Lappeenranta University of Technology
School of Engineering Science
Software Engineering

Bachelor's thesis

Janne Föhr

CONTEXT-AWARENESS THROUGH GOOGLE AWARENESS API

Examiner: Professor Ajantha Dahanayake

Supervisor: Professor Ajantha Dahanayake

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
School of Engineering Science
Tietotekniikan koulutusohjelma

Janne Föhr

Kontekstitietoisuus käyttäen Google Awareness API rajapintaa

Kandidaatintyö

2019

37 sivua

Työn tarkastaja: Professori Ajantha Dahanayake

Hakusanat: kontekstitietoisuus, Google Awareness API
Keywords: context-awareness, Google Awareness API

Tämä kandidaatintyö tutkii kontekstia ja kontekstitietoisuutta, eri tapoja kerätä kontekstuaalista tietoa sekä kontekstitietoisia sovelluksia ja kontekstitietoisuuden käyttötapoja. Näitä aiheita tutkitaan kirjallisuuskatsauksen kautta, mikä kattaa 30 artikkelia. Google Awareness API rajapintaa tarkastellaan tutkimalla Googlen tarjoamia ohjeita ja dokumentaatiota. Kirjallisuuden perusteella havaittiin, että Google Awareness API rajapintaa voidaan käyttää kontekstitietoisten sovellusten kehittämiseen, jotka käyttävät hyväkseen tietoa käyttäjän sen hetkisestä sijainnista, semanttisesta paikasta, beacon tietoja, käyttäjän sen hetkisestä aktiviteetistä, semanttisesta ajasta, säätiedoista ja kuulokkeiden tilasta. Rajoittavina tekijöinä rajapinnassa ovat yksityisyysongelmat, jotka johtuvat siitä, että kehittäjä saa pääsyn käyttäjän sijainti- ja aktiviteettihistoriaan. Toisena rajoittavana tekijänä nähdään rajapinnan keskittyminen vain älypuhelimiin, eikä muita kontekstitiedon lähteitä tueta kuten esimerkiksi älykelloja.

ABSTRACT

Lappeenranta University of Technology
School of Engineering Science
Software Engineering

Janne Föhr

Context-awareness through Google Awareness API

Bachelor's Thesis

37 pages

Examiner: Professor Ajantha Dahanayake

Keywords: context-awareness, Google Awareness API

This bachelor's thesis studied the context and context-awareness, ways to gather the contextual information and context-aware applications, and use cases. These topics were examined via literature review covering 30 articles. The Google Awareness API was examined by studying the guides and documentation provided by the Google. It was found that the Google Awareness API could be used to develop context-aware applications, which required information on the user's current location, nearby places, beacon information, user's activity information, semantic time intervals, weather information, and headphone state. Limitations of the API were the privacy issues, which came from the ability of the developer to collect sensitive data about the user's location and activity history. In addition, the API only supported smartphones and no other devices e.g., smartwatches as the contextual information sources.

ACKNOWLEDGEMENTS

I would like to thank professor Ajantha Dahanayake for patience waiting for this bachelor thesis to get finished. I would also like to thank Evgen Multia for proof reading the thesis and for the support. In addition, thanks to friends and family for constantly asking if the thesis is already done, and for making sure that it gets done.

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

ACM	Android Context Management
API	Application Programming Interface
AI	Artificial Intelligence
AR	Augmented Reality
CAMA	Context Aware Mobile Application
CBT	Cognitive Behavioral Therapy
GPS	Global Positioning System
GSM	Global System for Mobile communications
HTTP	Hypertext Transfer Protocol
IE	Intelligent Environments
NFC	Near-Field Communication
MOSS	Mobile Sensing and Support
P2P	Peer-to-Peer
SDK	Software Development Kit
SMS	Short Message Service
UI	User Interface
WAID	What Am I Doing

1 INTRODUCTION

This bachelor's thesis focuses on the context and context-awareness, how the contextual information can be gathered and used. The thesis will also elaborate what kind of applications and use cases are presented in the recent studies on the topic, and what does the Google Awareness Application Programming Interface (API) offer. A literature review is performed to get an idea about the topic after which the Google Awareness API and its capabilities are studied.

1.1 Goals and delimitations

The goal of the work is to get understanding of the context and context-awareness overall and how it is used currently. This is done by performing a literature review on the topic. Using the literature review base the Google Awareness API is studied on the basis what it can offer for context-aware application development.

The literature review is restricted to articles published on 2016 or later, with the exception when studying the context and context-awareness where also earlier articles were studied. The Google Awareness API is only studied by examining the guides and documentation provided by the Google. No prototype application was developed as part of the thesis.

1.2 Structure of the thesis

Section 2 contains the literature review done on the topics of context and context-awareness, solutions to gather context information and providing context-awareness, different kinds of context-aware applications and use cases, and finally the usage of Google Awareness API. Section 3 contains the details of the Google Awareness API, its usage possibilities and limitations. Section 4 contains the concluding remarks and section 5 summarizes the thesis.

2 LITERATURE REVIEW

This chapter studied the current view on context and context-awareness, different solutions to gather context information and to provide context-awareness, and the different kinds of applications and use cases of context-awareness. In addition, the current usage of the Google Awareness API was studied. The aim of this literature review was to get understanding of the current trends in context-awareness with a focus on the use in smartphones.

The literature review focused on studies published on 2016 and later with the exception when defining context and context-awareness where earlier studies were also used. Total number of studies studied for the literature review was 30.

2.1 Context and context-awareness

In this subsection, the previous work on definition of context and context-awareness were studied.

Most of the studies read for this literature review use the Dey's and Abowd's notions of the context and context-awareness. Dey & Abowd (1999) defined that context-aware applications should look at who is, where is, when is and what is, and use this information to determine why the situation is happening and what is happening. They listed contexts that were most important: location, identity, activity, and time. The activity meant knowledge about what was happening in the situation. This list added to the list by Schilit, Adams & Want (1994): where you are, whom you are with, and what objects are around you. Information about activity and time were also important to use when trying to characterize a situation.

Dey & Abowd (1999) presented a categorization of context-aware features that context-aware applications may support. The three categories were:

1. presentation of information and services to a user
2. automatic execution of a service

3. tagging of context to information for later retrieval (Dey & Abowd 1999, p. 8).

Dey wrote, “Context is any information that can be used to characterize 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 applications themselves” (Dey 2001, p. 5). This categorization was thought to help application developers to see what kind of contexts they should be thinking about while developing context-aware application. Context-awareness was defined by him as follows: “A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user’s task” (Dey 2001, p. 5). He made this generalization because he found that the older definitions were too strict to be usable in general context-aware application development and research.

A survey done by Augusto et al. (2017) found that the context has been discussed from the 1900s in the philosophical, linguistic, and logic studies. They point out that the notion of context has been studied since the early stages of computing, starting from the studies of formal languages. In their view, the context-awareness had two different kinds of viewpoints. The one that focused on the artificial intelligence (AI) and the other that focused on intelligent environments (IE). They stated that the AI started to study the field of context-awareness, followed later by the IE. The AI focused on how different kinds of contexts could be noticed and utilized so that when the context changed, the system should be able to act according to the correct context. While the IE focused on the service orientation of systems, and how to utilize context-awareness in those. Augusto et al. (2017) pointed out that the more the system had information about the user and the context that the user is in, the more effective the system could be. The AI focused more on the ways to represent concepts using philosophy, logic, linguistics, psychology, and mathematics. While the IE focused on the technological developments, and studies what could be realistically done using the currently available technology. The survey also listed what kind of sources could be used to gather the information about the context using sensors. The list contained light, sound, motion and acceleration, touch and pressure, location and distance, temperature and humidity, biometrical and size. Nowadays the users have access to different kinds of devices with different sensors that can be used to gather this

information, and utilizing this information, the current context of the user can be derived. The study stated that to get the proper context of the user, there was a need to understand the relationship and dependency of different kinds of context, to create a full picture of the current context of the user. The survey also found that the context could be categorized in three categories, which were - data collected of the user, the environment, and the system.

The study done by Qin et al. (2017) studied how cultural differences might affect the notion of context-awareness, and how people from different cultures saw the context-awareness. They stated that the aim of the context-awareness was to make the interaction with the device more intuitive and closer to what users were accustomed to, when interacting with other human beings. They argued that the higher level of contexts were more important for the user compared to the lower. The higher level of contexts meant activity, psychological status, and place, while the lower level of contexts were location, noise, gravity, and time. To give the user the proper higher-level context, it was necessary to interpret correctly the user's needs and the available lower levels of context. The higher level of context could be derived from the lower levels of contexts. The context-aware system should consider the culture of the user, so that the context is correctly interpreted, and correct service could be offered to the user.

In their survey, Alegre et al. (2016) found that while the term context was commonly understood it was still hard to describe for the people. In their survey they found that the adjective context-aware, was used to describe systems which could use context. Terms such as smart and intelligent were also used to describe systems, which could be seen as context-aware systems as well. Their survey found that, since there was no clear consensus on what context was, it led to difficulties to state clearly what contextual-awareness meant. There was also problem, since the same context might have different meaning from user to user.

To summarize, the contexts were the observations of the environment and the user utilizing different kinds of sensors and obtained contextual information was used to create context-awareness. Through different contextual information sources, a better picture of the current context could be derived. Context-awareness could be used to make applications react to

different contexts and changes in contexts. In addition, cultural differences should be considered when developing context-aware systems.

2.2 Solutions for context information gathering and providing context-awareness

This subsection focused on the previous studies made on the different solutions for context information gathering and how to provide context-awareness published during few previous years.

Rivero-Rodrigues et al. (2016) stated that the context-aware systems included two main components, which were context providers, and context-aware services. The context providers were for example mobile sensor frameworks, while the context-aware services provide the context reasoning. The authors stated that there could be three different ways to extract contextual information: straight from the sensors, using a middleware to get the already prepared information from the sensors about the context, and utilizing a server to gather contextual information. Getting the information straight from the sensors was not wise, especially when there were middleware systems available that provided the required information more easily and cost effectively. Five different kinds of sensors could be used to obtain required contextual information. These were physical sensors, virtual sensors, combined sensors, social media, and direct user input. The physical sensors could gather physical data from the user's environment, for example via the smartphone sensors. The virtual sensors were used to gather virtual information, for example from other applications. These kinds of sensors may be problematic, since the information could be in unstructured format, and not all the services had a public API to retrieve this information. Combined sensors could be viewed as a service, which combined information from multiple sources, and provided the combined information. Social media could be viewed as a sensor, which provided information about the user, and the relations between different users and entities. The problem with the social media as a source was that the information could be in hard to use format or the service provider did not want to share publicly the information contained on the site that made the use of the information difficult. Direct user input could be used as an information source; the risk here was not to overwhelm the user with requests for the information, but to use it when it provides the maximum amount of

benefit to the system, as well as for the user. Rivero-Rodrigues et al. (2016) listed different ways to gather contextual information. These were positioning techniques, semantic location, activity recognition, sentiment analysis and opinion mining, social network analysis, user profile inference, and file annotation. The positioning techniques used the built-in physical sensors to get the user's current location, while the semantic location was used to derive the meaning of the current location of the current user. Activity recognition was used to derive the current activity the user was doing. Sentiment analysis and opinion mining was used to derive the opinions of the user, for example related to the restaurant types users liked. This type of information could be gathered for example from the social media. The social network analysis could be used to gather information about the user and user's relations to other users. The user profile inference was used to get the idea about the user to offer the correct services for this specific user. The file annotation was similar to the user profiling, but it was done via studying the documents the user was viewing. Currently, the main way of creating context-aware applications was to use the different kinds of mobile sensor frameworks to gather the data, but the amount of data that could be collected this way was limited. To make greater use of the available information on the internet, new ways needed to be found that would allow easier ways to utilize the unstructured information that is available. One problem that the study found was that the frameworks developed for developing context-aware applications were often developed to focus on a specific need of an application, and were not widely usable. The study also pointed out that while the amount of context-aware systems increased the user privacy concerns grew.

Qin et al. (2017) stated that the device puts a restraint on how close to human-human interaction the context-aware application could become. The limiting factors were the processing power and the available sensors. The amount of information available to be used for context-aware applications was increasing via the information available online, as well as the new sensors and the data collected on the devices. This led to a situation where the smartphones could provide the services more tailored for user's needs. In addition, with the advancements in sensors and networks, the devices could sense other devices and users around them. Some of the concerns found in the context-aware applications were the users feeling of not being in control of their device and the users concerns about their privacy.

The users also felt that there was too much information delivered to them. Since the users were different from one to another, the contexts and their meaning changed from the user from time to time, the users experience may also differ when using a context-aware application. In addition, there could be situations where the user used the application in a context where the designers did not anticipate it to be used. The researchers stated that when developing a context-aware application, the designers and developers should take in to account the user's attitude, intention, and activity. This way the context-aware application could be seen more tailored for the user. The study found that the more intelligent the mobile applications became, the more fearful the users were about the evolution of humans and the requirements of cognitive processing.

Bedogni et al. (2016) studied a way to detect the transportation mode of the user without using the Global Positioning System (GPS) data. In the study, they automatically identified the resources of the user's smartphone and adjusted the used mechanism to detect the transportation mode so that the mode was detected in the most energy efficient way on the device without losing too much of the accuracy. The framework was a standalone application running on Android and it collected the data from the smartphone's sensors, and utilized this data to determine the user's current transportation mode. This information was then provided through Context Provider for other applications to utilize. The study stated that it was easy to recognize based on the speed, if the user was walking or driving a car, but it became hard when there was a need to specify if the user was for example using own car or if he was sitting on a bus. That was why the speed value could not be used as a sole factor when detecting the transportation mode. In addition, issues arose from the user's habits, since the user could carry the mobile device in different places. For example, the data collected by the sensors was different if the device was in a pocket or in a bag. To tackle the issue of battery life, the researchers presented a way to use different kinds of learners ordered from the least complex to most complex, and this way to get the most accurate information with the least complex learner possible. It was found that the costlier learning models provided the more accurate information about the current transportation mode. It was also found that the use of multiple sensors data guaranteed more accurate mode detection, and the use of all the available sensors provided the highest accuracy.

In their study, Elazhary et al. (2017) presented a tool to get easy access to the internal sensors of smartphones. With the tool, the raw data was processed to higher-level context for easier use in mobile applications. The tool recognized the available sensors of the device and adapted its functionality accordingly. The study found that the current context-aware frameworks had a narrow usability field and were focused on a specific task, and were not for a broad use in the context-awareness field. In their research, they divided the mobile device sensors into three categories: motion sensors, position sensors, and environmental sensors. The sensors could be further divided in hardware-based and software-based sensors. The motion sensors detected the motion of the mobile relative to time and space. These sensors were accelerometer, gravity sensor, gyroscope, rotation vector sensor, and step counter. The position sensors were used to determine the physical position of the mobile device. These sensors were magnetic field sensor and proximity sensor. The environmental sensors provided information about the surroundings of the mobile device. These sensors were ambient temperature sensor, light sensor, pressure sensor and relative humidity sensor. The Android Context Management (ACM), which the researchers proposed in their study, had a multi-layered architecture. At the lowest level were the raw context sources such as sensors and hardware features. These provided information to context manager in different context levels. The data could be asked straight from the raw context sources or it could go through preprocessing, and from there to higher level context and features, and finally it could also be post-processed to high level context and features. It all depended on what kind of information the developer required for his application. The developer could request specific context from the context manager, or request a specific context change notification from the context manager. The aim of the ACM was to provide the developers with more easily usable context values to support their development, without having to self-write complex queries to sensors, and interpret this data correctly.

Alegre et al. (2016) studied the literature about the development of context-aware systems. When developing context-aware systems the developers tried to simplify the complex world by reducing the complex observations to simpler ones, and use simplified models to gain access to the underlying patterns. Study saw a problem in this, since the simplification of problems did not give a full view of the situation. The aim should be to predict the

possible situations where the user could use the application. According to the study, it was nearly impossible to know beforehand all the possible situations the user might end up using the application. The researchers stated that the aim of the context-aware system was to monitor the context and act accordingly without user interaction, and if the system made a mistake interpreting the context, the user might reject the application. This could be helped by giving the user more control over the functioning of the system. The researchers listed different ways that could be used to interact with context-aware system. These were personalization, where the user could manually input preferences in to the application, passive context-awareness, where the application monitored the environment, and then offered the user different choices for actions, and active context-awareness, where the application monitored the environment and automatically acted on context changes. The researchers divided the interaction with context-aware systems in two different categories, which were execution and configuration. In the execution, the system acted based on the specific context, and in configuration, the user could adjust this action. The configuration could be automatic, where the system automatically changes its functionality based on what it had learned, or passive where the user could itself decide on different actions. The researchers listed the life cycle of context information as following: acquisition, modelling, reasoning, and finally dissemination. The privacy concerns were also pointed out, since the contextual data contains important and possibly private data to the user. The researchers stated that a middleware should be used to collect the contextual data from the sources such as sensors. This helped the developer in their task, since the collection of the context from the sources was separated from how the context was used, and since the collection style stayed the same from application to application, and only the context used needed to be changed. It was hard to create a middleware system, which could be used universally in different kinds of applications.

In their research, Zhang et al. (2017) presented a context-awareness middleware system, which combined a Software Development Kit (SDK) and cloud computing engine to support development of contextual-aware services. Contextual-aware system was defined as something that could modify functions based on the current context without the user's need to intervene. The researchers stated that while nowadays the operating system provided APIs, which could be used to get the context information, the low-level sensor

data was not enough for today's developer's demands when developing context-aware applications. Zhang et al. (2017) stated that the middleware, Senz, they were proposing in the article was energy efficient, had high context recognition accuracy and high performance. The most innovative part of the middleware was the capability to use online resources to make the context recognition accuracy higher, according to researchers. The middleware consisted of three different parts. First, the SDK collected the data, refined it and transmitted the data to the server, and listened on the server for notifications. The second part was the backend server, which held the collected data, analyzed it, and gave notifications back to the SDK about the changes in the context. The third part was the web portal, which visualized the data and gave configuration options for the middleware system. The context recognition had motion status recognition, ambient sound recognition, location point of interest analysis using public map providers, and context activity classifier. The energy efficiency of the system came from the fact that most of the computations were run in the backend and not locally on the device.

To summarize, the context-aware systems could be divided in two components: context providers, which handled the gathering of the contextual information and deriving the context from the data, and context-aware services, which utilized this information to provide contextual reasoning. In addition, the use of middleware to deal with the information extraction from the sensors was recommended. Multiple different sources could be used to extract contextual information such as different kinds of sensors, social media, and user input. It was also stated that the context-aware systems should allow some configuration options to the users, or automatically adapt to the changing situations. The studies also found issues with current frameworks. It was stated multiple times that the current frameworks were too focused on solving a single issue and could not be used universally. Another issue in the development of context-aware systems was the resource limitations of the devices, especially the battery life. The battery life could be made better by making the architecture of the context-aware system to utilize a server, to make the computations to derive the context. In addition, privacy concerns were raised by the researchers related to the contextual information gathering and use. These privacy concerns became more of an issue when the computations were done on the server and not on the device.

2.3 Different kinds of context-aware applications and their use cases

In this subsection different kind context-aware use cases and applications were presented found during the literature review.

The survey done by Augusto et al. (2017) listed different kinds of application domains in the context-awareness field. The list contained location-based services, information providers, recommender systems, education, health and sport, traveling and tourism, logistic, e-democracy, smart homes and cities, and crowd-based applications. The location-based services adapted their behavior to the user's current location. The information providers tried to provide the user the required information through prefiltering the available information based on the user's preferences and context. The recommender systems were used to provide recommendation to the user based on the user's preferences and context. In the education domain, the context-awareness could be used to personalize the learning process for every user. The health and sport domain could use context-awareness to make the healthcare systems and sport applications work more intuitively and help the user. In the traveling and tourism domain, the context awareness could be used to support the user's travel experience. In the logistics domain, the context-awareness could be used to recommend the user the best possible route from A to B based on the user's preferences and context as well as the historical data available. In the e-democracy domain, the context-awareness could be used to study the opinions of the user to support the work of the governments to fulfil the needs of the people. In the smart homes and smart cities domain, the context-awareness could be used to support the functions. The smart environments were full of sensors, which provided information, and the context-awareness used this provided information to make the inhabitants of the smart environments to benefit from the provided information. In the crowd-based applications domain, context-awareness could be used to connect people who were using the system. The study found that the location-based services were currently the dominating domain where the context-awareness was used.

Bedogni et al. (2016) studied how to detect the transportation mode using context-awareness when developing Android applications using the sensors present in

smartphones. The focus was on trying to detect the transportation mode without utilizing the GPS and computationally heavy algorithms, to spare the resources of the smartphone. Their aim was to utilize the accelerometer and gyroscope data of the smartphone accompanied with the classifier, which was developed as part of the study to detect the transportation mode of the user. They also focused on keeping the user's information private, since the multitude of the different kinds of sensor data, which could be collected from the user's smartphone, was huge. The focus in transportation mode recognition was to identify automatically what kind of vehicle the user was currently using. The transportation mode recognition was a sub class of the activity recognition. The researchers provided five different kinds of use case scenarios for the utilization of the transportation mode information. These were collection of the mobility data for use in planning the overall transportation. Profiling user habits and their health monitoring by studying where the users moved at different times of day, and how much they got physical exercise. The smartphone functions adapted based on the user's transportation mode. For example, the ringtone volume was adjusted based on if the user was walking or sitting still. The advertisement could be focused more accurately based on the information gathered of the user's transportation mode. For example, when the user was using his own car, the advertisement could be provided about near gas stations, while if the user was on a public transportation, this kind of information was not useful to the user. The researchers also viewed the overall adaptation of services as one of the potential use cases. For example, while using a route planning application, the user would not need to provide the transportation mode that the user was currently using, and the application would automatically detect this information. The researchers present a What Am I Doing (WAID) framework. It could run on the background or foreground on the Android. It had two different modes: training and predicting. While in the training mode, the framework asked for the user input to determine what kind of transportation mode the user is in currently, and collected the sensor data at fixed rate. Once the data was collected, the model builder used the four different kinds of learners to generate models from the training set. While the framework was in predicting mode, the sensor data was gathered, and the models generated in the training mode were used to recognize the user's current transportation mode. This information could then be used by other Android applications by asking the Content Provider for the information. The issue the researchers saw in their approach was

the requirement for the user to interact with the framework in the training mode. To overcome this issue, they were considering providing the framework with pre-defined training sets, which could be extended by voluntary user's input. There were also issues in determining the transportation mode between, for example, walking and biking, and this issue could be overcome by supporting the framework with GPS data.

In their survey, Alegre et al. (2016) stated that the context-awareness was an important part when developing systems for Intelligent Environments, Pervasive & Ubiquitous Computing, and Ambient Intelligence. The need came from that these systems required higher understanding of situations where specific services and functionalities should be provided. This way the user's efforts could be minimized by automating functionalities. They also found out that the challenges in this field were different from each other and typically complex, which lead the solutions to be made specifically to solve a single issue and not to be universal solutions. The researchers stated that it was easy to make a simple context-aware application, for example, by just using the user's location as a trigger to provide specific service, but there were visions that were more ambitious in context-aware development to create much more intelligent services. The researchers proposed a following list of features of context-aware system: "1. Presentation of information to the stakeholders, 2. Active or passive execution of a service, 3. Active or passive configuration of a service and 4. Tagging context to information".

Deore et al. (2016) presented an Android application, which detected the microenvironment using built in smartphone sensors. The application, Sherlock, had three modules: phone placement detection, phone interaction detection, and backing material detection. These modules were used to provide the following features: auto call picker, pressure sensor, which was used for security, unauthorized access protection, battery saving, ringer, and vibrate toggle on soft surfaces. Auto call picker functioned in a way that if the application noticed that the user was in situation that he could not answer the phone by swiping; the application would automatically answer the call when the phone was brought next to ear. The pressure sensor was used for the protection in a way that if the application noticed that the user was providing more pressure than usually on the screen, a notification was sent to the configured number or email address. The unauthorized access

protection functioned in a way that if incorrect pattern was used when trying to open the phone, an email was sent to the user, containing photo from the front camera, along with the datetime information accompanied with the GPS coordinates. The battery saving function was done in a way that the application automatically noticed if the user was not actively using the phone, and shut down the unnecessary processes. The vibrate toggle was done in a way that the application noticed if the phone was on a soft surface, and if a call was received, the application changed from vibrate to ring tone, so that the user noticed that there was a call incoming.

Masango et al. (2016) proposed an Android application, Context Aware Mobile Application (CAMA), where the user could control authentication to different applications based on the user location. The user could set safe zones where no authentication was asked, but after leaving the safe zone, a pin code was asked to enter the application. If the user went far enough from the safe zone, the user was asked for a username and password combination to get access to application.

Hadjioannou et al. (2016) presented an Android application, which provided a feature to track people, and notify other preset people via Short Message Service (SMS) if the user went out of any of the preset locations. The aim was to help track elderly people, children, or anyone who might be needing assistance if they got lost. Other functionality in this application was that could be three different gestures that one could perform after which a SMS would be sent to people listed to receive said SMS.

Künzler et al. (2017) studied the notification management systems that used context-awareness. They stated that these kinds of systems have gained increasing attention, making the notifications the smartphone presented to users more context-aware. What this meant was that the notifications should be presented based on the context the user was in to minimize the distractive nature of notifications. Through their literature review, they found that the sources of context information to achieve this were: location, microphone, Bluetooth, User Interface (UI) events, communication, user defined, Wi-Fi, Time, Accelerometer, and others, which included battery status or if the phone is covered with something.

Schick et al. (2016) developed an iOS and Android applications to support quitting smoking. The app functioned in a way that first two weeks the quitter added entries of each smoke and cravings. After this, the app started to give supportive notifications based on the data entered during the first two weeks. Location and time were used to derive the need for supportive notifications.

Posdorfer and Maalej (2016) presented an iOS application prototype to be used for adding context-awareness to surveys. This was done with the support of using iBeacon technology from Apple. The prototype functioned in a way that it noticed the location of the user and the duration the user has been at that location. That way the survey could be modified based on the rules set by the developers.

Grubert et al. (2016) studied the use of context-awareness in Augmented Reality (AR) setting. They found that current AR implementations used very narrow field of contexts to produce the service. The current implementations of AR used context-awareness to modify the system input based on the context, adapting the system configuration based on the context and adapting the system output based on the context by changing the content, or information presentation. They proposed that the context-awareness could be used to preserve battery life by choosing only necessary sensors, to provide the current AR experience, the UI could be better adapted to the motion of the user, detecting other available devices to, for example, make it possible for user to move seamlessly from one display or input method to other, and to improve the privacy of such systems by making sure that other people's privacy was respected. They also proposed that while nowadays gaze-trackers, velocity sensors, heart rate monitors, and pedometers were not widely used in AR field, these could improve the development of contextually aware AR services.

Wang et al. (2016) proposed a way to detect if the user is indoors or outdoors, using Global System for Mobile communications (GSM) signal strength from four nearby GSM stations supported by machine learning algorithm.

Sugijarto et al. (2018) presented a way to use context-awareness in Android application in

support to blood donation process of the user. The start and end of the process were recognized using Near-Field Communication (NFC) tags. The application guided the user through the process. There was also possibility for the user to set the application to send a notification if there was an active blood donation campaign nearby. The application also offered other functions related to the blood donating, such as sending messages to donors, collecting points from blood donations, checking donation history, etc.

Liotou et al. (2016) presented a way to improve the quality of experience of video streaming while the watcher was moving by using context-awareness. They proposed to use context-awareness to support predicting if the user was going to enter a zone where the internet connection was weaker or lost, and this way prevented the stalling of the streaming by using Hypertext Transfer Protocol (HTTP) adaptive streaming strategies. What this meant was that when it was noticed that the user would enter in a zone where the network availability was unstable, the quality of the video requested from the server would be lowered, so that more video could be buffered on the device.

Wahle et al. (2016) presented an application, Mobile Sensing and Support (MOSS), which could be used to support people with depression. The application used context-awareness methods to track the user's behavior and utilizing this information provided recommendations for the user. For example, the user was encouraged to take a walk if the user had been sitting all day at home. The application also provided different kinds of interventions to the user based on the contextual information gathered. These interventions were based on therapeutic cognitive behavioral therapy (CBT) interventions. The contextual information gathered by the app to achieve this were general activity, walking time, time at home, phone usage, geographic movement, number of unique Wi-Fi fingerprints, number of text messages, number of calls, and number of calendar events.

Pejovic et al. (2016) studied the way context-awareness could be used to study human behavior using mobile devices. The mobile device was used to sense the context the user was in, and trigger a survey to the user when an interesting moment happened. The survey along with the data collected of context could be then transferred to the researchers. Context-awareness could also be used to recognize a correct time to ask the user to fill the

survey, so that it did not interfere with user's life.

Shoaib et al. (2016) studied the use of smartphone and smartwatch sensors to recognize human activities. They evaluated the use of accelerometer, gyroscope and linear acceleration sensors in recognizing the human activity. They found that using the combination of both smartphone and smartwatch sensors increased the accuracy of detecting the correct activity, especially in complex activities. They also found that increasing the window size of collecting data increased the accuracy, when detecting complex activities such as smoking, drinking coffee, eating, or giving a talk. Simpler activities could be recognized by only using either smartphone or smartwatch alone.

Kashevnik et al. (2018) proposed the usage of context-awareness in electronic bikes. The context-awareness was used to control how much the motor helped the cyclist as well as giving recommendations to the user while driving. This was achieved by implementing sensors on the bike as well as using the smartphone sensors and cameras.

Vaizman et al. (2017) recognized that context-awareness could be used in healthcare to support health monitoring or to support aging people to detect any aging related issues, or support at home. They also proposed that context-awareness could be used to time the interventions at the right time, for example, for person who was trying to quit smoking, or to warn a recovering alcoholic that there might be a risk to relapse. To achieve this kind of functionality they proposed the use of smartphone sensors supported by smartwatch sensors. This way the functionality could be implemented in an unobtrusive way, since these devices were normal for people to wear.

To summarize, the location-based context-aware services seem to be dominating the current context-aware systems. Other domains found during literature review, where context-awareness was used, were different kinds of health-related applications, information providers, recommender systems, education, traveling, smart environments, activity tracking, notification management, survey management, making services more automated and intelligent, authentication methods, tracking people, improving augmented reality and human behavior studies.

2.4 The usage of Google Awareness API

In this subsection recent publications were reviewed which used Google Awareness API in their work.

Garcia-de-Prado et al. (2018) used Google Awareness API in their Air4People service, which provided information about the air quality through smartphone notifications. The Google Awareness API was used to gather the user's current context. The contexts the developers were interested in were the location of the user, the physical activity the user was in, nearby places for suggestions where the air quality was better than where the user currently was in, and weather conditions to make suggestions more personal, for example, in a case where the user was more prone to get a cold. These suggestions were based on the information provided by the user, which the researchers call personal context.

Hassani et al. (2017) used Google Awareness API in their Android prototype application, which is aimed to create a Peer-to-Peer (P2P) network in emergencies where network connection was not available. Google Awareness API was used to add contextual information to the prototype to be used to achieve the P2P network in changing situations, and to add information about the conditions that the user was in when making notes about the situation in the field.

Beierle et al. (2017) proposed the use of Google Awareness API to get the user's current time, location, nearby places, nearby beacons, headphone state, activity, and weather. This information was proposed to be used alongside the other data to improve the friend recommendations in social networking services.

Beierle et al. (2018) stated that while Google Awareness API could be used to retrieve different context data such as time, location, places, beacons, headphones, activity, and weather, it was not enough when trying to track the user's full context and other means were needed to achieve this alongside of the Google Awareness API.

Gedeon et al. (2018) presented a framework, which could be used to make micro-storages for data work for mobile devices at the edge of the network. To achieve this, they used

context-awareness to get the user's context so that the correct micro-storage could be used. The Google Awareness API was used to get the user's current activity context, while other means were used for other contextual information gathering.

In conclusion, not many studies have been published utilizing Google Awareness API. The reason for this was probably that the Google Awareness API is still quite new service.

3 GOOGLE AWARENESS API

In this chapter, the Google Awareness API was studied. The findings from the literature review were reflected to the examination of the Google Awareness API. In addition, the possibilities and the limitations of the API were studied. The API was studied trough the guide and documentation provided by Google[1][2].

3.1 Overview

The Google Awareness API is a context provider for the developers of context-aware applications. It aims to make it simpler for the developers to create applications, which can react to the user's current context and to the changes in the user's context. It allows the developer to get access to seven different contextual information types. These are time, location, places, beacons, headphones, activity and weather information. The API also handles the management of the system resources, especially the power consumption and memory usage that are automatically optimized.

The context types provided by the API include the information gathered from the device's sensors. These are the location, the place type, and the activity the user is in. At the moment, the Awareness API provides the following context types: semantic time intervals such as morning, night, weekend, etc., location, place which is the user's current semantic location, activity, beacons, headphones, and the current weather conditions such as temperature, feels-like temperature, dewpoint and humidity. The activity context contains values for on foot, running, walking, biking, staying still, and in vehicle activities. To make a full use of the API and to make the application more context-aware, it is advised to combine the data from these information sources. The beacon context can be used only on devices running Android API level 18 or higher, which starts to be nowadays old Android phones running Android 4.3.

To use the Google Awareness API, the application requires permissions to access the ACTIVITY_RECOGNITION and ACCESS_FINE_LOCATION. The activity recognition

[1] Google Awareness API. Retrieved Apr 7, from https://developers.google.com/awareness/overview
[2] What is the Awareness API?. Retrieved Apr 7, from https://developers.google.com/android/reference/com/google/android/gms/awareness/package-summary

permission allows the gathering of the information used to recognize the physical activity the user is in, and the fine location permission allows the access to the information used to get the precise location of the user.

Google provided four guidelines to follow when developing context-aware application using the Google Awareness API. These guided the developer to inform the user of the features of the application so that they would not surprise the user. The developers were also guided not to overwhelm the user with notifications. While the API was designed to preserve the system resources of the device, if the developer extensively used the API, it could still lead to a situation where the battery or memory gets depleted. Because of this, the developers were guided to make wise decisions on how extensively the API would be used on the application. Finally, Google pointed out that the API should be used to provide awareness to the application, and if that was not what the developer aimed for, other means should be used to achieve the required features.

The API was divided into two different APIs. These were Fence API and Snapshot API. The Fence API allowed the developer to setup fences, which meant that it allowed reacting on the changes of the user's environment. While the Snapshot API allowed the developer to gain access to the user's current environmental information.

3.2 Fence API

The Fence API allowed the developer to setup different fences, which makes it possible to create rules combining multiple context conditions. Once these conditions were met, the API gave a notification, and the application could act on this information to provide better user experience.

The Fence API was based on the concept taken from the geofencing, which were based on the actions made on the application when the user enters and exits a specific region. This was expanded to use also other contextual information, not just the location information. The Fence API could use the user's location information, the current activity the user was in, the information on if the headphones were plugged in, and the information about the nearby beacons. Using these information sources, the developer could create different

kinds of fences using the Boolean operator AND, OR and NOT. This way the application could use these created fences to act on when the specific combination of different contexts was met.

There are two different kind of fences the developer could create (Google calls them primitive fences and combination fences). The difference was that primitive fences focused on acting on information from a single fence, while the combination fences combined multiple different primitive fences using the Boolean operators. Google provided an example of a primitive fence as a fence, which detected if the user was walking, and the developer could act upon on this information. While with the combination fence, the Google example extended the walking fence to cover also the context of the headphone state. The developer could check if the user was walking and the headphones were plugged in, and when these conditions were met act upon it accordingly.

The status of the context could be also examined in the fences as transitions. The developer could ask for information if an activity was starting or stopping as well as ask for information that the activity was currently happening. Same information was also provided in regard to the headphone state. The fence could be built to notice when the user was plugging the headphones in or unplugging them, or to notice that the headphones were currently plugged in. Same kind of information could be also utilized when using the location information to create a fence. It could be observed that the user was entering or exiting a specific coordinate accompanied with the radius information, as well as the information that the user was staying in specific location.

It should be noted that while the Google Awareness API also provided contextual information on the nearby places and the weather conditions these could not be used while creating fences. The developers were instructed to use the Snapshot API to gain access to these information sources to support their application development.

3.3 Snapshot API

The Snapshot API allowed the developers to make requests on the user's current contextual information. The contextual signals Snapshot API provided were the user's

current activity, the nearby beacons, the information on the headphone state i.e. were they plugged in the device or not, the location of the user, the place the user was in currently, the current time interval, and the current weather conditions at the user's location. This information was cached on the device to make it fast to retrieve the information. If the information was not found in the cache, it was freshly sensed and analyzed so it could then be provided.

The activity recognition provided a list of the probable activities the user was in or has recently been. This list was ordered from the most probable to the least probable. In addition, information about only the most probable activity could be fetched. The confidence level of the recognized activity could be retrieved through the API as well.

The nearby places information also contained a list of the potential places the user was currently located. In this list, the places were ranked by the likelihood that the user was in that specific place.

3.4 Possibilities

The Google Awareness API provided possibilities to create context-aware applications, which used the following contexts to achieve their tasks: beacons, activity, headphone state, location, places, time intervals, and weather. The possibility either to track the contextual changes through fences or to get the current context could be considered as positive. These options allowed the developers to create different kinds of solutions without having to write complex context analyzers themselves.

The provided context sources gave an easy way to create location-based context-aware applications, which were found according to the literature review to be dominating the context-aware applications field. In addition, these sources could be used to make the service more intelligent and automated. The activity recognition was also one of the themes of the context-aware research found throughout the literature review. In addition, beacon usage was mentioned in the literature review, which the API supports. Different kind of people tracking solutions could use the location and places information in fences to provide a system to track the application users, which was also one of the topics mentioned

in the literature review.

3.5 Limitations

The limitations of the Google Awareness API when developing context-aware applications were the missing context sources. According to the literature review, for example social media and internet sources in general were such context sources. This kind of information could be used to develop better information providers and recommender systems that were one of the suggested focus points for context-aware applications. Of course, the places and location information could be used to support these kinds of applications to some extent. In addition, there was the limitation of the system specific contexts, since the API only allowed to gain access to the information about the headphone state. This could be extended, for example, to cover also information about the user's application usage on the device, access to the microphone and information about the overall usage of the device. This could be a seen as a privacy issue, which could already be seen by the usage of the API. The activity recognition alongside the location and places tracking provides a lot of information to the application developers about the user.

There were also limitations in the activity recognition. In the literature review, it was studied how to detect different methods of transportation. The Google Awareness API did not differentiate between, for example, public transportation and driving your own car. This could be a possible improvement point of the API.

The API was focused on smartphone usage. This could be extended to cover also the sensors available on smartwatches. For example, while the activity recognition could be used to support creation of health-related context-aware applications, the support for usage of the smartwatches could add to this. Through the smartwatch's sensors, for example, heartrate data could be used. In addition, according the literature review it was found that the combination of the use of sensors from both smartphone and smartwatch improved the activity recognition.

4 RESULTS & CONCLUSIONS

In the literature review the definition of the context and context-awareness were studied. It was found that contexts were information pieces about the environment and the user, and utilizing this information, context-aware applications could be created. The more information about the contexts were available, the better picture of the context could be generated. The context-aware applications should react to the context the user was currently in and to the changes in the context of the user.

The literature review covered also the research done on the contextual information gathering. It was found that the contextual information should be gathered by using some sort of middleware, so that the developers would not need to get the information straight from the sensors. The Google Awareness API studied for this thesis provided this kind of service, and the application developer could focus on the implementation of the context-awareness without having to deal with getting the information straight from the source. The Google Awareness API also provided the contextual reasoning so that the developer could focus on the application development without having to make context reasoning. Other issue found out during the literature review were regarding the resource management of the device when developing context-aware applications. The Google Awareness API provided help with this issue by automatically managing the resources of the application. Third issue found out during the literature review, was the privacy of the user. When using the Google Awareness API this issue was relevant, since the application creator got access to the user's location and activity histories that could be a privacy issue.

The literature review studied also the different kinds of applications and use cases in the context-awareness domain. The Google Awareness API could be easily used to create location-based context-aware applications, which are currently dominating the context-aware services. In addition, different kinds of automatizations could be created by using the contextual information provided by the API to make the user experience better in applications.

The Google Awareness API was found to provide set of tools to create context-aware

applications, which utilized the information about the user's location and activity. In the literature review, it was found that there was an increasing need to utilize information from the social medias and the internet overall, but for this the API did not offer a tool. Also, the API did not provide support to use all the available sensors of the device, e.g. microphone or cameras. In addition, the API was focused on the smartphone devices, and addition of smartwatches could be seen beneficial. This information could be gathered by other means, but the focus of the thesis was on the Google Awareness API and it does not provide this kind of support.

For future work, a prototype application utilizing the Google Awareness API should be implemented to see how well it works in real-world situations.

5 SUMMARY

In this bachelor's thesis, a literature review covering 30 articles was done. The literature review focused on the topic of context and context-awareness, how to gather the contextual information to create context-aware applications, what kind of applications and use cases were present in the research articles, and how was the Google Awareness API used currently.

The Google Awareness API was studied by examining the guides and documentation provided by the Google. It was found that the Google Awareness API provided tools for the developers to create context-aware applications without having to get the contextual information straight from the sensors and writing complex code to derive the context from these information sources. The Google Awareness API provided the developers with information about the user's current location, activity, nearby beacons, weather, headphone state, and the semantic time. Utilizing these information sources, the developer could create application, which was context-aware.

REFERENCES

Alegre, U., Augusto, J. C., & Clark, T. (2016). Engineering context-aware systems and applications: A survey. *Journal of Systems and Software*, *117*, 55-83.

Augusto, J., Aztiria, A., Kramer, D., & Alegre, U. (2017). A survey on the evolution of the notion of context-awareness. *Applied Artificial Intelligence*, *31*(7-8), 613-642.

Bedogni, L., Di Felice, M., & Bononi, L. (2016). Context-aware Android applications through transportation mode detection techniques. *Wireless communications and mobile computing*, *16*(16), 2523-2541.

Beierle, F., Grunert, K., Göndör, S., & Schlüter, V. (2017, June). Towards psychometrics-based friend recommendations in social networking services. In *2017 IEEE International Conference on AI & Mobile Services (AIMS)* (pp. 105-108). IEEE.

Beierle, F., Tran, V. T., Allemand, M., Neff, P., Schlee, W., Probst, T., ... & Zimmermann, J. (2018, May). TYDR–Track Your Daily Routine. Android App for Tracking Smartphone Sensor and Usage Data. In *2018 IEEE/ACM 5th International Conference on Mobile Software Engineering and Systems (MOBILESoft)* (pp. 72-75). IEEE.

Deore, P., Bhagwat, V., Phad, S., & Ghule, G. (2016). Android application for Micro-environment detection using mobile sensors. *IRJET*, *3*, 1173-1175.

Dey, AK & Abowd, GD 1999, 'Towards a better understanding of context and context-awareness', *In HUC'99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*.

Dey, AK 2001, 'Understanding and using context', *Personal and ubiquitous computing*, vol. 5, no. 1, pp. 4-7.

Elazhary, H., Althubyani, A., Ahmed, L., Alharbi, B., Alzahrani, N., & Almutairi, R. (2017). Context Management for Supporting Context-aware Android Applications Development. *International Journal of Interactive Mobile Technologies*, *11*(4).

Garcia-de-Prado, A., Ortiz, G., Boubeta-Puig, J., & Corral-Plaza, D. (2018). Air4People: a Smart Air Quality Monitoring and Context-Aware Notification System. *Journal of Universal Computer Science*, *24*(7), 846-863.

Gedeon, J., Himmelmann, N., Felka, P., Herrlich, F., Stein, M., & Mühlhäuser, M. (2018, February). vStore: A context-aware framework for mobile micro-storage at the edge. In *International Conference on Mobile Computing, Applications, and Services*(pp. 165-182). Springer, Cham.

Grubert, J., Langlotz, T., Zollmann, S., & Regenbrecht, H. (2017). Towards pervasive augmented reality: Context-awareness in augmented reality. *IEEE transactions on visualization and computer graphics*, *23*(6), 1706-1724.

Hadjioannou, V., Mavromoustakis, C. X., Mastorakis, G., Markakis, E. K., Valavani, D., & Pallis, E. (2016, July). Context awareness location-based android application for tracking purposes in assisted living. In *2016 International Conference on Telecommunications and Multimedia (TEMU)* (pp. 1-7). IEEE.

Hassani, A., Haghighi, P. D., Burstein, F., & Davey, S. (2017, June). Context Aware Data Synchronisation During Emergencies. In *International and Interdisciplinary Conference on Modeling and Using Context* (pp. 406-417). Springer, Cham.

Kashevnik, A., Pilla, F., Russo, G., Timoney, D., Sweeney, S., Shorten, R., & Ordonez-Hurtado, R. (2018). Context-Based Cyclist Intelligent Support: An Approach to e-Bike Control Based on Smartphone Sensors. In *Internet of Things, Smart Spaces, and Next Generation Networks and Systems* (pp. 16-22). Springer, Cham.

Künzler, F., Kramer, J. N., & Kowatsch, T. (2017, October). Efficacy of mobile context-aware notification management systems: A systematic literature review and meta-analysis. In *2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*(pp. 131-138). IEEE.

Liotou, E., Hoßfeld, T., Moldovan, C., Metzger, F., Tsolkas, D., & Passas, N. (2016, May). Enriching HTTP adaptive streaming with context awareness: A tunnel case study. In *2016 IEEE International Conference on Communications (ICC)* (pp. 1-6). IEEE.

Masango, M., Mouton, F., Nottingham, A., & Mtsweni, J. (2016, August). Context aware mobile application for mobile devices. In *2016 Information Security for South Africa (ISSA)* (pp. 85-90). IEEE.

Pejovic, V., Lathia, N., Mascolo, C., & Musolesi, M. (2016). Mobile-based experience sampling for behaviour research. In *Emotions and Personality in Personalized Services* (pp. 141-161). Springer, Cham.

Posdorfer, W., & Maalej, W. (2016). Towards context-aware surveys using bluetooth beacons. *Procedia Computer Science*, *83*, 42-49.

Qin, X., Tan, C. W., Bødker, M., Sun, W., & Clemmensen, T. (2017, September). Culturally informed notions of mobile context awareness-lessons learned from user-centred exploration of concepts of context and context awareness. In *IFIP Conference on Human-Computer Interaction* (pp. 420-440). Springer, Cham.

Rivero-Rodriguez, A., Pileggi, P., & Nykänen, O. A. (2016). Mobile context-aware systems: technologies, resources and applications. *International Journal of Interactive Mobile Technologies*, *10*(2), 25-32.

Schick, Rob & Humphris, Gerry & Kelsey, Thomas & Marston, John & Samson, Kay. (2016). MapMySmoke–A Context Aware Mobile Phone Application Targeted at Smoking Cessation. 10.13140/RG.2.2.27703.32165.

Schilit, B, Adams, N, & Want, R 1994, December, 'Context-aware computing applications', *Mobile Computing Systems and Applications, 1994*, pp. 85-90, IEEE.

Shoaib, M., Bosch, S., Incel, O., Scholten, H., & Havinga, P. (2016). Complex human activity recognition using smartphone and wrist-worn motion sensors. *Sensors*, *16*(4), 426.

Sugijarto, D. P., Mukhtar, M., Safie, N., & Sulaiman, R. (2018). Developing Context Awareness Mobile Application for Blood Donation. *JOIV: International Journal on Informatics Visualization*, *2*(3), 118-126.

Vaizman, Y., Ellis, K., & Lanckriet, G. (2017). Recognizing detailed human context in the wild from smartphones and smartwatches. *IEEE Pervasive Computing*, *16*(4), 62-74.

Wahle, F., Kowatsch, T., Fleisch, E., Rufer, M., & Weidt, S. (2016). Mobile sensing and support for people with depression: a pilot trial in the wild. *JMIR mHealth and uHealth*, *4*(3), e111.

Wang, W., Chang, Q., Li, Q., Shi, Z., & Chen, W. (2016). Indoor-outdoor detection using a smart phone sensor. *Sensors*, *16*(10), 1563.

Zhang, H., Huang, T., Liu, Y., Zhu, S., Gui, G., & Chi, Y. (2017, June). Senz: A Context Awareness Middleware System Used in Mobile Devices. In *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)* (pp. 1-7). IEEE.