Context-Awareness and Real-Time Information in an Intelligent Smartphone Application

Christoffer Jun Marcussen **Lars Moland Eliassen**
{chrimarc,larsmola}@stud.ntnu.no

Rune Sætre **Björn Gambäck**
{satre,gamback}@idi.ntnu.no

Department of Computer and Information Science
Norwegian University of Science and Technology

Abstract: With the constant increase in smartphone sales, integrated sensors and map navigation have now become available to the average user. This allows for mobile applications to use context-awareness to provide more relevant information. An interesting use-case for such applications is a route information system for buses.

The paper describes an application which interfaces over a mobile phone to BusTUC, a natural language-based reasoning-system for bus routes in Trondheim. By combining user context, BusTUC reasoning and real-time data from the bus service provider, the user-interaction is simplified, compared to a standard information system. We discuss issues on supporting context-awareness and real-time information in this system, comparing it to other available route information systems. Feedback from beta-testers indicates that the application suits the needs of typical bus travellers well.

1 Introduction

Smartphones today are pervasive and personal: they are everywhere, almost always turned on, and customised to each user. Hence smartphones are well suited for context-aware applications [ROPT05], and indeed *context-awareness is at the core of location-aware computing* [HSK]. Location-aware systems utilise information about the user's location, through location-sensing technology such as GPS or Wi-Fi, or by short range transmission technology (e.g., Radio Frequency Identification, RFID). An example of RFID usage in a context-aware application is to place tags in doorways, to track passing people.

Context-awareness can be utilised in several types of applications. For example, the tour guide for the Nidaros Cathedral, implemented in the context-aware *Nidaros Framework* [WSB+05] can use location information to display zones and nearby objects. The domain of route guidance is particularly well suited for context-awareness: the systems can, e.g., monitor the user's behaviour through sensor input, and use this data to provide route suggestions or other information. With the recent progress in smartphone technology, several route guidance applications have become available. Here, we present TABuss, an intelligent application developed to explore new possibilities and to utilise more smartphone

capabilities, within the bus route information domain. To this end, some important design decisions will be discussed, in particular which specific smartphone capabilities are most relevant to the users in this type of setting, how the user interface best should be designed, and whether developing for one particular type of phone using native code is the optimal choice or if cross-platform development through a web application is a better option.

TABuss is based on BusTUC [Amb00], a natural-language query system which became publicly available to the inhabitants of Trondheim in 1998, supporting them in getting information on the time tables of "Trondheim Trafikkselskap", the city's bus company as the time. BusTUC was commercialised by LingIT AS in 2001, and approximately one million queries have been posed to the system every year since then. Now hosted by the current public transportation provider in Trondheim, AtB (`www.atb.no`), BusTUC can be accessed both through the web and a Short Message Service (SMS).

The main topic of the paper is context-awareness as implemented in TABuss. The system also incorporates real-time capabilities by providing information on the actual arrival times of all buses. Queries are sent using a SOAP[1] interface to a server hosted by AtB. The only necessary input parameter is a bus stop's real-time ID which can be retrieved from the same server as a list mapping each bus stop ID to a real-time ID. AtB updates this list from time to time, so an updated list is necessary in order to query the correct real-time data.

The rest of the paper is laid out as follows: Section 2 discusses the concept of context awareness, and its application within a bus route information system. Section 3 describes some other relevant systems. Section 4 moves on to the design decisions taken in the TABuss application, including the tradeoff between native and web-based applications, while Section 5 details the actual TABuss application. Section 6 reports the results of system testing and initial user feedback. Finally, Section 7 further discusses the experiences of the development of TABuss and suggests ways in which the work could be extended.

2 Context Awareness

To describe and implement a context-aware bus route information system, the concepts *context* and *context-awareness* will be used quite specifically: context only uses location information, while context-awareness introduces time and destination as additional factors.

Several researchers have defined context and context-awareness. Pascoe (1998) defined context as "a subset of physical and conceptual states of interest to a particular entity" (e.g., a person), where the importance of the involved states has to be determined [Pas98]. Schilit and Theimer (1994) introduced three factors necessary to define context: location, descriptions of people in the immediate surroundings, and objects (with the changes these objects go through) [ST94]. Ryan *et al.* (1997) added time as a factor, defining context as the user's location, environment, identity, and time. They generalised context by including a number of physical and logical attributes, assumed to affect the user's environment [RPM98]. Dey and Abowd (1999) gave an even more general definition of context.

[1] `http://www.w3.org/TR/soap/`

Rather than enumerating a list of factors needed to be matched, they wrote: "context is any information that can be used to characterise the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and application themselves" [DA99]. This definition simplifies declaring functionalities and theories as context. If any piece of information can describe or help the user at a given time, it can be called context.

Dey and Abowd's definition will be adopted here, partially since it is the most general, stating that a factor or an entity is a part of the context, as long as it concerns the user. It also allows for context to be either implicit or explicit. This means that context information both can be provided by the user and automatically detected by the system. Dey and Abowd further define context-awareness as the system's responses to changed context, without having to determine whether the system should initiate an action automatically or not. For example, the system should not automatically start downloading the real-time data for one or more bus stops, only because the user has changed his/her location. However, in an intelligent application, it is still an advantage to have the option to do so.

Pascoe's definition cannot be used here, because of its lack of modularity. If adding additional factors to determine context, the individual importance of the factors would have to be determined. This is difficult, as situations change (what is rated as an important factor in one setting, might be less important in another). The definition by Schilit and Theimer as well as the one by Ryan *et al.* would also — considering the limited and changing factors — not be adaptable to our purposes.

As mentioned above, the concepts of context and context-awareness are used in specific ways in a bus route information system. Context is used during the general location tracking of the user, which triggers the loading of nearby bus stops. This happens dynamically as the user moves and can be seen on the map while moving. Clickable Bus stop icons are added or removed as the user changes his/her location. This is an example of implicit context: the user does not provide the location information manually (the location is automatically detected and tracked by the system). However, location technology, such as GPS, Wi-Fi or 3G must be enabled on the phone. Context-awareness in TABuss, which uses more factors (e.g., time and day), is described in Section 5.1.

3 Related Work

Several intelligent route information systems include natural language interfaces. The *Let's go* system [RLBE03] uses speech from phone calls as input, and returns route information. The system was developed for "elderly and non-native English speakers", providing information for the city of Pittsburgh. Speech is recognised by comparison and retrieval of the closest match, with the emphasis on creation of a grammar model for spoken language, and on including an overall generality regarding different structuring of sentences with the same meaning. Several challenges with speech processing and route information were identified, with the main challenge being that different users use different phrases, when referring to bus stops or places.

TravelMan was developed for the city of Tampere, Finland [THK+07]. The input consists of locations or addresses, provided to the system as text or speech. The user can also set personal preferences, such as exclusion of specific transportation options, as *TravelMan* covers metro and tram in addition to bus transportation. Guidance functionality for visually impaired users was implemented, to provide what was referred to as "an unbroken trip chain". That is, a successful trip should be complete with full system guidance. An interesting feature in *TravelMan* is the use of context and user location. The real-time guidance relies on location information, which can also be used to infer departure addresses.

Access to real-time bus information has previously been addressed in, e.g., the *MyBus* system [MD], which predicted real-time arrival of buses based on historical data, the bus schedule and a prediction algorithm. The algorithm produced estimates, where traffic and passenger information was used as noise, affecting the original schedule. A mobile version of the system, *Mobile MyBus* [MD01] used WAP communication. *MyBus* gave users real-time information based on input consisting of destination and a route number. Both inputs were provided as digits, since devices at the time did not have input mechanisms similar to a computer keyboard. The functionality can be compared to retrieving a real-time ID for a bus stop in TABuss, before sending a SOAP-request to fetch the real-time data. Today, buses have GPS-trackers on-board, where location information is continuously transmitted to a server. Such real-time data is updated at specific time intervals and is fairly accurate. The real-time service was introduced in Trondheim by AtB early in 2011.

OneBusAway focuses on context awareness in addition to the use of real-time data. *OneBusAway* uses the location of the user to automatically display the closest bus stops on a map, similar to TABuss. Context aware functionality has proven to be useful through user testing, where 93% of the existing users of *OneBusAway* reported that more concise information was provided [FWB10]. What distinguishes TABuss from *OneBusAway* is that the main query functionality in TABuss only needs the user's destination as input. *OneBusAway* needs to know both the departure stop and which route to select in order to reach the planned destination.

There are also several smartphone applications developed for bus transportation in Trondheim, many of which use the BusTUC oracle service. All these applications have been downloaded by several users, indicating that they have attractive functionality. For our project, it was important to investigate what has to be done to move the concept of a bus route information application to a higher level. We also compare the different levels of artificial intelligence in the given apps. Two of the existing solutions are close to what we want to make: *Alf's ByBuss* and *Bartebuss*. They both use BusTUC, maps and the real-time functionality. *Bartebuss*[2] is developed in HTML5 and uses the *BusBuddy*[3] API(Application Programming Interface). *Bartebuss* has the option to store favourites, it can find near-by bus stops or search for specific bus stops, and it can use BusTUC and show maps on the phone. The use of HTML5 makes the map less responsive than in *native applications*. The user interface, on the other hand, is intuitive and easy to navigate, but it might provide too many choices to the user. *Alf's ByBuss*[4] is a native Android application

[2] `http://bartebuss.no/om`
[3] `http://busbuddy.norrs.no/`
[4] `http://bybuss.alfsimen.com`

that also uses the *BusBuddy* API. *Alf's ByBuss* appears more responsive than *Bartebuss* during map navigation, but the user interface is not as polished.

4 Developing Native Applications vs. Web Applications

When developing for mobile platforms, there are several technology decisions to be made. An important choice is whether to develop *native applications* or *web applications*. Native development has been the main choice for platforms such as Android and iOS, as earlier versions of HTML did not provide enough framework possibilities. Lately a new option has emerged, with the release of HTML5. Applications written in HTML5 and JavaScript have now become the new competitors to the *native applications*.

The basic definition of a *web application* is: "having no ties to a specific operating system or device". *Web applications* do not rely on any platform-specific API or SDK(Software Development Kit). The user interface can be designed to resemble *native applications*, but the *web applications* can be deployed on all major platforms. *Web applications* can consist of web code only, or they can be hybrids with both HTML and native code. A *hybrid application* runs the web code in addition to some parts implemented in native code. The native parts can be just simple parts of the user interface, or large amount of back-end code. A user interface written for the web is different from a native one in that the rendering will be done in a web browser instead of by a native graphic component on a specific device.

Native applications are developed using platform-specific SDKs and have (through the underlying operating system) direct access to the device's hardware. The main disadvantage with native development is that the applications are not directly portable to other platforms.

Although *web applications* can perform many of the same functions as *native applications*, the end result is not always as satisfying. *Web applications* (as of today) perform slower than *native applications*, and large background computing is not supported by build systems such as PhoneGap[5] which maps web code to native code. Still, it is debatable how many applications actually need the most optimised performance to function properly.

Hardware access has been a problem for early *web applications*, as the possibilities were limited compared to native code. Today, with libraries that give access to components such as GPS, camera and compass, the limitations are less visible. However, some are still noticeable, e.g., when using maps in iOS. Then, the rendering speed is similar to native code, while on Android the performance is slower.

Both *web applications* and *native applications* have advantages and disadvantages, so the choice depends on the complete context. TABuss is developed as a *native application* (for Android), since *native applications* are ideal for research within mobile development: The end goal for research is seldom mass distribution, but proof-of-concept. This can be realised by having the newest (native) tools available.

[5] `http://www.phonegap.com`

5 The TABuss Implementation

The development goal of TABuss was to make the application as easy to use as possible. TABuss is divided into several *Android activities*, where the top activity defines the main(home) screen. The phone's menu and buttons start *sub-activities*, and the user can choose for whether or not to use the map.

5.1 Context Awareness in TABuss

Context is mainly extracted from the user's location. The application automatically loads the closest bus stops based on the user's location whenever a location change has been detected. Real-time data for these bus stops can be accessed from the map or through a list available in the menu. The closest bus stops also play an important part in the main query functionality, where the user's location determines which bus stops are included as departure stops. TABuss distinguishes itself from other existing solutions by giving the user the option to let the application guess where he/she is going. A simplified version of case-based reasoning is implemented, by logging each query as a case (see [AP94]). The queries are stored locally, in a database, and each case consists of the departing area, the time of day, the day of week and the destination. The departure area is a 500×500 metre square, with a defined area code stored in a separate table. Whenever a new case is created, a new area is created if the origin location is not covered by an existing area.

Queries with similar origin and time are fetched from the database in order to retrieve relevant cases. Similarity is indicated by an overlapping area or by closeness in the time of day for the query (+/- 2 hours). The retrieved cases are rated by the Euclidean distance between the locations, the time-difference and whether they happened on the same day of the week. The best matching destination is presented to the user, based on loose time matching, since exact matches are rare. The stored cases could be extended, for example, to include delayed bus departures, and bus departures from within a time period.

When TABuss suggests a route, the user can respond by validating the result. Positive user feedback currently triggers a query run, while negative feedback has no effect. The level of intelligence is fairly low, but still higher than in approaches based on direct look-ups.

5.2 Language Processing

TABuss has an option to switch between two different natural language modes: the "new" mode which assumes that the user wants to depart from one of the closest located bus stops, and the "standard" mode which allows for user-defined departure stops (and other complete natural language queries about buses), as examplified below, showing queries in the "standard" (1) and "new" (2) modes (where the n represents walking distance, in this case to the bus stop 'Samfundet'). Switching between the two language modes can be done in the home screen menu.

(1) *Når går bussen fra Samfundet til Torvtaket?*
When goes the-bus from Samfundet to Torvtaket?
When does the bus to Torvtaket depart from Samfundet?

(2) *(Samfundet +n, Prinsen +n) til Torvtaket.*
(Samfundet +n, Prinsen +n) to Torvtaket.

TABuss relies on an existing text messaging service when no data network is available.[6] An SMS (text message) query starts with "*rute*" (route), followed by a natural language query. This has been incorporated in two ways. If the "new" mode is chosen from the home screen menu, TABuss uses the closest bus stop to the user's location as the departure stop. If the "standard" mode is chosen, the user has to provide a complete sentence including the place of departure and the destination.

5.3 Real-time Functionality

Real-time data can be accessed from the map by pressing a bus stop icon or through the home screen menu. Both access methods utilise the user's location to retrieve and display the n closest bus stops.

The retrieval of a bus stop's ID is done by comparing the chosen bus stop's location with the locations of each of the n closest bus stops. If matched, the found bus stop ID is used to extract the real-time ID. The real-time ID is then sent via SOAP to the real-time server, which returns the five next bus departures. The user can also search for bus stops that are not among the n closest, by providing a bus stop name as input. This option also lets the user select which direction to retrieve real-time data for (either from or towards the city centre) before the real-time query is sent.

5.4 Answer Display

The TABuss application is best explained through some screenshots. Figure 1a shows the start menu, where text buttons represent shortcuts stored on the SD-card of the device. Figure 1b is the answer screen with route suggestions shown in a list view. The displayed routes with updated departure times are the results of an HTTP query sent to BusTUC. Walking distances to the bus stops are shown within parentheses. "**Overgang**" indicates that the suggested route includes a transfer.

The context-awareness of the TABuss application is shown in a second set of screenshots. Figure 2a shows a map displaying the user's location and the closest bus stops represented by clickable bus stop icons. The map is displayed by selecting an element in the results list. Finally, Figure 2b displays the result of a real-time data query for a specific bus stop. The query is either initiated from the menu or by pressing a bus stop icon on the map.

[6] `https://www.atb.no/spoer-bussorakelet/category228.html`

(a) Start screen

(b) Answer screen

Figure 1: Screenshots from TABuss

(a) Map showing the closest bus stops and the user's current location

(b) Real-time bus arrival information for a specific bus stop

Figure 2: Context-awareness in TABuss

6 Evaluation

A small-scale user test was performed to get feedback on the TABuss' application. An extensive user test was not conducted because of time limitations and delays in the public release of the application. Hence all the test subjects were Trondheim inhabitants and experienced bus travellers.

The general user opinion indicates that the application is easy to use. It is clear that the users appreciate the user interface. Positive feedback has been received on both the colour combinations and the layout. Most users prefer the application functionalities detached from the map, and feedback suggest that the map should only be optional. Users find the query functionality useful. The main functionality with the "new" BusTUC mode is seen as interesting. Users requested the possibility to use the standard BusTUC mode for queries not involving the closest located bus stops, something which was not an option in the early development stage. Another suggestion was to include a "settings"-screen, allowing the user to set preferences (such as the number of bus stops to use in queries). The real-time data functionality for the closest bus stops was easy to access and use. This applies to the real-time functionality on the home screen, as this required less navigation than through the map.

It is difficult to draw concise conclusions after this small user test, but the feedback received from the target users is valuable. The suggestions and error reports give an indication that TABuss suits the needs of bus travellers.

7 Discussion and Future Work

In Section 3, the applications *Bartebuss* and *Alf's ByBuss* were compared to TABuss. TABuss' functionalities are more focused on user location and context-awareness than *Bartebuss*. The level of "intelligence" is what separates TABuss from *Bartebuss*, and also from other apps tested in Trondheim. In order for a bus route information application to "be intelligent", the natural input source has to be context data. Still, whether TABuss can be classified as "better" than *Bartebuss* is an open question: *Bartebuss* has been developed over a longer period of time, and been through more extensive user testing. However, TABuss represents a more complete approach, with a good market potential, since no other application has the exact same functionalities.

Section 4 discussed the advantages and disadvantages of native vs. web development, and stated that native development was the preferred choice for a research prototype. In retrospect, we are satisfied with the choice of technology. Compared to *Bartebuss*, a notable technology difference lies in the storage functionalities. For Bartebuss to work cross-platform, and also through a regular browser, "web storage" through "local storage" is used. The size limit of local storage depends on which browser is used, but it cannot be larger than 10 megabytes (Internet Explorer). TABuss uses the devices' external storage, where the size limit depends on the size of the mounted SD-card, which can normally store gigabytes of information. It is possible that future releases of TABuss will need more stor-

age space than 10 megabytes. The storage limitation of also affects the iOS version of Bartebuss, where the internal storage optimally is used instead (since no external storage is available).

The problems with lagging maps in web applications deployed on the Android platform are avoided in TABuss, where the map is much more responsive. Native development also allows for pinch zooming, which is an important feature when navigating maps.

Although web applications can be deployed on multiple platforms, native applications provide the best user experience for Android and the bus route domain. It is preferable to develop a competitive application for a specific platform, rather than to deploy a "working" solution to multiple platforms (given today's web application performance on Android). Future SDK updates will benefit web application development and improve the browser rendering. One problem is that older devices will not receive these updates (and it will take time for newer devices to get them). The release to newer devices usually happens only after the different manufacturers have adapted their own distributions. Developers will then have a dilemma regarding which SDK versions to target (or which users to exclude).

TABuss uses location data as context input. An extension is to use more sensors than only the location sensor, such as in *ContextPhone* [ROPT05]. *ContextPhone* uses four sensors: location, user interaction, communication behaviour and physical environment. This means that besides from location information, *ContextPhone* monitors what actions the user performs, calls and SMSs, and surrounding devices. Such sensor information could be used to introduce context awareness to the TABuss user interface. The user interface could track the user's actions through sensors, register trends and then adjust visibility and availability accordingly. The tracking of the user's trends could also be used to create better route suggestions. People of different ages have different levels of mobility, and different walking speeds. This has been addressed in *UbiBus* which considers different people's and vehicle's mobility, and other factors that can affect which bus departures that are most "attractive" [VCS11]. An interesting idea is for AtB to contribute to such functionalities in order to improve route suggestions. Buses have cameras installed, which, for example, could be used to monitor how crowded a bus is.

Acknowledgments

Thanks to LingIT AS and AtB AS for allowing the use of the BusTUC and the real-time bus information services, respectively, and to Magnus Raaum for providing the source code of the original Android application. We would also like to thank our beta-testers: Jirka Konietzny, Trond Bøe Engell, Jostein Klakegg, Ola Hast, Marita Gjerde, Håvard Axelsen and Morten Fornes. Runar Andersstuen and Trond Bøe Engell provided valuable ideas and feedback during the project. Special thanks to the late Tore Amble for developing the BusTUC system and for initiating this work.

The work of the third author was partly funded by the Research Council of Norway, through the project UbiCompForAll (`www.sintef.no/Projectweb/UbiCompForAll`) under contract nr. 187865/S10.

References

[Amb00] Tore Amble. BusTUC: A Natural Language Bus Route Oracle. In *6th Conference on Applied Natural Language Processing*, pages 1–6, Seattle, Washington, 2000. ACL.

[AP94] Agnar Aamodt and Enric Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(1):39–59, 1994.

[DA99] Anind K. Dey and Gregory D. Abowd. Towards a better understanding of context and context-awareness. In *1st International Symposium on Handheld and Ubiquitous Computing*, 1999.

[FWB10] Brian Ferris, Kari Watkins, and Alan Borning. OneBusAway: Location-Aware Tools for Improving Public Transit Usability. *IEEE Pervasive Computing*, 9(1):13–19, 2010.

[HSK] Mike Hazas, James Scott, and John Krumm. Location-Aware Computing Comes of Age. *IEEE Computer*, 37(2).

[MD] Stuart D. Maclean and Daniel J. Dailey. In *7th Annual World Congress on Intelligent Transport Systems*.

[MD01] Stuart D. Maclean and Daniel J. Dailey. Real-time Bus Information on Mobile Devices. In *Intelligent Transportation Systems. Proceedings*, pages 988–993, 2001.

[Pas98] Jason Pascoe. Adding Generic Contextual Capabilities to Wearable Computers. In *2nd IEEE International Symposium on Wearable Computers*, 1998.

[RLBE03] Antoine Raux, Brian Langner, Alan W. Black, and Maxine Eskenazi. LET'S GO: Improving Spoken Dialog Systems for the Elderly and Non-natives. In *8th European Conference on Speech Communication and Technology*, 2003.

[ROPT05] Mika Raento, Antti Oulasvirta, Renaud Petit, and Hannu Toivonen. ContextPhone: A Prototyping Platform for Context-Aware Mobile Applications. *IEEE Pervasive Computing*, 4:51–59, 2005.

[RPM98] Nick S. Ryan, Jason Pascoe, and David R. Morse. Enhanced Reality Fieldwork: the Context-aware Archaeological Assistant. In V. Gaffney, M. van Leusen, and S. Exxon, editors, *Computer Applications in Archaeology 1997*. 1998.

[ST94] Bill N. Schilit and Marvin M. Theimer. Disseminating Active Map Information to Mobile Hosts. *IEEE Network*, 1994.

[THK+07] Markku Turunen, Jaakko Hakulinen, Anssi Kainulainen, Aleksi Melto, and Topi Hurtig. Design of a Rich Multimodal Interface for Mobile Spoken Route Guidance. In *10th European Conference on Speech Communication and Technology*, pages 2193–2196, Antwerp, Belgium, 2007. ISCA.

[VCS11] Vaninha Vieira, Luiz Rodrigo Caldas, and Ana Carolina Salgado. Towards an Ubiquitous and Context Sensitive Public Transportation System. In *4th International Conference on Ubi-Media Computing*, pages 174–179, 2011.

[WSB+05] Alf Inge Wang, Carl-Fredrik Sørensen, Steinar Brede, Hege Servold, and Sigurd Gimre. Development of Location-Aware Applications: The Nidaros framework. In *2nd International Working Conference on Mobile Information Systems*, 2005.