Institut für Softwaretechnologie
Universität Stuttgart
Universitätsstraße 38
D--70569 Stuttgart

Diplomarbeit Nr. 3371

Modellierung von Straßennetzen auf Basis diskreter Messpunkte

Felix Krause

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. rer. nat. Stefan Wagner Prof. Dr.-Ing. Wolfram Ressel
Betreuer:	Dipl.-Inf. Asim Abdulkhaleq Dipl.-Ing. Kai Tejkl Dipl.-Ing. Marcos Manuel Sanchez
begonnen am:	9. Juli 2012
beendet am:	9. Januar 2013
CR-Klassifikation:	J.2, I.5.1

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

- **CLI**: Common Language Infrastructure (Spezifikation von Microsofts .NET-Infrastruktur). Siehe [3].
- **CSV**: Comma-Separated Values, siehe [12].
- **GPS**: Global Positioning System, siehe 2.3.1.
- **GTK**: GIMP Toolkit, siehe 2.7.
- **MVC**: Model-View-Controller, siehe 2.6.
- **UTM**: Universal Transverse Mercator, siehe 2.3.2.
- **WGS**: World Geodetic System, siehe 2.3.1.

Begriffsverzeichnis

- **Achse**: Linie, die den geometrischen Verlauf einer Straße darstellt. Von der Achse aus wird der Querschnitt der Straße modelliert.
- **Controller**: Konzept der Steuereinheit aus dem Model-View-Controller Architekturmuster. Siehe 2.6.
- **Knotenpunkt, Kreuzung**: Geografischer Bereich, an den mindestens drei Straßen grenzen. Der weniger präzise Begriff *Kreuzung* wird verwendet, da er der Terminologie der OpenDRIVE-Spezifikation (englisch: *Junction*) entspricht, siehe 2.5. Beide Begriffe sind synonym.
- **Messplan**: Name der Software, die im Rahmen dieser Diplomarbeit weiterentwickelt wird. Die Softwareversion, von der ausgegangen wird, trägt die Nummer 1 (zur Abgrenzung auch als 1.0 bezeichnet). Die Version, die die im Rahmen dieser Diplomarbeit entwickelten Erweiterungen enthält, trägt die Nummer 1.1.
- **Model**: Konzept des Datenmodells aus dem Model-View-Controller Architekturmuster. Siehe 2.6.
- **Steuerelement**: siehe *Widget*.
- **Tab**: Steuerelement der Benutzeroberfläche, mit dem man zwischen verschiedenen Ansichten hin- und herschalten kann. Vereinzelt wird auch die deutsche Bezeichnung *Karteireiter* verwendet. In diesem Dokument wird die englische Version verwendet, da diese auch im deutschen Sprachraum weiter verbreitet ist.
- **Trassierung, Trassierungselement**: Trassierung beschreibt die Linienführung von Verkehrwegen. Mithilfe verschiedener Trassierungselemente wird eine durchgehende räumliche Linie definiert, die den Verlauf des Verkehrsweges beschreibt. Im Kontext dieser Diplomarbeit ist der beschriebene Verkehrsweg immer eine Straße. Siehe 2.4.1.
- **Verknüpfungspunkt**: Im Kontext von Kreuzungen ist ein Verknüpfungspunkt ein Punkt, im dem eine Straße an eine Kreuzung anschließt.
- **View**: Konzept der Anzeigeeinheit aus dem Model-View-Controller Architekturmuster. Siehe 2.6.
- **Widget**, synonym *Steuerelement*: Abstrakte Bezeichnung für ein Element einer grafischen Benutzeroberfläche. Im Kontext dieser Diplomarbeit ist ein Widget außerdem im programmatisches Objekt, das eine Instanz von einer der Widget-Klassen ist, die GTK zur Verfügung stellt. Siehe 2.7.

Zusammenfassung

Diese Diplomarbeit beschäftigt sich mit der Erstellung einer Software, mit der existierende Straßennetze digital modelliert werden können. Grundlage der Modellierung sind Messpunkte, die von einem Messfahrzeug aufgezeichnet werden, welches das Straßennetz befährt und dabei in festen Zeitabständen seine Position erfasst.

Die Software wird auf Basis von Messplan 1.0 entwickelt, einem Programm, das bereits die Erfassung einzelner Straßen beziehungsweise Streckenabschnitten beherrscht. Davon ausgehend wird Funktionalität implementiert, um Kreuzungen erfassen und diese mit Straßen verknüpfen zu können. Das Resultat ist eine neue Version von Messplan, mit der die Erfassung von Straßennetzwerken möglich ist.

Kapitel 1

Einführung

1.1 Überblick

Messplan ist eine Software, die im Rahmen eines Studienprojekts am Institut für Straßen- und Verkehrswesen der Universität Stuttgart erstellt wurde. Sie ermöglicht es, aus einer Liste aus Koordinatenpunkten halbautomatisch die Trassierungselemente einer abgefahrenen Strecke zu erkennen. Diese Strecke kann anschließend in verschiedene Formate einschließlich OpenDRIVE exportiert werden.

OpenDRIVE [1] ist ein offenes, XML-basiertes Dateiformat, in dem Straßennetzwerke modelliert werden können. Es wird vor allem für Fahrsimulation genutzt. Es können alle wesentlichen Elemente wie Geraden, Kurven, Übergangsbögen, die Anzahl und Breite von Fahrspuren, Kreuzungen und die möglichen Relationen darauf sowie Höhenprofile und weitere Eigenschaften des Straßennetzes in OpenDRIVE abgelegt werden. Das Format wird in der Industrie herstellerübergreifend eingesetzt.

Im Zuge dieser Diplomarbeit wird die Software Messplan dahingehend erweitert, dass es möglich ist, Kreuzungspunkte zu definieren und mehrere Strecken an diesen Kreuzungspunkten zu verknüpfen, sodass ein komplettes Straßennetz, das mit einem Messfahrzeug abgefahren wurde, eingelesen und erkannt werden kann. Dabei sollen dem Nutzer Interaktionsmöglichkeiten gegeben werden, um falsche oder ungenaue Erkennung von Strecken und Kreuzungen manuell verbessern zu können. Schließlich soll das Resultat als Straßennetz in das OpenDRIVE-Format exportiert werden können, um das Netz anschließend in einem Fahrsimulator befahren zu können.

1.2 Motivation

Die Software Messplan wurde entwickelt, um existierende Straßen beziehungsweise Streckenabschnitte halbautomatisch mithilfe eines Messfahrzeugs digitalisieren zu können. Dabei wird versucht, die Trassierungselemente, mit denen Straße ursprünglich entworfen wurde, möglichst exakt zu rekonstruieren. Damit eignet sich Messplan als Werkzeug für Situationen, in denen die ursprünglichen Bauunterlagen einer Straße nicht mehr verfügbar sind.

Die Erweiterung von Messplan soll es nun möglich machen, ein ganzes Straßennetz zu digitalisieren und damit eine zusammenhängende Menge von Netzdaten erfassen zu können, die groß genug ist, um sie beispielsweise mit einem Fahrsimulator befahren zu können. Durch den Export in das OpenDRIVE-Format war das bisher schon für einzelne Strecken möglich, die fehlende Verknüpfung zu einem Netz beschränkte allerdings die Anwendungsmöglichkeiten.

1.3 Zielstellung

Ziel der Diplomarbeit ist, die Software Messplan dahingehend zu erweitern, dass in ihr statt einzelnen Straßen / Streckenabschnitten ganze Straßennetze auf Basis von gegebenen Messpunkten erfasst werden können. Messpunkte sind eine Liste von GPS-Koordinaten, die von einer Befah-

rung des Straßennetzes mit einem Messfahrzeug stammen. Ihm Rahmen dieser Aufgabe sollen folgende Funktionalitäten auf Umsetzbarkeit überprüft und implementiert werden:

- Erfassung von Knotenpunkten in Messplan. Dies ist bisher nicht implementiert und ist die Grundvoraussetzung für andere Funktionalitäten.
- Automatische Erkennung von Knotenbereichen auf der Basis der gegebenen Messdaten.
- Integration von Luftbildern, um die Messdaten besser interpretieren zu können.
- Erfassung aller möglichen Fahrbeziehungen in einem Knotenpunkt.
- Angleichung der Höhenpläne der Achsen dieser Fahrbeziehungen. Da sich die Achsen im Knotenpunkt überschneiden, sollte es keine Versätze in den Höhenplänen der Achsen geben. Wenn nötig, kann für diesen Schritt im Rahmen einer ausreichenden Genauigkeit von den zugrundeliegenden Messdaten abgewichen werden.
- Verknüpfung der erfassten Knotenpunkte mit Streckenabschnitten, die mit dem bisherigen Verfahren mit Messplan erfasst wurden.
- Für einzelne Streckenabschnitte soll es möglich sein, Querschnittsinformationen zu erfassen. Diese müssen mindestens die Anzahl der Fahrstreifen sowie gegebenenfalls auch die Breite einzelner Fahrstreifen beinhalten. Dazu ist zu überprüfen, inwieweit die vorliegenden Messdaten eine Bestimmung dieser Information ermöglichen. Besonderes Augenmerk ist auf Bereiche mit Fahrstreifenadditionen und -Subtraktionen (vor allem im Knotenpunktbereich) zu legen.
- Verfahrensentwicklung zur fahrstreifenbezogenen, logischen Verknüpfung aller Streckenabschnitte und Knotenpunktverbindungen mit Hilfe eindeutiger IDs. Im Hinblick auf eine Weiterverarbeitung der Messplan-Ergebnisse sind hierbei die Spezifikationen des OpenDRIVE-Formats zu berücksichtigen.
- Für eine Nutzung der Messplan-Ergebnisse muss die (erweiterte) Datenstruktur in das OpenDRIVE-Format überführt werden.

Bei der Implementierung neuer Funktionalitäten müssen folgende Anforderungen beachtet werden:

- Aktuelle Systemvoraussetzungen von Messplan 1.0.0 sollen beibehalten werden.
- Die Software-Umsetzung hat innerhalb der aktuellen Messplan-Umgebung zu erfolgen (Benutzeroberfläche, Programmiersprache).
- Die Weiterentwicklung hat unter der GNU GPL 2.0 Lizenz zu erfolgen.

Im Rahmen der Erweiterung der Software Messplan 1.0.0 soll deren Architektur analysiert und mit dem Model-View-Controller Architekturmuster verglichen werden mit dem Ziel, problematische Abhängigkeiten zu identifizieren. Außerdem sollen die durch die neue Funktionalitäten beeinflussten Komponenten und Schnittstellen hinsichtlich ihrer Erweiterbarkeit untersucht werden. Für die Analyse wird das Softwarewerkzeug ConQAT [2] verwendet.

1.4 Aufbau der Diplomarbeit

Im Kapitel *Grundlagen* werden relevante Technologien, Methoden und Standards aus dem Themenkomplex dieser Diplomarbeit vorgestellt und erklärt.

Das Kapitel *Analyse* beschreibt den Ist-Zustand der Software Messplan 1. Erläutert werden die interne Softwarearchitektur sowie die Benutzeroberfläche. Schließlich wird der Zustand des Programmcodes bewertet im Hinblick auf die durchzuführenden Erweiterungen.

Im Kapitel *Entwurf* werden die Erweiterungen, die an Messplan vorgenommen werden, beschrieben. Insbesondere wird die neue Benutzeroberfläche skizziert und die Änderungen am Datenmodell dokumentiert.

Das Kapitel*Resultat* beschreibt das fertige Softwareprodukt Messplan 1.1. Es erläutert die Bedienung der neuen Oberfläche und die neu implementierte Funktionalität.

Im Kapitel *Fazit und Ausblick* schließlich wird das Resultat mit der Zielsetzung verglichen und die Möglichkeit zur Weiterentwicklung diskutiert.

Kapitel 2

Grundlagen

2.1 Überblick

Dieses Kapitel gibt einen Überblick über die Themengebiete, die von dieser Diplomarbeit berührt werden. Es werden die benutzten Koordinatensysteme und Dateiformate vorgestellt, Konzepte zur Trassierung und Modellierung von Straßennetzen werden erläutert, und schließlich wird ein Überblick über die Funktionen der Software Messplan 1.0 gegeben.

2.2 .NET und C#

.NET ist eine Software-Technologie von Microsoft, die eine Umgebung für die Ausführung von Anwendungen bereitstellt. Darüber hinaus definiert sie mehrere Programmiersprachen, mit denen Anwendungen für diese Umgebung entwickelt werden können.

Der Kern von .NET ist die *Common Language Infrastructure*, die von der International Organization for Standardization standardisiert ist, siehe [3]. Sie definiert die *Common Intermediate Language*, die den Befehlssatz der Umgebung darstellt. Compiler für .NET-Programmiersprachen geben diese Common Intermediate Language aus.

Neben Microsofts eigener Implementierung existieren noch mehrere weitere Implementierungen. Bedeutend ist vor allem die freie Implementierung *Mono*, die von der Firma Xamarin entwickelt und bereitgestellt wird. Mono ist als .NET-Implementation für Linux und MacOSX verbreitet. Ferner bietet Xamarin proprietäre Implementationen für die mobilen Betriebssysteme iOS und Android an.

Unter den Programmiersprachen für die .NET-Umgebung sind vor allem C# und VisualBasic.NET prominent. Die im Rahmen dieser Diplomarbeit weiterzuentwickelte Software ist größtenteils in C# geschrieben und wird auch in C# weiterentwickelt.

2.3 Geodätische Koordinatensysteme

Da die Erdoberfläche gekrümmt ist, braucht es spezielle Koordinatensysteme, um Punkte auf ihr und deren Relationen zueinander zu beschreiben. Zwei dieser Systeme sind für diese Arbeit relevant:

2.3.1 WGS 84

Das *World Geodetic System* [6] ist ein sphärisches Koordinatensystem, das Positionen auf der Erdoberfläche mittels Längen- und Breitengraden definiert. Die aktuelle Revision wurde 1984 veröffentlicht und 2004 zuletzt überarbeitet. Sie wird mit *WGS 84* abgekürzt.

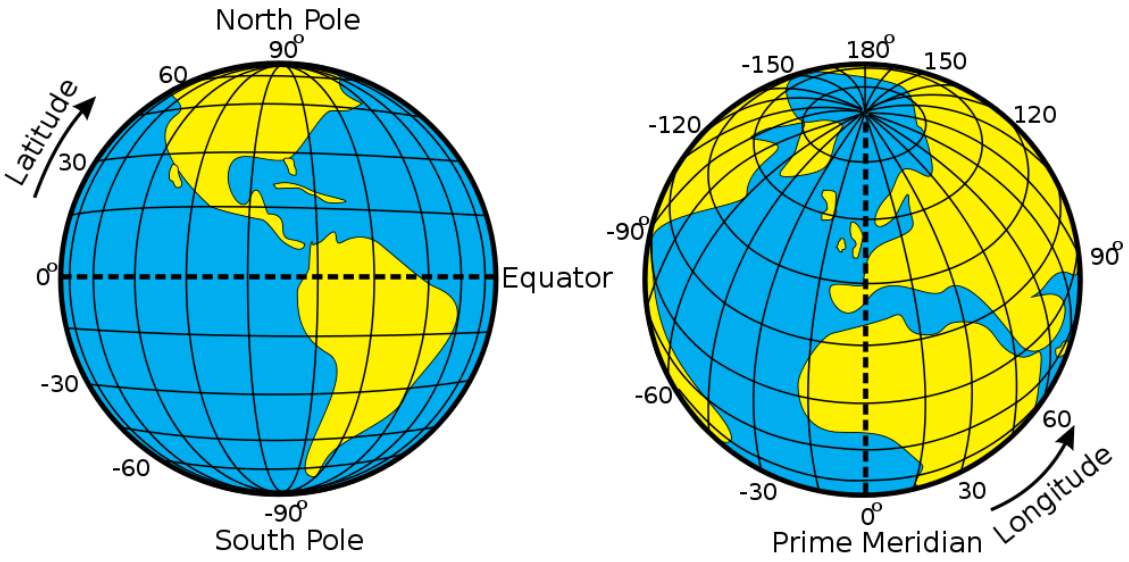


Abbildung 2.1: Visualisierung von Längen- und Breitengrad
Der Äquator definiert den Breitengrad (im Bild: *Latitude*) 0. Der Längengrad (im Bild: *Longitude*) 0 ist durch den vom International Earth Rotation and Reference Systems Service (IERS) definierten *International Reference Meridian* festgelegt.

Höhenangaben sind aufgrund der tatsächlichen Form der Erde nichttrivial. Zunächst existiert ein Referenzellipsoid, der die Form der Erde bestmöglich annähert. Es handelt sich dabei um einen Rotationsellipsoid, der an den Polen einen geringeren Durchmesser hat als am Äquator.

Zusätzlich wird ein sogenannter *Geoid* benutzt, der die theoretische Höhe des Meeresspiegels relativ zum Referenzellipsoid definiert. Diese wird von der ungleichen Masseverteilung in der Erde beeinflusst; die Störung durch die Gezeiten wird herausgerechnet.

Auf Basis dieses Geoids können Höhenangaben „über / unter dem Meeresspiegel" gemacht werden. Der Geoid in der aktuellen Form wurde 1996 als *Earth Gravitational Model 1996* (EGM96) definiert und ist seitdem Bestandteil von WGS 84.

WGS ist das Referenzkoordinatensystem für das Global Positioning System (GPS).

2.3.2 UTM

Das *Universal Transverse Mercator* (UTM) Koordinatensystem [7] ist im Gegensatz zu WGS 84 ein zweidimensionales kartesisches Koordinatensystem. Ein kartesisches Koordinatensystem vereinfacht Entfernungsberechnungen und ermöglicht die einfache Modellierung von Objekten auf der Erdoberfläche wie etwa Straßen.

Um die Erdoberfläche auf ein zweidimensionales Koordinatensystem abzubilden, benötigt man eine Projektion. UTM teilt die Erde in 60 Bänder auf, die jeweis von zwei Breitengraden begrenzt werden, die 6 Grad auseinanderliegen. Jedes dieser Bänder wird nun auf einen Zylinder projeziert, der so angelegt wird, dass er den zentralen Breitengrad der Zone (den sogenannten Meridian) berühren würde; der Radius wird jedoch um den Faktor 0.9996 skaliert, sodass der Zylinder das Band an zwei Linien durchstößt, siehe Abbildung 2.2.

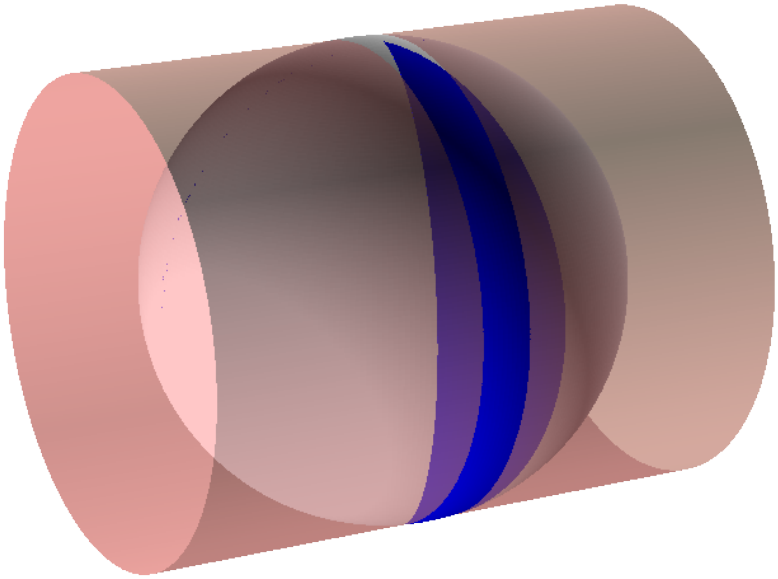


Abbildung 2.2: Transversale Mercator-Projektion
Das blau hervorgehobene Band wird auf den roten Zylinder projeziert. An den beiden Linien, wo der Zylinder das Band durchstößt, findet keine Verzerrung statt.

Als Grundlage für den Radius des Zylinders wird der Referenzellipsoid von WGS 84 genommen. Das Band wird nun orthogonal auf die Zylinderebene projeziert. Der aufgerollte Zylinder ergibt das zweidimensionale Koordinatensystem, in dem alle Positionen auf dem aktuellen Band Dargestellt werden können. Die Projektion erhält die Winkel zwischen Linien, verzerrt aber die Abstände zwischen Punkten.

2.4 Straßennetze

2.4.1 Trassierung

Die Planung, Anlage und Erweiterung von Straßennetzen ist ein komplexes Thema, das verschiedene Aspekte und Teilgebiete umfasst. Für diese Diplomarbeit interessant ist im Wesentlichen die Linienführung in Straßennetzen, spezifisch in der Bundesrepublik Deutschland. Für die rechtlichen Grundlagen siehe [4].

Hinsichtlich der Linienführung lassen sich Straßen in zwei Kategorien unterteilen:

- Innerortsstraßen werden üblicherweise nach fahrgeometrischen Gesichtspunkten geplant. Dabei spielen die Randnutzung, die Anbindung an andere Verkehrswege, die Erschließung des Stadtgebiets und auch Aufenthaltsmöglichkeiten eine große Rolle.
- Straßen, die hauptsächlich eine Verbindungsfunktion haben, werden dagegen nach fahrdynamischen Gesichtspunkten entworfen. Hierbei handelt es sich um Außerortsstraßen, bei denen durch den Wegfall von Aufenthalts- und Erschließungsfunktion andere Aspekte in den Vordergrund treten.

Innerortsstraßen sind üblicherweise kurvenarm und bestehen aus kurzen Straßenabschnitten, die durch nah beieinanderliegende Knotenpunkte voneinander abgegrenzt werden. Für die Rekonstruktion in MessPlan weitaus anspruchsvoller sind die Außerortsstraßen. Diese werden nach folgenden Gesichtspunkten entworfen:

- **Sichere Befahrbarkeit**: Die Straße muss einem Fahrer, der sie bei Nässe mit der angesetzten Entwurfsgeschwindigkeit befährt, ein festgelegtes Mindestmaß an Sicherheit bieten - dies umfasst etwa eine ausreichende Sichtweite, ausreichende Fahrbahnbreite, einfache Erkennbarkeit des Straßenverlaufs und die Einhaltung der Grenzen der fahrmechanischen Sicherheit (beispielsweise Fliehkräfte in Kurven).
- **Verkehrsqualität**: Hierunter fallen vor allem ausreichende Sichtverhältnisse für Überholungen und Minimierung von Überholbedürfnissen (etwa durch Vermeidung von Steigungen).
- **Schonung vorhandener Werte**: Die Straße sollte ins Landschaftsbild eingepasst werden. Natur- und Kulturstätten sollten unberührt bleiben, insbesondere, wenn sie nicht wiederherstellbar sind. Auch gilt es, Lärm- und Schadstoffimmissionen zu minimieren.
- **Witterungsunabhängige Funktionsfähigkeit**: Hierbei spielen Windrichtung, die Nähe zu Gewässern und ähnliche Faktoren eine Rolle.
- **Wirtschaftlichkeit**: Neben den Baukosten spielen auch die Betriebs-, Unterhalts- und Zeitkosten eine Rolle. Diese werden etwa durch den Baugrund und die Notwendigkeit von Geländeeinebnungen sowie Brücken- oder Tunnelbau beeinflusst.

Basierend auf diesen Gesichtspunkten wird die Straße als Folge von geometrischen Formen entworfen. Dies sind im Wesentlichen:

- **Geraden**: Straßenabschnitte mit einer konstanten Krümmung von 0.
- **Kreisbögen**: Straßenabschnitte mit eine konstanten Krümmung ungleich 0.
- **Übergangsbögen**: Straßenabschnitte mit variabler Krümmung. Diese werden mathematisch mit Klothoiden beschrieben und verbinden zwei Abschnitte mit unterschiedlicher Krümmung.

2.4.2 Modellierung

Es gibt verschiedene Ansätze zur Modellierung von Straßennetzen mit unterschiedlichen Zielsetzungen. Allen gemeinsam ist die Abbildung die Grundstruktur des Netzes bestehend aus einzelnen Strecken, die durch Knotenpunkte verknüpft werden. In den meisten Fällen wird die geografische Lage der Strecken modelliert. Welche weiteren Eigenschaften modelliert werden, hängt von dem Zweck des Modells ab. Mögliche Zusatzinformationen wären etwa:

- Für Routenplanung und Navigationszwecke:
  - Erlaubte Fahrtrichtungen auf Straßen
  - Erlaubte Abbiegerelationen an Kreuzungen
  - Gewichtung von Kanten (zum Beispiel erlaubte Höchstgeschwindigkeit)
  - Spuranzahl von Straßen
- Für Erhaltungszwecke:
  - Straßenbelag
  - Baulicher Zustand
  - Relevante Umweltbedingungen, die Einfluss auf den Zustand haben
  - Unterhaltungszuständigkeiten (siehe Abbildung 2.3)
- Für Simulationszwecke:
  - Fahrspuren, die für die einzelnen Abbiegerelationen auf Kreuzungen verwendet werden sollen
  - Genaue Lage der Straße inklusive Längs- und Querneigung
  - Modelle der Umgebung (etwa 3D-Modelle von Häusern, Höhenprofil der Landschaft etc.)

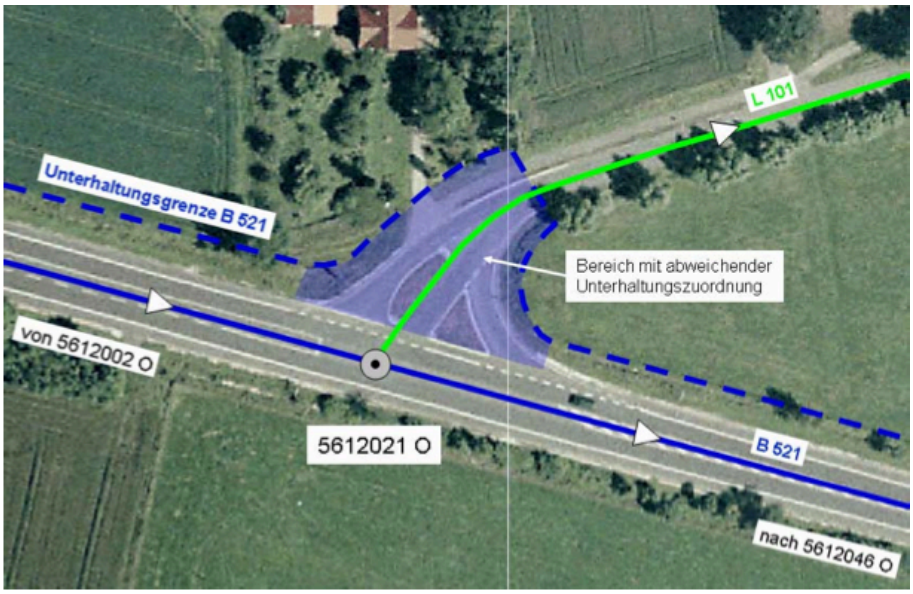


Abbildung 2.3: Modellierung einer Kreuzung von Straßen mit unterschiedlicher Unterhaltungszuständigkeit, entnommen aus [8]

Selbstverständlich können Zusatzinformationen für verschiedene Zwecke benutzt werden, die hier aufgeführte Zuordnung ist nur ein Beispiel.

Unterschiede können auch darin gemacht werden, wie die Knotenpunkte zwischen Straßen modelliert werden. Abbildung 2.4 zeigt ein Straßennetz und hebt die erfassten Netzknoten durch

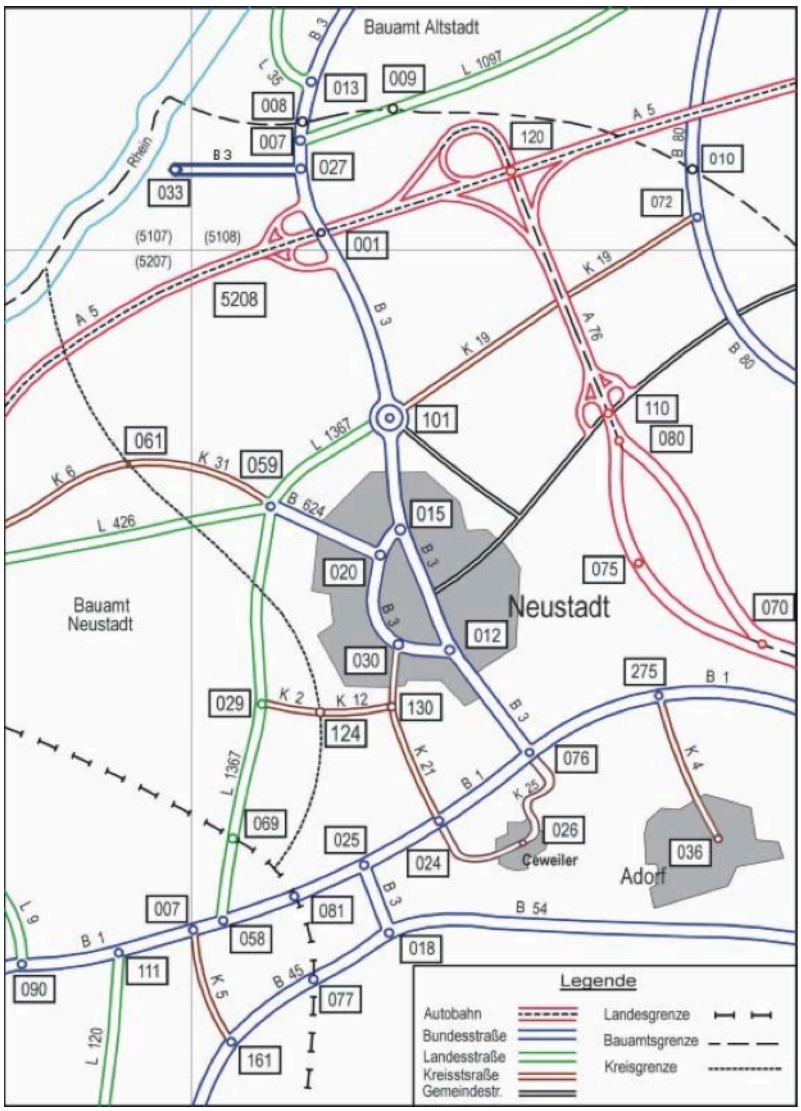


Abbildung 2.4: Modellierung eines Straßennetzes, entnommen aus [9]

numerierte Kreise hervor. Wie man etwa an Netzknoten 120 sieht, werden auch komplexe Strukturen (hier: ein Autobahndreieck) als ein Knoten modelliert. Statt dessen könnte ma auch eine Menge von Knoten modellieren, die alle Abzweigungen und Einmündungen enthält.

In dieser Diplomarbeit liegt der Schwerpunkt der Modellierung auf simulationsrelevanten Aspekten des Straßennetzes. Die Straßen sollen möglichst wirklichkeitsgetreu abgebildet werden. Damit soll es möglich sein, das erfasste Straßennetz möglichst authentisch in einen Fahrsimulator zu laden und dort abzufahren.

2.5 OpenDRIVE

OpenDRIVE [1] ist ein Dateiformat, das von der VIRES Simulationstechnologie GmbH entwickelt wurde, um die proprietären Dateiformate verschiedener Fahrsimulatorhersteller abzulösen. Das Format ermöglicht die Modellierung eines Straßennetzes in einer Weise, dass es von einem Fahrsimulator geladen und zur Simulation der Fahrt in einem Fahrzeug über die Strecke benutzt werden kann.

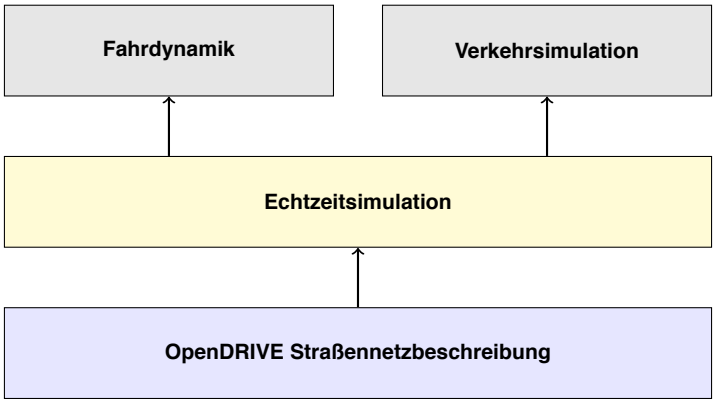


Abbildung 2.5: Schema der Nutzung von OpenDRIVE zur Fahrsimulation

2.5.1 Koordinatensysteme

OpenDRIVE benutzt zunächst ein rechtshändiges Koordinatensystem, in dem die Koordinatenachsen wiefolgt definiert sind:

- x West (negativ) - Ost (positiv)
- y Süd (negativ) - Nord (positiv)
- z Unten (negativ) - Oben (positiv)

Weiterhin wird für Positionen relativ zur Straßenachse ein rechtshändiges Koordinatensystem verwendet mit folgenden Achsen:

- s In XY-Richtung der Straße (die Neigung der Straße wird ignoriert)
- t Orthogonal zur Straßenrichtung; liegt auf der Querneigungsebene der Straße; nach links positiv
- h Nach oben positiv, wobei „oben" in diesem Zusammenhang von der Querneigung der Straße abhängt.

Abbildung 2.6: Ausrichtung des Koordinatensystems auf der Straßenachse

Schließlich gibt es ein lokales Koordinatensystem, das an jeden Punkt mit jeder Orientierung angelegt werden kann und dort diese Richtungen definiert:

- u Nach vorne
- v Nach links
- z Nach oben

2.5.2 Straßenmodellierung

Straßen in OpenDRIVE haben eine oder mehrere Fahrspuren. Jede Fahrspur hat eine Fahrtrichtung. Eine Straße kann einen oder mehrere Abschnitte haben, in denen die Anzahl der Spuren konstant ist. Ändert sich die Spurenanzahl, muss dafür im Modell ein neuer Abschnitt angelegt werden.

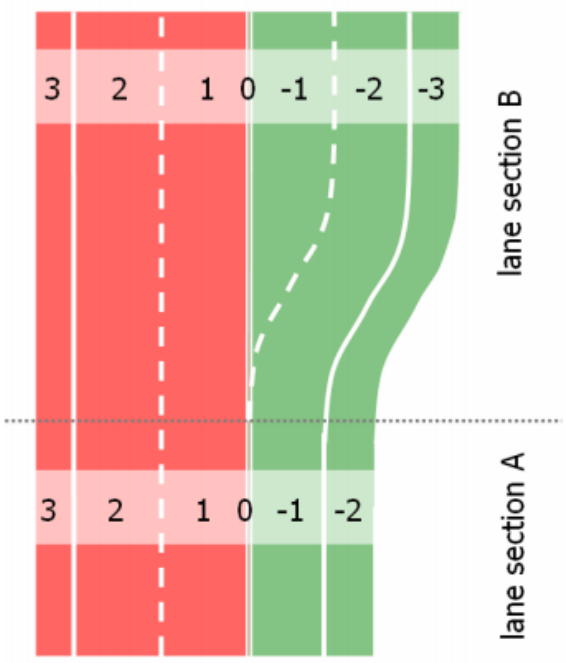


Abbildung 2.7: Änderung der Anzahl Spuren einer Straße in OpenDRIVE

Als mögliche Trassierungselemente, aus denen Straßenabschnitte aufgebaut werden können, stehen zur Verfügung:

- Gerade
- Kreisbogen
- Spirale (Kurve, in der die Krümmung sich linear ändert)
- Kubisches Polynom

Die Parameter dieser Trassierungselemente werden im für jedes Trassierungselement in oben beschriebenen lokalen Koordinatensystem angegeben. Die Aneinanderreihung von Trassierungselementen ergibt den Straßenverlauf.

2.5.3 Netzmodellierung

Straßen können in OpenDRIVE durch eine Vorgänger-Nachfolger-Relation verknüpft werden. Auf Kreuzungen können Straßen mehrere Vorgänger beziehungsweise Nachfolger haben. Kreuzungen werden als Polygon in der XY-Ebene definiert, in welches die sich kreuzenden Straßen einmünden (siehe Abbildung 2.8). Innerhalb des Polygons werden alle möglichen Relationen zwischen den einmündenden Straßen definiert.

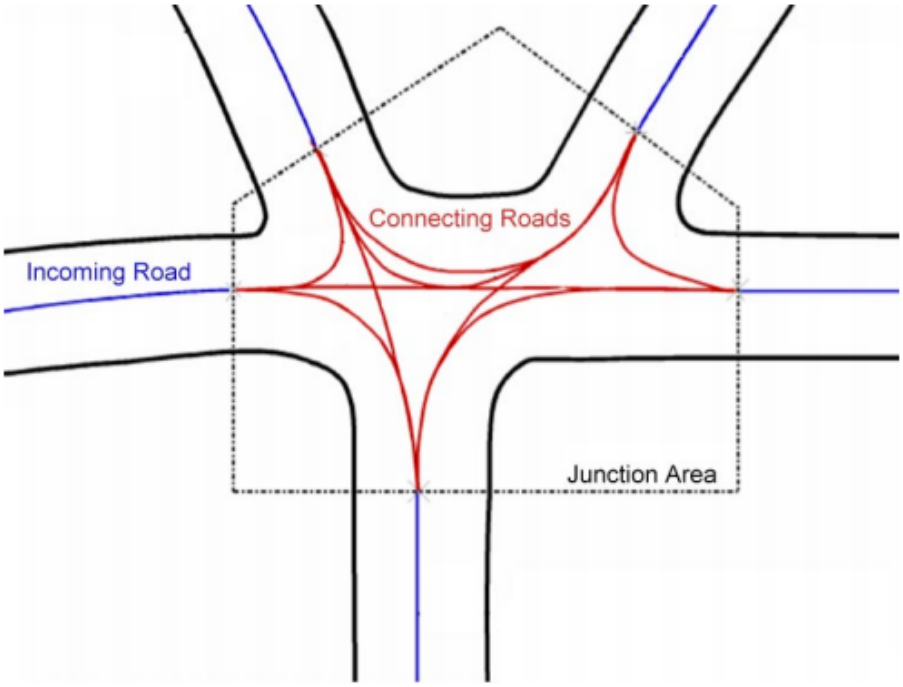


Abbildung 2.8: Kreuzungsmodellierung in OpenDRIVE

Das dargestellte Polygon ist in einer OpenDRIVE-Instanz selbst nicht existent. Die Kreuzung wird ausschließlich durch die angegebenen Verknüpfungspunkte definiert. Insofern ist es nicht notwendigerweise so, dass alle Elemente in dem dargestellten Kreuzungspolygon auch Teil der Kreuzung sind.

Die einzelnen Relationen zwischen den Verknüpfungspunkten werden als Pfade dargestellt, die wie Straßen aus Trassierungselementen bestehen. Allerdings besitzt jeder Kreuzungspfad nur eine Fahrspur und ist damit richtungsspezifisch. Für die Hin- und Rückrichtung zwischen zwei Verknüpfungspunkten müssen zwei unabhängige Pfade angelegt werden.

An den Verknüpfungspunkten kann definiert werden, welcher Pfad auf welche Fahrspur der anknüpfenden Straße übergeht. So kann beispielsweise eine Linksabbiegespur mit dem Pfad verknüpft werden, der zum in Fahrtrichtung links gelegenen Verknüpfungspunkt führt.

2.6 Model-View-Controller

Model-View-Controller (MVC) [13] ist ein Softwarearchitekturmuster. Es wird verwendet, um Software mit grafischer Benutzeroberfläche zu entwickeln. Dabei wird die Software in drei Komponenten unterteilt: Model, View und Controller.

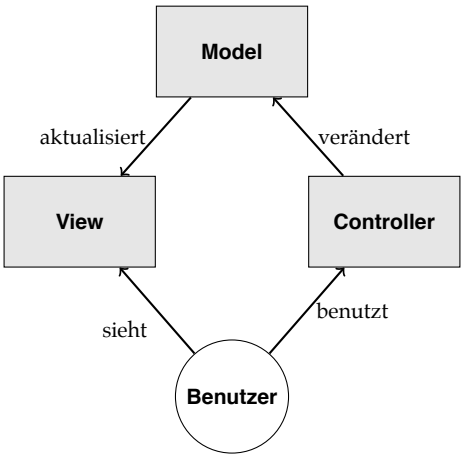


Abbildung 2.9: Model-View-Controller Architekturmuster

Das *Model* implementiert die Datenhaltung. Der *View* visualisiert die Daten für den Benutzer. Der *Controller* nimmt Eingaben vom Nutzer entgegen und führt daraus resultierende Änderungen am Model durch. Diese werden vom Model wiederum an den View weitergereicht, der die Visualisierung entsprechend verändert.

Als Architekturmuster gibt MVC einen groben Anhaltspunkt, wie man eine Softwareanwendung, die eine grafische Benutzeroberfläche umfasst, strukturieren kann. In diesem Dokument werden die Begriffe *Model*, *View* und *Controller* häufig verwendet, um Komponenten oder Funktionalität der beschriebenen Software in das MVC-Muster einzuordnen oder Abweichungen aufzuzeigen.

2.7 GTK

GTK+ [14] ist eine Softwarebibliothek, die Komponenten für grafische Benutzeroberflächen bereitstellt (auch bekannt unter der Bezeichnung *Widgets*). Sie wurde ursprünglich für das GNOME Image Manipulation Program (GIMP) unter dem Namen GIMP Toolkit (GTK) entwickelt. Das Toolkit wurde nach der ersten Version noch einmal komplett neu geschrieben auf der Basis von objektorientierten Paradigmen und in GTK+ umbenannt. In dieser Diplomarbeit wird statt GTK+ der Lesbarkeit halber das Kürzel GTK verwendet.

Die einzelnen Widgets werden als Klassen im objektorientierten Sinn bereitgestellt. Widgets können entweder ausschließlich zur Anzeige von Daten dienen oder auch Interaktion ermöglichen. Im Hinblick auf MVC werden in interaktiven Widgets View und Controller vereinigt. Bei komplexen Widgets werden auch Schnittstellen zum Model vordefiniert und einfache Implementationen angeboten. So bietet etwa das Widget *TreeView*, das für Tabellen und Baumdarstellungen verwendet wird, zwei Model-Schnittstellen an, *ListStore* und *TreeStore*, die abhängig von der Benutzung des Widgets verwendet werden können.

Die GTK-Schnittstelle für C# namens *GtkSharp* ist Teil des Mono-Projekts von Xamarin und wird als native Benutzeroberflächen-Bibliothek von Mono beworben. Während die API in Mono fest enthalten ist, gibt es für Windows auch ein Softwarepaket, das GtkSharp in Microsofts .NET-Umgebung installiert.

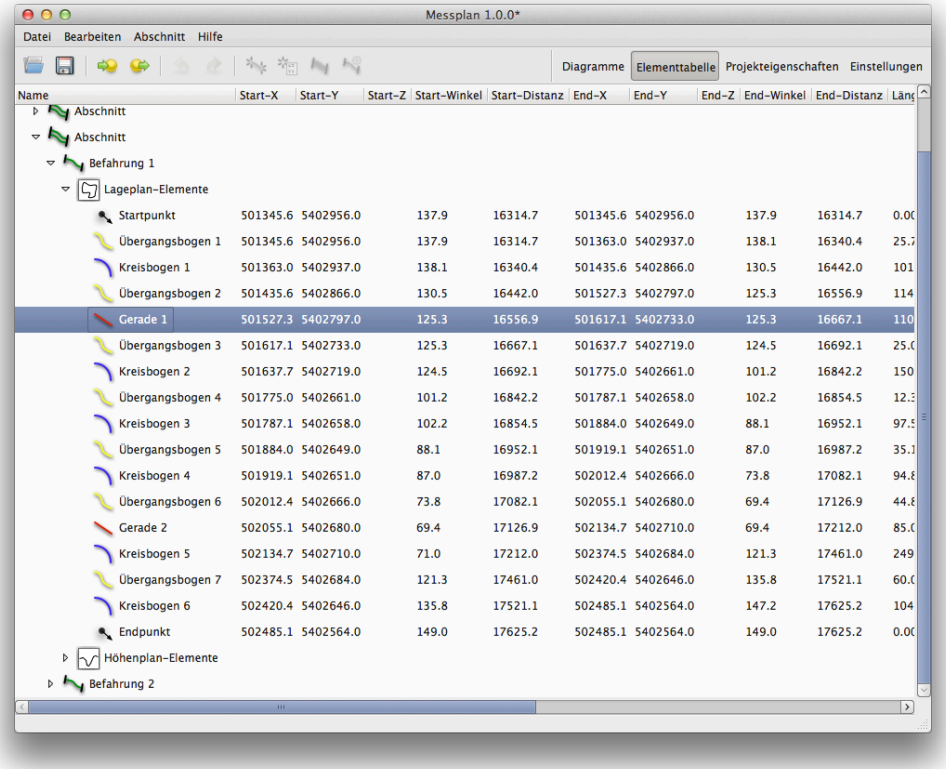


Abbildung 2.10: Beispiel der Verwendung eines TreeView-Widgets in Messplan

2.8 Messplan

2.8.1 Überblick

Messplan ist ein Werkzeug zur Datenaufbereitung. Es arbeitet auf Datensätzen, die von einem Messfahrzeug aufgezeichnet werden. Diese Datensätze enthalten Messpunkte, die die Position, die Orientierung und die zurückgelegte Strecke des Fahrzeugs zu einer bestimmten Zeit enthalten. Diese Messpunkte werden in festen Zeitschritten von Sensoren im Fahrzeug erfasst.

In MessPlan kann man auf Basis dieser Messpunkte die Trassierungselemente der Strecke von einem Algorithmus erkennen lassen und anschließend nachbearbeiten. Dazu wählt man zunächst auf den importierten Messpunkten den Streckenabschnitt aus, mit dem man arbeiten möchte, startet die Elementerkennung und hat dann die Möglichkeit, in einem interaktiven Editor die Elemente nachzubearbeiten.

Schließlich kann man das Resultat in verschiedene Formate exportieren.

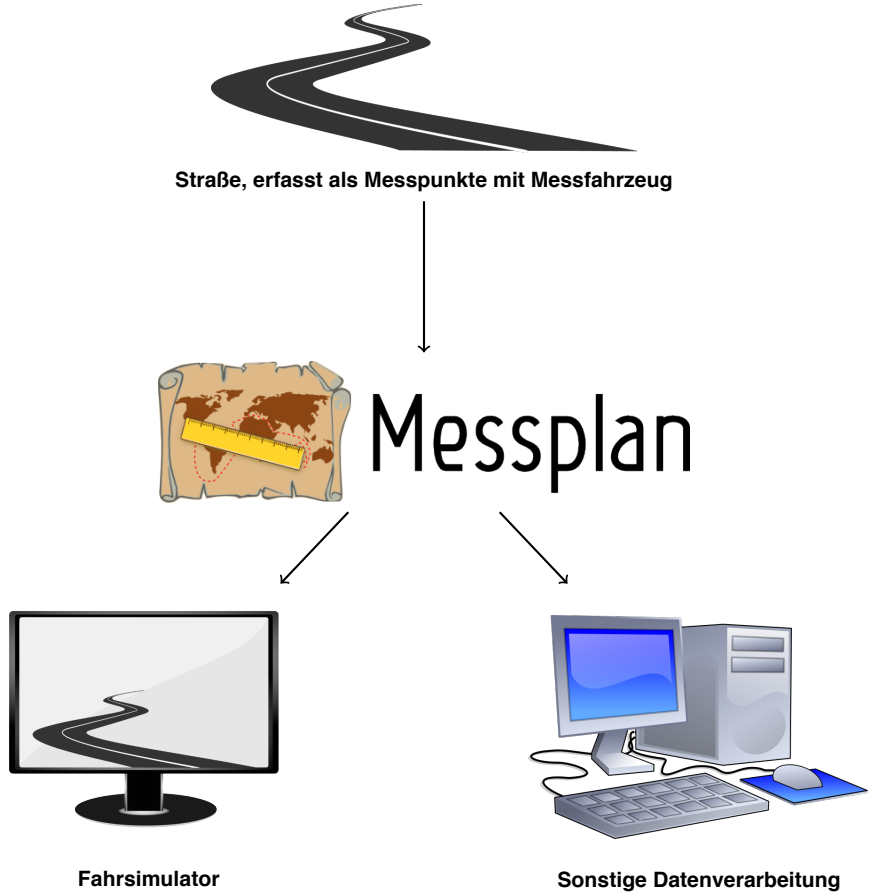


Abbildung 2.11: Anwendungsskizze von Messplan

2.8.2 Entstehungsgeschichte

Messplan war ein Studienprojekt für Softwaretechnik-Studenten am Institut für Straßen- und Verkehrswesen (ISV) der Universität Stuttgart. Der Diplomstudiengang Softwaretechnik schreibt vor, dass die Studenten im Hauptstudium an zwei Studienprojekten teilnehmen, deren Ziel die Entwicklung eines Softwareprodukts ist. Die Studienprojekte sind auf zwei Semester Entwicklungszeit ausgelegt und werden von den Instituten der Universität Stuttgart angeboten. Üblicherweise wird das erste Studienprojekt an einem der informatiknahen Institute absolviert, während das zweite an einem fachfremden Institut durchgeführt wird. Ein Studienprojekt wird von sechs bis zwölf Studenten durchgeführt, wobei jeder typischerweise zehn Stunden pro Woche an dem Projekt arbeitet.

Messplan wurde vom ISV als Studienprojekt B - also als Projekt an einem fachfremden Institut - angeboten. Das Team umfasste umfasste sieben Studenten der Softwaretechnik. Vom Institut waren Kai Tejkl und Sabrina Klötzl als Betreuer, Markus Weise als Kunde sowie Prof. Dr.-Ing. Wolfram Ressel als Prüfer beteiligt. Das Studienprojekt lief vom 1. Mai 2010 bis zum 30. April 2011.

Das Ziel von Messplan war, eine Software zu entwickeln, die die Trassierungsparameter von Straßen, deren Bauunterlagen nicht verfügbar sind, berechnen kann. Außerdem sollte es möglich sein, die Strecke in einem Fahrsimulator abzufahren. Dafür wurde das OpenDRIVE-Format verwendet.

Als Vorgehensmodell für die Entwicklung wurde Scrum [10] gewählt. Nach der Angebotsphase im Mai wurde im Juni mit dem ersten Sprint begonnen. Das Produkt Messplan wurde nach 10 Sprints im April 2011 fertiggestellt.

2.8.3 Umfang

Die Benutzung von Messplan gliedert sich in mehrere Phasen:

Import: Hier werden die Messdaten aus einem unterstützten Format eingelesen. Beim Einlesen werden die Messpunkte vom WGS 84- in das UTM-Koordinatensystem abgebildet. Am Ende des Imports werden die Messdaten im Hauptfenster angezeigt.

Unterstützte Formate sind das NCOM-Format [11] von Oxford Technical Solutions, dem Hersteller, von dem das Messgerät im Messfahrzeug des ISV stammt, und CSV [12], wobei die CSV-Datei von der Struktur her dem NCOM-Format entsprechen muss und sich nur in der Darstellung der Daten von NCOM unterscheidet. Abbildung 2.12 zeigt das Hauptfenster von Messplan während des Importvorgangs.

Abschnittsauswahl: Nun kann der Benutzer auf dem Messdatensatz einen oder mehrere Abschnitte auswählen. Abschnitte sind in diesem Fall Teilstrecken auf einer Straße - wenn im Messdatensatz eine Strecke mehrfach abgefahren wurde, werden alle diese Befahrungen in dem ausgewählten Abschnitt zusammengefasst. Die Erkennung der Befahrungen auf einem Abschnitt wird von Messplan heuristisch auf Basis der Abstände der Messpunkte zueinander durchgeführt. Besteht ein Abschnitt aus mehreren Befahrungen, können die einzelnen Befahrungen gemittelt werden. Hierbei entsteht eine neue Befahrung, die die gemittelten Messpunkte enthält. Abbildung 2.13 zeigt einen Abschnitt, der auf den Messdaten ausgewählt wurde. Ein zweiter Abschnitt ist im Projektexplorer sichtbar, wird in den Diagrammen jedoch nicht angezeigt, weil er nicht ausgewählt ist.

Elementerkennung: Auf einer (eventuell gemittelten) Befahrung können nun Trassierungselemente erkannt werden. Messplan versucht zunächst, die Elemente automatisch zu erkennen und zeigt sie über der Befahrung im Hauptfenster an. Danach kann der Benutzer die Start- und Endpunkte der Elemente manipulieren, Elemente hinzufügen oder löschen, oder die Parameter von Elementen ändern (zum Beispiel Krümmung einer Kurve). Abbildung 2.14 zeigt eine Liste von Elementen an, die in Messplan auf einem Abschnitt erkannt wurden.

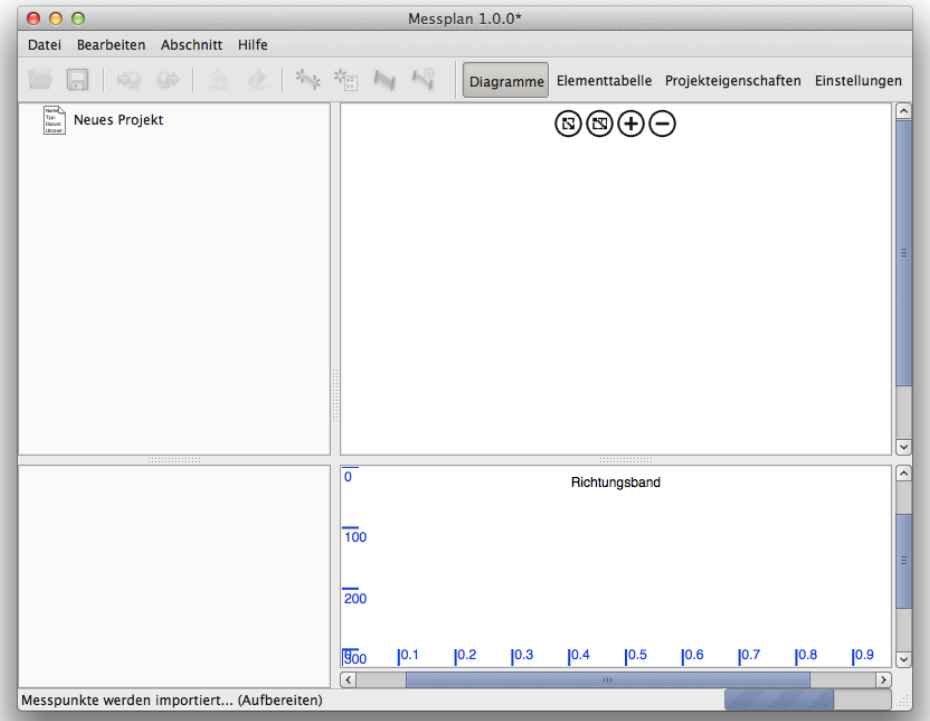


Abbildung 2.12: Importieren von Messdaten in Messplan

Export: Schließlich kann die Befahrung in Form der erkannten Trassierungselemente in verschiedene Formate exportiert werden. Momentan werden neben dem Export der visuellen Darstellung als Bilddatei oder PDF die Formate OpenDRIVE und OKSTRA-XML (Objektkatalog für das Straßen- und Verkehrswesen) angeboten. OKSTRA ist ein deutschlandweiter Standard, der von der Bundesanstalt für Straßenwesen beauftragt und von der interactive instruments GmbH entwickelt wurde. Er ist auf `http://www.okstra.de/` dokumentiert, Messplan verwendet Version 1.015.

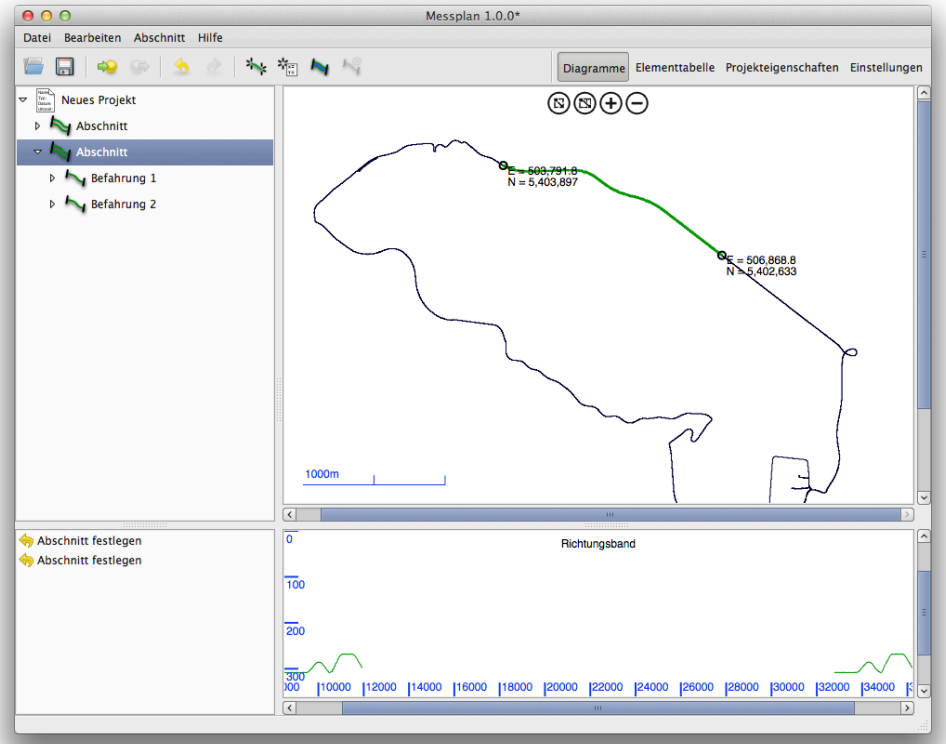


Abbildung 2.13: Abschnittsauswahl in Messplan

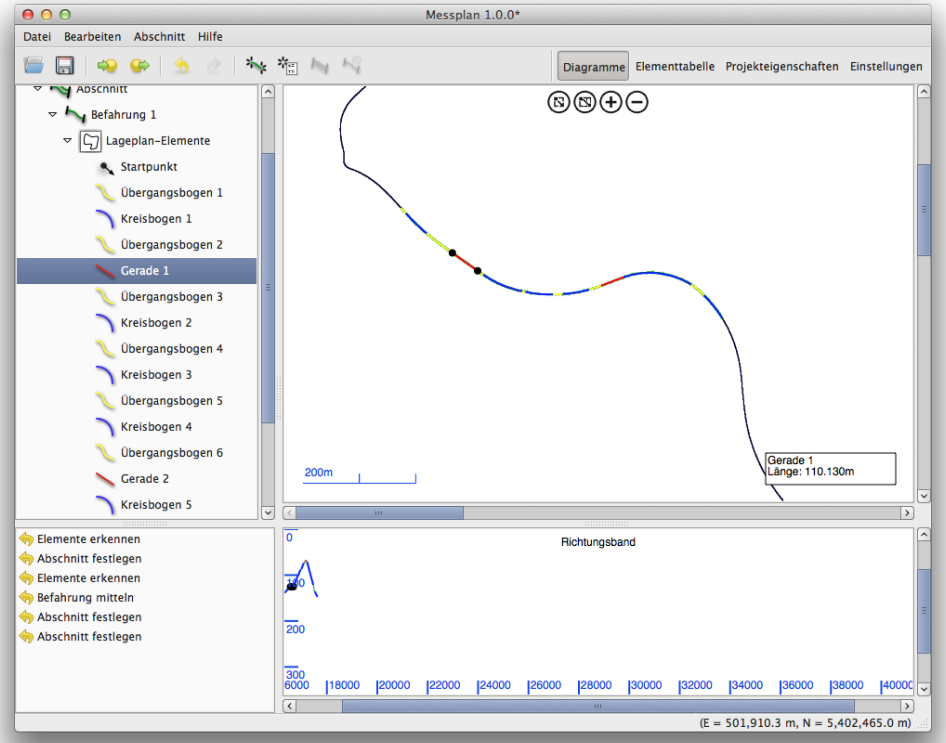


Abbildung 2.14: Erkannte Elemente auf einer Befahrung

Kapitel 3

Analyse

3.1 Anwendungsfälle

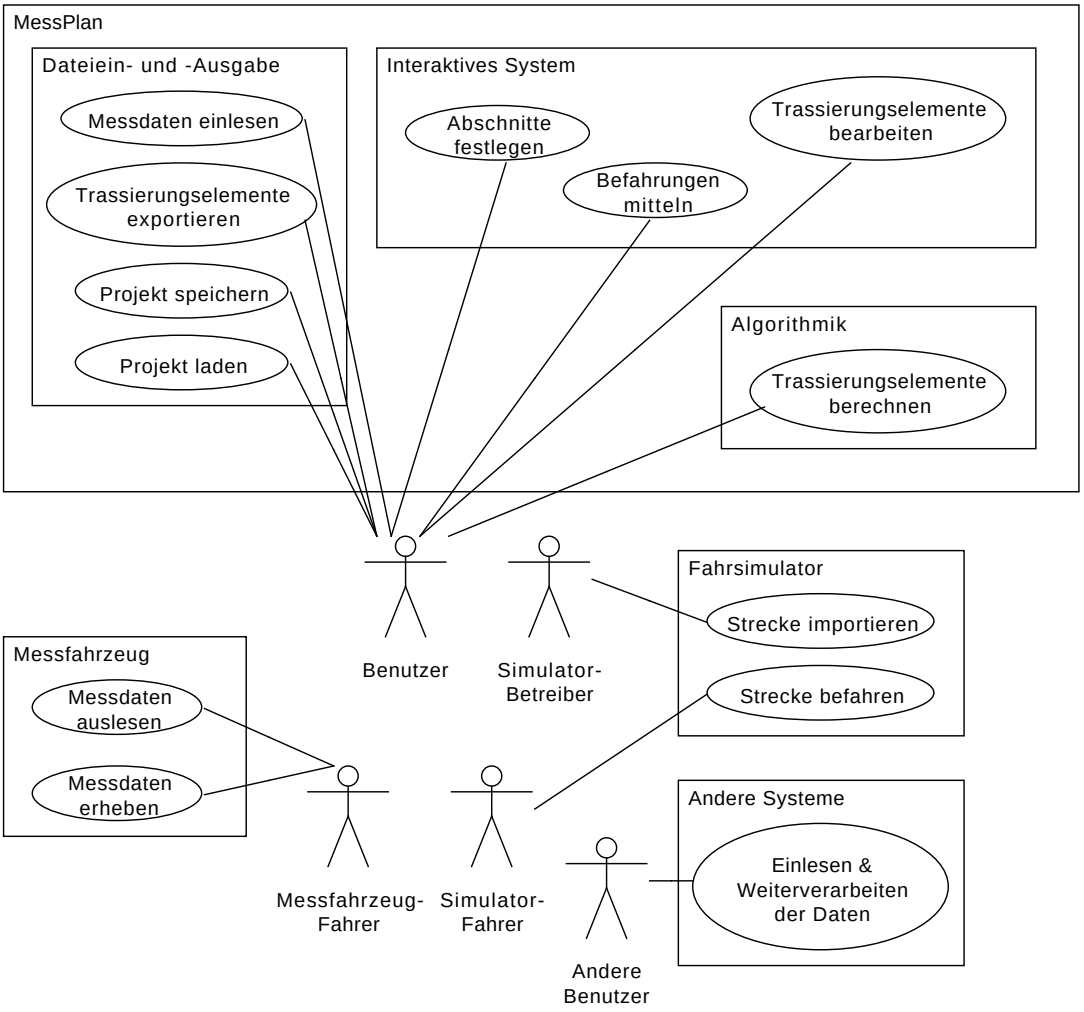


Abbildung 3.1: Anwendungsfälle von Messplan

Messplan ist ein Werkzeug zur Datenverarbeitung und -Aufbereitung. Als solches definiert es einen typischen Arbeitsablauf, dem der Benutzer folgen muss. Die einzelnen Arbeitsschritte sind folgende:

- **Datenerfassung:** Diese kann von einer anderen Person durchgeführt werden. Zur Erfassung der Daten wird die Strecke, auf der mit Messplan gearbeitet werden soll, mit einem Messfahrzeug befahren, üblicherweise mehrfach. Das Fahrzeug zeichnet dabei Koordinaten und Orientierung des Fahrzeugs in festen Zeitabständen auf, was zu einem Datensatz von diskreten Messpunkten führt.
- **Datenimport:** Die erfassten Messdaten werden von Messplan importiert. Dabei werden die GPS-Koordinaten automatisch ins UTM-Koordinatensystem umgerechnet, das als kartesisches Koordinatensystem besser für die visuelle Darstellung und die Verarbeitung der Messdaten geeignet ist. Die Messpunkte werden beim Import in eine hierarchische Baumstruktur eingehängt, die vor allem den Zugriff auf die Datenpunkte und die Darstellung derselben erheblich beschleunigt.
- **Abschittsauswahl:** Der Benutzer kann nun auf dem Lageplan einen Abschnitt definieren, auf dem er arbeiten möchte. Da ein Abschnitt üblicherweise mehrfach befahren wurde, versucht Messplan, alle diese Befahrungen zu erfassen, indem es in der Umgebung der vom Benutzer ausgewählten Anfangs- und Endpunkte nach Messpunkten sucht und dann heuristisch alle Teilmengen in den Messpunkten ermittelt, die eine Befahrung des vom Benutzer ausgewählten Abschnitts darstellen.
- **Mittelung von Befahrungen:** Dieser Schritt ist optional. Der Benutzer kann mehrere Befahrungen mitteln, um Ungenauigkeiten in der Messung auszugleichen. Das Resultat ist eine neue Befahrung, die aus den gemittelten Messpunkten besteht.
- **Elementerkennung:** Die Elementerkennung wird auf einer Befahrung ausgeführt. Der Benutzer kann für die Erkennung verschiedene Parameter konfigurieren, die das Resultat beeinflussen. Die Erkennung selbst läuft dann automatisch durch und erstellt geometrische Streckenelemente, die auf die gewählte Befahrung passen.
- **Elementbearbeitung:** Der Benutzer kann nun die erkannten Elemente verändern. Dafür stehen verschiedene Interaktionsmöglichkeiten zur Verfügung: Einmal die direkte Änderung an Interaktionspunkten in der Darstellung der Elemente, zum Anderen die textuelle Änderung der Elementparameter. Außerdem können Elemente entfernt und hinzugefügt werden.
- **Export:** Schließlich kann das Projekt exportiert werden. Unterstützte Exportformate sind das OpenDRIVE-Format für Fahrsimulatoren sowie in das OKSTRA-Format, das im Straßen- und Verkehrswesen benutzt wird.
- **Projekt speichern und laden:** Das Projekt kann jederzeit in einem XML-basierten Format gespeichert und geladen werden.

Die resultierende Ausgabe kann in verschiedenen Bereichen weiterverwendet werden. Im Zuge dieses Projekts ist vor allem der Export in das OpenDRIVE-Format wichtig.

3.2 Datenfluss

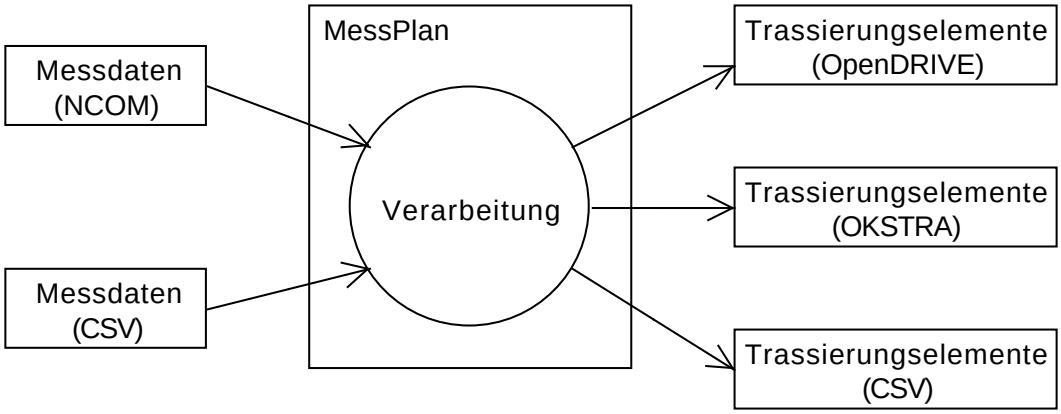


Abbildung 3.2: Datenfluss in der Umgebung von Messplan, Level 0

Die Datenerfassung wird einem Messgerät durchgeführt, das neben GPS-Daten auch die Orientierung des Messfahrzeugs mithilfe eines Gyroskops erfasst. Die Daten können im binären NCOM-Format oder im textbasierten CSV-Format ausgelesen werden. Messplan kann beide Formate importieren.

Beim Export werden nur die geometrischen Streckenelemente exportiert; die zugrundeliegenden Messdaten spielen in den exportierbaren Formaten keine Rolle. Eine Übersicht über die unterstützten Ein- und Ausgabeformate gibt Abbildung 3.2.

Die Verarbeitung der Eingabedaten ist in Abbildung 3.3 dargestellt.

Die rohen GPS-Eingabedaten liegen als geographische Breite, geographische Länge und der Höhe über dem Meeresspiegel gemäß WGS 84 vor. Beim Import werden diese Werte ins UTM-Koordinatensystem umgerechnet, das als kartesisches Koordinatensystem besser zur Darstellung und Verarbeitung der Werte geeignet ist. Auch die Ausgabeformate basieren auf UTM-Koordinaten.

Messplan definiert ein internes Format, in dem ein Projekt gespeichert werden kann. Die zugrundeliegenden Messdaten werden nicht direkt in die Projektdatei geschrieben, sondern lediglich in ihrem Ursprungsformat in denselben Ordner wie die Projektdatei gelegt. Um ein Projekt zu laden, müssen diese Ursprungsdaten verfügbar sein. Außerdem gilt zu beachten, dass von gemittelten Befahrungen nur die Parameter gespeichert werden und die Mittelung beim Laden erneut ausgeführt wird. Dies betrifft nur die Messpunkte der gemittelten Befahrung, nicht die darauf erkannten Trassierungselemente.

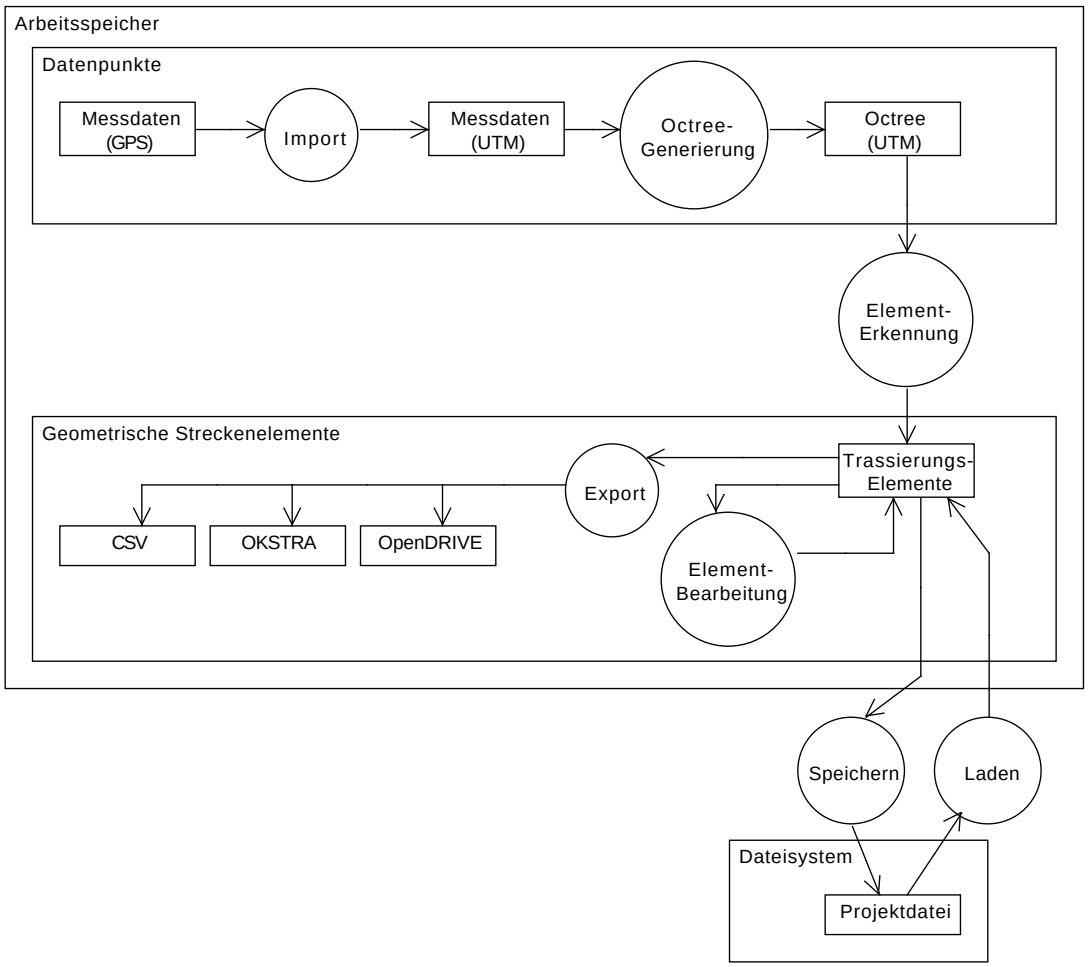


Abbildung 3.3: Datenfluss in Messplan, Level 1

3.3 Analyse mit ConQAT

Bei dem Versuch, die Softwarearchitektur mit ConQAT zu analysieren, traten unvorhergesehene Probleme auf. ConQAT unterstützt zwar laut Handbuch C# als Eingabesprache, aber beim Laden des Programmcodes trat reproduzierbar ein Fehler auf, der dazu führte, dass der Ladevorgang abbrach.

Trotz verschiedener Versuche, die Ursache des Fehlers zu finden, hatte ich keinen Erfolg damit, den Programmcode zu laden. Ich probierte es mit verschiedenen .NET-Versionen, aber alle führten zum selben Fehler. Die Ausgabe von ConQAT beim Auftreten des Fehlers sah wiefolgt aus:

```
 ERROR : No factory found for element Messplan/Backup/Storage/Import/-
CoordinateTransformation.cs
(org.conqat.engine.dotnet.scope.SolutionProjectScope.text-resource-
builder.resource-builder)
```

Derselbe Fehler wurde für jede einzelne Codedatei von Messplan ausgegeben. Offensichtlich hat ConQAT grundsätzlich Unterstützung für .NET, sonst hätte es die Projektdateien von Messplan nicht laden können. Aus diesen hat es die Pfade zu den Quellcodedateien ausgelesen, die dann die oben angeführten Fehler produzierten.

Letztendlich musste ich die Analyse mit ConQAT aufgeben und ohne sie weiterarbeiten.

3.4 Architektur

3.4.1 Überblick

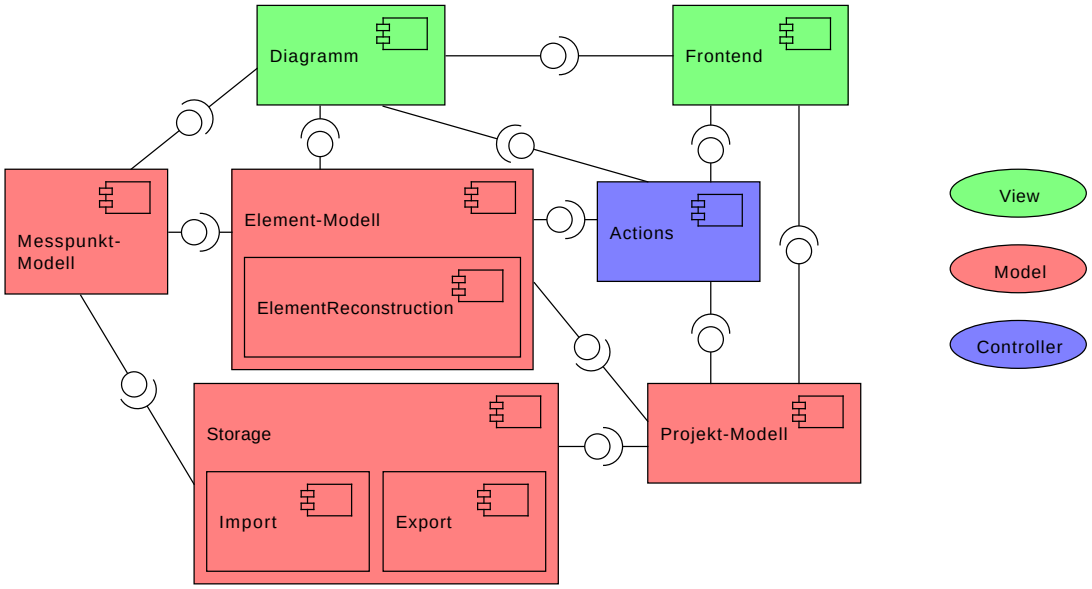


Abbildung 3.4: Komponenten von Messplan

Messplan ist eine interaktive grafische Anwendung. Für die Darstellung der Benutzeroberfläche sind die Komponenten *Frontend* und *Diagramm* zuständig, wobei ersteres auch das Handling von Benutzereingaben übernimmt.

Actions verwaltet die Benutzeraktionen, und ermöglicht es, Aktionen rückgängig zu machen. Die meisten Aktionen, die eines der Datenmodelle ändern, werden über Actions angesprochen.

Es gibt drei typische Modellklassen: Das *Messpunkt-Modell* ist die Datenstruktur, in der die importierten Messpunkte abgelegt werden. Im *Element-Modell* werden die Trassierungselemente, die aus den Messpunkten berechnet werden, gespeichert. Die untergeordnete Komponente *Element-Rekonstruktion* enthält die Algorithmik zur Erkennung der Trassierungselemente aus den Messpunkten. Das *Projekt-Modell* speichert Metadaten des Projektes.

Storage schließlich ist die Komponente für den Dateizugriff, sie speichert und läd Messplan-Projektdateien. Über *Import* werden Messdaten eingelesen. *Export* kann das aktuelle Projekt in verschiedene Formate exportieren.

3.4.2 Komponenten

3.4.2.1 Frontend

Das Frontend implementiert den Großteil der grafischen Benutzeroberfläche. Da es sich bei Messplan um eine Software handelt, die darauf ausgelegt ist, nur ein Fenster zu benutzen, stellt der Frontend eine entsprechende Widget-Hierarchie bereit, in die die einzelnen Ansichten und Steuerelemente der Benutzeroberfläche eingefügt werden. Der Frontend implementiert auch die Logik zum Wechsel zwischen verschiedenen Ansichten. In Abbildung 3.6 sind die zwei Ansichten, zwischen denen das Hauptfenster wechseln kann, dargestellt: Einmal der Willkommensbildschirm, in dem man ein neues Projekt erstellen oder ein vorhandenes öffnen kann, und zum Anderen die Hauptansicht, die wiederum über eine Tableiste verfügt. Die Tableiste kann in der Hauptansicht zwischen verschiedenen Teilansichten wechseln; in der Abbildung ist der Tab *Einstellungen* aktiv.

Die Codestruktur orientiert sich stark an den Mustern, die durch die Benutzung des GUI-Frameworks GTK vorgegeben sind: Jedes Steuerelement ist eine Klasseninstanz, die sowohl ihr Aussehen wie auch die möglichen Benutzerinteraktionen darauf festlegt. Benutzerinteraktionen, nur die Ansicht verändern, werden direkt im Frontend implementiert. Aktionen, die das Modell verändern, werden an die Komponente *Actions* weitergereicht.

Interessant ist die Selektion von Objekten im Projektexplorer (Abbildung 3.5). Der Projektexplorer ist eine Baumansicht, in der die einzelnen Objekte, die momenten im Projekt vorhanden sind, angezeigt werden. Wenn eines oder mehrere Trassierungselemente im Projektexplorer ausgewählt werden, werden die entsprechenden Elemente auch in der Diagrammansicht hervorgehoben. Der Projektexplorer und die Diagrammansicht kommunizieren dabei nicht direkt miteinander, sondern ändern das Selektionsmodell, welches ein Teil des Projektmodells ist (siehe 3.4.2.4).

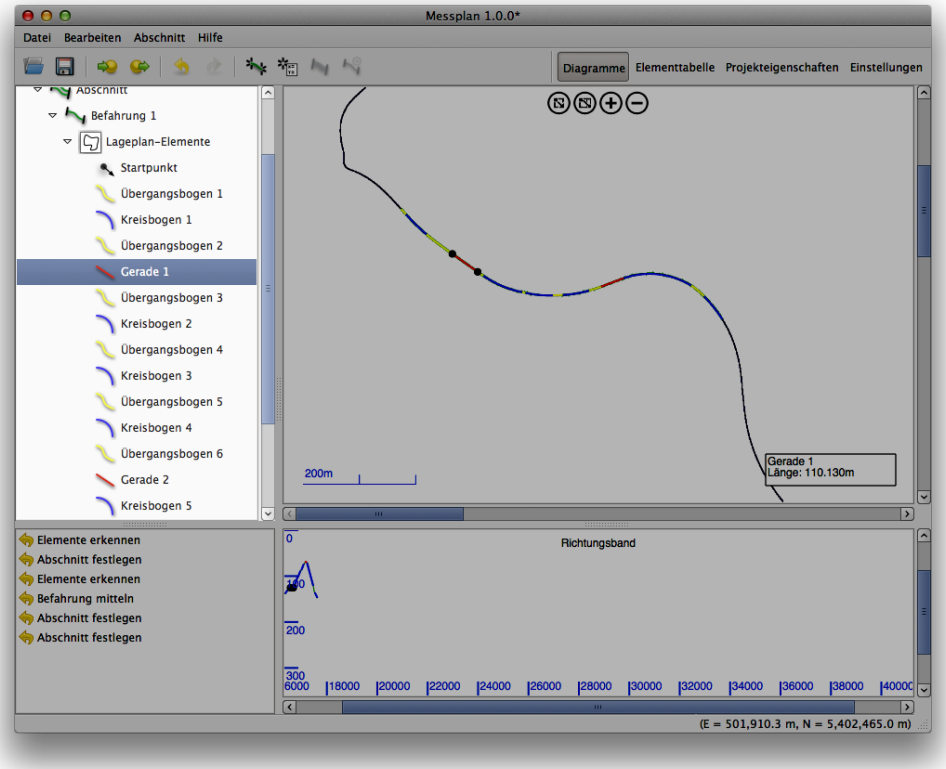


Abbildung 3.5: Projektexplorer

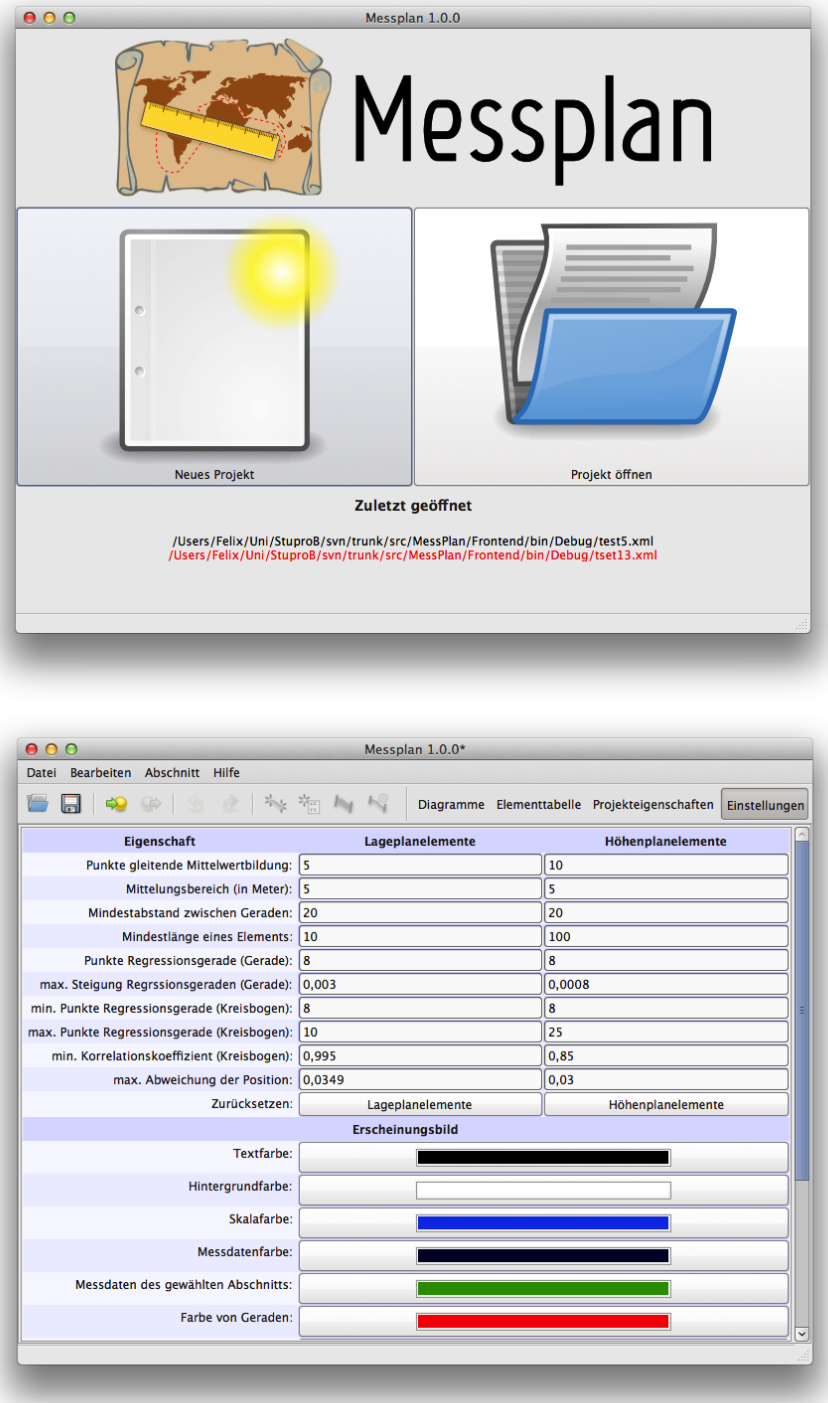


Abbildung 3.6: Verschiedene Ansichten im Hauptfenster.
Oben: Startmenü. Unten: Hauptansicht mit aktivem Tab *Einstellungen*.

3.4.2.2 Diagramm

Das Diagramm-Modul stellt grafische Steuerelemente zur Verfügung, die die Messdaten und die Trassierungselemente in verschiedenen Diagrammtypen darstellen. Die Diagrammtypen unterscheiden sich darin, welche Dimensionen auf ihren Achsen dargestellt werden:

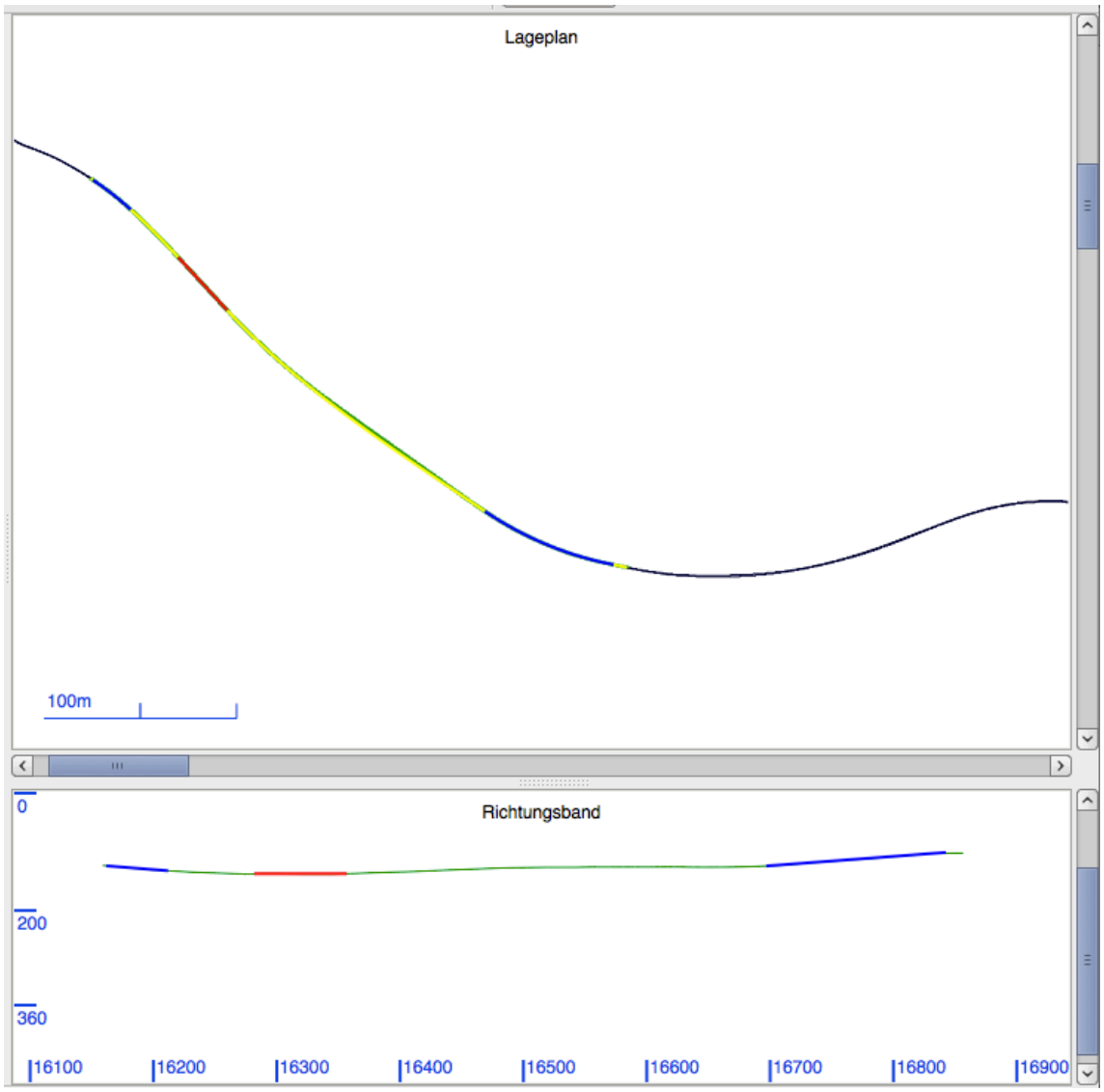


Abbildung 3.7: Lageplan und Richtungsband

- **Lageplan** (X- und Y-Dimension). Dies ist im Prinzip eine orthogonale Projektion der Elemente auf eine horizontale Ebene. Üblicherweise arbeitet der Benutzer mit dieser Darstellung am Häufigsten.
- **Richtungsband** (Richtungs- und Distanzdimension). Hier wird die Fahrtrichtung (in Grad) abhängig von der zurückgelegten Distanz dargestellt. Dieses Diagramm wird zusammen mit dem Lageplan angezeigt. Die Darstellung im Richtungsband eignet sich gut für die manuelle Erkennung und Einpassung von Trassierungselementen. Eine Eigenheit dieses Diagramms ist, dass es theoretisch eine Zylinderform hat - das obere Ende der Skala (0°) ist mit dem unteren Ende (360°) identisch. Entsprechend ist es möglich, dass Elemente das Diagramm nach oben hin verlassen und von unten fortgeführt werden.
- **Höhenplan** (Höhen- und Distanzdimension). Dies ist die Hauptansicht zur Bearbeitung von Höhenplanelementen. Er stellt ein Höhenprofil der befahrenen Strecke dar.

- **Neigungsband** (Neigungs- und Distanzdimension). Hier wird die Neigung des Fahrzeugs abhängig von der zurückgelegten Strecke dargestellt. Das Neigungsband wird zusammen mit dem Höhenplan angezeigt.

Alle Diagrammtypen, die die Distanz als eine ihrer Dimensionen haben, haben das Problem, dass die Folge von Trassierungselementen, die aus den Messpunkten generiert wird, üblicherweise nicht mit der Streckenlänge, wie sie von den Messpunkten angegeben wird, korreliert: Ein Kreisbogen-Trassierungselement etwa hat eine andere Länge als die zugrundeliegenden Messpunkte. Die Abweichung mag gering sein, aber nach mehreren Elementen weicht die zurückgelegte Distanz auf den Trassierungselementen immer weiter von der zurückgelegten Distanz auf dem Abschnitt in den Messpunkten, aus dem ein Trassierungselement generiert wurde, ab. Deshalb kann man in den entsprechenden Diagrammen die Darstellung der Messpunkte gegen die Darstellung der Trassierungselemente verschieben.

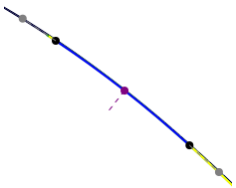


Abbildung 3.8: Änderbare Knotenpunkte an einem Kreisbogen

Die Nachbearbeitung von Elementen ist vor allem grafisch durch das Ziehen von Interaktionspunkten am aktuell auswählten Element implementiert. Die Interaktionspunkten sind der Start- und Enpunkt, sowie bei Kreisbögen noch die Start- und Endrichtung sowie die Mitte des Kreisbogens. Erst, wenn ein Interaktionspunkt wieder losgelassen wird, wird mit der Komponente *Actions* kommuniziert, sodass die Änderung rückgängig gemacht werden kann. Ansonsten ist die Bearbeitung von Trassierungselementen vollständig im Diagramm implementiert.

3.4.2.3 Actions

Das Modul Actions implementiert die meisten Benutzeraktionen, die das Modell verändert (ausgenommen Änderungen, die ausschließlich das Selektionsmodell betreffen, siehe *Projektmodell*). Für jede hier definierte Aktion ist die Durchführung und das Zurücknehmen implementiert, sodass jede Aktion rückgängig gemacht werden kann.

Neben den Aktionen implementiert das Actions-Modul auch ein grafisches Steuerelement, das alle Aktionen anzeigt, die momentan rückgängig gemacht werden können. Die Aktionen werden auf einem Stack verwaltet. Wenn Aktionen rückgängig gemacht werden, können sie danach wiederhergestellt werden, solange keine neue Aktion auf den Stack gelegt wird.

3.4.2.4 Projektmodell

Das Projektmodell implementiert die Datenhaltung für alle projektbezogenen Daten. Dies sind im Einzelnen (gemäß der Benennung im Code):

- `MeasuredDataLocations`: Absolute Dateisystempfade zu den importierten Messdaten. Diese Daten werden nicht in der Projektdatei gespeichert, sondern nur referenziert. Dies ist notwendig aufgrund der großen Datenmenge, die ein Satz Messdaten üblicherweise enthält.
- `DataBuilder`: Speichert den internen Zustand des Werkzeugs zum Messpunkt-Import. Dies ist notwendig, um weitere Messdaten in denselben internen Datensatz zu importieren, der die bereits importierten Daten enthält.
- `Sections`: Die Abschnitte, die der Benutzer angelegt hat. Für diese wird eine `ObservableItemList` verwendet, da Änderungen an und in den Abschnitten in die GUI-Module propagiert werden müssen.

  In den Abschnitten werden die einzelnen Befahrungen angelegt, die wiederum die geometrischen Streckenelemente enthalten können (siehe *Elementmodell*).
- `BaseSlice`: Eine Sicht auf die importierten Messdaten. Außerhalb des Messpunkt-Modells können die Messpunkte nur über die `ISlice`-Klasse gelesen werden; diese stellt eine Sicht auf die Messpunkte dar (näheres siehe *Messpunkt-Modell*).
- `GlobalSelection`: Instanz des Selektionsmodells, siehe dort.
- `metaData`: Metadaten des Projekts. Diese sind privat, der Zugriff darauf erfolgt über Setter- und Getter-Methoden.

Das Projektmodell benutzt das Event-Framework des CLI, um Änderungen im Modell bekannt zu machen. Die Komponenten des Frontends registrieren sich auf diese Events.

3.4.2.5 Selektionsmodell

Das Selektionsmodell implementiert das Markieren von Trassierungselementen durch den Benutzer. Eine Markierungsaktion des Benutzers, die im Frontend entweder im Projektexplorer oder auf einem der Diagramme ausgelöst wird, wird hierher weitergeleitet. Das Selektionsmodell ist der einzige Teil des Projektmodells, der beim Speichern nicht in der Projektdatei abgelegt wird.

3.4.2.6 Messdatenmodell

Aufgrund der üblicherweise recht großen Menge von Eingabedaten muss die Struktur, in der die Datenpunkte abgelegt werden, gewissen Performanzkriterien genügen. Folgende Funktionen müssen eine Laufzeit von deutlich unter einer Sekunde aufweisen, damit der Benutzer mit der Oberfläche effizient arbeiten kann:

- Ausgabe der abgefahrenen Strecke in mehreren Detailstufen zur visuellen Darstellung im Frontend.
- Abbildung eines Koordinatenpunkts auf den nächstliegenden Messpunkt. Dies wird für die Abschnittauswahl per Mausklick benötigt.
- Suche nach Streckenabschnitten, die mehrfach befahren wurden. Dies ist wichtig, um mehrere Befahrungen zu einem Abschnitt zusammenfassen zu können.

Das Messdatenmodell ordnet die Datenpunkte logisch in einem dreidimensionalen Raum an. Die drei Achsen stellen dabei die X-Koordinate, die Y-Koordinate und die zurückgelegte Strecke dar. Dieser Raum wird in Kuben mit einer Kantenlänge von einem Meter unterteilt. Über diesen Kuben wird eine Baumstruktur aufgebaut, die von Struktur und Eigenschaften einem Octree, wie er in der 3D—Computergrafik eingesetzt wird, gleicht. Daher wird das Messdatenmodell im Code und der Dokumentation *Octree* genannt.

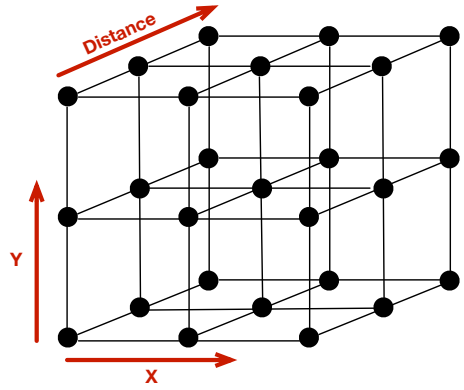


Abbildung 3.9: Gerasterter Datenraum der Messpunkte

3.4.2.7 Elementmodell

Das Elementmodell definiert und speichert die geometrischen Streckenelemente, die der Benutzer auf den Messpunkten erkennen lassen hat. Hier ist auch der Algorithmus für die Elementerkennung implementiert.

3.4.2.8 Storage

Die Storage implementiert die Ein- und Ausgabe aller unterstützten Dateiformate. Für das Einlesen von NCOM-Dateien wird eine externe C-Bibliothek verwendet.

3.5 Bewertung

3.5.1 Architektur

In Abbildung 3.4 ist durch die Farbgebung die Einordnung der einzelnen Komponenten in das Komponenten-Schema, wie es vom MVC-Architekturmuster vorgegeben wird, angedeutet. Allerdings wird das Schema an mehreren Stellen verletzt:

- Die Komponenten Frontend und Diagram implementieren selbt einiges an Funktionalität, das nach dem MVC-Pattern eigentlich in die Controller- Komponente gehört. allerdings ist dies durch die interaktiven Funktionen, die vor allem das Diagramm bietet, nicht vermeidbar.
- Das Selektionsmodell sollte deutlich vom restlichen Datenmodell getrennt werden, da es nur den Zustand des Views speichert.

Die angeführte Problematik des Selektionsmodells ist kein tiefsitzendes Problem. Es ist bereits so, dass die Widgets der Benutzeroberfläche Aktionen, die das Datenmodell ändern, an die Controller-Klasse weitergeben, während Aktionen, die nur die Selektion ändern, nicht über die Controller-Klasse gehen. Insofern müsste nicht allzu viel Code angefasst werden, um das Selektionsmodell auszulagern.

3.5.2 Code

Die größten Probleme von Messplan liegen nicht in der Architektur der Anwendung, sondern in der Codestruktur und in verschiedenen Stellen des Datenmodells. Dass der Code uneinheitlich strukturiert und dokumentiert ist, sei nur am Rande erwähnt - dies lässt sich mit geeigneten Werkzeugen relativ einfach beheben.

3.5.2.1 Datenstrukturen für Punkte

Im Modell werden verschiedene Strukturen für Punkte verwendet, die untereinander nicht kompatibel sind. Es gibt die Interfaces `IPoint`, `IPoint2D` und `IPoint3D`, die für die Messpunkte verwendet werden und so definiert sind, dass man über das jeweilige Interface die einzelnen Datenfelder nur lesen, aber nicht schreiben kann. Die einzelnen Werte werden als Fließkommazahlen mit einfacher Genauigkeit gespeichert. Der Gedanke dahinter war wohl, dass man die Messpunkte, die aus dem Datenmodell kommen, nicht nachträglich modifizieren darf, da nur Referenzen herausgegeben werden. Allerdings könnte man auch einfach eine Kopie der Werte zurückgeben.

Als unmodifizierbare Objekte waren diese Strukturen offensichtlich ungeeignet für die Klassen der Trassierungselemente, die viel mit Koordinaten arbeiten. Daher gibt es eine abstrakte Klasse `AbstractElementPoint`, deren Werte änderbar sind und darüber hinaus mit doppelter Genauigkeit gespeichert werden.

Es ist zumindest fragwürdig, ob man derartig komplexe Strukturen für Messpunkte oder Koordinaten braucht. Im Code führt das dazu, dass etliche simple Berechnungen wie etwa der Abstand zwischen zwei Punkten für jede Art von Punkten mehrfach implementiert sind. Es gibt zwar eine Klasse `MathHelpers`, doch diese wird nicht konsequent verwendet.

3.5.2.2 Übermäßige Verwendung von Events

C# bietet sogenannte Events an: Das sind Felder einer Klasse, zu denen man eine beliebige Anzahl an von Methodenreferenzen (genannt *delegate*) hinzufügen kann. Wenn das Event dann ausgelöst wird, werden alle angehängten Methoden nacheinander aufgerufen.

In Messplan werden diese Events durchgehend verwendet. Prinzipiell lässt sich damit die Idee von MVC, dass das Model den View über Änderungen benachrichtigt, zwar gut umsetzen - das Problem ist allerdings, dass die Events nicht nur in diese Richtung benutzt werden, sondern beispielsweise auch für die Behandlung von Benutzerinteraktionen. An manchen Stellen kann nicht

entschieden werden, ob nun beispielsweise der Modus eines Widgets (gedrückt / nicht gedrückt) sich änderte, weil der Benutzer darauf klickte, oder, weil sich das Model so änderte, dass ein Event-Handler im View ansprang und den Modus des Widgets aktualisierte.

Letztendlich führt das dazu, dass an einigen Stellen im Code ein Event-Handler von einem Event entfernt wird, dann eine Änderung am Widgetzustand gemacht wird, und schließlich das Event wieder dort hinzugefügt wird, wo es ausgehängt wurde. Würde das nicht so gemacht werden, käme es zu einer Endlosschleife: Eine Änderung des Widgetmodus im Code führte dazu, dass das Model geändert wird, wodurch wieder die Widgetmodus aktualisiert wird und so weiter.

Bedenklich ist, dass dies kein Einzelfall ist, sondern an vielen Stellen so gemacht wird. Es ist schwer zu sagen, ob das Problem primär an der Art, wie GTK die Events benutzt, liegt, oder ob es grundsätzlicher Natur ist. In jedem Fall ist es für den Entwickler sehr schwer, nachzuvollziehen, welche Auswirkung der Aufruf bestimmter Funktionen hat, weil im Code of nicht direkt ersichtlich ist, welche Komponenten Handler auf bestimmte Events registrieren.

3.5.3 Erweiterbarkeit

Vor allem die Benutzeroberfläche ist durch ihre starre Struktur nur mit viel Aufwand erweiterbar. Das hat mehrere Gründe:

- Die Struktur des Hauptfensters hat eine sehr tiefe Hierarchie. Elemente aus der Mitte der Hierarchie können verändert, aber nur schwer ganz ausgetauscht werden.
- Die Klassen, die nach MVC dem Controller zuzuordnen sind, kennen die Hierarie der Widgets im Fenster im Detail. Das ist so, weil etwa die Funktion zum Exportieren des aktuellen Diagramms als PDF direkten Zugriff auf das Widget braucht, das dieses Diagramm anzeigt. Ändert man etwas an der Hierarchie der Widgets, muss auch der Controller umgeschrieben werden.
- Das Code ist stark davon abhängig, dass das Hauptfenster das einzige Fenster ist. Dem Controller wird also bei Benutzerinteraktionen kein oder kaum Kontext gegeben, weil der Controller die einzige Instanz des Hauptfensters kennt und sich den Kontext von dort herleiten kann. Möchte man weitere Fenster mit komplexen Interaktionsmöglichkeiten hinzufügen, muss man große Teile sowohl der Benutzeroberfläche als auch des Controllers umschreiben.

Tendenziell ist auch vorauszusehen, dass größere Änderungen am bestehenden Code Seiteneffekte aufgrund der komplexen Event-Infrastruktur, die unter 3.5.2.2 angesprochen wurde, haben werden.

3.5.4 Nötige Änderungen

Momentan erstellt der Benutzer auf den Messdaten Abschnitte, die aus mehreren Befahrungen bestehen. Um ein Straßennetzwerk erstellen zu können, sollten die Abschnitte in *Straßen* umbenannt werden. Ein wesentliches Problem ist dann, dass die Trassierungselemente nach aktuellem Datenmodell zu einer Befahrung der Straße gehören und nicht zur Straße selbst. Das ist zwar gut, um zu wissen, auf der Grundlage welcher Befahrung die Elemente berechnet wurden, aber logisch stellen die Elemente die Straße dar. Befahrungen sind abundante Merkmale, die von der Messdatenerfassung kommen, und sollten beim Modellieren des Straßennetzwerks keine Rolle mehr spielen, es sei denn, sie werden zur Festlegung der einzelnen Fahrspuren verwendet.

Messplan 1.0 stellt dem Benutzer keine Möglichkeit zur Verfügung, um Abschnitte zu löschen, nachdem sie angelegt wurden. Das Fehlen dieser Funktion ist kaum bemerkbar, weil üblicherweise nur mit einem einzigen Abschnitt gearbeitet wurde. Bearbeitet man jedoch ganze Straßennetze, wird die Funktion benötigt, um Fehler in der Abschnittsauswahl zu korrigieren. Die Möglichkeit, das Hinzufügen einer Straße über die *Rückgängig*-Funktion zu widerrufen, reicht hier nicht aus.

Ein Straßennetzwerk, das logisch über den bisherigen Abschnitten steht, lässt sich nicht gut in die bisherige Benutzeroberfläche einbinden. Es sollte ein Diagramm geben, in dem man das Straßennetzwerk modellieren kann, während die Details der Straßenführung, also die Trassierungselemente, ausgeblendet werden. Die Erfassung und Veränderung der Trassierungselemente dagegen sollte in einer getrennten Sicht passieren, in der nur die Straße sichtbar ist. Damit sieht der

Benutzer jederzeit die relevanten Informationen, ohne dass die Oberfläche überladen wirkt. In dem Maßstab, in dem man ein Straßennetzwerk üblicherweise betrachtet, wären einzelne Trassierungselemente auch nicht vernünftig editierbar.

Kapitel 4

Entwurf

4.1 Überblick

Dieses Kapitel beschreibt eine Erweiterung der Software Messplan 1, um folgende Anforderungen umzusetzen:

- Erfassung von Knotenpunkten.
- Automatische Erfassung von Fahrachsen in Knotenpunkten.
- Verlinkung von Straßen mit Knotenpunkten und anderen Straßen.
- Export der neuen Elemente und Verknüpfungen ins OpenDRIVE-Format.
- Anpassung der Benutzeroberfläche, sodass die Bearbeitung einer Straße und die Bearbeitung des Straßennetzwerks in getrennten Fenstern durchgeführt werden können.

Besonders wichtig ist hierbei die Erweiterung des Datenmodells von Messplan, um einen Datentyp für Kreuzungen zu haben und Straßen und Kreuzungen miteinander verknüpfen zu können. Die Umsetzung der aufgeführten Anforderungen ist Grundlage für alle anderen Anforderungen, die in der Zielstellung beschrieben werden.

Auf der Basis der Analyse von Messplan 1 schätzte ich den Aufwand für den Entwurf und die Implementierung dieser Grundfunktionalität auf insgesamt 4 Wochen. Tatsächlich benötigte ich insgesamt 8 Wochen. Verschätzt habe ich mich vor allem beim Aufwand für die Anpassung der Benutzeroberfläche, wo ich im Lauf der Implementierungen wegen der starken Abhängigkeiten der Komponenten untereinander deutlich mehr vom existierenden Code verändern musste als ich voraussah.

4.2 Benutzeroberfläche

4.2.1 Überblick

Dieser Teil des Entwurfs beschreibt die Benutzeroberfläche von Messplan 1.1 auf der Basis der Benutzeroberfläche von Messplan 1.0. Eine Beschreibung der Benutzeroberfläche von Messplan 1.0 findet sich im Handbuch dieser Version [15].

4.2.2 Hauptfenster

Bislang werden alle Aktionen auf den Messdaten, von der Abschnittaufwahl bis zur Elementbearbeitung, im Hauptfenster vorgenommen. Die Benutzeroberfläche hat eine flache Struktur; alle momentan möglichen Aktionen sind direkt verfügbar.

Da das Einfügen von Abschnitten in ein Straßennetz eine neue Hierarchieebene in der Datenstruktur erschließt, sollte entsprechend auch in der Benutzeroberfläche eine Hierarchie eingeführt werden. Konkret wird die Bearbeitung von Abschnitten in ein eigenes Editorfenster ausgelagert, in

dem mit den einzelnen Befahrungen eines Abschnitts gearbeitet, die Elementerkennung durchgeführt, und die erkannten Elemente editiert werden können. Abschnitte werden fortan als Straßen bezeichnet.

Für das Hauptfenster heißt das, dass die Straßen (bisher Abschnitte) im Projektexplorer keine sichtbaren Kindelemente mehr haben. Sie können im Straßeneditor bearbeitet werden, der in 4.2.3 vorgestellt wird. Zu den einzelnen Straßen wird angezeigt, ob auf ihnen bereits Elemente erkannt wurden oder ob sie momentan nur aus rohen Messdaten bestehen. Dies geschieht durch eine unterschiedliche Darstellung im Lageplan.

Neu hinzu kommen Kreuzungen, die im Projektexplorer auf derselben Ebene wie Straßen dargestellt werden. Kreuzungen können im Hauptfenster durch die Selektion von mindestens drei Punkten, die auf den geladenen Messpunkten liegen, erstellt werden. Die Straßenselektion funktioniert wie bisher, allerdings wird in der Nähe einer Kreuzung immer der Kontaktpunkt der Kreuzung als Endpunkt selektiert. Dasselbe gilt auch für andere Straßen: Endet eine Straße in der Nähe der Mauskoordinaten, so wird die neue Straße an die bereits existierende Straße angeschlossen. So können Straßen auf freier Strecke, an einer anderen Straße oder an einer Kreuzung enden.

Wie in Abbildung 4.1 zu sehen, enthält das Hauptfenster nur noch den Lageplan, nicht mehr das Richtungsband. Dieses ist für das Arbeiten auf einem Straßennetzwerk nicht notwendig und wird deshalb nur noch im Straßeneditor benutzt.

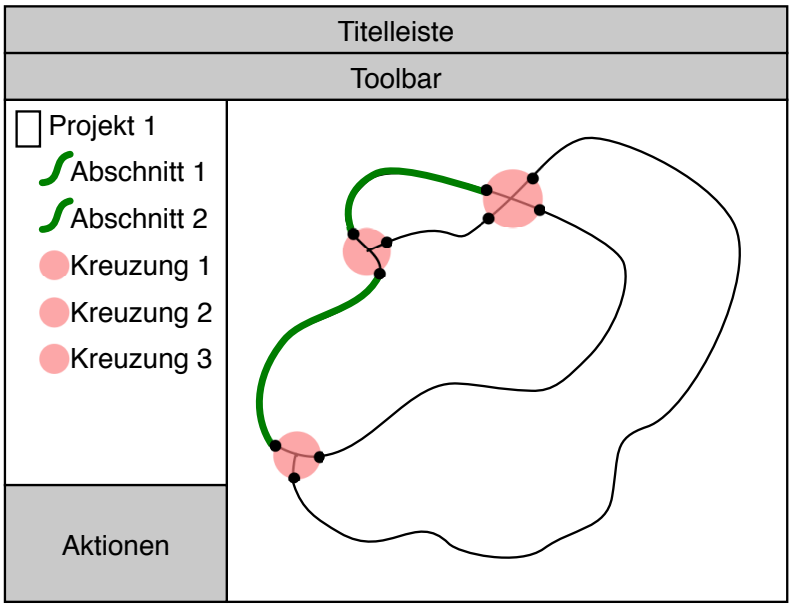


Abbildung 4.1: Skizze des Hauptfensters

4.2.3 Straßeneditor

Die einzelnen Befahrungen der Straße sowie die einzelnen Trassierungselemente sind über ein neues Editorfenster einsehbar und editierbar. Dies ist der Straßeneditor. Er ähnelt dem Hauptfenster von Version 1.0 mit folgenden Anpassungen:

- Die Zuordnung der Trassierungselemente zu einer Befahrung wird aufgehoben. Die Trassierungselemente sind nun direkt der Straße untergeordnet.
- An die Stelle des Projektexplorers tritt ein Straßenexplorer.
- Die Toolbar enthält nur noch die für die Mittelung und Elementerkennung relevanten Funktionen.

Für eine Skizze der Benutzerführung bei der Erstellung einer Straße siehe Abbildungen 4.2 und 4.3.

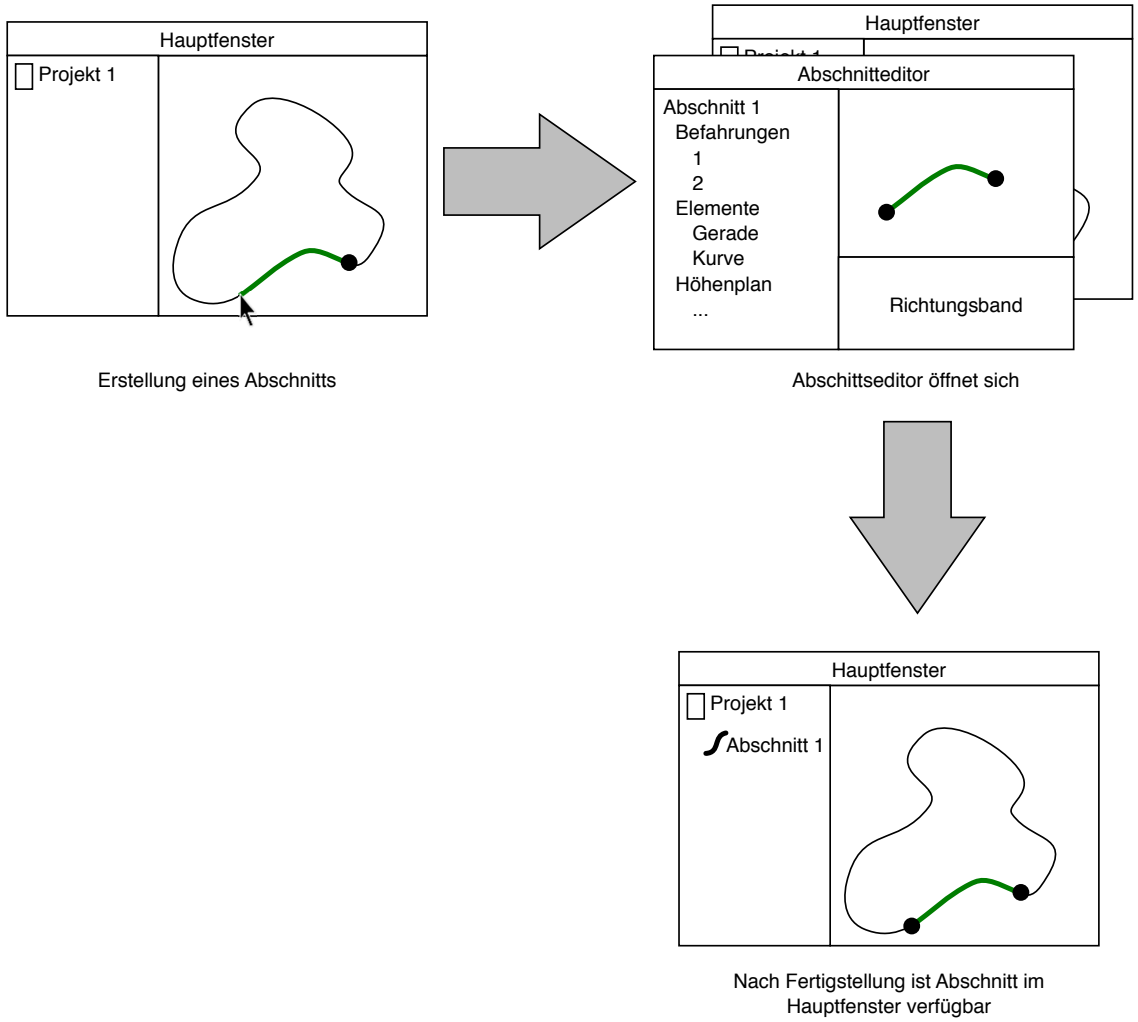


Abbildung 4.2: Erstellung und Bearbeitung einer Straße

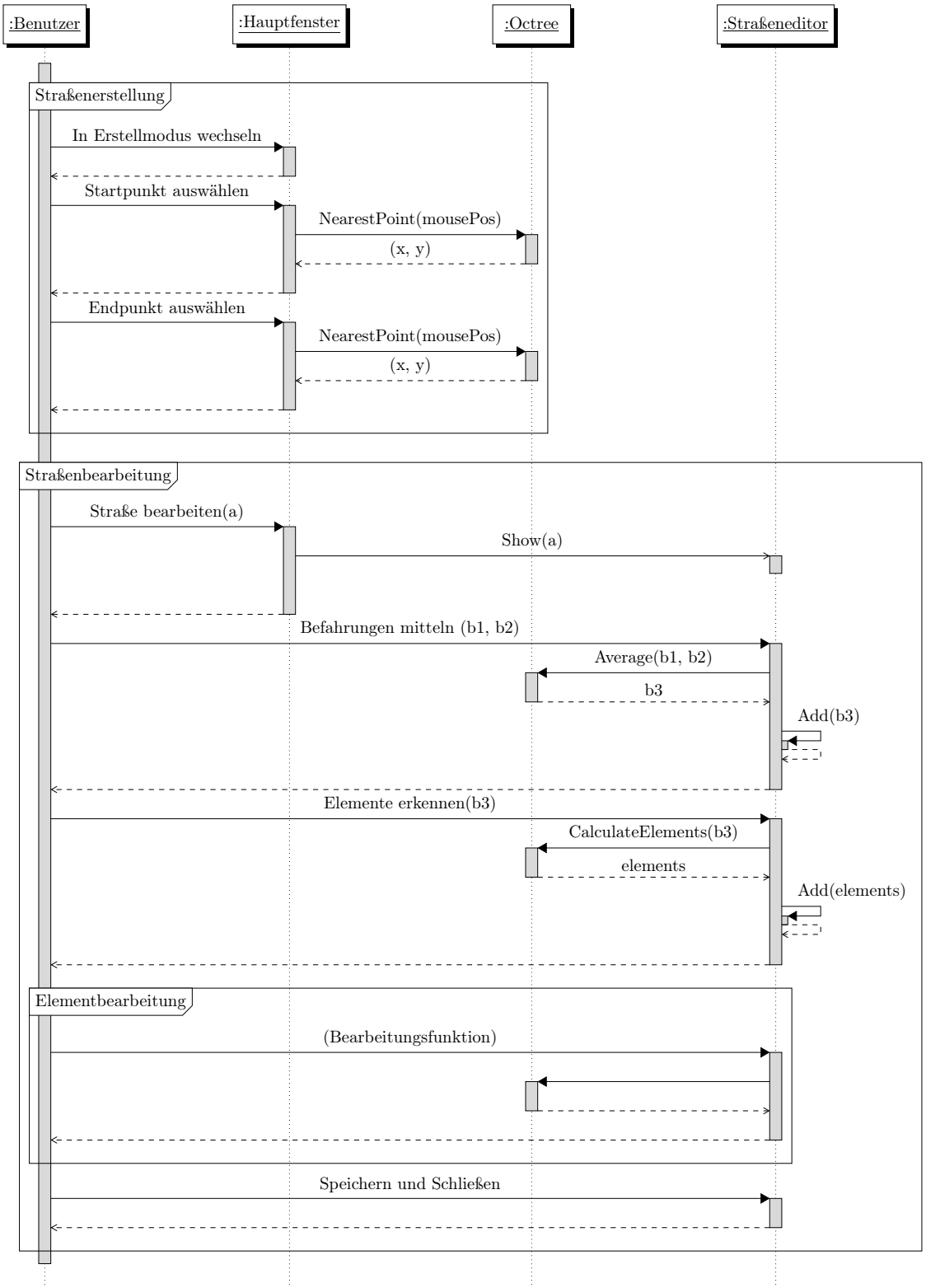


Abbildung 4.3: Sequenzdiagramm der Erstellung und Bearbeitung einer Straße

4.2.4 Erstellung und Darstellung von Kreuzungen

Das Arbeiten mit Kreuzungen ist nichttrivial. Eine Kreuzung gemäß OpenDRIVE ist eine Menge an Kontaktpunkten und Verbindungen zwischen diesen. Will man die Kreuzung nun als Fläche darstellen (was für den Benutzer viel angenehmer ist als nur die Einfärbung der Verbindungsstrecken), muss aus den Punkten ein Polygon konstruiert werden. Dazu müssen die Verbindungspunkte eine Reihenfolge haben, aus der sich der Umriss des Polygons ergibt.

Da man nicht davon ausgehen kann, dass der Benutzer die einzelnen Punkte des Polygons gemäß seines Umrisses nacheinander anklickt, muss ein Algorithmus benutzt werden, um die Punkte zu ordnen. Dieser ist relativ einfach: Man nimmt den ersten Punkt als Startpunkt und stellt die anderen Punkte relativ zum Startpunkt in einem Polarkoorinatensystem dar. Dann ordnet man die Punkte nach dem Wert des Parameters θ, siehe Abbildung 4.4. Natürlich erhält man auf diese Weise nicht das einzig richtige Polygon auf Basis der Messpunkte, aber eines, das hinreichend gut darstellbar ist.

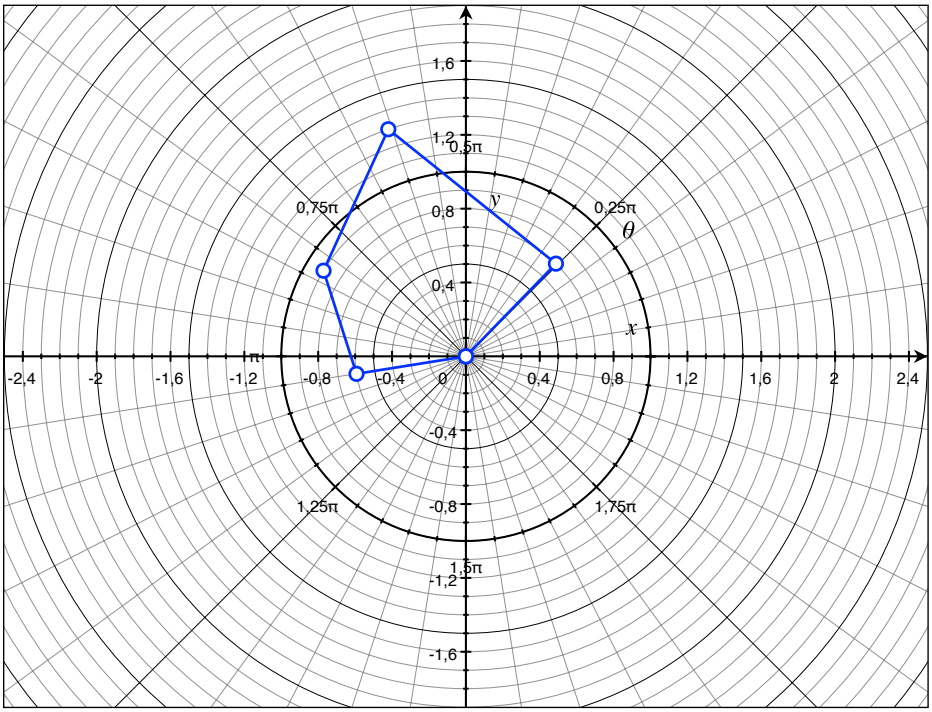


Abbildung 4.4: Konstruktion eines Polygons aus Punkten im Polarkoorinatensystem
Vom Ursprung ausgehend nimmt der Wert des Parameters θ konstant zu (gegen den Uhrzeigersinn).

Nachdem nun eine Reihenfolge der Punkte festlegbar ist, ist die Frage, wie aus dieser nun eine visuelle Repräsentation der Kreuzung erstellt werden soll. Die einfachste Möglichkeit ist, die Punkte eben genau als das Polygon, als das sie geordnet wurden, darzustellen. Das hätte aber zur Folge, dass die Straßen an den Eckpunkten in das Polygon eintreten, wodurch auch die Gefahr besteht, dass die Verbindungsstrecken der Kreuzung aus dem Polygon herausragen, wenn eine Ecke einen entsprechend spitzen Winkel hat.

Daher wird aus den Eckpunkten mithilfe von Bezier-Kurven eine abgerundete Fläche erstellt. die Kontrollpunkte der Kurven werden so gewählt, dass die Richtung der Kurve an einem Verknüpfungspunkt der Richtung der Linie zwischen dem vorherigen und dem nachfolgenden Verknüpfungspunkt entspricht. Abbildung 4.5 zeigt, wie auf diese Weise aus einem Dreieck eine abgerundete Fläche erstellt wird.

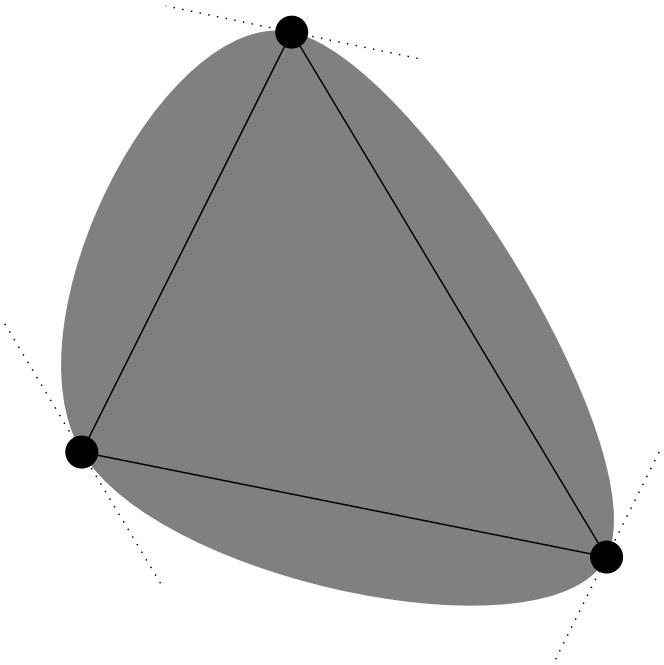


Abbildung 4.5: Konstruktion einer abgerundeten Fläche aus einem Dreieck

4.3 Architektur

4.3.1 Modellierung des Straßennetzwerks

Das Straßennetzwerk wird verwaltet von der Klasse `Project`. Diese enthält drei Listen, die zusammengenommen das Straßennetz darstellen:

- **Roads** ist eine Liste von Straßen. Diese werden als Instanzen der Klasse `Road` gespeichert. Die Liste enthält alle Straßen außerhalb von Kreuzungen, die der Benutzer auf Basis der Messpunkte angelegt hat.
- **Connections** verwaltet sämtliche Verknüpfungspunkte zwischen den Straßen. Jede Straße startet und endet an einem Verknüpfungspunkt. Die Straßen halten eine Referenz auf die beiden Verknüpfungspunkte, die sie begrenzen, aber verwaltet werden die Punkte von der Klasse `Project`. Die Verknüpfungspunkte werden als Instanzen der Klasse `SingleConnection` gespeichert.
- **Junctions** verwaltet alle vom Benutzer definierten Kreuzungen. Kreuzungen enthalten mindestens zwei Verknüpfungspunkte. Während die Verknüpfungen in `Project` je zwei Straßen miteinander verknüpfen, verknüpfen die Verknüpfungspunkte einer Kreuzung je eine Straße außerhalb der Kreuzung mit einer beliebigen Menge von Verbindungsstraßen. Die Verbindungsstraßen verbinden je zwei Verknüpfungspunkte der Kreuzung miteinander.

  Kreuzungen werden mit der Klasse `Junction` erstellt, Verknüpfungspunkte der Kreuzung mit `MultiConnection`. Die Verknüpfungspunkte werden von der Kreuzung verwaltet, ebenso die Verbindungsstraßen. Für die Verbindungsstraßen wird wie auch für die Straßen außerhalb die Klasse `Road` verwendet.

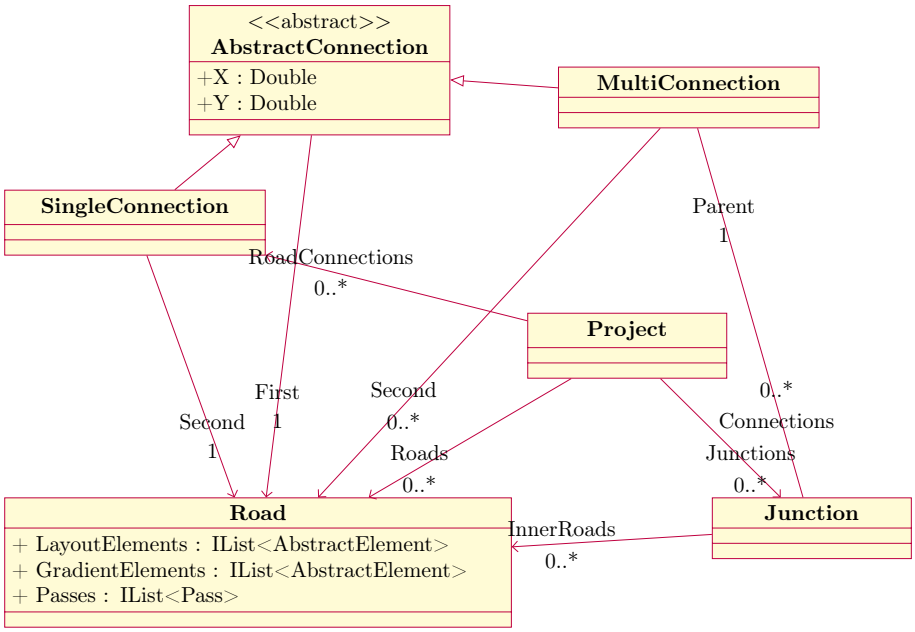


Abbildung 4.6: Klassendiagram der für die Netzwerkmodellierung relevanten Klassen

4.3.2 Struktur der Benutzeroberfläche

4.3.2.1 Layout der Hauptfenster

Die Benutzeroberfläche besteht aus zwei Editorfenstern: dem `MainWindow` und dem `RoadWindow`. Im `MainWindow` sieht der Benutzer einen Lageplan des Netzwerks und kann Straßen und Kreuzungen anlegen. Das `RoadWindow` ermöglicht es, die Trassierungselemente einer Straße zu berechnen und zu editieren. Im Wesentlichen umfasst die Funktionalität von `RoadWindow` die von Messplan 1.0.

Da beide Fenster ein ähnliches Layout haben, haben sie eine gemeinsame Vaterklasse `AbstractEditorWindow`. Damit im Hauptfenster der Willkommensbildschirm angezeigt werden kann, existiert die abstrakte Widget-Klasse `AbstractView`. Diese repräsentiert ein Widget, welches den gesamten Platz im Fenster einnimmt. Im `MainWindow` kann dies entweder `WelcomeView` oder `ProjectView` sein. Im `RoadWindow` ist es immer `RoadView`.

`RoadView` und `ProjectView` stellen durch ihre gemeinsame Vaterklasse `AbstractEditorView` eine Menüleiste, eine Toolbar und eine Tab-Leiste, mit der zwischen verschiedenen Ansichten umgeschaltet werden kann, zur Verfügung. Diese verschiedenen Ansichten sind alle von der abstrakten Klasse `AbstractPane` abgeleitet.

Die Hauptansicht beider Fenster ist jeweils eine Ansicht auf der Basis von `AbstractDiagramPane`. Im `MainWindow` ist dies `NetworkDiagramPane`, mit der man das Straßennetzwerk bearbeiten kann; im `RoadWindow` ist es `RoadDiagramPane`, mit dem die Trassierungselemente einer Straße bearbeitet werden können. Die Hauptansicht besteht jeweils aus einer Baumansicht auf der linken Seite und einem oder mehreren Diagrammen auf der rechten Seite.

Neben der jeweiligen Hauptansicht stellt das `RoadWindow` noch die `ElementPane` zur Verfügung, in der die Parameter der einzelnen Trassierungselement aufgelistet sind; das `MainWindow` hat die beiden Ansichten `SettingsPane` und `ProjectSettingsPane` für globale beziehungsweise projektspezifische Einstellungen.

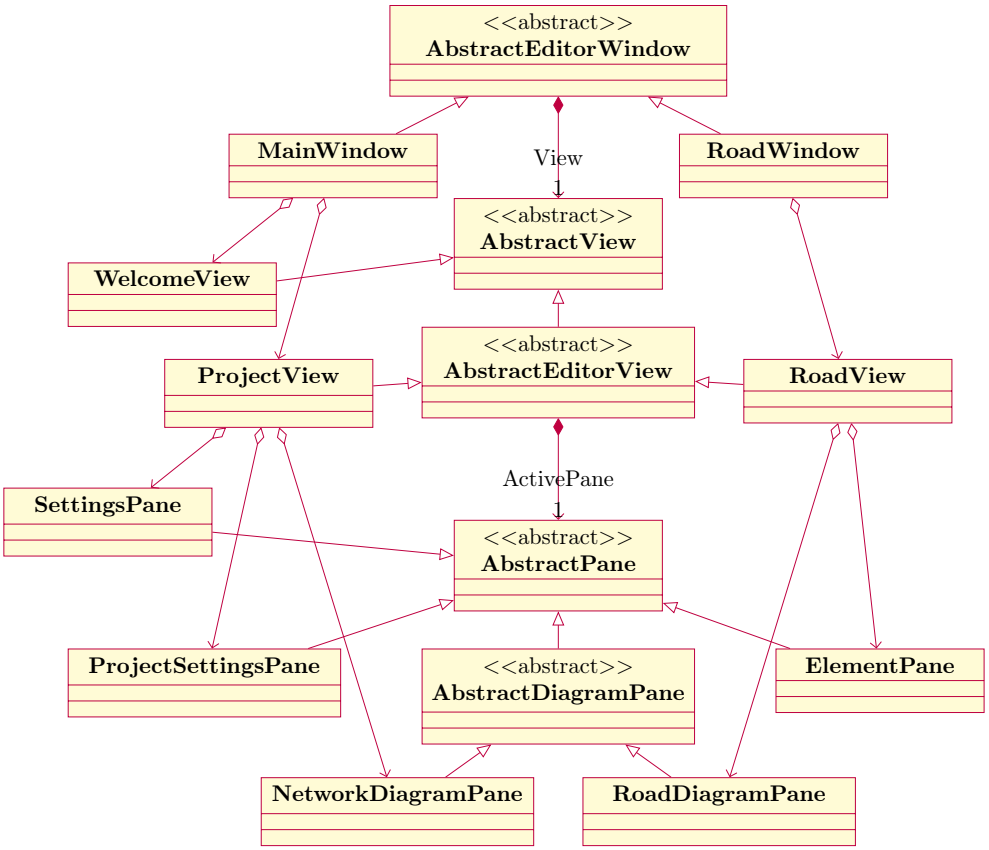


Abbildung 4.7: Klassenhierarchie der Benutzeroberfläche

4.3.2.2 Zustand der Benutzeroberfläche

Jede Instanz von `MainWindow` und `RoadWindow` hat einen Zustand, der definiert, welche Elemente der Benutzeroberfläche momentan sichtbar, ausgewählt etc. ist. Dieser Zustand wird von dem Modul `GuiState` verwaltet. `GuiState` bietet zwei Sichten auf diesen Zustand an, über die der Zustand abgefragt und geändert werden kann:

- **Baumbasierte Sicht**: Diese Sicht wird für die verschiedenen Explorerwidgets bereitgestellt. Die ausgewählten Elemente werden als Index-Pfad dargestellt: An erster Stelle steht der Index des Wurzelelements (immer 0), dahinter kommt der Index eines Elements eine Ebene unter dem Wurzelelement und so weiter, bis hin zum ausgewählten Element.
- **Diagrammbasierte Sicht**: Diese Sicht wird von den verschiedenen Diagrammwidgets benutzt. Im Gegensatz zur baumbasierten Sicht lassen sich die ausgewählten Objekte in der diagrammbasierten Sicht über den Typ der Objekte auswählen: So lassen sich etwa alle momentan ausgewählten Trassierungselemente abfragen. Die Hierarchie der Elemente, die in der Baumansicht der Explorerwidgets dargestellt wird, wird in der diagrammbasierten Sicht wegabstrahiert.

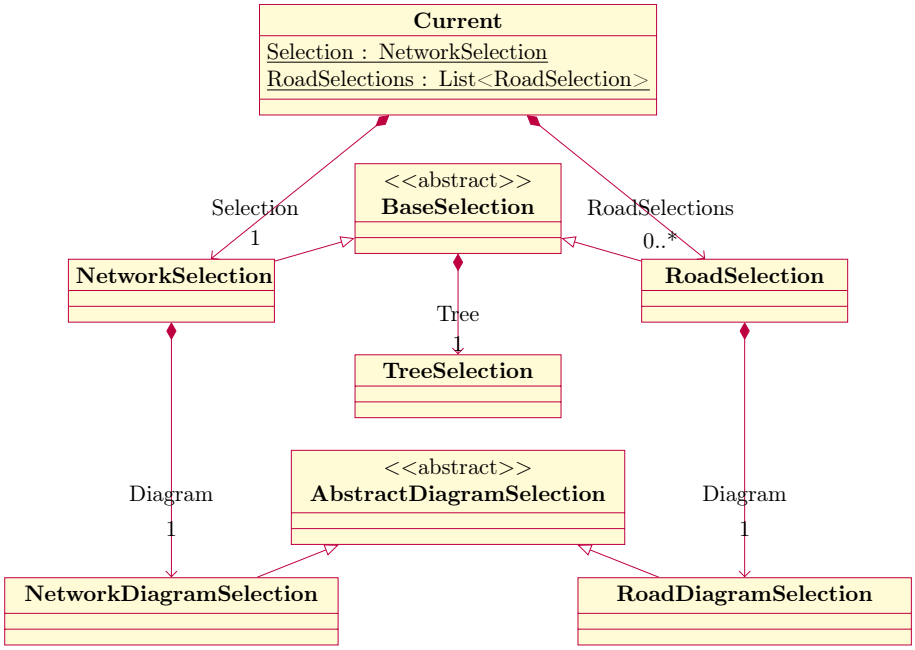


Abbildung 4.8: Klassenhierarchie des Moduls `GuiState`

Der Zustand des `MainWindow` wird von der Klasse `NetworkSelection` verwaltet, deren einzige Instanz ein statisches Feld der Klasse `GuiState.Current` ist. Der Zustand aller Instanzen von `RoadWindow` werden von je einer Instanz von `RoadSelection` verwaltet. Diese werden in `GuiState.Current` in der statischen Liste `RoadSelections` abgelegt.

Die Baumsicht wird in beiden Fällen von einer Instanz von `TreeSelection` zur Verfügung gestellt. Da diese Klasse agnostisch gegenüber den Objekttypen ist, die in dem dazugehörenden Explorerwidget angezeigt werden, benötigt sie keine Spezialisierungen für die beiden Anwendungsfälle. Die Diagrammsicht `AbstractDiagramSelection` dagegen kennt die verwalteten Objekttypen und hat deshalb zwei Spezialisierungen `NetworkDiagramSelection` und `RoadDiagramSelection`.

4.3.3 Die Controller-Klassen

Alle Benutzerinteraktionen, die das Ziel haben, das Datenmodell zu verändern (also etwa das Anlegen neuer Straßen und Kreuzungen oder das Erkennen von Elementen) laufen über eine Controllerkomponente gemäß MVC. Das Event, das von GTK für eine Benutzerinteraktion erzeugt wird, wird allerdings zunächst im Code der View-Klassen abgefangen und dann erst an den Controller weitergereicht. Der Grund dafür ist, dass der objektorientierte Ansatz, auf dem GTK aufgebaut ist, ein Stück weit dem Konzept von MVC widerspricht: Ein Widget als Objekt definiert alle Funktionen, die darauf ausgeführt werden können - inklusive der Benutzerinteraktionen.

Es gibt verschiedene Möglichkeiten, mit dieser Architektur umzugehen. In Messplan werden Benutzerinteraktionen von den View-Klassen abgefangen und eventuell an die Controllerklassen weidergereicht. Diese werden für Interaktionen benutzt, die das Datenmodell verändern; Aktionen, die nur die Selektion verändern, werden direkt in den View-Klassen verarbeitet.

Neu in Messplan 1.1 ist die Einführung mehrerer Controller. Bisher gab es nur einen Controller, der alle Aktionen beinhaltete; nun wird dieser aufgeteilt in einen `ApplicationController`, der Aktionen enthält die global sind oder das Hauptfenster betreffen, und einen `RoadController`, der die Aktionen, die beim Editieren einer Straße möglich sind, bereitstellt.

Neben den Methoden, die eine Benutzerinteraktion ausführen, enthalten die Controller-Klassen Methoden, die prüfen, ob bestimmte Aktionen möglich sind, und entsprechend `True` oder `False` zurückliefern. Ob eine Aktion möglich ist, hängt üblicherweise von der aktuellen Selektion ab und davon, welcher Tab im Fenster gerade aktiv ist. Darüber hinaus sind die meisten Aktionen nicht möglich, wenn gerade Messpunkte importiert werden (dies kann mehrere Minuten dauern, die Benutzeroberfläche bleibt währenddessen aktiv).

Alle Aufrufe der Controller laufen über die Klassen `ApplicationCommands` und `RoadEditorCommands`. Diese Klassen enthalten ein festes Set von Objekten der `Command`-Klasse und stellen die eigentliche Schnittstelle der Controller-Klassen dar. Wie in Abbildung 4.9 zu sehen ist, sind diese Objekte in `ApplicationCommands` statisch, weil es nur ein Hauptfenster gibt. Von `RoadEditorCommands` wird dagegen für jedes `RoadWindow` ein eigenes Objekt erstellt.

Ein `Command`-Objekt kennt die Methode, die die von ihm dargestellte Aktion ausführt, sowie die Methode, die überprüft, ob die entsprechende Akion ausführbar ist. Als Bindeglied zwischen Controller und View kann ein `Command` zwei Widgets erstellen: Einen Knopf und einen Menüeintrag. Beide führen dann bei einem Klick automatisch die Aktion des `Command` aus und werden deaktiviert, wenn die Aktion nicht möglich ist. Der Knopf wird üblicherweise in die Werzeugleiste eines Fensters eingebunden, der Menüeintrag in das Fenstermenü und das Kontextmenü des Explorer-Widgets. Das Aussehen der Widgets hängt von den Werten der Felder von `Command` ab; so kann etwa das Icon des Knopfes oder der Text des Menüeintrags gesetzt werden, siehe Abbildung 4.9.

Die Methoden in den Controllern erstellen meistens ein Objekt vom Typ `IAction`. Dies ist ein Interface für alle Aktionen, die rückgängig gemacht werden können. Für jede derartige Aktion existiert eine Klasse, die `IAction` implementiert und die gewünschte Aktion durchführen und rückgängig machen kann. Die Controller-Klassen selbst erstellen nur ein Objekt aus einer dieser Klassen und fügen es dem `ActionStack` zu, der die reversiblen Aktionen verwaltet. Nachdem eine Aktion rückgängig gemacht wurde, lässt sie sich auch über den `ActionStack` wiederherstellen. Der Ablauf der Erstellung einer Aktion ist in Abbildung 4.10 dargestellt. Der Übersichtlichkeit halber sind die Aufrufe zum Rückgängig machen und Wiederherstellen direkt vom Benutzer zum `ActionStack` dargestellt; tatsächlich interagiert der Benutzer mit den entsprechenden Knöpfen in der Toolbar, den Menüeinträgen oder der grafischen Darstellung der durchgeführten Aktionen im Hauptfenster.

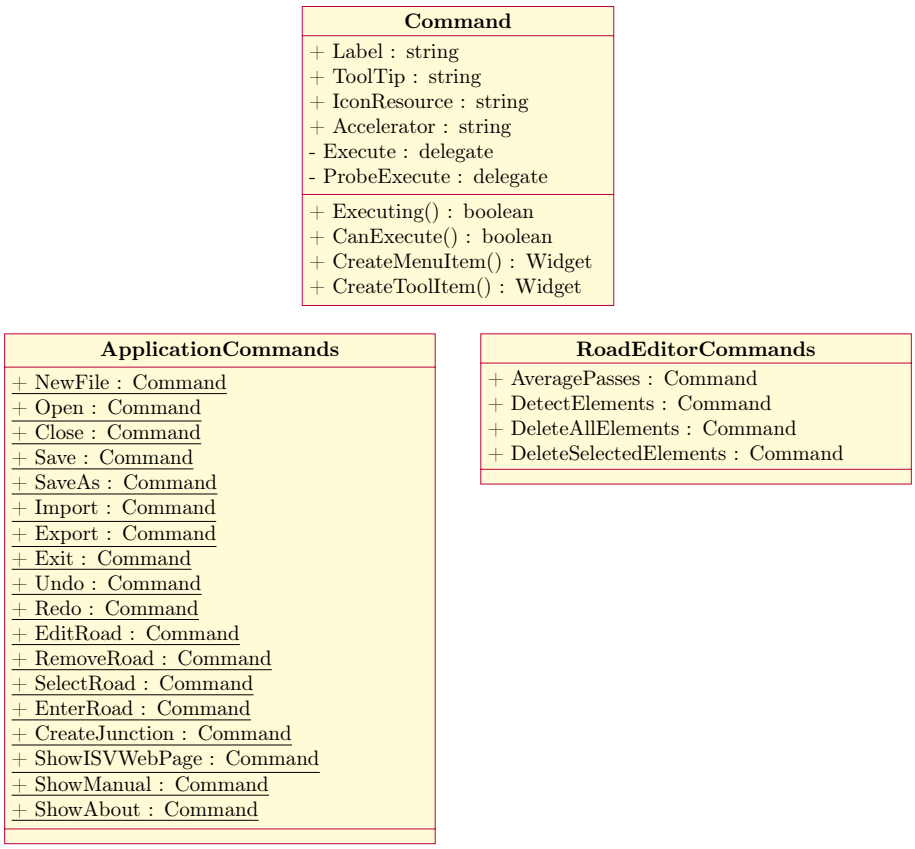


Abbildung 4.9: Schnittstelle der Controller-Klassen

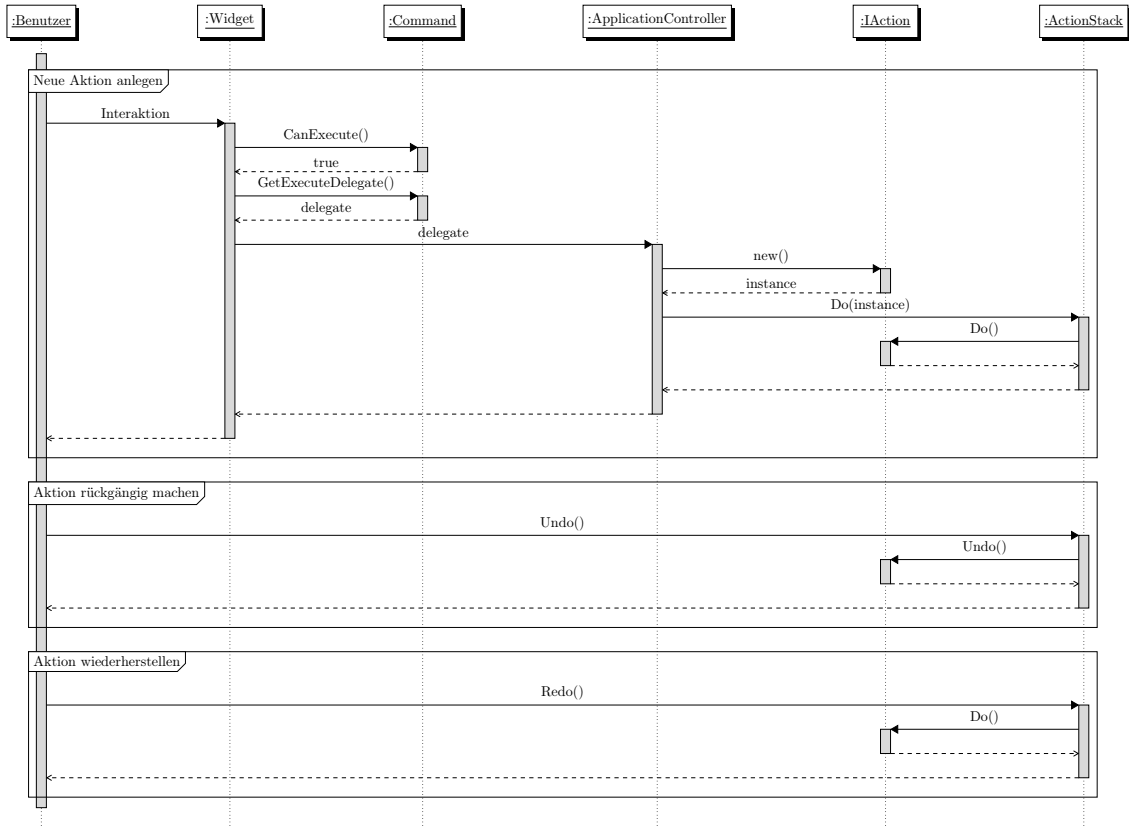


Abbildung 4.10: Benutzerinteraktion, die zur Ausführung einer reversiblen Aktion führt

Kapitel 5

Resultat

5.1 Übersicht

Dieses Kapitel beschreibt die neuen Funktionen, die in Messplan implementiert wurden. Diese erweiterte Version von Messplan trägt die Versionsnummer 1.1.

Zunächst wird gezeigt, wie die bisherige Funktionalität von Messplan in der neuen Version erreichbar ist. Dabei wird auch der neue Straßeneditor vorgestellt. Danach wird erläutert, welche neuen Funktionen Messplan 1.1 hat und wie diese benutzt werden können.

5.2 Anlegen einer Straße

5.2.1 Hauptfenster

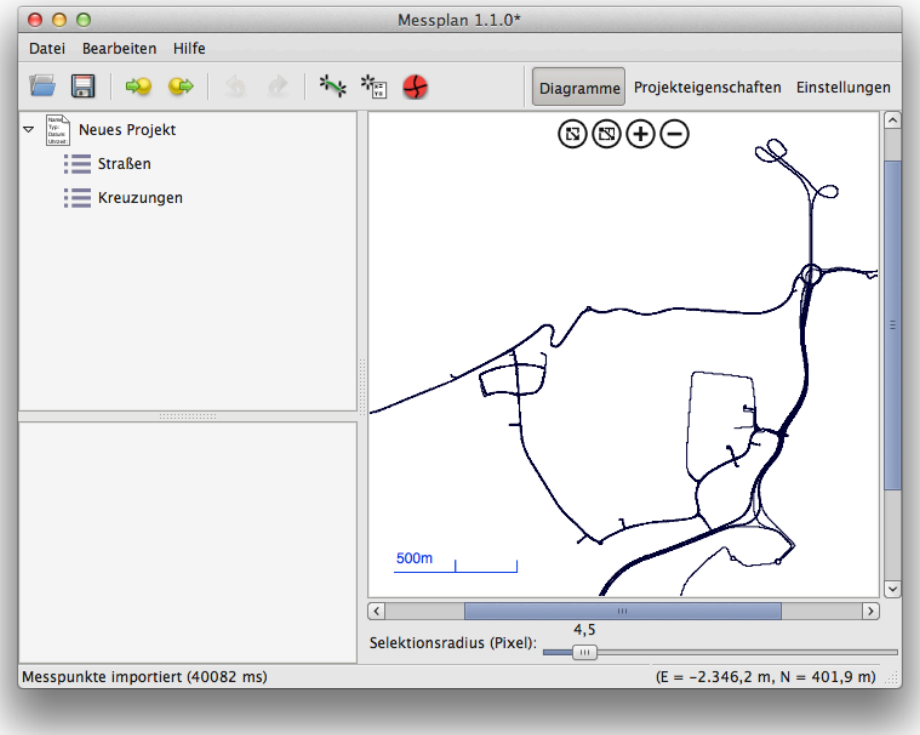


Abbildung 5.1: In Messplan 1.1 importiertes Straßennetz

Das neue Hauptfenster ist in Abbildung 5.1 zu sehen. Folgende Änderungen wurden am ursprünglichen Hauptfenster (Abbildung 2.12) vorgenommen:

- Der Projektexplorer enthält nun unter dem Wurzelknoten („Projekt") zwei Listen: Die der Straßen und die der Kreuzungen. Beide sind anfangs leer. Dies ist eine leichte Abweichung vom Entwurf der Oberfläche, die dazu dient, dass man die beiden Listen getrennt auf- oder zuklappen kann.
- Das Richtungsband fehlt wie im Entwurf beschrieben.
- der Tab *Elementtabelle* wurde entfernt. Da dieser die Elemente einer Straße im Detail auflistet, wurde er in den Straßeneditor verschoben.
- Die Knöpfe zum Erkennen von Elementen und zum Mitteln von Befahrungen kurden ebenfalls in den Straßeneditor verschoben.
- Es wurde der neue Knopf *Kreuzung erstellen* zur Werkzeugleiste hinzugefügt (roter Kreis mit zwei schwarzen Linien). Mit diesem lässt sich der Kreuzungsauswahlmodus aktivieren.
- Der Selektionsradius lässt sich über einen Slider unter dem Netzwerkdiagramm einstellen. Bisher war er unveränderlich auf 4 Pixel festgelegt. Der Selektionsradius gibt an, in welcher Umgebung eines Verknüpfungspunktes nach Befahrungen gesucht werden soll, die in ihn aufgenommen werden sollen. Dies ist vor allem für Kreuzungen wichtig, da man zur Erfassung dieser relativ nahe in die Daten hineinzoomt und die harte Einstellung von vier Pixeln nicht mehr ausreicht, um alle Fahrspuren in einem Verbindungspunkt auszuwählen.

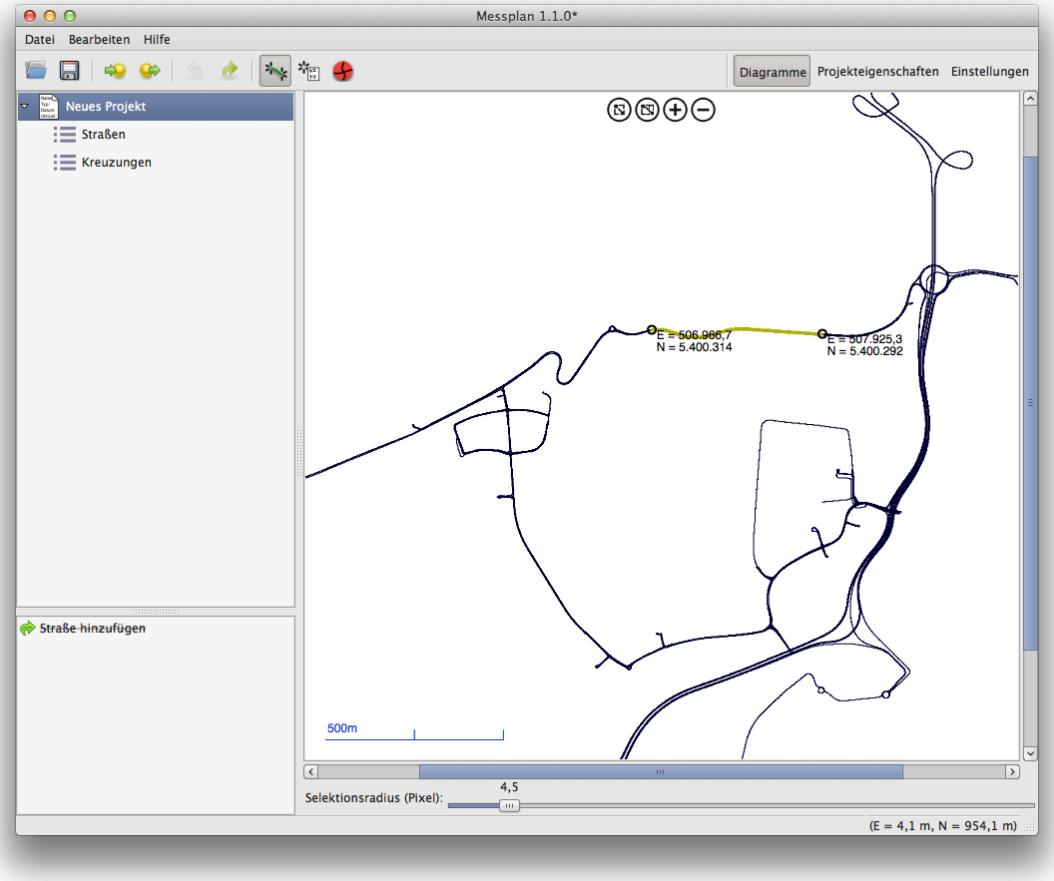


Abbildung 5.2: Hauptfenster von Messplan 1.1 bei während der Auswahl eines Abschnitts

Wie bisher lässt sich eine neuen Straße mit dem Knopf *Neuen Abschnitt erstellen* auf den Messdaten auswählen. Neu ist, dass während der Auswahl eines Abschnitts der Knopf gedrückt bleibt, um dem Benutzer den aktuellen Modus der Oberfläche zu signalisieren. Der Knopf zur Auswahl von Kreuzungen verhält sich analog. Abbildung 5.2 zeigt die Auswahl eines Abschnitts, von dem der erste Punkt (im Bild links) bereits fest gewählt wurde und eine Vorschau der Straße angezeigt wird, deren Ende sich an dem Messpunkt befindet, der dem Mauszeiger am nächsten ist (aus technischen Gründen ist der Mauszeiger im Bild nicht sichtbar).

Messplan 1.0 zeigte an den Start- und Endpunkten von Abschnitten direkt im Diagramm die Koordinaten des Punktes an (Abbildung 2.13). Dies wurde in Messplan 1.1 entfernt, da es bei Kreuzungen schnell zu Überlappungen der Textdarstellung dieser Koorinaten kommt. Die Koordinaten werden nur während der Auswahl eines Abschnitts angezeigt.

Nachdem die Straße ausgewählt wurde, erscheint sie im Projektexplorer. Die Befahrungen und Elemente der Straße sind im Gegensatz zu Messplan 1.0 im Hauptfenster nicht sichtbar. Statt dessen kann man über das Kontextmenü der Straße der Straßeneditor öffnen (Abbildung 5.3). Er öffnet sich über den Menüeintrag *Straße bearbeiten*. Neu ist auch die Möglichkeit, die Straße wieder zu löschen.

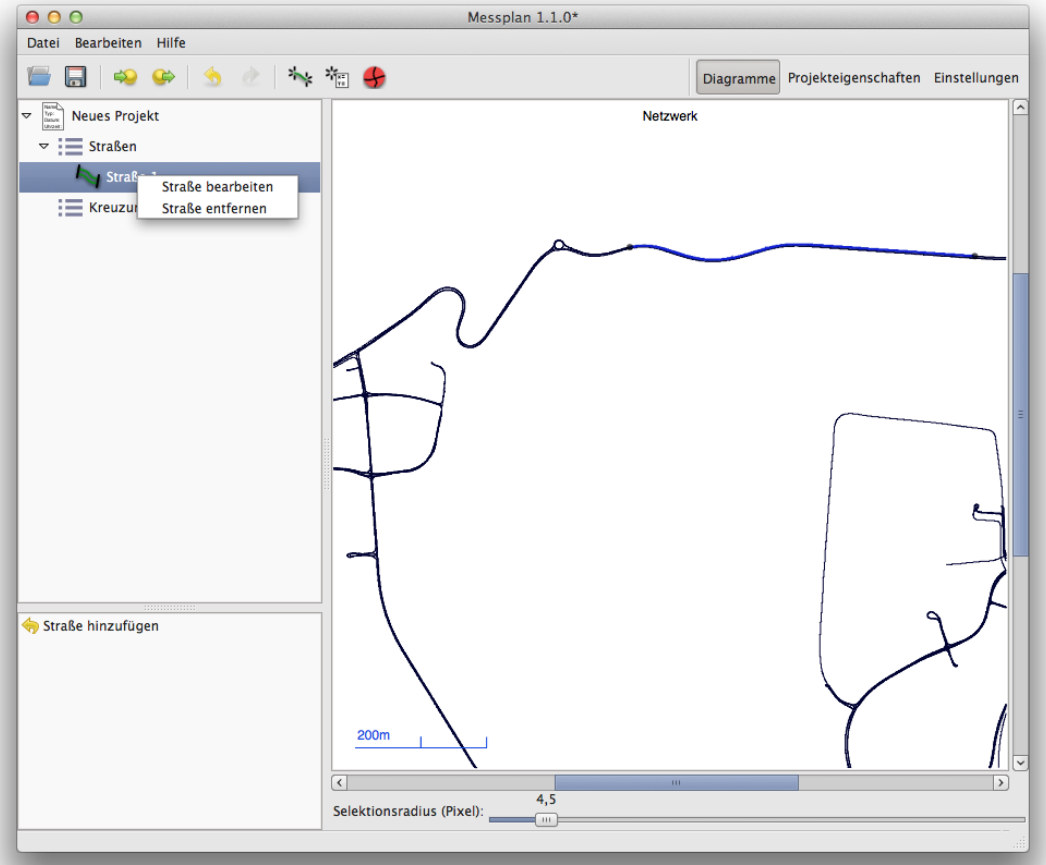
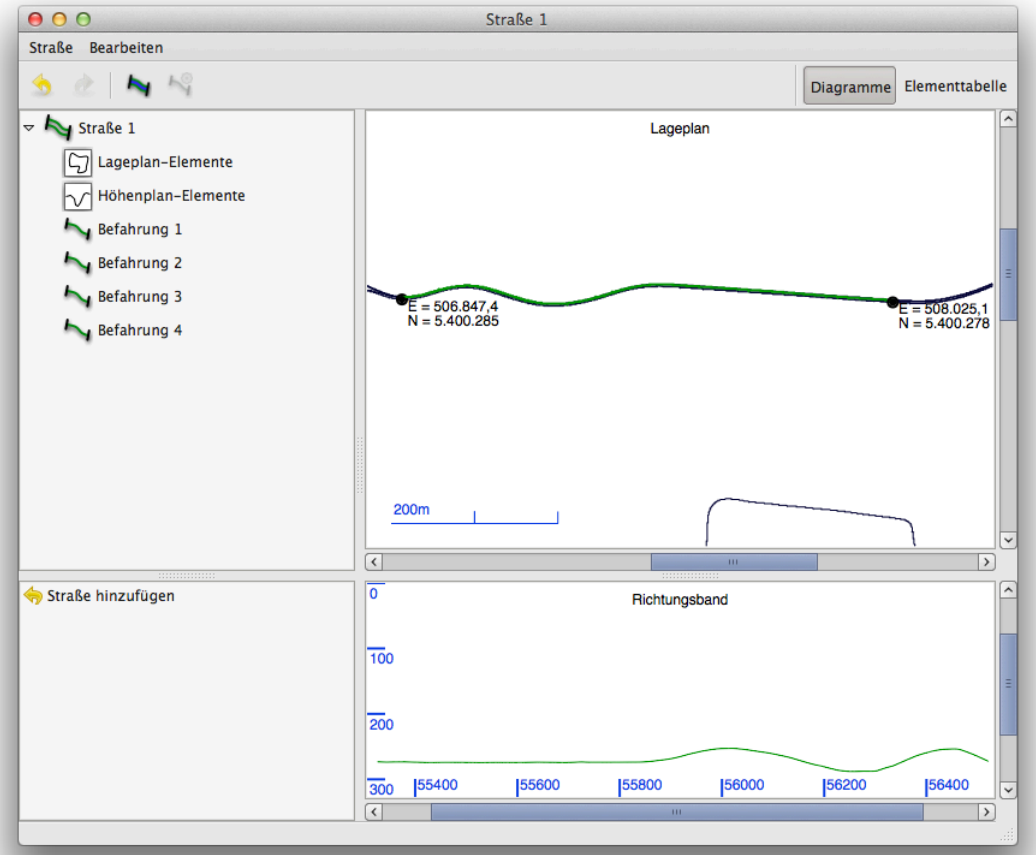


Abbildung 5.3: Kontextmenü einer Straße

5.2.2 Straßeneditor



Im Straßeneditor finden sich die Funktionen und Ansichten, die aus dem Hauptfenster entfernt wurden. Wie in Abbildung 5.2.2 zu sehen, besteht die Hauptansicht aus dem Straßenexplorer, der Aktionshistorie und zwei Diagrammen, die entweder den Lageplan und das Richtungsband anzeigen, oder (wenn Höhenplanelemente ausgewählt sind) den Höhenblan und das Neigungsband. Die Aktionshistorie ist identisch mit der im Hauptfenster; dies führt zwar dazu, dass im Straßeneditor Aktionen gelistet werden, die im Hauptfenster ausgeführt wurden, aber da die Aktionen alle eine textuelle Beschreibung haben, erschließt sich das dem Benutzer sofort und wird kaum zu Verwirrung führen.

Der Straßeneditor ist kein modales Fenster. Der Benutzer kann jederzeit zum Hauptfenster zurückwechseln und auch mehrere Straßeneditoren zur selben Zeit offen haben. Ein wenig problematisch scheint die Ähnlichkeit der Fenster: Auf den ersten Blick erkennt man nur schwer, welches der Fenster ein Straßeneditor und welches das Hauptfenster ist. Das liegt vor allem daran, dass beide Fenster aus dem ursprünglichen Hauptfenster von Messplan 1.0 hervorgegangen sind.

Wie entworfen sind Lage- und Höhenplanelemente nun direkt der Straße untergeordnet, die als Wurzelknoten im Straßenexplorer erscheint. Die Befahrungen sind darunter aufgeführt. Elemente kann man weiterhin erkennen, indem man eine Befahrung markiert und die Aktion *Elemente erkennen* aus der Werkzeugleiste, dem Kontextmenü oder dem Fenstermenü auswählt. Es ist im Nachhinein allerdings nicht mehr ersichtlich, auf Grundlage welcher Befahrung die Elemente erkannt wurden. Falls der Benutzer dies vermerken will, kann er die Möglichkeit nutzen, die Namen der Befahrungen entsprechend zu ändern.

Befahrungen lassen sich weiterhin über den entsprechenden Knopf oder die Menüeinträge mitteln.

5.2.3 Bearbeitungsstand von Straßen

Straßen können nur exportiert werden, wenn auf ihnen Trassierungselemente erkannt wurden. Im Straßeneditor ist direkt ersichtlich, ob die Straße bereits Trassierungselemente besitzt oder nicht. Im Hauptfenster wird dies dadurch gekennzeichnet, dass die Straße im Netzwerkdiagramm unterschiedlich dargestellt wird. In Abbildung 5.4 sind vier Straßen zu sehen. Zwei davon werden breit und mit dem für Straßen typischen Mittelstreifen angezeigt. Das bedeutet, dass für diese Straßen bereits Elemente erkannt wurden und die Darstellung der Straßen auf diesen Elementen beruht. Die anderen Straße werden als einfache Linie dargestellt; sie bestehen momentan nur aus Messdaten und besitzen in Messplan noch keine Trassierungselemente.

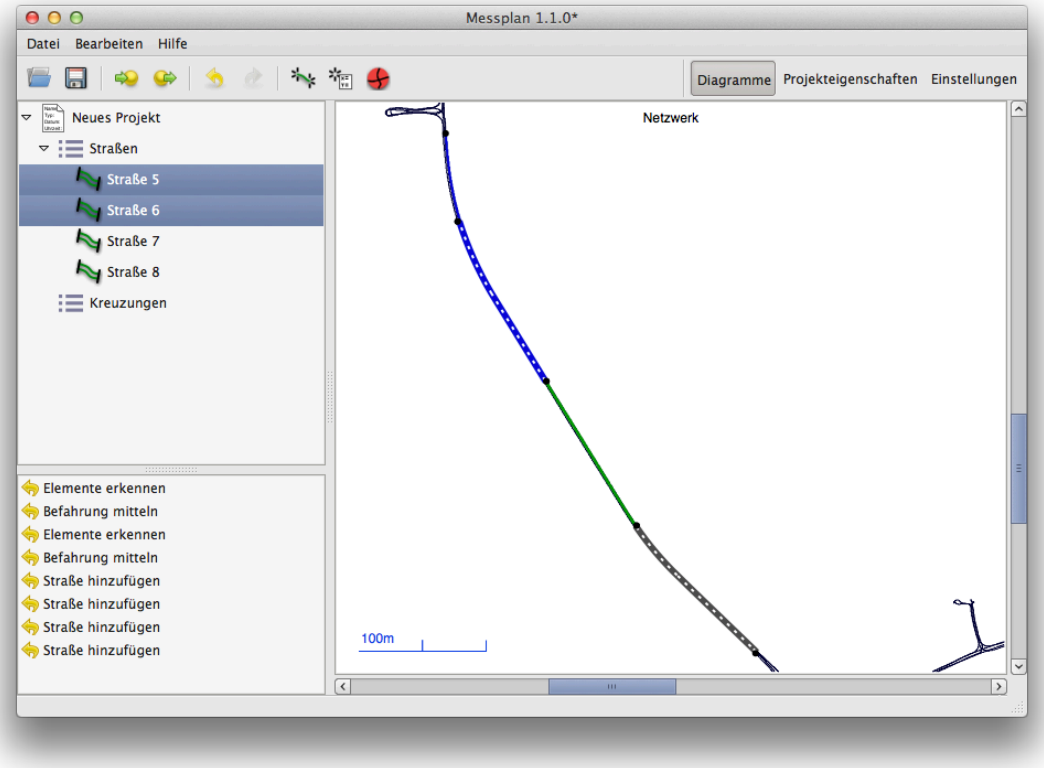


Abbildung 5.4: Darstellung von Straßen mit und ohne Trassierungselemente

Die Darstellung der Straßen wurde auch dahingehend erweitert, dass die momentan selektierte Straße(n) in blau statt in grün (für nicht bearbeitete Straßen) beziehungsweise grau (für bearbeitete Straßen) gezeichnet werden. In Messplan 1.0 wurde nur der aktuell selektierte Abschnitt in den Lageplan gezeichnet, daher war diese Funktion dort noch nicht notwendig. Um das erfasste Straßennetz überblicken zu können, war es notwendig, alle in Messplan 1.1 angelegten Straßen dauerhaft sichtbar zu machen, weswegen die Selektionsvisualisierung notwendig wurde. In Abbildung 5.4 sind zwei Straßen mit und zwei ohne Trassierungselementen zu sehen, von denen jeweils eine selektiert ist.

Exportiert der Benutzer im Hauptfenster den aktuellen Stand, werden nur die Straßen exportiert, die Trassierungselemente besitzen. Alle *rohen*, also unbearbeiteten Straßen werden ignoriert.

5.3 Anlegen von Kreuzungen

5.3.1 Ablauf

Kreuzungen können wie auch Straßen im Hauptfenster angelegt werden. Gestartet wird der Vorgang über den Knopf *Neue Kreuzung anlegen* in der Werkzeugleiste oder den entsprechenden Eintrag im Fenstermenü *Bearbeiten*. Danach kann der Benutzer mit der Maus im Netzwerk-Diagramm die Verknüpfungspunkte der Kreuzung auswählen - das heißt die Stellen, an denen Straßen in die Kreuzung einmünden. Wie auch bei der Anlage von Straßen in Messplan wird die Mausposition auf den nächsten Messpunkt abgebildet.

Sobald der erste Punkt ausgewählt ist, wird bei jeder Mausbewegung eine automatische Vorschau der Kreuzung in das Netzwerkdiagramm gezeichnet. Dazu wird die aktuelle Mausposition wieder auf die Messdaten abgebildet und mit diesem und den bisher ausgewählten Punkten eine Vorschau der Kreuzungsfläche gezeichnet.

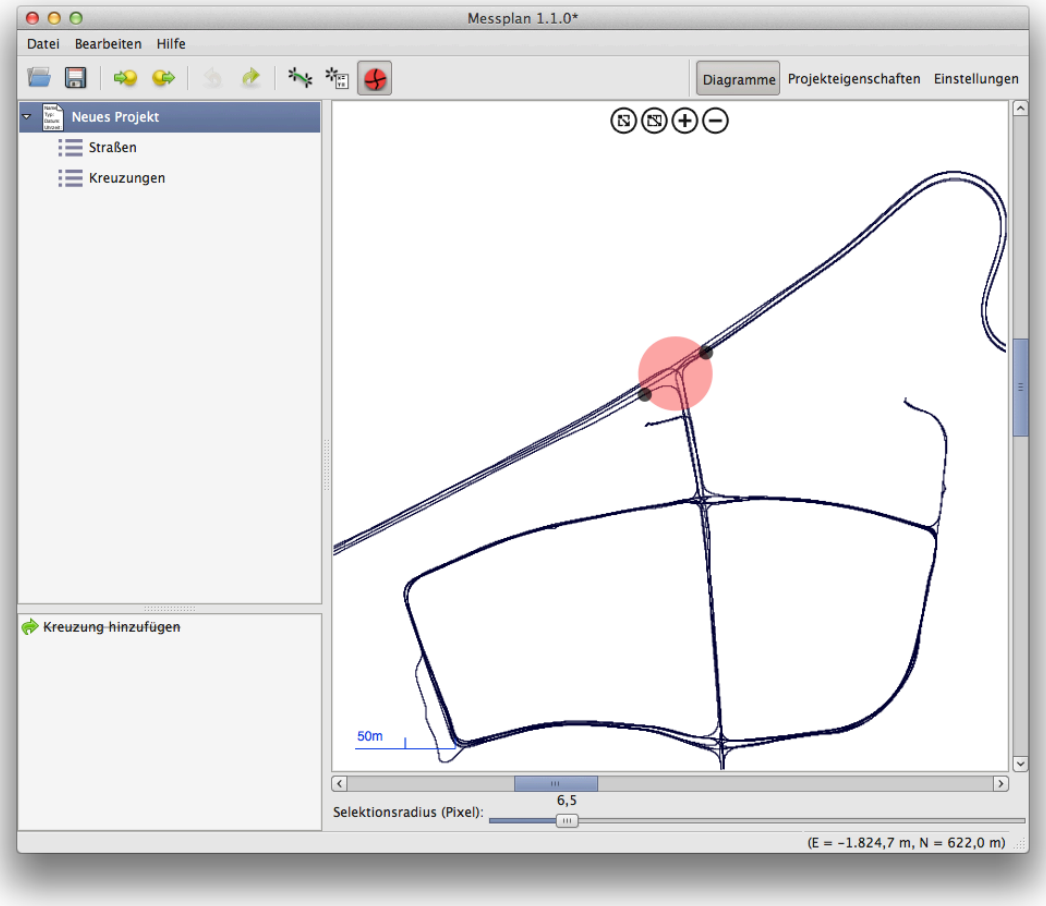


Abbildung 5.5: Vorschau bei Auswahl des zweiten Kreuzungspunktes

In Abbildung 5.5 ist zu sehen, wie auf der Basis zweier Punkte die Vorschau der Kreuzungsfläche gezeichnet wird. Bei zwei Punkten wird die Fläche als Kreis gezeichnet, dessen Mittelpunkt genau zwischen den beiden Punkten liegt und dessen Radius dem Abstand zwischen den beiden Punkten entspricht. Ab drei Punkten werden für die Visualisierung Bezier-Kurven verwendet, wie in 4.2.4 erläutert. Wichtig dabei ist, dass die Visualisierung ausschließlich als Veranschaulichung für den Benutzer dient. Sie ist allenfalls eine heuristische Annäherung an die tatsächliche Kreuzungsfläche. Es kann durchaus vorkommen, dass die Verbindungspfade in der Kreuzung aus der gezeichneten Fläche herausragen oder dass Messpunkte von Strecken, die nicht zur Kreuzung gehören, innerhalb der gezeichneten Kreuzungsfläche liegen. Allerdings ist dies aufgrund der üblicherweise annähernd rechtwinkligen Lage der Anschlussstraßen zueinander nur selten

der Fall.

Die Größe der Kreise, mit denen die ausgewählten Verknüpfungspunkte dargestellt werden, orientierst sich am ausgewählten Selektionsradius, der mit dem Schieberegler unter dem Diagramm eingestellt werden kann. Die Kreise geben dem Benutzer ein visuelles Feedback darüber, welche Teilstrecken von seiner Auswahl erfasst werden und welche nicht: Es werden in einem Verknüpfungspunkt genau die Teilstrecken erfasst, die innerhalb des dargestellten Kreises liegen.

Eine Kreuzung kann mit beliebig vielen Verknüpfungspunkten definiert werden. In der Praxis werden es üblicherweise drei oder vier, seltener fünf oder sechs sein. Um die Erstellung einer Kreuzung abzuschließen, muss der Benutzer bei der Auswahl des letzen Punktes die rechte statt der linken Maustaste benutzen. Dadurch wird der Punkt hinzugefügt und die Auswahl von Verknüpfungspunkten abgeschlossen. Daraufhin wird ein vollautomatischer Prozess angestoßen, der die Kreuzung auf Basis der Messpunkte erfasst:

- Zunächst wird das Kreuzungspolygon selbst gespeichert, wie vom Benutzer eingegeben. Mit gespeichert wird auch der Selektionsradius, mit dem die Verknüpfungspunkte ausgewählt wurden. Die Verknüpfungspunkte werden weiterhin mit diesem Radius gezeichnet, um auch im Nachhinein den verwendeten Selektionsradius anzuzeigen.
- Nun werden für jedes Paar von Verknüpfungen alles Befahrungen gesucht, die von der einen zur anderen Verknüpfung führen und umgekehrt. Dafür wird der bereits in Messplan 1 implementierte Algorithmus zur Selektion von Abschnitten verwendet. Als Selektionsradius wird der vom Benutzer eingestellte Wert verwendet.
- Da alle Messdaten zusammenhängen, muss überprüft werden, ob eine potentielle Befahrung tatsächlich zur Kreuzung gehört; es könnte auch eine Strecke sein, die über einen Teil des restlichen erfassten Straßennetzes von einem Verknüpfungspunkt zum anderen führt. Dazu wird der Durchmesser des Kreuzungspolygons an dessen dickster Stelle berechnet und überprüft, ob das kleinste Rechteck, in das alle Punkte der Befahrung passen, eine geringere Breite und Höhe als diesen Durchmesser hat. Dies ist eine Heuristik, die sich gut in die annähernde Darstellung der Kreuzung einpasst: Nimmt eine Befahrung einen größeren Raum ein als das Kreuzungspolygon, wird mit großer Wahrscheinlichkeit auch ihre Darstellung im Diagramm die angezeigte Kreuzungsfläche verlassen. Damit ist für den Benutzer ersichtlich, warum eine Befahrung nicht als der Kreuzung zugehörig erkann wurde.
- Die Befahrungen zwischen zwei Verknüpfungspunkten werden nach ihrer Richtung aufgeteilt. Sofern zwischen einem Verknüpfungspunktpaar mindestens eine Befahrung existiert, wird dafür ein Pfad in der Kreuzung angelegt. Falls eine weitere Befahrung in der Gegenrichtung existiert, wird dafür ebenfalls ein Pfad angelegt. Existieren mehrere Befahrungen in dieselbe Richtung für ein Verknüpfungspunktpaar, so werden diese in einem Pfad zusammengefasst.
- Die so erstellten Pfade werden in derselben Struktur wie eine Straße gespeichert. Damit kann ein Pfad Lageplanelemente, Höhenplanelemente und eine oder mehrere Befahrungen besitzen. Im Gegensatz zu Straßen werden für Pfade jedoch automatisch Trassierungselemente angelegt. Diese werden aus der ersten Befahrung ermittelt. Dies dient dazu, dem Benutzer sofort alle Pfade als aus Elementen bestehende breite Linien anzeigen zu können.

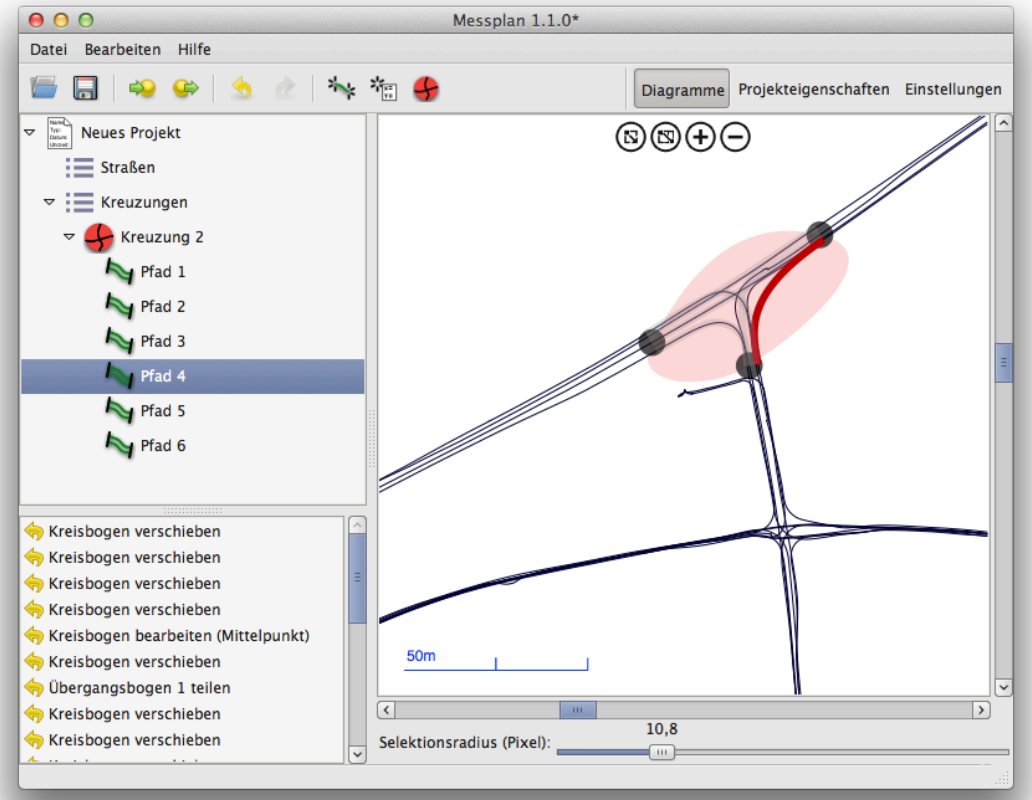


Abbildung 5.6: Eine in Messplan erfasste Kreuzung mit mehreren Pfaden

In Abbildung 5.6 ist eine fertig erfasste Kreuzung zu sehen. Die Kreuzung wird halbtransparent rot dargestellt, die Pfade halbtransparent grau. Der ausgewählte Pfad ist rot markiert. Die einzelnen Pfade sind auch im Projektexplorer sichtbar; um sie zu bearbeiten, kann wie auch für Straßen der Straßeneditor verwendet werden. Wie zu sehen ist, liegen die erkannten Pfade nicht immer genau auf den Messpunkten. Das liegt im Wesentlichen daran, dass für die automatische Elementerkennung dieselben Parameter benutzt werden wie auch für Straßen. Beispielsweise ist standardmäßig eine Mindestlänge eines Elements von 10 Metern eingestellt. In Kreuzungen werden wesentlich kürzere Elemente benötigt, um etwa Kreisbögen von Kurven darzustellen. Natürlich kann der Benutzer über den Straßeneditor entsprechende Elemente von Hand hinzufügen.

Pfade von Kreuzungen lassen sich wie Straßen über das Kontextmenü im Projektexplorer entfernen. Allerdings lassen sich keine zusätzlichen Pfade hinzufügen. In dem Fall, dass ein Pfad in den Messdaten existiert, der nicht automatisch zur Kreuzung hinzugefügt wurde, muss der Benutzer die Verknüpfungspunkte der Kreuzung anders wählen. Der wahrscheinlichste Grund für das Fehlen des Pfades ist, dass der Selektionsradius zu klein gewählt war und ein Pfad deshalb nicht erfasst wurde.

Was als Kreuzung definiert wird, bleibt dem Benutzer überlassen. Abbildung 5.7 etwa zeigt eine komplexe Kreuzung, die einen Kreisverkehr enthält. Statt das Konstrukt als eine einzige Kreuzung zu erfassen, hätte man auch die Anschlussstellen an den Kreisverkehr als einzelne Kreuzungen und die Abschnitte dazwischen als Straßen erfassen können. Es ist außerdem ersichtlich, dass man relativ große Selektionsradien braucht, wenn eine Straße wie in diesem Fall getrennte Fahrbahnen pro Fahrtrichtung hat.

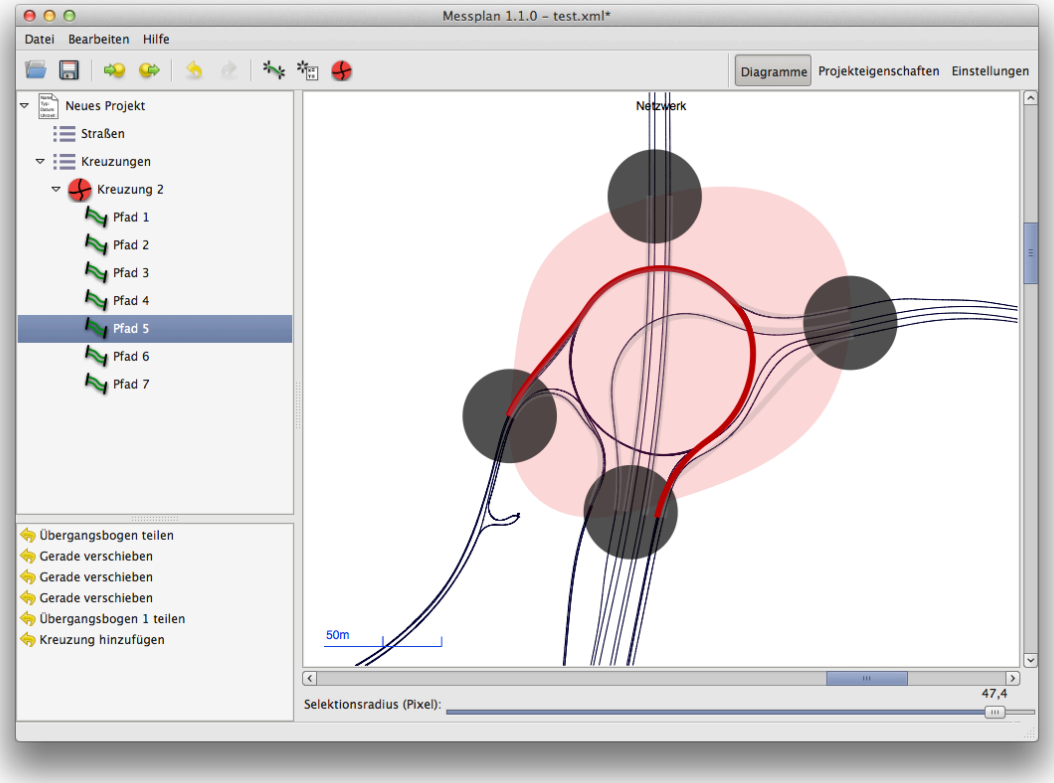


Abbildung 5.7: Eine komplexe Kreuzung mit Kreisverkehr in Messplan

Besonders wichtig bei der Kreuzungserfassung ist, dass an jedem Verknüpfungspunkt der Kreuzeng nur eine Straße enden kann. Betrachtet man Abbildung 5.7 genauer, stellt man fest, dass im unteren Anschlusspunkt keinesfalls nur eine Straße abgeht: Die ganz linke Befahrung verläuft noch ein gutes Stück neben den Fahrbstreifen der eigentlichen Straße her, und der Punkt, an dem sie mit ihnen verschmilzt, ist außerhalb des Bildausschnitts. Insofern ist die Kreuzung fehlerhaft erfasst; der untere Verknüpfungspunkt müsste viel weiter unter angesetzt werden, oder man müsste die Kreuzung in kleinere Kreuzungen aufteilen.

5.3.2 Elementerkennung bei Kreuzungspfaden

Wie im vorherigen Abschnitt beschrieben, werden die Elemente der Kreuzungspfade automatisch erkannt. Dabei benutzen sie den Algorithmus, der auch für die Elementerkennung von Straßen benutzt wird. Die Parameter für den Algorithmus sind in den Einstellungen von Messplan änderbar und sollten für das Anlegen von Kreuzungen geändert werden. Es wäre von Nutzen, ein Set von Parametern zu haben, das für Pfade in Kreuzungen gute Ergebnisse liefert. Das Problem hierbei ist, dass die Elementerkennung für freie Strecken ausgelegt ist, auf denen die tatsächlichen Trassierungselemente primär in Hinsicht auf die Fahrdynamik angeordnet sind. Im Kreuzungsbereich ist die Situation grundsätzlich eine Andere, da es üblicherweise keine oder kaum Übergangsbögen gibt, die Kreisbögen deutlich enger sein können und allgemein die Elemente nur fahrlogischer Natur sind, da eine Kreuzung bautechnisch im simpelsten Fall aus einer ebenen Fläche besteht, auf der eventuell Fahrbahnmarkierungen angebracht sind.

Die Anwendung des Elementerkennungsalgorithmus für Kreuzungspfade ist also nur ein Notbe-

helf. Um Kreuzungspfade gut zu erkennen, wird es wahrscheinlich nicht ausreichen, ein eigenes Parameterset für Kreuzungen zu finden; es sollte ein eigener Algorithmus auf Basis der baulichen Eigenschaften von Kreuzungen entwickelt werden. Dies übersteigt jedoch den Rahmen dieser Diplomarbeit. Um mit dem existiertenden Algorithmus halbwegs brauchbare Ergebnisse zu bekommen, sollte in den Einstellungen die Mindestlänge von Elementen reduziert werden. Vor allem für Kreisbögen habe ich jedoch keine Einstellung gefunden, die ein Ergebnis liefert, das mit nur wenig Nachbearbeitung übernommen werden kann. In den meisten Fällen, die nachbearbeitet werden müssen, handelt es sich um Pfade, die einen Abbiegevorgang modellieren. Hier muss üblicherweise ein Kreisbogen eingepasst werden.

Abbildung 5.8 zeigt eine Kreuzung, deren Pfade automatisch erkannt wurden. Die starken Ausschläge mancher Pfade kommen daher, dass nur ein Übergangsbogen zwischen die beiden Verknüpfungspunkte gelegt wird. Übergangsbögen sind Polynome dritten Grades und werden aus Start- und Endpunkt sowie der Steigung an diesen berechnet. Die Steigung gibt jedoch nicht die Richtung an, weshalb es dazu kommt, dass manche Pfade aus der falschen Richtung an ihren Endpunkt anschließen. Dies kommt genau dann vor, wenn der Unterschied zwischen der korrekten Richtung in Start- und Endpunkt mehr als 90° beträgt, da die Richtung im Startpunkt für die Berechnung als 0° definiert wird. Als mathematische Funktion kann ein Polynom, wenn es mit Steigung 0 anfängt, keine Kurve von 90° oder mehr vollziehen, was zu dem Problem der Richtungsumkehr führt.

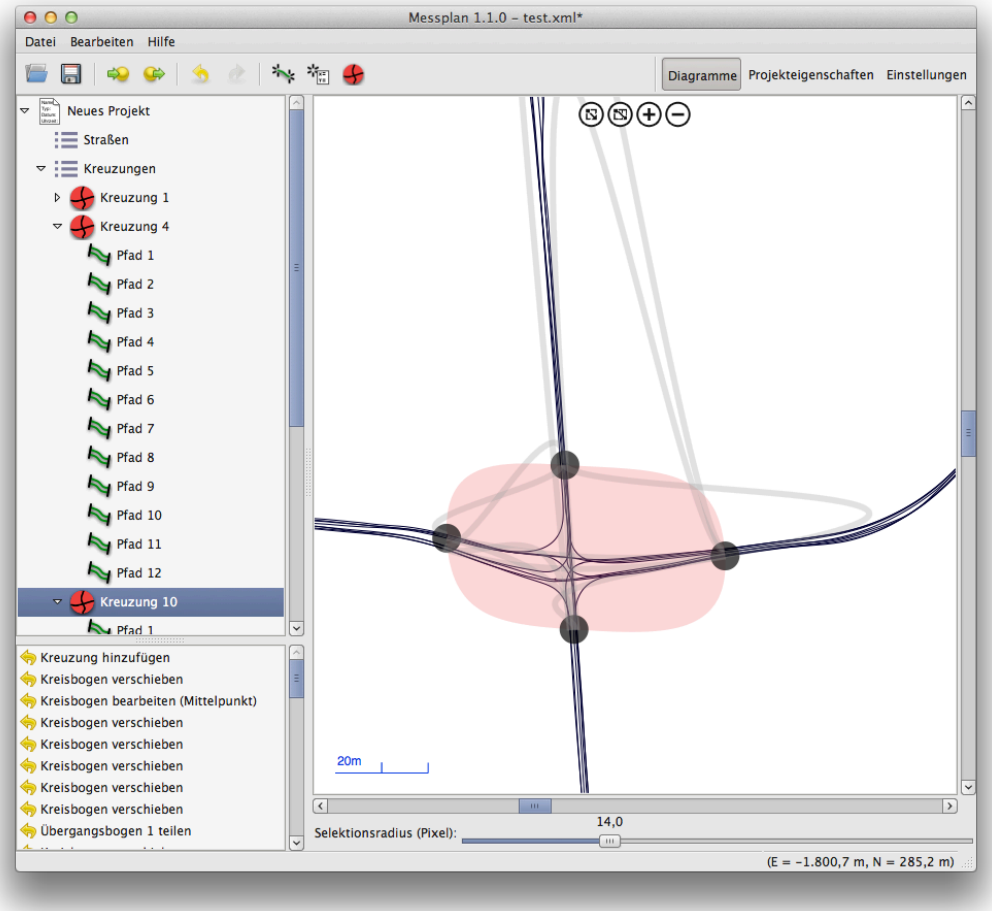


Abbildung 5.8: Schlecht ermittelte Pfade in einer Kreuzung

Bei Tests habe ich kein Parameterset gefunden, das die Elementerkennung veranlasst, akzeptable Elemente für Kreuzungspfade zu generieren. Insbesondere scharfe Kurven, die auf gerade Strecken folgen, werden nicht als Kreisbögen erkannt. Diese Beschreibung trifft auf die meisten Abbiegepfade zu.

5.4 Aufbau eines Straßennetzwerks

Kreuzungen und Straßen können in Messplan verknüpft werden, um ein Straßennetzwerk aufzubauen. Beim Anlegen von Straßen sucht Messplan bei der Auswahl von Start- und Endpunkt nach existierenden Verknüpfungspunkten in der Nähe der Mausposition. Wird ein Punkt innerhalb eines Radius von 20 Pixeln gefunden, wird dieser Verknüpfungspunkt anstatt dem nächsten Punkt aus den Messdaten verwendet. Dem Benutzer wird dies angezeigt, indem ein roter Kreis um den Verknüpfungspunkt gezeichnet wird. Abbildung 5.9 zeigt eine solche Markierung während dem Anlegen eines Straße. Nachdem die Straße erstellt wurde, wird der Verknüpfungspunkt wieder mit dem üblichen Farbschema gezeichnet.

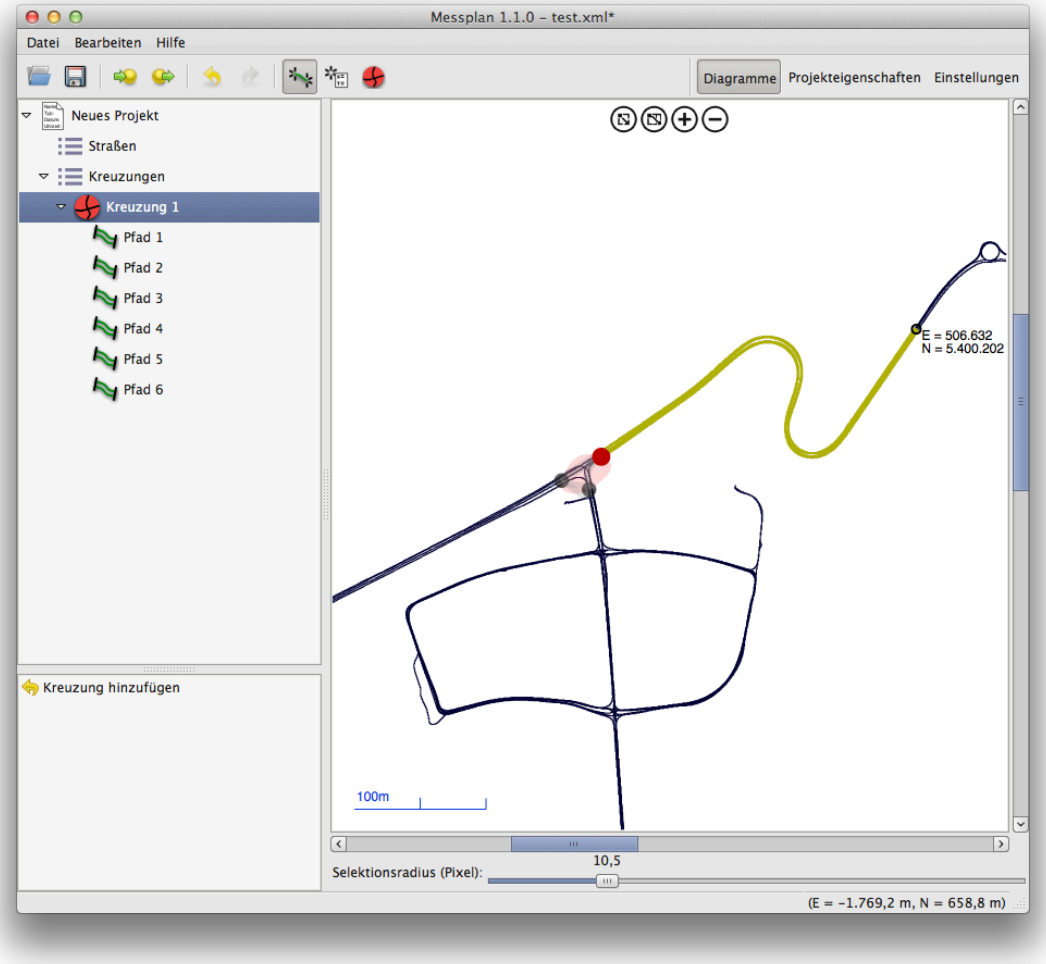


Abbildung 5.9: Verknüpfung einer Straße mit einer Kreuzung

Ein Hindernis für das Bearbeiten des Straßennetzes ist, dass der Start- und Endpunkt einer Straße, nachdem sie angelegt wurde, nicht mehr geändert werden kann. Insbesondere kann keine Kreuzung an eine existierende Straße angefügt werden. Deshalb ist es anzuraten, zunächst die Kreuzungen zu definieren und danach die Straßen zwischen den Kreuzungen zu erfassen. Die Unveränderlichkeit des Start- und Endpunktes ist vor allem existierendem Code von Messplan 1 geschuldet. Dort wurde davon ausgegangen, dass die nichttriviale Auswahl von mehreren Befahrungen endgültig ist: Eine Verschiebung von einem der Punkte hätte zur Folge, dass alle Befahrungen neu berechnet werden müssten und je nach Messdaten sogar Befahrungen wegfielen. Da in Messplan 1 die Trassierungselemente Kindelemente der Befahrungen waren, wären sie bei einer Neuberechnung der Befahrungen weggefallen. Die ist nun nicht mehr der Fall, doch ist die Einschränkung nicht groß genug, dass sich der Aufwand, sie zu beheben, gelohnt hätte.

Straßen können auch mit anderen Straßen verknüpft werden. Dies funktioniert auf dieselbe Art

wie bei Kreuzungen. Sowohl für Straßen als auch für Kreuzungen gilt, dass an einen existierende Verknüpfungspunkt nur eine einzige Straße angeschlossen werden kann (bei Verknüpfungspunkten von Straßen gilt das natürlich zusätzlich zu der Straße, die den Verknüpfungspunkt definiert). In allen anderen Fällen - also wenn zwei oder mehr Straßen an einen Verknüpfungspunkt angeschlossen werden sollen - handelt es sich offensichtlich um eine Kreuzung, die entsprechend erfasst werden sollte.

Die Möglichkeit, Straßen miteinander zu verknüpfen, hat mehrere Zwecke:

- Sollte sich die Anzahl der Fahrstreifen einer Straße ändern, muss dafür in OpenDRIVE zumindest ein neuer Straßenabschnitt erstellt werden. Da Messplan momentan keine Straßenabschnitte unterstützt (jede Straße hat genau einen Abschnitt), könnte das durch aufeinander folgende Straßen mit unterschiedlicher Fahrstreifenanzahl realisiert werden. Leider reichte die Zeit nicht aus, um dies tatsächlich zu implementieren.
- Werden zusätzliche Messdaten später zum Projekt hinzugefügt, kann man an Straßen, die auf freier Strecke enden, anknüpfen, ohne die Straße löschen und eine neue, längere Straße erstellen zu müssen. Dies kann sinnvoll sein, weil jedes Löschen einer Straße auch die - eventuell vom Benutzer nachbearbeiteten - Trassierungselemente entfernt.
- Die zugrundeliegende Straße könnte auf freier Strecke verwaltungstechnisch in eine andere Straße übergehen. Dies sollte modelliert werden können.

Abbildung 5.10 zeigt ein Straßennetzwerk, das im Messplan modelliert wurde. Anhand der Darstellung der Straßen ist erkennbar, dass auf ihnen allen Trassierungselemente erkannt wurden. An der Größe der Verknüpfungspunkte der Kreuzungen ist erkennbar, mit welchem Selektionsradius sie erstellt wurden. Insgesamt ist dies ein Beispiel dafür, was mit Messplan 1.1 möglich ist.

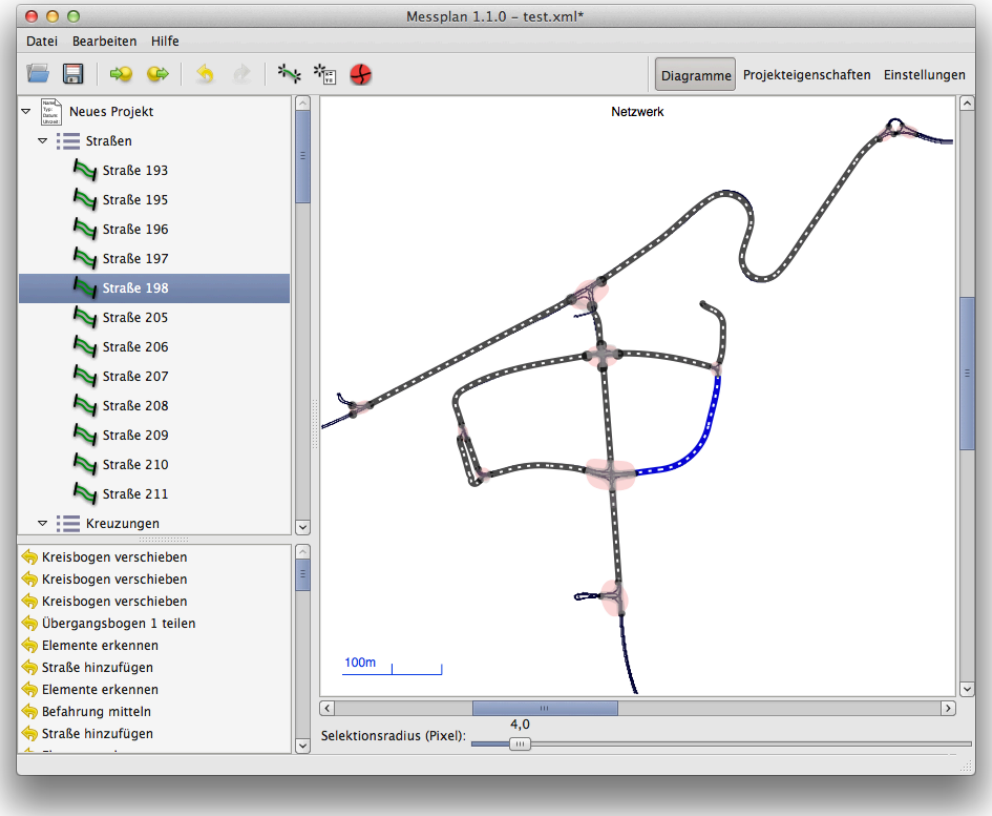


Abbildung 5.10: In Messplan modelliertes Straßennetzwerk

Kapitel 6

Fazit und Ausblick

6.1 Bewertung des Resultats

Hinsichtlich der Anforderungen der Zielsetzung ist die erweiterte Messplan-Version relativ mager an neuen Funktionen. Die Zeit reichte mir nicht, um auf der Basis der Grundlagenfunktionalität an den weitergehenden Anforderungen zu arbeiten. Ich habe den Aufwand für die Implementierung der Grundfunktionalität deutlich unterschätzt.

Die neue Funktionalität enthält keine neuen Algorithmen zur halbautomatischen Erkennung von Kreuzungen auf der Basis der Messdaten. Auch fehlt die Möglichkeit, manuell oder automatisch die Anzahl von Fahrstreifen für einen Streckenabschnitt festzulegen. Gerade die automatische Erkennung derartiger Merkmale wäre interessant gewesen.

6.2 Ansätze zur Erweiterung

6.2.1 Benutzerfreundlichkeit

An verschiedenen Stellen kann die bestehende Funktionalität von Messplan im Hinblick auf Benutzerfreundlichkeit verbessert werden. Beim Testen sind vor allem folgende Probleme aufgefallen:

- Nach der Erstellung einer Straße verändert sich der Bildausschnitt so, dass er die gesamten Messdaten zeigt. Dies ist unnötig und irritiert den Benutzer. Bei der Implementierung versuchte ich, das Problem zu beheben, fand jedoch die Ursache nicht und gab es zugunsten der Implementierung der Grundfunktionalität auf. Die hauptsächliche Schwierigkeit besteht darin, die Stelle im Code zu finden, die geändert werden muss. Da sie durch eine Verkettung von Events aus dem Modell heraus aufgerufen wird, ist es nicht trivial, den Programmfluss von der Änderung im Modell bis zur fraglichen Stelle nachzufollziehen.
- Der Straßeneditor arbeitet auf dem kompletten Messdatensatz statt nur auf dem Teil, der die Straße darstellt. Dies war schon in Messplan 1 so. Bei der Bearbeitung von Elementen spielen die Messdaten insofern eine Rolle, dass neue Elemente, die eingefügt werden, ihren Start- und Endpunkt auf den nächstgelegenen Messpunkt setzen. Dabei wird von der Position ausgegangen, die dem Punkt auf einem beziehungsweise zwei Dritteln der Länge des Übergangsbogens, auf dem das Element eingefügt wird, entspricht. Handelt es sicht nun um grob ungenaue Übergangsbögen (wie etwa in Abbildung 5.8), so kommt es vor, dass Start- und Endpunkt des Elements, das neu hinzugefügt wird, außerhalb der Messpunkte liegen, die zur Straße gehören. In Messplan 1 fiel das kaum auf, da vor allem auf Streckenabschnitten ohne Kreuzungen gearbeitet wurde. Im Kreuzungsbereich tritt der Fall viel häufiger auf.
- Zusätzlich zum vorherigen Punkt stellt sich auch die Frage, ob der Straßeneditor überhaupt alle Messpunkte anzeigen sollte. Wenn ausschließlich die Befahrungen der Straße angezeigt werden, geht der Kontext der Straße - vor allem bei Kreuzungen - verloren, aber man könnte

den Bildausschnitt etwa auf ein Rechteck begrenzen, das doppelt so groß ist wie das kleinste die Straße entaltende Rechteck.

- Kreuzungen könnten bei der Erstellung die etwaig vorhandenen Verknüpfungspunkte vorhandener Straßen benutzen. Dies geht bisher nicht aufgrund einer Unzulänglichkeit im Datenmodell: Wenn eine Straße erstellt wird, wird in ihren freien Enden jeweils ein Verknüpfungspunkt erstellt, an den nur eine andere Straße angeschlossen werden kann. Soll eine Kreuzung angeschlossen werden, muss dieser Punkt durch einen Verknüpfungspunkt der Kreuzung ersetzt werden. Dies sollte mit relativ geringem Aufwand implementierbar sein.

- Alle Änderungen im Straßeneditor werden sofort im Datenmodell des Projekts vollzogen und sind damit auch im Hauptfenster sichtbar. Der Straßeneditor ist ein Dokumentenfenster, und üblicherweise verhält sich ein Dokumentenfenster so, dass alle Änderungen am Dokument (in diesem Fall der Straße) erst nach einer expliziten Speicheraktion übernommen werden. Das Problem an einer solchen Änderung wäre, wie man in diesem Fall mit der Aktionshistorie umgeht, die dann für beide Fenster getrennt werden müsste. Ich bin mir unsicher, ob sich der Aufwand lohnt.

- Das Importieren von Messpunkten kann nicht rückgängig gemacht werden. Das liegt daran, dass eine Löschung zeitaufwändig wäre, weil nicht einfach der gesamte Container, der die Messpunkte enthält, weggeworfen werden kann; zumindest in dem Fall, wenn mehrere Sätze von Messdaten hintereinander importiert wurden. Im Hinblick auf die Konsistenz der Benutzeroberfläche wäre es dennoch konsequent, eine solche Funktion zu implementieren.

- Im Netzwerkdiagram können weder Straßen noch Kreuzungen durch Anklicken ausgewählt werden. Für Trassierungselemente ist diese Funktion bereits implementiert. Straßen, die bereits Trassierungselemente enthalten, könnten diese Funktion nutzen, sodass beim Klick auf eines ihrer Trassierungselemente die Straße ausgewählt wird. Für Straßen, die keine Trassierungselemente enthalten, kann die Suche nach dem nächsten Messpunkt an einer Koordinate verwendet werden, die auch bei der Erfassung von Straßen und Kreuzungen benutzt wird. Für Kreuzungen müsste ein neuer Algorithmus auf der Basis der angezeigten Fläche geschrieben werden. Auch wäre zu überlegen, ob die Pfade einer Kreuzung einzeln anwählbar sein sollen.

6.2.2 Neue Funktionalitäten

Da viele in der Zielsetzung vorgestellte Erweiterungsmöglichkeiten nicht umgesetzt wurden, lassen sich daraus Möglichkeiten zur Erweiterung von Messplan schöpfen. Nachfolgend werden diese Möglichkeiten im Rahmen des Standes von Messplan 1.1 bewertet und Ansatzpunkte zur Implementierung vorgestellt.

6.2.2.1 Veränderung des Start- und Endpunktes von Straßen

Start- und Endpunkt von Straßen können, nachdem sie angelegt wurden,nicht mehr verändert werden. Wie in Abschnitt 5.4 beschrieben, ist dies eine Einschränkung, die in Messplan 1.1 obsolet ist, da die Trassierungselemente keine Kindelemente der Befahrungen mehr sind. Es wäre daher sinnvoll, dem Benutzer zu ermöglichen, die Start- und Endpunkte zu verschieben, um die zur Verfügung stehenden Bearbeitungsfunktionen zu vervollständigen. Bei der Implementierung dieser Funktion sind folgende Punkte zu beachten:

- Existieren Trassierungselemente, so müssen deren Start- und Endpunkt ebenfalls verschoben werden. Üblicherweise sind das erste und das letzte Element jeweils ein Übergangsbogen, sodass dieser nur neu berechnet werden muss. Es ist jedoch auch möglich, dass die Elemente Geraden oder Kreisbögen sind. In diesem Fall sollte überprüft werden, ob diese Elemente bis zur neuen Position erweitert werden können oder ob sie dann zu weit von den Messpunkten abweichen. Dazu sollte die entsprechende Funktionalität aus der Elementerkennung benutzt werden. Eventuell muss dann ein neuer Übergangsbogen zwischen Element und Start- beziehungsweise Endpunkt der Straße eingefügt werden.

- Die Befahrungen müssen neu berechnet werden. Es kann vorkommen, dass eine Befahrung hinzu kommt oder heraus fällt. Dies wäre allerdings ein Anzeichen dafür, dass die Straße über einen Bereich fährt, der eigentlich einen Knotenpunkt darstellt. Wird dies erkannt, könnte der Benutzer darüber benachrichtigt werden.
- Gemittelte Befahrungen können entweder einfach gelöscht werden oder es muss intern eine Abbildung von den alten auf die neuen Befahrungen stattfinden, sodass automatisch eine neue Mittelung stattfinden kann, die die neuen Abschnitte benutzt, auf die die alten Abschnitte, die ursprünglich gemittelt wurden, abgebildet wurden. Eine Abbildung von alten auf neue Abschnitte ist relativ trivial, da nur der *Distance*-Wert der enthaltenen Messpunkte verglichen werden muss. Dieser ist über die gesamten Messpunkte monoton steigend, wodurch Befahrungen nicht verwechselt werden können.

Die Möglichkeit der Veränderung von Start- und Endpunkten von Straßen ist notwendige Voraussetzung für die Veränderlichkeit der Verknüpfungspunkte von Kreuzungen: Da Kreuzungen Pfade enthalten, die als Straßen modelliert sind, sind die Verknüpfungspunkte gleichzeitig Start- oder Endpunkte von Straßen. Dasselbe gilt auch für Straßen, die in die Kreuzung einmünden.

6.2.2.2 Automatische Erkennung von Kreuzungsbereichen

Ein Ansatz, um Kreuzungsbereiche automatisch erkennen zu können, ist, einen Abschnitt zu finden, an dem mehrere Befahrungen in dieselbe Richtung beziehungsweise in die Gegenrichtung verlaufen. Unter der Voraussetzung, dass es sich am erkannten Abschnitt um eine Straße handelt, kann er verfolgt werden, bis mindestens eine Befahrung soweit von den Anderen abweicht, dass davon ausgegangen werden kann, dass es sich um einen Knotenpunkt handelt, Durch Nachverfolgung aller Befahrungen und Erkennung weiterer Befahrungen, mit denen die ursprünglichen Befahrungen wieder in einen straßenartigen Abschnitt übergehen, kann der Knotenpunkt abgegrenzt werden.

Die Probleme dieses Ansatzes sind im Wesentlichen:

- Die Breite der Straßen variiert, sodass es schwierig ist, einen Parameter zu finden, um zu definieren, was der maximale Abtand zwischen Befahrungen ist, sodass diese als eine Straße erkannt werden. Straßen mit mehreren Fahrbahnen wie etwa Bundesstraßen oder Autobahnen sind hier das Extrem. In Abbildung 5.7 beispielsweise ist eine Zubringerstraße zu einer zweibahnigen Bundesstraße zu sehen, die im erfassten Kreuzungsbereich noch nicht in die Bundesstraße übergeht, wie auch im Abschnitt 5.3.1 erläutert. Ohne Zusatzinformation ist dies für einen Algorithmus unmöglich aus den vorliegenden Messdaten zu erkennen.
- Im Bezug auf die erwähnte Abbildung ist auch die Frage, welche Parameter einstellbar sein müssten, damit der Benutzer einstellen kann, ob ein solches komplexes Gebilde als eine große Kreuzung (wie in der Abbildung) oder mehrere kleine Kreuzungen erkannt wird. Denkbare Parameter wären etwa die Länge der Strecke, die mehrere Befahrungen nebeneinander herlaufen müssen, damit sie als Straße außerhalb der Kreuzung erkannt werden, oder die maximale Größe einer Kreuzung.
- Wichtig ist auch, dass nur Strecken als Straßen erkannt werden, die außerhalb des geographischen Kreuzungsbereichs liegen. Wird dies nicht implementiert, so läuft man Gefahr, dass etwa einspurige Zubringerstraßen aufgrund ihrer Länge als eigene Straßen erfasst werden, obwohl man auch komplexe Knotenpunkte als eine einzige Kreuzung erfassen möchte.

Damit eine automatische Erkennung von Kreuzungen überhaupt sinnvoll ist, muss es eine Möglichkeit geben die Kreuzungen nachzubearbeiten. Dazu muss es vor allem möglich sein, die Verknüpfungspunkte zu verschieben. Dies hat, wie im vorherigen Abschnitt beschrieben, die Voraussetzung, dass Start- und Endpunkt von Straßen verschoben werden können müssen.

Insgesamt ist der Aufwand, den eine automatische Erkennung von Kreuzungen verursacht, im Verhältnis zum Nutzen relativ hoch. Die Entwicklung eines entsprechenden Algorithmus scheint allenfalls für den Fall, dass man gesamte Straßennetze vollautomatisch erfassen will, interessant

zu sein. Im Rahmen von Messplan bringt es nur wenig Mehrwert, da das manuelle Erfassen von Kreuzungen für überschaubare Straßennetze (bis zu 20 Kreuzungen) nur wenig Zeit in Anspruch nimmt.

6.2.2.3 Integration von Luftbildern

Die Integration von Luftbildern macht es möglich, dem Benutzer durch die Einbindung von zusätzlichen Informationen die Erfassung von Straßen und Kreuzungen zu erleichtern. Auf Luftbildern sind insbesondere Bauten wie etwa Abgrenzungen zwischen Straßen und Fahrbahnen sichtbar, was die Entscheidungsfindung, welche Befahrungen zu einer Straße gehören, wo eine Zubringerstraße in die Zielstraße übergeht und so weiter, einfacher macht.

Es gibt mehrere Möglichkeiten, Luftbilder in Messplan einzubinden. Zwei werden hier vorgestellt:

- Manuelles Laden von Luftbildern, die der Benutzer bereitstellt. Der Benutzer muss auf dem geladenen Luftbild, bevor es hinder die Messdaten gelegt werden kann, zwei Referenzpunkte festlegen, deren Koordinaten er kennt. Dies wäre über einen Importdialog für Luftbilder möglich. Sobald er mit der Maus zwei Punkte auf dem Bild ausgewählt und deren Koordinaten eingegeben hat, kann das Bild eindeutig hinter die Messpunkte gelegt werden.
- Automatisches Laden von Luftbildern aus einer frei verfügbaren Quelle. Als Beispiel sei Google Maps [1] genannt, das eine Schnittstelle bietet, mit der man Satellitenbilder in gewünschter Größe mit gewünschtem Zoomfaktor abrufen kann. Die Schnittstelle ist unter [2] definiert. Um die Schnittstelle benutzen zu dürfen, müssen die Bedingungen für die Benutzung [3] eingehalten werden. Insbesondere bedeutet das, dass Messplan gemäß Abschnitt 9.1 der Bedingungen kostenlos und öffentlich verfügbar sein muss. Aus Benutzersicht ist die Verwendung einer solchen Schnittstelle intuitiver, weil das Laden der Luftbilder automatisch passiert und der Benutzer die Luftbilder nicht erst einpassen muss. Allerdings ist Messplan dann auch an die Qualität dieser Luftbilder gebunden.

Ideal wäre natürlich, beide Möglichkeiten anzubieten. So hat der Benutzer auf jeden Fall ein Luftbild verfügbar, kann es aber ersetzen, wenn ihm bessere Luftbilder zur Verfügung stehen. Beide Optionen sind mit überschaubarem Aufwand umsetzbar, wobei bei der zweiten Option eben bedacht werden muss, dass die öffentliche Verfügbarkeit von Messplan, die umgesetzt werden muss, auch die Erstellung und Unterhaltung einer Webseite oder ähnlichem beinhaltet.

6.2.2.4 Angleichung der Höhenpläne von Kreuzungspfaden

Ziel einer Angleichung von Höhenplänen ist, dass alle Pfade einer Kreuzung auf derselben Ebene verlaufen. Der Höhenplan eines Pfades wird durch seine Höhenplanelemente bestimmt. Diese wurden in der Diplomarbeit bisher kaum erwähnt, da sie für die umgesetzten Funktionen ohne Bedeutung waren. Sie sind eine Funktionalität von Messplan 1 und werden zusammen mit den Lageplanelementen auf einer Straße erkannt.

Bei Kreuzungen, die tatsächlich auf einer Ebene verlaufen, sollten natürlich die erfassten Pfade auch auf dieser Ebene verlaufen. Ein einfacher Ansatz, dies zu realisieren wäre, alle Messpunkte innerhalb der Kreuzung zu mitteln und die mittlere Höhe dann als Fixe Höhe für alle Pfade zu verwenden. dazu muss eventuell die Höhe in den Verknüpfungspunkten der Kreuzung geändert werden. Tatsächlich speichern diese aber nur X- und Y-Koordinate; der Höhenwert existiert nur in den Höhenplanelementen der Pfade. Verändert man den Wert nur dort, kommt es an den Verknüpfungspunkten der Kreuzung zu einem Versatz mit den einmündenden Straßen. Die erste notwendige Änderung wäre also, die Z-Koordinate mit in die Verknüpfungspunkte zu speichern und bei einer Änderung alle angrenzenden Straßen mitzuändern. Dies ließe sich auch im Rahmen der Implementierung der Änderung von Start- und Endpunkten von Straßen (wie in Abschnitt 6.2.2.1 beschrieben) tun.

[1] `https://maps.google.de/`
[2] `https://developers.google.com/maps/documentation/staticmaps/`
[3] `https://developers.google.com/maps/terms`

Für komplexe Kreuzungen wie etwa die in Abbildung 5.7 dargestellte lässt sich dieses Verfahren natürlich nicht anwenden, weil die Pfade der Kreuzung durch Brückenbauwerke unterschiedliche Höhen haben. Eine Implementierung der automatischen Höhenplanangleichung nimmt dem Benutzer also die Entscheidung ab, ob er komplexe Knotenpunkte als eine oder mehrere Kreuzungen erfassen will: Die Annahme, dass alle Pfade einer Kreuzung dieselbe Höhe haben, führt dazu, dass komplexe Knotenpunkte als mehrere Kreuzungen erfasst werden müssen.

Man könnte für Kreuzungen die Option anbieten, die Höhenpläne anzugleichen, ohne den Schritt automatisch bei der Kreuzungserstellung auszuführen. Sofern die Kreuzung dann in einer Art visualisiert wird, mit der der Benutzer erkennt, ob eine solche Angleichung durchgeführt wurde, wäre dies eine Lösung, die dem Benutzer alle Optionen offen lässt.

6.2.2.5 Erfassung von Querschnittsinformationen

Die Erfassung von Querschnittsinformationen von Straßen gehört zu den kompliziertesten Erweiterungsmöglichkeiten von Messplan. Gemäß OpenDRIVE kann eine Straße mehrere Abschnitte haben, die unterschiedlich viele Fahrstreifen haben. Eine Möglichkeit der Unterteilung einer Straße in Abschnitte ist in Messplan bisher nicht vorgesehen. Ausgehend vom aktuellen Stand können allerdings für jeden Abschnitt, dessen Anzahl von Fahrstreifen sich von der des vorherigen Abschnitts unterscheidet, eine eigene Straße angelegt werden. Beim Export können dann aneinanderhängende Straßen zu einer einzigen Straße mit mehreren Abschnitten zusammengefasst werden. Ob dies für die Bedürfnisse des Benutzers ausreicht, hängt vom Anwendungszweck ab.

Deutlich besser wäre es allerdings, die Möglichkeit, eine Straße in Abschnitte zu unterteilen, zu implementieren. Dadurch hätte man die Möglichkeit, bei der Erstellung der Straße eine automatische Unterteilung der Straße in Abschnitte vorzunehmen. Diese automatische Unterteilung sollte einen änderbaren Parameter haben, der die Breite von Fahrstreifen angibt. Idealerweise könnte man die Regelquerschnitte von Straßen, wie sie in [5] definiert sind, auswählen. Dort werden beispielsweise auch Querschnitte definiert, bei denen die einzelnen Fahrstreifen unterschiedliche Breiten haben; so definiert der Regelquerschnitt RQ 35,5 etwa eine sechsspurige Straße mit getrennten Fahrbahnen pro Richtung, bei der die jeweils rechte Spur breiter ist als die übrigen Spuren der Fahrbahn. [4]

Hat man eine Fahrstreifenbreite oder einen Regelquerschnitt ausgewählt, können die Befahrungen der Straße analysiert werden und es kann festgestellt werden, welche Befahrung zu welchem Fahrstreifen gehört, und wann ein Fahrstreifen beginnt oder endet. Voraussetzung dafür ist natürlich, dass alle Fahrstreifen in voller Länge mit dem Messfahrzeug abgefahren wurden. Trivial ist die Entscheidung, in welche Richtung ein Fahrstreifen befahren werden darf: Hier kann man einfach die Richtung der Befahrung übernehmen.

Mit der Erfassung der Querschnittinformationen geht auch die Verknüpfung der einzelnen Fahrstreifen mit Kreuzungspfaden einher. Hat eine Straße mehr als einen Fahrstreifen in eine Richtung, muss beim Anschluss an eine Kreuzung festgelegt werden, welcher Fahrstreifen auf welche Pfade führt. Sobald man für eine Straße die Anzahl von Fahrstreifen und Abschnitte festlegen kann, müssen auch die Datenstrukturen für Straßen und Kreuzungspfade getrennt werden, denn ein Kreuzungspfad hat immer genau einen Fahrstreifen und einen Abschnitt. Entsprechend muss auch der Straßeneditor angepasst werden, sodass er für Straßen andere Optionen bietet wie für Kreuzungspfade.

Insgesamt geht mit der Implementierung für Querschnittsinformationen also eine deutliche Erweiterung der Benutzeroberfläche einher. Um realistisch Straßennetzwerke modellieren zu können, ist diese Funktion aber unverzichtbar. Sie zu implementieren wäre nach der erfolgten Implementierung von Kreuzungen die nächste große Erweiterung von Messplan.

[4] Anmerkung: Der zitierte Regelquerschnitt ist durch die Einführung der *Richtlinien für die Anlage von Autobahnen* im Jahr 2008 obsolet geworden. Für die Erfassung von existierenden Straßen ist er natürlich dennoch von Bedeutung.

6.3 Ausblick

Da die Software Messplan 1.1 unter der GNU GPL 2.0 Lizenz zur Verfügung gestellt wird, ist eine Weiterentwicklung problemlos möglich. Ansatzpunkte für die Weiterentwicklung sind, wie im vorigen Abschnitt beschrieben, ausreichend vorhanden. Ein Hindernis für die Weiterentwicklung ist natürlich, dass diese voraussichtlich von jemandem durchgeführt werden würde, der nicht zu den ursprünglichen Autoren gehört - diese haben inzwischen alle ihr Studium beendet oder stehen kurz davor. Eine Einarbeitung in die recht umfangreiche Codebasis von Messplan (21365 Codezeilen ohne Kommentare und Leerzeilen) ist auf jeden Fall ein beträchtlicher Aufwand, insbesondere da es sich um eine Anwendung mit grafischer Benutzeroberfläche handelt, die erfahrungsgemäß schwer zu durchschauen sind.

Sollte das Institut für Straßen- und Verkehrswesen genug Nutzen aus dem Einsatz von Messplan ziehen, ist es trotzdem gut vorstellbar, dass es weitere Projekte zur Erweiterung der Funktionalität geben wird. Durch die Studienprojekte für Softwaretechniker an der Uni Stuttgart sind die Kontakte dafür auf jeden Fall vorhanden.

Literaturverzeichnis

[1] Marius Dupuis et al.: *OpenDRIVE Format Specification, Rev. 1.3*; VIRES Simulationstechnologie GmbH, Rosenheim; August 2010.

[2] F. Deissenboeck, T. Seifert: *Kontinuierliche Qualitätsüberwachung mit ConQAT*; Informatik 2006, Jahrestagung der Gesellschaft für Informatik, Workshop Software-Leitstände, 2006

[3] International Organization for Standardization: *Common Language Infrastructure (CLI)*; ISO/IEC 23271:2012, 2012.

[4] Forschungsgesellschaft für Straßen- und Verkehrswesen: *Richtlinien für die Anlage von Straßen (RAS) Teil: Linienführung (RAS-L)*. FGSV-Verlag, Köln; Ausgabe 1995, Berichtigter Nachdruck 1999.

[5] Forschungsgesellschaft für Straßen- und Verkehrswesen: *Richtlinien für die Anlage von Straßen (RAS) Teil: Querschnitte (RAS-Q)*. FGSV-Verlag, Köln; Ausgabe 1996.

[6] NIMA - National Imagery And Mapping Agency: *Department of Defense World Geodetic System 1984, Its Definition and Relationships With Local Geodetic Systems*; Technical Report, TR 8350.2, 3rd edition; Januar 2000.

[7] Defense Mapping Agency: *The Universal Grids - Universal Transverse Mercator (UTM) and Universal Polar Stereographic (UPS)*; DMA Technical Manual, DMATM 8358.2; September 1989.

[8] Bundesministerium für Verkehr, Bau und Stadtentwicklung, Abteilung Straßenbau, Straßenverkehr: *ASB - Anweisung Straßeninformationsbank, Teilsystem: Bestandsdaten*; Version 2.02, Februar 2011.

[9] Bundesministerium für Verkehr, Bau und Stadtentwicklung, Abteilung Straßenbau, Straßenverkehr: *ASB - Anweisung Straßeninformationsbank, Teilsystem: Netzdaten*; Version 2.02, März 2011.

[10] Schwaber, Ken; Beedle, Mike: *Agile software development with Scrum*; Prentice Hall, 2002.

[11] Oxford Technical Solutions: *NCOM Description*; Document Revision 090820, 2009.

[12] Y. Shafranovich: *Common Format and MIME Type for Comma-Separated Values (CSV) Files (RFC 4180)*; The Internet Engineering Task Force, 2005.

[13] Krausner, Glenn E.; Stephen T. Pope: *A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System* (Report); ParcPlace Systems, Inc., 1988.

[14] GNOME Foundation: *GTK+ 2 Reference Manual*; `http://developer.gnome.org/gtk/stable/`; Abgerufen am 4.10.2012.

[15] Krause, F.; Dilli, J.; Dittrich, C.; Fellger, W.; Pflüger, D.; Prib, M. und Schulz, C.: *Messplan-Handbuch*; Version 1.0, Universität Stuttgart, 2011

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

_____

(Felix Krause)