOLAN, SHAWN H.; HOUSTON, MELINDA; MARTIN, SCOTT B. (2011): Survey results of the training, nutrition, and mental preparation of triathletes: Practical implications of findings. In: *Journal of Sports Sciences* 29 (10), S. 1019–1028.

ESRI (2001): Linear Referencing and Dynamic Segmentation in ArcGIS 8.1. URL: http://www.unigis.ac.at/fernstudien/unigis_professional/Materialien/module/modul2/dynamic_segmentation_arcgis.pdf, Zugriff: 18.02.2016.

ESRI (2016a): Calculating slope. URL: http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?TopicName=Calculating%20slope, Zugriff: 12.02.2016.

ESRI (2016b): Esri ASCII-Raster-Format. URL: http://desktop.arcgis.com/de/desktop/latest/manage-data/raster-and-images/esri-ascii-raster-format.htm, Zugriff: 15.03.2016.

ESRI (2016c): GIS Dictionary. URL: http://support.esri.com/en/knowledgebase/GISDictionary/term/aggregation, Zugriff: 15.02.2016.

ESRI (2016d): Gleichungsbasierte Methoden zur geographischen Transformation. URL: http://desktop.arcgis.com/de/desktop/latest/guide-books/map-projections/equation-based-methods.htm, Zugriff: 13.02.2016.

ESRI (2016e): Topology Rules. URL: http://help.arcgis.com/de/arcgisdesktop/10.0/help/001t/pdf/topology_rules_poster.pdf, Zugriff: 09.02.2016.

FRIEL, JOE (2012): The Six Most Common Triathlon Mistakes. URL: http://triathlete-europe.competitor.com/2012/08/20/joe-friel-the-six-most-common-triathlon-mistakes, Zugriff: 15.08.2013.

FRIEL, JOE; VANCE, JIM (2013): Triathlon science. Champaign, IL: Human Kinetics.

GARMIN (2013): Edge 810 Handbuch. URL: http://static.garmincdn.com/pumac/Edge810_OM_DE.pdf, Zugriff: 25.03.2016.

GEOFABRIK (2016): OpenStreetMap Data Extracts. URL: https://download.geofabrik.de/, Zugriff: 12.03.2016.

HALL, GARY (2010): Pro WPF and Silverlight MVVM. New York: Springer Science.

HEIMBUCHNER, KLAUS (2014): GIP Day. In: Thomas Blaschke &. Gerald Griesebner Josef Strobl (Hg.): Angewandte Geoinformatik 2014. Berlin: Wichmann Verlag, S. 467–478.

KLEINE ZEITUNG (2013a): Ironman 2014 ausverkauft. URL: http://www.kleinezeitung.at/sport/ironman/3347238/ironman-2014-klagenfurt-schon-ausverkauft.story, Zugriff: 16.08.2013.

KLEINE ZEITUNG (2013b): Teilnehmer beim Ironman in Klagenfurt. URL: http://www.kleinezeitung.at/sport/ironman/3343422/3-000-teilnehmer-beim-ironman-klagenfurt-start.story, Zugriff: 16.09.2015.

KONING, JOS J. DE; FOSTER, CARL; BAKKUM, ARJAN; KLOPPENBURG, SIL; THIEL, CHRISTIAN; JOSEPH, TRENT; COHEN, JACOB; PORCARI, JOHN P.; LUCIA, ALEJANDRO (2011): Regulation of Pacing Strategy during Athletic Competition. In: *Plus One* 6 (1), S. 1–6.

KROMREY, HELMUT (2002): Empirische Sozialforschung. Opladen: Leske und Budrich.

LANGE, NORBERT DE (2013): Geoinformatik in Theorie und Praxis. Berlin: Springer Spektrum.

LAURSEN, PAUL B. (2011): Long distance triathlon: demands, preparation and performance. In: *Journal of Human Sport and Exercise* 6 (2), S. 247–263.

LEHMANN, JENNIFER (2016): Persönlichkeitsstruktur und Motive von Ironman-Teilnehmern im Vergleich zu Marathon-Läufern. URL: http://www.uni-regensburg.de/psychologie-paedagogiksport/sportwissenschaft/medien/studienergebnisse.pdf, Zugriff: 23.01.2016.

LEPERS, ROMUALD (2008): Analysis of Hawaii Ironman Performances in Elite Triathletes from 1981 to 2007. In: *Medicine & Science in Sports & Exercise* 40 (10), S. 1828–1834.

LUND, BILL (1996): Triathlon extreme Sports. Mankato, MN: Capstone Press.

MADDISON, RALPH; NI MHURCHU, CLIONA (2009): Global positioning system: a new opportunity in physical activity measurement. In: *International Journal of Behavioral Nutrition and Physical Activity* 6 (1), S. 73–81.

MORRIS, SCOTT (2016): TopoFusion PRO 5.2. URL: http://topofusion.com/, Zugriff: 19.01.2016.

NATIONAL WEATHER SERVICE WEATHER FORECAST OFFICE (2016): NOAA Honolulu Hawaii. URL: http://www.prh.noaa.gov/hnl/, Zugriff: 20.02.2016.

NEUMANN, GEORG; PFÜTZNER, ARNDT; HOTTENROTT, KUNO (2004): Das grosse Buch vom Triathlon. Aachen: Meyer und Meyer.

OPEN GEOSPATIAL CONSORTIUM (2016): OGC Standards. URL: http://www.opengeospatial.org/docs/is, Zugriff: 13.02.2016.

ÖSTERREICHISCHER RUNDFUNK (2011): "Licht ins Dunkel" Ironman Austria 2012. URL: http://lichtinsdunkel.orf.at/?story=3711, Zugriff: 16.08.2013.

PENTEK TIMING (2010): Results of the Ironman Austria 2010. URL: http://results.pentek-timing.at/results/show_results.php?v=11322, Zugriff: 24.01.2016.

PETSCHNIG, STEFAN (2007): 10 Jahre Ironman Triathlon Austria. Aachen: Meyer & Meyer.

POLAR (2016): V800 Anleitung. URL: http://www.polar.com/e_manuals/V800/Polar_V800_user_manual_Deutsch/manual.pdf, Zugriff: 26.01.2016.

RESEARCH STUDIOS AUSTRIA iSPACE (2016): Windatlas Österreich. URL: http://ispacevm11.researchstudio.at/index_v.html, Zugriff: 20.02.2016.

RÜST, CHRISTOPH A.; KNECHTLE, BEAT; KNECHTLE, PATRIZIA; WIRTH, ANDREA; ROSEMANN, THOMAS (2012): A Comparison of Anthropometric and Training Characteristics among Recreational Male Ironman Triathletes and Ultra-Endurance Cyclists. In: *The Chinese Journal of Physiology* 55 (2), S. 114–124.

SCHRÖTER, ECKHART (2004): Bewegung im Raum durch G-Belastung. URL: http://www.dhv.de/web/index.php?id=1199, Zugriff: 28.03.2016.

THIELER, E. ROBERT; SMITH, THERESA L.; KNISEL, JULIA M.; SAMPSON, DANIEL W. (2013): Massachusetts Shoreline Change Mapping and Analysis Project. US Geological Survey.

TOMTOM (2016): Multi-Sport Referenzhandbuch. URL: http://download.tomtom.com/open/manuals/Runner_Multi-Sport/refman/TomTom-Runner-Multi-Sport-RG-de-de.pdf, Zugriff: 26.01.2016.

TOPOGRAFIX (2016): GPX for Developers. URL: http://www.topografix.com/gpx_for_developers.asp, Zugriff: 13.02.2016.

TURNER, KEITH (1997): What's the Difference Among 2-D, 2.5-D, 3-D and 4-D? In: *GIS WORLD* 10 (3), S. 45–55.

WILSON, DAVID G.; PAPADOPOULOS, JIM; WHITT, FRANK R. (2004): Bicycling science. Cambridge, MS: Massachusetts Institute of Technology Press.

WORLD TRIATHLON CORPORATION (2016): Ironman Austria. URL: http://www.ironman.com/de-at/triathlon/events/emea/ironman/austria, Zugriff: 20.02.2016.

WORLD WIDE WEB CONSORTIUM (2016): About W3C. URL: https://www.w3.org/, Zugriff: 08.03.2016.

WU, SAM SHI XUAN; ABBISS, CHRIS R.; PEIFFER, JEREMIAH; BRISSWALTER, JEANICK; NOSAKA, KAZUNORI (2014): Factors influencing pacing in triathlon. In: *Open Access Journal of Sports Medicine*, S. 223–234.

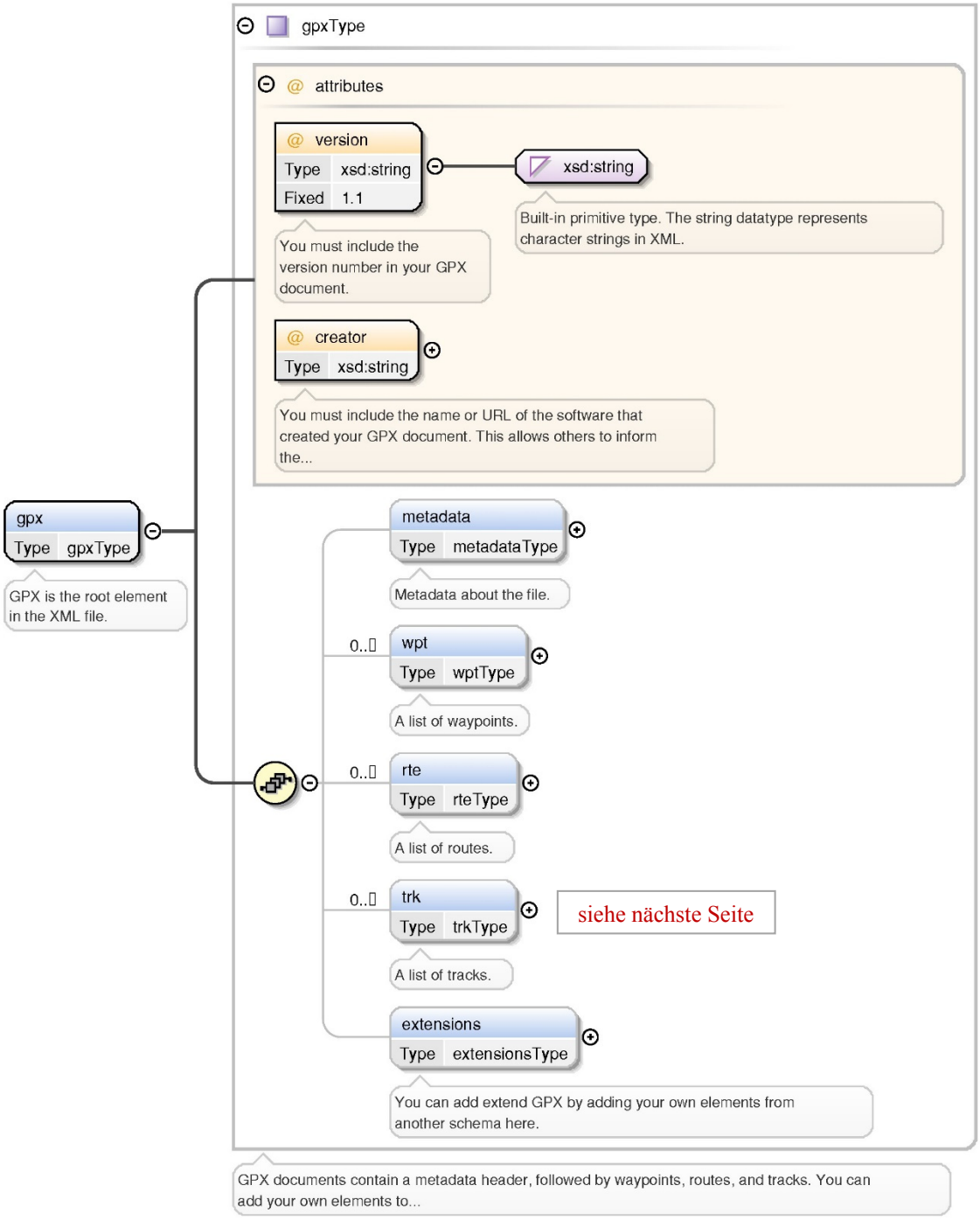
ZEITLINGER, MICHAEL (2013): Als Ironman wird man durch ganz Kärnten getragen. URL: http://www.ktn.gv.at/27987p_DE-ktn.gv.at?newsid=21651, Zugriff: 16.08.2013.

ZIMMERMANN-JANSCHITZ, SUSANNE (2014): Statistik in der Geographie. Berlin: Springer Spektrum.

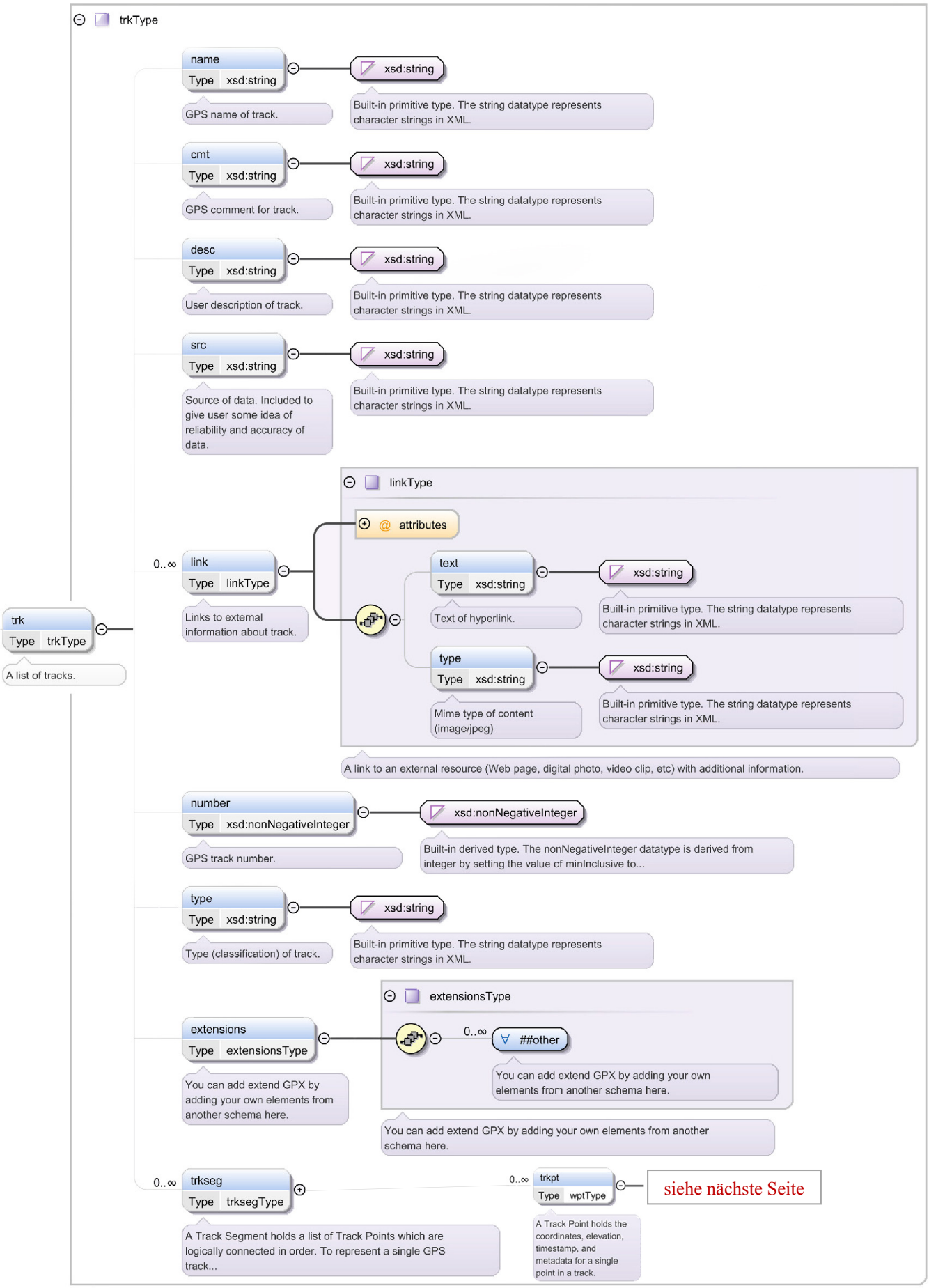
9. Anhang

9.1 XSD Schema des GPX Formates

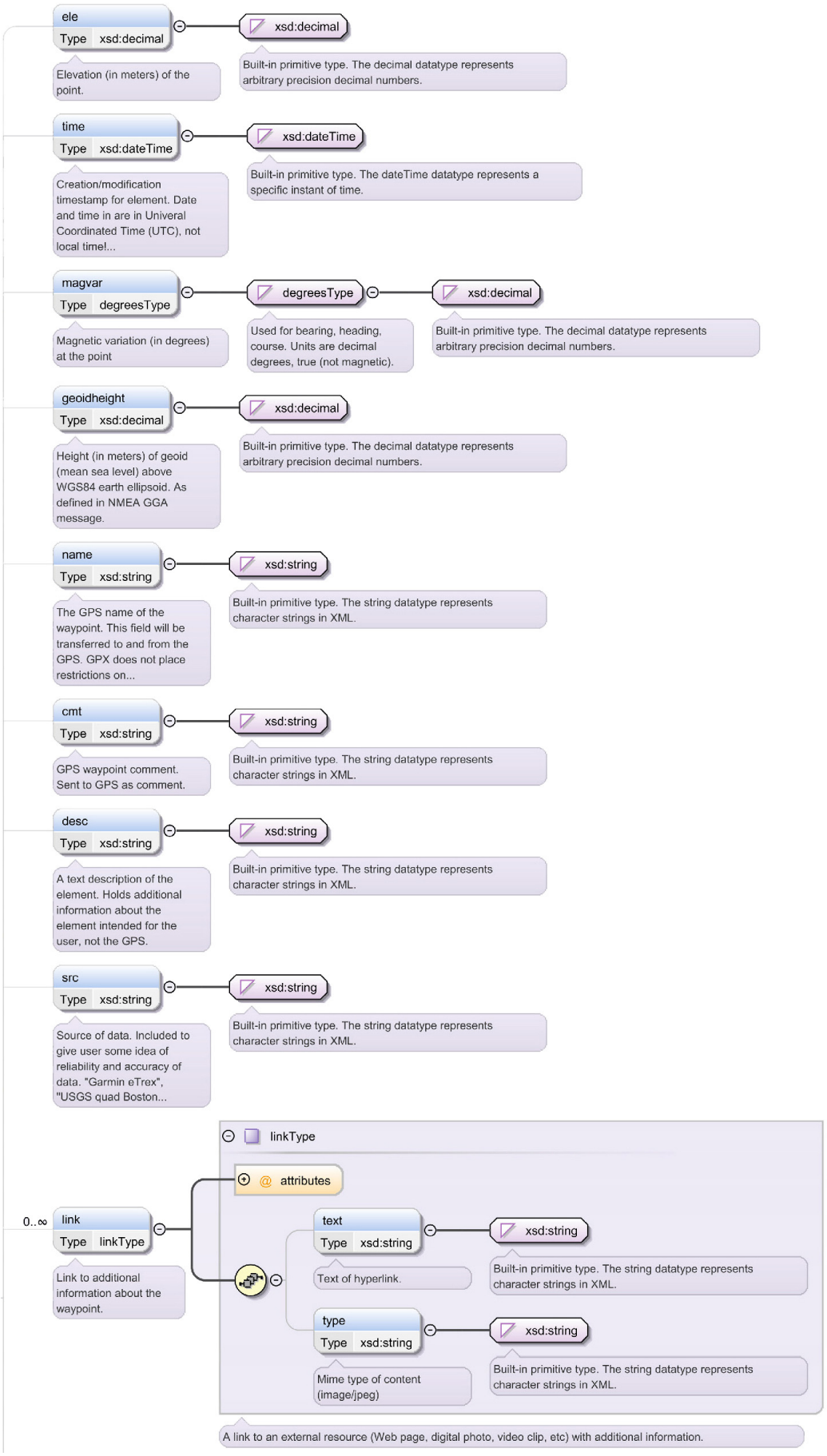
Teil I von III



GPX Schema Teil II von III (Track)



GPX Schema Teil III von III (Track Point)



9.2 Programmcode zur Modellierung und GeoSimulation

App.xaml 1

```
1  <Application x:Class="GpxWpfApplication.App"
2               xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3               xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4               Startup="App_Startup"
5               ShutdownMode="OnMainWindowClose">
6      <!-- Class gibt die Code-behind-Datei der App.xaml an.
7           Das Startup Attribut legt den Programmeinstieg der WPF Applikation
              fest.
8           Mittels xmlns werden die fuer das Projekt notwendige Namespaces
              angegeben.-->
9      <Application.Resources />
10 </Application>
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Configuration;
4  using System.Data;
5  using System.Linq;
6  using System.Threading.Tasks;
7  using System.Windows;
8  using ESRI.ArcGIS.Framework;
9
10 namespace GpxWpfApplication {
11     /// Die Code-behind-Logik der App.xaml
12     public partial class App : Application {
13
14         private static App _instance;
15
16         protected App() {
17         }
18
19         //Singleton Design Pattern der App-Klasse
20         public static App Instance() {
21             return _instance ?? (_instance = new App());
22         }
23
24         //Property fuer das aktuelle Arcmap Dokument
25         public IDocument Doc { get; set; }
26
27         //Erstellung eines neuen ViewModels
28         public void App_Startup(object sender, StartupEventArgs e) {
29             var viewModel = new ViewModel();
30             //CreateView erstellt eine neue View
31             viewModel.CreateView();
32         }
33
34         //Der, ueber das 'Startup'-Element der App.xaml definierte,
             Programmeinstieg des WPF-Projekts (wird von OnClick des AddIns
             aufgerufen)
35         public void App_Startup(IDocument doc) {
36             this.Doc = doc;
37             //Rufe die ueberladene 'App_Startup' Methode auf
38             App_Startup(null, null);
39         }
40     }
41 }
42
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Text;
4  using System.IO;
5  using System.Windows;
6  using ESRI.ArcGIS.Framework;
7  using GpxWpfApplication;
8
9  namespace ArcMapAddinGpx {
10     public class ArcMapAddinGpx : ESRI.ArcGIS.Desktop.AddIns.Button {
11
12         public ArcMapAddinGpx() {
13         }
14
15         //Die OnClick-Methode wird ausgefuehrt sobald der OnClick-Event des
             AddIn-Buttons feuert.
16         protected override void OnClick() {
17             IDocument doc = ArcMap.Application.Document;
18             //Wenn noetig wird ein neues App Objekt initialisiert.
19             App app = null;
20             try {
21                 if (Application.Current == null) {
22                     app = App.Instance();
23                     app.ShutdownMode = ShutdownMode.OnExplicitShutdown;
24                 }
25                 else if (Application.Current.GetType() == typeof
                     (GpxWpfApplication.App)) {
26                     {
27                         app = Application.Current as GpxWpfApplication.App;
28                     }
29                 }
30                 else {
31                     return;
32                 }
33                 //Aufruf des WPF-Projekts
34                 if (app != null) app.App_Startup(doc);
35             }
36             catch (InvalidOperationException ex) {
37                 MessageBox.Show(ex.Message + ex.ToString());
38             }
39             catch (Exception ex) {
40                 MessageBox.Show(ex.Message + ex.ToString());
41             }
42             ArcMap.Application.CurrentTool = null;
43         }
44         protected override void OnUpdate() {
45             Enabled = ArcMap.Application != null;
46         }
47     }
48
49 }
50
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System;
using System.Diagnostics;
using System.Linq;
using System.Runtime.InteropServices;
using System.Windows;
using ESRI.ArcGIS.Geodatabase;

namespace GpxWpfApplication {
    //Da in .NET Konstruktoren in der Vererbungshierarchie von oben (Superklasse) nach unten (Subklassen), virtuelle Methoden werden jedoch vom untersten Objekt der Vererbungshierarchie aufgerufen.
    //Sobald versucht wird eine abgeleitete Methode im Konstruktor der Basisklasse aufzurufen, ist die Gefahr einer null reference exception immanent, da womoeglich im Basiskonstruktor auf Eigenschaften zugegriffen werden, die erst in der abgeleiteten Klasse initialisiert werden.
    //Die hier angewandte Loesung implementiert eine abstrakte Basisklasse, dessen virtuelle Methode nicht im Basiskonstrktor, stattdessen im Konstruktor der Kindklassen aufgerufen wird, sowie, falls noetig, in der entsprechenden Subklasse uebeschrieben werden kann
    abstract class BaseQuery {
        protected readonly int _column;
        protected readonly IQueryFilter _queryFilter;
        protected ICursor _cursor;
        protected IRow _row;
        protected readonly ITable _table;
        protected double _sum;

        //Konstruktor der Klasse
        public BaseQuery(ITable table, int column, string where) {
            if (!Enumerable.Range(0, table.Fields.FieldCount).Contains(column)) throw new ArgumentOutOfRangeException("column");
            _table = table;
            _column = column;
            _queryFilter = new QueryFilterClass {
                WhereClause = where
            };
        }

        //Die virtuelle Methode zum Initialisieren des Cursor-Objekts erlaubt das Ueberschreiben, um so eine spezialisierte Implementierung in den Subklassen zu erlauben
        protected virtual ICursor InitCursorObject(ITable table) {
            Debug.WriteLine(@"Debug-@BaseQuery: protected virtual ICursor InitCursorObject(ITable table)");
            return table.Search(_queryFilter, false);
        }

        //Destruktor der Klasse, um mittels Marshal 'unmanaged'-COM Objekte freizugeben
        ~BaseQuery() {
```

```
43              Debug.WriteLine(@"SearchQuery's destrucor fired for " +
                  base.ToString());
44              if (_cursor != null) {
45                  Marshal.ReleaseComObject(_cursor);
46              }
47          }
48
49          //InitializeWhereStringForDataGridView erlaubt das Setzen der
              generische Where-Klauel fuer die haeufig benoetigte DataGridView-
              UserControl
50          internal static string InitializeWhereStringForDataGridView(double?
              min = null, double? max = null) {
51              string where = string.Empty;
52              if (min != null && max != null) {
53                  if (!double.IsNaN((double)max)) {
54                      where = String.Format("\"steig_proz\" >= {0} AND
                        \"steig_proz\" < {1}", min, max);
55                  } else if (min < 0) {
56                      where = String.Format("\"steig_proz\" < {0}", min);
57                  } else if (min > 0) {
58                      where = String.Format("\"steig_proz\" >= {0}", min);
59                  }
60              }
61              return where;
62          }
63
64          //Gibt einen Integer-Wert zurueck, der den Index der Spalte mit dem
              Namen des Parameters 'requiredFieldName' darstellt.
65          //Erstellt eine passende Spalte mittels der Methode
              'CreateRequiredField', falls diese im Table noch nicht vorhanden
              ist.
66          internal int GetIndexOfEnsuredSumField(ITable tbl, string
              requiredFieldName) {
67              Debug.WriteLine(@"Debug-
                  @UpdateQuery.GetIndexOfEnsuredSumField.requiredFieldName: " +
                  requiredFieldName);
68              if (requiredFieldName == null) {
69                  return -1;
70              }
71              try {
72                  IFields2 fields = (IFields2)tbl.Fields;
73                  int fieldPos = -1;
74                  for (int i = 0; i < fields.FieldCount; i++) {
75                      if (fields.Field[i].Name == requiredFieldName) {
76                          fieldPos = i;
77                          Debug.WriteLine("resulting integer of fieldPos (for
                        loop): " + fieldPos);
78                          return fieldPos;
79                      }
80                  }
81                  int temp = CreateRequiredField(tbl, requiredFieldName);
82                  if (temp == -1) {
83                      throw new System.Exception("Could not create a field named
                         " + requiredFieldName);
84                  }
85                  Debug.WriteLine("resulting integer of CreateRequiredField: " +
```

```
                    temp);
86                  return temp;
87              } catch (Exception ex) {
88                  Debug.WriteLine(ex.Message + Environment.NewLine + ex.ToString
                      ());
89                  return -1;
90              }
91          }
92
93          //CreateRequiredField erstellt eine neue Spalte in der Shape-Datei
94          private int CreateRequiredField(ITable tbl, string requiredFieldName)
              {
95              Debug.WriteLine(@"Debug-
                  @UpdateQuery.CreateRequiredField.requiredFieldName: " +
                  requiredFieldName);
96              //Setzt die fuer die Spalte notwendigen Attribute mittels der in
                  der Klasse DataRepository hinterlegten Spalteneigenschaften,
                  sofern der Wert des Parameters requiredFieldName' auch in der
                  dahinterliegenden Collection gefunden wurde.
97              if (DataRepository.ContainsField(requiredFieldName)) {
98                  IFieldEdit2 field = new FieldClass() {
99                      IFieldEdit2_Name_2 = DataRepository.GetField
                        (requiredFieldName).Name,
100                     IFieldEdit2_Type_2 = DataRepository.GetField
                        (requiredFieldName).Type,
101                 };
102                 double defaultDouble;
103                 if (double.TryParse(DataRepository.GetField
                      (requiredFieldName).DefaultValue, out defaultDouble)) {
104                     field.DefaultValue_2 = defaultDouble;
105                 } else {
106                     field.DefaultValue_2 = DataRepository.GetField
                        (requiredFieldName).DefaultValue;
107                 }
108                 if (DataRepository.GetField
                      (requiredFieldName).Precision.HasValue) {
109                     field.Precision_2 = DataRepository.GetField
                        (requiredFieldName).Precision.Value;
110                 }
111                 if (DataRepository.GetField(requiredFieldName).Scale.HasValue)
                      {
112                     field.Scale_2 = DataRepository.GetField
                        (requiredFieldName).Scale.Value;
113                 }
114                 try {
115                     tbl.AddField(field);
116                     return tbl.Fields.FieldCount - 1;
117                 } catch (Exception ex) {
118                     MessageBox.Show(ex.Message + Environment.NewLine +
                        ex.ToString());
119                 }
120             }
121             return -1;
122         }
123     }
124 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Runtime.InteropServices;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace GpxWpfApplication {
9      //Die Klasse erlaubt es innerhalb der WPF eine Konsole aufzurufen, die im
         Projekt zum Debuggen dient.
10     //Das DllImport Attribut ermoeglicht den Aufruf von unmanaged Code
         innerhalb einer managed Applikation
11     //kernel32.dll liefert den Zugriff auf viele grundlegende Win32 APIs.
12     public class ConsoleHelper {
13         //Oeffnet eine neue Command Prompt (Windows Console), sofern nicht
             bereits offen.
14         [DllImport("kernel32.dll")]
15         public static extern Boolean AllocConsole();
16
17         //Trennt die Console wieder von der WPF Application:
18         [DllImport("kernel32.dll")]
19         public static extern Boolean FreeConsole();
20     }
21 }
22
```

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Diagnostics;
 4  using System.Globalization;
 5  using System.Text;
 6  using System.Windows;
 7  using ESRI.ArcGIS.Carto;
 8  using ESRI.ArcGIS.Geodatabase;
 9  using ESRI.ArcGIS.ArcMapUI;
10  using System.Runtime.InteropServices;
11  using System.Threading;
12  using ESRI.ArcGIS;
13  using ESRI.ArcGIS.Geoprocessing;
14  using ESRI.ArcGIS.esriSystem;
15  using System.Windows.Threading;
16  using System.Windows.Controls;
17  using ESRI.ArcGIS.Framework;
18
19  namespace GpxWpfApplication {
20      public class DataAccessLayer {
21
22          private readonly ViewModel _vm;
23
24          //Konstruktor der Klasse, der fuer das Feld _vm eine Referenz auf das
              ViewModel setzt
25          public DataAccessLayer(ViewModel viewModel) {
26              _vm = viewModel;
27          }
28
29          //Diese Methode berechnet mittels eines SearchQuery-Objekts die
              Geschwindigkeit in den Labestationen bezogen auf die Leistungsklasse
30          //Sie benoetigt das ViewModel als Parameter, da die Methode statisch
              ist und somit aufgerufen werden kann, bevor ein Objekt ueber den
              Konstruktor erstellt wurde
31          public static void CalculateAidStationVelocity(ViewModel vm) {
32              //Erstellung der Where-Klausel fuer das SearchQuery-Objekt.
33              StringBuilder aidStationQueryFilterAsString = new StringBuilder();
34              for (int i = 0; i < vm.ViewModelConfig.AidStationIDs.Length / 2; i
                  ++) {
35                  aidStationQueryFilterAsString.Append("(\"FID\" >= " +
                      vm.ViewModelConfig.AidStationIDs[i, 0] + " AND \"FID\" <= "
                      + vm.ViewModelConfig.AidStationIDs[i, 1] + ")");
36                  if (i != vm.ViewModelConfig.AidStationIDs.Length / 2 - 1) {
37                      aidStationQueryFilterAsString.Append(" OR ");
38                  }
39              }
40              try {
41                  SearchQuery sq = new SearchQuery(vm.ViewModelConfig.Table,
                      vm.ViewModelConfig.RadioValue,
                      aidStationQueryFilterAsString.ToString());
42                  //Aktualisierung der View im UI-Thread mittels Dispatcher
43                  Application.Current.Dispatcher.Invoke((Action)(() => {
44                      vm.View.RefreshLabel.Content = Math.Round(sq.CalculateMean
                          (), 2).ToString(CultureInfo.CurrentCulture);
45                  }));
46              } catch (Exception ex) {
```

```
47                  MessageBox.Show(ex.ToString() + Environment.NewLine +
                      ex.Message);
48              }
49          }
50
51          //ParseUserinputToDouble eine beliebige Zeichenkette entgegen und
              versucht diese als Gleitkommazahl zurueckzugeben
52          private double ParseUserinputToDouble(string input) {
53              //Da die Eingabe als Geschwindigkeitsangabe oder als Steigung in
                  Grad zu interpretieren ist, wird auf eine Verwendung der
                  CultureInfo-Klasse verzichtet und stattdessen praeventiv in ein
                  Punkt-Dezimaltrennzeichen konvertiert
54              input = input.Replace(".", ",");
55              double candidate;
56              //Der boole'sche Rueckgabewert von TryParse wird als
                  Abbruchbedingung des Programms verwendet
57              if (double.TryParse(input, out candidate)) {
58                  return candidate;
59              }
60              //Bei erfolglosem Parsen die Not A Number Konstante zurueckgeben
61              return double.NaN;
62          }
63
64          //ThrowExceptionOnEmptyString wirft eine Exception an den Aufrufer
              zurueck, sollte das Argument aus einer leeren Zeichenkette bestehen
65          private string ThrowExceptionOnEmptyString(string s) {
66              if (s == String.Empty) {
67                  throw new Exception("Ein Fehler ist aufgetreten. Bitte füllen
                      Sie alle Felder aus");
68              }
69              return s;
70          }
71
72          public void StartQuerying() {
73              //Stelle sicher, dass eine Leistungsklasse gewaehlt wurde.
74              if (_vm.ViewModelConfig.RadioValue == 0) {
75                  MessageBox.Show("Ein Fehler ist aufgetreten. " +
76                                  "Bitte überprüfen Sie, ob eine Leistungsklasse
                          gewählt wurde", "Warnung");
77                  return;
78              }
79
80              Debug.WriteLine(@"Debug-@DataAccessLayer Enum value: " +
                  _vm.VMSelectedTabAsEnum);
81              //TODO replace any tab selection checks by this method: if
                  (_vm.VMSelectedTabAsEnum == ViewModel.TabEnum.Simulation) {
82              if (_vm.VMSelectedTabAsEnum == ViewModel.TabEnum.Simulation) {
83                  //slManager dient als Collection fuer die Benutzeranpassungen
                      im Reiter 'Simulation'.
84                  List<SimulationLine> slManager = new List<SimulationLine>();
85                  try {
86                      //Der Zugriff auf UI Elemente muss mittels Dispatcher des
                        UI-Threads erfolgen
87                      System.Windows.Application.Current.Dispatcher.Invoke(
88                          //Diese Lambda-Operation erstellt fuer jedes
                        WrapPanel-Objekt ein neues SimulationLine-Objekt
```

```
89                      //und fuegt es der Collection hinzu.
90                      (Action)(() => {
91                          int counter = 0;
92                          foreach (object wp in _vm.View.StackPanel1.Children) {
93                              counter++;
94                              if (wp is WrapPanel && counter <= SimulationLineGui.Counter) {
95                                  UIElementCollection uiec = (wp as WrapPanel).Children;
96                                  slManager.Add(new SimulationLine(
97                                      ThrowExceptionOnEmptyString(((ComboBox)uiec[2]).Text.ToString()),
98                                      ((ComboBox)uiec[5]).Text, //Darf keine Exception werfen, um Queries wie "steig_proz <= -11" zu erlauben
99                                      ParseUserinputToDouble(((TextBox)uiec[3]).Text),
100                                     ParseUserinputToDouble(((TextBox)uiec[6]).Text),
101                                     ParseUserinputToDouble(((TextBox)uiec[8]).Text))
102                                 );
103                             }
104                         }
105                     })
106                 );
107                 double min, max;
108                 string simLineQueryFilterAsString = string.Empty;
109                 //Iteration der Elemente in der Collection 'slManager' mittels Lambda Operation
110                 slManager.ForEach(simLine => {
111                     min = simLine.Slope1;
112                     max = simLine.Slope2;
113                     //Erstelle fuer jedes Element in der Collection 'slManager' eine passende Where-Klausel
114                     if (!double.IsNaN(max)) {
115                         simLineQueryFilterAsString = String.Format(
116                             "\"steig_proz\" {2} {0} AND \"steig_proz\" {3} {1}", min, max, simLine.Delimiter1,
117                             simLine.Delimiter2);
118                     } else if (min < 0) {
119                         Debug.Assert(double.IsNaN(max));
120                         simLineQueryFilterAsString = String.Format("\"steig_proz\" < {0}", min);
121                     } else if (min >= 0) {
122                         Debug.Assert(double.IsNaN(max));
123                         simLineQueryFilterAsString = String.Format("\"steig_proz\" >= {0}", min);
124                     }
125                     //Erstelle ein SearchQuery und UpdateQuery-Objekt.
126                     SearchQuery sq = new SearchQuery(_vm.ViewModelConfig.Table, _vm.ViewModelConfig.RadioValue, simLineQueryFilterAsString);
127                     UpdateQuery uq = new UpdateQuery
```

```
                    (_vm.ViewModelConfig.Table,
                    _vm.ViewModelConfig.RadioValue,
                    simLineQueryFilterAsString);
128                     //Berechne die Differenz der Benutzereingabe und des
                    tatsaechlich gespeicherten Wertes in der Datenbank in
                    Prozent.
129                     //Ermittle die neuen Werte mittels der Prozentangabe
                    und schreibe diese in die Datenbank.
130                     uq.UpdateDatasetByPercent(sq.CalculatePercentage
                    (simLine.Velocity));
131                 });
132
133                 #region Pacing Strategy and/or Labestation - Start
134                 bool pacingFlag = false;
135                 double pacingDistance = double.NaN;
136                 double pacingVelocity = double.NaN;
137                 double refreshNew = double.NaN;
138                 //Greife mittels Dispatcher auf den UI-Thread zu, um
                    Angaben bezueglich Pacing Strategie und Labestationen
                    auszulesen.
139                 System.Windows.Application.Current.Dispatcher.Invoke
                    ((Action)(() => {
140                     if (_vm.View.PacingSlow.IsChecked != null && (bool)
                    _vm.View.PacingSlow.IsChecked) {
141                         pacingFlag = true;
142                         pacingDistance = ParseUserinputToDouble
                    (_vm.View.PacingDistance.Text);
143                         pacingVelocity = Double.Parse
                    (_vm.View.PacingVelocity.Text);
144                     }
145                     refreshNew =
146                         (ParseUserinputToDouble(String.IsNullOrEmpty
                    (_vm.View.RefreshTbx.Text)
147                             ? "0.0"
148                             : _vm.View.RefreshTbx.Text));
149                 }));
150
151                 if (pacingFlag) {
152                     //Starte die Berechnung der neuen Geschwindigkeit wenn
                     eine Slow Starting Strategie gewaehlt wurde.
153                     string pacingQueryFilterAsString = String.Format
                    ("\"FID\" < {0}", pacingDistance * 100);
154                     SearchQuery sq = new SearchQuery
                    (_vm.ViewModelConfig.Table,
                    _vm.ViewModelConfig.RadioValue,
                    pacingQueryFilterAsString);
155                     UpdateQuery uq = new UpdateQuery
                    (_vm.ViewModelConfig.Table,
                    _vm.ViewModelConfig.RadioValue,
                    pacingQueryFilterAsString);
156                     uq.UpdateDatasetByPercent(sq.CalculatePercentage
                    (pacingVelocity));
157                 }
158                 //Starte die Berechnung der neuen Geschwindigkeit
                    innerhalb der Labestationen, vorausgesetzt in der TextBox
                    'RefreshTbx' wurde ein valider Wert eingegeben.
```

```
159                     if (refreshNew != 0) {
160                         StringBuilder aidStationQueryFilterAsStringBuilder =
                        new StringBuilder();
161                         for (int i = 0; i <
                        _vm.ViewModelConfig.AidStationIDs.Length / 2; i++) {
162                             aidStationQueryFilterAsStringBuilder.Append
                        ("(\"FID\" >= " + _vm.ViewModelConfig.AidStationIDs[i, 0]
                        +
163                                                                 " AND \"FID
                        \" <= " + _vm.ViewModelConfig.AidStationIDs[i, 1] + ")");
164                             if (i !=
                        _vm.ViewModelConfig.AidStationIDs.Length / 2 - 1) {
165                                 aidStationQueryFilterAsStringBuilder.Append("
                        OR ");
166                             }
167                         }
168                         string aidStationQueryFilterAsString =
                        aidStationQueryFilterAsStringBuilder.ToString();
169                         SearchQuery sq = new SearchQuery
                        (_vm.ViewModelConfig.Table,
                        _vm.ViewModelConfig.RadioValue,
                        aidStationQueryFilterAsString);
170                         UpdateQuery uq = new UpdateQuery
                        (_vm.ViewModelConfig.Table,
                        _vm.ViewModelConfig.RadioValue,
                        aidStationQueryFilterAsString);
171                         uq.UpdateDatasetByPercent(sq.CalculatePercentage
                        (refreshNew));
172                         sq = new SearchQuery(_vm.ViewModelConfig.Table,
                        _vm.ViewModelConfig.RadioValue,
                        aidStationQueryFilterAsString);
173                         //Aktualisiere die View
174                         System.Windows.Application.Current.Dispatcher.Invoke
                        ((Action)(() => {
175                             _vm.View.RefreshLabel.Content =
176                                 (Math.Round(sq.CalculateMean(), 2)).ToString
                        (CultureInfo.CurrentCulture);
177                         }));
178
179                     }
180                     #endregion Pacing Strategy and/or Labestation
181                 } catch (NullReferenceException ex) {
182                     MessageBox.Show(ex.ToString() + Environment.NewLine +
                        Environment.NewLine + ex.Message);
183                 } catch (Exception ex) {
184                     MessageBox.Show(ex.ToString() + Environment.NewLine +
                        Environment.NewLine + ex.Message);
185                 } finally {
186                     this._vm.Dispatcher.Invoke(new Action(() =>
                        this._vm.ClearSimulationLineGui()),
                        DispatcherPriority.Input);
187                     slManager.Clear();
188                     SimulationLineGui.Counter = 0;
189                 }
190             }
191             //Flag fuer Debugging-Zwecke
```

```
192             if (true) {
193                 ITable tbl = _vm.ViewModelConfig.Table;
194                 //Berechne die Summe der Spalte, in der die benoetigte Zeit
                      der Segmente gespeichert ist - wobei die aktuelle Summe fuer
                       jede Zeile akkumulierend zur vorhergehenden Zeile
                      beschrieben wird
195                 SearchQuery sq = new SearchQuery(_vm.ViewModelConfig.Table,
                      _vm.ViewModelConfig.RadioValue, null);
196                 UpdateQuery uq = new UpdateQuery(_vm.ViewModelConfig.Table,
                      _vm.ViewModelConfig.RadioValue, null);
197                 try {
198                     int sumColNum = sq.GetIndexOfEnsuredSumField
                      (_vm.ViewModelConfig.Table, "Sum1");
199                     int sumRowNum = uq.CalculateSumAndGetRowCounter
                      (sumColNum);
200                     int tagColNum = sq.GetIndexOfEnsuredSumField
                      (_vm.ViewModelConfig.Table, "Tagged1");
201                     UpdateQuery tagUq = new UpdateQuery
                      (_vm.ViewModelConfig.Table, tagColNum, null);
202                     string timeTotal = tagUq.FillTagColumn(sumRowNum);
203                     System.Windows.Application.Current.Dispatcher.Invoke
                      ((Action)(() => {
204                         this._vm.ViewModelConfig.LapTimeX2 = timeTotal;
205                     }));
206                 } catch (Exception ex) {
207                     MessageBox.Show(ex.Message + ex.ToString());
208                 }
209
210                 //Aktualisierung des ToCs
211                 ITableWindow tableWindow = new TableWindowClass();
212                 tableWindow = tableWindow.FindViaTable(tbl, false);
213                 if (tableWindow != null && tableWindow.IsVisible) {
214                     tableWindow.Refresh();
215                 }
216
217                 try {
218                     //Versuche das Tool zu starten, das die GPX Datei
                      erstellt.
219                     bool wasSuccessful = RunSpatialETL();
220                     //Benachrichtige die View, ob die Ausfuehrung erfolgreich
                      war
221                     System.Windows.Application.Current.Dispatcher.Invoke
                      ((Action)(() => {
222                         _vm.ViewModelConfig.WasSuccessful = wasSuccessful;
223                     }));
224                 } catch (COMException ex) {
225                     MessageBox.Show("COMException error: " +
                      ex.ErrorCode.ToString() + ": " + ex.Message);
226                 } catch (Exception ex) {
227                     MessageBox.Show("Exception error: " + ex.Message);
228                 } finally {
229                     RefreshMxDoc(_vm);
230                 }
231             }
232         }
233         //ArcMap aktualisieren
```

```
234         private void RefreshMxDoc(ViewModel vm) {
235             vm.ViewModelConfig.MxDoc.ActiveView.ContentsChanged();
236             vm.ViewModelConfig.MxDoc.UpdateContents();
237             vm.ViewModelConfig.MxDoc.ActiveView.Refresh();
238         }
239 
240         //TODO add a brief description of SpatialETLTool
241         private bool RunSpatialETL() {
242             //Suche mittels des RuntimeManagers nach einem passenden ArcgGis
                Desktop Ordner
243             IEnumerable<RuntimeInfo> info = RuntimeManager.InstalledRuntimes;
244             string version = string.Empty;
245             foreach (RuntimeInfo item in info) {
246                 if (item.Path.Contains("Desktop10")) {
247                     string[] folders = item.Path.Split('\\');
248                     foreach (string folder in folders) {
249                         if (folder.Contains("Desktop10")) {
250                             version = folder;
251                             break;
252                         }
253                     }
254                     break;
255                 }
256             }
257             //Wenn es sich um eine Versionnummer handelt, die aelter als
                Version 10 ist, breche aus der Funktion aus.
258             bool hasProduct = RuntimeManager.Bind
                (ProductCode.EngineOrDesktop);
259             if (hasProduct) {
260                 var activeRunTimeInfo = RuntimeManager.ActiveRuntime;
261                 if (Double.Parse(activeRunTimeInfo.Version) < 10) {
262                     return false;
263                 }
264             } else {
265                 return false;
266             }
267             //Erstelle ein GeoProcessor-Objekt, fuege die notwendige Toolbox
                hinzu und fuehre das Tool aus.
268             IGeoProcessor2 gp = new GeoProcessorClass();
269             string path = Environment.GetFolderPath
                (Environment.SpecialFolder.ApplicationData) +
270                           "\\ESRI\\" + version + "\\ArcToolbox\\My Toolboxes\
                    \";
271 
272             IVariantArray paras = new VarArrayClass();
273             gp.AddToolbox(path + "toolbox_data_interoperability.tbx");
274             //Erstelle ein GeoProcessEvent-Objekt und registriere dieses beim
                GeoProcessor-Objekt.
275             gp.RegisterGeoProcessorEvents3(new GeoProcessEvent());
276             //Beende das Programm falls das notwendige Werkzeug nicht gefunden
                 werden kann.
277             IGpEnumList list = gp.ListTools(null);
278             Debug.WriteLine("Tools found: ");
279             string toolInfo = list.Next();
280             bool hasToolFound = false;
281             while (!String.IsNullOrEmpty(toolInfo)) {
```

```
282                 Debug.WriteLine(toolInfo);
283                 if (toolInfo == "SpatialETLTool") hasToolFound = true;
284                 toolInfo = list.Next();
285             }
286             if (!hasToolFound) {
287                 Debug.WriteLine("Error finding ETL tool in" +
                      Environment.NewLine + path);
288                 _vm.CloseOnPanic("Das notwendige Werkzeug im angegebenen
                      Verzeichnis konnte nicht gefunden werden");
289             }
290             //Iteriere durch die GPMessages des Result-Objekts und gib den
                  Wert 'false' zurueck, sobald es sich bei einer Nachricht um
                  einen Fehler handelt.
291             IGeoProcessorResult2 result = gp.Execute("SpatialETLTool", paras,
                  null) as IGeoProcessorResult2;
292             if (result != null) {
293                 IGPMessages msgs = result.GetResultMessages();
294                 for (int i = 0; i < msgs.Count; i++) {
295 #if DEBUG
296                     Debug.WriteLine(Environment.NewLine);
297                     Debug.WriteLine(msgs.GetMessage(i).Description +
                        Environment.NewLine + msgs.GetMessage(i).Type);
298 #endif //DEBUG
299                     if (msgs.GetMessage(i).IsError())
300                         return false;
301                 }
302                 return true;
303             }
304             return false;
305         }
306
307     }
308 }
309
```

```
<UserControl x:Class="GpxWpfApplication.DataGridView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:GpxWpfApplication"
             mc:Ignorable="d"
             d:DesignHeight="300" d:DesignWidth="452"
             >
    <UserControl.Resources>
        <!-- Der Converter erlaubt es mittels IValueConverter die Daten in der GUI beliebig anzupassen (siehe SlopeToStringConverter.cs)-->
        <local:SlopeToStringConverter x:Key="SlopeConverter" />
    </UserControl.Resources>
    <StackPanel>
        <Button x:Name="GenerateGpx" Content="Generiere GPX-Datei" IsDefault="True" Command="{Binding ButtonGpx}" Margin="10"
        HorizontalAlignment="Stretch" />
        <ProgressBar x:Name="Pbar" DockPanel.Dock="Top" Height="16" Margin="4" IsIndeterminate="True" Visibility="Collapsed"></ProgressBar>
        <ListView Name="LvUsers" ScrollViewer.HorizontalScrollBarVisibility="Hidden" DockPanel.Dock="Top" HorizontalAlignment="Stretch">
          <ListView.View>
              <GridView>
                  <GridViewColumn Width="151" DisplayMemberBinding="{Binding Kategorie, Mode=TwoWay}">
                      <GridViewColumn.Header>
                          <GridViewColumnHeader Tag="Kategorie" Click="lvUsersColumnHeader_Click">Steigungsklasse</GridViewColumnHeader>
                      </GridViewColumn.Header>
                  </GridViewColumn>
                  <GridViewColumn Width="151" DisplayMemberBinding="{Binding Slope, Mode=OneWay, Converter={StaticResource ResourceKey=SlopeConverter}}">
                      <GridViewColumn.Header>
                          <GridViewColumnHeader Tag="SlopeSortHelper" Click="lvUsersColumnHeader_Click">Wertebereich (in %)</GridViewColumnHeader>
                      </GridViewColumn.Header>
                  </GridViewColumn>
                  <GridViewColumn Width="150" DisplayMemberBinding="{Binding Velocity, Mode=TwoWay}">
                      <GridViewColumn.Header>
                          <GridViewColumnHeader Tag="Velocity" Click="lvUsersColumnHeader_Click">km/h</GridViewColumnHeader>
                      </GridViewColumn.Header>
                  </GridViewColumn>
              </GridView>
          </ListView.View>
        </ListView>
    </StackPanel>
</UserControl>
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Windows;
4  using System.Windows.Controls;
5  using System.Windows.Documents;
6  using System.Windows.Input;
7  using System.ComponentModel;
8
9  namespace GpxWpfApplication {
10     // Die Code-behind-Logik fuer DataGridView.xaml
11     public partial class DataGridView : UserControl {
12         public DataGridView() {
13             InitializeComponent();
14         }
15
16         private GridViewColumnHeader _listViewSortCol = null;
17         private SortAdorner _listViewSortAdorner = null;
18
19         //Dieses Event sortiert die Properties der LapTime-Klasse, welche an
             die GridView gebunden wurden, mittels korrespondierenden Tags.
20         private void lvUsersColumnHeader_Click(object sender, RoutedEventArgs
             e) {
21             try {
22                 GridViewColumnHeader column = (sender as GridViewColumnHeader);
23                 if (column != null) {
24                     string sortBy = column.Tag.ToString();
25                     if (_listViewSortCol != null) {
26                         AdornerLayer.GetAdornerLayer(_listViewSortCol).Remove
                       (_listViewSortAdorner);
27                         LvUsers.Items.SortDescriptions.Clear();
28                     }
29
30                     ListSortDirection newDir = ListSortDirection.Ascending;
31                     if (Equals(_listViewSortCol, column) &&
                       _listViewSortAdorner.Direction == newDir)
32                         newDir = ListSortDirection.Descending;
33
34                     _listViewSortCol = column;
35                     //SortAdorner zeichnet Icons in den GridViewColumnHeader,
                       die Auskunft ueber die Sortierrichtung geben.
36                     _listViewSortAdorner = new SortAdorner(_listViewSortCol,
                       newDir);
37                     AdornerLayer.GetAdornerLayer(_listViewSortCol).Add
                       (_listViewSortAdorner);
38                     LvUsers.Items.SortDescriptions.Add(new SortDescription
                       (sortBy, newDir));
39                 }
40             } catch (Exception ex) {
41                 MessageBox.Show(ex.ToString());
42             }
43         }
44     }
45 }
46
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ESRI.ArcGIS.Geodatabase;

namespace GpxWpfApplication {
    //Diese statische Klasse dient als Template-Manager der zu erstellenden Spalten.
    public static class DataRepository {

        public static List<DefinedField> DefinedFields;

        static DataRepository() {
            //Die generische DefinedFields-List bietet die Moeglichkeit die Eigenschaften der zu erstellenden Spalten mittels DefinedField-Objekten zentral zu verwalten.
            DefinedFields = new List<DefinedField>();
            DefinedFields.Add(new DefinedField("Sum1", esriFieldType.esriFieldTypeDouble, "0", scale: 15, precision: 18));
            DefinedFields.Add(new DefinedField("Tagged1", esriFieldType.esriFieldTypeString, String.Empty));
        }

        //Die GetField Methode gibt das entsprechende Objekt zurueck.
        public static DefinedField GetField(string fieldName) {
            foreach (DefinedField tmp in DefinedFields) {
                if (tmp.Name == fieldName) {
                    return tmp;
                }
            }
            return null;
        }

        //ContainsField erlaubt es in der Collection nach einer Spalte zu suchen und zeigt an, ob ein passendes Template gefunden wurde.
        internal static bool ContainsField(string fieldName) {
            foreach (DefinedField tmp in DefinedFields) {
                if (tmp.Name == fieldName) {
                    return true;
                }
            }
            return false;
        }
    }

    public class DefinedField {

        string _defaultValue;
        string _name;
        int? _precision;
        int? _scale;
        esriFieldType _type;
```

```
50
51          //Konstruktor der DefinedField-Klasse, der im statischen Konstruktor
              der DataRepository-Klasse beim Hinzufuegen neuer Objekte in der
              DefinedFields Collection aufgerufen wird.
52          public DefinedField(string name, esriFieldType type, string
              defaultValue, int? scale = null, int? precision = null) {
53              this.Name = name;
54              this.Type = type;
55              this.DefaultValue = defaultValue;
56              this.Scale = scale;
57              this.Precision = precision;
58          }
59
60          //Properties fuer die privaten Felder, die im Konstruktor gesetzt
              wurden.
61          public string Name
62          {
63              get { return _name; }
64              set { _name = value; }
65          }
66
67          public esriFieldType Type
68          {
69              get { return _type; }
70              set { _type = value; }
71          }
72
73          public string DefaultValue
74          {
75              get { return _defaultValue; }
76              set { _defaultValue = value; }
77          }
78
79          public int? Scale
80          {
81              get { return _scale; }
82              set { _scale = value; }
83          }
84
85          public int? Precision
86          {
87              get { return _precision; }
88              set { _precision = value; }
89          }
90      }
91  }
92
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Input;
using System.Windows;

namespace GpxWpfApplication {
    //Die Klasse erlaubt mittels ICommand-Interface und dessen Implementierung durch die Methoden Execute und CanExecute die Umsetzung des Command-Entwurfmusters.
    public class GenericButton : ICommand {

        public event EventHandler CanExecuteChanged;
        private readonly Action<object> _execute;
        private readonly Func<object, bool> _canExecute;

        //Wird im Konstruktor nur das Action-Delegate gesetzt, wird zusaetzlich der überladenen Konstruktor mit der Signaturauf "Action, Func<bool>" auf, wobei GenericCanExecute() als Func<bool>-Argument dient.
        public GenericButton(Action execute)
            : this(execute, GenericCanExecute) { }

        public GenericButton(Action execute, Func<bool> canExecute) {
            if (execute == null)
                throw new Exception("The Action-delegate was not specifiec when creating the button");

            if (canExecute == null) {
                throw new Exception("The Func-delegate was specified when creating generic button to evaluate canExecute state");
            }
            //Setze mittels Lambda-Operation die Delegates auf die dafuer vorgesehenen Methoden an.
            _execute = _ => execute();
            _canExecute = _ => canExecute();
        }

        public bool CanExecute(object parameter) {
            return _canExecute(parameter);
        }


        public object CommandParamObj;
        public void Execute(object parameter) {
            CommandParamObj = parameter;
            if (parameter == null || (parameter is string && parameter.ToString().Trim().Length == 0)) {
                CommandParamObj = this;
            }
            _execute(parameter);
            CommandParamObj = null;
        }

        private static bool GenericCanExecute() {
```

```
49              return true;
50          }
51
52          public void RaiseCanExecuteChanged() {
53              if (CanExecuteChanged != null)
54                  CanExecuteChanged(this, new EventArgs());
55          }
56      }
57  }
58
```

```
1  using ESRI.ArcGIS.Geoprocessing;
2  using System.Diagnostics;
3  using ESRI.ArcGIS.esriSystem;
4  using ESRI.ArcGIS.Geodatabase;
5
6  namespace GpxWpfApplication {
7
8      //Diese Klass erlaubt mittels Implementierung der beiden Interfaces eine
         ausserordentliche Hilfestellung bei der Suche nach Fehlern und bei der
         Informationsbeschaffung bzgl. des GeoProcessors.
9      class GeoProcessEvent : IGeoProcessorEvents3, IGeoProcessorEvents {
10
11         public void OnProcessMessages(IGeoProcessorResult gpResult, IGPMessages
              gpMessages) {
12             Debug.WriteLine("GeoProcessEvent.OnProcessMessages {0}",
                 gpResult.Status);
13         }
14         public void OnProgressMessage(IGeoProcessorResult gpResult, string
             message) {
15             Debug.WriteLine("GeoProcessEvent.OnProgressMessages {0}",
                 gpResult.Status);
16         }
17         public void OnProgressPercentage(IGeoProcessorResult gpResult, double
             percentage) {
18             Debug.WriteLine("GeoProcessEvent.OnProgressPercentage {0}",
                 gpResult.Status);
19         }
20         public void OnProgressShow(IGeoProcessorResult gpResult, bool show) {
21             Debug.WriteLine("GeoProcessEvent.OnProgressShow {0}",
                 gpResult.Status);
22         }
23         public void PostToolExecute(IGeoProcessorResult gpResult) {
24             Debug.WriteLine("GeoProcessEvent.PostToolExecute {0}",
                 gpResult.Status);
25         }
26         public void PreToolExecute(IGeoProcessorResult gpResult) {
27             Debug.WriteLine("GeoProcessEvent.PreToolExecute {0}",
                 gpResult.Status);
28         }
29
30         void IGeoProcessorEvents.OnMessageAdded(IGPMessage gpMessage) {
31             Debug.WriteLine("GeoProcessEvent.OnMessageAdded {0} {1}",
                 gpMessage.Description, gpMessage.Type);
32         }
33
34         void IGeoProcessorEvents.PostToolExecute(IGPTool tool, IArray values,
             int result, IGPMessages gpMessages) {
35             Debug.WriteLine("GeoProcessEvent.PostToolExecute {0}", tool.Name);
36         }
37
38         void IGeoProcessorEvents.PreToolExecute(IGPTool tool, IArray values,
             int processId) {
39             Debug.WriteLine(tool.IsLicensed()
40                 ? "GeoProcessEvent.PreToolExecute-Event fired"
41                 : "Warning\nThis tool is not licensed to run in the current
                   setup");
```

```
42          }
43
44          void IGeoProcessorEvents.ToolboxChange() {
45              Debug.WriteLine("GeoProcessEvent.ToolboxChange-Event fired");
46          }
47
48      }
49  }
50
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Diagnostics;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace GpxWpfApplication {
9      //Ein Objekt der LapTime-Klasse repraesentiert eine Zeile der DataGridView
         und erlaubt mittels der IComparable-Implementierung einen Vergleich
         untereinander.
10     public class LapTime : IComparable {
11
12         //Properties der Klasse
13         public string Kategorie { get; set; }
14         public double[] Slope { get; set; }
15         public double Velocity { get; set; }
16
17         public string ToStringWrapper
18         {
19             get { return ToString(); }
20             set { if (value == null) throw new ArgumentNullException
                   ("value"); }
21         }
22
23         public double SlopeSort { get { return Slope[1]; } }
24
25         public LapTime SlopeSortHelper
26         {
27             get { return this; }
28             set { if (value == null) throw new ArgumentNullException
                   ("value"); }
29         }
30
31         //Die ToString Methode wird ueberschrieben und liefert einen String,
             der fuer den Anwender leserlich ist.
32         public override string ToString() {
33             return string.Join(" bis ", Slope);
34         }
35
36         //implementierung von IComparable.CompareTo
37         public int CompareTo(object other) {
38             LapTime lt = other as LapTime;
39             if (lt == null) {
40                 return -1;
41             }
42             Debug.Assert(Slope != null, "Slope != null");
43             if (Slope[0] != lt.Slope[0]) {
44                 return ((int)(Slope[0] - lt.Slope[0]));
45             } else {
46                 return ((int)(Slope[1] - lt.Slope[1]));
47             }
48         }
49     }
50
51 }
```

```
<Window x:Class="GpxWpfApplication.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:local="clr-namespace:GpxWpfApplication"
        Title="MainWindow" Height="236" Width="452"
        ResizeMode="NoResize"
        mc:Ignorable="d"
        d:DesignHeight="350" d:DesignWidth="300"
        >
    <StackPanel>
        <TabControl Name="TabControl"  SelectedIndex="{Binding
          VMSelectedTabIndex, Mode=TwoWay}" IsEnabled="{Binding IsGuiEnabled,
          UpdateSourceTrigger=PropertyChanged, Mode=TwoWay}"
                    d:DataContext="{d:DesignInstance local:ViewModel}">
            <TabItem Name="TabMod" Header="Modellierung">
                <StackPanel>
                    <local:RadioSelection />
                    <local:DataGridView />
                </StackPanel>
            </TabItem>
            <TabItem Name="TabSim" Header="Simulation">
                <StackPanel>
                    <StackPanel.Resources>
                        <BooleanToVisibilityConverter
                    x:Key="BoolToVisibility" />
                    </StackPanel.Resources>
                    <local:RadioSelection />
                    <DockPanel Name="MyStackPanel" LastChildFill="False"
                    Margin="10,0,10,0" Height="110">
                        <WrapPanel Orientation="Vertical"
                    DockPanel.Dock="left">
                            <Label Content="Pacing Strategy" FontStyle="Italic"
                     FontWeight="Bold"/>
                            <RadioButton Name="PacingEven" GroupName="pacing"
                    IsChecked="True">Even Pacing</RadioButton>
                            <RadioButton Name="PacingSlow"
                    GroupName="pacing">Slow Starting</RadioButton>

                            <StackPanel Visibility="{Binding IsChecked,
                    ElementName=PacingSlow, Converter={StaticResource
                    BoolToVisibility}}" Margin="4,0,0,22"
                    ToolTipService.ShowOnDisabled="False"
                    Orientation="Vertical" HorizontalAlignment="Left">
                                <StackPanel.ToolTip>
                                    <StackPanel>
                                        <TextBlock FontWeight="Bold">Slow
                    Starting</TextBlock>
                                    </StackPanel>
                                </StackPanel.ToolTip>
                                <WrapPanel>
                                    <TextBox Name="PacingVelocity"
                    TextChanged="TextBox_TextChanged" Width="32" FontSize="10"
                    HorizontalContentAlignment="Right" Text="0,0" MaxLength="5"
                     HorizontalAlignment="Left"/>
```

```
41                          <Label Content="km/h
                  Geschwindigkeitsverminderung" Margin="0,-4,0,-4"
                  FontSize="10"/>
42                      </WrapPanel>
43                      <WrapPanel>
44                          <Label Content="über eine Distanz von"
                  FontSize="10" Margin="0,-4,0,-4" />
45                          <TextBox Name="PacingDistance"
                  TextChanged="TextBox_TextChanged" Width="32" FontSize="10"
                  HorizontalContentAlignment="Right" Text="0,0" MaxLength="5"
                   HorizontalAlignment="Left" Margin="0,0,0,0"/>
46                          <Label Content="km" Margin="0,-4,0,-4"
                  FontSize="10"/>
47                      </WrapPanel>
48                  </StackPanel>
49              </WrapPanel>
50              <WrapPanel DockPanel.Dock="Right">
51                  <Grid>
52                      <Grid.RowDefinitions>
53                          <RowDefinition Height="Auto" />
54                          <RowDefinition Height="Auto" />
55                          <RowDefinition Height="Auto" />
56                      </Grid.RowDefinitions>
57                      <Grid.ColumnDefinitions>
58                          <ColumnDefinition Width="Auto"/>
59                          <ColumnDefinition Width="Auto"/>
60                          <ColumnDefinition Width="Auto"/>
61                      </Grid.ColumnDefinitions>
62                      <Label Grid.Row="0" Grid.Column="0"
                  Grid.ColumnSpan="2" Content="Labestationen (Angaben in km/
                  h)" FontStyle="Italic" FontWeight="Bold"/>
63                      <Label Grid.Row="1" Grid.Column="0"
                  Content="Aktueller Durschschnitt:"/>
64                      <Label Name="RefreshLabel"  Grid.Row="1"
                  Grid.Column="1" Content="00,00"
                  HorizontalContentAlignment="Center" Width="50"/>
65                      <Label Grid.Row="2" Grid.Column="0"
                  Content="Neuer Durchschnitt:"/>
66                      <TextBox Name="RefreshTbx" Grid.Row="2"
                  Grid.Column="1" TextChanged="TextBox_TextChanged"
                  HorizontalContentAlignment="Center" Text="00,00"
                  MaxLength="5" Margin="0,0,0,0"
                  VerticalContentAlignment="Center"
                  LostFocus="refreshTbx_LostFocus"/>
67                  </Grid>
68              </WrapPanel>
69          </DockPanel>
70          <local:DataGridView />
71          <Label FontWeight="Bold" FontStyle="Italic"
                  Margin="0,15,0,0">
72              Individuelle Leistungsparameter
73          </Label>
74          <ScrollViewer Name="SimulationLineScrollViewer"
                  VerticalAlignment="Top"
                  HorizontalScrollBarVisibility="Disabled"
                  VerticalScrollBarVisibility="Auto" DockPanel.Dock="Top"
```

```
                    Margin="0,5,0,0" Height="160">
75                      <StackPanel  Name="StackPanel1" Margin="0"
                    DockPanel.Dock="Top">
76                      </StackPanel>
77                  </ScrollViewer>
78              </StackPanel>
79          </TabItem>
80      </TabControl>
81  </StackPanel>
82 </Window>
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Linq;
5  using System.Runtime.CompilerServices;
6  using System.Text;
7  using System.Text.RegularExpressions;
8  using System.Threading.Tasks;
9  using System.Windows;
10 using System.Windows.Controls;
11 using System.Windows.Data;
12 using System.Windows.Documents;
13 using System.Windows.Input;
14 using System.Windows.Media;
15 using System.Windows.Media.Imaging;
16 using System.Windows.Navigation;
17 using System.Windows.Shapes;
18
19 namespace GpxWpfApplication {
20     //Code-behind-Logik fuer MainWindow.xaml
21     public partial class MainWindow {
22         public MainWindow() {
23             InitializeComponent();
24         }
25
26         //Ueberladener Konstruktor, die den DataContext auf das ViewModel
           setzt, und so eine der Grundlagen fuer das MVVM-Patterns liefert.
27         public MainWindow(ViewModel viewModel) {
28             this.DataContext = viewModel;
29             InitializeComponent();
30         }
31
32         //Das Ueberschreiben der OnClosing-Methode erlaubt eine angepasste
           Beendigung der Applikation.
33         protected override void OnClosing(CancelEventArgs e) {
34             ViewModel viewModel = this.DataContext as ViewModel;
35             if (viewModel != null && (!
                 viewModel.ViewModelConfig.HasCloseOnPanicRequests &&
                 viewModel.IsGuiEnabled)) {
36                 MessageBoxResult res = MessageBox.Show("Do you really want to
                     close?", "Exit", MessageBoxButton.YesNo);
37                 if (res == MessageBoxResult.No) {
38                     e.Cancel = true;
39                 }
40             }
41             base.OnClosing(e);
42         }
43
44         //Dieser Event erlaubt es die Texteingabe des Anwenders abzufangen und
           entsprechend anzupassen. Der Vorteil ist eine erheblich bessere
           Lesbarkeit als eine Umsetzung mittels regulaerer Ausdruecke.
45         private void TextBox_TextChanged(object sender, TextChangedEventArgs e)
             {
46             TextBox textBox = sender as TextBox;
47             StringBuilder cleanText = new StringBuilder();
48             int counter = 0;
```

```
49              if (textBox != null) {
50                  //Erlaubt jeweils nur ein Dezimaltrennzeichen und numerische
                      Eingaben
51                  foreach (char c in textBox.Text.ToCharArray()) {
52                      if (Char.IsDigit(c) || ((c == '.' || c == ',') && counter
                      == 0)) {
53                          if (c == '.' || c == ',') {
54                              counter++;
55                          }
56                          cleanText.Append(c);
57                      }
58                  }
59                  textBox.Text = cleanText.ToString();
60                  textBox.SelectionStart = textBox.Text.Length;
61              }
62          }
63
64          private void refreshTbx_LostFocus(object sender, RoutedEventArgs e) {
65              if (((sender as TextBox).Text == "0") || ((sender as TextBox).Text
                  == String.Empty)) {
66                  (sender as TextBox).Text = null;
67              }
68          }
69      }
70  }
71
```

```
<UserControl x:Class="GpxWpfApplication.RadioSelection"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="350" d:DesignWidth="300">
    <Grid DockPanel.Dock="Top">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
        </Grid.RowDefinitions>
        <Label Grid.Row="0" FontWeight="Bold" FontStyle="Italic">
            Leistungskategorie
        </Label>
        <DockPanel Grid.Row="1" Grid.Column="0" DockPanel.Dock="Top"
          HorizontalAlignment="Center" LastChildFill="False">
            <!-- Die RadioButtons werden ueber die RadioSelecton-Property an
              das Code-behind gebunden. Der CommandParameter dient dabei als
              Identifizierung -->
            <RadioButton DockPanel.Dock="Top" Content="4:45 h"
              Command="{Binding RadioSelection}" CommandParameter="Avg_04h45"
              GroupName="generic" ></RadioButton>
            <RadioButton DockPanel.Dock="Top" Content="5:00 h"
              Command="{Binding RadioSelection}" CommandParameter="Avg_05h00"
              GroupName="generic"></RadioButton>
            <RadioButton DockPanel.Dock="Top" Content="5:15 h"
              Command="{Binding RadioSelection}" CommandParameter="Avg_05h15"
              GroupName="generic"></RadioButton>
            <RadioButton DockPanel.Dock="Top" Content="5:30 h"
              Command="{Binding RadioSelection}" CommandParameter="Avg_05h30"
              GroupName="generic"></RadioButton>
            <RadioButton DockPanel.Dock="Top" Content="5:45 h"
              Command="{Binding RadioSelection}" CommandParameter="Avg_05h45"
              GroupName="generic"></RadioButton>
            <RadioButton DockPanel.Dock="Top" Content="6:00 h"
              Command="{Binding RadioSelection}" CommandParameter="Avg_06h00"
              GroupName="generic"></RadioButton>
        </DockPanel>
        <DockPanel Grid.Row="1" Grid.Column="1" DockPanel.Dock="Top"
          HorizontalAlignment="Center" LastChildFill="False">
            <RadioButton DockPanel.Dock="Top" Content="6:15 h"
              Command="{Binding RadioSelection}" CommandParameter="Avg_06h15"
              GroupName="generic"></RadioButton>
            <RadioButton DockPanel.Dock="Top" Content="6:30 h"
              Command="{Binding RadioSelection}" CommandParameter="Avg_06h30"
              GroupName="generic"></RadioButton>
            <RadioButton DockPanel.Dock="Top" Content="6:45 h"
              Command="{Binding RadioSelection}" CommandParameter="Avg_06h45"
              GroupName="generic"></RadioButton>
```

```
34              <RadioButton DockPanel.Dock="Top" Content="7:00 h"
                  Command="{Binding RadioSelection}" CommandParameter="Avg_07h00"
                  GroupName="generic"></RadioButton>
35              <RadioButton DockPanel.Dock="Top" Content="7:15 h"
                  Command="{Binding RadioSelection}" CommandParameter="Avg_07h15"
                  GroupName="generic"></RadioButton>
36              <RadioButton DockPanel.Dock="Top" Content="7:30 h"
                  Command="{Binding RadioSelection}" CommandParameter="Avg_07h30"
                  GroupName="generic"></RadioButton>
37          </DockPanel>
38      </Grid>
39  </UserControl>
40
```

```
1  using System.Windows.Controls;
2
3  namespace GpxWpfApplication {
4
5      public partial class RadioSelection : UserControl {
6          public RadioSelection() {
7              InitializeComponent();
8          }
9      }
10 }
11
```

```
1  using System;
2  using System.Linq;
3  using System.Runtime.InteropServices;
4  using System.Windows;
5  using ESRI.ArcGIS.Geodatabase;
6
7  namespace GpxWpfApplication {
8      //Die SearchQuery-Klasse liest mittels der ICursor-Schnittstelle Daten aus
         dem ToC und berechnet Prozent sowie arithmetisches Mittel
9      sealed class SearchQuery : BaseQuery {
10
11         private double _counter;
12
13         //Konstruktor- und Super-Konstruktoraufruf
14         public SearchQuery(ITable table, int column, string @where) : base
             (table, column, @where) {
15             _cursor = base.InitCursorObject(table);
16         }
17
18         //Berechnet die Abweichung in Prozent zwischen Benutzereingabe und
             CalculateMean-Rueckgabewert
19         public double CalculatePercentage(double userVelocity) {
20             return userVelocity / (CalculateMean()) * 100;
21         }
22
23         //Berechnet den Durchschnitt der Spalte <<instance>>SearchQuery._column
             in Bezug auf den QueryFilter des Objekts
24         public double CalculateMean() {
25             _sum = _counter = 0;
26             _row = _cursor.NextRow();
27             while (_row != null) {
28                 _counter++;
29                 _sum += (double)_row.Value[_column];
30                 _row = _cursor.NextRow();
31             }
32             return _sum / _counter; ;
33         }
34     }
35 }
36
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.Windows;
7
8  namespace GpxWpfApplication {
9      //Die SimulationLine Klasse dient als Code-behind der SimulationLineGui.cs, ↵
           dessen Aufgabe wiederum ist nur das Setzen der WPF-Elemente in die View.
10     class SimulationLine {
11         //Die Methode ist konvertiert die Unicode-Sonderzeichen der            ↵
               Vergleichsoperatoren in fuer die Queries lesbare Zeichen.
12         private string convertToAsciiDelimiter(string delimiter) {
13             switch (delimiter) {
14                 case "\u2264":
15                     return "<=";
16                 case "\u2265":
17                     return ">=";
18                 default:
19                     return delimiter;
20             }
21         }
22
23         //Fields und Properties der Klasse
24         private string _delimiter1;
25
26         public string Delimiter1
27         {
28             get { return convertToAsciiDelimiter(_delimiter1); }
29             set { _delimiter1 = value; }
30         }
31
32         private string _delimiter2;
33
34         public string Delimiter2
35         {
36             get { return convertToAsciiDelimiter(_delimiter2); }
37             set { _delimiter2 = value; }
38         }
39
40         private double _slope1;
41
42         public double Slope1
43         {
44             get { return _slope1; }
45             set { _slope1 = value; }
46         }
47
48         private double _slope2;
49
50         public double Slope2
51         {
52             get { return _slope2; }
53             set { _slope2 = value; }
54         }
```

```
55
56          private double _velocity;
57
58          public double Velocity
59          {
60              get { return _velocity; }
61              set { _velocity = value; }
62          }
63
64          //Einziger Konstruktor der Klasse
65          public SimulationLine(string delimiter1, string delimiter2, double
              slope1, double slope2, double velocity) {
66              this._delimiter1 = delimiter1;
67              this._delimiter2 = delimiter2;
68              this._slope1 = slope1;
69              this._slope2 = slope2;
70              this._velocity = velocity;
71          }
72
73          public void PrintProperties(ViewModel vm) {
74              System.Windows.Application.Current.Dispatcher.Invoke((Action)(() =>
                  {
75                   MessageBox.Show("tick1" + vm.View.PacingEven.IsChecked.ToString
                       () + Environment.NewLine +
76                     "tick2" + vm.View.PacingSlow.IsChecked.ToString() +
                       Environment.NewLine +
77                     "distanz km:  " + vm.View.PacingDistance +
                       Environment.NewLine +
78                     "geschw km/h: " + vm.View.PacingVelocity);
79              }));
80          }
81
82          public override string ToString() {
83              return String.Format("{2} {0} x {1} {3}: {4} km/h", Delimiter1,
                    Delimiter2, Slope1, Slope2, Velocity);
84          }
85      }
86  }
87
```

```
 1  using System;
 2  using System.Windows;
 3  using System.Windows.Controls;
 4
 5  namespace GpxWpfApplication {
 6      //Die SimulationLineGui-Klasse dient zur Erstellung mehrerer WPF-Elemente ↵
          und fuegt diese der View hinzu
 7      class SimulationLineGui {
 8
 9          private static int _counter = 0;
10
11          public static int Counter
12          {
13              get { return SimulationLineGui._counter; }
14              set { SimulationLineGui._counter = value; }
15          }
16
17          internal static StackPanel Sp;
18          internal static ScrollViewer Sv;
19
20          public SimulationLineGui() {
21              WrapPanel wp = new WrapPanel {
22                  Orientation = Orientation.Horizontal
23              };
24              CheckBox cbxInitial = new CheckBox {
25                  Name = "cxb1",
26                  Margin = new Thickness(5, 0, 5, 0),
27                  VerticalAlignment = VerticalAlignment.Center
28              };
29              wp.Children.Add(cbxInitial);
30              ComboBox cb1 = new ComboBox {
31                  IsEnabled = false
32              };
33              ComboBox cb2 = new ComboBox {
34                  IsEnabled = false
35              };
36
37              ComboBoxItem cbi1 = new ComboBoxItem { Content = "<", };
38              ComboBoxItem cbi2 = new ComboBoxItem { Content = "\u2264" };
39              ComboBoxItem cbi3 = new ComboBoxItem { Content = ">" };
40              ComboBoxItem cbi4 = new ComboBoxItem { Content = "\u2265" };
41              ComboBoxItem cbi5 = new ComboBoxItem { Content = "<", };
42              ComboBoxItem cbi6 = new ComboBoxItem { Content = "\u2264" };
43              ComboBoxItem cbi7 = new ComboBoxItem { Content = ">" };
44              ComboBoxItem cbi8 = new ComboBoxItem { Content = "\u2265" };
45              cb1.Margin = new Thickness(5, 0, 5, 0);
46              cb1.Items.Add(cbi1);
47              cb1.Items.Add(cbi2);
48              cb1.Items.Add(cbi3);
49              cb1.Items.Add(cbi4);
50              cb1.Width = 36;
51              cb1.Margin = new Thickness(5, 0, 5, 0);
52              cb2.Items.Add(cbi5);
53              cb2.Items.Add(cbi6);
54              cb2.Items.Add(cbi7);
55              cb2.Items.Add(cbi8);
```

```
56              cb2.Width = 36;
57              cb2.Margin = new Thickness(5, 0, 5, 0);
58              Label lbl1 = new Label {
59                  Margin = new Thickness(5, 0, 5, 0),
60                  Content = "Steigung von"
61              };
62              TextBox tbxLowBorder = new TextBox {
63                  IsEnabled = false,
64                  Width = 30
65              };
66              wp.Children.Add(lbl1);
67              wp.Children.Add(cb1);
68              wp.Children.Add(tbxLowBorder);
69              Label lblX = new Label {
70                  Margin = new Thickness(5, 0, 5, 0),
71                  Content = "bis"
72              };
73              wp.Children.Add(lblX);
74              wp.Children.Add(cb2);
75              TextBox tbxHighBorder = new TextBox {
76                  IsEnabled = false,
77                  Width = 30
78              };
79              wp.Children.Add(tbxHighBorder);
80              Label lbl2 = new Label {
81                  Margin = new Thickness(5, 0, 5, 0),
82                  Content = "km/h:"
83              };
84              wp.Children.Add(lbl2);
85              TextBox tbxVelocityInput = new TextBox {
86                  IsEnabled = false,
87                  Width = 30,
88                  Margin = new Thickness(5, 0, 5, 0)
89              };
90              wp.Children.Add(tbxVelocityInput);
91              Sp.Children.Add(wp);
92
93              cbxInitial.Checked += new RoutedEventHandler(AddCheckbox);
94              cbxInitial.Unchecked += new RoutedEventHandler(RemoveCheckbox);
95
96          }
97
98          //Erstelle ein neues SimulationLineGui-Objekt, wenn die Checkbox
              aktiviert wurde
99          void AddCheckbox(object sender, RoutedEventArgs e) {
100             _counter++;
101             string s = String.Empty;
102             CheckBox checkBox = sender as CheckBox;
103             if (checkBox != null) {
104                 WrapPanel wrapPanel = checkBox.Parent as WrapPanel;
105                 if (wrapPanel != null)
106                     foreach (var item in wrapPanel.Children) {
107                         s += item.ToString() + "\n";
108                         UIElement uiElement = item as
                        System.Windows.UIElement;
109                         if (uiElement != null) uiElement.IsEnabled = true;
```

```
110                     }
111                 }
112                 SimulationLineGui sl = new SimulationLineGui();
113                 Sv.ScrollToBottom();
114             }
115 
116             //Loesche das entsprechende SimulationLineGui-Objekt aus der View,
                  wenn dessen Checkbox deaktiviert wurde
117             void RemoveCheckbox(object sender, RoutedEventArgs e) {
118                 _counter--;
119                 CheckBox checkBox = sender as CheckBox;
120                 if (checkBox != null) {
121                     WrapPanel wrapPanel = (checkBox.Parent) as WrapPanel;
122                     if (wrapPanel != null) {
123                         StackPanel stackPanel = wrapPanel.Parent as StackPanel;
124                         if (stackPanel != null)
125                             stackPanel.Children.Remove(wrapPanel);
126                     }
127                 }
128             }
129         }
130     }
131 
```

```
1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
4  using System.Globalization;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8  using System.Windows;
9  using System.Windows.Data;
10
11 namespace GpxWpfApplication {
12     //Die Klasse erlaubt es mittels des IValueConverter-Interfaces Daten der
         GUI beliebig anzupassen.
13     class SlopeToStringConverter : IValueConverter {
14
15         public object Convert(object value, Type targetType, object parameter,
             CultureInfo culture) {
16             if ((value as IEnumerable) == null) {
17                 throw new Exception("unable to convert " + value.ToString() + "
                       to " + targetType.ToString());
18             }
19             //Konvertiert die Steigungsbereiche der SimulationLines in die
                 beschraenkte Intervall-Notation
20             double[] range = value as double[];
21             if (range != null && !double.IsNaN(range[1])) {
22                 return ("[" + String.Join(" bis ", (value as double[])) + ")");
23             } else {
24                 if (range != null)
25                     return ("[" + range[0].ToString() + " bis " + ((range[0] >
                       0) ? "\u221E)" : "-\u221E)"));
26             }
27             return null;
28         }
29
30         public object ConvertBack(object value, Type targetType, object
             parameter, CultureInfo culture) {
31             return null;
32         }
33     }
34 }
35
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Shapes;
using System.Windows.Media;
using System.ComponentModel;
using System.Windows.Documents;

namespace GpxWpfApplication {
    //SortAdorner zeichnet mittels der Media-Library die Indikatoren der Sortierrichtung.
    public class SortAdorner : Adorner {

        private static readonly Geometry AscGeometry =
                Geometry.Parse("M 0 4 L 3.5 0 L 7 4 Z");

        private static readonly Geometry DescGeometry =
                Geometry.Parse("M 0 0 L 3.5 4 L 7 0 Z");

        public ListSortDirection Direction { get; private set; }

        public SortAdorner(UIElement element, ListSortDirection dir)
            : base(element) {
            this.Direction = dir;
        }

        protected override void OnRender(DrawingContext drawingContext) {
            base.OnRender(drawingContext);

            if (AdornedElement.RenderSize.Width < 20)
                return;

            TranslateTransform transform = new TranslateTransform(
                AdornedElement.RenderSize.Width - 15,
                (AdornedElement.RenderSize.Height - 5) / 2
            );
            drawingContext.PushTransform(transform);

            Geometry geometry = AscGeometry;
            if (this.Direction == ListSortDirection.Descending)
                geometry = DescGeometry;
            drawingContext.DrawGeometry(Brushes.Black, null, geometry);

            drawingContext.Pop();
        }
    }
}

```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using ESRI.ArcGIS.Geodatabase;
7  using System.Diagnostics;
8  using System.Runtime.InteropServices;
9  using System.Windows;
10 using ESRI.ArcGIS.Carto;
11
12 namespace GpxWpfApplication {
13     sealed class UpdateQuery : BaseQuery {
14         private double _tmpValue;
15
16         //Konstruktor- und Super-Konstruktoraufruf
17         public UpdateQuery(ITable table, int column, string where)
18             : base(table, column, where) {
19             _cursor = InitCursorObject(table);
20         }
21
22         protected override ICursor InitCursorObject(ITable table) {
23             Debug.WriteLine(@"Debug-@UpdateQuery: protected override ICursor
                 InitCursorObject(ITable table)");
24             _cursor = null;
25             return table.Update(_queryFilter, false);
26         }
27
28         //Berechnet die neuen Werte der Zelle anhand der Prozentwerte im
             Argument.
29         public void UpdateDatasetByPercent(double percent) {
30             if (_sum != 0.0D) _cursor = InitCursorObject(_table);
31             _row = this._cursor.NextRow();
32             while (_row != null) {
33                 _tmpValue = (double)_row.get_Value(_column) * percent / 100;
34                 _row.set_Value(_column, _tmpValue);
35                 _row.set_Value(_column + 1, 10 / _tmpValue * 3.6);
36                 _cursor.UpdateRow(_row);
37                 _row = _cursor.NextRow();
38             }
39             _cursor.Flush();
40         }
41
42         //Berechnet die Summe der benoetigten Zeiten in der angegebenen Spalte,
             wobei jede Zelle die Summe der Werte der vorherigen Zeile und der
             aktuellen Zeile beinhaltet.
43         //Der Rueckgabewert gibt Auskunft ueber die Anzahl der Zellen fuer
             weitere Berechnungen.
44         public int CalculateSumAndGetRowCounter(int sumColNum) {
45             double currentSum = 0;
46             int rowCounter = 0;
47             _row = this._cursor.NextRow();
48             while (_row != null) {
49                 _row.Value[sumColNum] = (double)_row.Value[_column + 1] +
                     currentSum;
50                 currentSum = (double)_row.Value[sumColNum];
```

```
51                  _cursor.UpdateRow(_row);
52                  _row = _cursor.NextRow();
53                  rowCounter++;
54              }
55              _cursor.Flush();
56              return rowCounter;
57          }
58
59          //Konvertiert die Zeitangabe der Sektoren in ein fuer GPS Geraete
              lesbares XML Element.
60          public string FillTagColumn(int sumRows) {
61              string returnVal = "Error while writing tags in table";
62              double timeInSeconds;
63              int numberOfRowsInTagged1 = 0;
64              DateTime dummyTime = new DateTime(2000, 1, 1, 0, 0, 0);
65              DateTime dummyTimeAdd;
66              string pre = "<2000-01-01T";
67              string post = "Z>";
68              Debug.WriteLine("flag 4.1");
69              Debug.WriteLine("tagColNum: " + _column);
70              Debug.WriteLine("sumRows: " + sumRows);
71              _row = this._cursor.NextRow();
72              while (_row != null) {
73                  timeInSeconds = Math.Round((double)_row.Value[32], 3) * 1000;
74                  dummyTimeAdd = dummyTime.AddMilliseconds((int)timeInSeconds);
75                  _row.Value[_column] = pre + dummyTimeAdd.ToString
                      ("HH:mm:ss.fff") + post;
76                  _cursor.UpdateRow(_row);
77                  _row = _cursor.NextRow();
78                  numberOfRowsInTagged1++;
79                  if (sumRows == numberOfRowsInTagged1) {
80                      returnVal = (dummyTime.AddMilliseconds((int)timeInSeconds *
                          2)).ToString("HH:mm:ss");
81                  }
82              }
83              Debug.WriteLine("numberOfRowsInTagged1: " + numberOfRowsInTagged1);
84              Debug.WriteLine("returnVal: " + returnVal);
85              _cursor.Flush();
86              return returnVal;
87          }
88      }
89  }
90
```

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.ComponentModel;
 4  using System.Linq;
 5  using System.Runtime.CompilerServices;
 6  using System.Text;
 7  using System.Threading.Tasks;
 8  using System.Windows;
 9  using System.Diagnostics;
10  using ESRI.ArcGIS.ArcMapUI;
11  using ESRI.ArcGIS.Carto;
12  using System.Windows.Media;
13  using System.Windows.Controls;
14  using ESRI.ArcGIS.Geodatabase;
15  using ESRI.ArcGIS.esriSystem;
16  using System.Windows.Data;
17  using System.Windows.Documents;
18  using System.Windows.Input;
19  using System.Windows.Media.Imaging;
20  using System.Windows.Navigation;
21  using System.Windows.Shapes;
22  using System.Windows.Controls.Primitives;
23  using System.Windows.Media.Effects;
24
25  namespace GpxWpfApplication {
26      // Die Klasse muss von DependencyObject erben, sowie das Interface
          INotifyPropertyChanged implementieren, um DependencyProperties handhaben
          zu koennen.
27      public class ViewModel : DependencyObject, INotifyPropertyChanged {
28
29          public MainWindow View;
30          internal static ViewModel CurrentViewModel;
31          internal BackgroundWorker BgWorker;
32
33          //CreateView erzeugt ein neues MainWindow-Objekt welches als View
              dient und die GUI des Programms repraesentiert.
34          public void CreateView() {
35              View = new MainWindow(this) {
36                  Topmost = true
37              };
38              //InitializeDependencies stellt sicher, dass die notwendigen
                  Objekte geladen werden konnten und setzt zugleich einige
                  Properties des ViewModels.
39              InitializeDependencies();
40              View.DataContext = this;
41              //Oeffnet eine neue Windows Command Line, sofern es sich um  eine
                  Debug-Konfiguration des Builds handelt.
42  #if DEBUG
43              ConsoleHelper.AllocConsole();
44              var listeners = new TraceListener[] { new TextWriterTraceListener
                  (Console.Out) };
45              Debug.Listeners.AddRange(listeners);
46              Debug.WriteLine(@"console output for debugging initialized");
47  #endif //DEBUG
48              //Zeige die Benutzerschnittstelle wenn keine Abbruch-Flag gesetzt
                  wurde.
```

```
49              if (!ViewModelConfig.HasCloseOnPanicRequests) {
50                  View.ShowDialog();
51              }
52          }
53
54          //IsGuiEnabled ist eine Dependency-Property die mittels
              INotifyPropertyChanged in der Lage ist die gesamte
              Benutzerschnittstelle zu deaktivieren.
55          public bool IsGuiEnabled
56          {
57              get
58              {
59                  return !(bool)this.GetValue(IsGuiEnabledProperty);
60              }
61              set
62              {
63                  this.SetValue(IsGuiEnabledProperty, (!(bool)value));
64                  OnPropertyChanged("IsGuiEnabled");
65              }
66          }
67
68          public static readonly DependencyProperty IsGuiEnabledProperty =
69              DependencyProperty.Register("IsGuiEnabled", typeof(bool), typeof
                (MainWindow), new PropertyMetadata(false));
70
71          protected void OnPropertyChanged(string name) {
72              PropertyChangedEventHandler handler = PropertyChanged;
73              if (handler != null) {
74                  handler(this, new PropertyChangedEventArgs(name));
75              }
76          }
77
78          #region InitializeDependencies
79          //InitializeDependencies legt fuer die Applikation notwendige Werte
              und Referenzen, meist in Abhaengigkeit der App.Doc Eigenschaft,
              fest.
80          private void InitializeDependencies() {
81              try {
82                  App app = Application.Current as GpxWpfApplication.App;
83                  if (app != null) {
84                      ViewModelConfig.MxDoc = app.Doc as IMxDocument;
85                      if (ViewModelConfig.MxDoc != null) ViewModelConfig.Map =
                        ViewModelConfig.MxDoc.FocusMap;
86                      else throw new NullReferenceException(Environment.NewLine
                        + "Dokument ungueltig");
87                      if (ViewModelConfig.Map.LayerCount == 0) throw new
                        ArgumentException(Environment.NewLine + "Es wurde kein
                        Layer gefunden");
88                      ViewModelConfig.Layer = ViewModelConfig.Map.Layer[0] as
                        ILayer2;
89                      if (ViewModelConfig.Layer != null &&
                        ViewModelConfig.Layer.Name == "PointtoGPX_Modell")
90                          ViewModelConfig.Table = ViewModelConfig.Layer as
                        ITable;
91                      else throw new NullReferenceException(Environment.NewLine
                        + "Ung\u00fcltiger Layer auf Position 1");
```

```
92                      CurrentViewModel = this;
93                  }
94              } catch (ArgumentException ex) {
95                  CloseOnPanic(ex.Message);
96              } catch (NullReferenceException ex) {
97                  CloseOnPanic("Notwendige Referenz nicht vorhanden" +
                      ex.Message);
98              }
99              //InitializeSlopeList erzeugt eine neue Liste, die Teil des Code-
                  behinds der DataGridView.xaml ist.
100             InitializeSlopeList();
101             //Erhalte mittels FindNode eine Referenz auf das zur Darstellung
                  der Leistungsklassen verwendete XAML-Objekt.
102             ViewModelConfig.LvUsers = FindNode(View.TabMod, "LvUsers") as
                  ListView;
103         }
104         #endregion
105
106         //CloseOnPanic ist eine Methode, die es erlaubt den Programmfluss auf
              sicherem Wege abzubrechen.
107         #region Close MainWindow on panic
108         internal void CloseOnPanic(string arg) {
109             ViewModelConfig.HasCloseOnPanicRequests = true;
110             MessageBox.Show("Fehler: " + arg);
111             Application.Current.MainWindow.Close();
112             Application.Current.Shutdown();
113         }
114         #endregion
115
116         //Die SlopeCategory-Structure findest bei der Befuellung der SlopeList
              verwendung.
117         #region Struct SlopeCategory
118         public struct SlopeCategory {
119             public double[] Slopes;
120             public string Name;
121             public SlopeCategory(double[] slopes, string name) {
122                 this.Slopes = slopes;
123                 this.Name = name;
124             }
125         }
126
127         //SlopeList versorgt die 'List<LapTime> Items' indirekt mit Daten, da
              mit Hilfe der SlopeCategory-Objekte die notwendigen Werte zur
              Erzeugung von LapTime-Objekten geliefert werden.
128         private void InitializeSlopeList() {
129             ViewModelConfig.SlopeList = new List<ViewModel.SlopeCategory> {
130                 new ViewModel.SlopeCategory(new double[] { -11, double.NaN },
                      "Starkes Gef\u00e4lle") ,
131                 new ViewModel.SlopeCategory(new double[] {-11, -5 },
                      "Mittleres Gef\u00e4lle"),
132                 new ViewModel.SlopeCategory(new double[] { -5, -3 }, "Leichtes
                       Gef\u00e4lle"),
133                 new ViewModel.SlopeCategory(new double[] { -3, 3 }, "Ebene"),
134                 new ViewModel.SlopeCategory(new double[] { 3, 5 }, "Leichte
                      Steigung"),
135                 new ViewModel.SlopeCategory(new double[] { 5, 11 }, "Mittlere
```

```
                    Steigung"),
136                 new ViewModel.SlopeCategory(new double[] { 11, double.NaN },
                      "Starke Steigung")
137             };
138         }
139         #endregion
140
141         //Command-Implementierung fuer den GenerateGpx-Button der
              DataGridView.xaml Datei.
142         #region ButtonGpx
143         private GenericButton _buttonGpx;
144         public GenericButton ButtonGpx
145         {
146             get { return _buttonGpx ?? (_buttonGpx = new GenericButton
                      (GenerateGpx)); }
147             set { _buttonGpx = value; }
148         }
149
150         //Die Methode, die dem ButtonGpx-Objekt als Action-Delegate fuer die
              Execute-Methode des GenericButtons dient.
151         private void GenerateGpx() {
152             //Die ProgressBar und den Button der View referenzieren, um deren
                  Sichtbarkeit und Animation zu steuern.
153             var pbar = FindNode(View.TabControl.SelectedContent as
                  DependencyObject, "Pbar") as
                  System.Windows.Controls.ProgressBar;
154             if (pbar != null) {
155                 pbar.Visibility = Visibility.Visible;
156                 var button = FindNode(View.TabControl.SelectedContent as
                      DependencyObject, "GenerateGpx") as
                      System.Windows.Controls.Button;
157                 if (button == null) return;
158                 button.Effect = new BlurEffect();
159                 //Erstellung eines DataAccessLayer-Objekts. Der Konstruktor
                      benoetigt das ViewModel als Argument.
160                 DataAccessLayer dal = new DataAccessLayer(this);
161                 //Der BackgroundWorker in System.ComponentModel implementiert
                      das Event-based Asynchronous Pattern um so einen Worker-
                      Thread zu starten, der ueber Fortschitt und Beendigung
                      berichtet, ohne explizit die Synchronisation handhaben zu
                      muessen.
162                 BgWorker = new BackgroundWorker() { WorkerReportsProgress =
                      true };
163                 BgWorker.DoWork += (s, ea) => {
164                     //Weise den BackgroundWorker an, die GUI im aktuellen
                        Thread zu aktualisieren und im Worker-Thread die Datenbank
                         handzuhaben.
165                     pbar.Dispatcher.Invoke(() => pbar.Value += 1);
166                     this.Dispatcher.Invoke(() => IsGuiEnabled = false);
167                     dal.StartQuerying();
168                 };
169                 BgWorker.ProgressChanged += (s, ea) => { };
170                 BgWorker.RunWorkerCompleted += (s, ea) => {
171                     //Bei Beendigung des Worker-Threads feuert im
                        urspruenglichen Thread der RunWorkerCompleted-Event.
172                     pbar.Visibility = System.Windows.Visibility.Collapsed;
```

```
173                     string ergebnisTab = String.Empty;
174                     string folder = String.Empty;
175                     if (this.VMSelectedTabIndex == (int)TabEnum.Modellierung)
                        {
176                         folder = "Modellierung";
177                     } else {
178                         Debug.Assert(this.VMSelectedTabIndex == 1);
179                         folder = "Simulation";
180                     }
181                     button.Effect = null;
182                     if (ViewModelConfig.WasSuccessful) {
183                         MessageBox.Show(String.Format("GPX-Datei mit einer
                        Fahrzeit von {0} Stunden wurde erfolgreich unter \"C:\\{1}
                        \\Ironman_Austria\\\" generiert",
                        ViewModelConfig.LapTimeX2, folder), String.Format
                        ("Ergebnis {0}", folder), MessageBoxButton.OK,
                        MessageBoxImage.Information);
184                     }
185                     IsGuiEnabled = true;
186                 };
187             }
188             //RunWorkerAsync feuert den DoWork-Event des BackgroundWorkers.
189             BgWorker.RunWorkerAsync();
190
191         }
192         #endregion
193
194         //RadioSelection Eigenschaft
195         private GenericButton _radioSelection;
196         public GenericButton RadioSelection
197         {
198             //Falls das Objekt zur Zeit des Data-Bindings nicht erstellt
                wurde, wird es mittels ??-null-operator-Test erzeugt.
199             get
200             {
201                 return _radioSelection ??
202                       (_radioSelection = new GenericButton(RadioSelected,
                    CanChangeRadio));
203             }
204         }
205         private bool CanChangeRadio() {
206             return IsGuiEnabled;
207         }
208         //Die Methode fuer das Action-Delegate der Radio Buttons.
209         private void RadioSelected() {
210             //Anpassung der Benutzerschnittstellengroesse.
211             if (_vmSelectedTabIndex == 0) {
212                 if (!ViewModelConfig.HasChangedSizeForListView[0]) {
213                     ViewModelConfig._heightMod +=
                        ViewModelConfig.DataGridViewHeight;
214                     Application.Current.MainWindow.Height =
                        ViewModelConfig._heightMod;
215                     ViewModelConfig.HasChangedSizeForListView[0] = true;
216                 }
217             } else {
218                 if (!ViewModelConfig.HasChangedSizeForListView[1]) {
```

```
219                     ViewModelConfig._heightSim +=
                        ViewModelConfig.DataGridViewHeight;
220                     Application.Current.MainWindow.Height =
                        ViewModelConfig._heightSim;
221                     ViewModelConfig.HasChangedSizeForListView[1] = true;
222                 }
223             }
224
225             //Den Parameter des Commands versuchen auszulesen.
226             var something = _radioSelection.CommandParamObj as string;
227             if (something == null)
228                 return;
229             // Falls der Parameter eine Zeichenkette ist, wird der Wert im
                  ColumnsDict-Dictionary ausgelesen
230             ViewModelConfig.RadioValue =
                  ViewModelConfig.ColumnsDict.ContainsKey(something) ?
                  ViewModelConfig.ColumnsDict[something] : -1;
231             //Falls der aktuell selektierte Reiter der Simulationsreiter ist,
                  wird sogleich die Geschwindigkeit innerhalb der Labestationen
                  berechnet und in der GUI angezeigt.
232             //if (_vmSelectedTabIndex == 1) {
233             if (_vMSelectedTabAsEnum == TabEnum.Simulation) {
234                 DataAccessLayer.CalculateAidStationVelocity(this);
235             }
236             //PopulateList gibt den Anstoss fuer die Berechnung der
                  Geschwindigkeiten der Steigungsklassen in Bezug auf die aktuelle
                   Leistungsklasse.
237             PopulateList(ViewModelConfig.RadioValue);
238         }
239
240         //Die FindNode-Methode sucht im XAML-Logical Tree ab dem
              ueberlieferten DependencyObject abwaerts nach dem Objekt mit dem
              angegebenen Namen und gibt eine Referenz davon zurueck.
241         public object FindNode(DependencyObject obj, string name) {
242             object node = LogicalTreeHelper.FindLogicalNode(obj, name);
243             if (node != null) {
244                 return LogicalTreeHelper.FindLogicalNode(obj, name);
245             } else {
246                 CloseOnPanic(String.Format("Node {0} nicht gefunden", name));
247                 return null;
248             }
249         }
250
251         //ClearSimulationLineGui loescht alle Framework-Elemente im StackPanel
               namens 'StackPanel1'.
252         internal void ClearSimulationLineGui() {
253             this.View.StackPanel1.Children.RemoveRange(0,
                  this.View.StackPanel1.Children.Count - 1);
254         }
255
256         //PopulateList liefert der Listview im Front-End den notwendigen
              DataContext.
257         private void PopulateList(int column) {
258             ViewModelConfig.Items = new List<LapTime>();
259             ViewModelConfig.LvUsers.DataContext = null;
260             ViewModelConfig.LvUsers.ItemsSource = null;
```

```
261             ViewModelConfig.LvUsers.Items.Clear();
262             foreach (ListViewItem item in ViewModelConfig.LvUsers.Items) {
263                 ViewModelConfig.LvUsers.Items.Remove(item);
264             }
265             ViewModelConfig.LvUsers.Items.Refresh();
266             for (int i = 0; i < ViewModelConfig.SlopeList.Count; i++) {
267                 //Fuege der Collection ein neues LapTime-Objekt hinzu, welches
                       Angaben ueber Steigungskategorie, Steigungswerte und
                    Geschwidigkeitsdurchschnitt beinhaltet.
268                 ViewModelConfig.Items.Add(CreateListItems(column,
                    ViewModelConfig.SlopeList[i].Slopes[0],
                    ViewModelConfig.SlopeList[i].Slopes[1],
                    ViewModelConfig.SlopeList[i].Name));
269             }
270             ViewModelConfig.LvUsers.ItemsSource = ViewModelConfig.Items;
271         }
272
273         //Erstellt mittels der uebergebenen Argumente ein neues LapTime Item
              und gibt dieses zurueck.
274         private LapTime CreateListItems(int column, double min, double max,
              string name) {
275             return new LapTime() {
276                 Kategorie = name,
277                 Slope = new double[] { min, max },
278                 //Berechnung des Wertes fuer die Durchschnittsgeschwindigkeit.
279                 Velocity = EpsilonComparison(GetMeanValue(column, min, max),
                    epsilon: 0.000000001)
280             };
281         }
282
283         //Die Methode EpsilonComparison kompensiert moegliche
              Genauigkeitsverluste durch die staendigen Gleitkommazahloperationen,
               indem der Wert zur naechsten Ganzzahl gerundet wird, wenn die
              Differenz unter der in Epsiolon angegebenen Grenze liegt.
284         private double EpsilonComparison(double meanValue, double epsilon) {
285             if ((meanValue % 1) != 0 && Math.Abs(meanValue - Math.Round
                  (meanValue, MidpointRounding.AwayFromZero)) < epsilon) {
286                 Debug.WriteLine(@"Debug-@ViewModel.EpsilonComparison triggerd
                      by param0: " + meanValue + " /epsilon of " + epsilon);
287                 return Math.Round(meanValue, MidpointRounding.AwayFromZero);
288             }
289             return meanValue;
290         }
291
292         //Berechnung des Durchschnitts fuer die GUI mittels der SearchQuery
              Klasse.
293         private double GetMeanValue(int column, double min, double max) {
294             string where = BaseQuery.InitializeWhereStringForDataGridView(min,
                  max);
295             SearchQuery sq = new SearchQuery(ViewModelConfig.Table, column,
                where);
296             return sq.CalculateMean();
297         }
298
299         #region TabIndex
300         //Enumerationstyp zur leichteren Identifizierung der Tabs.
```

```
301         public enum TabEnum {
302             Modellierung = 0,
303             Simulation = 1
304         };
305
306         //Felder fuer die Indikation des aktuellen Tabs.
307         private int _vmSelectedTabIndex;
308         private TabEnum _vMSelectedTabAsEnum;
309
310         //Erstellung eines ViewModelConfig-Objekts, welches als Singleton
              implementiert wurde.
311         private ViewModelConfig _viewModelConfig = ViewModelConfig.Instance();
312
313         //Property fuer den Reiterindex, der zugleich die Property
              VMSelectedTabAsEnum setzt.
314         public int VMSelectedTabIndex
315         {
316             get { return _vmSelectedTabIndex; }
317             set
318             {
319                 _vmSelectedTabIndex = value;
320                 _vMSelectedTabAsEnum = (TabEnum)value;
321                 NotifyPropertyChanged();
322             }
323         }
324         //Property fuer den Reiterindex als Enumerationstyp.
325         public TabEnum VMSelectedTabAsEnum
326         {
327             get { return _vMSelectedTabAsEnum; }
328             set { _vMSelectedTabAsEnum = value; }
329         }
330
331         //Property fuer das ViewModelConfig-Singleton.
332         public ViewModelConfig ViewModelConfig
333         {
334             get { return _viewModelConfig; }
335         }
336
337         public event PropertyChangedEventHandler PropertyChanged;
338
339         //Die NotifyPropertyChanged-Methode des Interfaces
              INotifyPropertyChanged wird bei jedem Set-Accssor-Aufruf
              ausgefuehrt.
340         //Durch Anwendung des CallerMemberName-Attributs wird eine
              generischere Methode implementiert, die keine Auskunft ueber den
              Property-Namen benoetigt.
341         private void NotifyPropertyChanged([CallerMemberName] String
              propertyName = null) {
342             if (PropertyChanged != null) {
343                 //Erzeugt beim Eintritt der View in den Simulations-Tab eine
                      neue Zeile an GUI-Elementen, die den Benutzer die Anpassung
                      der Daten erlaubt.
344                 if (_vmSelectedTabIndex == (int)TabEnum.Simulation) {
345                     //Setze die ListView Collection des ViewModelConfig-
                        Singletons auf die in der Benutzerschnittstelle sichtbare
                        ListView.
```

```
346                     ViewModelConfig.LvUsers = FindNode(View.TabSim, "LvUsers")
                         as ListView;
347                     if (!ViewModelConfig._isDirty) {
348                         System.Windows.Application.Current.MainWindow.Height =
                        ViewModelConfig._heightSim;
349                         SimulationLineGui.Sp = View.StackPanel1;
350                         SimulationLineGui.Sp.Width = double.NaN;
351                         SimulationLineGui.Sv =
                        View.SimulationLineScrollViewer;
352                         SimulationLineGui sl = new SimulationLineGui();
353                         //Die isDirty-Flag gibt darueber auskunft, ob der
                        Reiter 'Simulation' zuvor bereits selektiert wurde. Diese
                        Auskunft ist fuer eine genauere Anpassung der GUI-Hoehe
                        notwendig.
354                         ViewModelConfig._isDirty = true;
355                     } else if (ViewModelConfig._isDirty) {
356                         System.Windows.Application.Current.MainWindow.Height =
                        ViewModelConfig._heightSim;
357                     }
358                 } else if (_vMSelectedTabAsEnum == TabEnum.Modellierung &&
                      ViewModelConfig._isDirty) {
359                     System.Windows.Application.Current.MainWindow.Height =
                        ViewModelConfig._heightMod;
360                     ViewModelConfig.LvUsers = FindNode(View.TabMod, "LvUsers")
                         as ListView;
361                 }
362                 //Event PropertyChanged feuern.
363                 PropertyChanged(this, new PropertyChangedEventArgs
                      (propertyName));
364             }
365         }
366         #endregion
367     }
368 }
369
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Windows.Controls;
4  using ESRI.ArcGIS.ArcMapUI;
5  using ESRI.ArcGIS.Carto;
6  using ESRI.ArcGIS.Geodatabase;
7
8  namespace GpxWpfApplication {
9      //Die ViewModelConfig-Klasse ist als Singleton Design Pattern implementiert
           und liefert Felder, die hauptsaechlich dem ViewModel dienlich sind.
10     public class ViewModelConfig {
11
12         private bool _hasCloseOnPanicRequests;
13         public List<ViewModel.SlopeCategory> SlopeList = null;
14         public bool WasSuccessful = false;
15         public string LapTimeX2 = String.Empty;
16         public bool[] HasChangedSizeForListView = new bool[2];
17         public int DataGridViewHeight = 147;
18         public List<LapTime> Items;
19         public ListView LvUsers;
20
21         //Singleton
22         private static ViewModelConfig _instance;
23
24         protected ViewModelConfig() { }
25
26         public static ViewModelConfig Instance() {
27             return _instance ?? (_instance = new ViewModelConfig());
28         }
29
30         //Das Dictionary ColumnsDict liefert anhand der angegebenen
             Spaltennamen die dementsprechenden Spaltenindizes.
31         public Dictionary<string, int> ColumnsDict = new Dictionary<string,
             int>()
32         {
33             {"Avg_04h45",  8},
34             {"Avg_05h00",  10},
35             {"Avg_05h15",  12},
36             {"Avg_05h30",  14},
37             {"Avg_05h45",  16},
38             {"Avg_06h00",  18},
39             {"Avg_06h15",  20},
40             {"Avg_06h30",  22},
41             {"Avg_06h45",  24},
42             {"Avg_07h00",  26},
43             {"Avg_07h15",  28},
44             {"Avg_07h30",  30}
45         };
46
47         //Das zweidimensionale Array liefert die von Labestationen betroffenen
             Sektoren.
48         internal int[,] AidStationIDs = new int[,] {
49             {2036, 2049},
50             {3138, 3153},
51             {5036, 5056},
52             {6278, 6285},
```

```
53              {7373, 7380},
54              {8604, 8630}
55          };
56
57          public bool _isDirty = false;
58          public int _heightSim = 552;
59          public int _heightMod = 236; //315
60
61          public bool HasCloseOnPanicRequests
62          {
63              get { return _hasCloseOnPanicRequests; }
64              set { _hasCloseOnPanicRequests = value; }
65          }
66
67          public int RadioValue { get; set; }
68          private ITable _table;
69
70          public ITable Table
71          {
72              get { return _table; }
73              set { _table = value; }
74          }
75
76          private IMxDocument _mxDoc;
77          private IMap _map;
78          private ILayer2 _layer;
79
80          public IMxDocument MxDoc
81          {
82              get { return _mxDoc; }
83              set { _mxDoc = value; }
84          }
85
86          public IMap Map
87          {
88              get { return _map; }
89              set { _map = value; }
90          }
91
92          public ILayer2 Layer
93          {
94              get { return _layer; }
95              set { _layer = value; }
96          }
97      }
98  }
```