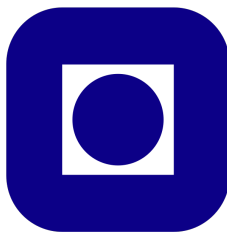


**Marius Qvam Wollamo**

# BusTUC - Geographic expansion

TDT4501 - Computer Science, Specialization Project, Technology. <sup>1</sup>  
Fall 2012

Department of Computer and Information Science  
Faculty of Information Technology, Mathematics and Electrical Engineering



---

<sup>1</sup><http://www.idi.ntnu.no/emner/tdt4501/>



## **Abstract**

BusTUC has been a valuable resource in the bus route information domain in Trondheim for several years. This project focuses on the steps needed to expand the domain in BusTUC to cover a greater part of Norway. Public transportation agencies want their data to be as easy accessible as possible to reach out to the customers. By implementing an automatic upload procedure of BusTUC an interface for the data exchange could be evaluated. The interface needs to be based on standard format that is supported by BusTUC. The results show that a automatic process could be implemented in order to achieve a scalable and modifiable solution, that could be extended in the future.



# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Task Description . . . . .	3
1.2	Background and Motivation . . . . .	4
1.3	Goals . . . . .	5
1.3.1	Geographic Expansion of BusTUC . . . . .	5
1.3.2	Automate the Schedule Update Process . . . . .	5
1.4	Add Real-time Information to BusTUC . . . . .	6
<b>2</b>	<b>Background</b>	<b>7</b>
2.1	BusTUC . . . . .	7
2.2	The RegTopp Format . . . . .	9
2.3	RegTopp Automaton . . . . .	9
2.4	The Real-time Service: MultiBRIS . . . . .	11
2.5	Server . . . . .	11
2.6	Related Work . . . . .	12
2.6.1	AtB's Travel Planner . . . . .	12
2.6.2	Google Transit . . . . .	12
2.6.3	Comparison . . . . .	15
<b>3</b>	<b>Method</b>	<b>19</b>
3.1	Functional Requirements . . . . .	19
3.1.1	Uploading a Data Set . . . . .	19
3.1.2	Converting the Data Set . . . . .	20
3.1.3	Extending BusTUC . . . . .	20
3.2	Non-Functional Requirements . . . . .	21

<b>4</b>	<b>Modeling</b>	<b>23</b>
4.1	Automating the New Schedules Upload Process . . . . .	23
4.2	Architecture . . . . .	23
4.3	Viewpoints of the Architecture . . . . .	24
4.3.1	Scenarios . . . . .	24
<b>5</b>	<b>Contributions</b>	<b>29</b>
5.1	Development Focus . . . . .	29
5.2	RegTopp to Prolog Converter . . . . .	29
5.3	Uploading and Verifying a Schedule . . . . .	32
5.4	Converting Data Sets . . . . .	32
5.5	Consistency in BusTUC . . . . .	32
5.6	JSON Answers . . . . .	33
5.7	Consistent Encoding . . . . .	34
<b>6</b>	<b>Results</b>	<b>35</b>
6.1	Automatic Update of BusTUC . . . . .	35
6.1.1	Performance . . . . .	35
6.1.2	RegTopp Automaton . . . . .	35
6.1.3	RegTopp to Prolog Converter . . . . .	36
6.2	Encoding . . . . .	36
6.3	BusTUC Answers . . . . .	36
6.3.1	The JSON Format . . . . .	36
6.3.2	Consistent Answers . . . . .	36
<b>7</b>	<b>Discussion</b>	<b>39</b>
<b>8</b>	<b>Future Work</b>	<b>41</b>
8.1	System Testing . . . . .	41
8.2	Extending to Other Transportation Types . . . . .	41
8.3	Knowledge Base . . . . .	42
8.4	Automating Processing: New Schedules from Transport Agency	42
8.5	Adopting New Bus Stops . . . . .	42
8.6	User Testing . . . . .	43
<b>9</b>	<b>Acknowledgments</b>	<b>45</b>

---

<b>A</b>	<b>A Revision of the Semantic Knowledge Base</b>	<b>47</b>
A.1	Batch Test . . . . .	47
A.1.1	New Routes . . . . .	47
A.2	Updating the Knowledge Base . . . . .	50
<b>B</b>	<b>Source Code</b>	<b>55</b>
	<b>Bibliography</b>	<b>57</b>





# List of Figures

2.1	The RegTopp Automaton . . . . .	11
2.2	AtB’s Travel Planner map result . . . . .	13
2.3	AtB’s Travel Planner text result . . . . .	14
2.4	Google Transit result . . . . .	15
4.1	Upload and convert . . . . .	23
4.2	Logic view when uploading new data set . . . . .	25
4.3	Process view of uploading . . . . .	26
4.4	Development view . . . . .	26
4.5	Deployment view . . . . .	27
4.6	Update new schedule scenario . . . . .	28
5.1	Class diagram Java Converter . . . . .	31
5.2	BusTUC no information answer . . . . .	33
5.3	BusTUC ambiguous place name: Solbakken . . . . .	33
5.4	BusTUC answer . . . . .	34
6.1	Batch test sample of result before revision . . . . .	37
6.2	Batch test sample result after revision . . . . .	37



# List of Tables

2.1	BusTUC web sites . . . . .	7
2.2	RegTopp version 1.2 . . . . .	10
2.3	Validating the RegTopp format . . . . .	10
2.4	Server information for busstjener.idi.ntnu.no and busstuc.idi.ntnu.no	12
2.5	Features in BusTUC, AtB’s Travel Planner and Google Transit .	16
3.1	Functional requirements for the RegTopp Automaton . . . . .	20
3.2	Functional Requirements for the Converter . . . . .	20
3.3	Functional requirements for a BusTUC extension . . . . .	20
3.4	Non-functional requirements . . . . .	21



## Terminology and Abbreviations

- **AtB** - Public transport agency in Trondheim, Norway
- **BusTUC** - A natural language bus route information system
- **IDI** - Department of Computer and Information Science
- **IME** - Faculty of Information Technology, Mathematics and Electrical Engineering
- **FUIROS** - The Future Ultimate Intelligent Route Information System
- **JSON** - JavaScript Object Notation, small data-objects for retrieving information
- **MultiBRIS** - Multiple-platform approach to the Ultimate Bus Route Information System
- **NTNU** - Norwegian University of Science and Technology
- **PHP** - A server-side scripting language popular in web-development to create dynamic web pages
- **RegTopp** - Regional Transit Information format
- **SMS** - Short Message Service. Allows sending short messages between phones and other hand-held devices
- **TT** - Team Trafikk, former bus provider in Trondheim
- **TUC** - The Understanding Computer, a natural language processor developed at IDI. BusTUC is an adaption of TUC to the bus domain.



# Chapter 1

## Introduction

This introduction will present the task description first, before presenting the background and motivation, and ending with the goals.

### 1.1 Task Description

The assignment, started by Tore Amble, was given by Rune Sætre at IDI. The FUIROS-project aims towards being the ultimate route information system for the future:

*FUIROS - Fremtidens ultimate intelligente  
ruteopplysningssystem*

*BusTUC is a natural language bus route system for Trondheim. It gives information about bus schedules, and has some information about the real passing times. AtB has installed GPS tracking on all the buses, giving access to real passing times and delays, and not only for Trondheim. With new smart phones arriving rapidly on the market, there are possibilities for GPS localizations, connections to maps and voice input. The project will be based on an existing advanced smart phone application called TABuss, developed by four MSc students. The current Bus Oracle system should be extended from Trondheim to cover a greater part of Norway, and it should support automatic updates whenever AtB change their regular schedules around holidays, and between winter and summer. The system could also be extended to include information about trams, boats, trains and airplanes.*

## 1.2 Background and Motivation

After the initial digitizing of bus schedules, various bus route information systems have been developed worldwide, aiming to present the information in a simple and understandable way.

”When is the next bus from Tiller to Samfundet leaving?” This question is one of many questions BusTUC, described in Amble (2000) and Bratseth (1997), a natural language bus route information system, has to answer daily. BusTUC is providing answers to the bus schedules in Trondheim and if a user asks:

*When does the next bus to Samfundet pass Moholt?*

This question will give an answer such as:

*Bus 36 passes by Moholt at 12:05 pm and at 12:35 pm and arrives at Studentersamfundet, 8 minutes later.*

This service provides the users information more efficiently than reading the entire time table.

BusTUC strives to return an answer that is reasonable for the question given. It has existed since 1997 and its expert knowledge makes it a valuable and intelligent application for bus travelers in Trondheim. After the startup of BusTUC the semantics<sup>2</sup> has been specialized for the bus domain in Trondheim and the system has been claimed to have reached a savant level of intelligence as described in Amble (2009). As a result BusTUC is easy to use and understands simple phrases such as: ”Solsiden Torget”. It processes that question as if the user had written: ”When does the bus from Solsiden to Torget leave?” This is one of several different ways of posing a question to BusTUC. The ability to understand a problem posed in different ways, which should give the same answer, is making BusTUC user-friendly. That is, a user does not have to learn a specific way of querying BusTUC.

Statistics show that BusTUC answers about 3000 questions on a daily basis. The service is accessible through the web, text message (SMS), and from various smart phone applications, making BusTUC a source of information that many rely on. The future vision is to extend BusTUC to give answers about trams, boats and trains. Currently BusTUC only provides schedules from the bus route operator AtB, but the FUIROS project aims to cover a greater part of Norway.

---

<sup>2</sup><http://en.wikipedia.org/wiki/Semantics>



Therefore, to provide an automatic update of BusTUC whenever AtB change their regular schedules, would be the start of expanding the system. To achieve an automatic update process, several formalities have to be considered. Data sets fed to the system should be in a supported format and use the new information together with its current semantic knowledge base.

## 1.3 Goals

The goals for this project is to automate the update process whenever the schedules from AtB are changed, prepare it for a geographic expansion and make this the basis for implementing real-time information in the master thesis.

The following sections explain how this project is going to implement these functionalities and modifications.

### 1.3.1 Geographic Expansion of BusTUC

The geographic expansion should provide an interface for public transportation types such as trams, trains and boats, to exchange their schedules so they could be used in BusTUC. To deal with other sources than AtB's schedules, BusTUC needs to generalize how the answers are returned. BusTUC provides answers in both text and JSON format. To prepare BusTUC for such changes, high cohesion and loose coupling should be the priority. The creation of JSON answers should be modified regarding the geographic expansion.

### 1.3.2 Automate the Schedule Update Process

Add support for automatic updates of BusTUC whenever AtB's schedules are changed. This is important to provide correct and intelligent answers. Automating such a process could help eliminating any human errors and make the update less time consuming. This process should contain a validation of schedules and convert the given data set to interpretable Prolog files. All scenarios which could trigger a change in schedules are likely to happen several times throughout the year. Therefore an automated solution is a good solution.

## 1.4 Add Real-time Information to BusTUC

The two goals described above should be the base for integrating real-time information as a part of the text answer in BusTUC for the Master thesis.

The previous FUIROS project MultiBRIS made a web service which provides real-time information about when the next buses are arriving at a specified bus station in a JSON format. This service is already accessible from the MultiBRIS client<sup>3</sup> and the android application TABuss.<sup>4</sup> A solution to integrate this information into BusTUC could be interesting. Collecting information from different sources into one system could make it get more users.

---

<sup>3</sup><http://busstuc.idi.ntnu.no/MultiBRIS>

<sup>4</sup><https://play.google.com/store/apps/details?id=test.BusTUC>

## Chapter 2

# Background

This section describes the technologies used for the development.

### 2.1 BusTUC

BusTUC can be accessed through both Short Message Service (SMS) and the web (see table 2.1). The answers are given as plain text or in JSON format. The latter allows simple integration in mobile applications.

Web site number three in Table 2.1 is the commercial version hosted by AtB. The two first web sites in Table 2.1 are used for development and testing purposes, before deploying to the commercial version. Note that the commercialized version could give different answers compared to the ones used for development. By asking BusTUC: "What version is running?", the answer should be "TUC has version AtB-I Date 121001". The date in this answer is date of the last update of BusTUC.

The BusTUC system consists of several components, where each has its own responsibility.

#	Web sites
1	<a href="http://busstuc.idi.ntnu.no/">http://busstuc.idi.ntnu.no/</a>
2	<a href="http://www.idi.ntnu.no/~tagore/">http://www.idi.ntnu.no/~tagore/</a>
3	<a href="http://www.atb.no">http://www.atb.no</a>

Table 2.1: BusTUC web sites

- Parsing the input text: This component consists of a parser that uses a dictionary and grammar to build a representation of the input in a parse tree. This parse tree is then evaluated to provide a semantic meaning.
- Semantical knowledge base: This database consists of a semantics network that maps a word to a meaning in a specific domain.
- Logical knowledge base: This database consists of several rules that need to be fulfilled in order to make an answer of the input.

Whenever a question is asked to BusTUC, it must perform an analysis and a reasoning before drawing a conclusion of what to return as an answer. BusTUC's interpretation relies on that TUC performs an analysis and translates a question into a first order expression without context. The context is provided by the semantic knowledge base in BusTUC.

First a lexical analyzer splits the question into tokens in a syntactic parse tree if and only if the question consists of words that are recognized by the analyzer. Each leaf node in the tree is representing the words such as a noun or a verb phrases. For example, if the question: "Who is superman?" is asked, the result returned is "Incomprehensible words: superman". Because the word "superman" is not in BusTUC's knowledge base and therefore the strict parser cannot translate the question into first order logic. This is to preserve the problem domain of BusTUC. However, if a user asks BusTUC the question "what is a man?" the returned answer is "I don't know". This is since the noun "man" is stored in the semantic knowledge base of BusTUC.

BusTUC translates from the tree representation to a first order logic expression called Tuc Query Logic (TQL). TQL expressions consist of predicates, functions, constants and variables. It gives an interpretation of a question as understood by BusTUC. The TQL expression is the input for doing a reasoning before BusTUC is able to provide the answer.

The semantic knowledge base in BusTUC is composed of rules and facts in Prolog. An example is when a user asks BusTUC "When is the next bus from downtown to Tiller leaving?" The TQL expression containing the word "Tiller" is recognized as a bus stop since this is a fact declared in the semantic knowledge base of BusTUC. This approach also applies to street addresses because they are mapped to the nearest bus stop.

The system is created to understand some misspelled words, and therefore, these are mapped to the correct word in the knowledge base.

BusTUC supports both Norwegian and English. The language detection is performed for each question by counting the number of words unrelated to both the languages. The system will answer in the language with the least unrelated words.

The reasoning in BusTUC is designed so that if a user does not specify two bus stops, the system assumes that the user wants to leave from downtown area of Trondheim. This assumption enables BusTUC to give a reasonable answer to an incomplete question.

If a user asks "From Tiller to Samfundet" and does not specify the time, an answer with the next departures from Tiller is returned. The time plays a very important role in the reasoning in BusTUC. When asking for a time that has passed, the system will return routes for tomorrow. This awareness is one of the features contributing to returning a reasonable answers to the user.

## 2.2 The RegTopp Format

Regional Trafikkopplysnings-format<sup>5</sup> (RegTopp), is a standardized format which is a result of the project "REGTOPP" which was running in the 1990s as described in Trafikanten (1996). It is used for exchanging transit schedules between public transportation agencies and travel planners. This format allows the use of different systems and still provides an understandable format when exchanging information. Together all the files in Table 2.2 make up the RegTopp format version 1.2 and define information such as the bus stops, where routes are going to, and what numbers the buses have.

## 2.3 RegTopp Automaton

The RegTopp Automaton<sup>6</sup> is a web site, developed by Rune Martin Andersen at the Faculty of Information Technology, Mathematics and Electrical Engineering, that allows a user to upload and download data sets<sup>7</sup> in a zip-file. The purpose of this site is to validate new data sets from AtB and make them publically available.

---

<sup>5</sup><http://labs.trafikanten.no/2011/4/13/dokumentasjon-av-regtopp-formatet.aspx>

<sup>6</sup><http://busstuc.idi.ntnu.no/regtopp>

<sup>7</sup>Data set - The format with the current schedule from a transport agency.

Files
FORMPAR.FRM
TURIX.TIX
TURMSTR.TMS
HPL.HPL
DAGKODE.DKO
DESTNAVN.DST
MERKNADMRK
GANGVEI.GAV
SAMTFK.SAM
SONE.SON
LINJE.LIN
VOGNLØP.VLP
TABVER.TAB
PERIODE.PER
RUTEPKT.RUT

Table 2.2: RegTopp version 1.2

Step	Check
1	Invalid zip file?
2	Too many data sets?
3	Missing file types in RegTopp format?

Table 2.3: Validating the RegTopp format

Before a data set gets stored at the host and ready for download, a simple validation of the zip file is executed. This process will run through the steps in Table 2.3 to decide if the data set should be approved or not. This validation contributes to preserving the consistency of the RegTopp format.

The approved data sets are listed and available for download as presented in Figure 2.1. A JSON format with the same information is also available.

AtB REGTOPP

Rutedata Last opp

Tilgjengelige datasett

[Liste over datasett som JSON\(P\)](#)

ID	Gyldighetsperiode	Oppdatert	Versjon	Filer	
R1613	25.12.2012-01.01.2013	07.12.2012 08:12	1.2	dko dst frm hpl ini mrk per rut sam tab tix tms vlp	<a href="#">Last ned</a>
R1614	24.12.2012-31.12.2012	07.12.2012 08:12	1.2	dko frm hpl ini mrk per rut sam tab tix tms vlp	<a href="#">Last ned</a>
R1612	24.10.2012-24.03.2013	23.10.2012 15:10	1.2	dko dst frm hpl mrk per rut sam tab tix tms	<a href="#">Last ned</a>
R1612	01.10.2012-23.12.2012	21.11.2012 19:11	1.2	dko dst frm hpl mrk per rut sam tab tix tms	<a href="#">Last ned</a>
R1617	12.09.2012-23.12.2012	17.10.2012 15:10	1.2	dko dst frm hpl mrk per rut sam tab tix tms	<a href="#">Last ned</a>
R1602	13.08.2012-23.12.2012	17.10.2012 15:10	1.2	dko dst frm hpl mrk per rut sam tab tix tms	<a href="#">Last ned</a>
R1602	12.01.2012-26.06.2012	17.10.2012 15:10	1.1-D*	dko hpl tda tix vlp	<a href="#">Last ned</a>

Figure 2.1: The RegTopp Automaton

## 2.4 The Real-time Service: MultiBRIS

The real-time service MultiBRIS Andersstuen and Engell (2011) returns the arrival times for all the buses and allows setting parameters as the current GPS position, the number of bus stops, and destination of the wanted trip. MultiBRIS is using a SOAP web service hosted by AtB. The client and the server use a JSON object for communication. By setting a bus stop's real-time ID as the input parameter, the query result should return a JSON object with information about the five next bus arrivals for the chosen bus stop.

## 2.5 Server

During the development of this project the server in Table 2.4 has been used. It provides the web service for BusTUC<sup>8</sup> and MultiBRIS<sup>9</sup>. It also provides the web interfaces for BusTUC<sup>10</sup> and the RegTopp Automaton<sup>6</sup>.

<sup>8</sup><http://busstjener.idi.ntnu.no/busstuc/oracle?q=samfundet>

<sup>9</sup><http://busstjener.idi.ntnu.no/MultiBRISserver/RealTime?bID=16010495>

<sup>10</sup><http://busstuc.idi.ntnu.no>

Attribute	Value
CPU	2x 5.2GHz, VMware shared pool
Memory	4GB dedicated
OS	Ubuntu 12.04.1 LTS

Table 2.4: Server information for busstjener.idi.ntnu.no and busstuc.idi.ntnu.no

## 2.6 Related Work

This section will identify similar solutions to BusTUC. BusTUC is the only natural language system that provides bus route information in Trondheim. Comparing the AtB Travel Planner and Google Transit could be a basis for new ideas. AtB Travel Planner is a direct competitor of BusTUC within the same domain, while Google Transit is a solution aimed at the entire world. This comparison should try to identify important features and properties that BusTUC could benefit from.

### 2.6.1 AtB's Travel Planner

AtB's Travel Planner<sup>11</sup> together with BusTUC constitutes the service provided at AtB's web site. This system requires the user to type a start and an end station before returning a travel plan. Both local and regional bus routes with an origin in Trondheim are supported. The travel route is presented as a list with detailed information of what kind of transport is possible and the departure times. An example of this is illustrated in Figure 2.3. In addition to the list, a feature that displays the selected route in a map can be chosen. Figure 2.2 shows a result of this feature.

### 2.6.2 Google Transit

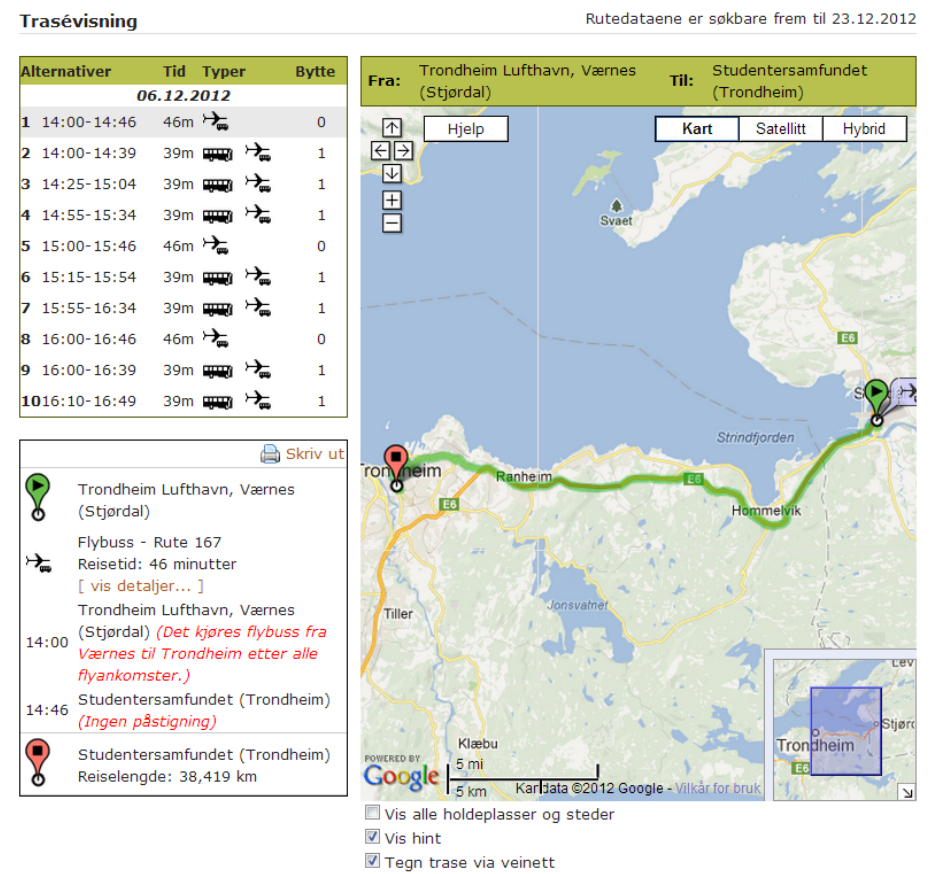
Google Transit<sup>12</sup> is a public transportation planning tool that integrates real-time and static transit data with the use of Google Maps<sup>13</sup>. It allows data providers to feed their data as long as it is in the defined format specified by Google. The two standard formats for exchanging static or real-time transit information is General

<sup>11</sup><https://www.atb.no/atbreiseplanlegger/>

<sup>12</sup><http://www.google.com/intl/com/landing/transit/>

<sup>13</sup><http://maps.google.no>





## Søkeresultat

Rutedataene er søkbare frem til 23.12.2012

## Søk fra Trondheim lufthavn, Værnes (Stjørdal) til Studentersamfundet (Trondheim)

(høyreklikk lenke for å bokmerke søket for senere bruk)

Torsdag 6. desember 2012 kl. 13:59


 Del søkeresultat

&lt;&lt; Dagen før &lt; Tidligere Senere &gt; Neste dag &gt;&gt;





Endre reise Returreise Videre Ny reise

▼ Skjul reisedetaljer

Start: 06.12.12 14:00		Slutt: 06.12.12 14:46		0 Overgang(er)		Reisetid: 46min		 	
Avgang	Type	Rutenr.	Fra	Til		Ankomst	Merknad		
14:00		167	Trondheim lufthavn, Værnes (Stjørdal)	Studentersamfundet (Trondheim)		14:46			

Start: 06.12.12 14:00		Slutt: 06.12.12 14:39		1 Overgang(er)		Reisetid: 39min		 	
Avgang	Type	Rutenr.	Fra	Til	Ankomst	Merknad			
14:00a		169	Trondheim lufthavn, Værnes (Stjørdal)	Torget (Trondheim)	14:35	Vent 1 minutt			
14:36		9	Torget (Trondheim)	Studentersamfundet (Trondheim)	14:39				

Start: 06.12.12 14:25		Slutt: 06.12.12 15:04		1 Overgang(er)		Reisetid: 39min		 	
Avgang	Type	Rutenr.	Fra	Til	Ankomst	Merknad			
14:25a		169	Trondheim lufthavn, Værnes (Stjørdal)	Torget (Trondheim)	15:00	Vent 1 minutt			
15:01		60	Torget (Trondheim)	Studentersamfundet (Trondheim)	15:04				

Start: 06.12.12 14:55		Slutt: 06.12.12 15:34		1 Overgang(er)		Reisetid: 39min		 	
Avgang	Type	Rutenr.	Fra	Til	Ankomst	Merknad			
14:55a		169	Trondheim lufthavn, Værnes (Stjørdal)	Torget (Trondheim)	15:30	Vent 1 minutt			
15:31		8	Torget (Trondheim)	Studentersamfundet (Trondheim)	15:34				

Start: 06.12.12 15:00		Slutt: 06.12.12 15:46		0 Overgang(er)		Reisetid: 46min		 	
Avgang	Type	Rutenr.	Fra	Til		Ankomst	Merknad		
15:00		167	Trondheim lufthavn, Værnes (Stjørdal)	Studentersamfundet (Trondheim)		15:46			

Figure 2.3: AtB's Travel Planner text result

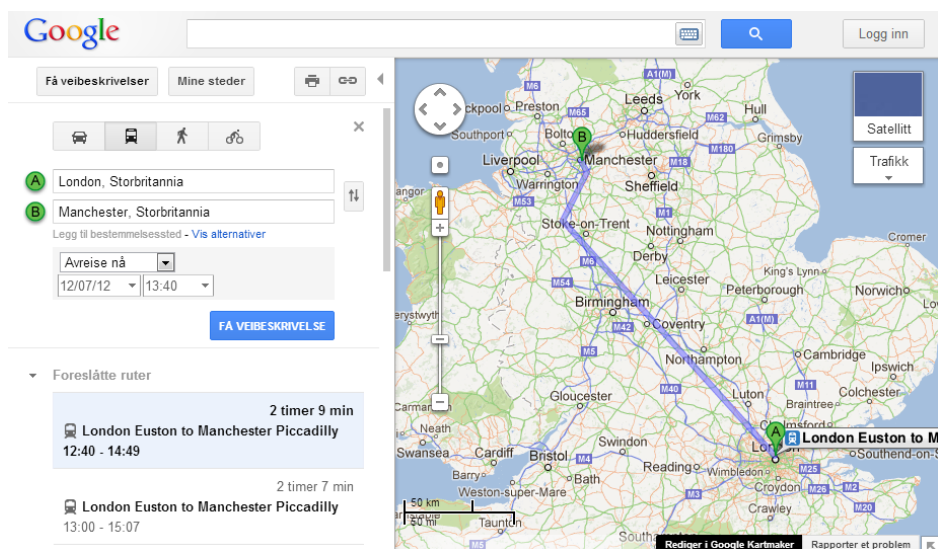


Figure 2.4: Google Transit result

Transit Feed Specification (GTFS)<sup>14</sup> and GTFS-realtime<sup>15</sup> respectively. Public transportation agencies all around the world are the targeted data providers. Fed data is updated once a week and allows users of Google Maps to search for this information. If a user wants to travel from place A to B, Google Maps could provide a result for public transportation possibilities. A search from London to Manchester, for instance, could return the travel plan by taking the train as in Figure 2.4.

### 2.6.3 Comparison

Both Google Transit and AtB's Travel Planner have various features and in Table 2.5 these are listed and compared to BusTUC.

<sup>14</sup><https://developers.google.com/transit/gtfs/?hl=no>

<sup>15</sup><https://developers.google.com/transit/gtfs-realtime/?hl=no>

Application	BusTUC	AtB travel planner	Google Transit
Real-time	No	No	Yes
Map	No	Yes	Yes
Multi language support	Yes	No	Yes
Local schedules	Yes	Yes	Yes
Regional schedules	No	Yes	Yes

Table 2.5: Features in BusTUC, AtB’s Travel Planner and Google Transit

The similar solutions all have different domains of interest. As Google Transit currently only provides public transportation from NSB<sup>16</sup> in Norway, a comparison of bus route information could not be made. However, Google Transit has a solution which utilizes the transit in a map. This visualizes where the suggested route is going for the user. This feature is also provided in AtB’s Travel Planner.

As the FUIROS project aims towards being the ultimate bus route information system for the future, one of the goals is to have as many users as possible. To keep its existing user base the answers returned should not consist of any false information that could lead to less confidence in the system. To preserve its current popularity, the identification of improvements has to be performed. That is, it is necessary to figure out what makes BusTUC easier to use than any other service. Albeit aiming for the future, BusTUC should also be backward-compatible in the supported schedule formats. After several contributions to the FUIROS project, BusTUC is now integrated in various smart phone applications. This strengthens its position in the market. By being an ultimate system, it is necessary to support an import of schedules that could easily be used. The FUIROS project has as a research project over the years brought new functionality and services. As the android application TaBuss, the cross platform application MultiBRIS, and the Speech-recognition functionality in TaBuss. However, the knowledge base in BusTUC has not been tested thoroughly. New bus routes and bus stops may not be updated in the semantic knowledge base.

As Google Transit<sup>17</sup> states on their website:

*”We believe that common formats for exchanging public transit information are the answer.”*

<sup>16</sup><http://www.nsb.no>

<sup>17</sup><https://developers.google.com/transit/overview?hl=no>

---

This quote is stating the vital part of providing a standardized and well defined format that is easy and understandable for the transportation agencies. The format that BusTUC currently relies on is the RegTopp format. As this is used by many transportation agencies in Norway, the reasonable choice is to keep supporting it as long as it is popular. That is, BusTUC should be an application that adapt to the environment and not the other way around, as formats come and go.



# Chapter 3

## Method

### 3.1 Functional Requirements

In order to achieve a robust automatic update process several requirements are set. These are the main focus when implementing this process. Each requirement is given an identifier, description and a priority for the development. The whole process of making BusTUC up to date with new schedules depends on the RegTopp Automaton and the converter described in Section 5.2. The functional requirements for these two components and BusTUC are described below.

#### 3.1.1 Uploading a Data Set

When AtB change their schedules a new data set is made. This data set must then be imported into BusTUC. An interface between the data providers and BusTUC will make this process simpler. The following requirements describe the functionality the RegTopp Automaton must provide in order to achieve this interface and initiate the automatic update process of BusTUC.

ID	Description	Priority
F1	The RegTopp Automaton shall execute the Converter programs	H
F2	The RegTopp Automaton shall give feedback to the user when the upload process is complete	M

Table 3.1: Functional requirements for the RegTopp Automaton

### 3.1.2 Converting the Data Set

When the new data set is uploaded, a converter process needs to be initiated. The conversion must import the information from the data set and update some existing files. To enable this the requirements in Table 3.2 are identified.

ID	Description	Priority
F3	The program shall update the version of BusTUC	H
F4	The program shall import the new data set in BusTUC	H
F5	The program shall write all the generated files in a UTF-8 format	L

Table 3.2: Functional Requirements for the Converter

### 3.1.3 Extending BusTUC

By extending some functionality in BusTUC, the issue with SMS answers could be addressed. The requirements are listed in Table 3.3.

ID	Description	Priority
F6	The system shall only return plain text for SMS queries	M
F7	The system shall provide reasonable answers for all the bus stops in its database	M

Table 3.3: Functional requirements for a BusTUC extension



## 3.2 Non-Functional Requirements

Table 3.4 identifies the non-functional requirements that the implementation of the automation should achieve.

ID	Description	Priority
NF1	The RegTopp Automaton shall approve and convert an up-loaded data set within one minute	H
NF2	All files should be in a UTF-8 encoding	M

Table 3.4: Non-functional requirements



# Chapter 4

## Modeling

### 4.1 Automating the New Schedules Upload Process

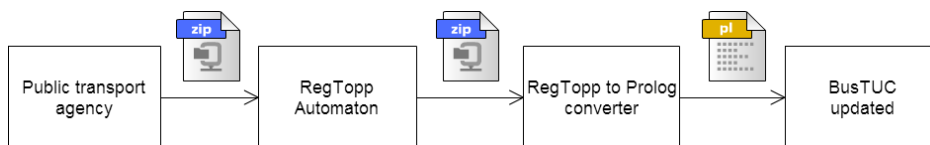


Figure 4.1: Upload and convert

Both the functional and non-functional requirements are the focus when implementing the automatic update process of BusTUC in Figure 4.1. An interaction between these components is required to achieve this solution.

### 4.2 Architecture

In this section the architecture of the automatic update process is presented. This is further divided into sections that describe the different views in the architecture.

The stakeholders for the architecture are listed below.

**AtB**

- **Useability:** The RegTopp Automaton should provide the means of uploading their schedules in a simple manner.

#### **Developer**

- **Architecture:** The upload process should set Modifiability as most important. This will take care of possible new data set formats in the future.

### **4.3 Viewpoints of the Architecture**

An architectural viewpoint is provided to present a broad and simple overview of the entire automation process. The viewpoints presented in this project is using Kruchten's 4+1 model as described in Kruchten (1995). This method allows the use of UML diagrams to represent the architecture and make it understandable for the stakeholders.

1. **Logic view** - Figure 4.2 displays what happens when the user uploads a new data set on the RegTopp Automaton.
2. **Process View** - Figure 4.3 is concerned with the steps after a new data set is uploaded to the RegTopp Automaton.
3. **Development View** - Figure 4.4 shows the different components and what kind of programming language they use.
4. **Physical View** - Figure 4.5 displays where the different components reside on the server.

#### **4.3.1 Scenarios**

The use case in Figure 4.6 represents the actions the two different actors need to carry out in order to update BusTUC with new information. The actors are represented by the user and system. The system is representing the automatic update process in BusTUC and the user is representing a data provider. AtB is the user which feeds BusTUC with data sets.

- **RegTopp Automaton:** The user must be able to upload its data on a website to let the data set be accessible and ready for an import to BusTUC.

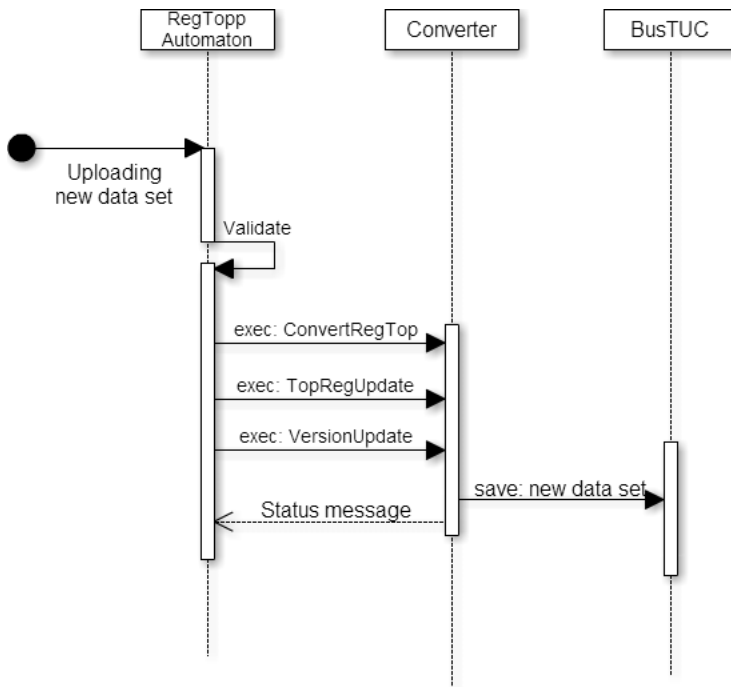


Figure 4.2: Logic view when uploading new data set

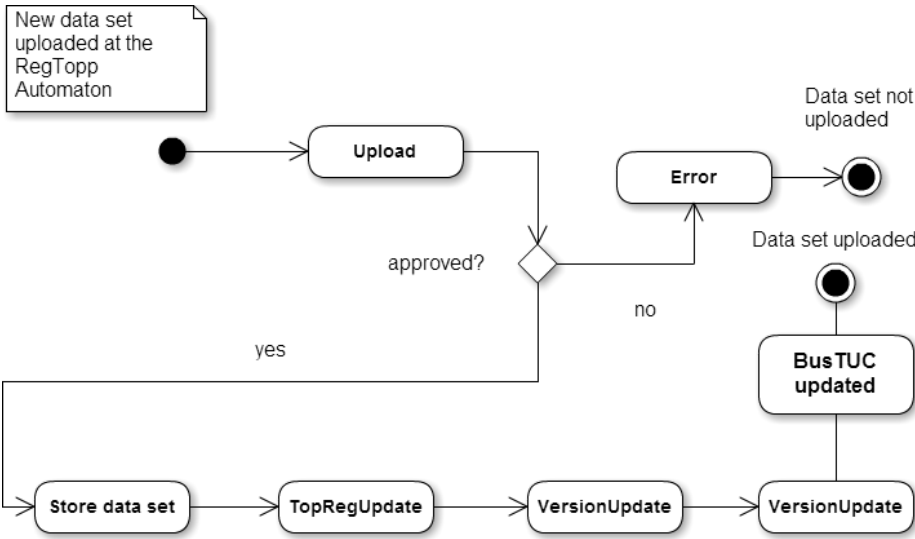


Figure 4.3: Process view of uploading

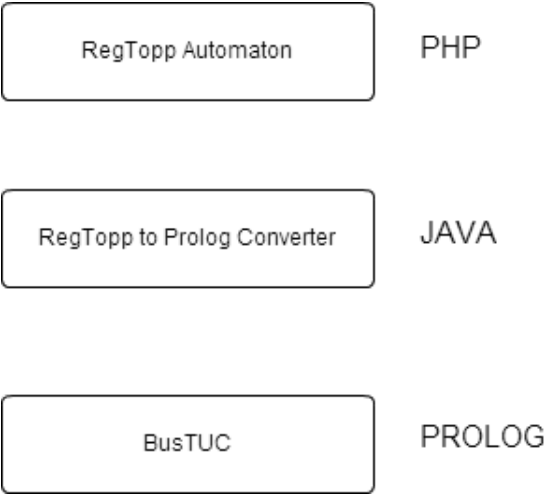


Figure 4.4: Development view

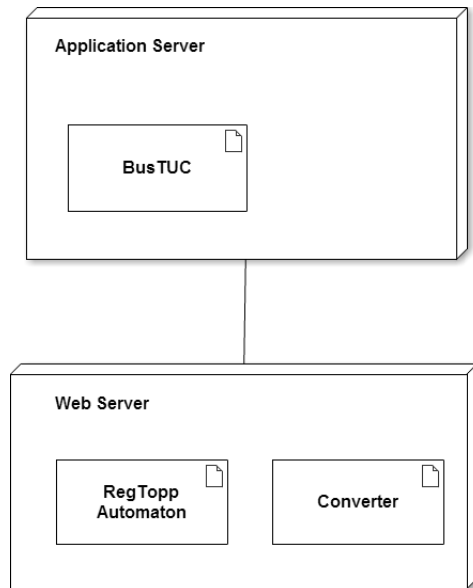


Figure 4.5: Deployment view

- **Convert data set:** The update system should perform a converting process that makes a data set useful for BusTUC.
- **Recompile BusTUC:** The update system shall by all means initiate a re-compilation of BusTUC to recognize and use the new data set.

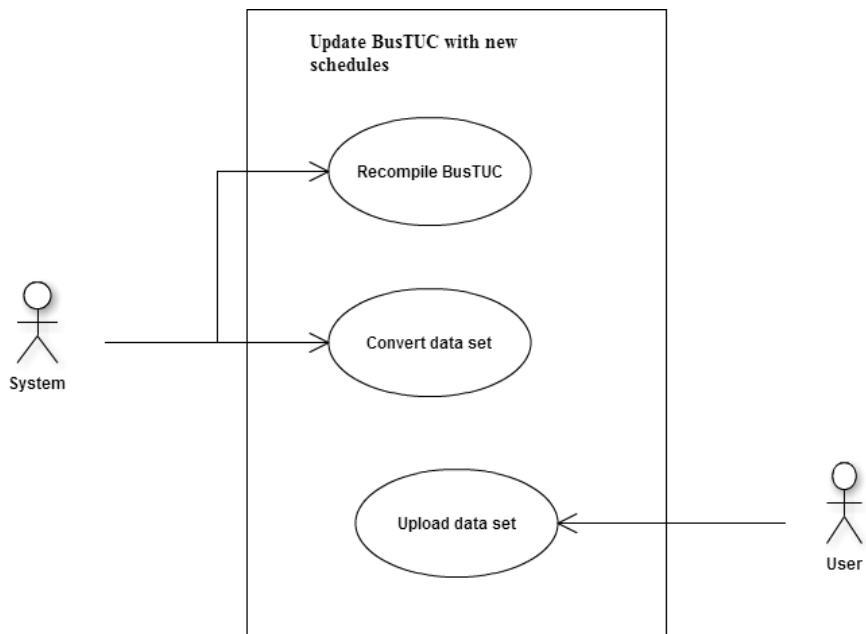


Figure 4.6: Update new schedule scenario



# Chapter 5

## Contributions

This chapter describes how the automatic update process of new data schedules into BusTUC was performed.

### 5.1 Development Focus

The development part focuses on making the components in Figure 4.1 communicate with each other. A public transport agency has its own management tools for making a schedule. If BusTUC shall cover a greater part of Norway, it is important that schedules are exchanged in a common format. This means that the agencies feeding BusTUC with data have to use a standardized format. The RegTopp format is currently the accepted and widely used in this area. Therefore, BusTUC should accept schedules provided in a RegTopp format. The different parts constituting the import process is described next.

### 5.2 RegTopp to Prolog Converter

The Converter is the program responsible for converting a data set into Prolog predicates that is interpretable for BusTUC. It is written in Java and designed for supporting data sets in a RegTopp format. The RegTopp format is well documented and specifies the index for each attribute. Thus, static methods are used to store the information in the RegTopp format as strings. The class diagram for the converter is depicted in Figure 5.1 and the coloring has the following

meaning:

- **Orange:** These two classes are developed by the author. They are necessary in order to update BusTUC with a new schedule.
- **Blue:** These packages contain utilities and objects used when processing the data set.
- **Green:** This is the program that converts the data set into Prolog predicates.

The converter package consists of the three executable programs: `ConvertRegTop`, `VersionUpdate` and `TopRegUpdate`. These constitute the interface between a RegTopp data set and the BusTUC system. A description of the programs follows:

**VersionUpdate** - The purpose of this program is to update the version of BusTUC. Since the last date of modification in BusTUC is constituting the version, then whenever new data sets are uploaded, the current date must be stored as well. This is performed by the `VersionUpdate` program. It basically creates a new date object and overwrites the current version file in BusTUC with the new information.

**TopRegUpdate** - The responsibility of `TopRegUpdate` is to add a predicate that informs BusTUC of the new schedules in its database. The file `topreg` consists of all the schedules in BusTUC's database. It is important that those are sorted by the end date. That is, to identify when new schedules should take over for the old ones, for instance, from summer to winter routes. The `TopRegUpdate` program does all of this. The imported schedule is added to `topreg` and then all predicates are sorted by end date.

Both `TopRegUpdate` and `VersionUpdate` will insert a comment if given when new schedules are uploaded at the RegTopp Automaton. This to easier understand why the different schedules are uploaded, for instance, a new route or changes in an existing route.

As these programs generate and modify files, a time stamp and the name of program are printed at the top of all the affected files. This provides a simple way of checking the time of the last update.

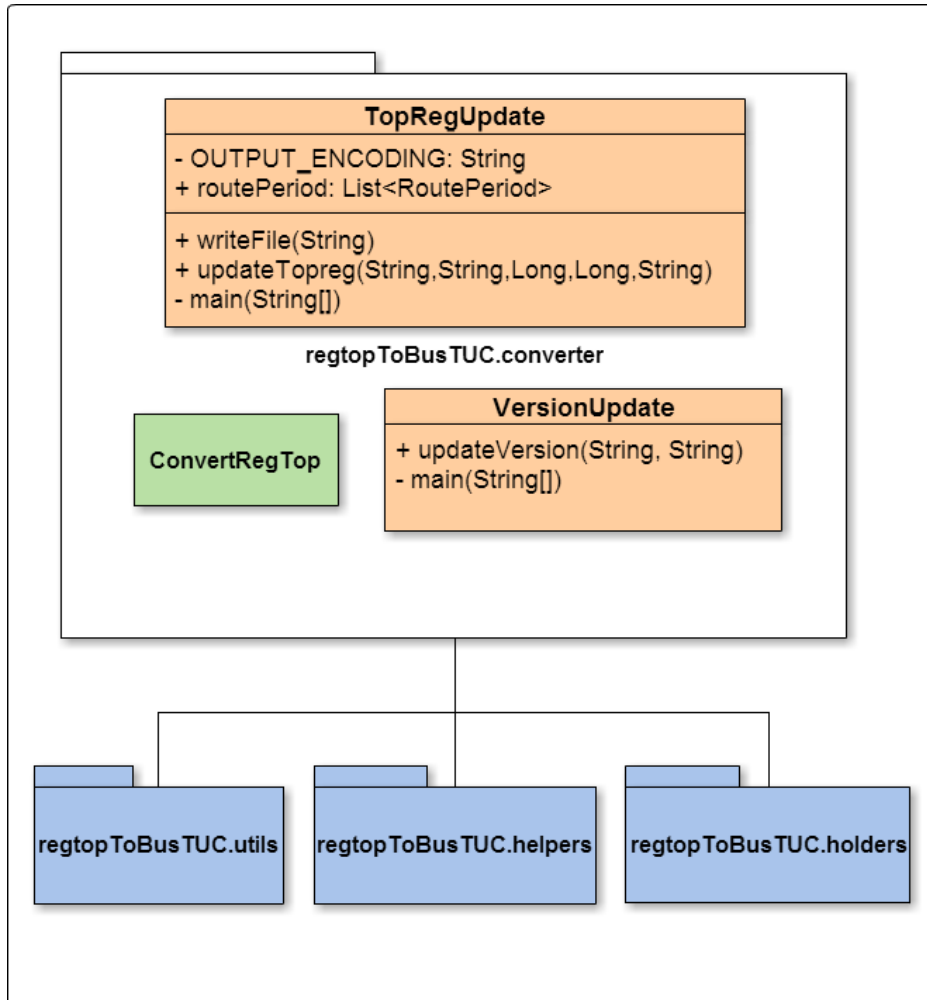


Figure 5.1: Class diagram Java Converter

All the programs have been modified to generate UTF-8 encoded files to maintain a consistent encoding throughout the system as explained in Section 5.7.

### 5.3 Uploading and Verifying a Schedule

The functionality of the RegTopp Automaton was extended to initiate the converting process. As both the upload web page and the Converter 5.2 are hosted on the same server an interaction is not a problem. PHP<sup>18</sup> provides a method for executing an external program, the `exec()`<sup>19</sup> method. By making the web site execute the Converter, the requirement F1 can be accomplished.

### 5.4 Converting Data Sets

Since BusTUC is written in Prolog, any new information provided must be converted to Prolog facts.<sup>20</sup> To make the Converter update BusTUC, two new classes were created. A new class was created to update the version of BusTUC when the new schedules were added. The other class was designed so that it could read and generate a file containing the names of the schedules in the database sorted by end date. This to tell BusTUC that the new schedule imported exists and is ready to be used from the valid period specified.

### 5.5 Consistency in BusTUC

Until the summer of 2010, it was TT who had the responsibility for providing a bus service for Trondheim and vicinity. After AtB took over the routes in August 2010, the change of service provider entailed new routes, like bus 301 to Stjørdal, Hommelvik or Brekkåsen. As BusTUC is dependent on RegTopp 2.2, new bus stops should automatically be interpreted as bus stops. However, the knowledge base has over the years been modified to state which places BusTUC has no information about. The result is better presented in Figure 5.2.

---

<sup>18</sup><http://www.php.net/>

<sup>19</sup><http://php.net/manual/en/function.exec.php>

<sup>20</sup>[http://en.wikipedia.org/wiki/Prolog#Rules\\_and\\_facts](http://en.wikipedia.org/wiki/Prolog#Rules_and_facts)

```
N: When is the bus from Samfundet to Malvik leaving?
```

```
~~~~~
```

```
I have only routes for buses in Trondheim .
```

```
~~~~~
```

Figure 5.2: BusTUC no information answer

The problem was that after importing the bus stops from the data set, the logic told BusTUC that Malvik was a foreign place. The interpretations to some bus stops clearly has to be changed manually in the semantic knowledge base. This could be a challenge with regard to the time consumption and by forgetting some bus stops. Being a trusted source of bus route information, BusTUC should be consistent with the current bus schedule. To fix this problem a manual revision was done giving a new answer, see Figure 5.4, to the same question as in Figure 5.2. However, new difficulties arose. Some of the bus stop names were ambiguous so BusTUC did not understand what the user wanted.

```
N: When is the bus from Samfundet to Solbakken leaving?
```

```
~~~~~
```

```
The place Solbakken is not unique .
```

```
Please use a more specific name.
```

```
Possible alternatives are: Solbakken bru , Solbakken skole .
```

```
~~~~~
```

Figure 5.3: BusTUC ambiguous place name: Solbakken

As in the Figure 5.3 the name "Solbakken" is a place which could refer to several locations in the Trondheim region. The listed alternative does not display the place Solbakken located in Hommelvik.

## 5.6 JSON Answers

By manually reviewing the implementation of the JSON functionality the functional requirement F6 in Table 3.3 could be accomplished. BusTUC has the

```

N: When is the bus from Samfundet to Malvik leaving?

~~~~~
{"transfer":"false",
 "timeset":"false",
 "departures" : [
 {"busstopname":"Studentersamfundet","busstopnumber":16011476,"busnumber":301,"time":2004,"duration":40,"destination":"Malvik"} ]}

Bus 301 passes by Studentersamfundet at 8:04 pm , at 9:04 pm , at 10:04 pm and at 11:04 pm
and arrives at Malvik , 40 minutes later .

The hours indicate the earliest passing times.
~~~~~

```

Figure 5.4: BusTUC answer

ability to give answers in JSON format as Figure 5.4 is showing. By default both the text and JSON answers are built. In cases when sending an SMS, a JSON format is not needed. The implementation of JSON in BusTUC was done ad-hoc making the modifiability of JSON answers rather poor. Therefore, the JSON functionality was moved to a higher level in the call stack, making it possible to omit the JSON format.

## 5.7 Consistent Encoding

One of the important parts of making an automatic update process is to ensure consistent encoding throughout the system. This means that files generated by the Converter and files on the server should be encoded in UTF-8 encoding. The reason for having only one encoding is to avoid problems that may occur when building or compiling programs. Since different encodings would give false character representations in cases where one of the Norwegian characters æ, ø or å is used. BusTUC has previously used an ISO-8859 encoding when running on the server *furu.idi.ntnu.no*. After migration to the server in Table 2.4, the encoding was set to UTF-8. Therefore, some of the files are still in a ISO-8859 encoding that could provoke issues when interpreting as UTF-8 encoding.

# Chapter 6

## Results

The following section describes the results of the development to enable a geographic expansion of BusTUC.

### 6.1 Automatic Update of BusTUC

#### 6.1.1 Performance

While no automatic update solution for importing new schedules existed earlier, the time of updating BusTUC is now taking about 2 minutes. The overall process from the time the user hits 'approve' when uploading a new schedule on the RegTopp Automaton until it gets approved and converted into an interpretable Prolog code takes less than a minute. Thus, this makes the update process more robust and free from human typing errors. The non-functional requirement NF1 applies in the cases where only one data set is uploaded at the time.

#### 6.1.2 RegTopp Automaton

Both the functional requirements F1 and F2 for the RegTopp Automaton were accomplished. An AJAX call is used to execute the converter programs. The response returned informs the user if the convert process was a success or not.

### 6.1.3 RegTopp to Prolog Converter

The extended functionality in the converter did succeed in fulfilling the functional requirements F3 and F4.

## 6.2 Encoding

After setting the language in the server 2.4 to UTF-8, all new files created were stored in this encoding and completing the requirement NF2.

### 6.3 BusTUC Answers

#### 6.3.1 The JSON Format

Due to the ad-hoc implementation of printing JSON to the screen inside a recursive predicate, unwanted side effects could happen. That is because of the impure predicates which are allowed in Prolog. A review of this implementation was performed and it was decided to extract the entries used for the JSON format in list formats. Then the ability to print all the JSON in one predicate was possible. This implementation omits the problems of side effects, and gives a more modifiable approach for extending this feature. The functional requirement F6 was accomplished and BusTUC now returns only plain text for an SMS. This helps reducing the data cost and the amount of data transferred to the client.

#### 6.3.2 Consistent Answers

There are several examples of consistency issues with BusTUC's answers compared to the current bus schedule. When importing new schedules, information like bus stops and places could conflict with the semantic knowledge base. There are no procedures to detect such problems. Therefore, a test was conducted to identify what kind of information gave incorrect answers. After executing the test, see Appendix A.1.1, several bus stops that should have been answered did not. An example of this answer is demonstrated in Figure 6.1. Here "Muruvik" should be interpreted as a valid bus stop, but the semantic knowledge has declared "Muruvik" as foreign. That is, BusTUC should not return any answer if a user asks for this specific bus stop. This problem applied to several bus stops.



```

E: muruvik.

~~~~~

I have only routes for buses in Trondheim .

~~~~~

```

Figure 6.1: Batch test sample of result before revision

```

E: muruvik.

~~~~~
{"transfer": "false",
 "timeset": "false",
 "departures": [
 {"busstopname": "Torget", "busstopnumber": 16011012, "busnumber": 301, "time": 1937, "duration": 52, "destination": "Muruvik"},
 {"busstopname": "Munkegata - M4", "busstopnumber": 16010004, "busnumber": 301, "time": 1945, "duration": 44, "destination": "Muruvik"} ]}

Bus 301 passes by Torget at 7:37 pm , at 9:07 pm and at 9:37 pm
and arrives at Muruvik , 52 minutes later .
Bus 301 passes by Munkegata M4 at 7:45 pm , at 9:15 pm and at 9:45 pm
and arrives at Muruvik , 44 minutes later .

The hours indicate the earliest passing times.

~~~~~

```

Figure 6.2: Batch test sample result after revision

After reviewing the test results, several bus stops matched the answer given in Figure 6.1. A manual revision of the semantic knowledge base was performed to enable BusTUC to answer the bus stops omitted, see Appendix A.2. Figure 6.2 presents the result after doing the revision.

The revision was able to detect bus stops from just one bus route since it was a time consuming task to perform. Therefore the functional requirement is not met for all the bus stops in BusTUC's database.



## Chapter 7

# Discussion

The automatic update process of BusTUC whenever AtB change their schedules is considered to be a success. Since most of the components already existed, a lot of time was used to learn the BusTUC system, the RegTopp Automaton and the RegTopp to Prolog Converter to find out how they worked.

One of the main goals for this project was to make BusTUC ready for an expansion to cover a greater part of Norway. This involves the generalization of formats which BusTUC has to support. Currently only the RegTopp format is supported by the upload website RegTopp Automaton. However, to provide information about bus stops and routes on a map in a format equal to that provided by Google Transit would be of interest. Since the only customer currently is AtB the support of a RegTopp format is expected. The process of updating BusTUC has become more robust and scalable because of its automatic import.

Adding real-time information in BusTUC was not implemented. This has its cause in the extra amount of time it took to get the overview of BusTUC that was needed in order to understand where the real-time information should be added. The MultiBRIS service which provides real-time information was not returning any results when queried. A parsing error occurred since the real-time information in AtB's real-time service had changed the JSON object. This problem was attempted to be fixed; however, the source code available was older than the one running on the server. Therefore, this could not be solved before the deadline of this project. However, the revision of JSON answers facilitates the implementation of real-time and has been one of the main focuses when doing the revision.

The RegTopp Automaton was made available to the author, which simplified the process of building an interface with the same functionality. An advantage is that it got documented and is a base for further improving the automation process.

One of the advantages of this project is the new documentation of the RegTopp Automaton and Java Converter which previously did not exist.

One of the issues of a geographic expansion is the already established semantic knowledge base. It contains more than 100,000 lines of code. This means that a manual revision is not a sufficient approach when introducing new schedules. In order to make use of new information, a revision has to detect all the possible connections a word has in the knowledge base. Making this process automatically may solve these issues.

## **Chapter 8**

# **Future Work**

### **8.1 System Testing**

Current testing runs a batch test containing 100 different questions. Every answer is approved manually by a tester. A way to improve this could be to do a batch test at least three times a day, at different times, or by simply setting BusTUC's clock. This could simulate the answers given in the evening/nights; mornings and mid-day, check that they are correct and give reasonable answers. Another way could be to try to set the specific values, such as the time, day, and holiday to get answers for each possible outcome. In some cases, like night bus and SMS it would not be a problem to set the values in the batch test. However, the time could be difficult to set since it is set and checked several times while executing the query. Also the night bus flag is triggered to be set if the question asked implies that night bus time tables are relevant to the answer.

### **8.2 Extending to Other Transportation Types**

The idea of extending BusTUC so it could be taken in use by other agencies implies that the architecture needs to be modified. Such planning should contain a set of requirements for what the input and the result from BusTUC should be returned as. The vast usage of the RegTopp format in public transportation suggests that it should be used as an input. By providing a general solution to many different agencies, the input should always be interpreted and converted into code, making BusTUC able to give answers in each case. Creating modules

which are responsible for each part should be considered. BusTUC has been taking this question into consideration from the beginning by preparing the Prolog code for supporting other vehicles.

### **8.3 Knowledge Base**

As BusTUC have been under development for several years, it has built up a huge knowledge base. This knowledge is specialized for the bus route domain in Trondheim. To be able to provide the same level of knowledge and intelligence in other places or for other domains like trams and boats, loose coupling between the parts of the system should be considered. This would allow an easy scalable and maintainable solution for the coming years.

### **8.4 Automating Processing: New Schedules from Transport Agency**

Since the vision of BusTUC is to be a standard for all public transport information, it needs to have proper import procedures. BusTUC has support for importing RegTopp data sets, but in the future this format could be in a newer version and therefore backward compatibility is vital. This is a concern along with the probably introduction of new standard formats and that is why a general import process should be considered.

### **8.5 Adopting New Bus Stops**

The idea that BusTUC should be extended to provide information about more than just bus stops and other public transports outside Trondheim the import of data sets needs to update the existing knowledge in BusTUC. By now the knowledge of places is approximately at 16,000 lines. This means that performing manual revisions each time a new data set comes along, is a too time-consuming task for one person. An automated check to test if some of the places/bus stops are listed as foreign should be considered to maintain consistency with current schedules.

## 8.6 User Testing

As aiming for the future BusTUC needs to conduct a proper user test to identify what the users like and dislike. It would be a great resource in further development of the system.





## **Chapter 9**

# **Acknowledgments**

I would like to thank my supervisors Björn Gambäck and Rune Sætre for their guidance and support. Rune has been a valuable resource for his great knowledge of the BusTUC system. I would also like to thank Rune Martin Andersen that made the source code for the RegTopp Automaton<sup>6</sup> available to me. I would also like to thank the previous FUIROS students.



## **Appendix A**

# **A Revision of the Semantic Knowledge Base**

In order to make a change in BusTUC's semantic knowledge base, a batch test or error log is the initial point of interest. When BusTUC returns answers some bus stops or place names could be misinterpreted and give a incorrect answer. A batch test and a manual revision could fix such errors.

### **A.1 Batch Test**

A batch test is a text file with predefined questions that are answered in order. Then a manual review is performed to check if the answers returned are correct and that the functionality works properly.

A batch test should be executed after each major modification in the logic or semantic knowledge base of BusTUC.

#### **A.1.1 New Routes**

Some bus stops are not answered by BusTUC because they are declared as unimportant in BusTUC's knowledge base. Therefore, it could be a inconsistency between the answer and the official schedules. Thus, a batch test could figure out how many of the bus stops are affected.

A test was made by asking BusTUC the question: "Which stations is the bus 301 passing?" BusTUC then listed all the bus stops and these were imported into

the batch test A.1.1.

---

\nodebug

\set queryflag true

\set smsflag false

\starttimebatch

Hva er din versjon ?

Askeladdvegen.

Bakkegate .

Blåhammaren .

Bratsbergvegen .

Brekksåsen .

Brekksåsen snuplass .

Brubakk .

Buran .

City Syd E6 .

Dalen Hageby .

Einar Tambarskjelves gate .

Ekra .

Gildheim .

Gimse .

Gimseflata .

Gjevingåsen .

Grønberg .

Halstad .

Hansensvingen .

Haugan .

Havneveien .

Hell .

Hellsenteret .

Hommelvik høgda .

Hommelvik stasjon .

Hundhamaren .

Johan Tillers vei E6 .

Kroppanbrua .

Kvitland .

Leistadkrysset .  
Malvik .  
Melhus sentrum .  
Melhus skysstasjon .  
Melhusbrua .  
Midtsandan .  
Munkegata M4 .  
Munkegata MR .  
Muruvik .  
Naustkleiva .  
Nova kinosenter .  
Olderdalen .  
Presthus .  
Prinsen kinosenter .  
Prof Brochs gate .  
Ranheim fabrikker .  
Rosendal .  
Rota .  
Rønningsbakken .  
Saksvik .  
Sandmoen E6 .  
Sandsiloen Rota .  
Saxenborg alle .  
Sjølyst .  
Skottvold .  
Skovgård .  
Smeplassen .  
Smiskaret .  
Solbakken .  
Solsiden .  
St Olavs hospital .  
Stav .  
Stjørdal stasjon .  
Storler .  
Storsand 1 Fv950 .  
Storsand 2 E6 .  
Storsand 2 Fv950 .  
Strandveien .  
Strindheim .  
Studentersamfundet .  
Tempe kirke .

```
Tonstadkrysset E6 .  
Torget .  
Torp .  
Torvet .  
Stjørdal .  
Travbanen .  
Trøndertun .  
Valøyvegen .  
Vikelvveien .  
Vikhammer .  
Vikhammerløkka .  
Vikhov .  
Være .  
Væresholmen .  
Væsletta .  
Værnesbranden .
```

```
\stoptimebatch
```

```
\set smsflag false
```

```
hvor mange spørsmål har du besvart ?
```

```
\end
```

---

## A.2 Updating the Knowledge Base

The semantically knowledge base in BusTUC consists of several files that have different responsibilities. The one in interest in this project is the Prolog file "places.pl" that contains predicates that describe information about places. This file should be modified if any bus stops are not returning any answers in BusTUC.

A predicate called `foreign` is telling BusTUC that the place is not in its domain. This to provide the user a reasonable error message and BusTUC will return an answer saying that it does not have any information about this station.

After executing the test A.1.1 some modifications in `places.pl` was carried out to make new bus stops interpretable. The predicates were commented out and the green lines represent these. A differentiation of the version before and

after the test A.1.1 is listed below.

---

Index: places.pl

```
=====
--- places.pl (revision 350)
+++ places.pl (revision 349)
@@ -3391,7 +3391,6 @@
    synplace(munkvollgård,munkvoll_gård) .
    synplace(murens,ourens) .
    synplace(murevik,muruvik) .
-synplace(muruvika,muruvik). %% MW-121119
    synplace(museet,museum) .
    synplace(musikmuseum,museum) .
    synplace(myen,byen) .
@@ -4401,14 +4400,6 @@
    synplace(vikaasen,vikåsen) .
    synplace(vikahammer,vikhammer) .
    synplace(vikahmmer,vikhammer) .
-synplace(vikhamar, vikhammer). %% MW-121119
-synplace(vikhamaren, vikhammer). %% MW-121119
-synplace(vikhamarn, vikhammer). %% MW-121119
-synplace(vikhammar, vikhammer). %% MW-121119
-synplace(vikhammaren, vikhammer). %% MW-121119
-synplace(vikhammarn, vikhammer). %% MW-121119
-synplace(vikhamer, vikhammer). %% MW-121119
-synplace(vikhammeren, vikhammer). %% MW-121119
    synplace(vikamer,vikhammer) .
    synplace(vikelv,vikelvveien). %%Complicated
    synplace(vikhamaråsen,vikhammeråsen) .
@@ -8223,7 +8214,7 @@
    foreign(brekken) .
    foreign(brekkerød) .
    foreign(brekktrøa) .
-%% foreign(brekkåsen). %% MW-121119 bus301
+foreign(brekkåsen) .
    foreign(brekstad) .
    foreign(bremanger) .
    foreign(bremsnes) .
@@ -8500,7 +8491,7 @@
    foreign(gibostad) .
    foreign(gimle). %% ?
```

```

foreign(gimlekollen).
-%% foreign(gimse). %% ? = Gimle(veien) %% bus301 %%
  MW-121119
+foreign(gimse). %% ? = Gimle(veien)
  foreign(giæverbukta).
  foreign(giske). %% Ålesund
  foreign(gjelleråsen). %% oslo
@@ -8654,7 +8645,7 @@
  foreign(helgeland).
  foreign(helgelandsmoen).
  foreign(helgeroa).
-%% foreign(hell). %% bus301 %% MW-121119
+foreign(hell).
  foreign(hella).
  foreign(hellandsjøen).
  foreign(helleland).
@@ -8916,7 +8907,7 @@
  foreign(kvinesdal).
  foreign(kvinherad).
  foreign(kvithamar).
-%% foreign(kvitland). %% bus301 %% MW-121119
+foreign(kvitland).
  foreign(kvæfjord).
  foreign(kværnerbyen).
  foreign(kvål).
@@ -9116,8 +9107,8 @@
  foreign(mosvika).
  foreign(moum).
  foreign(movik).
-%% foreign(muruvik). bus301 %% MW-121119
-%% foreign(muruvika). bus301 %% MW-121119
+foreign(muruvik).
+foreign(muruvika).
  foreign(myre).
  foreign(myrland).
  foreign(myrvåg).
@@ -9371,7 +9362,7 @@
  foreign(sagene).
  foreign(sagstua).
  foreign(sagvåg).
-%% foreign(saksvik). malvik bus301 %% MW-121119

```



```
+foreign(saksvik).
  foreign(saksvikovollen).
  foreign(salhus).
  foreign(salten).
@@ -9434,7 +9425,7 @@
  foreign(sjoa).
  foreign(sjusjøen).
  foreign(sjøholt).
-% foreign(sjølyst). %% malvik bus301 %% MW-121119
+foreign(sjølyst).
  foreign(sjølystveien). %% (Oslo)
  foreign(sjørdal).
  foreign(sjørdalen).
@@ -9519,7 +9510,7 @@
  foreign(smedstua).
  foreign(smertu).
  foreign(smestad). %% oslo
-% foreign(smiskaret). %% MW-121119
+foreign(smiskaret).
  foreign(smøla).
  foreign(smørås).
  foreign(snartemo).
@@ -9574,7 +9565,7 @@
  foreign(statsbygd).
  foreign(statsbygda).
  foreign(staubø).
-% foreign(stav). %% malvik/hommelvik %% MW-121119
+foreign(stav). %% malvik/hommelvik
  % foreign(stavanger).
  foreign(stavern).
  foreign(stavsjø). %% TA-110502
@@ -9629,7 +9620,7 @@
  foreign(stornes).
  foreign(storo). %% Oslo
  foreign(storoddan).
-% foreign(storsand). %% MW-121119 bus 301
+foreign(storsand).
  foreign(storskogen).
  foreign(storslett).
  foreign(storsteinnes).
@@ -9780,7 +9771,7 @@
```

```
foreign(tonsenhagen) .
foreign(tonsåsen) . %% TA-100519
foreign(tornes) .
-%% foreign(torp) . %% MW-121119 bus301
+foreign(torp) .
  foreign(torshov) . %% Oslo
  foreign(torvastad) .
  foreign(torød) .
@@ -9964,18 +9955,18 @@
  foreign(vikeland) .
  foreign(vikersund) . foreign(vikersundbakken) .
    foreign(vikersundsbakken) .

-%% foreign(vikhamar) . %% MW-121119 bus301
-%% foreign(vikhamaren) .
-%% foreign(vikhamarn) .
-%% foreign(vikhammar) .
-%% foreign(vikhammaren) .
-%% foreign(vikhammarn) .
-%% foreign(vikhamer) .
-%% foreign(vikhammer) .
-%% foreign(vikhammeren) .
+foreign(vikhamar) .
+foreign(vikhamaren) .
+foreign(vikhamarn) .
+foreign(vikhammar) .
+foreign(vikhammaren) .
+foreign(vikhammarn) .
+foreign(vikhamer) .
+foreign(vikhammer) .
+foreign(vikhammeren) .

-%% foreign(vikhammeråsen) .
-%% foreign(vikhov) . %% MW-121119 bus301
+foreign(vikhammeråsen) .
+foreign(vikhov) .
  foreign(vikvarvet) .
  foreign(vikran) .
  foreign(vilberg) .
```

---

## Appendix B

# Source Code

The source code made under the development is available in the following SVN repositories:

- The RegTopp Automaton and the RegTopp to Prolog Converter  
<http://basar.idi.ntnu.no/svn/busstuc/regtopp/prolog>
- BusTUC  
<http://basar.idi.ntnu.no/svn/busstuc/regtopp>



# Bibliography

- Amble, T. (2000). Bustuc: a natural language bus route oracle. In *Proceedings of the sixth conference on Applied natural language processing*, pages 1–6. Association for Computational Linguistics.
- Amble, T. (2009). Bustuc—a savant level intelligent bus oracle. In *Norwegian Artificial Intelligens Symposium (NAIS)*. Tapir Akademisk Forlag.
- Andersstuen, R. and Engell, T. (2011). Multibris: A Multiple-platform approach to the Ultimate Bus Route Information System for Mobile Devices. Technical report, Department of Computer and Information Science, NTNU.
- Bratseth, J. S. (1997). Bustuc - a natural language bus traffic information system. Master’s thesis, NTNU.
- Kruchten, P. (1995). The 4+ 1 view model of architecture. *Software, IEEE*, 12(6):42–50.
- Trafikanten (1996). *Regtoppformatet versjon 1.2*. [http://labs.trafikanten.no/media/12753/RF\\_1-2-1-1.pdf](http://labs.trafikanten.no/media/12753/RF_1-2-1-1.pdf).