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

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 the user’s context to provide more relevant information. An interesting use-case for such applications is a route information systems for buses.

The paper describes an application which interfaces over a mobile phone to BusTUC, a reasoning-system for bus routes in Trondheim. By combining user context with BusTUC reasoning and real-time data from the bus route company, the user-interaction is simplified, compared to a standard information system. We discuss issues on supporting context-awareness and real-time information in the system, as well as the importance of the design and functionality of the user interface, comparing the system to other available route information systems. Feedback from beta-testers indicate that the application suits the needs of typical bus travellers.

1 Introduction

Smartphones today are pervasive and personal. This means that they are almost always turned on, and they are customised to each user. Hence smartphones are well suited for context-aware applications (see, e.g., [ROPT05]), and indeed context awareness is at the core of location-aware computing [HSK04]. Location-aware systems use the user's location in functionalities, through a location-sensing technology such as GPS or WiFi, or by short transmission range technology (e.g., Radio frequency identification, RFID). An example of RFID use in a context-aware application is to place tags in door entries, and track passing people.

Context-awareness can be utilised in several types of applications, for example in a tour guide for the Nidaros Cathedral which uses the user’s location to display zones and nearby objects implemented in the context-aware Nidaros Framework [WrB+05]. The domain of bus route information systems is particularly well suited for context-awareness. Such systems could be able to 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 bus route information applications have become available. Here, we present TABuss, an intelligent application developed to explore new possibilities within the bus route information domain and to utilise more smartphone capabilities.

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 the bus company Trondheim Trafikkelskap. Since its
commercialisation by LingIT AS in 2001, approximately one million queries have been posed to the BusTUC system every year. Now hosted by AtB on their page www.atb.no, BusTUC can be accessed through both the web and Short Message Service (SMS).

The paper describes context-awareness as implemented in TABuss. The system also incorporates real-time capabilities by providing information on the actual arrival times of all the buses. Queries and answers are sent and received using a SOAP\(^1\) interface to a server hosted by AtB. The only necessary input parameter is a bus stop’s real-time ID. The real-time IDs can be retrieved from the same server as a list that maps each bus stop ID to a real-time ID. AtB updates this list from time to time, and an updated list is necessary in order to query the correct real-time data.

The rest of the paper is laid out as follows: in Section 2 we discuss the concept of context awareness, with a particular view to its application within a bus route information system. Section 3 describes some other relevant systems in the field. Section 4 then moves on to the design decisions taken in the TABuss application, mainly issues related to whether a smartphone application should be developed native or web-based, while Section 5 details the actual TABuss application itself. Section 6 indicates the results of system and user testing. The experiences of the development and testing are topicalised in Section 7. Finally, Section 8 points to directions in which the present work could be extended.

\section{Context Awareness}

To describe and implement a context-aware bus route information system, we will use context and context-awareness in two different ways. The former context uses the user’s location, while the latter context introduces time and destination as additional factors.

Several researchers have attempted to define context and context-awareness. Pascoe (1998) defined context as “a subset of physical and conceptual states of interest to a particular entity”, where the importance of the involved states had to be determined [Pas98]. Schilit and Theimer (1994) introduced three factors to define context: location, descriptions of people in the immediate surroundings and objects with the changes these objects go through [ST94]. Ryan \textit{et al.} (1997) added time as a factor, defining context as the user’s location, environment, identity and time. They generalise the concept of context by including a number of physical and logical attributes, assumed to affect the user’s environment [RPM97]. However, Dey (1999) gave a more general formulation of context, rather than enumerating a list of factors needed to be matched: “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 formulation simplifies declaring functionalities and theories as context. If any part of information can describe or help the user at a given time, it can be called context.

Dey’s definition will be adopted here, partially since it is the most general, stating that a

\url{http://www.w3.org/TR/soap/}
factor 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 is both provided by the user and automatically detected by the system. Dey also defines “context-awareness” as the system’s response to changed context. It does not determine whether the system should initiate an action automatically or not. In a practical example from TABuss: the system should not automatically start downloading the real-time data for one or more stops, only because the user has changed his/her location. But in an intelligent application, it is still an advantage to have the option to do so.

Pascoe’s definition cannot be used, because of the lack of modularity. If adding additional factors to determine context, their importance 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 definitions by Schilit and Theimer’s and Ryan et al. would considering the limited and changing factors not be adaptable to our purpose.

As mentioned earlier, context and context awareness are used in two different ways in a bus route information system. In the first, context is used during the general location tracking of the user, and subsequent 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 (location is automatically detected and tracked by the system). However, a location technology such as GPS, WiFi or 3G must be enabled. The second way, 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. So does the Let’s go system [RLBE03] use speech through phone calls as input for returning 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 structured the same phrases differently, when referring to bus stops or places.

A similar solution, TravelMan was developed for the city of Tampere, Finland [THK et al. 07]. Input consists of locations or addresses, provided to the system by text or speech. The user can also set personal preferences, such as exclusion of transportation options, as Travel-Man in addition to bus transportation, covers metro and tram. Of other features, a guidance functionality for visually impaired users was implemented, to provide what was referred to as an “unbroken trip chain”, i.e., a successful trip, completed with full system guidance. An interesting feature in TravelMan related to TABuss, is the use of context and user location. The real-time guidance relies on location information, which also can be used to
infer departure addresses.

Access to real-time bus information has also been addressed previously, e.g., in MyBus [MD00] which predicts real-time arrival of buses by using historical data, the bus schedule and an prediction algorithm. This algorithm, based on Kalman filters [KB60], produced estimates, where traffic and passenger information is used as noise, affecting the original schedule. A mobile version of the system, Mobile MyBus [MD01] used communication through WAP. This system gave users real-time information based on received input. The input consisted of destination and a route number, sent by a URL request, and forwarded to the MyBus server. Both were provided as digits as devices at that time did not have any input mechanism similar to a computer keyboard. In TABuss, this functionality can be compared to retrieving a real-time ID for a bus stop, before sending a SOAP-request to fetch real-time data. Today, buses have GPS-trackers on-board, transmitting location information to a server. Real-time data is updated at specific time intervals and is fairly accurate. The real-time service was introduced in Trondheim by AtB.

OneBusAway [FWB10] 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 proved to be successful through user testing, where 93% of the existing users of OneBusAway reported that more concise information was provided. What distinguishes TABuss from OneBusAway is that the main query functionality in TABuss only needs the users destination as input. OneBusAway needs to know both the departure stop and which route to select in order to reach planned destination.

There are also several smartphone applications developed for getting information about bus transportation in Trondheim, many of which already use the BusTUC oracle service. All these applications for bus route information in Trondheim have been downloaded by several users, indicating that they have attractive functionalities. 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 new level, and also to compare the different levels of artificial intelligence in the apps. Two of the existing solutions are close to our goals: Alf’s ByBuss and Bartebuss. They both use BusTUC, maps and real-time functionality. Bartebuss² is developed in HTML5 and uses the BusBuddy API³. It has rich functionality, with options to store favourites, find near-by bus stops, search for specific bus stops, use BusTUC and show maps. Due to the use of HTML5, the map is not as responsive as in a native application. 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 ByBuss⁴ is a native Android application also using the BusBuddy API. Alf’s ByBuss appears more responsive than Bartebuss during map navigation, but the user interface is not as polished.

²http://bartebuss.no/om
³http://busbuddy.norr.no/
⁴http://bybuss.alfsimen.com
4 Developing Native Applications vs Web Applications

When developing for mobile platforms, there are several technology decisions to be made. An important one is regarding the choice to develop native applications or web applications. Native development has in later years 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 become the new competitors to 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. They can be designed to resemble native applications regarding the user interface, and they can be deployed on all major platforms. Web applications can consist of web code only, or they can be hybrids between web applications and native code. A hybrid application runs the web code in addition to some parts implemented in native code. This native part can span from simple parts of the user interface to larger amount of back-end code. What separates a user interface written for web from a native one is that the rendering is done by a browser. In native applications, the rendering is done by native graphic components in the device.

Native applications are developed using platform-specific SDKs. Native applications have (through the underlying operating system) direct access to a device’s hardware. The main disadvantage with native development lies in the meaning of the term: as applications are native, they are not 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 apps, and large background computing is not supported by build systems (such as PhoneGap which is a build API mapping JavaScript calls within web applications to the correct functionalities found in a native SDK). Still, it is debatable how many applications that actually need performance at a top level to function properly.

Hardware access has earlier been a problem for web applications, as the possibilities were limited compared to native. Today, provided libraries give access to components such as GPS, camera and compass, and limitations are less visible. However, some are noticeable, e.g., the usage of maps: for iOS, rendering results are similar to native, while on Android, performance is slower.

Overall, both development technologies have advantages and disadvantages; which one to use depends on the complete context. TABuss is developed as a native applications (for Android) since we see native applications as 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 best (native) tools available.

\(^{5}\)http://www.phonegap.com
5 The TABuss Implementation

The development of TABuss focused on making usage of the application as easy as possible. TABuss is divided into several activities, where the top activity defines a home screen. Menu and button presses start other activities, and the users can choose for themselves whether or not to use the map. By encouraging thumb-navigation, the map is reduced to an extra feature.

5.1 Context Awareness

Context is mainly extracted from the user’s location. The device’s location listener automatically loads the bus stop objects (from the bus stop list) closest to the user’s location, when a location change has been triggered. Real-time data for these 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. To distinguish TABuss from the many existing solutions, extra functionality is added by giving the user the option to let the application guess where he/she is going. A simplified version of case-based reasoning [AP94] is implemented, by logging each made query as a case. These are stored locally in a database, where each case consists of the departing area, time of day, day of week and destination. Departure areas are squares of $500 \times 500$ metres, with defined area codes stored in a separate table. Whenever a new log item is created, a new area is created if the origin location is not covered by an existing area.

To retrieve relevant cases, queries with similar origin and time are fetched from the database. Similarity is implied by identical areas and somewhat similar time of day. For now, $+/- 2$ hours is used. The retrieved cases are rated by the euclidean distance between each case, and the current time and weekday. The best matching destination is then presented to the user. $+/-$ for hours is used as finding a direct match is difficult. We want to also 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. At the current moment positive user feedback triggers a query run, while negative feedback has no effect. The level of intelligence is fairly low, but is still higher than in functionalities with direct look-ups.

5.2 Language Processing

To enhance the use of natural language in TABuss, new functionalities using BusTUC were implemented, including the option to switch between two different BusTUC syntaxes, the new which assumes that the user wants to depart from one of the closest located bus stops, and the standard which allows for user-defined departure stops. Switching between the two BusTUC syntaxes can be done in the home screen menu.
The second part involves AtB’s text messaging service\textsuperscript{6}. An SMS query starts with “route” (route), followed by text, to 2027. This has been incorporated in two ways. If the first BusTUC syntax is chosen from the home screen menu, TABuss uses the closest bus stop to the user’s location as the departure stop. If the second syntax is chosen, the user has to provide both departure and destination input.

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 functionalities use 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 (from/towards the city centre), before a real-time query is sent.

\textsuperscript{6}https://www.atb.no/spoer-bussorakelet/category228.html
6 Evaluation

The resulting TABuss application is shown in the screenshots of Figure 1. In the display of query answers, route suggestions are shown in a list view, where touch events on a list element trigger a map activity. This map activity shows the user’s location, the location of the selected departure bus stop, and a walking route between the two locations.

6.1 User Testing

A simplified user test was performed to get feedback of TABuss’ functionalities. An extensive user test was not conducted because of time limitations, and issues with permission for public release of the application. All of the test subjects were Trondheim inhabitants, and experienced bus travellers.

The general user opinion indicated that the application was easy to use. It was clear that the users appreciated our prioritising of user interface. Positive feedback was received on both the colour combinations and the layout. Most users preferred the application functionalities detached from the map, and feedback suggested the map should only be an add-on. Users found the query functionality to be useful. The main functionality with the new BusTUC syntax was seen as interesting. Users requested the possibility to use the standard BusTUC syntax for queries not involving the closest located bus stops. This was not an option at the development stage. Another suggestion was a “settings”-screen, allowing the user to set properties such as the number of bus stops to use in queries. The real-time data functionality for the closest bus stops was a functionality found quick to access and use. This especially applied to the real-time functionality accessible from the home screen, as this required less navigation than through the map.

It is difficult to draw a concise conclusion after a narrow user test, but the feedback we received from the target users was valuable. We received implementation suggestions, information on errors and an indication that TABuss suited the needs of bus travellers.

7 Discussion

It was concluded during research of existing solutions in Section 3, that Bartebuss and Alf’s ByBuss were comparable applications to TABuss. Compared to Bartebuss, TABuss’ functionalities are more focused on user location and context awareness. In our opinion, intelligence is what separates TABuss from Bartebuss, and also from the other test subjects for Trondheim. We claim that in order for an intelligent bus route information application to actually “be intelligent”, the natural input source is context data. Whether TABuss can be classified as “better” is unsure, as Bartebuss has been developed over a longer period of time, and been through more extensive user testing. Still, we feel TABuss represents a more complex approach, with market potential, as no other applications have the exact same functionalities.
We discussed in Section 4 the advantages and disadvantages with native and web development, and stated that both of the project members prefer native. In retrospect we are satisfied with our technology choice. Compared to Bartebuss, a technology difference is 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 is no bigger than 10 MB (Internet Explorer). TABuss uses the devices’ external storage, where the size limit depends on the size of the mounted SD-card, which normally can store gigabytes of information. While not necessary in the current version of TABuss, future extensions could need more storage space than 10 MB. The storage limitation of local storage also affects the iOS version of Bartebuss, where the internal storage optimally is used instead (external storage not available).

The map problems 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.

Although web applications can be deployed on multiple platforms, native applications provide, in our opinion, the best user experience for Android and our domain. We prefer to rather develop a competitive application for a specific platform, than to deploy a “working” solution to more. It has to mentioned that this is given today’s web application performance on Android. Future SDK updates will benefit web application development and improve the browser rendering. The problem is that older devices will not receive these updates, and it will also take time for newer devices to get them. The releases to newer devices almost always have to wait until the different manufacturers have adapted their own distributions. Developers will then have a dilemma on which SDK versions to target, and which users to exclude.

8 Future work

For context-awareness, 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. This sensor information could be used to introduce context awareness to the TABuss user interface. The user interface could through sensors track the user’s actions, register some trends and then adjust visibility and availability accordingly. The tracking of user trends could also be used to perfect route suggestions. People of different ages have different levels of mobility, and have different walking speeds. This has been addressed in UbiBus [VCS11] which considers different people’s and vehicle’s mobility, and other factors than can affect a bus departure. An interesting idea is for AtB to contribute to such functionalities in order to improve route suggestions. Buses have installed cameras, and could be used to monitor how crowded a bus is.
References


