HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

COMPUTER SCIENCE

INSTITUT
FÜR
SOFTWARE

# Computing Isochrones in Multi-Modal, Schedule-Based Public Transport Networks

## Bachelorarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2011


| | |
|---|---|
| Autor: | Marco Birchler |
| Betreuer: | Prof. Stefan Keller |
| Experte: | Claude Eisenhut, Eisenhut Informatik AG, Burgdorf |
| Gegenleser: | Prof. Dr. Andreas Rinkel |

Computing Isochrones in Multi-Modal,
Schedule-Based Public Transport Networks

Datum Erstellung: 05.10.2011
Letzte Aktualisierung: 22.12.2011

Impressum

Auflagenzahl: 3
Jahr: 2011
Herausgeber: Birchler, Marco
http://www.isochrone.ch

## Abstract

Die Pendlerströme haben sich in den letzten Jahrzehnten in der Schweiz sehr stark erhöht (Kuster & Meier, 2003) und dürften auch in den nächsten Jahren noch weiter zunehmen. Eine Hilfestellung um Wohnort und Arbeitsort, bei gegebenen Aufwand des Arbeitsweges, möglichst optimal wählen zu können, haben Kevin Lynn und Marco Birchler mit ihrer Semesterarbeit im Frühling 2011 geschaffen. Es wurde ein Algorithmus implementiert, der nach dem Erfassen eines Startpunktes, Gebiete markiert, die mit dem öffentlichen Verkehr innerhalb einer gewissen Zeit erreicht werden können. Dies kann eine wichtige Information bei einem Wechsel von Wohn- oder Arbeitsort sein.
In der vorliegenden Bachelorarbeit wurde dieser Algorithmus bezüglich Performance, Funktionalität und Genauigkeit stark erweitert. Durch die Vorberechnung eines Datensatzes konnten neue Funktionen wie z.B. die Visualisierung der durchschnittlichen Reisezeit erreicht werden. Zusätzliche Schnittstellen, welche die Datenabfrage für bekannte Datenformate erlauben, wurden geschaffen. Dadurch kann der Algorithmus in Drittapplikationen eingebunden werden.
Als Beispielanwendung wurde eine interaktive Webseite erstellt, welche die Nutzung des Algorithmus ermöglicht. Nach der Auswahl des Startpunktes, Eingabe der Fahrzeit und der Ausführung der Berechnung werden die Ergebnisse in einer Karte dargestellt. Die Webseite profitiert dabei von der erhöhten Performance und Genauigkeit der Algorithmus-Implementierung, welche in der Bachelorarbeit erreicht werden konnte. Zusätzlich implementierte Funktionen können ebenfalls mit der Webseite getestet werden.

Für weitere Informationen wird dem Leser der Besuch der Webseite empfohlen:
http://www.isochrone.ch

# Management Summary

*Ausgangslage*

Die Schweizer, ein Volk von Pendlern! Rund 20% (770‘000) der Pendler benützen für den Arbeitsweg die öffentlichen Verkehrsmittel (Bahn, Tram, Bus). 53‘000 Personen verbringen sogar täglich mehr als eine Stunde in den öffentlichen Verkehrsmitteln (Bundesamt für Statistik, 2000).
Mit der Semesterarbeit von Kevin Lynn und Marco Birchler im Frühling 2011 wurde der Grundstein für einen Service gelegt, welcher Informationen über Reisezeiten mit den öffentlichen Verkehrsmitteln in der Schweiz liefert. Der Service erhält vom Benutzer einen Startpunkt, eine Startzeit und eine maximale Reisezeit. Mit diesen Parametern und den Fahrplandaten von den SBB und dem ZVV generiert der Service Isochronen. Isochronen sind Linien auf einer Karte, welche eine gleiche Zeit darstellen. Im Falle des Services von Lynn und Birchler bedeutet eine solche Isochrone, dass alle Punkte dieser Linie ab dem Startpunkt mittels öffentlicher Verkehrsmittel gleich schnell erreicht werden können.
Der Titel der Arbeit „Computing Isochrones in Multi-Modal, Schedule-Based Public Transport Networks“ erwähnt ausserdem, dass das betrachtete Netzwerk sowohl „Multi-Modal“ wie auch „Schedule-Based“ ist. Unter „Multi-Modal“ wird verstanden, dass eine Reiseroute nicht auf eine einzige Fortbewegungsart beschränkt ist. Beispielsweise können Fusswege und öffentliche Verkehrsmittel in Kombination genutzt werden. „Schedule-Based“ besagt, dass ein zeitlicher Plan vorhanden ist. Es wird somit beispielsweise unterschieden, ob eine Reise am Mittag oder am Abend angetreten wird.
Der Service, welcher in Anlehnung an den offiziellen Titel auf den Namen „CIPT-Dienst“ getauft wurde, konnte die Ziele der Semesterarbeit erfüllen. Über das Internet aufgerufen lieferte er via standardisierter Schnittstellen (WFS und WPS) die gewünschten Ergebnisse. Das erreichte Ergebnis erfüllte jedoch noch nicht die Erwartungen von Marco Birchler, welcher die ursprüngliche Idee für die Semesterarbeit hatte. Die Berechnung der Isochronen dauerte vielfach zu lange und war noch zu ungenau. Ausserdem gab es noch einige zusätzliche Funktionen, die auf der Wunschliste standen. Motiviert durch Prof. Keller Stefan hat sich Birchler Marco deshalb entschieden, im Rahmen der Bachelorarbeit den CIPT-Dienst weiter zu entwickeln.

*Zielsetzung*

Die Ziele für die Bachelorarbeit setzten sich wie folgt zusammen: Auf der einen Seite galt es, Mängel, welche der CIPT-Dienst nach der Semesterarbeit aufwies, zu beheben. Auf der anderen Seite wurden Ziele definiert, welche dem Dienst neue Funktionalitäten verleihen sollten.
Der grösste Mangel war die ungenügende Performance für Abfragen mit vielen Isochronen und einer hohen maximalen Reisezeit. So benötigte eine Abfrage für 10 Isochronen und einer Reisezeit von 120 Minuten im Durchschnitt 5.7 Sekunden. Eine deutliche Reduzierung dieser durchschnittlichen Rechenzeit wurde deshalb als eines der wichtigsten Ziele bestimmt. Die Modellierung der Umsteigevorgänge und die Berücksichtigung von

Nachbarhaltestellen waren ebenfalls ungenügend. In der Bachelorarbeit sollten deshalb zusätzliche Informationen aus den Fahrplandaten verwendet werden. Damit kann einerseits die effektiv benötigte Zeit eines Umsteigevorgangs bestimmt und anderseits eruiert werden, welche Haltestellen nahe beieinander liegen und somit per Fussweg verbunden werden können.
Als wichtigste zusätzliche Funktion wurde gefordert, dass es möglich sein soll, die durchschnittliche oder minimale Reisezeit ab einem Startpunkt mit Isochronen darzustellen.

*Umsetzung*

Mittels „Profiling"-Messungen bei der Ausführung des CIPT-Dienstes konnte eruiert werden, dass die Vereinigung von Polygonen sehr viel Rechenzeit beansprucht. Solche Operationen werden zwangsläufig für die Generierung der Isochronen benötigt. Es wurde deshalb grosser Aufwand für eine Optimierung dieser Vereinigung betrieben. Durch diverse Änderungen an der Implementierung und dem Wechsel der verwendeten Bibliothek für Geometrie-Operationen konnte die oben erwähnte durchschnittliche Rechendauer von 5.7 Sekunden auf 1.2 Sekunden gesenkt werden. Ein neuartiges Konzept für die Polygonvereinigung, die „Diskrete Polygonvereinigung", wurde ebenfalls entworfen und teilweise implementiert. Eine weitere Entwicklung dieses Verfahrens könnte die Rechenzeit eventuell nochmals verringern.
Damit die durchschnittliche Reisedauer (beispielsweise von Rapperswil nach Zürich) berechnet werden kann, ist es nötig, alle möglichen Reisen zwischen Start- und End-Station zu kennen. Jedes Mal wenn ein Zug die Start-Station verlässt, wird eine neue potentielle Reise ermöglicht. Dies führt dazu, dass für die Durchschnittsberechnung sehr viele Reisen generiert werden müssen. Damit Anfragen an den CIPT-Dienst über die durchschnittliche Reisedauer jedoch innerhalb einiger Sekunden beantwortet werden können, ist es notwendig, all diese Reisen im Voraus zu berechnen. Bei einer Anfrage bezüglich der minimalen Reisedauer ist dies ebenfalls nötig.
Im Rahmen der Bachelorarbeit wurde diese Vorberechnung für alle möglichen und sinnvollen Reisen mit den öffentlichen Verkehrsmitteln in der Schweiz durchgeführt. Dies ergab ungefähr 9 Milliarden Datensätze, wobei jeweils die Start-, Ziel-Station und die Reisezeit abgelegt wurde. Damit diese Vorberechnung effizient durchgeführt werden konnte, wurde ein verteiltes System geschaffen, welches eine parallele Berechnung erlaubte.
Die Attraktivität des CIPT-Dienstes für Web-Entwickler wurde durch die Unterstützung von JSON als neues, zusätzliches Export-Format gesteigert. Ebenfalls wurde die Schnittstelle für den Zugriff per WPS angepasst, damit der Dienst nun reibungslos mit QGis zusammenarbeitet.
Die Implementierung eines intelligenten Cachings der errechneten Resultate erlaubt einen effizienten Zugriff auf bereits getätigte oder ähnliche Abfragen.
Damit diese Neuerungen demonstriert werden können, wurde eine webbasierte Anwendung geschaffen. Mit dieser Anwendung ist es möglich, Anfragen an den CIPT-Dienst zu definieren und zu senden. Drei verschiedene Visualisierungen der Resultate stehen zur Auswahl. Die Ergebnisse werden jeweils auf einer eigenen Ebene über einer Landkarte dargestellt und können komfortabel verwaltet werden. Webentwickler können diese Anwendung als Link- und HTML-Generator nutzen, um den CIPT-Dienst auf ihrer Webseite zu integrieren.

*Weiterentwicklung*

Es wurden in der Bachelorarbeit keine Untersuchungen über das Verhalten des CIPT-Dienstes in Überlastsituationen gemacht. Es kann auch nicht prognostiziert werden, wann eine solche Situation erreicht würde. Bevor eine Integration auf einer vielbesuchten Webseite durchgeführt würde, müssten diese Untersuchungen durchgeführt werden. Weitere Entwicklungen könnten mit der Integration zusätzlicher Datenquellen realisiert werden. Strassen- oder Gewässer-Karten könnten beispielsweise die Visualisierung der Restlaufzeit bei den Endstationen realistischer machen. Eine zusätzliche Performance-Steigerung könnte eventuell durch die Implementation der diskreten Polygonvereinigung erreicht werden. Schliesslich wäre es für eine einfache Integration des Dienstes in bestehende Webseiten sehr wünschenswert, wenn die generierten Karten und Isochronen visuell angepasst werden könnten.

## Inhaltsverzeichnis

## Aufgabenstellung von Prof. Stefan Keller

# Computing Isochrones in Multi-Modal, Schedule-Based Public Transport Networks (Fortsetzungsarbeit)

*Bachelorarbeit von Marco Birchler*
*Abteilung Informatik, Herbstsemester 2011*

Ausgangslage

Das Ziel dieser Fortsetzungsarbeit ist es, eine Applikation zu erstellen, die Isochronenkarten von Reisezeiten mit dem öffentlichen Verkehr (sog. Heat Maps) im Web darstellt. Diese Informationen sollen zudem auch weiteren Applikationen als Webservices zur Verfügung stehen. Die Eingaben (bzw. Parameter) bestehen 1. aus einem Standort (in GPS-Koordinaten), 2. der maximalen Reisedauer und 3. der Anzahl Isochronen und 4. ggf. Startzeit (optional). Die Daten stammen von der SBB und vom ZVV (Stand Fahrplan 2011). Die Daten werden vor einem topografischen Kartenhintergrund dargestellt (z.B. Google Maps, OpenStreetMap).

Aufgabenstellung

Diese Fortsetzungsarbeit soll die bestehende Software verbessern, konsolidieren und erweitern.
*Folgende Aufgaben sollen gelöst werden:*

- Qualität und Performance des Isochronen-Berechnungs-Algorithmus:
  - Algorithmus beschleunigen
  - Evaluation der Lösungsansätze zur Performance-Verbesserung (z.B. Vorberechnen)
  - Algorithmus erweitern (I):
    - Ein ggf. vorgegebene Startzeit durch günstigere ersetzen
    - Umsteigezeit berücksichtigen
    - weitere Verbesserungen und Erweiterungen im Verlaufe der Arbeit
  - Algorithmus erweitern (II): Berechnung der durchschnittlichen Reisezeiten (falls gewünscht, bzw. Startzeit nicht festgelegt)
    - Bedingt Vorberechung
    - Evaluation der Ergebnisse
  - Zusätzliche Webservices:
    - Webentwicklerfreundliches Webservices/API für die Isochronen (Web 2.0-API KML oder GeoJSON)
    - Ev. weitere Webservices (wie z.B. API für Abfahrtstabelle einer Station)
  - Testing mit Desktop GIS QuantumGIS (WPS-Plugin)

- Verschiedenes:
  - Webapplikation als Showcase für Web 2.0-API (ggf. bestehende überarbeiten):
    - Mit den erwähnten Erweiterungen oben.

        - Ggf. anstelle Klick auf Karte, einen Ort als Text eingeben (bedingt Geonamen-Service, wie Google oder geonames.org)
        - Integration KML-API
    - Demo für WPS mit Desktop GIS QuantumGIS
    - Screencast zur Vorstellung der Arbeit (kompatibel zu Youtube-Video in Half-HD Qualität)

Vorgaben

- Die Sprache ist Englisch in Code, Installationsanleitung, allfälliger Benutzerdokumentation, Präsentationsfolien. Sonst deutsch (insbesondere Teil I und II).
- Software und Server: Wie Vorarbeit.

Beteiligte

*Diplomand:*

Siehe oben.

*Projektpartner:*

GISpunkt HSR und Open Source Community.

*Betreuung HSR*

Betreuer: Prof. Stefan Keller, IFS-HSR
Experte: Claude Eisenhut, Eisenhut Informatik, Burgdorf
Gegenleser: Prof. Dr. Rinkel

Projektabwicklung

*Termine:*

- Beginn der Arbeit: Semesterbeginn. Abgabetermin: zwei Wochen nach Semesterende, 12h.
- HSR-Forum sowie weitere Termine: Siehe https://www.hsr.ch/Termine-Diplom-Bachelor-und.5142.0.html (intern).

*Arbeitsaufwand:*

Für die erfolgreich abgeschlossene Arbeit werden 12 ECTS angerechnet. Dies entspricht einer Arbeitsleistung von mind. 360 Stunden.

*Lieferdokumente:*

- Projektdokumentation und Software auf CD
- Website(s)
- Video

*Inhalt der Dokumentation:*

- Die fertige Arbeit muss folgende Inhalte haben:
    1. Abstract, Management Summary, Aufgabenstellung, Eigenständigkeitserklärung
    2. Technischer Bericht
    3. Dokumente der Projektdokumentation
    4. Anhänge (Literaturverzeichnis, CD-Inhalt)

- Die Abgabe ist so zu gliedern, dass die obigen Inhalte klar erkenntlich und auffindbar sind.
- Zitate sind zu kennzeichnen, die Quelle ist anzugeben.
- Verwendete Dokumente und Literatur sind in einem Literaturverzeichnis aufzuführen.
- Projekttagebuch, Dokumentation des Projektverlaufes, Planung etc.
- Weitere Dokumente (z.B. Kurzbeschreibung für Broschüre, Poster, Video) gemäss www.hsr.ch und gemäss Absprache mit dem Betreuer.

*Form der Dokumentation:*
Bericht (Struktur gemäss Beschreibung) gebunden (2 Exemplare) und in Ordner (1 Exemplar „kopierfähig“ in losen, gelochten Blättern).
Alle Dokumente und Quellen der erstellten Software auf CD; CD's sauber angeschrieben (3 Ex.).

*Bewertungsschema*:
Als Bewertungsschema gilt das in www.hsr.ch erwähnte, d.h. die folgenden Aspekte werden bewertet:

- Projektorganisation (Gewicht 1/6)
- Bericht (Gewicht 1/6): Inhalt, Gliederung, Sprache
- Inhalt (Gewicht ½): 1. Vorstudie, Anforderungsanalyse und Domainanalyse; 2. Entwurf (Systemarchitektur, Beschreibung des Entwurfs, Entwurf Benutzerschnittstelle); 3. Realisierung und Test
- Mündliche Prüfung (Gewicht 1/6)

Es gelten ansonsten die üblichen Regelungen zum Ablauf und zur Bewertung der BA-Arbeit des Studiengangs Informatik der HSR.

# Teil 1: Technischer Bericht

Der technische Bericht beschreibt nach einer kurzen Einführung die Ausgangslage für die Bachelorarbeit. In der Evaluation werden diverse Lösungsansätze diskutiert. Die besten Ansätze werden anschliessend in der Realisation umgesetzt. Der erste Teil der Bachelorarbeit wird durch die Präsentation der Ergebnisse und einem Blick auf mögliche Weiterentwicklungen abgeschlossen.

## 1. Einführung

Mit der Vorarbeit von Lynn und Birchler im Frühling 2011 konnte der erste Schritt für die Erfüllung der Vision, wie sie in (Birchler & Lynn, 2011, S. 13) beschrieben wurde, erreicht werden. Gegen Ende der Semesterarbeit wurde jedoch offensichtlich, dass es vorerst beim ersten Schritt bleiben würde und die Vision noch nicht erfüllt werden konnte. Über zwei Millionen ÖV-Verbindungen und etwa 22‘000 Haltestellen, die es bei jeder Abfrage zu berücksichtigen gilt, erfordern eine grosse Rechenleistung. Je nach Wahl der Parameter konnte die durchschnittliche Antwortzeit von einigen Sekunden schnell zu einigen Minuten hochsteigen. Antwortzeiten im Minutenbereich sind jedoch für eine Webanwendung, wie sie in der Vision erwähnt wurde, nicht praktikabel. Sogar Verzögerungen von einigen Sekunden strapazieren die Geduld von Webseitenbesuchern.
Birchler glaubte jedoch nach der Abgabe der Semesterarbeit weiterhin an die Erfüllung der Vision und entschied sich deshalb, im Rahmen der Bachelorarbeit daran weiterzuarbeiten.

### 1.1 Vision

Wie bereits beschrieben, ist die Vision nach wie vor die gleiche wie bei der Semesterarbeit (SA). Es sollte möglich sein, den Webseitenbesuchern die Reisezeit mit den öffentlichen Verkehrsmitteln ab einem bestimmten Punkt in alle Richtungen visuell zu präsentieren. Dieser Dienst soll in bestehende Webseiten eingebaut werden können. Somit könnten beispielsweise Webseiten mit den folgenden Funktionen erstellt werden:

- Ein Besucher möchte eine Filiale einer bestimmten Warenhauskette aufsuchen. Die Webseite der Warenhauskette und die Eingabe seiner Adresse (Startpunkt) ermöglichen ihm sofort die Filiale zu finden, welche mit den öffentlichen Verkehrsmitteln am schnellsten erreicht werden kann.
- Die Hochschule für Technik in Rapperswil möchte wissen, welche Studenten den zeitlich längsten Schulweg haben (unter der Annahme, dass diese den öffentlichen Verkehr benutzen).
- Eine Webseite zeigt auf einer Landkarte, wo es aktuell überall Nebel hat. Die Webseitenbesucher können nach der Erfassung ihres Standorts den schnellsten Weg aus dem Nebel mit den öffentlichen Verkehrsmitteln ermitteln.

## 1.2 Ziele

Der bestehende WPS- und WFS-Dienst (nachfolgend CIPT-Dienst) soll soweit ausgebaut werden, dass damit Anwendungen wie in der Vision beschrieben, erstellt werden können. Insbesondere soll der bereits nach der Semesterarbeit bestehende Dienst in folgenden Punkten erweitert werden:

- Die Reaktionszeit des CIPT-Dienstes (ohne Übertragungszeiten) soll verkürzt werden. Die in der Semesterarbeit (S. 19) gemessene durchschnittliche Reaktionszeit für Reisen von 120 Minuten betrug damals 5.74 Sekunden bei 10 Isochronen und 23.29 Sekunden für 60 Isochronen. Wünschenswert wäre eine Performance-Steigerung um 100%, d.h. eine Halbierung der durchschnittlichen Reaktionszeit.
- Die Korrektheit des Dienstes soll durch die Berücksichtigung der Zeit, die bei Umsteigevorgängen benötigt wird, erhöht werden. Zusätzlich sollen kurze Laufstrecken zwischen benachbarten Haltestellen in das Netzwerk miteinbezogen werden.
- Damit allgemeinere Aussagen als bisher, wie z.B. „Die HSR ist mit den öffentlichen Verkehrsmitteln besser erreichbar als die ETHZ“, visuell dargestellt werden können, muss der CIPT-Dienst tageszeitunabhängige Aussagen liefern können. Es wird beispielsweise die minimale oder durchschnittliche Reisedauer berechnet.
- Wiederkehrende identische Abfragen soll der CIPT-Dienst ohne Neuberechnung der Daten beantworten können. Ein solches Caching ist sehr nützlich, wenn z.B. eine Erreichbarkeits-Karte neben einem Wohnungs-Inserat auf einer Immobilien-Webseite angezeigt wird. Somit muss der CIPT-Algorithmus nur einmal pro neu erfasstem Inserat aufgerufen werden.
- Es soll ein erweitertes Angebot an Formaten für die Auslieferung der Daten unterstützt werden. (Geo-) JSON wurde bisher vom CIPT-Dienst gar nicht unterstützt – ist jedoch ein häufig eingesetztes Format in der Entwicklung von interaktiven Webseiten.
- Eine Webapplikation soll die erweiterten Fähigkeiten des CIPT-Dienstes aufzeigen können.

## 1.3 Abgrenzungen

Wie in der Zieldefinition (1.2) erläutert, soll die Bachelorarbeit die Möglichkeiten eröffnen, den CIPT-Dienst in Drittsysteme einzubinden. Die Palette der gewünschten Einsatzgebiete ist dabei sehr gross. Für einige Szenarien ist der CIPT-Dienst jedoch auch nach dem Update durch die Bachelorarbeit nach wie vor nicht geeignet.

### *1.3.1 Korrektheit des CIPT-Algorithmus*

Die Garantie für eine absolute Realitätstreue der angezeigten Daten kann u.a. auf Grund offensichtlicher Tatsachen nicht gegeben werden. Beispielsweise stellen die Daten der SBB für jeden einzelnen Tag im Jahr einen individuellen Fahrplan zur Verfügung. Der CIPT-Algorithmus operiert allerdings auf den zusammengefassten Daten.

Die Fahrplandaten werden in keiner Weise aktualisiert. Falls also beispielsweise ein Zugabschnitt ausfällt, wird dies vom CIPT-Algorithmus nicht berücksichtigt.
Die verbleibende Zeit bei den einzelnen Ziel-Haltestellen wird als Kreis um die Haltestelle visualisiert. Der Radius des Kreises ist dabei linear abhängig von der verbleibenden Zeit. Es wird die Annahme getroffen, dass man sich bei Erreichen einer Ziel-Haltestelle in jede Richtung mit konstanter Geschwindigkeit weiterbewegen kann, bis die verbleibende Zeit aufgebraucht ist. Dies entspricht bei längeren Restzeiten, zumindest in der Schweiz, nur in seltenen Fällen der Realität.

### *1.3.2 Bezug auf die Schweiz*

Der CIPT-Algorithmus benutzt auch nach dem Update intern das Schweizer Koordinatensystem für Berechnungen. Die Software kann deshalb nicht ohne Anpassungen für andere Länder genutzt werden.
Die Funktion des CIPT-Algorithmus und sämtliche Darstellungen beschränken sich auf die Schweiz.

### *1.3.3 Performance Prognose*

Von einem produktiven Einsatz der Software für eine gut besuchte Internetseite muss zurzeit noch abgeraten werden. Das Verhalten der Software und der Datenbank bei einer Überlastung des Webservers wurde nicht getestet und es können keine Vorhersagen gemacht werden, wann eine solche Überlastsituation eintreffen würde.

# 2. Ausgangslage

Die Ausgangslage für die Bachelorarbeit ist grundsätzlich der Endzustand der Semesterarbeit. Nach der Abgabe der Semesterarbeit bis zum Start der Bachelorarbeit wurden keine weiteren Entwicklungen an CIPT vorgenommen.

## 2.1 Semesterarbeit

Die folgende Tabelle zeigt einen Überblick über die funktionellen Erweiterungen, die mit der Bachelorarbeit erreicht werden konnten.

| Semesterarbeit | Zusätzlich in Bachelorarbeit |
|---|---|
| Berechnung schnellster Reiseweg ab frei gewählter Koordinate und Startzeit | Startpunkt kann auch Adresse sein |
| | Zeit bei Umsteigevorgängen berücksichtigt |
| | Laufdistanzen zwischen Haltestellen berücksichtigt |
| | Allgemeine Berechnung ab Startpunkt ohne Startzeit ermöglichen Aussagen über minimale, maximale und durchschnittliche Reisezeit |
| Generierung einer beliebigen Anzahl Isochronen | Präzision der generierten Isochronen kann angepasst werden |
| | Generierung der Isochronen durch „Bilderkennung“ (Entwicklung nicht abgeschlossen) |
| WPS- und WFS-Schnittstelle | WFS-Schnittstelle wird nicht mehr unterstützt, WPS erweitert und damit funktionsfähig mit QGIS |
| Datenexportformat: GML, KML | GeoJSON |
| Webbasierter Testclient (Google Map API) | Erweiterter Client (OpenLayers API, inkl. Google) |
| | Layer-Verwaltung mit dem Web-Client |
| | Exportmöglichkeiten und Link-Generierung |
| Einfache Website mit wichtigen Informationen | Erweiterte Webseite |

**Tabelle 1: Funktionsvergleich Semester- und Bachelorarbeit**

Zusätzliche nichtfunktionale Erweiterungen und Änderungen werden im Verlauf dieser Arbeit erwähnt und mit dem Endstand der Semesterarbeit verglichen.

## 2.2 Fahrplandaten

Als Datengrundlage werden, wie in der Semesterarbeit, die Fahrplandaten 2010 / 2011 der SBB und des ZVV verwendet. Es wurden keine Versuche unternommen, um die Daten für den kommenden Fahrplanwechsel zu beschaffen. Somit werden die Daten zum Zeitpunkt der Fertigstellung dieser Bachelorarbeit veraltet sein. Wie in (Birchler & Lynn, 2011, S. 20)

beschrieben, müssen die Rohdaten zuerst mit Hilfe eines Import-Tools extrahiert und in eine Datenbank geladen werden. Da der CIPT-Client nach dem Update zusätzliche Daten (Umsteigevorgänge, Laufdistanzen) benötigt, wurde das Import-Tool angepasst, damit auch diese geladen werden können.

### 2.3 Performance

Als Basis für die Performancemessungen gelten die Messungen die in (Birchler & Lynn, 2011, S. 19) dokumentiert sind.

## 3. Evaluation

Das folgende Kapitel dient der systematischen Evaluation von unterschiedlichen Möglichkeiten, um die gesteckten Ziele (1.2) der Bachelorarbeit zu erreichen. Das Kapitel „Realisierung“ erläutert danach, welche der Möglichkeiten tatsächlich umgesetzt wurden.

### 3.1 Algorithmus

Unter Algorithmus wird der Programmcode verstanden, der durchlaufen wird, um eine Anfrage an den CIPT-Dienst zu beantworten. Der Algorithmus wird grob in zwei Teile unterteilt. Der erste Teil hat die Aufgabe, eine Liste aller Haltestellen zu berechnen, die vom Startpunkt aus innerhalb eines Zeitlimits erreicht werden können. In der Semesterarbeit wurde dieser erste Teil als CIPT-Algorithmus oder modifizierter Dijkstra-Algorithmus bezeichnet. Der zweite Teil berechnet aus dem Ergebnis des CIPT-Algorithmus die Isochronen-Daten. Diese Daten können dann von einem Client direkt visuell dargestellt werden. Die folgenden Unterkapitel in 3.1 evaluieren mögliche Verbesserungen am Algorithmus, um die gesetzten Ziele zu erreichen.

#### *3.1.1 Performance-Steigerung*

Ein wichtiges Ziel der Bachelorarbeit ist die Erhöhung der Performance des CIPT-Dienstes. Eine einzelne Anfrage soll im Durchschnitt schneller beantwortet werden können. Dies bedeutet auch, dass in einer Zeiteinheit mehr Anfragen beantwortet werden. In den folgenden Kapiteln werden verschiedene mögliche Ansätze für das Erreichen dieses Ziels betrachtet.

##### Algorithmus Optimierung

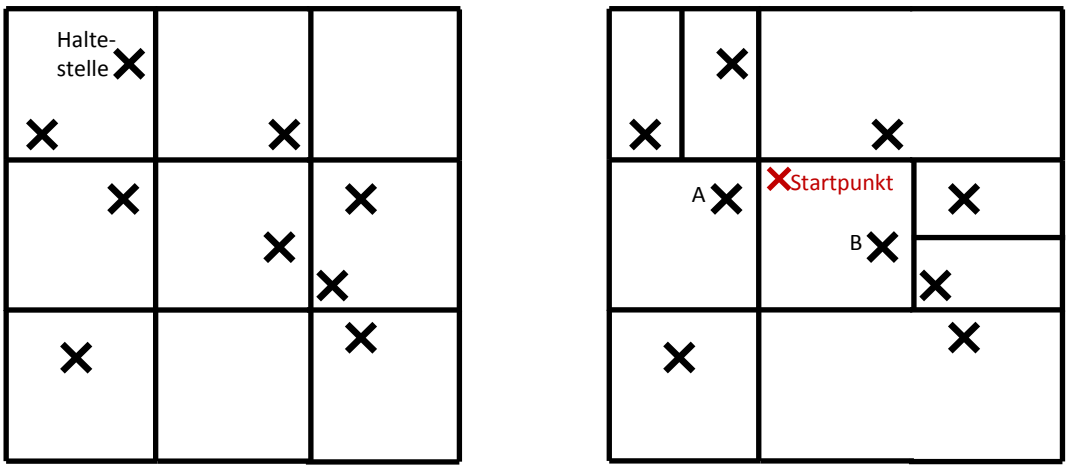
Bei jedem Aufruf des CIPT-Dienstes durchsucht der CIPT-Algorithmus ein Netzwerk mit 22‘000 Knoten. Zusätzlich handelt es sich dabei noch um ein zeitabhängiges Netzwerk (Time-Dependant). D.h. die Vernetzung der Knoten ist für verschiedene Startzeiten unterschiedlich. Angesichts dieser Tatsachen ist es erstaunlich, dass das Ergebnis überhaupt innerhalb einiger Sekunden bereitgestellt werden kann.
Der in der Semesterarbeit erstellte Algorithmus ist ein auf die Einsatzbedürfnisse optimierter Dijkstra-Algorithmus. In verschiedenen Quellen wie z.B. (Delling & Wagner, S. 4) wird

erwähnt, dass es einige bessere Algorithmen für das Suchen der schnellsten Route von A nach B gibt (z.B. der A* Algorithmus). Dies ist jedoch unerheblich für den CIPT-Algorithmus, da dieser das Ziel nicht kennt, resp. das Ziel alle Haltestellen sind, die in der maximalen Reisezeit erreicht werden können. Der Algorithmus kann also nicht zielorientiert arbeiten wie dies der A*-Algorithmus macht. Der bereits verwendete Dijkstra-Algorithmus gehört zu den effizientesten Algorithmen für die Berechnung der schnellsten Route vom Startpunkt zu allen möglichen Zielen (Geisberger, 2011, S. 22).

### Vorberechnung

In diesem Kapitel wird untersucht, ob durch eine Vorberechnung der CIPT-Abfragen die Performance verbessert werden kann. Insbesondere wird dabei auf die anfallende Datenmenge eingegangen, da dies ein wichtiger limitierender Faktor ist. Der Berechnung der Datenmenge liegt der aktuelle Fahrplan von SBB und ZVV zugrunde, welcher ca. 22‘000 Haltestellen und 2‘900‘000 Fahrten enthält. Durch die Vorberechnung kann die Performance des CIPT-Dienstes eventuell gesteigert werden, da bei einer Anfrage jeweils nicht der ganze CIPT-Algorithmus durchlaufen werden muss.

Die umfangreichste Möglichkeit der Vorberechnung wäre, das gesamte Resultat einer CIPT-Abfrage für jeden Startpunkt in der Schweiz im Voraus zu berechnen und in der Datenbank abzulegen. Es würde sich also um eine Liste handeln, die alle Reisezeiten vom Startpunkt zu allen Haltestellen in der Schweiz enthält. Eine solche Liste für einen Startpunkt wird in der weiteren Betrachtung als Datensatz bezeichnet.
Da eine Vorberechnung für jeden Punkt utopisch ist, könnten mehrere Punkte zu einer Fläche zusammengefasst werden und eine Vorberechnung für jeweils eine Fläche gemacht werden. Als Flächenform würde sich das Rechteck anbieten, welches mathematisch einfach behandelt werden kann.
Die Wahl der Grösse des Rechtecks ist entscheidend. Je kleiner die Fläche, desto mehr Rechtecke werden benötigt um die ganze Schweiz abzudecken und die Anzahl der zu berechnenden Datensätze steigt. Wird die Fläche grösser gewählt, reduziert sich die Anzahl Datensätze. Dafür muss man allerdings eine reduzierte Genauigkeit der Ergebnisse in Kauf nehmen.
Ein Kompromiss wäre, die gesamte Fläche in unterschiedlich grosse Rechtecke zu unterteilen. Dicht besiedelte Gebiete erhalten kleinere Rechtecke und somit exaktere Lösungen. Die Siedlungsdichte kann anhand der vorhandenen Haltestellen-Koordinaten in den HAFAS-Daten simuliert werden. Die Kombination von Haltestellen-Daten und Flächen führt zwangsläufig zu folgendem Lösungsansatz: Die Flächen werden so aufgeteilt, dass in jede Fläche genau eine Haltestelle zu liegen kommt.
Ein systematisches Vorgehen wäre, die ganze Schweiz zu Beginn als einzige Fläche zu betrachten. Nun wird diese Fläche unterteilt. Die dabei entstehenden kleineren Flächen werden immer wieder unterteilt, bis sich in jeder Fläche nur noch eine Haltestelle befindet.




Konstante Flächengrösse

Unterschiedliche Flächengrösse
(maximal eine Haltestelle pro Fläche)

**Abbildung 1: Vorberechnung der Resultate pro Kachel**

Dieser Lösungsansatz führt dazu, dass pro Haltestelle ein Datensatz vorberechnet werden kann. Der CIPT-Dienst muss dann die Fussstrecke vom Startpunkt zur Haltestelle in der gleichen Fläche berechnen und dies mit dem vorberechneten Datensatz der Haltestelle kombinieren. Diese Regel würde dazu führen, dass der rot markierte Startpunkt im rechten Bild der Abbildung 1 eine Route via Haltestelle B wählen würde.
Es ist offensichtlich, dass dies nicht zu einer optimalen Lösung führen wird, da die Haltestelle A effektiv näher beim Startpunkt liegt. Damit die Haltestelle A gewählt wird, muss vom Startpunkt aus eine Umkreissuche gemacht werden.
Durch dieses Vorgehen zerfällt der Nutzen der Flächeneinteilung komplett. Eine Suche ab dem Startpunkt für die nächste Haltestelle und das Laden des vorberechneten Datensatzes für die gefundene Haltestelle würde bedeuten, dass der CIPT-Algorithmus bei einer Anfrage nicht mehr benötigt würde und somit das Resultat wohl schneller zur Verfügung steht.

### *Kombination von vorberechneten Datensätzen*

Die genaue Betrachtung des Lösungsansatzes mit der Umkreissuche und dem Laden des vorberechneten Datensatzes für die nächste Haltestelle zeigt jedoch, dass dieser Ansatz nicht in jedem Fall zum gleichen Ergebnis führen würde, wie ein normaler Aufruf des CIPT-Algorithmus. Ist die nächste Haltestelle eine wenig frequentierte Haltestelle, wäre es allenfalls besser, zu Fuss noch weiter zur nächsten Haltestelle zu gehen, welche eventuell besser an den öffentlichen Verkehr angeschlossen ist. Um mit Sicherheit die absolut schnellsten Reiserouten zu finden ist es grundsätzlich nötig, alle Haltestellen, welche zu Fuss erreicht werden können, zu untersuchen. Dazu müssen die vorberechneten Datensätze dieser Haltestellen untereinander verglichen werden. D.h. für jeden potentiellen Zielbahnhof muss eruiert werden, welche Start-Haltestelle gewählt werden muss, um die Reisezeit vom Startpunkt zum Zielbahnhof zu minimieren.
In der praktischen Umsetzung müssten wahrscheinlich nicht alle potentiell zu Fuss erreichbaren Haltestellen betrachtet werden, um ein relativ wirklichkeitsgetreues Resultat zu erhalten. Wo sich ungefähr die Grenze für die Anzahl Haltestellen befindet, müsste durch empirische Tests eruiert werden.

*Speicherbedarf eines Datensatzes*

Ein Datensatz enthält das Ergebnis einer einzigen CIPT-Abfrage. Dies ist eine Start-Haltestelle, eine Start-Zeit und eine Reisezeit-Tabelle von Ziel-Haltestellen (resp. deren ID) mit der Reisezeit von der Start- zur Ziel-Haltestelle. Die Grösse des Sets wird maximal wenn alle anderen Haltestellen des Fahrplans erreicht werden können. Die Reisezeit-Tabelle hätte also ungefähr 22‘000 Einträge.

In einer praktischen Umsetzung müssen alle Reisezeit-Tabellen aller Datensätze der Haltestellen in einer einzigen Master-Reisezeit-Tabelle (MR-Tabelle) kombiniert werden. Um die einzelnen Einträge dieser MR-Tabelle wieder einem Datensatz (Haltestelle) zuordnen zu können, muss eine zusätzliche Verweis-Spalte (Foreign-Key) eingeführt werden.

Ein einzelner Eintrag in der MR-Tabelle besteht also aus der Datensatz-Identifikation (Integer, 4 Byte), der ID der Ziel-Haltestelle (Integer, 4 Byte) und der Reisezeit, welche in Minuten abgelegt werden kann (Small-Integer, 2 Byte). Der Eintrag würde somit rechnerisch 10 Byte benötigen. Dieser rechnerische Ansatz lässt den Speicherbedarf für die Verwaltung der Daten ausser Acht. Dass dieser Overhead beträchtlich ist, zeigt Tabelle 2 sehr deutlich. Bei einer Testtabelle mit 14‘000 Einträgen benötigt ein einzelner Eintrag im Durchschnitt 44.47 Byte.
Die Entwicklung der Grösse eines Eintrags in Abhängigkeit von der Anzahl Einträge lässt vermuten, dass sich die Grösse eines Eintrags auch bei einer höheren Anzahl nicht mehr nennenswert senken wird.

| Anzahl Testeinträge | Grösse der Tabelle (Kilo Byte) | Grösse der Tabelle (Byte) | Grösse eines Eintrags (Byte) |
|---|---|---|---|
| **1000** | 48 | 49152 | 49.152 |
| **2000** | 88 | 90112 | 45.056 |
| **6000** | 264 | 270336 | 45.056 |
| **14000** | 608 | 622592 | 44.470 |

**Tabelle 2: Speicherbedarf einer PostgreSQL-Tabelle**

Der Einfachheit halber wurde bis anhin angenommen, dass pro Haltestelle ein Datensatz besteht. In Wirklichkeit generiert jedoch jede einzelne Abfahrt bei einer Haltestelle einen neuen Datensatz. Vermutlich können bei einer Startzeit um 6.00 Uhr ab Rapperswil praktisch alle anderen Haltestellen in der Schweiz noch am gleichen Tag erreicht werden. Wenn um 18.00 Uhr in Rapperswil gestartet wird, können viele Haltestellen nicht mehr erreicht werden. Der Datensatz für diese Abfahrt ist daher kleiner. Es ist anzumerken, dass nicht für jede Minute des Tages ein Datensatz pro Haltestelle nötig ist. Verlässt beispielsweise um 8.00 Uhr ein Zug Rapperswil und der nächste fährt um 8.10 Uhr, kann von 8.01 Uhr bis 8.10 Uhr der gleiche Datensatz benutzt werden.

Die Anzahl täglicher Fahrten in der Schweiz multipliziert mit der maximalen Anzahl Einträge in einem Datensatz multipliziert mit der durchschnittlichen Grösse eines Eintrags ergibt die theoretisch maximale Grösse der MR-Tabelle. Die effektive Grösse wird kleiner sein, da nicht mit jeder Fahrt am gleichen Tag noch alle anderen Haltestellen der Schweiz erreicht werden können.

| Fahrten in der Schweiz | Einträge in Datensatz | Grösse eines Eintrags (Byte) | Grösse aller Einträge (Byte) | Grösse aller Einträge (Giga-Byte) |
|---|---|---|---|---|
| 2‘900‘000 | 22‘000 | 44.47 | 2‘837‘186‘000‘000 | 2642 |

Tabelle 3: Maximaler Speicherbedarf der MR-Tabelle

Damit in der grossen MR-Tabelle ein effizientes Suchen möglich wird, ist es nötig, einen Index über die Spalte mit der Datensatz-Identifikation zu erstellen. Der Speicherbedarf eines Indexes für eine solch grosse Tabelle ist erheblich und darf nicht vernachlässigt werden. Dennoch kann man davon ausgehen, dass die Grösse des benutzten B-tree-Index nur im schlechtesten Fall gleich gross wird, wie die Tabelle selbst (Wikipedia - B-tree, 2011).

Der tatsächliche Speicherbedarf für die Vorberechnung, welcher sich aus der effektiven Grösse der MR-Tabelle und dem Index zusammensetzt, würde somit auf einen Wert zwischen 2 bis 5 Terabyte zu stehen kommen. Festplatten mit mehreren Terabyte Speicherkapazität sind aktuell auf dem Markt erhältlich. Somit ist eine Vorberechnung tatsächlich umsetzbar.

#### *Effektiver Geschwindigkeitsvorteil*

Ist die Lösung mit der Vorberechnung nun tatsächlich schneller als eine Just-in-Time Berechnung? Immerhin muss eine riesige Tabelle in der Datenbank (Sekundär-Speicher) durchsucht werden. Die bisherige Lösung operierte ausschliesslich mit Daten im Primär-Speicher, war jedoch extrem viel rechenintensiver.
Es kann an dieser Stelle keine abschliessende Antwort zu einem allfälligen Geschwindigkeitsvorteil gegeben werden. Denn die beiden Lösungsansätze unterscheiden sich sehr stark und sind ausserdem von verschiedener Hardware abhängig. Daher würde sich die Performance-Differenz zwischen den beiden Lösungen auf verschiedenen Systemen mit unterschiedlicher Hardware eventuell stark ändern.

Ein weiteres Ziel der Bachelorarbeit, die Visualisierung der durchschnittlichen Reisezeit, kann nur durch die Vorberechnung der Daten umgesetzt werden. Allein schon aus diesem Grund muss die Vorberechnung umgesetzt werden und es ist für die Evaluation unerheblich, ob mit dieser Lösung tatsächlich ein Performance-Vorteil erzielt werden kann.

### Isochronen-Generierung

Performance-Messungen beim Ausführen des CIPT-Programmcodes haben ergeben, dass das Generieren einer Isochrone aus dem Ergebnis des CIPT-Algorithmus einen grossen Einfluss auf die gesamte Performance des CIPT-Dienstes hat. Jede Erhöhung der gewünschten Anzahl Isochronen erhöht auch den Rechenbedarf für die Generierung. Diese Abhängigkeit ist linear.
Ein komplett neuer Ansatz wäre, serverseitig auf die Generierung der Isochronen zu verzichten. Der CIPT-Dienst würde dann nur die Liste aller Stationen, die erreicht werden können, als Punkte liefern. Zu jedem Punkt wird ausserdem die restliche Laufzeit mitgeliefert. Die grafische Aufbereitung dieser Daten müsste dann der Browser

übernehmen. Falls keine exakten Grenzen benötigt würden, sondern das Ergebnis beispielsweise durch eine Punktewolke präsentiert werden soll, würde dies bereits genügen. Möchte man jedoch exakt feststellen können, ob ein bestimmter Punkt ab dem Startpunkt innerhalb der Reisezeitlimite erreicht werden kann, müssen diese Isochronen zwangsläufig erstellt werden.

### *Geometrische Polygonvereinigung*

Ein detailliertes Profiling über die Programmausführung bei der Isochronen-Generierung hat ergeben, dass die Vereinigung einzelner Polygone, zu einem Multipolygon für den CIPT-Dienst rechenintensiv ist. Dies ist ein wichtiger und unverzichtbarer Zwischenschritt für die Erstellung der Isochronen. In der Semesterarbeit wurde diese Vereinigung als ein iteratives Verfahren implementiert. Dabei werden zuerst die ersten beiden Polygone miteinander vereinigt. Im nächsten Schritt wird dann das Ergebnis des vorherigen Schrittes mit dem nächsten Polygon vereinigt. Dieser Schritt wird solange wiederholt bis alle Polygone zu einem einzigen Multipolygon vereinigt wurden.

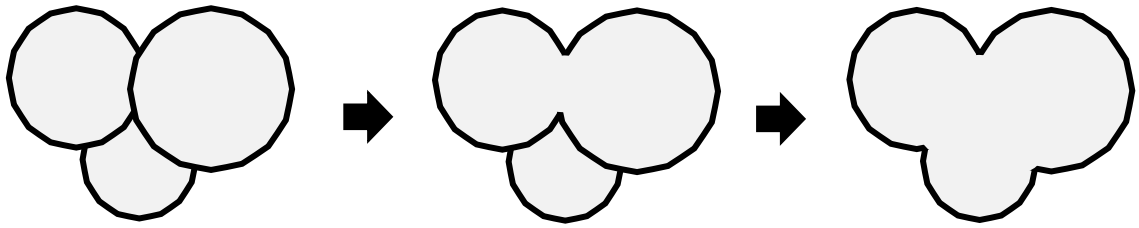


**Abbildung 2: Iterative Vereinigung von Polygonen**

In den letzten Jahren wurde jedoch eine neue Art der Polygonvereinigung bekannt, welche wahrscheinlich erstmals durch Martin Davis implementiert wurde (Davis, 2007) und von ihm „Cascaded Union“ genannt wurde. Diese Methode lädt zuerst alle Polygone in eine Baumstruktur. Der eigentliche Algorithmus arbeitet sich dann im Baum von unten hoch und vereinigt jeweils zwei Kindknoten. Der Performance-Vorteil beruht darauf, dass nur wenige Vereinigungen mit grossen, komplexen Polygonen durchgeführt werden müssen.
Die in der Semesterarbeit benutzte Bibliothek für Operationen mit Polygonen (Microsoft.SqlServer.Types) unterstützt leider diese Art der Vereinigung nicht. Mit der Nettopology Suite steht jedoch eine alternative zur Microsoft-Bibliothek aus der Open-Source Welt zur Verfügung, welche die „Cascaded Union“ Methode unterstützt.
Eine weitere Methode, um eine Sammlung von Polygonen zu vereinigen, wird in diversen Quellen wie z.B. (Davis, Secrets of the JTS Topology Suite, S. 11) erwähnt. Dabei wird ein Buffer um die vorhandene geometrische Struktur gelegt, welcher im Spezialfall der Vereinigung einen Abstand von 0 haben muss. Je nach Struktur der Polygone kann diese Methode jedoch zu falschen Ergebnissen führen.

| Methode | Bibliothek | Gemessene Zeit für 1940 Polygone | Zeit pro Polygon |
|---|---|---|---|
| Cascaded Union | Nettopology Suite Version 1.11.4307 | 8s | 4.1ms |
| Buffer (0) | Nettopology Suite Version 1.11.4307 | 36s | 18.6ms |
| Iterative Union | Nettopology Suite Version 1.11.4307 | 143s | 73.7ms |
| Iterative Union | Microsoft.SqlServer.Types 10.0 | 13s | 6.7ms |

Tabelle 4: Performance-Vergleich für die Polygonvereinigung

Tabelle 4 zeigt einen Vergleich von Zeiten, die für die Vereinigung von 1940 Polygonen für verschiedene Methoden gemessen wurden. Es zeigt sich, dass die Vereinigung mit Hilfe der Nettopology Suite und der Anwendung der „Cascaded Union“ Methode am schnellsten arbeitet.

### *Diskrete Polygonvereinigung*

In diesem Kapitel wird ein gänzlich anderer Ansatz für die Generierung der Polygone evaluiert.
Im ersten Schritt werden alle Polygone nacheinander in eine auf 0 initialisierte Matrix geladen. D.h. es werden die Einträge der Matrix auf 1 gesetzt, welche den Rand des Polygons berühren. Dieser Vorgang, welcher sehr schnell durchgeführt werden kann, wird in Abbildung 3 anschaulich dargestellt.

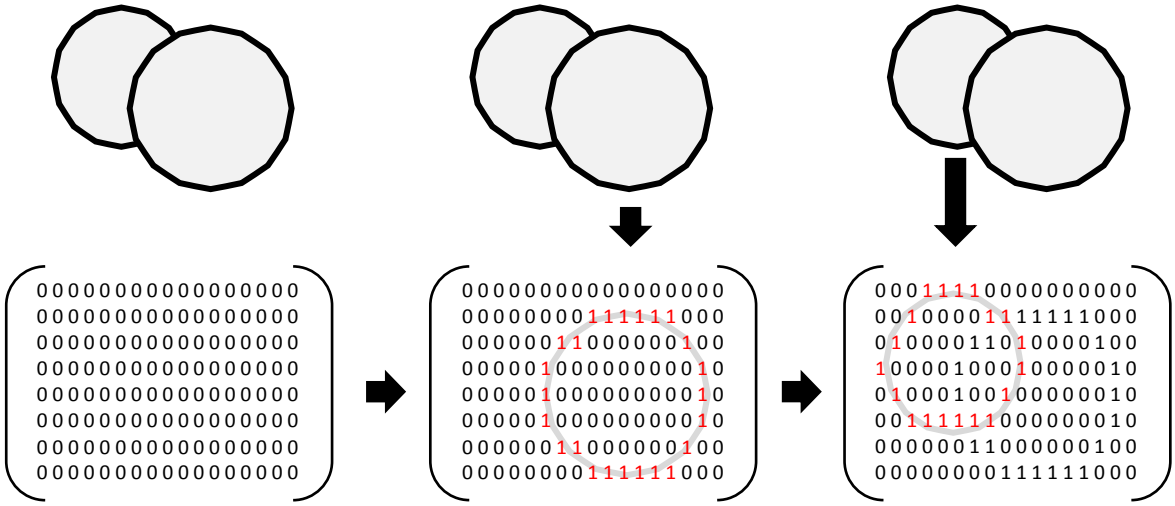



Abbildung 3: Eintragen von Polygonen in Matrix

Nach dem alle Polygone in der Matrix abgebildet sind, wird diese iterativ durchlaufen. Trifft man dabei auf ein gesetztes Bit (Wert ist 1) hat man den Rand eines Polygons erreicht. Eine Such- und Markiermethode versucht nun entlang dem Rand des entdeckten Polygons in der Matrix zu gehen bis wieder der Ausgangspunkt in der Matrix erreicht ist. Gleichzeitig markiert sie dabei das gefundene Polygon, indem sie die Einträge der besuchten Matrix-Elemente von 1 auf 2 setzt (Abbildung 4).

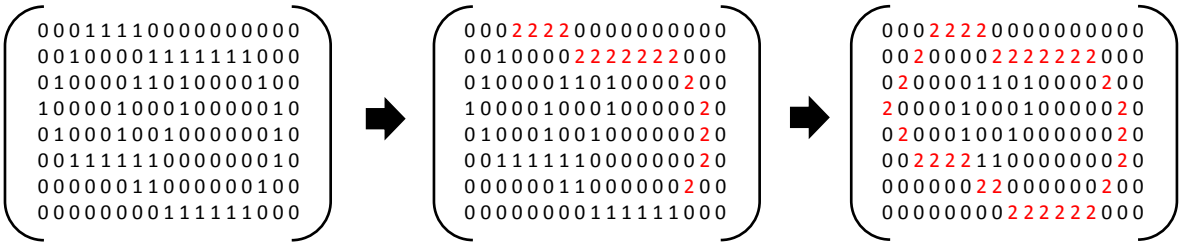



Abbildung 4: Markiervorgang einer Polygonhülle

Nach dem Markiervorgang wird die Matrix weiter durchlaufen bis das nächste Polygon gefunden wird, welches noch nicht markiert ist. Der Vorgang des Suchens und Markierens wird solange weitergeführt, bis die ganze Matrix durchlaufen ist. Es ist zu beachten, dass beim Matrixdurchlauf jeweils festgehalten werden muss, ob man sich an der aktuellen Position bereits in einem markierten Polygon aufhält. Ist dies der Fall, so würde man beim Auffinden einer 1 in der Matrix keinen Markiervorgang starten.
Der Markiervorgang sollte gleichzeitig dazu genutzt werden, ein Polygon-Objekt zu erzeugen, welches einem Set von Polygon-Objekten als das Endresultat der Vereinigung angehängt werden kann.

Die Methode der diskreten Polygonvereinigung hat zwei Nachteile. Ein Verlust der Genauigkeit ist unumgänglich. Die Grösse dieses Verlusts ist abhängig von der Grösse der gewählten Matrix. D.h. der Verlust kann durch eine Vergrösserung der eingesetzten Matrix minimiert werden. Der zweite Nachteil besteht darin, dass Löcher in den Polygonen nicht erkannt werden. Durch eine Erweiterung des Polygon-Erkennungs-Algorithmus könnte dieses Problem behoben werden.

### Programmcode-Optimierung

Eine Erhöhung der Performance kann eventuell durch Optimierungen am Programmcode erreicht werden. Einige Beispiele für vielversprechende Verbesserungen bei der bisherigen C#-Implementation des CIPT-Dienstes aus (Microsoft Corporation, 2011) und (Varszegi, 2011):

- Unnötige Kopiervorgänge durch Referenzaufrufe ersetzen
- Multithreading verwenden wo sinnvoll
- Wenige grosse Assemblies (Programmbibliotheken) anstatt viele kleine verwenden
- „For“ anstelle „Foreach“ verwenden
- „Stringbuilder“ anstelle Stringkonkatenation verwenden
- „System.Object“ selten benutzen

#### Caching

Durch Caching kann nicht die Performance des CIPT-Algorithmus an sich gesteigert werden. Jedoch kann dadurch erreicht werden, dass Anfragen mit den gleichen Parametern nicht mehrmals berechnet werden müssen, sondern nach der erstmaligen Berechnung direkt aus dem Cache beantwortet werden. Dies könnte im praktischen Einsatz bei verschiedenen Web-Seiten eingesetzt werden. Eine beispielhafte Anwendung wäre die Einbindung des CIPT-Dienstes in eine Immobilien-Webseite. Wird eine bestimmte Immobilie ausgewählt, präsentiert die Webseite dem Besucher eine Erreichbarkeits-Karte, welche mit Hilfe des CIPT-Dienstes erstellt wird. Wiederholte Aufrufe der Info-Seite für diese bestimmte Immobilie senden jeweils immer wieder einen Request an den CIPT-Dienst mit den gleichen Parameter-Werten. Dank des Cachings wird nun lediglich der erste Aufruf an den CIPT-Algorithmus weitergeleitet. Aufrufe mit den gleichen Parametern können danach direkt aus dem Cache beantwortet werden. Die aktuelle Implementation des Caches speichert die Daten im Arbeitsspeicher des Webserver.

### *3.1.2 Zusätzliche Funktionen*

Zusätzliche Daten im HAFAS-Fahrplan, welche in der Semesterarbeit noch nicht benutzt wurden, erlauben den CIPT-Algorithmus mit neuen Funktionen zu erweitern.

#### Berücksichtigung von Nachbar-Haltestellen (METABHF)

Bis anhin berücksichtigte der CIPT-Algorithmus geografische Distanzen zwischen zwei Haltestellen in keiner Weise. Dies führt dazu, dass Haltestellen, die sich fast am gleichen Ort befinden, jedoch von den Fahrplandaten unterschieden werden (zum Beispiel der Zug-Bahnhof und der Schiffsteg in Rapperswil), nicht als Nachbarn erkannt werden und diese Knoten deshalb keine Verbindung im Verkehrsnetzwerk erhalten.
Die Erweiterung des CIPT-Algorithmus für Fusswege zwischen benachbarten Haltestellen kann mit Hilfe der Tabelle METABHF in den HAFAS-Daten der SBB und dem ZVV bewerkstelligt werden.

#### Berücksichtigung der Umsteigezeit (UMSTEIGB, UMSTEIGZ)

Ein weiteres Manko in der bisherigen Implementation war, dass eine allfällig benötigte Umsteigezeit an einer Haltestelle ignoriert wurde. Dies führte dazu, dass zu viele Verbindungen errechnet wurden. Kommt beispielsweise um 8.00 Uhr auf Gleis 52 in Zürich ein Zug an, so war es laut Algorithmus möglich, den Anschlusszug um 8.01 Uhr auf Gleis 13 zu erreichen. Dies ist in der Realität nicht möglich, da Gleis 52 und Gleis 13 zu weit auseinanderliegen, um die Distanz innerhalb einer Minute zu bewältigen.

Mit Hilfe der Information in den beiden Dateien „UMSTEIGB“ und „UMSTEIGZ“ ist es möglich, diesen Fehler zu beheben und das Ergebnis des Algorithmus näher an die realen Bedingungen zu bringen.

## 3.2 Services und Schnittstellen

Der CIPT-Dienst soll seine Fähigkeiten einem möglichst grossen Kreis von potentiellen Clients zur Verfügung stellen. Daher ist es wichtig, dass der Dienst wohldefinierte Schnittstellen anbietet, die von diesen Clients verstanden werden. Es ist selbstverständlich, dass dabei auf verbreitete Standards zurückgegriffen wird, damit der Aufwand für die Implementierung eines Clients, der den CIPT-Dienst nutzt, möglichst gering ist.

- Das Open Geospatial Consortium (OGC) hat diverse Standards im Bereich von raumbezogenen Daten geschaffen. Unter anderem den Web Processing Service (WPS) und Web Feature Service (WFS). Der WPS wurde bereits zu Beginn der Semesterarbeit implementiert. Um den CIPT-Dienst über WPS testen zu können, wurde ein Client (Quantum GIS) verwendet. Es zeigte sich jedoch, dass dieser Client WPS nicht sauber implementiert hatte. Aus diesem Grund, wurde zusätzlich noch ein WFS implementiert. Mittlerweile wurde jedoch die WPS-Implementation im Client verbessert. In dieser Bachelorarbeit wird deshalb der WFS nicht mehr weiter unterstützt.
- Zusätzlich zum WPS soll der CIPT-Dienst auch über den simplen Aufruf einer URL mit vordefinierten Parametern angesprochen werden können. Man spricht bei dieser Funktionsweise vielfach von REST (Representational State Transfer, Fielding, 2000).

## 3.3 Datenformate

Die Geodaten, die bei einem Aufruf des CIPT-Dienstes zurückgegeben, werden sind in einem vordefinierten Format. Im Rahmen der Semesterarbeit wurden die beiden Formate GML und KML verwendet. Neu hinzu kommt GeoJSON. Dies ist eine Erweiterung des bekannten JSON Formats, welches sehr gut mit Java-Script zusammenarbeitet und deshalb häufig für Ajax-Requests verwendet wird.

## 3.4 Geocoding

Der Begriff Geocoding bezieht sich u.a. auch auf den Prozess der Umwandlung einer geografischen Adresse zu einem Koordinatenpaar welches die Lokalisierung der Adresse auf der Erde ermöglicht. Damit dies erreicht werden kann, muss ein grosser Datenstamm gepflegt werden.

- Geonames.org ist ein Projekt, welches vom Schweizer Marc Wick erstellt wurde. Das Projekt steht unter der Creative Commons Attribution 3.0 Lizenz und die Daten können gratis heruntergeladen werden. Nach Aussagen der Projektwebseite werden über 8 Millionen Datensätze verwaltet. Für die Schweiz stehen aktuell (Dezember 2011) nur 22‘215 Datensätze zur Verfügung.
- Mit dem Google Maps API ist es auch möglich die Koordinaten von Adressen zu laden. Die Daten können nicht heruntergeladen werden, jedoch können Anfragen an den Google Server gestellt werden. Aktuell dürfen täglich bis 2‘500 Anfragen gemacht werden. Die Antworten des APIs werden im JSON-Format geliefert.

- Yahoo! PlaceFinder ist ein direktes Konkurrenzprodukt zum Google Service und funktioniert ähnlich. Bei diesem Service dürfen jedoch täglich bis 50‘000 Anfragen pro Webapplikation gemacht werden und die Antworten sind in einer XML-Struktur.

Ein kurzer Test mit einigen, dem Autor bekannten, Strassennamen hat ergeben, dass Geonames.org nicht zu gebrauchen ist. Der Datensatz für die Schweiz ist anscheinend viel zu klein. Gute Resultate wurden jedoch vom Google- und vom Yahoo-Service geliefert. Für die Implementation wurde schliesslich das Google Maps API gewählt, da dieses mit JSON antwortet und deshalb mit Java-Script einfacher behandelt werden kann.

## 3.5 Webapplikation

Die zu erstellende Webapplikation soll die Funktionalität des CIPT-Dienstes aufzeigen können und gleichzeitig ein Werkzeug für die Webentwickler sein, welche den CIPT-Dienst auch auf ihrer Webseite einführen möchten. Es soll daher ein Link-, respektive Code-Generator realisiert werden.

### *3.5.1 Vom Google Maps API zu OpenLayers*

Die Webapplikation soll die erstellten geografischen Daten sinnvoll präsentieren. Daher ist es unabdingbar, dass die Applikation eine Landkarte enthält. In der Semesterarbeit wurde dabei auf das Google Maps API zurückgegriffen. Dies ist ein weit verbreitetes API, welches für kleinere Projekte kostenlos verwendet werden darf. Dieses steht jedoch nicht unter einer freien Lizenz und kann jederzeit von Google deaktiviert werden. Mit OpenLayers steht eine opensource Alternative zur Verfügung, welche einen ähnlichen Funktionsumfang bietet (OpenLayers). Falls gewünscht, können mit OpenLayers auch die Google Kartendaten eingebunden werden. Beim Einsatz von OpenLayers werden jedoch meistens die Kartendaten von OpenStreetmap (OSM Foundation, 2011) genutzt.
Da durch den Einsatz von OpenLayers keine Abhängigkeit zu einem kommerziellen Anbieter entsteht, wird in der Bachelorarbeit neu OpenLayers eingesetzt.

### *3.5.2 Java-Script Bibliotheken*

Für die Erstellung einer interaktiven Webanwendung wird heutzutage meistens auf den Einsatz von Java-Script (JS) zurückgegriffen. Vorstellbar wäre es auch, eine interaktive Webseite mit Adobe Flash oder Microsoft Silverlight zu realisieren. JS hat aber den klaren Vorteil, dass es ohne die Installation von zusätzlichen Plug-Ins für den Webbrowser auskommt und auf praktisch allen Plattformen läuft. Für umfangreiche Programmfunktionen ist es empfehlenswert, auf JS-Bibliotheken zurückzugreifen. Es gibt eine grosse Anzahl solcher Bibliotheken, doch die am weitesten verbreitete Bibliothek ist jQuery (jQuery Project). Eine kurze Evaluation hat gezeigt, dass alle notwendigen Funktionen von jQuery unterstützt werden. In dieser Bachelorarbeit wird deshalb jQuery eingesetzt.

# 4. Realisierung

Im Kapitel Evaluation wurden verschiedene Lösungsansätze und Implementationsvorschläge diskutiert. Das aktuelle Kapitel geht nun konkret auf die ausgewählten Varianten ein und beschreibt detailliert, was, wie und weshalb gemacht wurde.

## 4.1 Vorberechnung

Die Evaluation hat gezeigt, dass eine Vorberechnung möglich ist und ein Geschwindigkeits-Vorteil erzielt werden kann. Die zu erwartende Datenmenge liegt im Bereich von 2 bis 5 Terabyte. Ausserdem muss für jede der ungefähr 2‘900‘000 Fahrten der CIPT-Algorithmus ausgeführt werden. Der Algorithmus soll für eine unbeschränkte Reisezeit ausgeführt werden. Die Erfahrungen aus der Semesterarbeit haben gezeigt, dass die Ausführungszeit des Algorithmus etwa 5 Sekunden beträgt. Total müssten also 14‘500‘000 Sekunden gerechnet werden – dies entspricht etwa 168 Tagen.
Da im Rahmen der Bachelorarbeit diese Zeit nicht zur Verfügung steht, musste eine Alternative gefunden werden. Ein System, welches ein verteiltes, paralleles und somit schnelleres Berechnen der Daten ermöglichte, war die Lösung. Eine PostgreSQL-Datenbank diente als zentrale Verwaltungsstelle, welche die zu berechnenden Datensätze auf Clients verteilt und danach die berechneten Resultate wieder zurückerhält und zentral abspeichert. Die Client-Software wurde in C# mit WPF implementiert und kann daher auf allen aktuellen Windows-Betriebssystemen installiert werden. Der Client ist sehr einfach zu bedienen. Nach dem Start der Applikation muss ein Name zur Identifikation eingegeben werden und der Berechnungsvorgang durch einen Klick gestartet werden. Alle weiteren Vorgänge laufen automatisch ab:

- Falls das System, auf dem der Client gestartet wird, nicht genügend freie Hardware-Ressourcen hat, wird die Berechnung wieder abgebrochen.
- Zu Beginn werden einmalig alle Fahrplandaten, die zur Berechnung nötig sind, heruntergeladen.
- Von der Datenbank wird eine neue Aufgabe zur Berechnung angefordert (Haltestelle und Abfahrtszeit).
- Die Berechnung wird durchgeführt und das Resultat an die Datenbank übertragen.

Die beiden letzten Schritte wiederholen sich, bis die Datenbank keine Daten zur Berechnung mehr hat oder der Client durch den Benutzer abgebrochen wird. Falls das System, auf dem der Client läuft, mehrere Prozessorkerne hat, werden diese beiden Schritte auch mehrfach, parallel in mehreren Threads ausgeführt. Ein zusätzlicher Watch-Dog-Thread überwacht die Ausführung und greift bei Problemen bei der Berechnung ein.

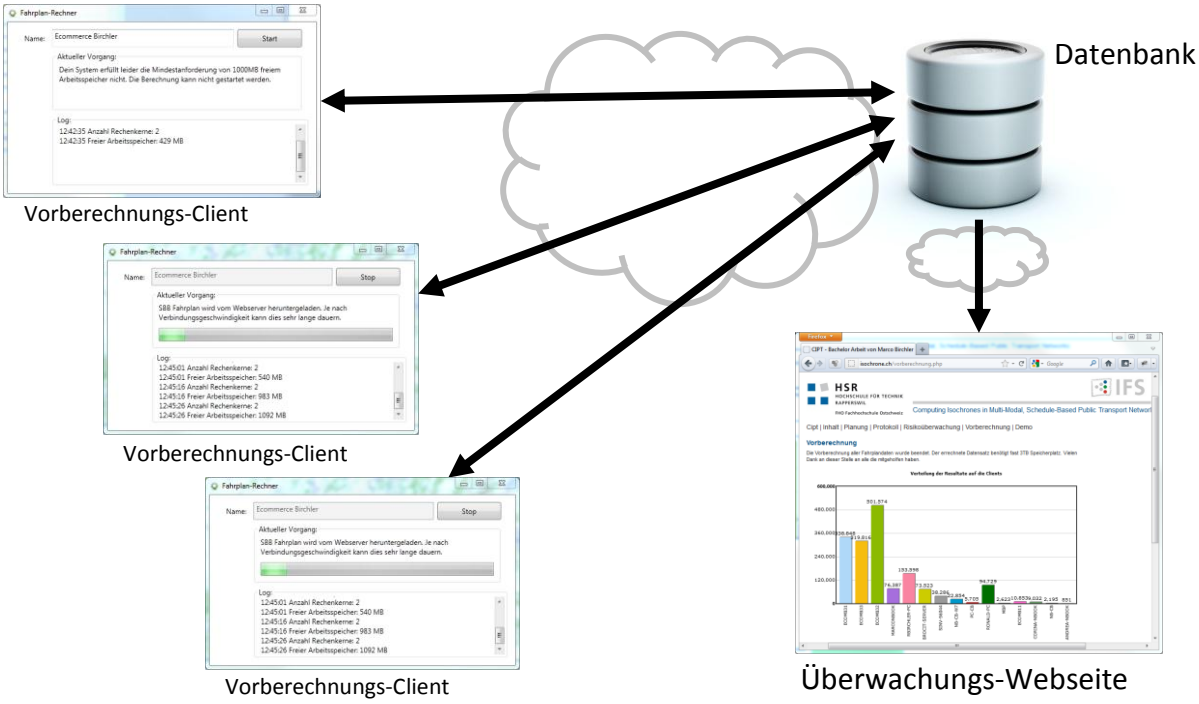



**Abbildung 5: Verteilte Berechnung der vorberechneten CIPT-Daten**

Die Abbildung 5 zeigt den Ablauf der Vorberechnung. Mit der Überwachungs-Webseite kann der Status der Vorberechnung angezeigt werden.
Bei der Ausführung der Vorberechnung zeigte sich, dass das verteilte System relativ schnell an eine Leistungsgrenze stösst. Der gleichzeitige Einsatz von fünf Clients erlaubte eine Rechenrate von etwa zwei Datensätzen pro Sekunde. Weitere Clients konnten die Performance nicht erhöhen, da der Datenbankserver die errechneten Datensätze nicht mehr verarbeiten konnte.
Die gesamte Vorberechnung konnte dank der parallelen Berechnung in etwas mehr als zwei Wochen durchgeführt werden.

## 4.2 Isochronen-Generierung

In der Semesterarbeit wurde eine Bibliothek von Microsoft für die geometrischen Berechnungen genutzt. Gemäss dem Vergleich im Abschnitt „Evaluation“ (Tabelle 4, Seite 21) können mit der „Cascaded Union“ Strategie geometrische Polygonvereinigungen schneller errechnet werden. Deshalb wurde die Microsoft Bibliothek nun durch die Nettopology Suite mit der „Cascaded Union“ Strategie ersetzt. Weitere umgesetzte Verbesserungen und Erweiterungen an der Isochronen-Generierung werden in den folgenden Unterkapiteln vorgestellt.

### 4.2.1 Variable Präzision der Berechnung

Die neue Bibliothek Nettopology erlaubt, die Präzision der zu berechnenden geometrischen Daten anzugeben. Dies wird durch die Angabe eines zusätzlichen Parameters beim Erstellen der einzelnen Polygone um die Ziel-Haltestellen erreicht. Dieser Parameter bestimmt die Anzahl Punkte, die für das Zeichnen eines kreisähnlichen Polygons in einem Quadranten benutzt werden. Abbildung 6 zeigt Polygone, welche um eine Ziel-Haltestelle erstellt werden. Dabei wurden 0, 1 und 3 als Parameterwerte, d.h. also für die Anzahl Punkte, die in einem Quadranten benutzt werden, verwendet.

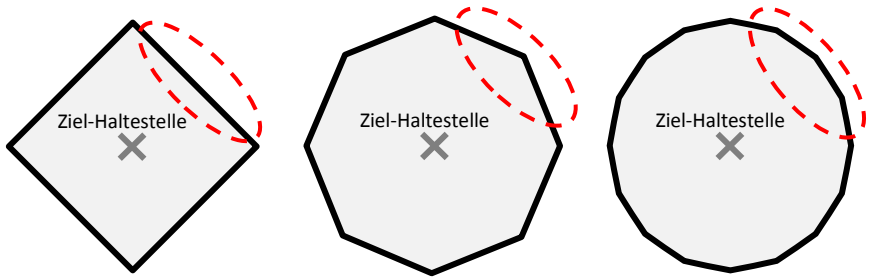



Abbildung 6: Verschieden präzise Restlaufzeit-Polygone um eine Haltestelle

Dieser Parameter kann dem CIPT-Dienst nun beim Aufruf mitgegeben werden. Fehlt der Parameter, wird bei der Generierung jedes einzelnen Polygons automatisch ein sinnvoller Wert bestimmt. Folgende Punkte beeinflussen den automatisch generierten Wert:

- Je höher die Gesamtzahl der erreichbaren Haltestellen, desto kleiner die Anzahl Punkte für das Erstellen eines einzelnen Polygons. Damit wird berücksichtigt, dass die Isochronen-Darstellung auf der Karte weiter herausgezoomt wird, wenn mehr Haltestellen angezeigt werden. Bei einer kleineren Zoomstufe genügt jedoch auch eine weniger präzise Darstellung.
- Die restliche Laufzeit der Haltestelle bestimmt den „Radius“ des Polygons. Je grösser die restliche Laufzeit, desto grösser wird das Polygon und desto mehr Punkte werden benötigt, um es zu zeichnen.

Durch die automatische Anpassung der Präzision konnte erreicht werden, dass der CIPT-Algorithmus für die gleichen Anfrageparameter schneller ein Ergebnis liefert. Es ist allerdings zu beachten, dass das gelieferte Ergebnis weniger genau ist, weshalb dies nicht als performanter bezeichnet werden sollte.

### 4.2.2 Diskrete Polygonvereinigung

Das in Kapitel „Diskrete Polygonvereinigung“ auf Seite 21 beschriebene Verfahren wurde ebenfalls implementiert. Dabei wurde festgestellt, dass diese Methode wohl das Potential hätte, schneller als die geometrische Vereinigung zu arbeiten. Leider reichte die Zeit der Bachelorarbeit nicht aus, um die Implementation auf ein qualitativ gutes Niveau zu heben. Deshalb benutzt der CIPT-Dienst nach wie vor die oben erwähnte geometrische Polygonvereinigung.

### 4.2.3 Vermeidung der Isochronen-Generierung durch Heatmaps

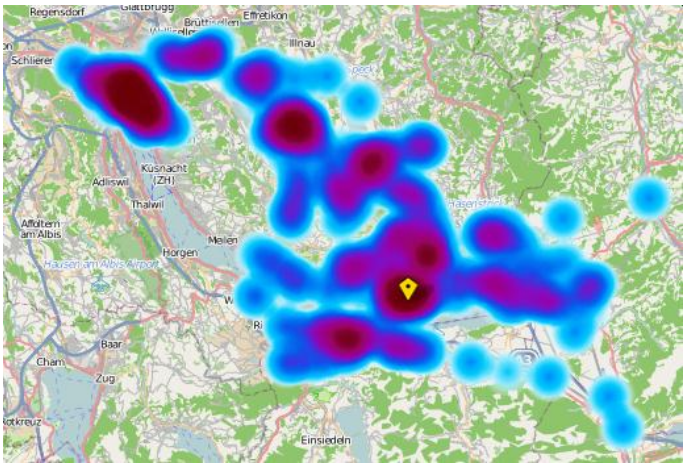



**Abbildung 7: Heatmap für eine Reisezeit von 55 Minuten ab Rapperswil**

Bei der Entwicklung des Web-Clients wurde eine zusätzliche Möglichkeit für die Anzeige der Ergebnisse implementiert. Eine Heatmap zeigt für jede Ziel-Haltestelle, welche innerhalb der gesetzten Reisezeit erreicht werden kann, einen Farbklecks auf der Landkarte. Für die Anzeige der Heatmap müssen keine Isochronen und damit auch keine komplizierten Polygonvereinigungen berechnet werden. Es muss lediglich eine Liste aller Zielstationen, die erreicht werden können, an den Client gesendet werden. Dadurch wird eine Heatmap im Web-Client viel schneller angezeigt als eine Isochronen-Karte. Der grosse Nachteil dieser Darstellung ist jedoch, dass keine exakten Auswertungen gemacht werden können, da keine Grenzlinien vorhanden sind. Die aktuell verwendete Javascript-Bibliothek, welche die Umsetzung der Punkte-Liste in die grafische Heatmap erledigt, hat ausserdem den Nachteil, dass eine höhere Dichte der Haltestellen an einem Ort zu intensiveren Farben führt. Daher ist die Intensität der Farbe nicht abhängig von der Restlaufzeit, wie dies bei der Einfärbung der Isochronen der Fall ist. Abbildung 7 zeigt beispielsweise die Heatmap für eine Reise von 55 Minuten ab Rapperswil. Der grosse, intensive Fleck über Zürich bedeutet in diesem Fall nicht etwa, dass Zürich sehr schnell von Rapperswil aus erreichbar ist, sondern dass es in dieser Region viele Haltestellen gibt, die innerhalb von 55 Minuten ab Rapperswil erreicht werden können.

## 4.3 Services und Schnittstellen

In der Semesterarbeit war ursprünglich der WPS als wichtigste Kommunikations-Schnittstelle für den CIPT-Dienst gedacht. Da QGis, respektive dessen Plug-In, welches noch in der Entwicklung steckte, nicht zur Zusammenarbeit mit dem erstellten WPS bewegt werden konnte, wurde in der Semesterarbeit zusätzlich eine WFS Schnittstelle implementiert.
Da es jedoch sehr wünschenswert wäre, dass der WPS auch mit QGis zusammenarbeitet, wurde zu Beginn der Bachelorarbeit das WPS Plug-In für QGis genauer untersucht. Dabei wurde entdeckt, dass das Plug-In nur eine von zwei in WPS definierten Betriebsarten unterstützt. Daraufhin wurde die Betriebsart der WPS Schnittstelle des CIPT-Dienstes angepasst. Der CIPT-Dienst kann nun mit QGis via WPS angesprochen werden. Dieser Erfolg liess nun die WFS Schnittstelle nutzlos werden. Da WPS viel flexibler eingesetzt werden kann als WFS, wurde beschlossen, die WFS Schnittstelle mit CIPT nicht mehr weiter zu unterstützen.
Daten können jedoch vom CIPT-Dienst nicht nur via WPS, sondern auch über einen simplen URL Aufruf, d.h. einen GET-Request an die Serveradresse bezogen werden.

| Parametername | Wertebereich |
|---|---|
| lat, lon | Diese beiden Parameter zusammen beschreiben die Startposition als eine WGS 84 Koordinate. |
| address | Dieser Parameter enthält den Startpunkt als Adresse. Mit der Hilfe eines Geocoding-Dienstes versucht der CIPT-Dienst, diese in eine Startkoordinate zu überführen. |
| travelminutes | Die maximale Reisedauer, welche angezeigt werden soll, wird damit festgelegt. Gültige Werte liegen zwischen 1 und dem in der Datei „Web.config“ hinterlegten Wert bei „maxTravelMinutes“. |
| timemodel | Dieser Parameter erwartet für die Auswahl eines Zeitmodells einen der folgenden String-Werte:<br>• *fixstarttime*: Der CIPT-Dienst liefert für eine fixe Startzeit alle Gebiete, welche vom Startpunkt aus innerhalb der maximalen Reisedauer erreicht werden können. Dieser Parameterwert benötigt für eine korrekte Funktionsweise zusätzlich den starttime-Parameter.<br>• *minimaltime*: Es werden alle Gebiete geliefert, welche vom Startpunkt aus innerhalb der maximalen Reisedauer erreicht werden können. Als Reisedauer zwischen der Haltestelle beim Startpunkt und den Haltestellen bei den Endpunkten wird die schnellstmögliche Verbindung herausgesucht.<br>• *averagetime*: Es werden alle Gebiete geliefert, welche vom Startpunkt aus innerhalb der maximalen Reisedauer erreicht werden können. Als Reisedauer zwischen der Haltestelle beim Startpunkt und den Haltestellen bei den Endpunkten wird die durchschnittliche Reisedauer zwischen diesen Haltestellen benutzt. |
| **starttime** | Dieser Integer-Parameter wird nur benötigt wenn als Zeitmodell „fixstarttime“ gewählt wurde und gibt an, zu welchem Zeitpunkt eine Reise beim Startpunkt gestartet werden soll. Der Parameterwert wird durch die Konkatenation von Stunden und Minuten der Zeitdarstellung gebildet. Wobei der Minuten-Teil immer zweistellig sein muss. 8:05 Uhr würde beispielsweise zu 805 transformiert. |
| **objecttype** | Dieser Parameter erwartet für die Auswahl eines Objekte-Typs, welcher zurückgeliefert wird, einen der folgenden String-Werte:<br>• *isochrones*: Der CIPT-Dienst generiert Polygone, welche jeweils durch deren äusseren Rand Isochronen darstellen.<br>• *stations*: Der CIPT-Dienst liefert lediglich Punktobjekte zurück, welche die Haltestellen symbolisieren, die erreicht werden können.<br>• *heatmap*: Der CIPT-Dienst liefert ebenfalls Punktobjekte zurück. Zurzeit behandelt der CIPT-Dienst die Parameterwerte „stations“ und „heatmap“ intern identisch. |
| **isocount** | Dieser Integer-Parameter wird nur benötigt, wenn als Objekttyp „isochrones“ gewählt wurde und gibt an, wie viele Isochronen |

| | |
|---|---|
| | gebildet werden sollen. Gültige Werte sind zwischen 1 und dem in der Datei „Web.config“ hinterlegten Wert bei „maxIsoNumber“. |
| **prezision** | Dieser Integer-Parameter wird nur benötigt, wenn als Objekttyp „isochrones“ gewählt wurde und gibt an, wie genau die Isochronen gebildet werden sollen. Gültige Werte sind zwischen 1 und dem in der Datei „Web.config“ hinterlegten Wert bei „maxPrezision“. Weitere Informationen dazu in Kapitel 4.2.1. |
| **format** | Dieser Parameter erwartet für die Auswahl eines Format-Typs, welcher zurückgeliefert wird, einen der folgenden String-Werte:<br>• *gml*: Das Resultat wird im Gml Format zurückgesendet.<br>• *kml*: Das Resultat wird im Kml Format zurückgesendet.<br>• *geojson*: Das Resultat wird im GeoJSON Format zurückgesendet.<br>• *map*: Das Resultat wird als HTML-Seite, welche die Daten auf einer Karte anzeigt, zurückgesendet. Die Seite enthält eingebettet die errechneten Daten in GeoJSON. |

**Tabelle 5: Übersicht der unterstützten Parameter des CIPT-Dienstes**

### *4.3.1 Definition des Startpunkts*

Die Angabe eines Koordinatenpaars oder des „Address“-Parameters ist obligatorisch. Werden alle drei Parameter gesetzt, versucht der CIPT-Dienst, die entsprechende Koordinate für den „Address“-Parameter herauszufinden. Führt dies zu keinem Ergebnis, werden die Lat/Lon-Koordinaten verwendet.
Die Adressangabe kann unter Umständen nicht eindeutig einer Koordinate zugeordnet werden, beispielsweise dann, wenn nur eine Strasse ohne Stadt angegeben wird und eine gleichnamige Strasse in mehreren Städten existiert. Der CIPT-Dienst verwendet in diesem Fall einfach das erste Ergebnis, welches vom Geocoding-Dienst geliefert wird. Damit dem CIPT-Dienst mit Sicherheit eine eindeutige Adresse geliefert werden kann, ist es möglich, vom CIPT-Dienst eine Liste aller möglichen Adress-Kandidaten für eine Teil-Adresse zu erfragen. Die untenstehende URL liefert beispielsweise jene Ortschaften in der Schweiz, welche eine Eisenbahnstrasse haben, im JSON-Format.

*http://map.isochrone.ch/term/eisenbahnstrasse*

### 4.3.2 URL-Rewriting

Es hat sich herausgestellt, dass das WPS Plug-In von QGis Probleme mit dem Et-Zeichen („&“, engl. Ampersand) in den URL-Requests hat. Auf dem Webserver wurden deshalb URL-Rewriting-Regeln konfiguriert. Tabelle 6 zeigt drei verschiedene Möglichkeiten um genau die gleichen Informationen vom CIPT-Dienst zu erfragen.

| URL |
| --- |
| http://map.isochrone.ch/client/Data.aspx?lat=47.221&lon=8.816&timemodel=minimaltime&format=gml&objecttype=isochrones&starttime=0800&travelminutes=30&isocount=5&prezision=0 |
| http://map.isochrone.ch/lat/47.221/lon/8.816/timemodel/minimaltime/format/gml/objecttype/isochrones/starttime/0800/travelminutes/30/isocount/5/prezision/0/ |
| http://map.isochrone.ch/47.221/8.816/minimaltime/gml/isochrones/0800/30/5/0/ |

**Tabelle 6: Vergleich von URL-Alternativen für die gleiche Abfrage**

## 4.4 Datenformate

Der CIPT-Dienst unterstützt nun KML, GML 1.0 und GeoJSON als Datenformate. Über die WPS-Schnittstelle wird automatisch GML geliefert. Bei einer URL-Abfrage kann das Format mit dem „format“-Parameter bestimmt werden.

## 4.5 Caching

Mit dem verwendeten ASP.NET Framework kann ein Caching auf zwei Arten realisiert werden. Entweder übernimmt der Webserver das Caching. Erkennt der Server einen GET-Request, den er bereits einmal beantwortet hat, lädt er die Antwort direkt aus dem Cache. Dies funktioniert jedoch nur für völlig identische GET-Requests (d.h. auch die Parameter-Werte müssen übereinstimmen). Ein intelligenteres Caching, wie dies der CIPT-Dienst benutzt, kann auf dem Applikations-Level implementiert werden. Dabei können auch Anfragen aus dem Cache beantwortet werden, die nicht die gleichen Parameter-Werte aufweisen. Beispielsweise können für Anfragen, welche sich lediglich im „format“-Parameter zwischen „geojson“ und „map“ unterscheiden, die gleichen Berechnungen durchgeführt werden. Lediglich die endgültige Antwort muss je nach Format noch angepasst werden.

# 5. Ergebnisse

In den folgenden Unterkapiteln werden die realisierten Ergebnisse mit den gesteckten Zielen aus Kapitel 1.2 verglichen. Es wird untersucht, ob die Performance des CIPT-Dienstes gesteigert werden konnte und ob die zusätzlichen Funktionen entsprechend den Vorgaben implementiert wurden. Zuletzt werden noch Probleme erläutert, die bei der Weiterentwicklung entstanden sind.

## 5.1 Performance

Bereits in der Semesterarbeit wurde eine Anwendung erstellt, welche das Testen der Performance des CIPT-Dienstes über die Web-Schnittstelle erlaubte. Dieses Tool wurde nun erweitert damit auch die neuen Parameter des CIPT-Dienstes unterstützt werden. An der Funktionsweise wurde nichts verändert. Das Tool wurde bei den Messungen, wie bei der Semesterarbeit, direkt auf dem Webserver ausgeführt. Die Resultate aus der Semesterarbeit und die neu gemessenen Ergebnisse sind absolut vergleichbar. Somit wird die Zeit für die Datenübertragung der Ergebnisse über das Internet und eine allfällige Zeit für die Aufbereitung der Daten im Client nicht mitgemessen. Effektiv gemessen wurde die Zeit in Sekunden zwischen dem Absenden eines GET-Requests an den Webserver und dem Erhalt der Antwort. Für jede Messung wurden jeweils 1‘000 einzelne CIPT-Abfragen durchgeführt und davon der Durchschnitt gebildet.

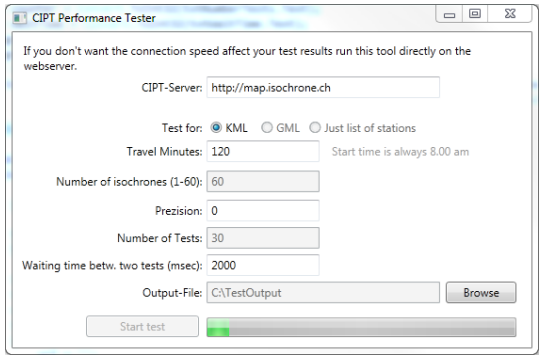



Abbildung 8: Performance-Test Tool

| | Reisezeit 30 Min. | Reisezeit 60 Min. | Reisezeit 120 Min. |
|---|---|---|---|
| 1 Isochrone | 0.01s | 0.10s | 2.30s |
| 10 Isochronen | 0.04s | 0.33s | 5.74s |
| 60 Isochronen | 0.28s | 1.13s | 23.29s |

Tabelle 7: Durchschnittliche Dauer einer CIPT-Abfrage nach der Semesterarbeit

Tabelle 7 zeigt die Ergebnisse der CIPT-Messung aus der Semesterarbeit (Birchler & Lynn, 2011, S. 19). Tabelle 8 zeigt die Ergebnisse der CIPT-Messungen nach der Implementation der Verbesserungen in der Bachelorarbeit. Der nun neu verfügbare Präzisions-Parameter wurde auf null gesetzt, dies bedeutet, dass der CIPT-Algorithmus die Präzision selber bestimmt.

| | Reisezeit 30 Min. | Reisezeit 60 Min. | Reisezeit 120 Min. |
|---|---|---|---|
| 1 Isochrone | 0.01s | 0.04s | 0.43s |
| 10 Isochronen | 0.03s | 0.14s | 1.19s |
| 60 Isochronen | 0.10s | 0.43s | 3.87s |

Tabelle 8: Durchschnittliche Dauer einer CIPT-Abfrage nach der Bachelorarbeit (Präzision automatisch best.)

Tabelle 9 vergleicht jeweils die beiden zusammengehörenden Werte und zeigt die Performance-Steigerung in Prozenten. Es ist gut ersichtlich, dass für die längste Reisezeit von 120 Minuten die höchste Performance-Steigerung erreicht werden konnte. Dies liegt daran, dass bei einer höheren Reisezeit die Präzision, mit der gerechnet wird, automatisch reduziert wird.

| | Reisezeit 30 Min. | Reisezeit 60 Min. | Reisezeit 120 Min. |
|---|---|---|---|
| **1 Isochrone** | 0% | 150% | 435% |
| **10 Isochronen** | 33% | 136% | 382% |
| **60 Isochronen** | 180% | 163% | 502% |

Tabelle 9: Performancesteigerung in Prozent

Die automatische Präzisionsanpassung kann ausgeschaltet werden, indem festgelegt wird, mit wie vielen Punkten ein Quadrant gezeichnet werden soll (Kapitel 4.2.1). Damit die Auswirkung der Präzisionsanpassung genauer untersucht werden kann, wurden zusätzliche Messungen mit 10 Isochronen und einer Reisezeit von 60 Minuten durchgeführt (Tabelle 10). Dabei zeigt sich, dass eine CIPT-Abfrage mit automatisch gewählter Präzision (0.14s) etwa gleich lange dauert wie eine Abfrage mit Präzision 40 (0.15s).

| | Präzision 1 | Präzision 5 | Präzision 20 | Präzision 40 |
|---|---|---|---|---|
| **10 Isochronen, Reisezeit 60 Min.** | 0.06s | 0.07s | 0.09s | 0.15s |

Tabelle 10: Durchschnittliche Dauer einer CIPT-Abfrage (versch. Präzisionen)

Eine Präzision von 40 bedeutet, dass für einen Quadranten eines „Polygon-Kreises“ 40 Punkte für die Zeichnung verwendet werden. Ein ganzer Kreis würde somit mit 160 Punkten gezeichnet werden und ein Punkt würde einen Winkel von 2.25 Grad (360 / 160) beschreiben. Je nach Anwendung würde es jedoch genügen, wenn grössere Winkel dargestellt würden. Eine Halbierung der Präzision auf 20 hätte Winkel mit 4.5 Grad zur Folge und würde gleichzeitig die Rechenzeit um 40% reduzieren.


Abbildung 9: Vergleich von 2.25 und 4.5 Grad Winkel

Abschliessend ist zu sagen, dass durch die Bachelorarbeit eine erhebliche Steigerung der Performance erreicht werden konnte und dass der CIPT-Dienst durch die Einführung des Präzisions-Parameters flexibel auf die Anforderungen angepasst werden kann.

## 5.2 Zusätzliche Funktionen

Die Genauigkeit des Dienstes konnte durch den Einbezug zusätzlicher Daten aus dem Fahrplan gesteigert werden. Die Vorberechnung der CIPT-Daten für alle Abfahrten ermöglichte es, durchschnittliche und minimale Fahrzeiten zwischen zwei Haltestellen zu berechnen. Somit können mit dem CIPT-Dienst nun auch allgemeine Aussagen gemacht werden, welche keine bestimmte Abfahrtszeit verlangen. So ist es beispielsweise möglich

alle Gebiete zu markieren, welche durchschnittlich in einer Stunde ab Rapperswil erreichbar sind.
Ein effizientes Caching konnte durch das ASP.NET Framework relativ einfach implementiert werden. Das GeoJSON Format wird nun ebenfalls unterstützt und erlaubt die einfache Einbindung des CIPT-Dienstes in Webapplikationen.
Ein Beispiel einer solchen Applikation wurde erstellt und erlaubt die Präsentation aller Funktionen, die der CIPT-Dienst bietet.

### 5.3 Probleme

Die meisten Probleme, welche bei der Weiterentwicklung des CIPT-Dienstes auftauchten, hatten ihren Ursprung in der grossen Menge der Daten, welche grundsätzlich mit normalen Desktopsystemen bewältigt werden mussten. Immer wieder mussten Zwangspausen eingelegt werden oder es musste kurzfristig umgeplant werden, weil beispielsweise die Datenbank während 48 Stunden einen Index neu aufbaute oder der Arbeitsspeicher auf dem Webserver nicht mehr ausreichte. Man könnte nun meinen, dass es einfacher gewesen wäre, die Entwicklung der Bachelorarbeit mit einem reduzierten Datensatz (z.B. nur der Fahrplan der Region Zürich) durchzuführen. Dies ist jedoch ein Trugschluss, denn genau diese grosse Datenmenge zeigte bereits bei der Entwicklung die Engpässe auf und erlaubte, Lösungen dafür zu finden.
Funktionen, welche während der Bachelorarbeit nicht mehr umgesetzt werden konnten, sind im folgenden Kapitel 6 beschrieben.

## 6. Weiterentwicklung

Die aktuelle Implementierung des CIPT-Dienstes bietet eine ausgezeichnete Basis für eine Weiterentwicklung. Der Dienst läuft stabil und die gelieferten Resultate sind korrekt. In den folgenden Abschnitten werden Probleme aufgeführt, welche bei der Entwicklung bis anhin nur unbefriedigend oder gar nicht gelöst werden konnten. Auch zusätzliche Funktionen, welche die Attraktivität des Dienstes steigern würden, könnten nachträglich implementiert werden.

### 6.1 Diskrete Polygonvereinigung

Die „Diskrete Polygonvereinigung“ (Seite 21) konnte während der Bachelorarbeit leider nur grob implementiert werden. Die Geschwindigkeit mit der eine Polygonvereinigung durchgeführt wurde, war langsamer als bei der geometrischen Vereinigung. Ausserdem wurden noch keine Löcher in den Polygonen erkannt. Es kann davon ausgegangen werden, dass diese beiden Defizite durch eine Hardware nähere Implementierung der Matrix und einem Ausbau des Algorithmus behoben werden können.

### 6.2 Geometrische Polygonvereinigung

Ein kritischer Blick auf Tabelle 4 (Performance-Vergleich für die Polygonvereinigung) zeigt, dass die „Iterative Union“ Methode mit der Microsoft-Bibliothek über zehnmal schneller

arbeitet als die „Iterative Union“ Methode mit der Nettopology Suite. Da die „Cascaded Union“ Methode, welche nur die Nettopology Suite anbietet, noch etwas schneller läuft, wurde die Nettopology Suite für die Implementation verwendet. Es wäre eventuell lohnenswert, zu untersuchen, wieso die Microsoft-Bibliothek so viel schneller arbeitet als die Nettopology Suite. Allenfalls könnte der Geschwindigkeitsvorteil auch auf die Nettopology Suite übertragen werden. Vielleicht wäre es auch möglich, die „Cascaded Union“ Methode selber, mit der Hilfe der schnellen Microsoft-Bibliothek, zu implementieren. Diese Bemühungen könnten zu einer weiteren Steigerung der Performance des CIPT-Algorithmus führen.

## 6.3 Einbezug der Topographie

Die Restlaufzeit bei einer Haltestelle wird momentan für den Radius des Kreises um diese Haltestelle verwendet. Dieser Restzeit-Kreis symbolisiert die Fläche, welche nach dem Verlassen der letzten Haltestelle zu Fuss noch erreicht werden kann. Diese Annäherung entspricht nur in seltenen Fällen der Wirklichkeit. Ein Fluss oder eine Autobahn können beispielsweise nicht beliebig überquert werden. Der Restzeit-Kreis würde von der Autobahn abgeschnitten werden. Auch Steigungen und Abhänge beeinflussen die Geh-Geschwindigkeit. Der Kreis würde durch Steigungen gestaucht und durch Abhänge gestreckt. Es gibt noch viele zusätzliche Informationen, welche für eine wirklichkeitsgetreuere Darstellung miteinbezogen werden müssten.

# Teil 2: Projekt-Dokumentation

Der vorliegende zweite Teil der Bachelorarbeit zeigt die erstellten Softwarekomponenten detailliert auf und beschreibt deren Implementierung. Ausserdem wird beschrieben, wie das gesamte Projekt geplant und durchgeführt wurde.

## 1. Analyse des Domänenmodells

Das bestehende Domänenmodell aus der Semesterarbeit wurde an mehreren Punkten erweitert. Das Domänen-Objekt „Precalculated Journey“ steht für eine vorberechnete Route zwischen zwei Haltestellen („Station“). „Footpath“ ist ein Fussweg zwischen zwei Haltestellen und enthält als Attribut die Geh-Zeit. Auf der rechten Seite von Abbildung 10 sind neue Domänenobjekte ersichtlich, welche im Zusammenhang mit der Modellierung des Umsteigevorgangs erstellt wurden.

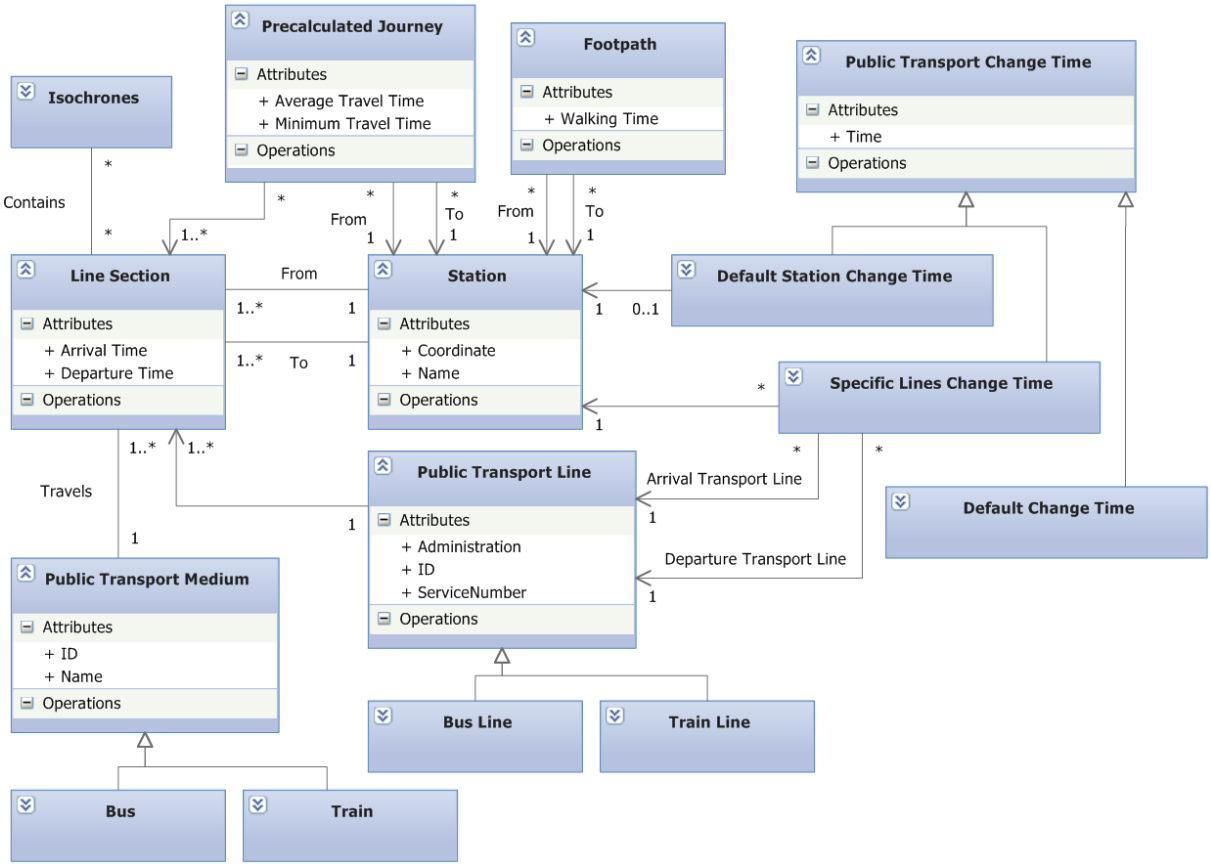



Abbildung 10: erweitertes Domänenmodell

# 2. Design und Implementation

Der Code für die Bachelorarbeit wurde in C# 4.0 geschrieben und das ASP.NET Framework führt den kompilierten Code auf dem Webserver aus. Als Datenbank wurde PostgreSQL v.9.0 eingesetzt. Es wurden somit die gleichen Technologien wie bei der Semesterarbeit eingesetzt.
Das nun folgende Kapitel beschreibt den Aufbau und die Funktionsweise des Codes im Detail.

## 2.1 System-Implementation

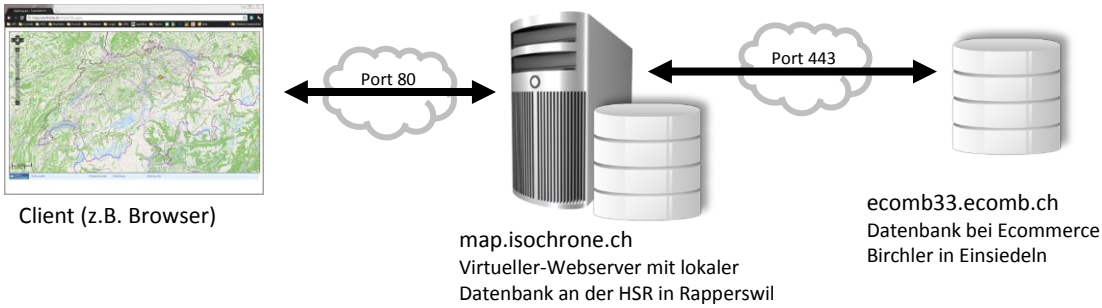



Abbildung 11: Effektive, physische Systemaufteilung nach der Bachelorarbeit

Abbildung 11 zeigt wie das CIPT-System nach dem Abschluss der Bachelorarbeit implementiert ist. Es ist neu eine zweite Datenbank (ecomb33.ecomb.ch) hinzugekommen, welche vom Webserver über das Internet erreicht werden kann. Diese Datenbank enthält die vorberechneten Daten. Der Grund für die Auslagerung ist, dass an der HSR im Rahmen der Bachelorarbeit nicht genügend online verfügbarer Speicherplatz organisiert werden konnte. Der Abbildung ist ausserdem zu entnehmen, dass beim Webserver (map.isochrone.ch) nach wie vor eine Datenbank vorhanden ist. Diese kleinere Datenbank enthält lediglich die Fahrplandaten, welche beim Starten des Webservers in den Arbeitsspeicher geladen werden. Würde die Datenbank beim Webserver genügend Speicherplatz bieten, wäre es aus Performance-Gründen empfehlenswert, die vorberechneten Daten dort abzulegen.

## 2.2 Datenbankschema

Das Schema der Webserver-Datenbank wurde um die Tabellen „umsteigb“, „umsteigz“ und „metabh“ erweitert. Die darin enthaltenen Daten wurden aus den gleich benannten Text-Dokumenten der HAFAS-Daten importiert. Der Import wurde mit dem Import-Tool, welches für die Semesterarbeit erstellt wurde, durchgeführt. Die Tabelle „metabhf“ enthält Haltestellen, welche durch einen Fussweg verbunden sind und die Zeit, die für diese Strecke gebraucht wird. „umsteigb“ enthält Umsteigezeiten für einzelne Haltestellen, für welche die Standardzeit der SBB von zwei Minuten nicht ausreicht. „umsteigz“ enthält ebenfalls eine Umsteigezeit, welche für spezifische Fälle gesetzt werden kann. Diese Umsteigezeit ist abhängig von der Haltestelle und zwei eindeutigen ÖV-Linien. Beispielsweise wird dort

eingetragen, dass wenn man in Rapperswil vom Voralpen Express auf die S5 umsteigen möchte, zehn Minuten für den Umsteigevorgang benötigt werden.
Abbildung 12 zeigt die Struktur der Datenbank beim Webserver mit den neuen Tabellen. Jedes Mal wenn der Webserver-Prozess gestartet wird, werden alle Daten in dieser Datenbank in den Arbeitsspeicher des Webservers geladen.

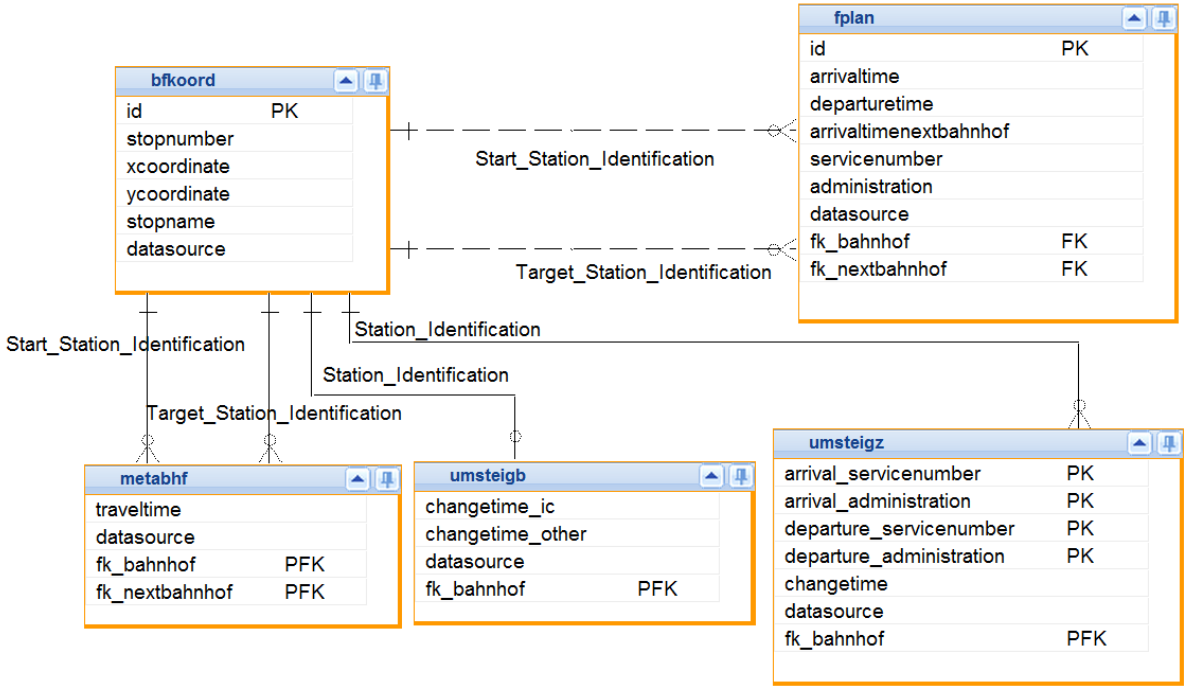



Abbildung 12: Schema der Webserver-Datenbank (map.isochrone.ch)

Die vorberechneten Daten wurden in eine zweite Datenbank geladen (Abbildung 13). Diese hat die Tabellen „traveltime“, „traveltimegrouped“, „fplan_calculated“ und „bfkoord“, wobei „bfkoord“ die gleichen Daten enthält, wie „bfkoord“ in der Webserver-Datenbank.
Die Tabelle „traveltime“ enthält die eigentlich vorberechneten Daten, die Reisezeiten, in der Spalte „travelminutes“. Mit Fremdschlüsselbeziehungen werden ausserdem die Ziel-Haltestelle und indirekt via „fplan_calculated“ die Start-Haltestelle referenziert.
„fplan_calculated“ diente vor allem der Verwaltung der Vorberechnung. Darin wurden Informationen über den Client, der die Vorberechnung für einen bestimmten Datensatz durchführte, abgespeichert.
Die Tabelle „traveltimegrouped“ enthält die aggregierten Daten von „traveltime“. Grundsätzlich wurde auf „traveltime“ eine „GROUP BY“-Operation ausgeführt und das Ergebnis in „traveltimegrouped“ abgelegt. Jedoch ist das Vorgehen effektiv etwas komplizierter, da „traveltime“ eine extrem grosse Tabelle ist und mit „fplan_calculated“ nur eine indirekte Beziehung zur Start-Haltestelle führt. Die SQL-Skripte, mit denen dieser Prozess durchgeführt werden kann, sind auf der CD enthalten.

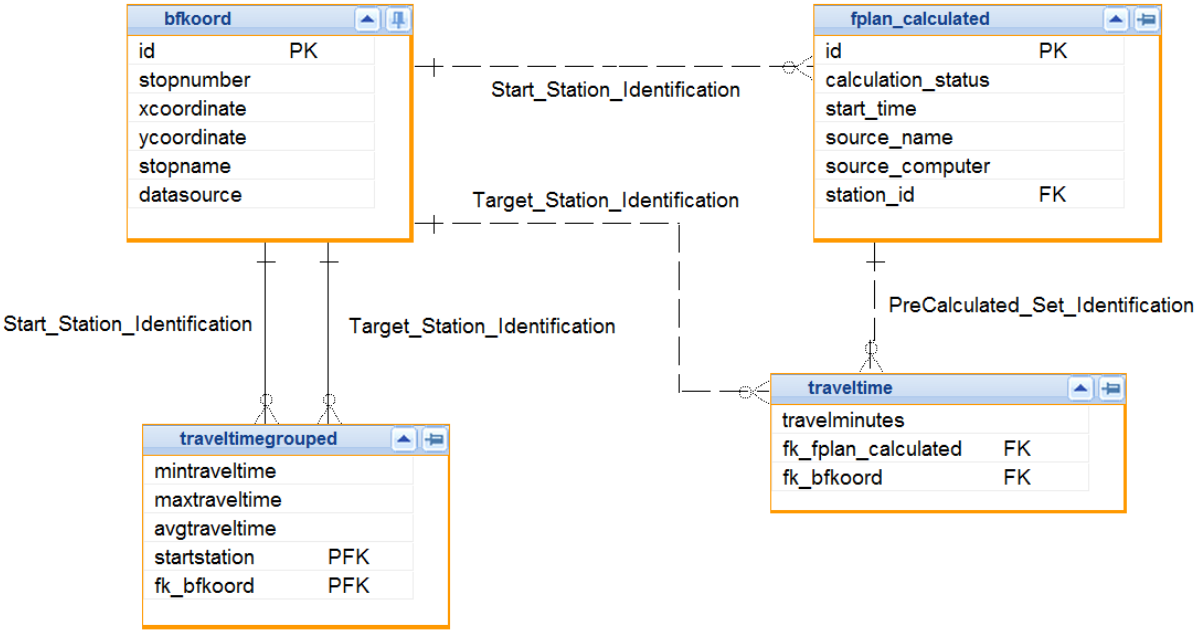



Abbildung 13: Schema der Vorberechnungs-Datenbank (ecomb33.ecomb.ch)

## 2.3 Software-Architektur

Der Aufbau der internen Architektur des CIPT-Dienstes wurde während der Bachelorarbeit an zwei Punkten angepasst. Die grobe Struktur wurde jedoch nicht verändert. Für grundlegende Informationen zur Architektur wird deshalb auf die Semesterarbeit verwiesen (Birchler & Lynn, 2011, S. 40ff.).
Die erste Anpassung wurde auf Grund der zusätzlichen Geo-Formate vorgenommen, welche der CIPT-Dienst exportieren kann. Bis anhin wurden die GML-Daten direkt im CIPT-Algorithmus erstellt. Eine neue erstellte Factory-Bibliothek wandelt nun das Resultat des CIPT-Algorithmus nach GML, KML oder GeoJSON. Es wäre nun relativ einfach möglich, die Factory-Bibliothek durch eine neue Klasse zu erweitern, um mit dem CIPT-Dienst ein neues Daten-Format zu unterstützen.
Die zweite Anpassung bezieht sich auf die Funktionsweise des WPS. Dieser muss nun eine vierte Anfrage beantworten können. Dies führte ebenfalls zu einer kleinen Anpassung der Architektur.

## 2.4 Sequenzdiagramme

Das System-Sequenzdiagramm (**Fehler! Verweisquelle konnte nicht gefunden werden.**) zeigt die verschiedenen Anfragen an den Server, welche bei der Verwendung des WPS gemacht werden. Die ersten drei Anfragetypen müssen von jedem WPS behandelt werden können. „GetCapabilities“ liefert eine Liste aller Prozesse/Funktionen, die der WPS zur Verfügung stellt. „DescribeProcess“ übermittelt eine detaillierte Beschreibung eines ausgewählten Prozesses und mit „Execute“ kann der Prozess effektiv ausgeführt werden.

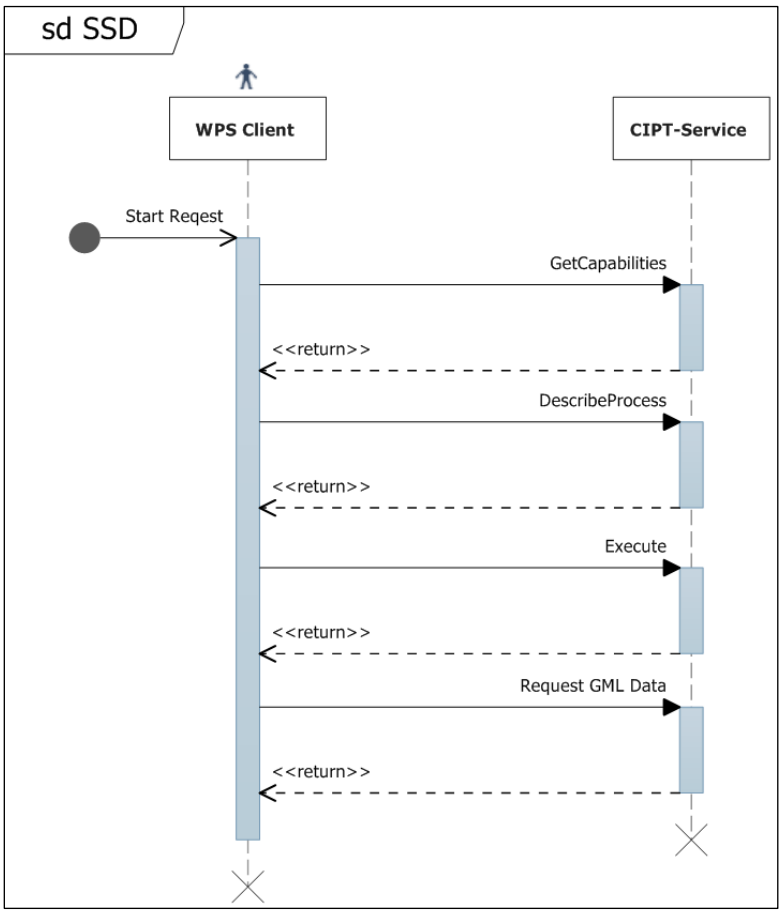



Abbildung 14: Systemsequenz-Diagramm für einen Datenabruf mit WPS

Bis anhin wurde versucht, einen Execute-Request des WPS direkt mit den errechneten CIPT-Daten zu beantworten. Dies ist nach den WPS-Spezifikationen (OGC Web Processing Service Standard) ein durchaus übliches und erlaubtes Vorgehen. Das WPS Plug-In für QGis erwartet jedoch als Antwort auf einen Execute-Request nicht direkt die Daten, sondern eine URL, welche die Daten zum Abruf bereithält. Dies hat zur Folge, dass der Abruf von Daten mittels WPS nun eine zusätzliche Anfrage an den CIPT-Dienst stellt. Neu hinzugekommen ist der letzte Aufruf, welcher als Antwort die Geo-Daten im GML-Format erhält.

Abbildung 15 zeigt den sequentiellen Ablauf einer CIPT-Abfrage mittels REST. Nachdem geprüft wurde, ob das geforderte Abfrageergebnis nicht bereits im Cache ist, wird bei einem „Cachefail“ eine Liste aller Haltestellen geladen, welche erreicht werden können. Je nach gewünschtem Objekttyp (Isochrone oder Punkte) wird diese Liste zusammen mit der Format-Angabe (GML, KML oder GeoJSON) an eine passende Factory-Klasse übergeben. Diese Factory generiert danach den Antwort-String.

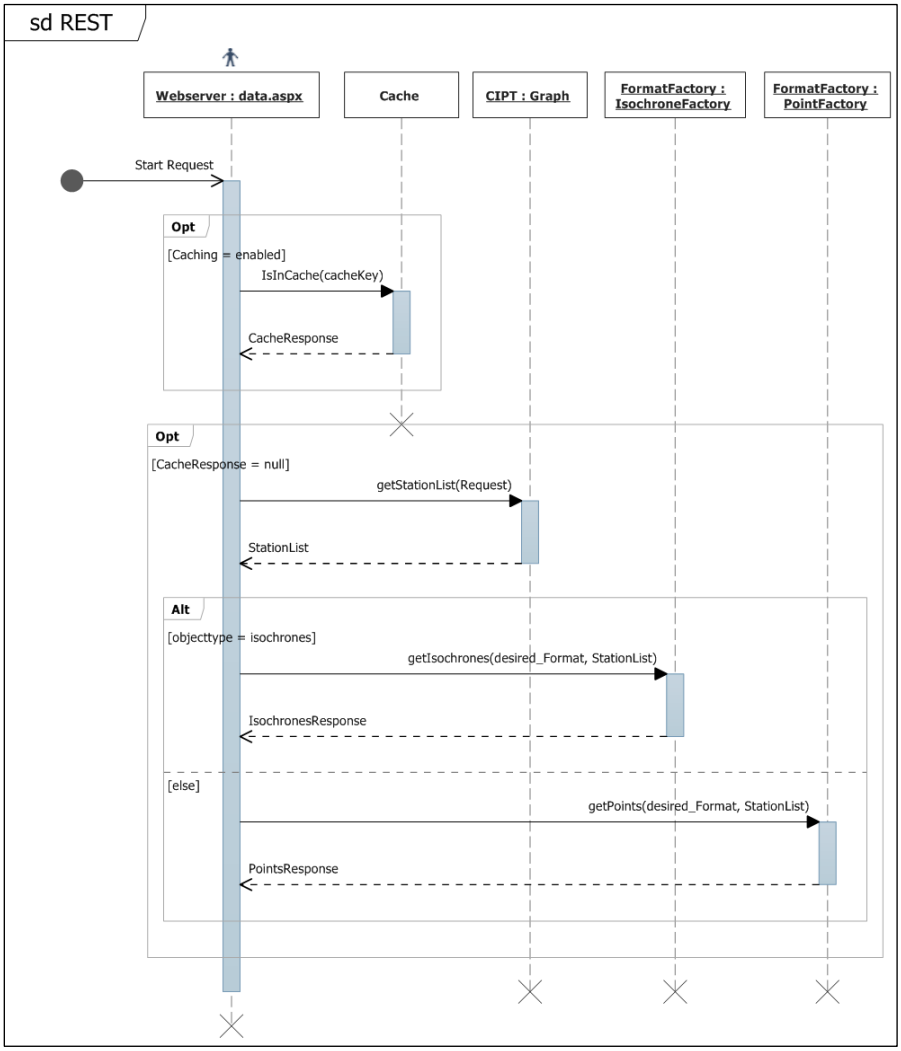



**Abbildung 15: Sequenzdiagramm, Datenabruf mittels REST**

## 2.5 Konfiguration

Der CIPT-Dienst hat nach dem Update keine wesentlich höheren Anforderungen für den Betrieb, ausser mehr Speicherplatz für die vorberechneten Daten. Für die Ausführung des ASP.NET Codes genügt ein aktueller Microsoft-Webserver (IIS 7.0). Die wichtigsten Konfigurations-Parameter können direkt in der Datei „web.config“ im Hauptverzeichnis der Applikation erfasst werden. Der Source-Code muss deshalb bei einer Parameter-Änderung nicht neu kompiliert werden.

| Parameter-Name | Funktion |
|---|---|
| **dbConnectionString** | PostgreSQL-Connection-String für die Datenbank, welche die Fahrplandaten enthält. |
| **preCalculatedDbConnectionString** | PostgreSQL-Connection-String für die Datenbank mit den vorberechneten Daten. |
| **walkingSpeedKmH** | Geh-Geschwindigkeit für die Berechnung des Rest-Zeit-Kreises bei den Endstationen. |
| **maxIsoNumber** | Obere Grenze für die Anzahl Isochronen, welche vom CIPT-Dienst für eine Anfrage berechnet werden. |
| **maxTravelMinutes** | Obere Grenze für die Reiseminuten, welche vom CIPT-Dienst für eine Anfrage verwendet werden. |
| **maxPrezision** | Obere Grenze für die Präzision, welche vom CIPT-Dienst bei der Zeichnung von Isochronen verwendet wird (Anzahl Punkte pro Quadrant). |
| **cacheMinutes** | Anzahl Minuten die ein Cache Eintrag erhalten bleiben soll. |

Tabelle 11: Konfigurations-Parameter

### 2.6 Codestatistik

Die untenstehende Tabelle gibt einen Überblick über einige statistische Werte des aktuellen Codes. Detaillierte Angaben können in einem Excel-Dokument auf der CD gefunden werden (/Codestatistik).

| Project | Maintainability Index | Cyclomatic Complexity | Depth of Inheritance | Class Coupling | Lines of Code |
|---|---|---|---|---|---|
| Cipt | 79 | 274 | 3 | 50 | 700 |
| CIPT Vorberechner | 96 | 107 | 1 | 49 | 173 |
| FormatFactory | 66 | 184 | 3 | 32 | 548 |
| OgcGeoServer | 79 | 145 | 2 | 32 | 268 |
| TestCipt | 67 | 28 | 3 | 19 | 163 |
| TestFormatFactory | 58 | 3 | 1 | 6 | 38 |
| TestOgcGeoServer | 54 | 26 | 1 | 8 | 160 |

Tabelle 12: Codestatistik

## 3. Tests

Während dem Programmieren des CIPT-Dienstes wurde der neu erstellte Code immer wieder mit Unit-Tests kontrolliert. Ein Build-Prozess, welcher automatisch einmal pro Tag gestartet wurde, führte die Unit-Tests ebenfalls aus und versendete Benachrichtigungen, falls dies einmal nicht funktionierte.
Durch den Web-Client war es während dem Programmieren des CIPT-Dienstes jederzeit möglich, die errechneten Ergebnisse des Algorithmus visuell darzustellen. Grob falsche Ergebnisse sind dabei sofort aufgefallen.

Ein Akzeptanz-Test wurde am Ende der Bachelorarbeit mit QGis (Version 1.8.0-Trunk) durchgeführt. Für alle drei Dateiformate (GML, KML und GeoJSON) wurden sowohl der Export von Isochronen (Polygone), sowie auch der Export der erreichbaren Haltestellen (Punkte) erfolgreich getestet.

# 4. Weiterentwicklungen

Mögliche Weiterentwicklungen der CIPT-Software wurden bereits in Kapitel 6 des ersten Teils besprochen.

# 5. Projektmanagement

Bei der Bachelorarbeit muss ein Arbeitsaufwand von ungefähr 360 Stunden innerhalb eines Semesters geleistet werden. Dabei ist es empfehlenswert, den Arbeitsaufwand möglichst ausgeglichen über die 14 Wochen des Semesters aufzuteilen. Um dies erfolgreich umsetzen zu können, ist es wichtig, das Projektmanagement sorgfältig durchzuführen.

## 5.1 Betreuung

Während der 14-wöchigen Projekt-Periode wurde Marco Birchler durch Herrn Prof. Stefan Keller betreut. Pro Woche wurde jeweils eine 90-minütige Sitzung am Arbeitsplatz von Herrn Keller abgehalten. Ebenfalls wurden zwischen den einzelnen Meetings per Email Fortschritte erörtert und Probleme diskutiert.

## 5.2 Meilensteine

Die Meilensteine, welche zu Beginn für das Projekt gesetzt wurden, sind in Tabelle 13 ersichtlich. Ebenfalls wird beschrieben, ob diese eingehalten werden konnten.

| Meilensteine | |
|---|---|
| **Entscheid Vorberechnung, 16.10.2011** | Aufgrund einer vorhergehenden Analyse wird entschieden, ob es eine sinnvolle und innerhalb der gegebenen Fristen mögliche Lösung für eine Vorberechnung der Daten gibt.<br>• Ja: die Vorberechnung wird umgesetzt<br>• Nein: keine Umsetzung, zusätzliche Erweiterungen für den Algorithmus implementieren |
| **Ergebnis:** | Vorberechnung kann sehr wohl umgesetzt werden. Der weitere Projektplan muss angepasst werden, da einige Verbesserungen/Änderungen am Algorithmus vor der Vorberechnung umgesetzt werden müssen. |
| **Algorithmus Freeze, 13.11.2011** | Der Algorithmus hat einen stabilen Zustand erreicht und wird nicht mehr verändert. Ob der Algorithmus nun mit einer Vorberechnung abläuft oder nicht, ist unerheblich. Die nun zu erstellende Demo-Webseite muss auf die vorhandenen Möglichkeiten und Limitationen des Algorithmus eingehen. |
| **Ergebnis:** | Der eigentliche CIPT-Algorithmus wurde nicht verändert. Gemäss |

| | |
|---|---|
| | Aussagen von seriösen Quellen gibt es für die Problemstellung unserer Anwendung keine bessere Lösung als den Dijkstra Algorithmus. Zeitmessungen haben ausserdem ergeben, dass der CIPT-Algorithmus gut skaliert. Die Genauigkeit der errechneten Ergebnisse wurde jedoch gesteigert indem der Algorithmus nun benachbarte Bahnhöfe mitberücksichtigt und die benötigte Zeit für einen Umsteigevorgang ermittelt. Das eigentliche Performance-Problem ist die Aufbereitung der Isochronen aus den Ergebnissen des CIPT-Algorithmus. |
| **Demo Webseite fertig, 11.12.2011** | Die Webseite für die Demonstration des Algorithmus und des Web 2.0 API ist fertig. Die restliche Zeit kann für den Abschluss der Dokumentation und die Erstellung des Videos und Plakats verwendet werden. |
| **Ergebnis:** | Der Meilenstein konnte zeitlich nicht eingehalten werden. Die Programmierung mittels OpenLayers benötigt mehr Einarbeitungszeit als vorgesehen. Grosse Fortschritte sind jedoch ersichtlich. Es kann deshalb mit der Fertigstellung der Webseite in der kommenden Woche gerechnet werden. |

**Tabelle 13: Definition der Meilensteine**

## 5.3 Beschlussprotokolle

Es wird darauf verzichtet, die Beschlussprotokolle hier einzufügen. Die Sitzungsprotokolle sind auf der CD (30_Projekt_Management / Protokoll-Sammlung.xls) enthalten.

## 5.4 Aufwandschätzung, Zeitplan, Projektplan

Der Projektplan wurde zu Beginn der Bachelorarbeit erstellt und musste danach noch zweimal angepasst werden. Tabelle 14 zeigt den Plan nach der zweiten Änderung. Die beiden früheren Versionen des Plans sind auf der CD (30_Projekt_Management / Projekt_Planung.xls) abgelegt. Die Anpassungen des Projektplans wurden auf Grund Meilenstein-Resultate vorgenommen.

| Kalenderwoche: | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Themenfindung | | | | | | | | | | | | | | | |
| Planung | | | | | | | | | | | | | | | |
| QGis mit WPS | | | | | | | | | | | | | | | |
| Analyse Vorberechnung | | | | | | | | | | | | | | | |
| Algorithmus erweitern 1 | | | | | | | | | | | | | | | |
| -Berücksichtigung Umsteigezeit | | | | | | | | | | | | | | | |
| -Berücksichtigung Nachbarhaltestellen | | | | | | | | | | | | | | | |
| Umsetzung Vorberechnung | | | | | | | | | | | | | | | |
| Algorithmus erweitern 2 | | | | | | | | | | | | | | | |
| Optimierung der Isochronengenerierung | | | | | | | | | | | | | | | |
| Web 2.0 API / Demo-Webseite erweitern | | | | | | | | | | | | | | | |
| Weitere Webservices* | | | | | | | | | | | | | | | |
| Video erstellen | | | | | | | | | | | | | | | |
| Plakat erstellen | | | | | | | | | | | | | | | |
| Benachr. und Dank an SBB/ZVV | | | | | | | | | | | | | | | |
| Risiko-Analyse | | | | | | | | | | | | | | | |
| Dokumentation | | | | | | | | | | | | | | | |
| Doku-Webseite | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| Milestones: | | | | | ← Entscheid Vorberechnung | | | | | | | | | | |
| | | | | | | | | | | ← Algorithmus Freeze | | | | | |
| | | | | | | | | | | | | | ← Demo Webseite fertig | | |
| *nur falls möglich | Haupttätigkeit | | | | Nebentätigkeit | | | | Optional | | | | | | |

Tabelle 14: Projektplan

## 5.5 Soll-Ist-Zeit Vergleich

Für eine Bachelorarbeit sollte folgende Anzahl Stunden aufgewendet werden:

| ECTS Punkte | Zeitaufwand |
|---|---|
| 1 ECTS | 25 – 30 h |
| 12 ECTS Bachelorarbeit | 360 h |

Tabelle 15: ECTS Punkte der Bachelorarbeit

Die eingesetzte Zeit wurde etwa wie folgt auf die verschiedenen Arbeitswochen aufgeteilt:

| KW | Soll (Stunden) | Ist (Stunden) | Haupt-Tätigkeit |
|---|---|---|---|
| **38** | 10 | 5 | Themenfindung |
| **39** | 20 | 15 | Planung |
| **40** | 20 | 20 | QGis WPS Plug-In |
| **41** | 20 | 20 | Analyse Vorberechnung |
| **42** | 25 | 20 | Algorithmus-Erweiterung 1 |
| **43** | 25 | 25 | Algorithmus-Erweiterung 1 |
| **44** | 25 | 35 | Umsetzung der Vorberechnung |
| **45** | 25 | 25 | Umsetzung der Vorberechnung |
| **46** | 25 | 25 | Algorithmus-Erweiterung 2 |
| **47** | 30 | 20 | Algorithmus-Erweiterung 2 |
| **48** | 30 | 30 | Opt. Isochronen-Generierung |
| **49** | 40 | 55 | Webseite und Demo-Client |
| **50** | 35 | 50 | Dokumentation |
| **51** | 30 | 45 | Dokumentation |
| **Total:** | **360** | **390** | |

Tabelle 16: Arbeitsaufwand nach Arbeitswoche

### 5.6 Risiko-Management

Zu Beginn der Bachelorarbeit (3. Woche, d.h. nach der Themenfindung und Planung) wurde eine Risiko-Analyse durchgeführt. Die fortlaufende Überwachung der identifizierten Risikobereiche führte dazu, dass die Risikobeurteilung in der 5. und 8. Woche wiederholt werden musste. Beide Male konnten die Eintrittswahrscheinlichkeiten und somit auch die Risiken insgesamt nach unten korrigiert werden, da der Inhalt der Bachelorarbeit bei jeder Prüfung konkreter wurde. Die detaillierten Risiko-Analysen sind auf der CD zu finden (30_Projekt_Management / Risikoplanung Bachelor-Arbeit.xls).

## 6. Softwaredokumentation

Die Softwaredokumentation ist auf der CD enthalten, zu finden unter „40_Software-Dokumentation“.

# Appendix A: Abbildungen

# Appendix B: CD Inhalt

| Inhalt | Pfad |
|---|---|
| Dieses Dokument | 10_Bachelor-Arbeit_CIPT.pdf |
| Poster | 20_Poster_BA_CIPT.pdf |
| Projekt-, Risiko-Management, Protokolle | 30_Projekt_Management |
| Softwaredokumentation, Manuals, Screencasts | 40_Softwaredokumentation |
| CIPT-Code | 50_CIPT_Code |
| Fahrplan-Import Tool (inkl. Code) | 60_HAFAS_Importer |
| Vorberechnungs Client (inkl. Code) | 70_Vorberechner_Client |
| Diverse Test-Daten | 80_Testing |
| Performance Mess Tool (inkl. Code) | 80_Testing / Performance-Mess-Tool |
| Diverse SQL-Skripte | 90_SQL-Skripte |
| Diverses | 100_Verschiedenes |

# Appendix C: Wörterverzeichnis und Abkürzungen

**CIPT:** Calculated Isochrones for Public Transport
**CIPT-Algorithmus:** Algorithmus für die Berechnung des kürzesten Wegs vom Startpunkt zu allen Zielpunkten, modifizierter Dijkstra-Algorithmus beschrieben in (Birchler & Lynn, 2011, S. 22)
**CIPT-Dienst:** Softwarekomponente, die den Zugriff auf den CIPT-Algorithmus mittels diverser Schnittstellen erlaubt
**Datensatz:** Zusammengehörende Sammlung von Daten. Der in Unterkapitel „Vorberechnung“ (Seite 16) erwähnte Datensatz wird in „Speicherbedarf eines Datensatzes“ (Seite 18) definiert.
**Fahrt:** Unter Fahrt wird die Bewegung eines öffentlichen Verkehrsmittels von einer Haltestelle zur nächsten Haltestelle verstanden.
**GeoJSON:** Auf Geometrie-Daten spezialisiertes JSON (JavaScript Object Notation)
**HAFAS:** HaCon Fahrplan-Auskunfts-System, ein Rohdatenformat der Firma HaCon (Hannover Consulting). Die Fahrplandaten der SBB und des ZVV wurden in diesem Format zur Verfügung gestellt.
**Heatmap:** Englische Bezeichnung für “Hitzekarte”, durch verschiedene Farben werden ortsbezogene Daten auf einer Karte dargestellt
**ÖV:** Öffentlicher Verkehr
**ÖV-Verbindungen:** Sammelbegriff für Zug-, Bus-, Tram- und Schiffsverbindungen
**QGis:** Quantum GIS ist ein Geoinformationssystem zur Bearbeitung ortsbezogener Daten
**REST:** Representational State Transfer
**Update:** In diesem Dokument ist mit Update normalerweise der Versionssprung des CIPT-Dienstes vom Endergebnis der Semesterarbeit zum Endergebnis der Bachelorarbeit gemeint
**WFS:** Web Feature Service
**WPF:** Windows Presentation Foundation
**WPS:** Web Processing Service

# Appendix D: Bibliografie

*Wikipedia - B-tree*. (1. 12 2011). Von http://en.wikipedia.org/wiki/B-tree abgerufen

Birchler, M., & Lynn, K. (2011). *Computing Isochrones in Multi-Modal, Schedule-Based Public Transport Networks.* Student Research Project Thesis, Hochschule für Technik Rapperswil, Rapperswil.

Bundesamt für Statistik. (2000). *Volkszählung 2000 - Pendlermobilität - Verkehrsmittel*. Von http://www.bfs.admin.ch/bfs/portal/de/index/themen/11/07/01/03/verkehrsmittel.html abgerufen

Davis, M. (2007). *Lin.ear th.inking*. Von http://lin-ear-th-inking.blogspot.com/2007/11/fast-polygon-merging-in-jts-using.html abgerufen

Davis, M. (kein Datum). *Secrets of the JTS Topology Suite.* Von www.refractions.net. abgerufen

Delling, D., & Wagner, D. (kein Datum). *Time-Dependent Route Planning.* Universität Karlsruhe.

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures.* Irvine: University of California.

Geisberger, R. (2011). *Advanced Ropute Planning in Transportation Networks.* Karlsruher Institut für Technologie.

jQuery Project. (kein Datum). *jQuery Project*. Von http://jquery.org abgerufen

Kuster, J., & Meier, H. R. (01. 03 2003). *Institut für Raum- und Landschaftsentwicklung (IRL).* Abgerufen am 02. 12 2011 von http://www.irl.ethz.ch/plus/people/perlik/finalA1_de.pdf

Microsoft Corporation. (10. 11 2011). *Improving .NET Application Performance and Scalability*. Von http://msdn.microsoft.com/en-us/library/ff647717.aspx abgerufen

*OGC Web Processing Service Standard*. (n.d.). Retrieved from http://www.opengeospatial.org/standards/wps

*Open Geospatial Consortium, Inc.* (n.d.). Retrieved from http://www.opengeospatial.org/

OpenLayers. (kein Datum). *OpenLayers: Free Maps for the Web*. Von http://openlayers.org/ abgerufen

OSM Foundation. (20. 11 2011). *OpenStreetMap*. Von http://www.openstreetmap.org/ abgerufen

Ruderman, J. (20. 11 2011). *Same origin policy for JavaScript*. Von https://developer.mozilla.org/en/Same_origin_policy_for_JavaScript abgerufen

Varszegi, J. (10. 11 2011). *How to Write High-Performance C# Code*. Von http://dotnet.sys-con.com/node/46342 abgerufen

# Appendix E: Persönlicher Bericht und Verdankung

Die Arbeit an diesem Projekt hat mir im Grossen und Ganzen Spass gemacht. Da es sich um eine Fortsetzungsarbeit meiner Semesterarbeit handelte, war ich mit der Ausgangslage gut vertraut. Wie sich herausstellte, war das Herzstück der Semesterarbeit, der CIPT-Algorithmus, nicht das eigentliche Performance-Problem, sondern die Erstellung der Isochronen. Zu Beginn der Arbeit wäre es mir jedoch nie in den Sinn gekommen, dies in Frage zu stellen und somit genauer zu untersuchen. Ich denke, dieser Fehler ist darauf zurückzuführen, dass man bei einer Fortsetzungsarbeit bereits so stark in die bestehende Lösung vertieft ist, dass es sehr schwierig ist, einen Schritt zurück zu machen und das Ganze nochmals objektiv zu betrachten. Dies ist sicher eine wichtige Lehre, die ich für mich persönlich gerne aus diesem Projekt mitnehme.
Gerne möchte ich mich an dieser Stelle bei Herr Prof. Keller Stefan für die kompetente Betreuung meiner Bachelorarbeit bedanken. Ebenso möchte ich mich bei Herrn Eisenhut Claude als Experte und Herrn Prof. Dr. Rinkel Andreas als Gegenleser bedanken.

# Appendix F: Eigenständigkeitserklärung

**Erklärung**

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.

Ort, Datum: Einsiedeln, 22.12.2011

Name, Unterschrift: Marco Birchler