"A framework for context-aware applications using augmented reality:"
"A train station navigation proof-of-concept on Google Android"

by
Freek Uijtdewilligen

in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Telematics

at the University of Twente
Faculty of EEMCS
January 13, 2010

Thesis supervisors:

Dr. Luís Ferreira Pires	Luiz Olavo Bonino da Silva Santos
Dr. Marten van Sinderen	Martijn Vlietstra

Abstract

Augmented reality combines a view of reality with virtual content in real time in order to provide an interface to enhance perception of and interaction with the real world. Although a number of augmented reality frameworks and applications exist, none of these frameworks is capable of providing augmented reality indoor, without any necessary changes to the surroundings, while running on a mobile device.

Furthermore, these frameworks are not *context-aware*, i.e., they do not use context information to provide or adjust relevant services to the user.

This thesis presents a framework, called the ARCA Framework, which combines the application support for augmented reality and context-awareness. The framework consists of software running on a mobile device, and a server application, called the Context Information Service (CIS). The CIS facilitates the gathering and refining of context information, and the provisioning of context information and content to other software and systems, such as the mobile device. Furthermore, rules can be used to specify conditions of the context, along with notifications that are triggered when these conditions are met.

The software on the mobile device provides an augmented reality view, showing information and handling notifications that have been provided by the CIS. Means are provided to adjust the framework to facilitate application-specific behavior, user interfaces, graphics, and notification handling.

Although the framework has been designed and implemented to be independent of the used localization process, a probabilistic technique, based on WLAN access points, has been implemented to provide support for indoor environments. The framework functionality has been evaluated by implementing a train station navigation application, called the NS Navigator, which is supported by the framework.

Acknowledgements

This thesis has been written to describe the results of the final project I have carried out for the Master of Science in Telematics at the University of Twente.

Firstly, I would like to thank my supervisors from the university, Luís Ferreira Pires and Luiz Olavo Bonino da Silva Santos. Our discussions, their critical remarks (the returned report chapters with numerous red scribblings), and, of course, their overall support and encouragements definitely took the project, and especially the report, to a higher level.

Next, my gratitude goes out to Martijn Vlietstra, my supervisor from Logica, for his help and support, and for our conversations, both *on*-topic and sometimes very much *off*-topic. The time and effort he and Eric van der Laan have put in my future career have proven to be successful, so I am very grateful for their efforts, now it is up to me not to let them down. I would also like to thank the other graduate students for making the time spent together very enjoyable, and I wish them all good luck with their own careers.

From the Nederlandse Spoorwegen, I would like to thank Corine van Drunen for her guidance and creative ideas during our brainstorm sessions, as well as involving me in her work.

I would also like to thank my parents for supporting me throughout my years in Enschede, for believing in me, and providing me with the means to study, even though it took a bit longer than anticipated.

And last, but certainly not least, I would like to thank my girlfriend, Hanneke Stoel, for being there for me, listening to my endless, overly detailed explanations of my problems and victories during this project.

Arnhem, January 13, 2010

Freek Uijtdewilligen

Contents

1. Introduction **1**
- 1.1. Motivation . . . 1
- 1.2. Objectives . . . 3
- 1.3. Approach . . . 3
- 1.4. Structure of this report . . . 4

2. Background **5**
- 2.1. Context-awareness . . . 5
  - 2.1.1. Context . . . 5
  - 2.1.2. Context categorization . . . 6
  - 2.1.3. Context-awareness . . . 6
  - 2.1.4. Applications . . . 7
- 2.2. Augmented Reality . . . 8
  - 2.2.1. Requirements . . . 8
  - 2.2.2. Purpose and issues . . . 9
  - 2.2.3. Approaches . . . 9
- 2.3. Localization . . . 13
  - 2.3.1. Location estimation . . . 14
  - 2.3.2. Measurement parameters . . . 14
  - 2.3.3. Localization techniques . . . 16
  - 2.3.4. Location-based services . . . 22

3. Requirements **24**
- 3.1. Scenarios . . . 24
- 3.2. Stakeholders . . . 26
  - 3.2.1. Application stakeholders . . . 26
  - 3.2.2. Framework stakeholders . . . 27
- 3.3. Application requirements . . . 27
  - 3.3.1. Functional requirements . . . 27
  - 3.3.2. Non-functional requirements . . . 28
- 3.4. Framework requirements . . . 28
  - 3.4.1. Functional requirements . . . 29
  - 3.4.2. Non-functional requirements . . . 30

4. Framework architecture **32**
- 4.1. Components . . . 32

4.1.1. Context-awareness . . . 33
4.1.2. Augmented reality . . . 35
4.1.3. User interaction . . . 36
4.2. Context-awareness component . . . 37
4.3. Augmented reality component . . . 39
4.4. Component interaction . . . 40

5. Framework software design **42**
5.1. Localization . . . 42
5.2. Context awareness . . . 45
5.2.1. Context Information Service (CIS) . . . 46
5.2.2. Communication with mobile device . . . 50
5.3. Augmented reality . . . 51
5.4. Platform-independence . . . 52

6. Framework implementation **54**
6.1. Approach . . . 54
6.2. Localization . . . 55
6.3. Context-awareness component . . . 56
6.3.1. Context Management Service . . . 57
6.3.2. Announcement and Notifications Service . . . 60
6.4. Augmented reality component . . . 63
6.5. Application deployment . . . 69
6.5.1. Resources . . . 69
6.5.2. Server components . . . 71

7. Evaluation **72**
7.1. Application development . . . 72
7.1.1. Content . . . 73
7.1.2. Notifications . . . 75
7.1.3. Map . . . 76
7.2. Application evaluation . . . 77
7.3. Overall conclusion . . . 79

8. Final remarks **80**
8.1. Conclusions . . . 80
8.2. Future work . . . 82

Appendix **83**

A. Class diagrams **83**

Bibliography **90**

Chapter 1.

Introduction

This chapter introduces the motivation behind the research that has been carried out, along with the structure of the research and this thesis. It identifies the relevance of context-aware applications with a particular focus on the architecture design.

This chapter is further structured as follows: Section 1.1 presents the motivation of this work, Section 1.2 states the objectives of this thesis, Section 1.3 presents the approach adopted in the development of this thesis and Section 1.4 outlines the structure of this thesis by presenting an overview of the chapters.

1.1. Motivation

People perform tasks every day which are actions to accomplish their goals, for instance catching a train: getting to and through the train station to the correct platform in time. These tasks are performed in the real world and the current status of the real world will have an influence on the way people perform their tasks. It provides a dynamic context to the tasks and as the context changes, so will the actions.

As an aid in performing their tasks, people have been using personal computers for years. Instead of using a timetable to find out when the next train is leaving and from what platform, they now use a program or website that gives them this information. Personal computing has come a long way since its start: with mobile devices no longer being dedicated devices, but rather general purpose devices, this enables people to use computers as an aid anywhere they go.

Consequently, some tasks can now be done anywhere and thus the applications can be ran anywhere. The context changes, and with it the tasks change, but traditional applications generate output based on explicit input, not based on the changing context. *Context-aware computing* allows for this adaptation to the user's context to happen. Context in this case is defined as 'information to characterize the situation of the user" [1]. Within computing systems, this context is represented by *context information*.

Context-awareness can form a bridge between the real world in which the user performs his tasks and the virtual world where the applications exist. For example, based on the user's context, such as his location and travel information, a (mobile) application can show the user a map with the route towards the platform where his train leaves and the time he has got left.

Although this aids the user, it presents information about the *real* world in the *virtual* world. A different approach is to combine these two into *augmented reality*, as Milgram [2] did in his virtuality continuum (see Figure 1.1). Using augmented reality (AR), a layer with virtual objects presenting information is laid on top of footage of the real world in real time, enhancing the user's perception of the real world. For example, instead of showing a map with the route, a user can see the path he needs to walk in his view of the train station.

Figure 1.1.: Milgram's reality-virtuality continuum [2]

The application supporting a traveller in a train station is just one example, but a wide range of applications is possible, as shown by various researches [3, 4, 5]. Possible types of applications include the following:

- Navigation: guide users through an environment using path-lines;
- Medical: a visualization and training aid for surgery;
- Manufacturing and repair: instruction for assembly, maintenance, and repair of complex machinery;
- Annotation and visualization: show public or private information for objects and environments; and,
- Entertainment: virtual games in a real world setting.

Even though the domains of these applications might be totally different, there is a lot of common functionality amongst them and this amplifies the need for reusability. To facilitate this, and along with it a decrease in development time and cost, frameworks provide this reusability in two ways: *code reuse* and *design reuse*. Firstly, by code reuse, regularly reoccurring functionality is put in the framework layer and thus becomes available to every application that is supported by the framework, and secondly, by design reuse, the interface to the framework's functionality structures the way the application uses the framework and thus adds design structure to the application.

As identified in [6], a large number of context-awareness frameworks and middleware systems exist. Although these provide part of the functionality needed in a context-aware augmented reality system, they provide no support for augmented reality and some are not specifically designed for mobile applications.

Examples of existing frameworks for augmented reality are Studierstube by the Graz University of Technology and Vienna University of Technology [7], DWARF by the Technische Universität München [8], MARA by Nokia [9], ARToolkit by ARToolworks [10], the general purpose framework by the University of Michigan [11] and the Mobile Reality framework by Goose and Schneider [12]. These frameworks, however, are not capable of providing the functionality needed to realise these context-aware augmented reality mobile applications: either they need markers placed around the environment, are based on localization techniques that are limited to outdoor environments, provide too inaccurate results, use obsolete technologies or they have not been designed to run on mobile devices. Ideally, a framework for context-aware augmented reality in mobile applications would have a generic design, thus providing a reusable base for the rapid development of a wide range of applications, including those unanticipated during framework development.

1.2. Objectives

Based on the motivation above, the objectives for this thesis are:

1. to identify what techniques need to be combined to facilitate context-aware augmented reality in mobile applications.
2. to determine the capabilities and limitations of context-aware augmented reality in mobile applications using currently available technologies.

1.3. Approach

In order to reach our objectives, we have taken the following steps:

1. Investigate techniques for augmented reality, context-awareness and indoor localization in order to gain sufficient knowledge to be able to specify the user and system requirements of mobile applications with context-aware augmented reality;
2. Define the user and application requirements, based on the research and the identified stakeholders and their goals, and derive of architectural elements from those requirements;
3. Integrate the architectural elements into a system design, i.e., the framework architecture;

4. Implement the framework architecture, that supports the desired functionality; and,
5. Design and implement a proof of concept mobile application supported by the framework.

1.4. Structure of this report

The structure of this thesis reflects the order in which these issues have been dealt with throughout the research process. This thesis is structured as follows:

- Chapter 2 gives the background of the research, including concepts in context-awareness, augmented reality and indoor localization techniques, along with known context-awareness and augmented reality mobile applications and frameworks. This chapter is necessary to understand what kind of applications the framework should support, and what features make those applications context-aware and/or provide augmented reality;
- Chapter 3 describes the requirements of the framework, in order for it to support context-awareness augmented reality applications. These requirements are based on the background research and on the stakeholders identified for the applications and their goals. From the requirements, various architectural elements can be derived;
- Chapter 4 describes the design of the framework architecture proposed by this work. The design incorporates the architectural elements derived from the requirements, and was used to provide guidelines for the implementation process;
- Chapter 5 presents the software design of the framework, and discusses how the software packages and classes implement the framework architecture design;
- Chapter 6 reports on the implementation process of the proposed framework architecture;
- Chapter 7 describes the proof of concept application that has been built to demonstrate and validate the framework functionality; and,
- Finally, Chapter 8 presents our final conclusions and important remarks, and identifies possible future work.

Chapter 2.

Background

This chapter reports on the conducted literature study, presenting basic concepts on context-awareness, augmented reality and localization.

Section 2.1 discusses context and context-awareness, Section 2.2 discusses augmented reality, and, Section 2.3 presents various localization methods.

2.1. Context-awareness

This section elaborates on the concept of *context-awareness*, by identifying the precise meaning of context, context information and context-awareness, along with the features that make an application context-aware and how these features could be provided.

2.1.1. Context

Some studies have been done in the field of context-awareness and therefore a number of different definitions of *context* exist. Context can be described as the environment of a user or device. Within context-aware applications, this context is represented by *context information*. We have based our definition of context information on the definition of context in [1]. Our definition of context information is:

> *Context information 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 application themselves.*

This definition is chosen as it is generic of nature: one cannot enumerate which aspects of all situations are important as these may change for each specific application. If one takes for example a user with a shopping application displaying prices of products in the neighbourhood, information about the presence of other people will not have any influence on the user or the application and thus is not considered to be context

information in this particular case. Information about the location, however, will change which shops are displayed, and the currency of the prices that are shown. Since this influences the application behaviour, the user's location is considered to be context information in this case. Context information is acquired by using physical sensors which measure certain values, such as the user's location or the temperature of the user's physical environment. This process is known as *context sensing*.

2.1.2. Context categorization

A categorization of context information types should help application designers establish what context information is useful for their applications. Based on earlier research [13, 14, 15], four categories of context information have been distinguished:

- Location: *where is the user?*
- Identity: *who is the user?*
- Activity: *what is the user doing?*
- Time: *when is the user doing it?*

These four categories are the *primary* context information types to characterize an entity's situation. As well as being contextual information, the primary pieces of context information for one entity also act as indices to find *secondary* context information. These secondary pieces of context information share a common characteristic: they can be indexed by the primary context information because they are attributes of the entity with primary context. For example, a user's phone number (secondary context information) can be obtained by using the user's identity (primary context information) as an index for an information space like a phone book.

2.1.3. Context-awareness

In accordance with the definition of context information, we also base our definition of context-awareness on the definition in [15]. Our definition of context-awareness is:

> *A system is context-aware if it uses context information to provide or adjust relevant services to the user, where relevancy depends on the user's task.*

According to our definition, an application is context-aware if the services that the application offers are adapted to the context by using context information. This could be the provisioning of new services or adjusting its existing services, which could be as simple as displaying the changed context information. Instead of a requirement to detect, interpret and respond to context, this definition only requires the response to context, by adjusting the offered services, allowing detection and interpretation to be performed by other computing entities.

2.1.4. Applications

Dey [15] identifies three categories of features a context-aware application may support:

- presentation of information and services to the user;
- automatic execution of a service; and
- tagging of context to information for later retrieval.

Buchholz [16] introduced a "value chain" of context-awareness, presenting the different steps needed to provide one or more of the features mentioned above. Figure 2.1 shows this value chain.

Figure 2.1.: Context-awareness value chain [16]

The first step in this process is context sensing. In this part of the process, sensors perform measurements in the environment to acquire *low level* context information.

Context refinement is the second step, consisting of a possibly repeated refinement process to turn the low level context information into *higher-level* context information. This process consists of two subprocesses: interpretation and aggregation. An example of interpretation could be the conversion of a user's location to a room or street name. Aggregation combines context information of different sources, for instance, establishing the user's activity from his location, agenda and the current time.

The final step, context dissemination, is concerned with the storage and spread of the (higher-level) context information. This can either be done using a *push*-approach in which the context information is sent back to the user or a *pull*-approach in which the user retrieves the context information when needed.

In order to provide context information to context-aware applications, Buchholz [16] suggests the use of a Context Information Service (CIS). A CIS provides context information based on sensor measurements from different sources, including the mobile devices requesting the service. Based on these measurements, it performs context refinement and dissemination, providing higher-level context information back to mobile devices. Figure 2.1 shows the part of the value chain which is handled by a CIS.

The separation of "context information gathering" from "context information usage" is supported by various other developments [17, 18, 19, 20, 21], in which attempts have been made to standardize context representation and context information provisioning.

The definition and categorization of context information and context-awareness determines what such context information standards should support, and what such context information provisioning systems and context-aware applications should look like, in order to be context-aware.

Surveys of context-awareness middleware [6, 22, 23] have shown that a large number of systems provide support for context-awareness, with two main categories: those based on RMI and CORBA technologies and those based on web services (more on these technologies in [24]). The former are based on strongly coupled components and use strictly defined data structures, communication protocols and interfaces, whereas the latter uses standard HTTP connections and textual data representations, based on XML. The web services approach has the advantage of being faster for single, small data requests, next to the loose coupling between client and server, while the RMI and CORBA approach is faster for larger responses [24].

2.2. Augmented Reality

This section discusses augmented reality, covering marker-based and markerless augmented reality and mobile augmented reality.

2.2.1. Requirements

According to Azuma [3], there are three requirements an *augmented reality* (AR) system needs to fulfil:

- **Combine reality and virtuality**: This requirement is fundamental to AR systems, since their purpose is to support the reality with virtual content. As Milgram [2] describes in his virtuality continuum (See Figure 1.1 in Section 1.1), AR is a manifestation of "mixed reality" with a tendency towards reality: AR is mostly based on the reality with mixed-in virtual objects.
- **Interact in real time**: This implies that an AR system should react to the user in real time, unlike, for instance, computer graphics in movies. For example, a change of viewpoint of the user should be reflected in a change of displayed information.
- **Register in 3D**: Virtual objects are registered in the real world in 3D, i.e. virtual content is placed in the real world environment using a three-axis coordinate (x,y,z) in three-dimensional space, reflecting its latitude, longitude and altitude.

2.2.2. Purpose and issues

According to Vallino [25], the goal of augmented reality systems is to combine the interactive real world with an interactive computer-generated world in such a way that they appear as one. This statement however ignores that, apart from presenting a visualization of the augmented world, AR also provides an interface to applications. Wagner [26] point this out and states that AR research should aim at supporting human computer interfaces.

Schmalstieg [27] states that augmentation can be seen as a tool, not a final goal; the actual goal for the user is to enhance perception of and interaction with the real world. Neither [25, 27] or [3] mentioned earlier give a clear definition of AR, as also stated in [26]. In order to provide such a definition, we identified the most important concepts from literature. For the definition to be generally applicable, it should be valid for both AR systems using the real world (i.e., see-through glasses, known as optical AR) as well as systems using footage of the real world (known as video AR). Hence, we chose for the term "view of reality". In order to generalize the presented virtual information we chose the term "virtual content". Our definition of augmented reality is:

> *Augmented Reality combines a view of reality with virtual content in real time in order to provide an interface to enhance perception of and interaction with the real world.*

There are two important issues regarding the correct working of AR systems: 1. getting the position and orientation and 2. aligning reality and virtual content. These are known as *tracking* and *registration* respectively [3].

- **Tracking**: the process of measuring the location and direction of the device or user. In order for the AR application to function properly, this process needs to be done as accurately and robustly as possible and in real time. The process responsible for the tracking measurements and calculations is called the *tracker*.
- **Registration**: the process of keeping reality and the virtual content aligned. If this process fails, the user's experience of one mixed reality is compromised. Registration accuracy depends on the accuracy of the tracking process. Therefore, a highly accurate tracker is necessary for the proper functioning of the AR application.

2.2.3. Approaches

Marker-based AR

The first augmented reality applications, like those based on AR frameworks such as ARToolkit [10] and StudierStube [7], use *fiduciary markers*, which are specific objects

in the field of view that are used as points of reference. Figure 2.2 shows a step-by-step approach to the usage of these markers. The software recognises the markers in a video feed and calculates their position and orientation from it. Using this data as a reference, it places virtual content with the same orientation into the feed. The virtual content's position and orientation updates accordingly as the camera position or marker position changes.

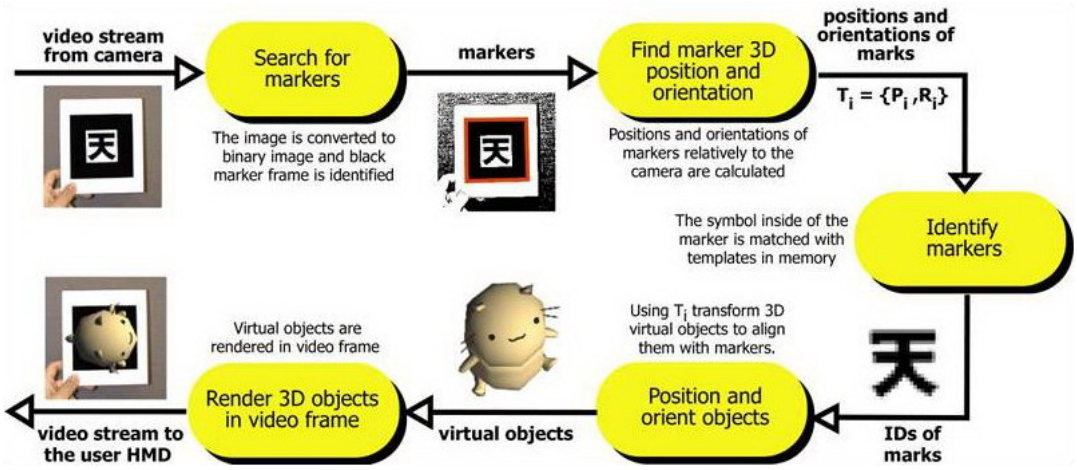

Figure 2.2.: Marker-based AR [28]

A large number of AR applications with marked-based trackers are spawning from research at institutes and universities [7, 8, 9, 10, 11], along with a few commercial applications. For example, GE [29] and Topps [30] have developed marketing applications using marker-based AR. Figure 2.3 shows the baseball cards of Topps on which a 3D representation of the player is projected using an AR application. In 2007, Sony launched a game called the Eye of Judgement [31] which uses cards with marks and a camera connected to the Playstation 3 game-console, which was the first internationally released, commercial marker-based AR application.

Markerless AR

In order to do away with markers, two different types of solutions have been developed that use other tracking techniques, namely *pose tracking* and *pattern matching*. Pose tracking can be used when the device containing the camera and screen is moving and observing a static environment, whereas in the pattern matching approach, patterns inside a regular picture are used to detect the position and orientation of the object containing the picture, with regard to a static displaying device.

Pose Tracking

By obtaining the location and orientation of the user or device through measurements, the system can determine the field of view (FOV). Figure 2.4 shows a graphic representation of pose tracking.

Figure 2.3.: Marker-based AR application from Topps 3D [30]

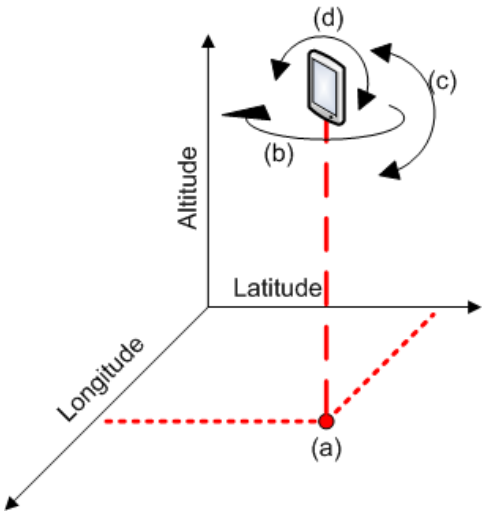

Figure 2.4.: Pose tracking using (a) location, (b) direction and (c) tilt

In order to determine the pose four parameters are needed: (a) the location, (b) the rotation or direction of the device, and (c) the tilt and roll (d) of the device. These parameters can be acquired by a localization technique (see Section 2.3), a digital compass and an orientation sensor, respectively. The advantage of this approach is that the surroundings do not have to be adapted with markers or patterns, and therefore, it is independent of the surroundings, but the device must be capable of acquiring the sensory data either from internal of external sensors.

The Tinmith Project [32] is a AR architecture developed by the Wearable Computer Lab at the University of South Australia and uses pose tracking for a user wearing an HMD and gloves with markers to be able to interact with virtual objects using gestures.

Pattern matching

The pattern matching approach is similar to the marker-based approach, but markers are replaced by less obvious patterns in regular pictures. This solution is therefore more suitable to be used in commercial applications as an addition to traditional media such as magazines, flyers or product packaging. Examples are the marketing applications of Lego [33] and Toyota [34], which have been developed using pattern matching-based AR. These are shown in Figures 2.5 and 2.6.

Figure 2.5.: Lego Digital Box [35]

Mobile applications

Traditional approaches to the realization of portable AR systems use a setup consisting of a notebook combined with a head-mounted display (HMD), represented by (a) in Figure 2.7. Handheld AR is represented by (b), (c) and (d) in Figure 2.7. The Tablet PC in (b) has a large touch screen, but no internal camera and its weight makes it too heavy to hold in one hand. The smartphone in (c) is like the traditional PDA with the incorporated functionality of the mobile phone (d), and has the added value of a

Figure 2.6.: Toyota AR ad [34]

larger (touch) screen and more processing power with regard to the mobile phone. Since smartphones are nowadays owned by a large number of people, their weight makes them easy to carry and work with, and the capabilities are suitable for a wide variety of AR applications, this category is most suitable to deploy mobile AR applications [26].

Wikitude [36] is the first released mobile AR application that uses pose tracking, and thus it is fully markerless and patternless. Based on pose tracking, the user's field of view is determined and this enables the application to show information about points of interest when the user is looking at them. Figure 2.8 shows the application on the Google Android platform.

2.3. Localization

This section describes the concept of location estimation and the measurement parameters, technologies and localization techniques used in the localization process. Services and applications based on location information are also described.

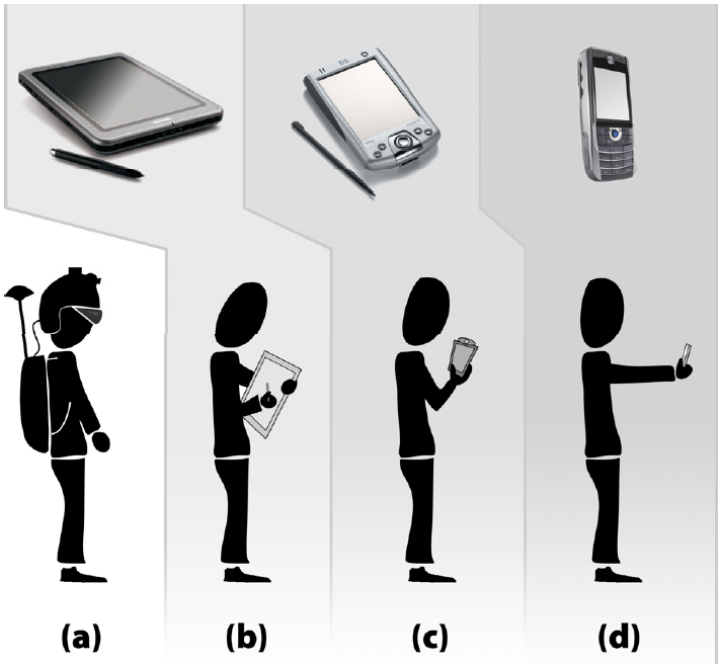

Figure 2.7.: Different types of mobile AR applications [26]

2.3.1. Location estimation

The process of location estimation, or localization, is used to determine the location of a mobile device. Depending on the platform on which the localization process has to take place (mobile phone, laptop, etc.), the available technologies and calculation capabilities might be limited. Most localization techniques use radio-frequency (RF) based technologies, but other techniques are possible, as described in Section 2.3.3. Most communication networks nowadays use RF, and thus, localization based on these network technologies requires no or little extra investments in hardware, lowering deployment costs. Measurement parameters vary, as does the possible range of use, but overall, these techniques present at least a reasonable accuracy at low costs. Besides RF, other wireless networking technologies exist, but are unsuitable for localization using currently available mobile devices: infrared (IR) has an overall low accuracy and lack of robustness, and ultra-wideband (UWB), although proven to have an accuracy of a few centimetres within a range of 100 metre indoors [37], requires nanosecond clock precision, which is currently only available in specialised equipment.

2.3.2. Measurement parameters

Location estimation is performed by measuring a location dependent parameter of the link between the user and/or his mobile device (referred to as the “node”) and some static network entities with known locations (referred to as the “base stations”). Using

Figure 2.8.: Wikitude [36]

calculations or reference information, the location of the node can be determined, either in the node or in the base station.

Time (difference) of arrival

The delay between sending a wireless signal from the base station and receiving it at a certain distance by the node is known as the time of arrival (ToA) or time of flight (ToF) and depends on the distance and the speed of the wireless signal (a known, constant value). From the delay between the node and the base station the distance can be calculated. Thus, for each base station, the node is known to be somewhere on a circle with that distance to the base station. By intersecting at least three of these circles, the location can be estimated [38]. Figure 2.9 illustrates this method, which is called *trilateration*.

For time difference of arrival (TDOA), the method uses time difference measurements rather than absolute time measurements as ToA does. The time difference is measured by comparing the ToAs of a signal from the node at two different base stations. Instead of the circles of the ToA approach, the TDOAs form a hyperbole, therefore the method is also known as the hyperbolic system. Two of the hyperbolas are intersected to determine the position. This is called *multilateration*.

Signal strength

The (average) level of the received signal from a base station, as measured at a certain location, is known as the signal strength (SS). These measurements can be used to

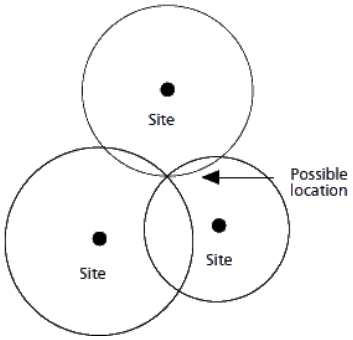

Figure 2.9.: Trilateration

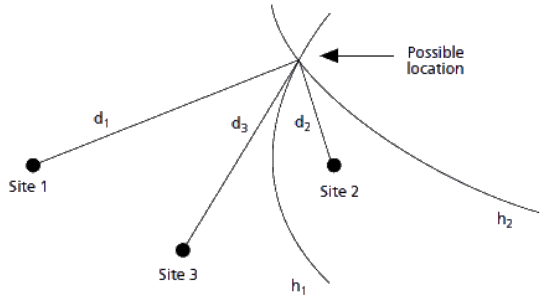

Figure 2.10.: Multilateration

Trilateration and multilateration [39]

determine the location by either comparing it to reference information, i.e., the fingerprinting approach, or using a model to calculate the distance to a base station, i.e., the trilateration approach.

Connectivity

Connectivity is a boolean value indicating whether a node is within the transmission range of a base station. Location can be estimated to regions where the transmission boundaries of certain base stations overlap. This requires a small computational cost, but provides a low accuracy, especially when base stations are thinly spread.

Angle of arrival

Using a directional antenna on base stations, by measuring the angles at which signals are received at a number of base stations, the node's location can be estimated. This approach is known as *triangulation* and does have some downsides: it requires base stations with a directional antenna and received angles are influenced by *multipaths* (additional received signals due to reflections), making it less suitable for indoor environments.

2.3.3. Localization techniques

Two areas of deployment can be identified, namely wide area and local area deployment. Within each of these two areas, given the physical layer and measurement parameters, several wireless network technologies can be used for location determination. Figure 2.11 depicts these technologies, and their appropriate areas of deployment.

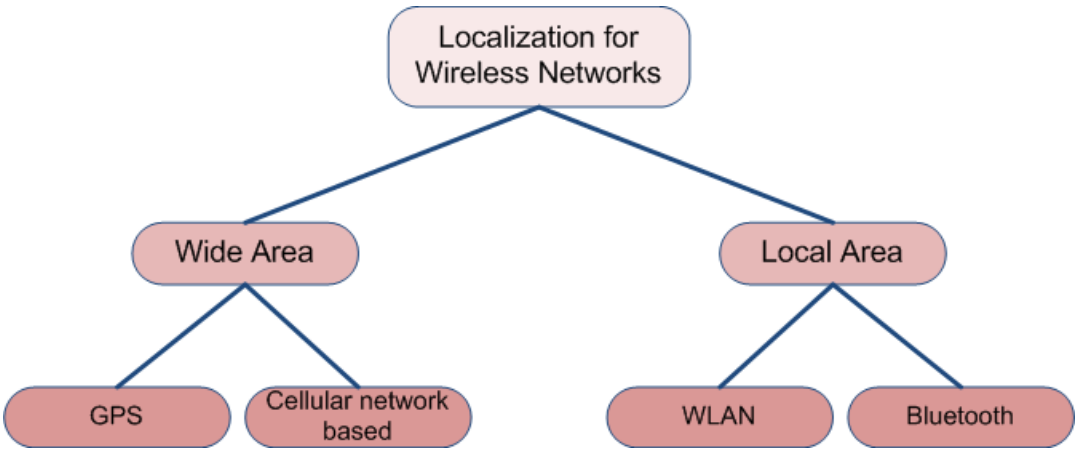

Figure 2.11.: Different localization technologies, categorized by deployment area

Global Positioning System (GPS)

This localization technique is based on satellite-communication used worldwide. A node uses multiple (at least four for a 3D location) of 21 satellites with a known location and synchronized clocks. The node measures the ToA of RF signals from different satellites to calculate the distance to them and uses *trilateration* to determine its own location with an accuracy of about 6-12 metres. ToA solves the multipath signal problems by using the firstly received signal, which in normal circumstances is the direct path. In cluttered or indoor environments this poses a problem when no direct path is possible, known as non-line-of-sight (NLOS).

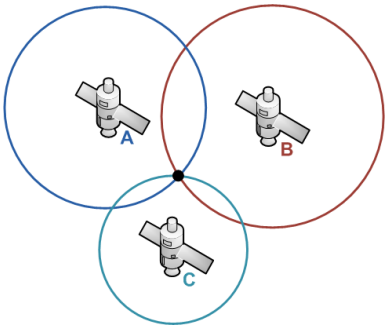

Figure 2.12.: GPS

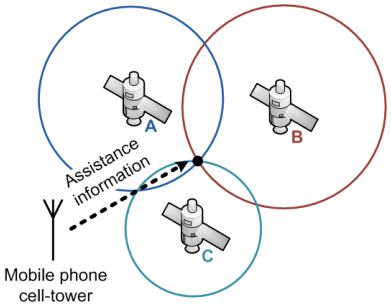

Figure 2.13.: Assisted GPS

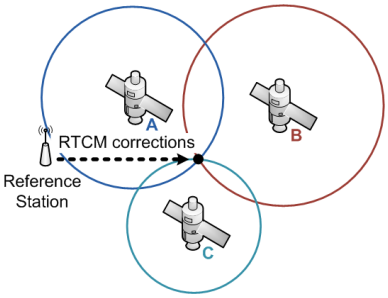

Figure 2.14.: Differential GPS

Different GPS-techniques

In order to improve the results of GPS localization, some extensions have been made, such as AGPS and DGPS. Figures 2.12, 2.13 and 2.14 schematically depict respectively GPS, AGPS and DGPS.

To enhance the startup performance of GPS on mobile phones, Assisted GPS (AGPS) can be used. This is useful in situations where signal conditions are poor, for instance, when high buildings in the surroundings create multipaths, or when indoor usage create NLOS problems. Using resources of the cellular network provides additional information to more quickly acquire satellites or do remote calculations. More information about AGPS can be found in [40, 41, 39].

When a higher accuracy is required, differential GPS (DGPS) can increase the accuracy by using an extra GPS receiver with a precisely known location, referred to as the reference station. By calculating the location at the reference station using GPS and comparing the result with the known location, the established difference can be applied to the GPS-position of the node to differentially correct its position. More information about DGPS can be found in [42].

Cellular-based systems

Although the number of cellphones with integrated GPS capability is increasing, the majority of the telephones lack GPS and to provide wide area localization on these phones, cellular-based systems can be used. In comparison with GPS, these systems have an added advantage of being able to work indoor properly, but on average, they have a lower accuracy than GPS in outdoor situations.

Within cellular-based systems, three categories of localization techniques can be distinguished [43]:

- **Mobile-based**: localization is carried out in the mobile device and if necessary sent back to the network.
- **Mobile-assisted**: measurements are performed in the mobile device, measurement results are sent to the network where a server calculates the location, which could be sent back to the device or further processed in the network.
- **Network-based**: localization is entirely done by the network.

Although cellular-based systems have the advantage of requiring no additional hardware in the mobile phone, with the provided localization accuracy of 50m and upwards [44], these techniques are unsuitable for usage in situations that require a higher accuracy.

Wireless LAN

Wireless LAN (Wireless Local Area Network, WLAN for short) is a widely used wireless network infrastructure based on RF signals. On top of this architecture computer communication is being carried out according to the 802.11-standard, which has been defined by the IEEE, a standardization organization. Most implementations support

Standard	Frequency	Max Bitrate	Indoor range (est)
802.11a	5GHz	54 Mbit/s	35m
802.11b	2.4GHz	11 Mbit/s	30m
802.11g	2.4GHz	54 Mbit/s	100m

Table 2.1.: Comparison of various 802.11-standards [45]

the 802.11a, 802.11b and 802.11g versions of this standard, which are compared in Table 2.1.

Commonly, two techniques are more suitable for WLAN-based localization, namely trilateration and fingerprinting. Although a triangulation is possible, this requires specialised hardware. Based on the range of the base stations, for WLAN known as access points (APs), WLAN-networks are most frequently used for local communication networks inside and around (office) buildings, but more widely deployed, the technique can be used in larger areas, such as university campuses or downtown urban areas.

Trilateration techniques
The trilateration approach, described earlier in Section 2.3.2 and Figure 2.9, uses measurements to determine the distance from the node to three or more base stations (APs) with a known location. As the measurement parameter, the received signal strength (RSS) is used, which can be converted into the AP-node distance using a signal propagation model. The simplest way of finding this propagation model, and thus the relation between the received strength and the distance, is to do reference measurements at points with a known location.

Since RF signal propagation is complex, the general empirical model, which is created using the reference measurements, only provides results with low accuracy. This complexity comes from signal attenuation due to distance, penetration losses through walls and floors, multipaths and interference from other signals. The orientation of the user, and, therefore, the orientation of the antenna, also influences the RSS, as does the presence and movement of people in the environment. In research, techniques are presented which improve accuracy by combining multiple algorithms and multiple sources [38] or combining it with coarse localization based on fingerprinting [46].

Fingerprinting techniques
The fingerprinting approach consists of two phases: the “offline” or “training” phase and the “online” phase in which the actual localization takes place. In the offline phase, the fingerprint database is built. At selected reference points (RPs), measurements for the RSS of all the APs are performed, establishing the characteristic feature (fingerprint) for that RP. This fingerprint is stored in the database and the procedure is repeated for all RPs. In the online phase, the node measures the RSS at its current location. These measurements are compared with the data in the database using an appropriate algorithm, resulting in the likeliest location of the node. Figure 2.15 illustrates the two phases of the fingerprinting approach.

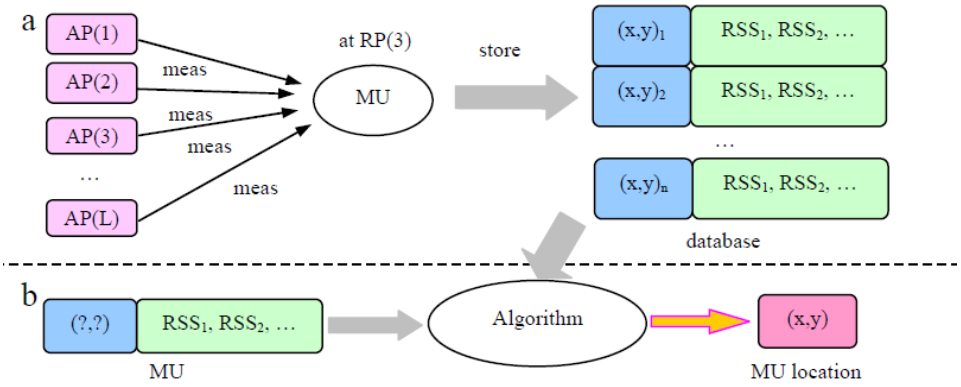

Figure 2.15.: Fingerprinting phases: (a) offline and (b) online [46]

Fingerprinting can provide a high accuracy, but has the offline phase as a downside: due to signal variations, a high number of measurements is needed at each RP, increasing the time and effort to be spent during the offline phase. Fingerprinting can be performed using either a *deterministic* or a *probabilistic* method.

Deterministic fingerprinting

The deterministic method for fingerprinting represents the SS of an AP at an RP by a scalar value, such as, for instance the mean value. Therefore, the fingerprint database structure is relatively simple, consisting of the average RSSs for all APs at each RP. To establish the location from the measurements during the online phase, many algorithms can be used. For instance, nearest neighbour (NN) determines the location by finding the RP with its SS-vector closest to the measured SS-vector. Other algorithms include K nearest neighbour [47] and smallest polygon [48].

In order to reduce the time necessary for the offline phase, a denser database can be generated by interpolation based on adjacent RPs using methods such as inverse distance weighting (IDW) and universal kriging (UK) [46].

Examples of systems that use deterministic fingerprinting are RADAR [49, 50], Aura (CMU-TMI) [51] and the technique presented in [52].

Probabilistic fingerprinting

The probabilistic approach uses the fact that more parameters than just the average characterize the SS at a certain location. The probabilistic fingerprinting approach to WLAN localization is a technique that accommodates this [53, 54].

Measurements of SSs at a location form a distribution, which is unique (statistically different) for that location, and thus, localization can be performed based on the distributions at different locations. These distributions, known as probability distribution functions (pdf), are stored for each RP. During the online phase, the current distribution is measured and probabilistic techniques are used to compare it with the stored pdf's and estimate the most likely location [55].

In addition to the high accuracy compared to other techniques, the probabilistic techniques provides ways to incorporate additional information in the calculation, such as, measurements from other techniques such as Bluetooth, previous locations estimates or motion sensor measurements.

Because of the improvements compared to the deterministic approach, a larger number of systems use the probabilistic approach, like Nibble [56], Horus [55] and the systems described in [53, 57, 54, 58].

Commercially available systems
A number of commercial applications of WLAN-localization methods are available today. The Ekahau Positioning Engine (EPE) [59] provides localization and tracking, along with software to accommodate deployment in the offline phase and hardware tags. PlaceEngine [60] is an open system, allowing users to include the localization capabilities in their own applications. Radiomaps are built and improved by the users themselves by running the PlaceEngine Client. Skyhook [61] is also an open system that combines localization functionality of GPS, cellular-based and WLAN-based localization. Zebra Enterprise Solutions [62] (formerly known as "WhereNet") provides location tracking software.

Bluetooth

Research has shown [63, 48, 38] that Bluetooth is not suitable for localization with high accuracy due to high fluctuations in the link quality measurements. Other parameters than the received signal strength have also proven to be unsuitable: the transmit power level takes long to stabilize, the link quality perceived at any location is rather sensitive to the transmitter's Bluetooth class, making it unsuitable as an overall solution, and although the received power level (RX) has a very high correlation with the distance, it is not available due to the Bluetooth specification.

Bluetooth could however be used in connectivity-based systems for coarse localization, especially when a high number of static base stations is being placed around the area. The advantage of using Bluetooth for localization is that many cellular phones include a Bluetooth-adapter.

Other technologies

Localization without wireless networks is possible, for instance, by using visual tracking or using markers, as described in Section 2.2.3. Using a visual tracking approach [64], localization could be done based on visual information. Photos of the surroundings are stored and compared with real-time pictures or video. This method however is not robust against a changing environment and would therefore need continuous recalibration.

A multitude of strategically placed markers could be spread around the localization area, and could be used to determine the location of the camera and thus the user. The need for a high resolution camera and the need for markers everywhere in the area are the main drawbacks of this approach.

2.3.4. Location-based services

A special kind of applications that offer functionality based on the location of a user or device are known as *location-based services* (LBS). Virrantaus [65] defines LBS as:

> *Information services accessible with mobile devices through the mobile network and utilizing the ability to make use of the location of the mobile device.*

Hence, LBS are context-aware services that solely use the user's location as context information. Functionality is offered to the user through his mobile device, which is located either by the mobile device or by the network. As stated in [66], due to increased performance and decrease in cost of smartphones, along with a decrease in mobile internet cost, LBS availability and usage is rising as the first mainstream type of context-aware applications. Mobile phone network operators implement localization methods (in Europe as obliged by the European Union to locate mobile emergency callers [67]), facilitating the development of LBS.

Based on the location, a service is provided in one of three forms [68]:

- **Push**: the service provider sends the information to the mobile device. For example, an automatic weather forecast could be sent every morning based on the user's location;
- **Pull**: the user places a request for an LBS. For instance, a user requests a local weather forecast when needed; and,
- **Track**: the user requests the location of a mobile device. For instance, a buddy finder finds out where other people are located.

Within these three types, some categories of applications are possible, as Figure 2.16 shows. This figure does not claim to be complete and the number of application categories is growing over time, but the figure does give an indication of what types of services can be provided only with a user's location.

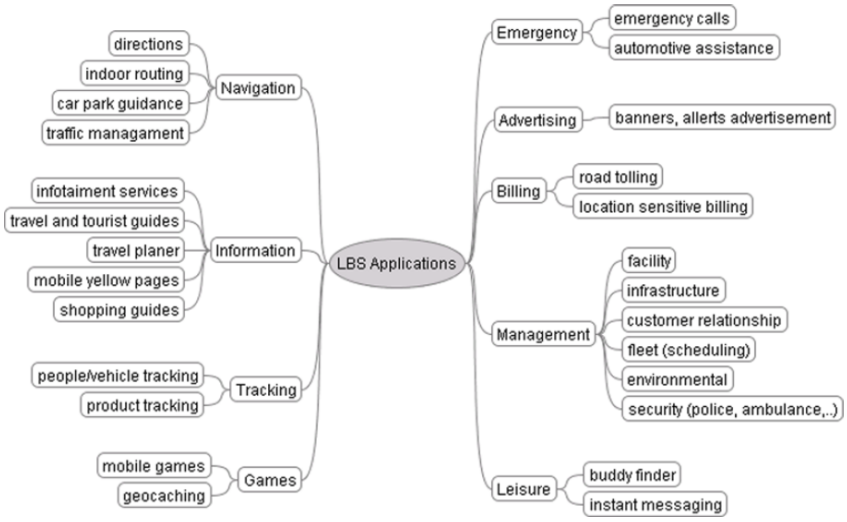

Figure 2.16.: LBS application categories [69]

Chapter 3.

Requirements

In this chapter, the requirements for the generic framework are presented. These requirements have been based on the requirements for a specific proof-of-concept application, which in turn have been extracted from use case scenarios. These scenarios describe the usage of a train station navigation application. This application, and other context-aware augmented reality applications, can be supported by the framework.

Section 3.1 presents the use case scenarios for the proof-of-concept application. Section 3.2 discusses the stakeholders of the application, and of the framework that supports this application. Section 3.3 gives the generic requirements for the context-aware augmented reality applications, and Section 3.4 gives the requirements for the framework.

3.1. Scenarios

Regarding the functionality of the context-aware augmented reality in mobile applications, a number of use case scenarios have been identified. These are identified based on a train station navigation application, which has been developed as a proof-of-concept, supported by the framework. The main goal of the mobile application is to assist a user travelling through a train station, providing information and guidance using context-aware augmented reality. Consider the following examples:

1. Alice searches a ticketing machine to buy her train ticket.
2. Bob is in the train station and wants more information about an interesting ad.
3. Charlie wants to check the map for an overview of the train station.
4. Dean wants to get to the train in time.
5. Eric sets what information will be shown.
6. Fred adds new information on the train station to the application.

Use case scenario #1: *Buying a train ticket*
Alice wants to buy a train ticket and consequently searches a ticket machine. She presses the "search"-button and chooses for ticket machines. An arrow displays the direction to the nearest ticket machine. She chooses to be guided through the station and a guiding-line shows the path to the machine. Unfortunately, the machine just got broken, and therefore, Alice presses the "find alternatives"-button to be guided to the nearest alternative ticketing machine.

Use case scenario #2: *Getting information*
Bob sees an interesting ad for a new movie. He points his mobile device to the banner and the application shows a label, identifying the ad for the movie. He presses on the label and a small window opens, showing some more detailed information, such as a small plot summary, ratings and the length of the movie. He presses the "ratings"-item, and some user comments are shown. He feels confident it is a good movie, and so he presses the "buy tickets"-button to go to a website that handles the ticket sale.

Use case scenario #3: *Checking the map*
When Charlie wants to see an overview of the train station, he changes his device to the map-view. This could be done by pressing the "map"-button, but also by pointing the mobile device downwards, hence with the camera pointing towards the ground. A zoomable, scrollable map is shown, displaying his current location and possibly his destination and the route towards it. Items are shown on the map, which Charlie could click on to see more information about the item or to set it as the new destination.

Use case scenario #4: *Getting to the train in time*
Dean enters the station to catch a train. He uses his mobile device to get information about various items in the train station, including information points, platforms, ticket machines, stores, etc. A notification comes in, indicating that the train he wanted to take is delayed. During spare time in previous visits, he got a cup of coffee and browsed the bookstore. Therefore, tips and offers for these stores show up on his device when he looks around the station with his mobile device. After spending some time in the bookstore, he gets a reminder to get to his train. Looking around the station now, information not related to the trip is left out.

Use case scenario #5: *Setting shown information types*
Train stations could contain a large number of information items, and thus, Eric specifies what information types he wants to see. Since he is not interested in food and drinks, he disables the displaying of information or offers for this type of items. He is however interested in books and information about service disruptions, so he sets that information to be displayed by default.

Use case scenario #6: *Adding information*
A new ticket dispenser has been added to the station and Fred needs to put it in the system. He logs in to the maintenance application using his personal credentials, proving he is authorized to make changes to the system, and enters a new item to the data on the respective location of the type "ticket dispenser". He also adds an item for a new store

that has been opened, filling in the properties of the store and a source where dynamic information can be found, such as the current offers.

3.2. Stakeholders

There are two categories of stakeholders that have been identified: those for the specific applications, and those for the generic framework. The stakeholders for each category are discussed in the two following sections.

3.2.1. Application stakeholders

Users
Since they potentially use the application, the users can directly benefit from it. They will use the system in ways described in the use case scenarios: moving around the train station, viewing information, receiving notifications, etc. They will provide the input the application needs, and allow it to gather context information about them, while the application will manage the gathered information securely and privately.

Context providers
The task of context providers is to gather context information about the user's context and provide it to the system. Although the gathering and processing of context information could be performed on the mobile device by the application, it could also be performed by external parties. The service of providing context information, based on available information, such as the user's location, presents these external parties with a new business model. They might offer a charged service of providing up to date information to users, adapted to their context.

Content providers
Their task is to deliver content for the application, destined for the user. In case of the described scenarios, the railway company should provide all information related to trains, tickets, departure times, etc. Other content providers are, for instance, the stores within the train station, advertising companies with banners in the train station, and basically all other "external" companies exploiting a service inside the train station. Regarding the business case for the content providers, the application has the advantage of providing information to the user when they are actually looking at it. It is thus a direct way of reaching people that are interested in the offered services, products or information.

System administrators
The task of the system administrators is to make sure the information provided to the users is up to date and accurate, adding more information and information sources when necessary. Making these adjustments should be facilitated, but only after authorization to prevent abuse.

3.2.2. Framework stakeholders

Context and content providers
The fact that a wide range of applications, running on a number of different platforms, can all be supported by the same framework, provides a uniform way in which context and content providers can offer their services. Therefore, along with the application benefit, they also benefit from the framework.

System administrators
The system administrators could benefit from the framework if they are in charge of the information for multiple applications: changing the information for multiple applications can be done in a uniform way, easing off their task.

Developers
The developers have the largest benefit from the framework, since development time and effort will decrease when using the framework as a basis for their applications. The focus can be on the application-specific functionality, while generic functionality can be handled by the framework. Once the framework has been used for some time, most bugs should be fixed, decreasing the chance for errors in the applications.

3.3. Application requirements

From the use case scenarios and the different stakeholders with their respective goals, the requirements for the train station navigation application have been determined. We have grouped these into two categories, namely functional and non-functional requirements, i.e., what it should do and how it should perform.

3.3.1. Functional requirements

With respect to the users, five functional requirements have been identified:

1. Providing static and dynamic information about items in the train station.
2. Providing guidance or the direction to a destination.
3. Showing a map of the station or textual information on the train station.
4. Adjusting displayed information and providing notifications based on context.
5. Setting shown information types.

For the system administrators, two functional requirements have been identified:

6. Adding information and sources to the application.
7. Viewing logs of the system.

3.3.2. Non-functional requirements

Performance
To work properly, i.e. providing a "believable" augmented reality view to the user, the application should respond rapidly to the user's movements. Moving the mobile device around changes the view of the user and thus, the displayed information should update accordingly. A response time of one second or less should be the minimum for the application to function properly [70].

Maintainability
Changing, adding and removing information and information sources should be facilitated. While users run the applications, it should be possible for the system administrator to make changes to the information base and make it available to the users afterwards.

Security
The user's personal data, logged data and other privacy sensitive information should be kept secure. When storing data on a system other than the mobile device, it should be made anonymous, i.e., removing all personal information from the data.

Interoperability
Used information representations and communication protocols should adhere to standards that are supported by various mobile platforms.

Scalability
Each mobile device should communicate with other systems, such as those of the content providers. These systems should be able to handle a large number of devices at the same time. For example, this could be done by optimizing the processes on these systems or balancing the load between multiple systems. All relevant information about a train station could also be provided to a mobile device directly, allowing it to use this information independent of other systems. This way, all potential users in a train station can be supported.

Usability
It should be easy for the users to install and use the application. Menus and functions should be clear, easy to find and either self-explanatory or well-documented.

3.4. Framework requirements

Based on the specific requirements for the application, requirements have been determined for the generic framework. Again, these have been grouped into functional and non-functional requirements.

3.4.1. Functional requirements

The main goal of the framework is to support a wide range of context-aware augmented reality applications. In order to achieve this, it should support the functionality that is common amongst these applications. This functionality is specified by a number of generic application functions, which have been deducted from the specific proof-of-concept application functions that have been described in Sections 3.1 and 3.3.1.

Figure 3.1 illustrates the seven generic functions (five user requirements and two system administrator requirements) that could be part of context-aware augmented reality applications. These functions should be supported by the framework. We will discuss these functions briefly:

User functions:

- **View information about item:** this includes the most elementary function of augmented reality: provide information on what item the user is looking at. An item can be described as an entity, at a specific location, to which information is bound. Information could be bound to the item directly, or be provided by linking to a source of information. This information could include information about actions that can be invoked. An example of this is given in use case scenario 2, where the user is presented with an option to buy movie tickets at a website, based on looking at a movie poster item.
- **Find destination:** a user can specify an item to go to, either directly, by specifying the type of item he is searching for, or indirectly, deducted from other information (such as, for example, the departure platform, that is deducted from his travel information). For a destination, the direction towards it, or the route to it, can be shown.
- **View information using other display types:** Besides the augmented reality view, the framework should support other views. In this way, when the augmented reality view would provide no useful result, for instance, when the user is pointing the device downward, information could be shown otherwise, for example in a map view or using text.
- **Specify displayed information types:** in order to control the amount of information shown to the user, the framework should support the enabling and disabling of types of information to be shown by the user.
- **Get information adapted to the context:** to make the framework context-aware, it should be possible to adapt the information provided to the user. More of less information could be shown, based on notifications or context information that have been received. It should also be possible to receive information, based on the log of the user. For example, if a user would regularly visit the bookstore at a train station, but never the fastfood restaurant, more information about new books and less special offers for the restaurant could be presented.

System administrator functions:

- **Enter information:** system administrators can add, remove and change the information. This could be performed to reflect changes in the area in which the application(s), that is/are supported by the framework, are working.
- **View logs:** it should be possible to see logs of the framework, and thus the application, to see what information has passed through the system and what actions have been invoked. This could be necessary to ensure proper working of the system.

3.4.2. Non-functional requirements

In order for the applications, that are supported by the framework, to meet the non-functional requirements, the framework should also meet them, and therefore, the non-functional requirements are inherited by the framework.

In addition to the inherited non-functional requirements of the specific application in Section 3.3.2, a number of framework-specific requirements are identified.

Documentation
Documentation regarding the use of the framework should be available for developers. This should make sure the framework can be used effectively and be time-saving, as this is one of the goals of the framework.

Portability
It should be possible to reuse the code of the framework on other platforms, i.e., other mobile devices. This should be achieved by separating the application logic from platform-dependent code.

Usability
Where the target group of the applications are the *users* of the applications, the target group of the framework are the *developers* of the applications. Since their goal is to build mobile applications with context-aware augmented reality, the framework should make it easier to reach this goal.

Extensibility
The framework should be designed to take functional extensions into consideration. This could be done, amongst others, by enabling the future addition of context sources, information sources and types, information visualisation types, etc.

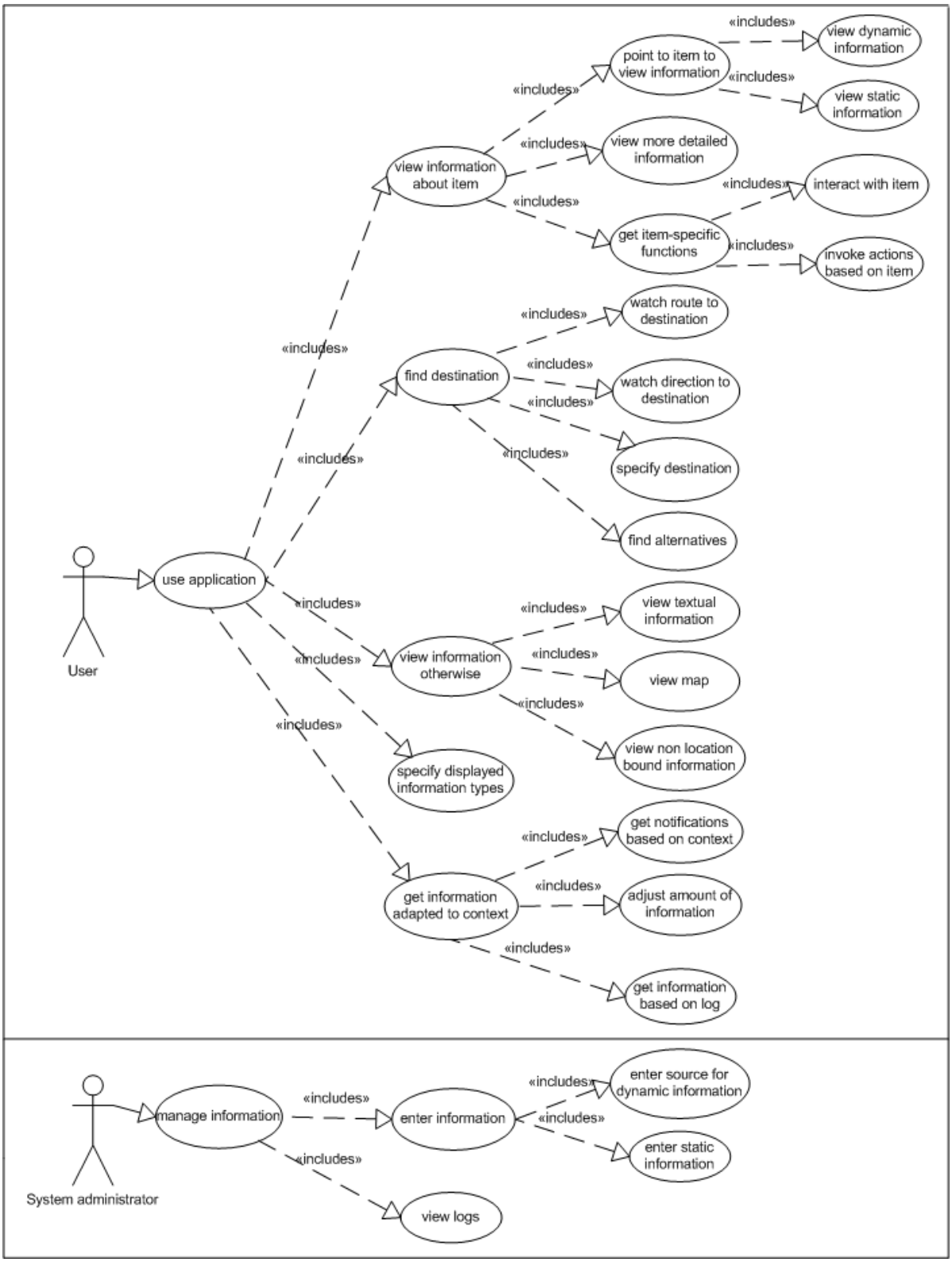

Figure 3.1.: Functional requirements for the framework

Chapter 4.

Framework architecture

This chapter presents a framework architecture, that meets the requirements, specified in Chapter 3. After presenting an overview of the architecture, the various components are discussed in more detail.

Section 4.1 presents the architecture at a higher level, along with the main components and their functionality. Sections 4.2 and 4.3 discuss in more detail the context-awareness component and the augmented reality component, respectively. Section 4.4 presents more information about the interaction between these components in

4.1. Components

The framework consists of two components, namely the context awareness component and the augmented reality component, to support the two distinct parts of functionality of context-awareness and augmented reality, respectively. These components are shown in Figure 4.1. Together, they provide the functionality to gather low-level context information, refine it to higher-level context information and display this information using augmented reality. The framework consists of software running on the mobile device, possibly interacting with software running on remote systems. The user application on the mobile device provides user input to the framework, and receives the augmented reality view from the framework.

Separating the two types of functionality into different components allows more flexibility when using the framework. Applications on mobile devices that are incapable of using the augmented reality component, for instance, due to the absence of a digital compass, can be supported by the context-awareness component, together with a different displaying component. By reusing only the augmented reality component, new or existing applications can be facilitated with augmented reality. Both components thus should be able to function separately and independently.

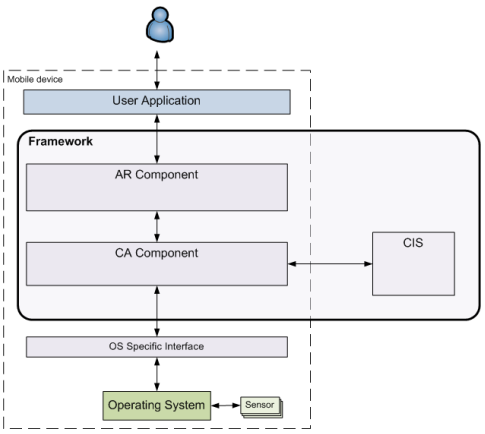

Figure 4.1.: Overview of the framework components

4.1.1. Context-awareness

This component should provide the features a context-aware application may support, as described in Section 2.1.1. For this thesis, we focus mainly on the presentation of information and services to the user. In order to do this, it should fulfil the process steps identified by Buchholz [16]:

- **Context sensing**: acquiring measurement data from the sensors as low-level context information.
- **Context refinement**: turning low-level context information into higher-level context information.
- **Context dissemination**: storage and provisioning of higher-level context information.

To retrieve the sensor measurements, the framework software communicates with the operating system (OS) of the mobile device. Since this communication is specific for the used OS, it should be kept separate from the framework. For this purpose, an OS specific interface layer should exist between the framework and OS, which is responsible for the transformation of OS specific data into a generic model. This layer can be adapted to accommodate the porting of the framework to a different OS.

The information that is provided by the component after performing the process steps, can consist of three types of information:

- **Context information**: information that describes the context of the user and application, for instance, the user's location.
- **Content**: information or links to information, provided by *content providers*, which can be bound to a location. For example, information about ticket prices is

bound to the location of the ticket dispenser, whereas information regarding the latest news is not bound to a specific location.

- **Notifications**: messages, triggered by monitoring the available context information and comparing this to rules set in advance. For example, a message could be triggered when the user is at a location with a temperature higher than 25°C.

As Buchholz suggested in [16], there are a few advantages of performing the context refinement process on a separate system. For this process, low-level context information needs to be supplied to multiple content providers, which requires a stable, sufficiently fast connection. Context refinement can also require a large set of reference data and computational power, such as a localization process, or a process providing all relevant information based on the user's location, i.e., selecting specific train station information from a database. Providing adequate computational power and storage in a separate system does not have the limitations of mobile device resources, and keeping the storage up to date is easier in a single system than in a large number of mobile devices. When one system combines context information for multiple users and content of multiple content providers, this system can monitor and trigger notifications based on these multiple sources. If, for instance, two friends are close to each other, the system can detect this and notify both users of this fact. For these reasons, context refinement should be carried out by a separate system. This system is called the context information service (CIS).

Consequently, low-level context information will have to be provided to the CIS by sources of context information, and higher-level context information will have to be returned to consumers of context information. The mobile devices fulfils both these roles, as it provides context information, such as sensor measurements or location information, to the CIS, and retrieves higher-level context information from the CIS. This can be done in either a push (publish) or pull (request) approach, represented in Figure 4.2 and 4.3 respectively.

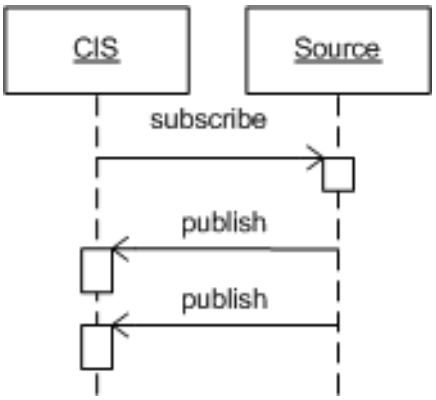

Figure 4.2.: Push approach

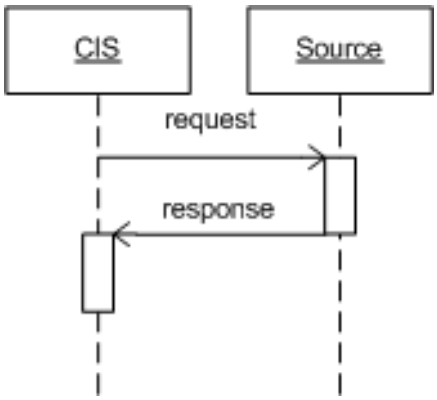

Figure 4.3.: Pull approach

Using the push approach, the source registers all interested entities, or *subscribers*, and autonomously publishes information to these subscribers when new context information is available. In the example of Figure 4.2, the CIS subscribes to the source. This approach is best suitable for regular information updates, since the source can only publish information when it is able to, i.e., if there is network connectivity, and when there is information to publish, such as a change in sensor measurements. If these conditions are met, the source can publish on a regular basis, for instance an update every second. These updates, called "events", generate "notifications" that need to processed by the subscriber, and therefore, this approach is also characterized as event-driven behaviour.

The pull approach is best suitable for context information updates on an irregular basis, like a temperature measurement in a room, which might only be needed when the user is in that room. An interested entity, in this example the CIS, requests a certain piece of information, which is sent back by the server in a response. Hence, this approach is also characterized as request-response behaviour.

In general, the CIS should be able to provide context information to multiple devices, combine data from multiple sources and support this functionality for a growing number of devices and context information types. It should therefore be designed and implemented in a scalable manner. Other important issues to be dealt with are privacy and security: context information is potentially private, and therefore, it should be dealt with securely with respect to the user's privacy. In order to maintain privacy with regard to the context information exchanged between the mobile device and the CIS, communication should be secure. This can be achieved by using a secure connection or encrypting the data. If context information is stored for a longer period of time, for logging or analysis purposes privacy should be warranted, for instance by anonymizing the data.

4.1.2. Augmented reality

The second component provides the augmented reality (AR) functionality, based on the information provided by the context-awareness component.

To display the content, the component calculates the location of the content in the field of view (FOV), matches it to a position in the representation of the real world and places the virtual content, representing the content, in the virtual content layer on top of the real world representation layer. Figure 4.4 shows this process, in which a real world object (a house) is represented in the real world representation layer. Content information (the description "house"), which is linked to the real world object's location, is displayed by a virtual content object in the virtual content layer.

For each location update, the FOV should be recalculated, along with the information to be displayed and where this information should be displayed.

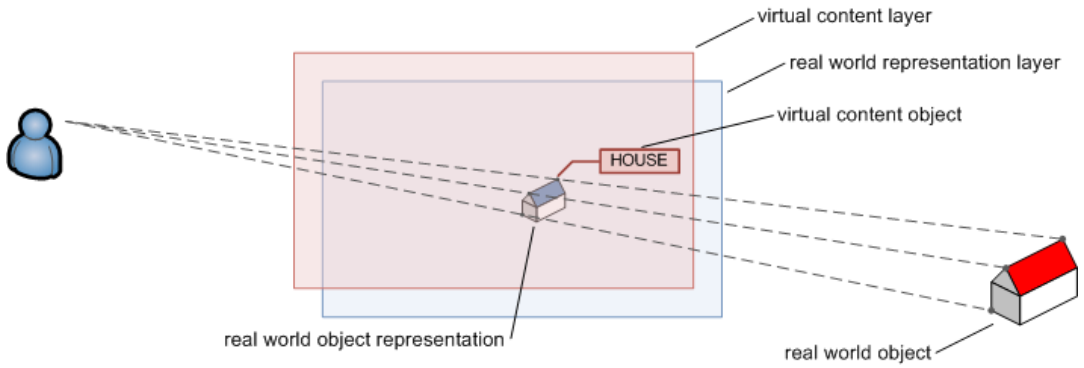

Figure 4.4.: Augmented reality layers

The processes of tracking and registration, as mentioned in Section 2.2.1, are of key importance for the correct working of AR. In order to correctly determine the FOV and keep the reality and virtual content properly aligned, the tracking process should provide results as accurately as possible. The direction accuracy depends solely on the accuracy of the sensors of the mobile device, while the location is the result of the entire localization process, including reference measurements, used algorithms and the wireless network infrastructure. By using a fast and accurate virtual content alignment process, the virtual content can be placed more precise and refreshed more often, making the augmented reality view more convincing and immersive.

Next to influencing *what* content is displayed, it should also be possible to change *how* virtual content is displayed, based on context information. For example, if the user holds the mobile device horizontally, the FOV thus pointing downwards, the display might change from AR to a map or textual view.

4.1.3. User interaction

Interaction with the user takes place through the application, which is supported by the framework. User input, like button clicks or text input, is received by the application and, when necessary, provided as input for the framework. This interaction with the user should be possible in three ways. Firstly, information is presented using virtual content in the AR view, based on the FOV the user chooses by moving around the mobile device. The user should be able to interact with the virtual content, for instance to display more detailed information. Secondly, if the AR view is disabled for certain context information, for instance when the FOV points downwards, textual and/or graphical content should be displayed. Thirdly, menus should allow the user to adjust settings of various parts of the application.

4.2. Context-awareness component

Figure 4.5 shows the internal structure of the context-awareness component.

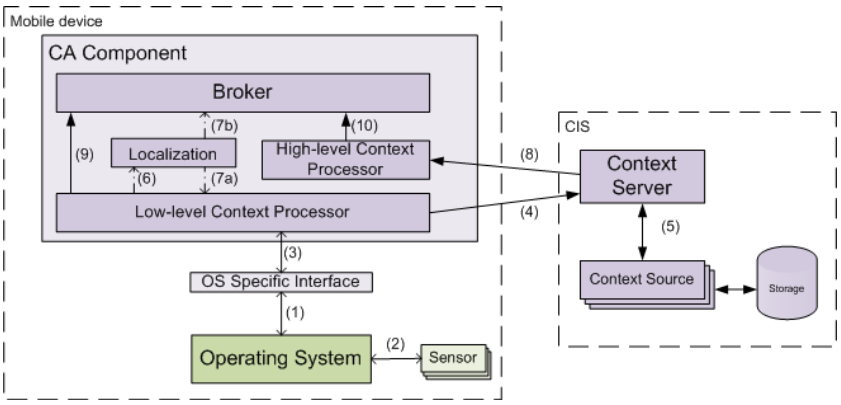

Figure 4.5.: CA component architecture

This component is responsible for performing the three process steps to support context-awareness, namely context sensing, context refinement and context dissemination. Context sensing is performed by context sources, which includes the mobile device and other sources of context information. The context refinement and dissemination steps are performed by the CIS. We discuss these steps in more detail below.

Context sensing

During the context sensing step, context sources retrieve measurement data or information and provide it as low-level context information. When the mobile device acts as a context source, the context-awareness component requests sensor data from the operating system of the mobile device (steps 1 and 2 in Figure 4.5). These sensor measurements are provided as low-level context information to the CIS. For representation of this information, a model should be used or developed. This model should support context information based on currently available and future sensors, although for the scope of this thesis, only context information based on the currently available sensors of the used mobile device platform are supported. To retrieve these measurements, the CA component communicates with the OS Specific Interface layer between the framework and the operating system (OS) of the mobile device. This layer transforms the OS specific data into the generic model (step 3).

Context refinement

Based on the low-level context information, received from the mobile device (step 4) and other context sources (step 5), the CIS provides the functionality of refining low-level

context information into higher-level context information. For example, the received signal strength measurements from the mobile device can be used by another context source, together with a database holding reference data, to determine the location of the mobile device. Other context sources and content providers can provide context information and content based on this location, for instance the local weather forecast or offers for stores in the vicinity.

A special kind of context refinement is the localization process. This process could be carried out by the CIS, but also by the mobile device. This can be achieved by either having the reference data already available on the mobile device or distributing it to the mobile device upon request. The localization process receives the measurements from the low-level context processor (step 6), determines the location, and returns the result as context information for the CIS (step 7a) and other components (step 7b).

There is a breakeven point when calculating the location, based on the measurements and the reference data, on the mobile device takes longer than sending data to the CIS, doing computations there and sending back the resulting location. This point depends mainly on the capabilities of the mobile device, capacity of the wireless network and size of the reference data. The localization logic should therefore be available both in the mobile device and in the CIS. This allows applications based on the framework to be used in both small and larger areas. Deployment of new or improved localization algorithms can be done faster by incorporating it on the external server, without the need for updating the software on the mobile device. This also holds for localization based on other techniques, such as Bluetooth.

The mobile device could choose to perform the localization process locally, or send the measurement data to the CIS to let the localization take place externally. The choice itself could also be passed on to the CIS, for instance, when no reference data for the current measurements is available. For this purpose, the mobile device should send the initial signal strength measurements to the CIS, which can return the reference data to perform the localization on the mobile device, or return a message, telling the mobile device to send the measurements to the CIS.

Context dissemination

The final step, context dissemination, has to deal with storing the higher-level context information and providing it to the mobile device (step 8).

Context information from the low-level context processor (step 9), the internal localization process (step 7b) and the CIS (step 10) is passed on to the broker. The broker provides the context information to other software components that use the context-awareness component, such as the augmented reality component.

Existing middleware

A large number of context-awareness middleware systems exist that can fulfil the role of the CIS (see Section 2.1.4). Such a middleware system should support the context sources and content providers that are needed to perform context refinement. These should be able to gather specific types of required context information, and provide information back to the system. Information destined for the mobile device should be automatically dispatched to this device. Possible extensions of context sources and content providers should be supported. The middleware system should also facilitate the creation of notification rules, the monitoring of the context information, based on these rules, and the triggering of notifications if a condition of a rule becomes valid.

Based on [6, 22, 23], we have chosen to use two parts of the system developed in the scope of the EC funded IST-IP Amigo project [71] as a context-awareness middleware system. The goal of the AMIGO project was to solve the key issues in the realization of ambient intelligence for the networked home environment. From this project, we have used the Context Management Service (CMS) and Awareness and Notifications Service (ANS) [72]. Both are implemented by modules on an OSGi platform [73] named Oscar [74].

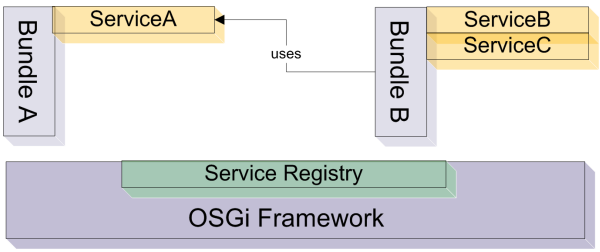

Figure 4.6.: OSGi structure

The OSGi Alliance has specified a dynamic, service-oriented module system for Java. This platform fosters extensibility, by facilitating the deployment of new modules, known as “bundles”, without a system restart. New modules that provide context information, possibly based on other (lower level) context information, can be added to the server in a simple manner, without requiring any changes in the mobile device. Figure 4.6 shows the basic structure of the OSGi platform.

4.3. Augmented reality component

Figure 4.7 shows the internal structure of the AR component.

In order to provide markerless AR functionality, the component needs the location, rotation, tilt and roll of the mobile device (see Section 2.2.3). These can be acquired from the CA component (step 1). Based on this data, the field of view (FOV) can be

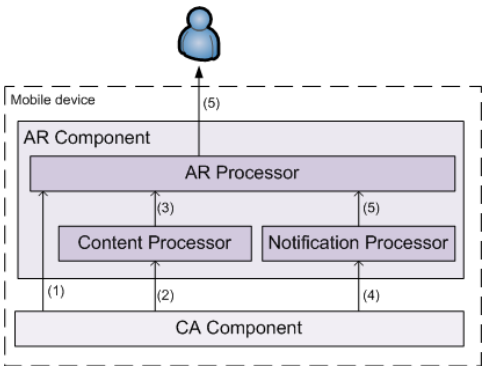

Figure 4.7.: AR component architecture

established. The CA component also provides content (step 2) and notifications (step 3), which have been received from the CIS. Location-bound content can be shown in the AR view when a user looks at the location the content is bound to, whereas location-independent content will be shown at some specific place in the view, for example a weather icon in the corner of the screen. For this purpose, the content is passed on to the AR Processor (step 4), which establishes what content is currently visible within the current FOV, and displays it accordingly (step 6). Notifications are also passed on to the AR Processor (step 5), which can, for example, change the AR view accordingly, like showing less content, or display information from the notification.

4.4. Component interaction

Interaction between the CA component and the AR component takes place through the Broker of the CA component, which facilitates the dissemination of context information, content and notifications. The AR component is registered at the Broker to receive this information.

Using this approach, the components can easily be reused separately from each other. Other components can use the CA component to add context-awareness to an application by registering at the Broker and process the received information. It is also possible to pass information to the AR component by registering it at another implementation of the Broker, which can then provide the necessary information to the AR component.

Figure 4.8 shows the overview of the framework's internal structure. The detailed component parts are shown, along with the flow of information between these parts.

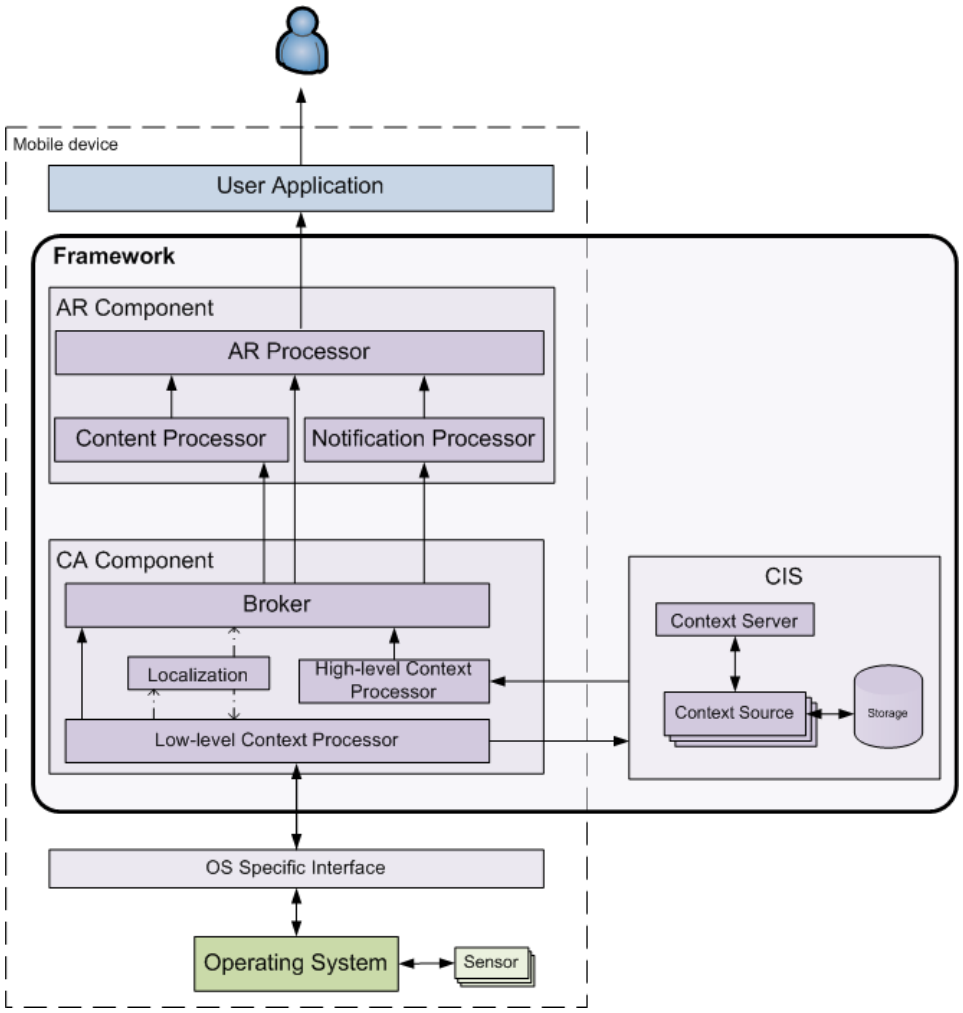

Figure 4.8.: Framework structure

Chapter 5.

Framework software design

This chapter describes the software design of the framework, and discusses how the software packages and classes implement the architecture described in Chapter 4. Since only some parts of the design are discussed here, the complete design can be found in Appendix A.

Section 5.1 discusses the used localization technique, Sections 5.2 and 5.3 discuss the context-awareness and augmented reality component, respectively. Section 5.4 presents the way platform-independence is incorporated in the design. For each of these parts, we identify what techniques are available to implement the functionality blocks in the architecture, and justify the choices that have been made regarding these techniques.

The prototype implementation of the framework and proof-of-concept application have been designed to run on Google Android [75], an operating system for mobile phones, using the Java programming language. The open source nature of the OS is beneficial, since it facilitates the access to the phone's functionality through well-documented code. This platform runs on the G1 Developer Version mobile phone. The G1 is one of the few currently available phones that contains an orientation sensor for all three axes, which is a key requirement for using markerless augmented reality.

5.1. Localization

The location of the user is necessary for both the CA and AR component: it serves as low-level context information for the context refinement process of the CA component, and it is needed for the tracking process of the AR component. For the latter in particular, the correct functioning of AR depends largely on the accuracy of the tracking process, and thus, on the accuracy of the localization process.

The localization techniques using WLAN are best suited for the framework, because of the high accuracy, increasing availability of WLAN adapters in mobile devices, and the requirement of no or little extra investments in hardware (see Section 2.3). A number of these techniques are compared in [76], which concludes that the Horus WLAN Location Determination System by Youssef and Agrawala [55] has the highest accuracy. For

this reason, and because of the availability of multiple papers describing the details of this technique, we chose this technique for localization. Since no implementation was publicly available, we have designed and implemented this technique according to the details given in the papers.

This probabilistic technique uses an offline phase, in which the fingerprint database is built by doing measurements on reference points. After this, in the online phase, the location is determined using a number of process steps. We describe these two phases below, and more detailed information can be found in [55, 77, 78, 79, 80].

Offline
During this phase, the reference material for the localization process (i.e., the online phase) is gathered. This data consists of three parts:

- Radio map, which contains the distribution of received signal strength measurements for each access point at each reference location.
- Clusters, which are groups of reference locations, based on the access points that cover those locations. Therefore, if on a reference location four access points are identified, this location becomes part of the clusters of these four access points.
- Parameters, which are used in the parametric signal strength distributions. These distributions are assumed to be Gaussian with mean μ and variance σ^2. In [79, 80], it is shown that the distribution of n correlated samples has mean μ and a variance given by

$$\sigma_n^2 = \frac{1+\alpha}{1-\alpha}\sigma^2 \tag{5.1}$$

Figure 5.1 shows the class structure of the offline localization process.

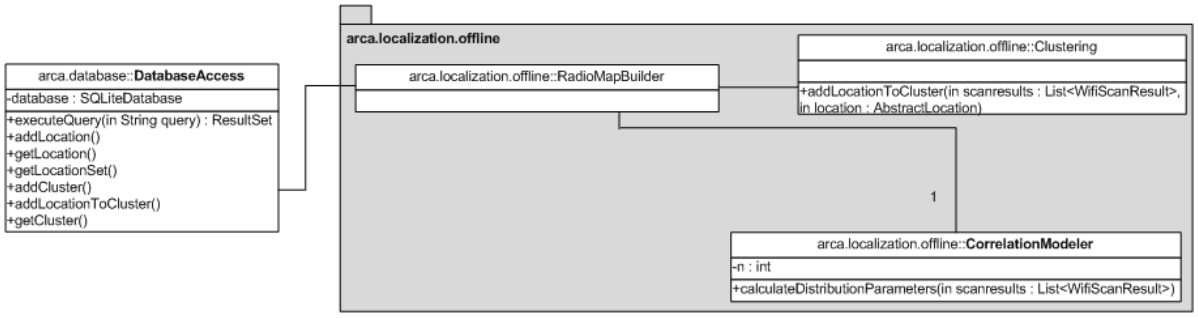

Figure 5.1.: Diagram of the offline localization process

The distribution parameters μ, σ^2 and α are calculated by a `CorrelationModeler` object and stored in the radio map. The clusters are built by the `Clustering` object. Both the radio map and the clusters are stored in the database by the `DatabaseAccess` object.

Online
In this phase, the actual location of the mobile device is estimated, based on current measurements of the received signal strengths and a set of reference data, gathered during the offline phase. Figure 5.2 shows the class structure of the online localization process.

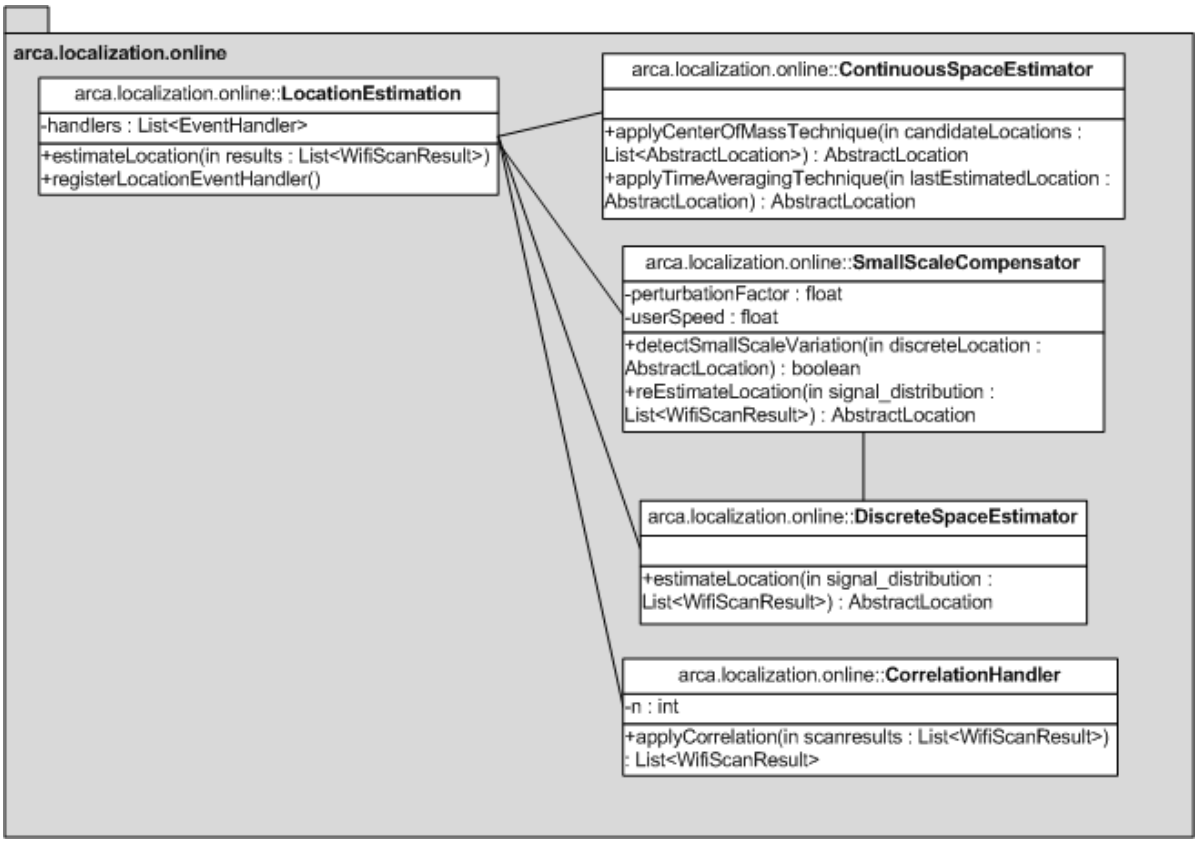

Figure 5.2.: Diagram of the online localization process

There are four steps in this localization process, each implemented in its own class:

1. `CorrelationHandler`: accounts for temporal signal-strength variations, by averaging n consecutive measurements. Correlation between successive measurements from one access point can be as high as 0.9, and therefore, earlier measurements are taken into account to improve results [79, 80].

2. `DiscreteSpaceEstimator`: uses the averaged signal strength vector to return the radio map location that has the maximum probability for the given signal distribution.

3. `SmallScaleCompensator`: compensates the signal-strength variations that occur due to the mobile device moving a small distance (order of wavelength), which is 12.5 cm for a 802.11g network. These variations can range up to a difference of

10dBm, and to account for this, these variations need to be detected and compensated. Detection is done by estimating the location, and calculating the distance between this location and the previous estimate. If this is above a threshold, determined based on the movement rate, a small scale variation is assumed present.

To compensate the variation, part or all of the entries in the signal strength vector are perturbed, i.e., a `perturbationFactor` is added or subtracted, locations are reestimated for all the variants of the vector and the location nearest to the previous location is chosen.

4. `ContinuousSpaceEstimator`: estimates the location in continuous space, based on the set of reference locations, which is provided by the discrete-space estimator. This set of locations is ordered by their probability, and therefore, the most likely location is the location at the top of the list. Using two techniques, a more accurate location from the continuous space can be determined, based on the ordered list of reference locations found by the discrete-space estimator:

   - The Center of Mass technique uses the ordered list of probable locations, acquired from the discrete-space estimator. By taking not just the single most-probable location, but N most-probable locations and establishing the center of mass for those locations, a more accurate location from the continuous space can be determined. The probabilities for the N locations are normalized, such that all the N probabilities sum up to 1. The location is then estimated by taking the centre of the weighted N locations.
   - The Time-Averaging technique uses the average of the last W estimated locations to smooth the newly estimated location. This technique can be used to compensate the shifting location due to small scale signal variations, like the SmallScaleCompensator does, but might provide less accurate results if the user moves considerably. The accelerometer in the mobile device could be used to detect this movement, after which W could be adjusted accordingly. This, however, is currently not considered in this design, but it is further discussed in Section 8.2.

   Both techniques could be used independently, based on the results of the discrete-space estimator, but to improve accuracy, we have used the output of the Center of Mass technique as input for the Time-Averaging technique.

5.2. Context awareness

The context information service (CIS) processes low level context information into higher-level context information on a server running in a remote machine with respect to the mobile device. In the CA component in the mobile device, sensor data needs to be

processed in order to be sent to the CIS. After receiving information from the mobile device, the CIS fulfils the task of gathering low level context information, processing it into higher-level context information, possibly in conjunction with other context sources, and disseminating the processed information back to the mobile device (see Section 4.1.1). This information is presented to other components using the CA component through the Broker.

5.2.1. Context Information Service (CIS)

An overview of the different components of the CIS and how they interact is shown in Figure 5.3.

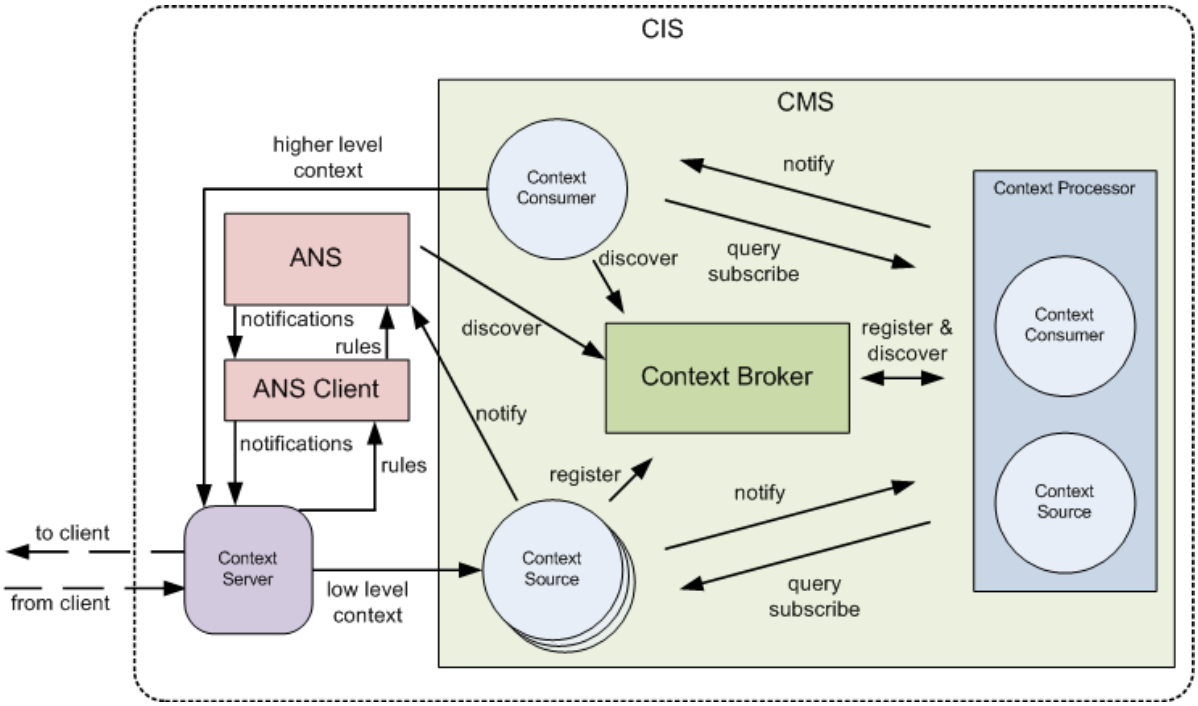

Figure 5.3.: Diagram of the CIS

Context Management Service

The Context Management Service (CMS) is the basic service for the registration of sources of context information and providing them upon request, and contains components that perform the following three roles:

- *Context Source*: provides context information, upon request (pull-approach) or subscription (push-approach), to the context consumers.

- *Context Broker*: maintains a registry of all available context sources and acts as a service directory for the context consumers.
- *Context Consumer*: looks up a context source at the broker and either subscribes to it to receive context information when it is available or performs a request-response interaction with the context source.

Besides these three roles, we defined the *Context Processor* role, which acts both as a context consumer and a context source. The processor retrieves lower level context information, processes it, and provides it back to either the mobile device or other context processors as higher level context information. For instance, wifi data is processed into a location, or a location is processed into a weather forecast for that location. By facilitating context processors, the CIS provides context refinement.

Within the AMIGO project [71], the Context Broker has been implemented, and the basic structure for the Context Source (CS) and Context Consumer (CC) has been specified. We have extended this functionality by implementing abstract modules for the CS and CC, which are reused to implement a number of Context Processors. These abstract modules provide implementations of basic functions, such as context source registration, subscription handling and context information interpretation.

Awareness and Notifications Service

The ANS monitors the available context information according to a set of monitoring rules. If a triggering condition holds for a certain rule, a notification is fired and forwarded to the framework application. To do this, besides the event monitor, the ANS provides components to manage the rules, store the rules and notify the client application. In the design of the CIS, modules are allowed to add rules to the ANS. This adds the ability to not only extend the context processing by deploying a new module, but also increases the awareness with regard to the context information provided by the new module.

The main functionality of the ANS has been implemented in the AMIGO project, but we have extended the implementation by adding new context information specifications. By doing this, the rules can be applied to newly specified context information types.

CIS processes

In the CIS, context information is represented using RDF (Resource Description Framework [81]). An ontology is specified, using the OWL Web Ontology Language [82], which represents the set of concepts within a domain, along with their properties and the relationships between them. Since the preexisting ontology of the AMIGO project was not applicable for this project, we defined a new basic ontology for the framework, describing general concepts, such as a location, a user and sensor data. For an application, a

separate ontology can be defined and used by the framework to describe domain-specific concepts.

Figure 5.4 shows the process of context source registration to the context broker. Using a Lookup-service, which is part of the CMS middleware, the context source retrieves the context broker. The context source then registers itself at the context broker, providing the `rdfCapability` String that describes the capability of the context source in RDF.

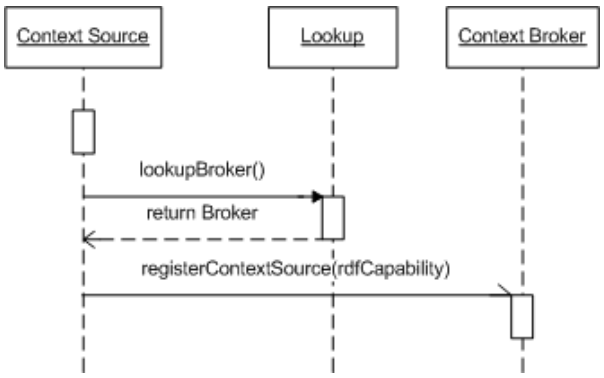

Figure 5.4.: Sequence diagram of the context source's registration process

Figure 5.5 depicts a context consumer's registration process. After retrieving the context broker, the context consumer uses the context broker to find context sources that match the `rdfCapability` String that describes the context consumer's information need. If one or more matches are found, addresses of these context sources are returned. These are used to retrieve the actual context source and subscribe to it. A `SubscriptionRefresher` is used to refresh the subscription at regular intervals, which would otherwise have been removed by the context source.

Whenever context information changes, the context source's `contextChanged()` method is called by the actual context source, i.e., the software that manages the sensors. For each subscribed context consumer, the specific query is retrieved and performed on the context information. The result is put in a `NotificationData` object and passed on to the subscribers by calling their `notify()` method. For example, a context consumer has registered at a location context source, with a specific query that retrieves the altitude. If the location of a user changes, the query is executed on the location information, returning only the altitude. This process is shown in Figure 5.6.

New modules that might be added to the system later, may generate new types on information. From the developer's point of view, these could be new subclasses of the `ContentEvent`-class. To handle these events, developers simply need to implement a new class that implements the `ContentInterpreter` interface, specifically for each new type of content information. Figure 5.7 shows these interpreter interfaces. The specific event handlers that implement the `EventHandler` interface receive

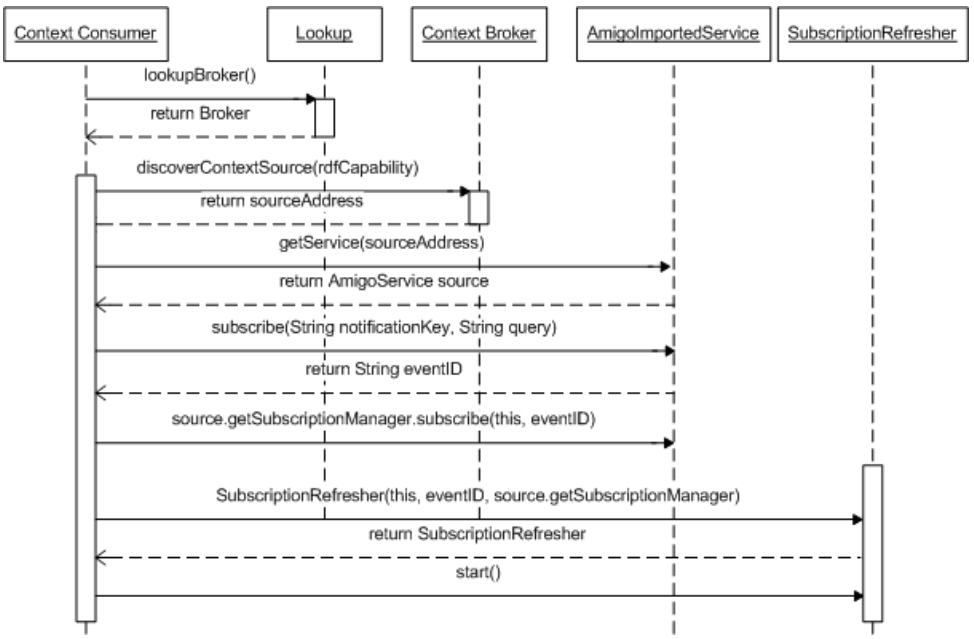

Figure 5.5.: Sequence diagram of the context consumer's registration process

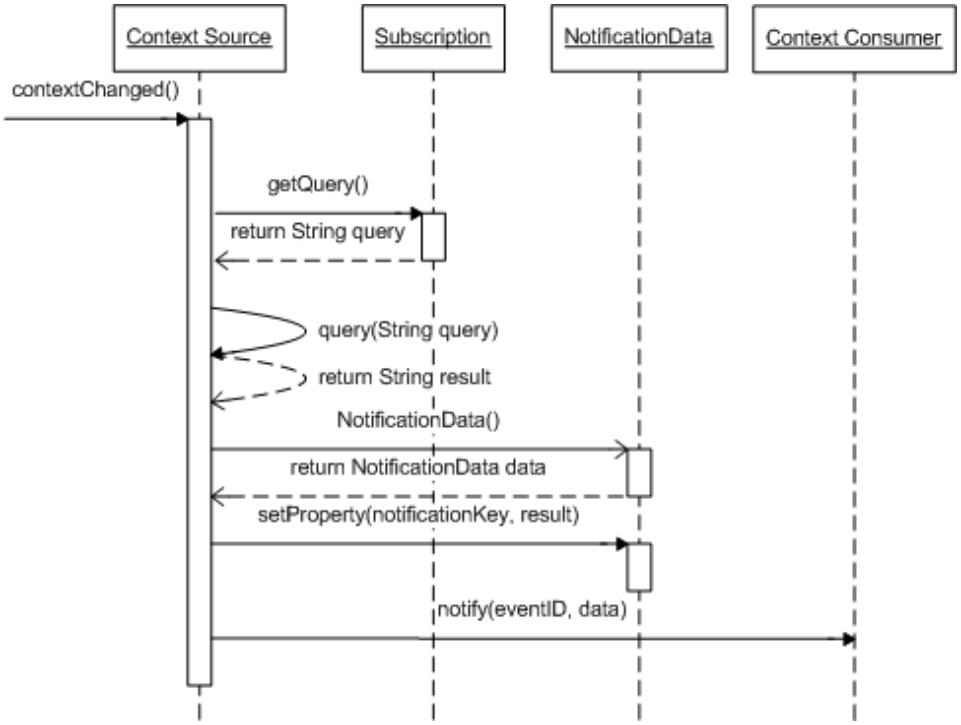

Figure 5.6.: Sequence diagram of the context notification process

`ContentEvents` and `NotificationEvents` from the dispatcher, and the `handle()` method of the appropriate interpreter is invoked to further process an event.

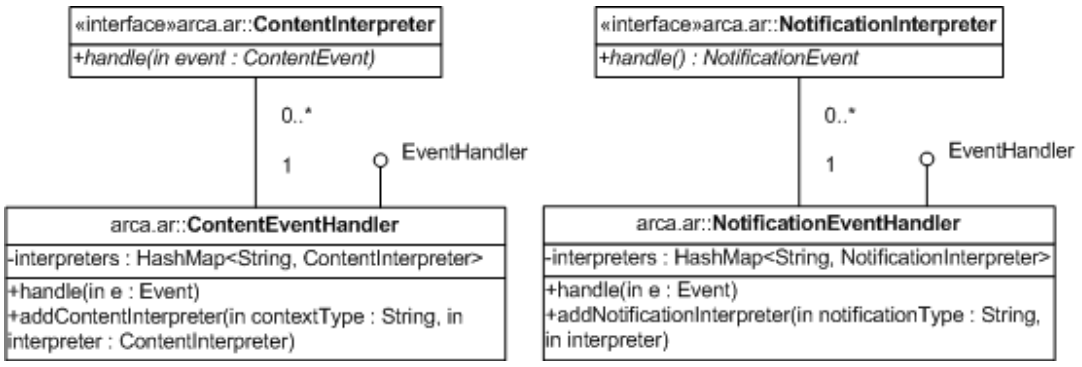

Figure 5.7.: Interpretation of content information

5.2.2. Communication with mobile device

In order to provide context information to and receive context information from the CIS, a communication channel between the mobile device and the CIS should be established. For this communication, multiple approaches could be used:

- Remote procedure calls (RMI/CORBA): based on calling functions on other remote systems.
- Web services: based on XML messages over the HTTP-protocol, directly sent between client and server.
- Message-oriented middleware (MOM): based on text messages, which could include XML, either sent directly or using a dedicated message server.

The protocol should support the fast transmission of mostly small messages. In order for the CIS to be reusable for different platforms, it should be platform-independent. Since the connection with the mobile device might drop out, the protocol should be able to deal with a lost connection between the mobile device and the CIS. Both the push and pull approach should be supported, using asynchronous transmission, i.e., the information transmission may or may not start immediately as requested by the sender.

These features are best supported by message-oriented middleware, and therefore, we chose this approach. The CIS based on the CMS and ANS provides its results as text, which can be put into messages without any transformations. Using a separate message queue for each connected mobile device, the retrieval of message is automatically adapted to the state of the connection between the mobile device and the CIS: whenever the CIS has information available, it can be placed in the message queue, and whenever the mobile device is ready for the next piece of information, it can take a message out of the queue.

We chose the Extensible Messaging and Presence Protocol (XMPP), which provides a set of open XML technologies for message-based communication, standardized by the IETF [83]. The Multi User Chat (MUC) extension has been used as a message queue

for the communication between the CIS and the mobile device. Figure 5.8 shows the communication architecture between the mobile device and the CIS. The mobile device connects directly to the Context Server (see also Figure 5.3), which forwards the various types of context information messages to the respective MUC. The context source (CS) for that type of context information retrieves the information from the MUC and provides it to the CMS.

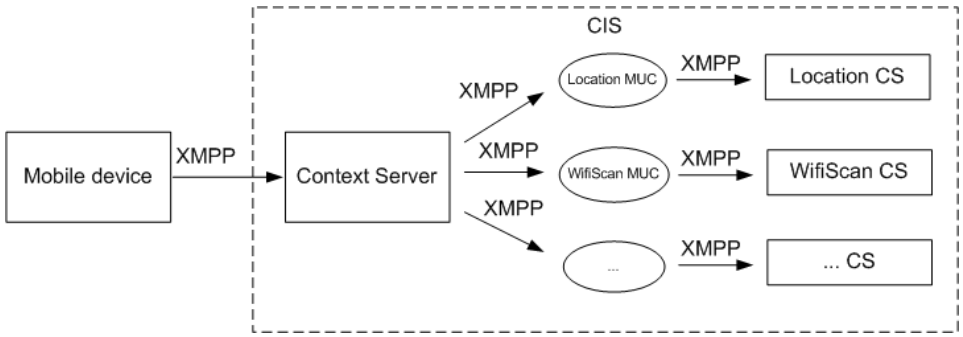

Figure 5.8.: Communication architecture between the mobile device and the CIS

5.3. Augmented reality

The augmented reality component of the framework is based on the `ARMain` object (Figure 5.9). This main class receives the input from other components and manages the output to the user. There are three basic types of input:

- Input for the tracking process of augmented reality, which are the location and orientation. These are provided by the `LocationEventHandler` and the `OrientationEventHandler`.
- Content information to be displayed, which is sent as events by the CA component and received by the `ContentEventHandler`. After an event has been processed by the `ContentInterpreter`, it is passed on to the main class.
- Notifications, which are triggered by rules on the server and received by the `NotificationEventHandler`. The main class handles the notifications after they have been processed by the `NotificationInterpreter`.

Based on the input from the various `EventHandlers`, the main class controls and invokes the various display modes, which are implemented in objects based on the `Activity` class [84]. An `Activity` is the main entry point of the application to the user, as it displays the user interface and gets the user's input.

No implementation for the AR component was available, so we have designed and implemented the AR component, based on the specification given in Chapter 4.

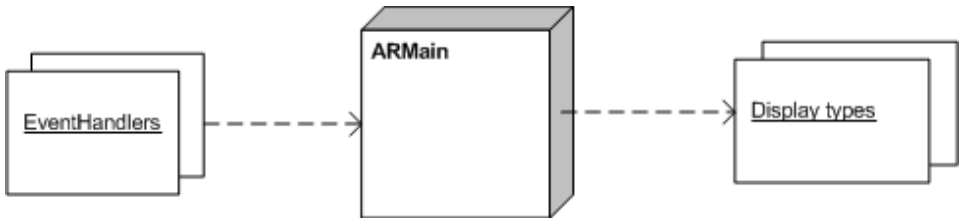

Figure 5.9.: Input and output of the ARMain-class

5.4. Platform-independence

A number of measures have been taken in the design to ensure the separation of platform-dependent and platform-independent code. By doing this, porting the framework to a different platform requires changes only in a limited number of classes. An example of this can be found in the processing of sensor events that are received from the operating system. The class structure of this functionality is shown in Figure 5.10, using the example of the orientation sensor. A similar structure is used for other sensors.

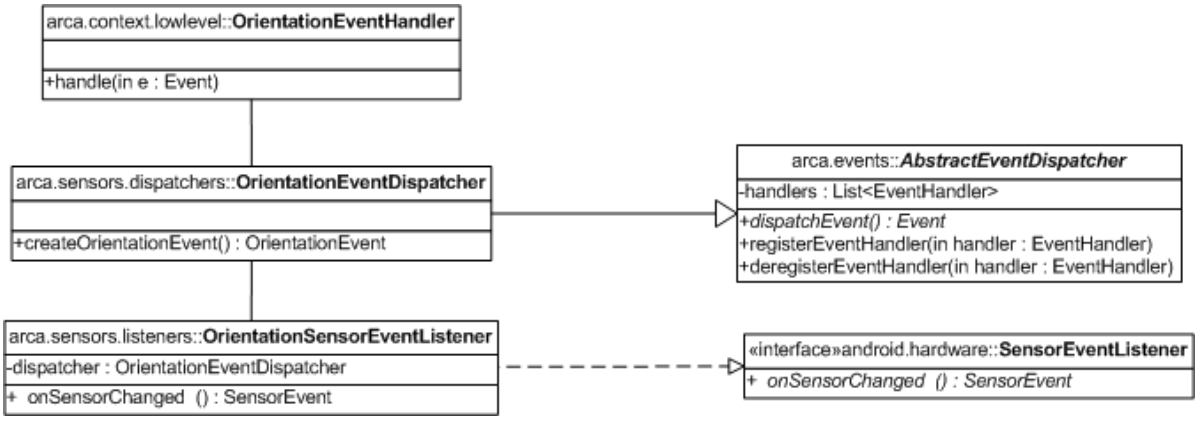

Figure 5.10.: Class structure of OS sensor event processing functionality

Events received by the `OrientationSensorEventListener` reflect a change in the device's orientation. This is detected by implementing the `SensorEventListener`-interface defined by the OS, and registering itself at the OS to receive this specific type of events. Since these events are OS-specific, information from these events is put into a new `OrientationEvent`, which is a generic event type, specified for the framework. This newly created event is dispatched to all `EventHandlers` that are registered at the `OrientEventDispatcher`. If changes are made to the OS, or the entire framework is ported to a new OS, only the `OrientationSensorEventListener` class needs to be changed, in order to put the data of the received OS specific events into the generic framework events.

The process of storing and receiving data has also been designed to be platform-independent. This is done by using an SQL database [85], which is already available in the Android

OS. SQL is a widely used standard and SQL database management systems (DBMS) are available for many platforms. As shown in Figure 5.11, the `DatabaseAccess` class uses the platform's `SQLiteDatabase` class to interact with the available database. In case a different database is used, only the `executeQuery()` method of the `DatabaseAccess` object and the database attribute have to be changed, so that queries can be routed to another DBMS, or even to a DBMS running on a remote server.

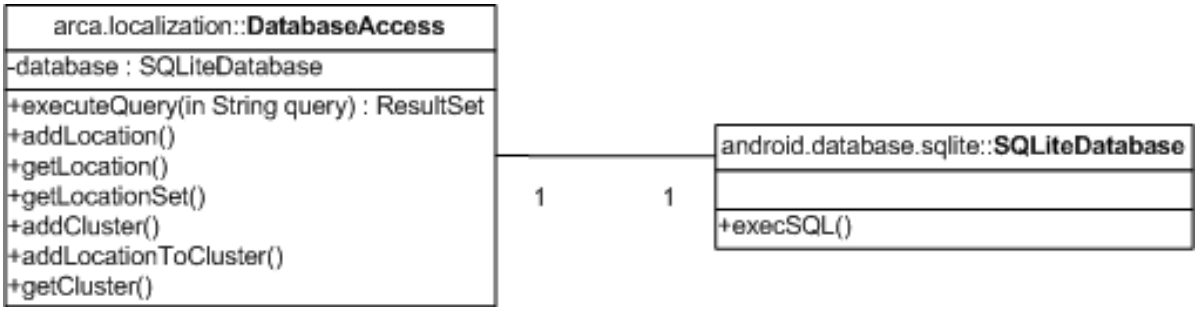

Figure 5.11.: Data storage in the framework

Chapter 6.

Framework implementation

This chapter reports on the development of the framework, which was implemented, based on the framework software design that has been described in Chapter 5. In order to evaluate the developed framework, we have implemented a proof of concept application that is supported by the framework. This evaluation is discussed in Chapter 7.

Section 6.1 presents the approach we have chosen to implement the framework. Sections 6.2, 6.3 and 6.4 present more details about the implementation of the localization process, the context-awareness component and the augmented reality component, respectively. Section 6.5 describes how the framework supports the process of deploying an application.

6.1. Approach

In order to implement the framework, a bottom-up approach has been chosen in which parts are added incrementally. We have first started implementing the following parts on the mobile device:

- The *OS specific interface* to retrieve (sensor) data from the mobile device's operating system (OS).
- The *localization* part, which provides location estimates based on WLAN measurements. For the offline phase, a tool is implemented to measure reference data and store it in a database.
- The *low level context processor*, which provides context information that has been acquired by the software on the mobile device to the CIS.

The mobile device can now act as a context source for the context information service (CIS). Subsequently, the following parts of the CIS are implemented:

- The *context server*, which facilitates the communication between the CIS and the mobile device.

- Abstract components for *context sources* and *context consumers*, which can be used to respectively provide context information to the CMS and retrieve context information from the CMS. Using these abstract components, a number of *context processors* have been implemented.
- A context consumer that can provide the context information that is destined for a mobile device to the context server.
- Adjustments and improvements to the ANS, which were necessary to enable the use of context information that uses the newly defined ontology.
- The *ANS client*, which registers rules at the ANS, receives triggered notifications and passes these notifications on to the context server, to be provided to the mobile device.

The CIS is now able to provide information and notifications to the mobile device. In order to provide the augmented reality view, a number of implemented parts are added to the mobile device's software:

- The *high level context processor* that receives information from the CIS, and a number of *event dispatchers* that place the information at other component's disposal.
- A storage system that stores the received context information, content information, and notifications.
- The *AR processor* that shows the available content information, based on the user's field of view. By processing received notifications, the displayed information can be adjusted accordingly.
- Other views, such as the map based view, and ways to switch between these views.

The various parts, and the interaction between these parts, have been shown before in Figure 4.8 in Section 4.4. The framework has been named the ARCA Framework, based on the abbreviation "Augmented Reality Context-Awareness".

6.2. Localization

The localization technique has been implemented as a subcomponent of the CA component in such way that the process can be carried out either on the mobile device or on the server. Both processes use an SQLite database, facilitating the interchanging of the database file without the need of any conversions.

To test the process during development of the framework, a testbed has been set up on floor 3C of the Logica building in Arnhem, the Netherlands. This area of 72 m^2 is covered by five access points and has been mapped by performing 100 WLAN received

Mobile device	Server	Communication	Movement	Time (ms)
•			•	1253
	•	•		1268
	•	•	•	1413
	•		•	14

Table 6.1.: Performance comparison for the localization process

signal strength measurements at each reference point on a grid with a spacing of 2m. There are a total of 24 reference points.

In order to compare the performance of the localization process on the mobile device and on the server, a number of time measurements have been conducted while performing 50 location estimations. The results of these measurements are shown in Table 6.1. A dot in the "Communication" column implies that communication between the mobile device and the server has been included in the measured time, whereas a dot in the "Movement" column implies that the mobile device was moved around the area during the localization process.

The average measured time between sending the WLAN measurements to the server and receiving a location estimate at the mobile device is only slightly higher than the average time it takes to calculate the location on the mobile device. If the number of reference points would be doubled, for instance, by doubling the size of the testbed or bringing down the grid spacing to 1m, the necessary time for the location estimation also increases. In a worst-case scenario, the calculation time would double. When performing the localization process on the mobile device, the calculation time could therefore double to 2506ms, whereas running the calculation on the server would increase the time only from 14ms to 28ms, raising the total time to 1427ms. The breakeven point, thus when performing the localization process on the server will provide results in the same amount of time as on the mobile device, can therefore be set at 25-30 reference points, on condition that the access point coverage remains at an average of five. An increase in access point coverage will lead to more calculations, thus lowering the breakeven point.

The estimated locations are represented as coordinates in a Cartesian coordinate system, in which each point is specified relative to the origin of the coordinate system. Each different area, in which the localization process is used, has its own origin, and, therefore, its own unique identifier, called the *reference*. Subsequently, each location object must contain both the coordinates and the reference.

6.3. Context-awareness component

On the mobile device, the primary goal of the context-awareness component is to send low level context information to the CIS, and receive information from the CIS. For

this interaction, the XMPP protocol [83] has been used, supported by an open source Java library called Smack [86]. This library was implemented to be used by applications that run on Java 5 or higher. The Android OS, however, contains only a part of the functionality of the Java 5 version, resulting in the malfunctioning of the library. To overcome this problem, we recompiled the main library code and the Multi User Chat (MUC) code using the Java version of Android, bypassing functionality that cannot be used due to the Java limitations.

Using XMPP messages, context information is sent to the Context Server, which forwards these messages to the according MUC, based on the type of message. In order to do this, both the mobile device and the Context Server connect to an open source XMPP server called Openfire [87]. This server has been setup with various user accounts and MUCs for the different types of information. Both the Openfire server and the Context Server application could be running on different server systems than the rest of the CIS, which consists of the CMS and the ANS. We will discuss these parts of the CIS in the following sections.

6.3.1. Context Management Service

In order to facilitate the development of context sources (CSs) and context consumers (CCs), we have implemented abstract modules for both. Figure 6.1 shows an example class structure of the location CS, but a similar approach has been used for other context sources. Each CS and CC is implemented as a separate OSGI bundle.

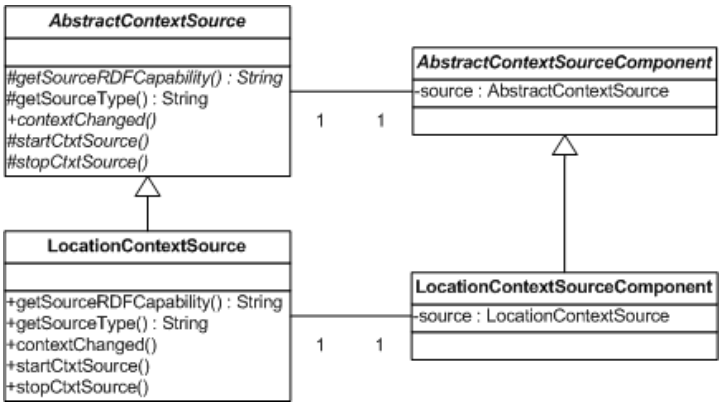

Figure 6.1.: Abstract context source parts with subclasses

Each context source consists of the following classes with a specific function:

- `AbstractContextSourceComponent`: provides the entrance point for the OSGI platform, which invokes its `active()` and `deactivate()` method to start and stop the bundle. It also exports the `subscribe()`, `unsubscribe()` and `query()` methods, which can be used by context consumers to retrieve context information, and registers the source at the Context Broker.

- `AbstractContextSource`: provides a generic implementation for the exported `subscribe()`, `unsubscribe()` and `query()` methods, and provides the `notifySubscribers()` method to subclasses, which can be called after the context information has been changed. It also contains a reference to the `OntModel` model, which is used to store the current context information. Each subscription has its own query, which is processed using the model, to provide the required result for each individual subscription.
- `LocationContextSourceComponent`: sets the source attribute to a specific subclass of `AbstractContextSource`.
- `LocationContextSource`: implements the `contextChanged()` method, in which the model is changed to reflect an update in the context information. By implementing the `getSourceType()` and `getSourceRDFCapability()` methods, it can provide information about its type and capabilities, which is used by the `AbstractContextSourceComponent` to register this context source at the Context Broker.

An example of a model for a Location CS is shown in Listing 6.1 (for readability, the complete URL to the ontology has been left out at each occurrence of "[..]").

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:j.0="http://www.freekuijtdewilligen.nl/Arca.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#">
  <owl:Class rdf:about=[..]"Person"/>
  <owl:Class rdf:about=[..]"CartesianLocation"/>
  <owl:ObjectProperty rdf:about=[..]"isContextOf"/>
  <owl:DatatypeProperty rdf:about=[..]"reference"/>
  <owl:DatatypeProperty rdf:about=[..]"id"/>
  <owl:DatatypeProperty rdf:about=[..]"x"/>
  <owl:DatatypeProperty rdf:about=[..]"y"/>
  <owl:DatatypeProperty rdf:about=[..]"z"/>
  <owl:DatatypeProperty rdf:about=[..]"timestamp"/>
  <j.0:CartesianLocation>
    <j.0:isContextOf>
      <j.0:Person>
        <j.0:id>Alice</j.0:id>
      </j.0:Person>
    </j.0:isContextOf>
    <j.0:reference>test_area_1</j.0:reference>
    <j.0:z>0.0</j.0:z>
    <j.0:y>0.2836</j.0:y>
    <j.0:x>4.0</j.0:x>
  </j.0:CartesianLocation>
</rdf:RDF>
```

Listing 6.1: Example model containing location information

Figure 6.2 shows the class structure of an example location CC, but a similar structure was used for other context consumers.

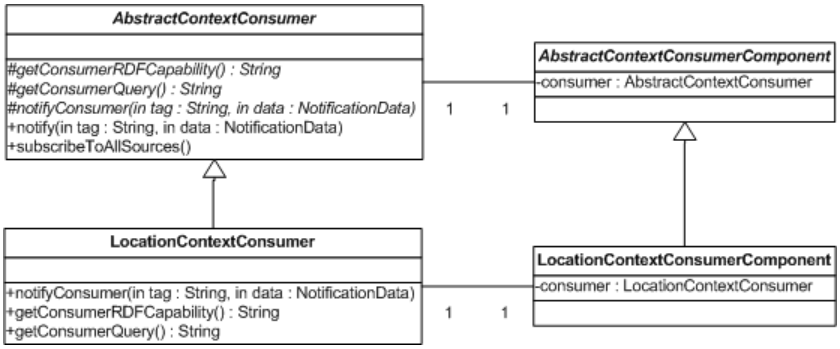

Figure 6.2.: Abstract context consumer parts with subclasses

Each context consumer consists of the following classes with a specific function:

- `AbstractContextConsumerComponent`: provides the entrance point for the OSGI platform, which invokes its `active()` and `deactivate()` method to start and stop the bundle. It also provides the `AbstractContextConsumer` with the Context Broker.
- `AbstractContextConsumer`: provides a generic implementation for the exported `subscribeToAllSources()` method, in which a subscription with a specific query is placed at all context sources that have a matching RDF description. These context sources are found using the provided Context Broker. It also provides an implementation of the `notify()` method, which gets invoked whenever a context source has provided new context information. In case the context consumer is used in a context processor, the received context information is passed on to the source part of the context processor, otherwise, the information is passed on to the subclass using the `notifyConsumer()` method.
- `LocationContextConsumerComponent`: sets the consumer attribute to a specific subclass of `AbstractContextConsumer`.
- `LocationContextConsumer`: implements the `notifyConsumer()` method to process the incoming context information. By implementing the `getConsumerRDFCapability()` and `getConsumerQueryString()` methods, it can provide information to the `AbstractContextConsumerComponent` to respectively find the appropriate context sources, and get the relevant information from these context sources after subscribing to them.

The query that is specified by the context consumer at the time of subscription is expressed in SPAQRL [88], a query language for RDF. Since the context information is specified in a model using RDF, SPARQL can be used to extract information from the model. An example query for a user's ID, the x, y, and z coordinates, and the reference of his location is shown in Listing 6.2.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX arca: <http://www.freekuijtdewilligen.nl/Arca.owl#>
SELECT ?userid ?x ?y ?z ?reference WHERE {
?id rdf:type arca:CartesianLocation .
?id arca:isContextOf ?p .
    ?p arca:id ?userid .
?id arca:x ?x .
?id arca:y ?y .
?id arca:z ?z .
?id arca:reference ?reference .
?id arca:timestamp ?timestamp .}
```

Listing 6.2: Example SPARQL query

To facilitate the localization process on the server, a context processor bundle has been implemented that acts as a context consumer for WLAN measurement context information, and as a context source for location information. To provide input to this processor, a `WifiscanContextSource` bundle has been implemented to retrieve WLAN measurement data from the respective MUC, and provide it to subscribers using the specified ontology. A `LocationContextConsumer` bundle subscribes to the context processor to receive location information updates, and provides this information to the respective mobile device. Next to these bundles, context sources for all types of context information that the mobile device can detect have been implemented, as well as a number of context processors and context consumers for the context information that is intended for the mobile device. Content is provided by the `ContentProvisioner`, which is implemented as a context processor. After receiving location information, it determines whether the area in which the user is located has changed. If so, it retrieves all the content items that are located in the respective area from its database, and provides these to the mobile device by means of the `ContentConsumer`.

6.3.2. Announcement and Notifications Service

The goal of the ANS is to allow rules to be entered, which describe what message should be sent and to whom, if a specified condition is met. This condition is described as a state of context information, and, therefore, the ANS works in conjunction with the CMS to subscribe to context sources that can provide the needed context information.

As stated in [89], the ANS had not been implemented fully and generically. In a number of classes, certain AMIGO specific data is hard coded into the source code, and only the information objects, or "concepts", and relations between these concepts that are used in the tutorials are fully developed. Besides these issues, we encountered multiple versions of the compiled code and source code. An overview of these versions is presented in Table 6.2:

Version	Source	Compiled	Description
1.0.0	•	•	Downloadable from the main AMIGO website [90], contains a lot of unfinished code.
1.1.0	•	•	Downloadable from the main AMIGO repository, not working with the latest source code version of the client.
1.1.1		•	Only accessible through the update location that has been specified in the manifest of the compiled ANS 1.1.0 bundle. The accompanying source, however, still is version 1.1.0.

Table 6.2.: Available implementation versions of the ANS

Since the source code versions of the ANS were not working properly with the ANS client, we have decompiled the compiled ANS version 1.1.1 and solved the problems that occurred due to the decompilation using the 1.1.0 source code. Using the obtained source code, we mapped out the process and information flow for the initialization of the ANS, the subscribing of a rule, the starting of a rule, and the processing of an incoming event that fires the rule. During this process, we identified in which classes and methods AMIGO specific code existed. Where possible, we changed the code to generic code, in the other cases, we rerouted the calls to a single reference class. This facilitates the addition of new concepts or relations by changing a single class, in addition to implementing the respective new subclass of the `Concept` or `Relation` class.

Besides these improvements, we have also expanded the functionality by allowing rules to be set with a "greater than" or "less than" operand in the relation. For example, this enables a rule that triggers a notification if the temperature is *above* 25°C. An example rule that triggers a notification "It's hot'!' if Alice is near "Item1" and the temperature is above 25°C is shown in Listing 6.3.

```
<?xml version="1.0" encoding="UTF-8"?>
<ECARule xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="ECAXML.xsd">
   <upon>
      <and>
         <operand_a>
            <event name="isLocatedNear"
             state_transition="EnterTrue">
               <param>
                  <literal value="User.Alice"/>
               </param>
               <param>
                  <literal value="Item1"/>
               </param>
```

```
            </event>
         </operand_a>
         <operand_b>
            <event name="hasLocationTemperature"
             state_transition="EnterTrue">
               <param>
                  <literal value="User.Alice"/>
               </param>
               <param>
                  <greaterthan>
                  <operand_a>
                     <literal value="hasLocationTemperature"/>
                  </operand_a>
                  <operand_b>
                     <literal value="25" />
                  </operand_b>

                  </greaterthan>
               </param>
            </event>
         </operand_b>
      </and>
   </upon>
   <do>
      <notification name="Notify">
         <param>
            <literal value="Alice"/>
         </param>
         <param>
            <literal value="It's hot!"/>
         </param>
      </notification>
   </do>
   <lifetime value="always"/>
</ECARule>
```

Listing 6.3: Example ECA-DL rule

The used notation is the XML-based variant of ECA-DL, a domain specific language developed with the purpose of specifying event-condition-action (ECA) rules to be used in context-awareness scenarios [91]. These rules are translated into the syntax used by JESS [92], the rule engine that is used by the ANS to derive conclusions, based on the received rules and facts. More information about the rule syntax can be found in [89].

Besides the improvements to the ANS, we have developed a client bundle that is capable

of registering rules, and forwarding received notifications to the mobile device. The latter is accomplished by placing the notifications in a dedicated MUC, which is read out by the Context Server. This server application maintains all the connections to mobile devices, and uses these connections to pass the notification on to the corresponding mobile device.

6.4. Augmented reality component

The goal of this component is to provide a graphical user interface (GUI) to the user, by facilitating the augmented reality view, other views, such as the view of a map of the area, and means to interact with the framework. In order to provide these views, and the information within these views, the component registers `EventHandlers` to a number of according `EventDispatchers`:

From the context-awareness component, it uses the following `EventDispatchers`:

- `ContentEventDispatcher`: provides `ContentEvents`, which represent items in the user's surroundings. These items are stored in a database on the CIS and provided, based on the estimated location of the user, by the `ContentProvisioner` processor.
- `NotificationEventDispatcher`: provides `NotificationEvents`, that contain notifications that have been received from the CIS.
- `LocationEventDispatcher`: provides location estimates, which are created by the localization process on the mobile device, or received from the CIS when the localization process is performed on a remote system.

The following `EventDispatchers` are used from the OS specific interface layer:

- `OrientationEventDispatcher`: provides the three-axis orientation of the mobile device, based on the sensors measurements of the orientation sensor.
- `AccelerometerEventDispatcher`: provides the acceleration of the mobile device along the three axes in m/s^2, based on the measurements of the accelerometer.

Orientation
As described in Section 2.2.3, location and orientation are needed to determine the user's field of view. While the orientation sensor of the G1 phone provides a three-axis orientation sensor, these axes are based on the upright, vertical, position of the device, i.e., with the screen in *portrait* mode. Figure 6.3 shows the different forms of rotation, along with the axis around which the rotation takes place and the name of the type of rotation.

Figure 6.3.: Three axes of the orientation sensor (based on [93])

While holding the device in the horizontal position, or *landscape* mode, the azimuth will remain to describe the angle between the current direction and the Magnetic North, given as an angle around the x-axis. However, tilting the device backwards or forwards (pitch), or rotating the device sideways (roll), will result in a rotation around both the y-axis and the z-axis simultaneously. To overcome this problem, we have used the accelerometer to detect the pitch and roll rotation. Figure 6.4 shows the axes of the accelerometer.

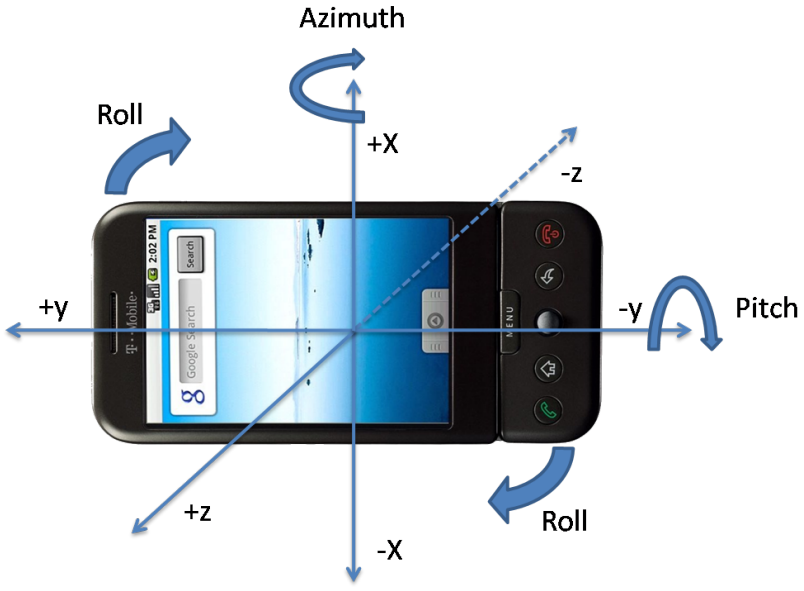

Figure 6.4.: Three axes of the accelerometer

In the horizontal position, the device experiences an acceleration of $0m/s^2$ on the y-axis and z-axis, and acceleration of -9.81m/s^2 on the x-axis, in accordance with the force of gravity. When performing a roll rotation on the device, the device experiences an increase (>0) of the force on the y-axis for a counterclockwise rotation, and a decrease (<0) of the force on the y-axis for a clockwise rotation. Performing a pitch rotation, the force on the z-axis will increase (>0) when the device is tilted forward, and it will decrease (<0) when the device is tilted backwards. The ratio of the increase or decrease can be used to calculate the difference in angle for the roll rotation with Equation 6.1 and for the pitch rotation with Equation 6.2:

$$\phi_{roll} = \frac{F_y}{9.81} * 90^\circ \text{ where } F_y = \text{ force on y-axis} \tag{6.1}$$

$$\phi_{pitch} = \frac{F_z}{9.81} * 90^\circ \text{ where } F_z = \text{ force on z-axis} \tag{6.2}$$

The calculated values for ϕ_{pitch} and ϕ_{roll} can be used to translate and rotate the canvas, on which graphics can be drawn, to reflect the changes of the mobile device's orientation. Figures 6.5 and 6.6 show the effects of the rotations on the canvas. By performing the translation and rotation, the drawn horizon will remain horizontal at the height of the actual horizon, if visible from the current field of view.

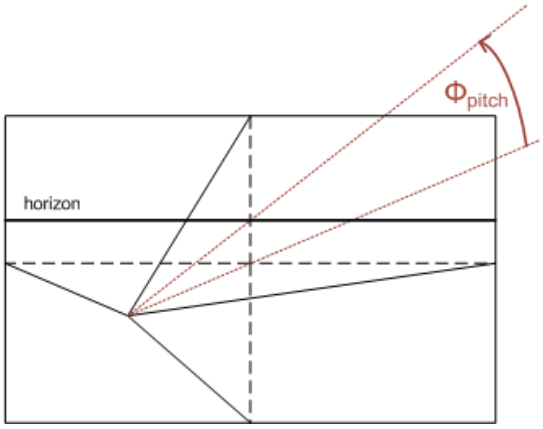

Figure 6.5.: Pitch

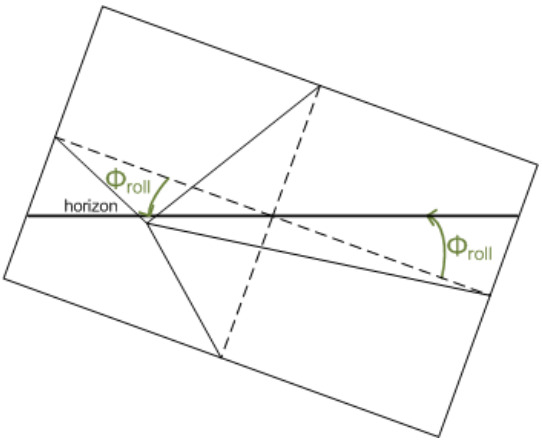

Figure 6.6.: Roll

Influence of the three types of rotation on the field of view

The angle between the current direction of the device and the Magnetic North is represented by $\phi_{azimuth}$. To this angle, a *rotation correction* of ϕ_{rc} is applied, which corresponds to the difference between the Magnetic North and the x-axis of the grid that is used for the coordinates of reference points, content locations and the estimates of the localization process. Figure 6.7 shows a graphical representation of the rotation correction angle.

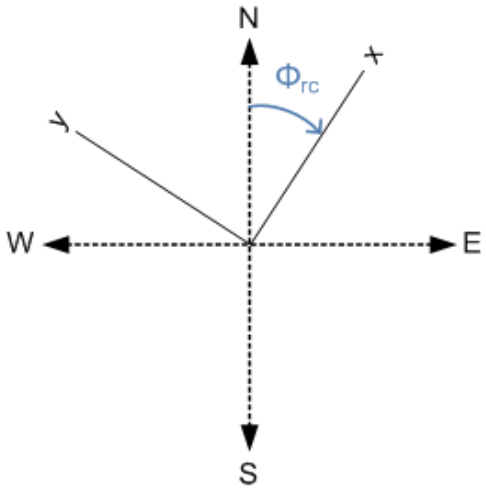

Figure 6.7.: Rotation correction

The relative rotation can be calculated using:

$$\phi_{relative\ rotation} = \phi_{azimuth} + \phi_{rc} \tag{6.3}$$

To determine whether a content object is visible, the difference between the relative rotation angle and the angle between the x-axis and the line between the current location and the location of the content object is calculated. If this angle difference ϕ_Δ is less than the horizontal view angle of the device, the object is visible, and the difference can be used to calculate the position on the screen where the content object representation should be drawn. Figure 6.8 shows an example with the view angles for the G1 mobile phone.

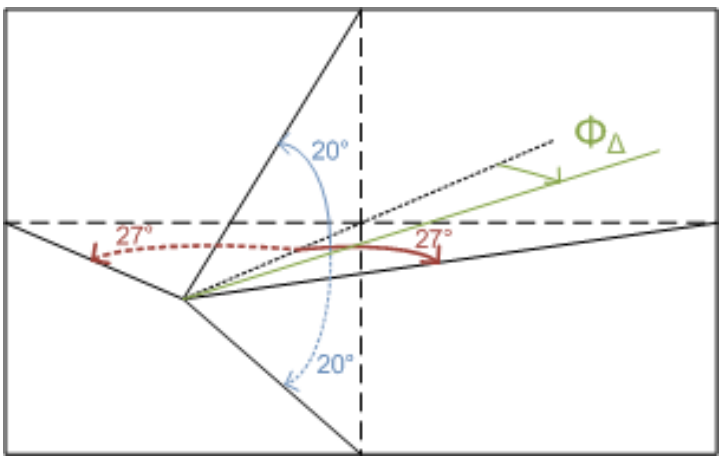

Figure 6.8.: View angles for the G1 mobile phone

If the angle difference ϕ_Δ is above -27° and below 27°, the ratio between the angle difference and the horizontal view angle can be used to calculate the rendering position on the horizon:

$$draw\ coordinate\ x = \frac{\phi_\Delta}{27°} * screen\ width \tag{6.4}$$

For content that also has an altitude, i.e., a non-zero z-coordinate, a similar calculation can be used to determine the drawing position above or below the horizon.

Content
The `ContentEvent` class provides the basis for all classes that represent content, by providing fields for the name, content type, a timestamp, the ID of the user for whom this content was intended, and the location. Figure 6.9 shows the class diagram of the `ContentEvent` class and the subclasses.

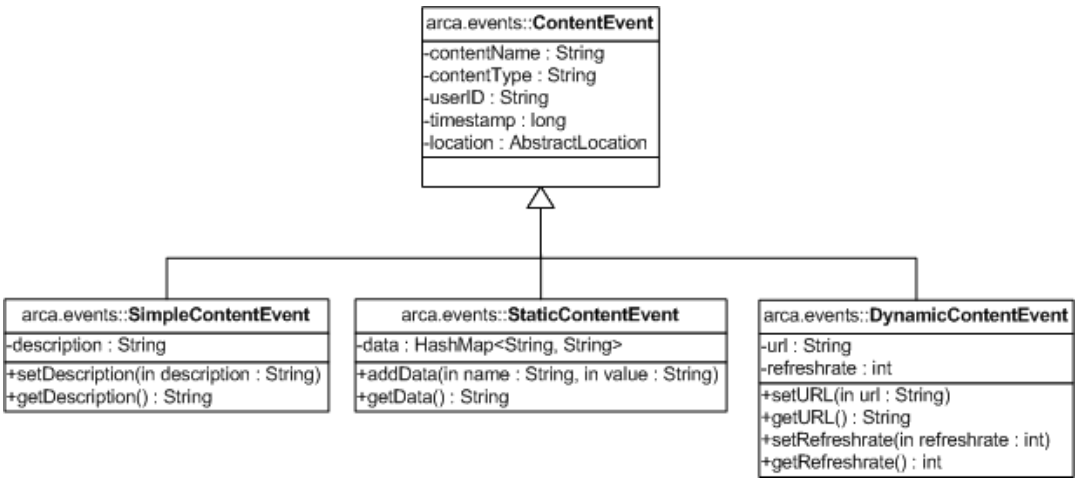

Figure 6.9.: The ContentEvent class and its subclasses

There are three types of content defined, each implemented by a specific subclass of `ContentEvent`:

- `SimpleContentEvent`: represents the most basic content type, with a single description field, for instance, an object representing a ticket machine in the train station.
- `StaticContentEvent`: represents content by providing configurable name-value pairs, i.e., an object representing a movie poster, with values for the movie title, rating, synopsis, and a link to the website.
- `DynamicContentEvent`: represents changing content by defining an URL to the source of the content information and the rate at which to refresh the information from the source, for instance an object corresponding to the departure

timetable of a train station platform, which should be refreshed regularly to keep the information up to date.

All received `ContentEvents` are stored in the `ContentContainer`, along with a `ScreenLocation`. This object stores the calculated coordinates at which the graphical representation of the content object is drawn. When a user clicks on the screen, the coordinates of the point where the user clicked is compared with these `ScreenLocations` to determine which item was clicked.

For every redraw of the screen, the most recent direction, pitch, and roll values are used to calculate which content objects are visible within the field of view, and at what location of the screen they should be rendered.

Class structure
Figure 6.10 shows the basic structure of the AR component.

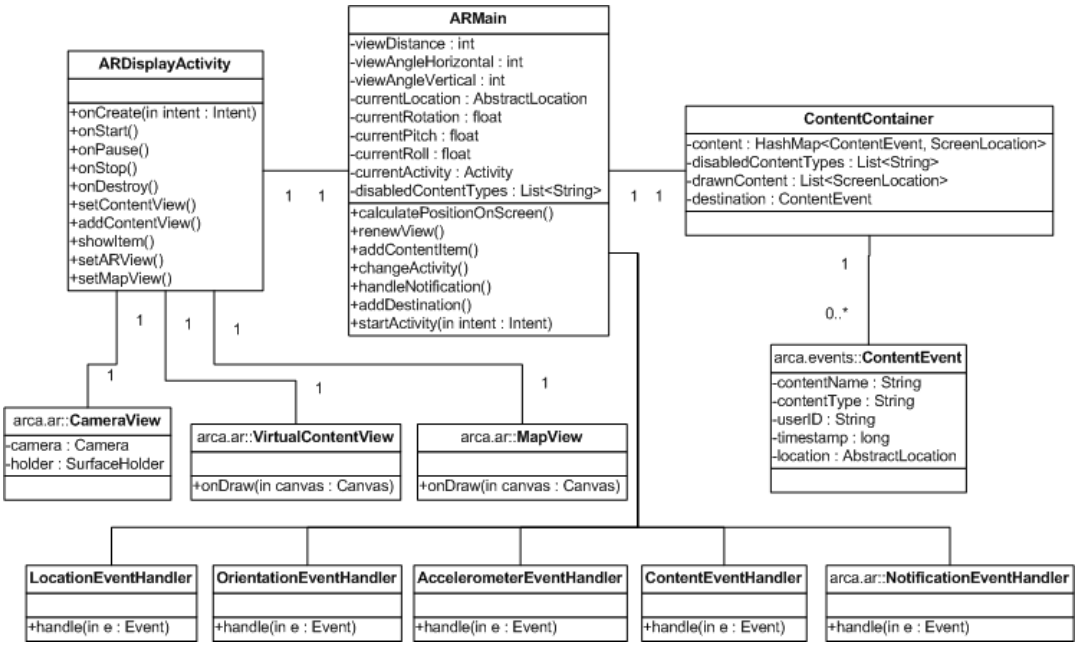

Figure 6.10.: The basic structure of the AR component

The `ARMain` class provides the generic functionality for the AR component, such as the storage and processing of events that have been received from the CA component and the OS specific interface layer, and calculations to determine the current orientation of the mobile device. This information and functionality is provided to the `ARDisplayActivity`, which creates and maintains the different `Views` of the application:

- `CameraView`: provides the camera feed of the mobile device.
- `VirtualContentView`: provides an overlay for the `CameraView`, on which information can be drawn, along with a number of basic drawing functions, i.e., methods to draw content, a horizon, a compass, etc.
- `MapView`: provides a map of the area the user is currently in, along with representations of the user's current location and direction, and the content items in the area.

Because of Android's specific methods of creating graphical user interfaces (GUIs) with `Views`, and, therefore, the absence of the standard Java GUI component called Swing [94], the GUI has been implemented using the aforementioned `View` subclasses. However, by placing all the GUI support functions in the `ARMain` class, the quick development of a GUI on a different platform is facilitated.

6.5. Application deployment

This section discusses how the framework can be used to support a context-aware augmented reality application. Adapting the framework to support application-specific functionality is facilitated at both the mobile device and the CIS.

6.5.1. Resources

The framework on the mobile device can be adapted for a specific application by extending the `Resource` classes. Figure 6.11 shows the structure of these classes.

A specific subclass of the abstract `AResources` class can be retrieved by the static `getInstance()` method, enabling the existence of only a single instance of this class, ensuring the lowest amount of system memory is used. Which specific subclass is created depends on what type of `IResourceFactory` implementation has been set using the `setResourceFactory()` method. The `AResources` class holds the references to the various `Resource` classes:

- `ABitmapResource`: maintains the various graphical elements, `Bitmaps`, that are drawn on the screen by the `View` classes. They can be retrieved by name or type, and customization can be achieved by replacing existing references to `Bitmaps` by new ones.
- `AMapResource`: provides a `MapHolder` object that corresponds with the current location of the user. The `MapHolder` contains a map that represent the current area the user is located in, the height and width of the map with respect to the coordinate system that is used (see Section 6.2), and the rotation correction for this map (see Section 6.4).

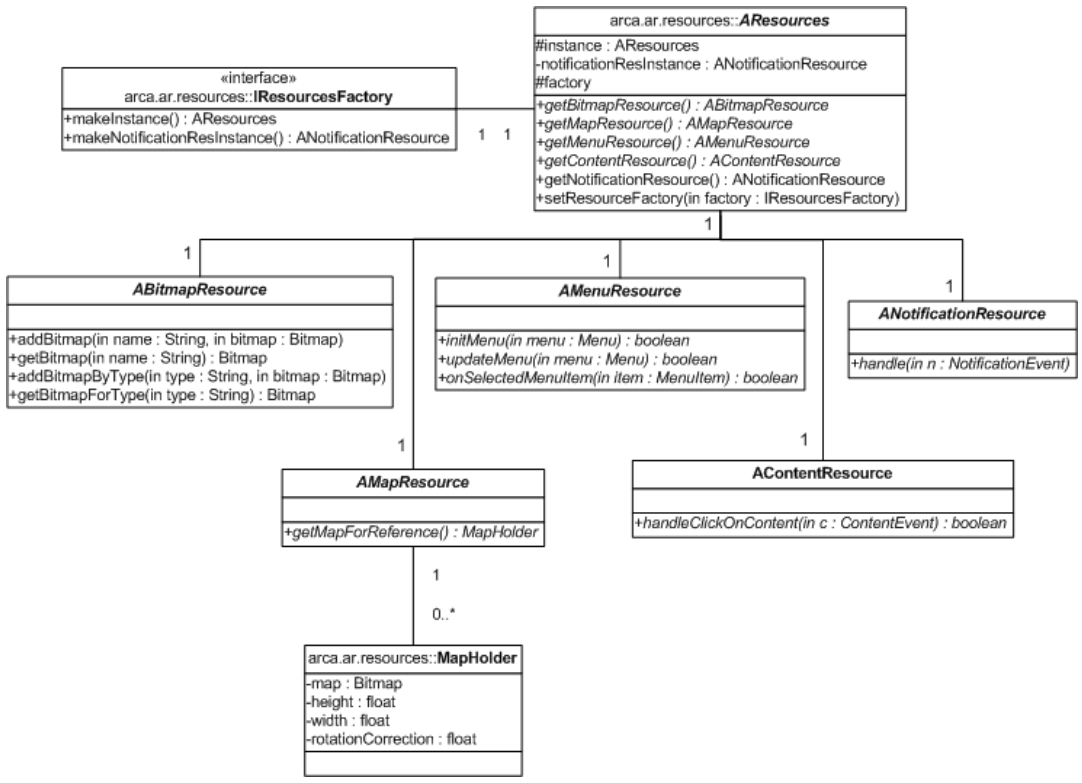

Figure 6.11.: Application-specific resources for the framework

- `AMenuResource`: creates and updates the menu of the GUI, facilitating application-specific menu items. It also handles clicks on the menu items, which can result in the invocation of application-specific functionality.
- `AContentResource`: facilitates application-specific handling of clicks on content representations of the `VirtualContentView`. By default, the displayed text switches from the name of the content to the description and back, but for example, clicking on content items that contain an URL could result in the opening of a web browser to display the corresponding website.
- `ANotificationResource`: handles incoming notifications from the CIS. By default, a dialog containing the message of the notification is shown, but based on the type of message, application-specific functionality can be invoked.

The framework uses the application-specific resources by calling the `getInstance()` method of the `AResources` class, retrieving the needed `Resource` class, and calling its methods to acquire the needed information or invoke the required functionality.

Furthermore, the GUI can be changed by subclassing the `VirtualContentView` or `MapView` (see Section 6.4), and adding the invocation of extra drawing functions to the `onDraw()` method. These subclasses can be used by subclassing the `ARDisplayActivity`

class and overriding its `setARView()` and `setMapView()` methods to use the new `View` subclasses.

6.5.2. Server components

The CIS facilitates application-specific functionality in a number of ways:

- New content items can be added to the database that the `ContentProvisioner` uses to retrieve content from, based on the user's location. When deploying an application in a new area, new content items, with the reference of the deployment area, can be added. The `ContentProvisioner` will automatically provide these content items when a user's location estimate contains the according reference. If the new content items are of a new type of content, changes might have to be made to the resources on the mobile device, as explained in Section 6.5.1.
- By implementing new context sources, either as a standalone context source or as part of a context processor, new context information can be provided to both the CIS and the mobile device (see Section 6.3.1 for more information about the implementation of context sources).
- New rules can be added to the ANS to enable the CIS to respond to new kinds of changes in the context of the user. If the new rules include new types of context information, changes will have to be made to the ANS, as described in Section 6.3.2. The notifications that are fired, based on the new rules, might also require changes in the notification handling part of the resources on the mobile device (See Section 6.5.1).

Chapter 7.

Evaluation

To evaluate the framework, we have used it to develop a prototype application. This application provides information and navigation functionality for train station environments.

Section 7.1 describes the development of the application using the framework and Section 7.2 discusses the deployment of the application and evaluation of the framework.

7.1. Application development

Using the methods that have been described in Section 6.5, the framework has been used to develop a train station navigation application, called the NS Navigator. The requirements for this specific application are presented in Section 3.3, and have been used to determine the generic requirements for the framework. These requirements have been determined in cooperation with stakeholders from the Nederlandse Spoorwegen (NS), the Dutch railway company.

To test the application, we have used a simulated train station environment, called the NS Trefpunt (Dutch for "meeting point"), which functions as a venue for conferences and meetings. All aspects of a traveller's train journey are present in this environment, including trains, platforms, ticket machines, shops, waiting areas, etc. Figure 7.1 shows a photo of the NS Trefpunt.

Figure 7.1.: Test environment NS Trefpunt

7.1.1. Content

Firstly, we have identified what objects are in the area, where they are placed, and how they could be represented in the application. Using a map of the area, shown in Figure 7.2, and a Cartesian coordinate system, with its origin at the top left corner of the map, content items can now be placed in the database of the `ContentProvisioner`.

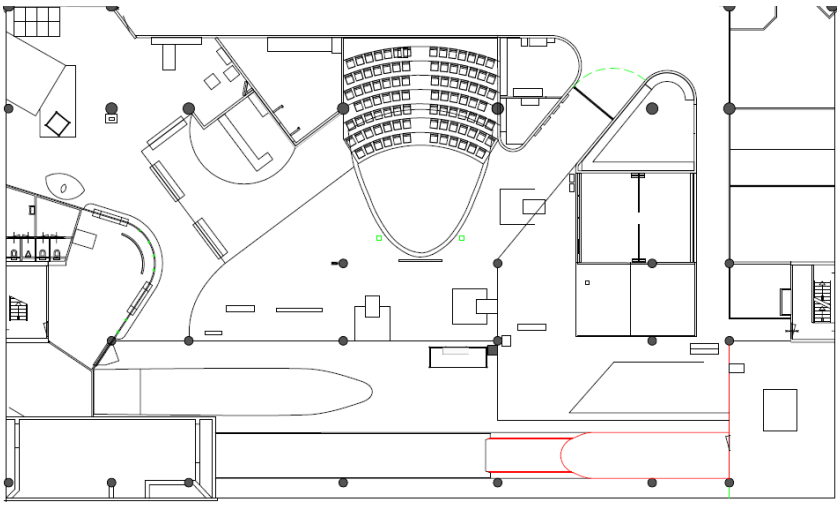

Figure 7.2.: Map of the NS Trefpunt

Table 7.1 shows some examples of content items in the "simple content" table of the database. For displaying purposes, the contents of the description field has been left out, but this field could contain textual descriptions of the content item.

When a user is determined to be in the test area, the `ContentProvisioner` automatically provides the content items to the mobile device. As explained in Section 6.4, the content items are stored in the `ContentContainer` on the mobile device, and, for each field of view update, the framework determines which items are viewable. To support the drawing of content item representations that are specific

Reference	Name	ID	Type	Description	X	Y	Z
NSTrefpunt	Waiting area	24	ns_waiting_area	..	31.5	24	0
NSTrefpunt	Ticket machine	25	ns_tickets	..	43.5	19	0
NSTrefpunt	Platform 1b	26	ns_platform	..	43.5	31	0
NSTrefpunt	AH to go	27	item_ahtogo	..	8.5	22.5	0
..	..	..	..	..	..	..	..

Table 7.1.: Content items in the *t_simple_content* table of the database

for the NS Navigator application, the `ABitmapResource` class is extended by the `NSNavBitmapResource` class, which adds `Bitmaps` for all the different types of content items that are present. Furthermore, the `VirtualContentView` has been subclassed by the `NSNavVirtualContentView`. This class changes the color of the panel, along with the color of the text within the panel that is used to display the name or description of a visible content item. An example of a content item representation for the platform of the ICE train is shown in Figure 7.3.

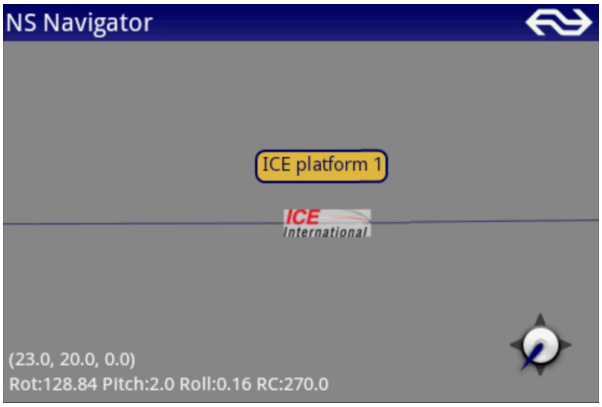

Figure 7.3.: Content item for the ICE train

As Figure 7.3 shows, in addition to the adjustments to the content item representations, a top bar with the application title and NS logo has been added to give it a recognizable "look and feel". This has been accomplished by overriding the `onDraw()` method of the `VirtualContentView` and performing the application-specific drawing operations, as well as invoking the original `onDraw()` method of the superclass.

Clicks on the content item representations are handled by the `NSNavContentResource` class, which is a subclass of the abstract `AContentResource` class. Clicks on content item representations that contain a description with more than 150 characters, or with an HTML markup, open a custom dialog, as shown in Figure 7.4.

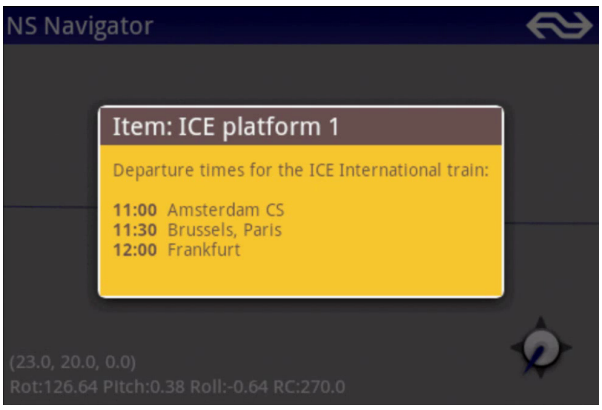

Figure 7.4.: Dialog showing the departure times for the ICE train

7.1.2. Notifications

Incoming notifications are processed by the `NSNavNotificationResource`. By default, a dialog is shown with the message that is contained in the notification. However, application-specific processing is possible, as Figure 7.5 shows.

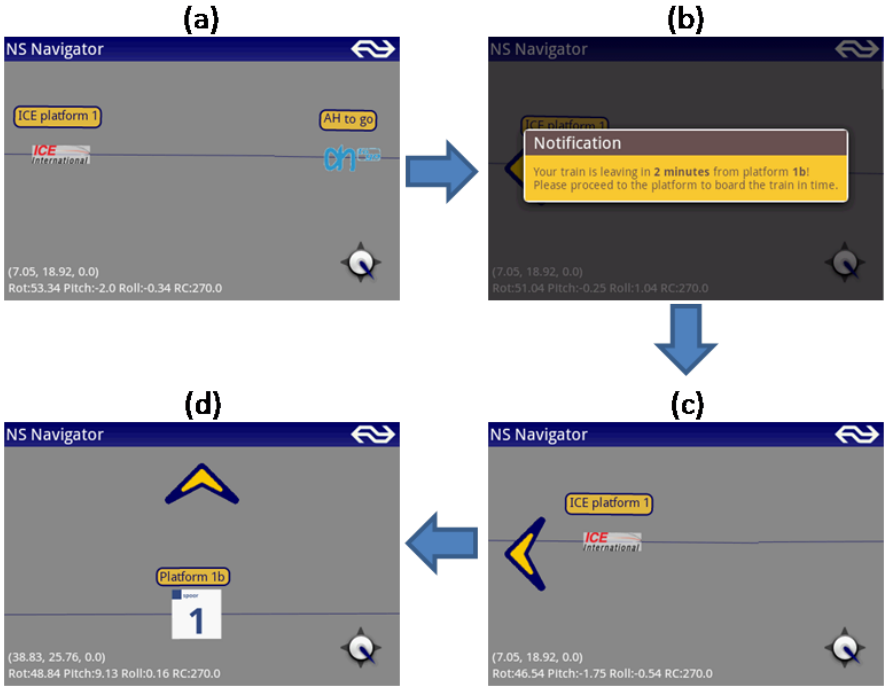

Figure 7.5.: Application specific handling of a notification

Figure 7.5 (a) shows the view before the notification is shown, containing two content items, namely the ICE platform, and a supermarket called “AH to go”. When a notification is received about the time that is left before the train leaves, three actions are

performed:

1. The notification message is shown in a dialog, shown in Figure 7.5 (b).
2. Non-train related content items are disabled, as shown by the absence of the "AH to go" in Figure 7.5 (c).
3. The platform, to which the user should go, is set as the current destination, indicated by the arrows in Figure 7.5 (c) and (d).

7.1.3. Map

To provide the map of the NS Trefpunt, the `AMapResource` class has been subclassed by the `NSNavMapResource` class, which has a `MapHolder` that contains the map and values for that specific area. This `MapHolder` is used by the `NSNavMapView`, a subclass of the `MapView`, to show the map of the NS Trefpunt in the right aspect ratio. Figure 7.6 shows the `NSNavMapView` of the NS Navigator application.

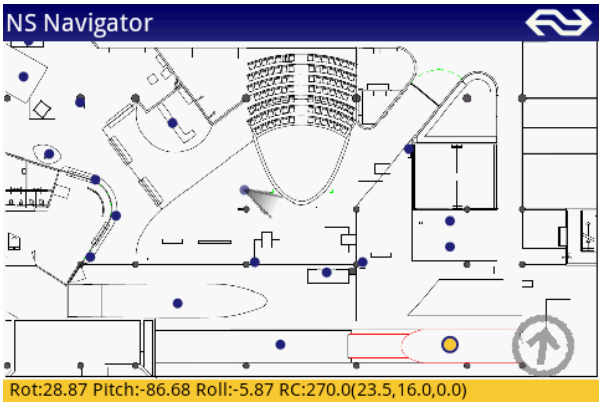

Figure 7.6.: Screenshot of the NSNavigator's Mapview

The user's current location is represented by a lightblue dot and his orientation is represented by a triangle that originates from the location representation. Content items are indicated by blue dots at their corresponding location on the map, whereby, for the current destination, this dot is replaced by a yellow circle. An arrow at the bottom right part of the screen shows the direction towards the destination, based on the mobile device's current orientation. As with the `NSNavVirtualContentView`, the `NSNavMapView`'s `onDraw()` method adds the blue bar at the top of the screen of the NS "look and feel".

7.2. Application evaluation

For the NS Navigator to work, it was necessary to build the radio map for the localization process. An already available wireless network infrastructure was used, providing an average coverage of eight access points. One access point was added to enable direct communication with the CIS. Due to time limitations, this map has been built for only part of the area. The grey box in Figure 7.7 shows this part.

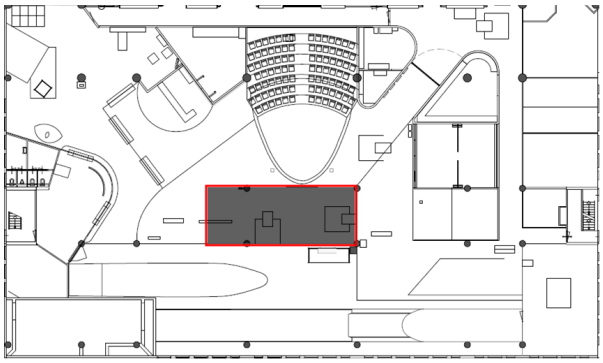

Figure 7.7.: Radio mapped area within the NS Trefpunt

In this area of 5 by 15 meters, 100 signal strength measurements have been taken along a grid with a spacing of 1 meter. These measurements were performed by the G1 mobile phone running Android and provide a radio map database with 96 reference points. This database was pulled from the mobile device and placed on the CIS. As indicated in Section 6.2, for an area with 96 reference points and an average access point coverage of eight, performing the localization process on the CIS, as opposed to running the process on the mobile device, provides result in the shortest amount of time possible.

The used localization technique, the Horus WLAN Location Determination System [55], has an average accuracy of 86 - 132 cm for 90% of the estimations [76], with a maximal error of 289 cm [55]. During the evaluation, we experienced similar results, with the maximal error varying around one to two meters. Given a localization error of two meters and the horizontal viewing angle of the G1 mobile phone of 27°, the minimal distance x between the mobile device and the content object, in order for the content object to represented in the `VirtualContentView`, can be calculated using Equation 7.1:

$$\tan 63° = \frac{x}{2m} \rightarrow x = 3.9m \tag{7.1}$$

Figure 7.8 shows a graphical representation of this calculation.

Besides the errors introduced by incorrect location estimates, wrong drawing of content representations can also occur due to errors in the process of determining the device's orientation. As described in Section 6.4, determination of the orientation is based on

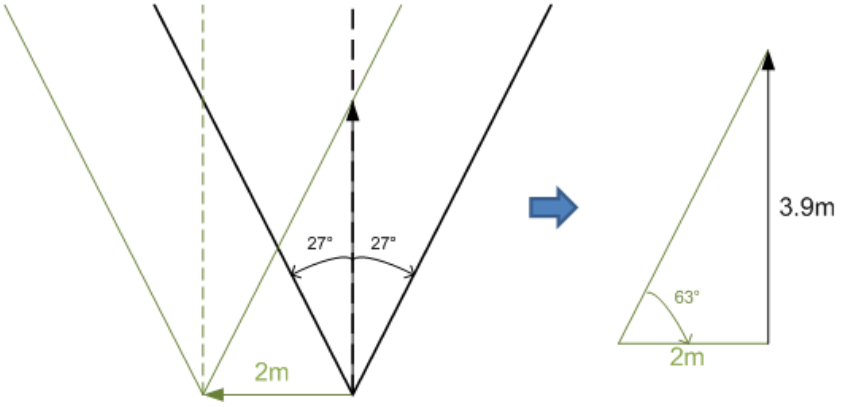

Figure 7.8.: Calculation of the minimal content object distance

values from the accelerometer and the digital compass. If no or only small movements are applied to the mobile device, the output of the accelerometer is solely based on the force of gravity. Therefore, this output provides a steady result, introducing no or negligible errors to the determination of the orientation.

The digital compass, however, uses the Earth's magnetic field to determine the direction of the mobile device, and, since this field is weak, the digital compass readings are vulnerable to distortion [95]. A phenomena called "local attraction" produces a deviation in the Earth's magnetic field, as shown in Figure 7.9, and, consequently, erroneous results in the compass. The source of the magnetic interference could be large steel objects, or objects that explicitly generate magnetic fields, such as mobile phones, microwaves, wireless networking equipment, etc. The amount of distortion of the digital compass's results depend on the distance from the source of the interference, and the amount of interference that it generates. Tests that have been performed during the framework implementation did not show any significant disturbance of the digital compass's readings while using WLAN access points for the localization process and communication, unless the device was held within 20-30 cm from computer equipment or access points.

However, during evaluation of the NS Navigator application we have experienced varying results with the digital compass's readings. In some areas, magnetic interference influenced the digital compass in such a way that providing an augmented reality view became impossible. Since the NS Trefpunt is frequently being used for conferences and meetings, a large number of trusses, audio and visual equipment, etc. are present, which might have been the source for these disturbances. The trains did not appear to be (main) sources of interference for the magnetic field within the NS Trefpunt, but since these were mere mockups, and, therefore, did not contain a chassis, these conclusions might not hold for real trains.

Within the areas where the digital compass was not noticeably disturbed, the location and orientation information was used to provide an accurate augmented reality view, as Figures 7.10 and 7.11 show.

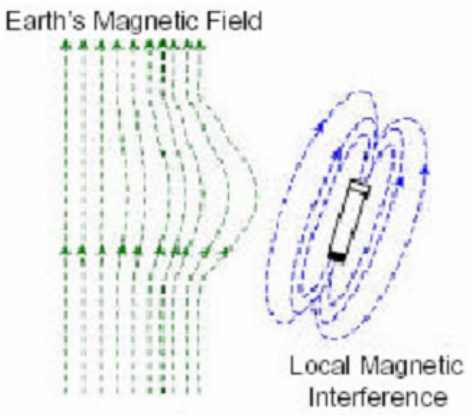

Figure 7.9.: Local attraction influencing the Earth's magnetic field [95]

Figure 7.10.: AH to go

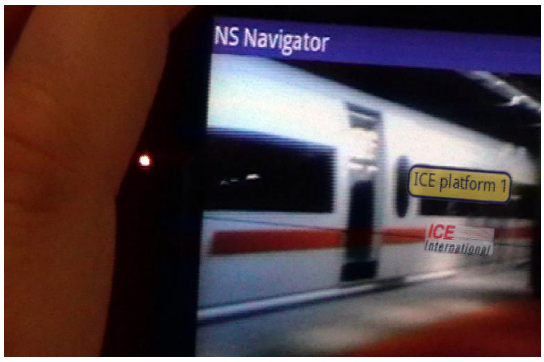

Figure 7.11.: ICE train platform

Screenshots of the NS Navigator application

7.3. Overall conclusion

The evaluation of our framework by implementing the NS Navigator application has shown that it is feasible to provide context-awareness and augmented reality by using the techniques that have been implemented in our framework. Although problem areas have been identified, such as the time that is needed in the offline phase of the localization process and the disturbances in the digital compass's readings, currently available techniques facilitate the support of context-awareness and augmented reality by our framework.

Chapter 8.

Final remarks

This chapter discusses the main contributions of this thesis, gives some relevant conclusions that were drawn, and identifies points where further research is necessary.

Section 8.1 discusses the conclusions that were drawn from the work presented in this thesis, Section 8.2 identifies some future work.

8.1. Conclusions

As described in Section 1.2, the objectives of this thesis are:

1. to identify what techniques need to be combined to facilitate context-aware augmented reality in mobile applications.
2. to determine the capabilities and limitations of context-aware augmented reality in mobile applications using currently available technologies.

Our efforts towards reaching the first objective included a background research into context-awareness, augmented reality and localization techniques, and the identification of requirements for a framework that supports context-aware augmented reality applications.

None of the existing frameworks for augmented reality is capable of providing markerless augmented reality that uses pose tracking, while running on a mobile device. Firstly, although applications exist that provide this functionality, they depend solely on GPS as a localization technique, and, therefore, do not work properly in indoor environments. Secondly, they are not context-aware, i.e., the applications are incapable of acting upon changes in the user's context. For these reasons, we have proposed a new framework, called the ARCA Framework, which consists of two main components: the context-awareness (CA) component and the augmented reality (AR) component. Although they were introduced as parts of a greater whole, the components can be used separately to provide context-awareness or augmented reality to new or existing applications. This

new framework is independent of both the used localization technique and the information domain of the application that is supported by the framework.

The CA component consists of software running on a mobile device, and a server application, called the Context Information Service (CIS). Since a number of middleware systems exist, which can provide support for context-awareness, the CIS integrates the system that best meets our requirements. The server part of the component facilitates the gathering of low-level context information, for instance, from the mobile device, refining low-level context information into higher-level context information, and the provisioning of context information and content to other software and systems, such as the mobile device. Furthermore, rules can be used to specify conditions of the context, along with notifications that are triggered when these conditions are met.

The AR component is implemented in the mobile device and uses the information that is received from the CIS, along with orientation information from the mobile device itself, to provide information about the current field of view to the user using augmented reality. Received notifications are processed to enable the adaptation to changes in the context.

To provide support for indoor environments, we have investigated the available localization techniques, and chosen a probabilistic technique which uses a WLAN access points infrastructure to provide location estimates with the highest possible accuracy. Running the localization process on the mobile device is preferred for privacy. This solution can operate standalone, and provides results faster when used in small areas, running the localization process on the CIS is preferred for devices with limited resources, and provides results faster when used in larger areas. Therefore, our framework supports both approaches.

In order to reach the second objective, we have designed and implemented the framework, based on the identified requirements, and evaluated it by implementing a prototype application, called the NS Navigator. Purpose of this application is to provide information and guidance to users inside a train station, adjusted to their current situation.

Implementation of the prototype application has shown that the currently available technologies are suitable to provide context-aware augmented reality. Although a localization method with a higher accuracy would improve results and enable augmented reality for objects at closer distances, the chosen technique has been used successfully to provide location estimates for the prototype application.

The middleware system that was used to fulfil the role of the CIS offered a decent basis for the context-awareness component, but, due to limitations in the middleware system implementation, time and effort is required to extend its functionality. However, by creating abstract components for context sources and context consumers, implementing a number of context processors based on these components, and improving the middleware system's source code, the system was successfully used to provide context refinement, context and content provisioning, and therefore, to support context-awareness.

The implementation platform, Google Android running on a G1 mobile phone, was used successfully to provide augmented reality. The device's resources are suitable to perform the necessary calculations and graphical rendering. The accelerometer and digital compass of the device can be used to accurately determine its orientation, as long as the compass's readings are not disturbed by magnetic interferences. When deploying an application which uses augmented reality, these possible interferences have to be taken into account.

Overall, our framework provides a structure and support for the development of context-aware augmented reality applications, with an emphasis on generality and extensibility.

8.2. Future work

As the scope of the project was limited to the development of the framework to the integration of new and existing components and techniques to provide the required functionality, various improvements to the used techniques, along with fields for further research, have been identified. The following list presents the topics that have been identified for further research:

- As mentioned in Section 2.3.3, taking reference measurements for the localization process takes a lot of time and effort. Further research could identify means to decrease this time and effort, or ways to automate this process partly or fully.
- The used localization technique compares current measurements with data from reference locations. As identified in Section 5.1, the localization technique could be improved by using the accelerometer to detect movement, and subsequently change the number of reference locations used in the localization process, based on the amount of movement. Likewise, the number of past locations used in the Time-Averaging technique could be adjusted in accordance with the detected amount of movement.
- As discussed in Section 6.3.2, the ANS had not been implemented fully and generically, and, although changes have been made to enable the integration of the ANS into the ARCA Framework, further improvements should increase its usability and extensibility.
- Although almost the complete set of functional requirements of Section 3.3.1 has been implemented in the framework, integrating the generation of routes to a destination, based on map data, was not feasible within the scope of this project.
- Apart from the technical focus of this project, research into the user experience of both context-aware applications and augmented reality applications can provide insights into the deployment and improvement of such applications.

Appendix A.

Class diagrams

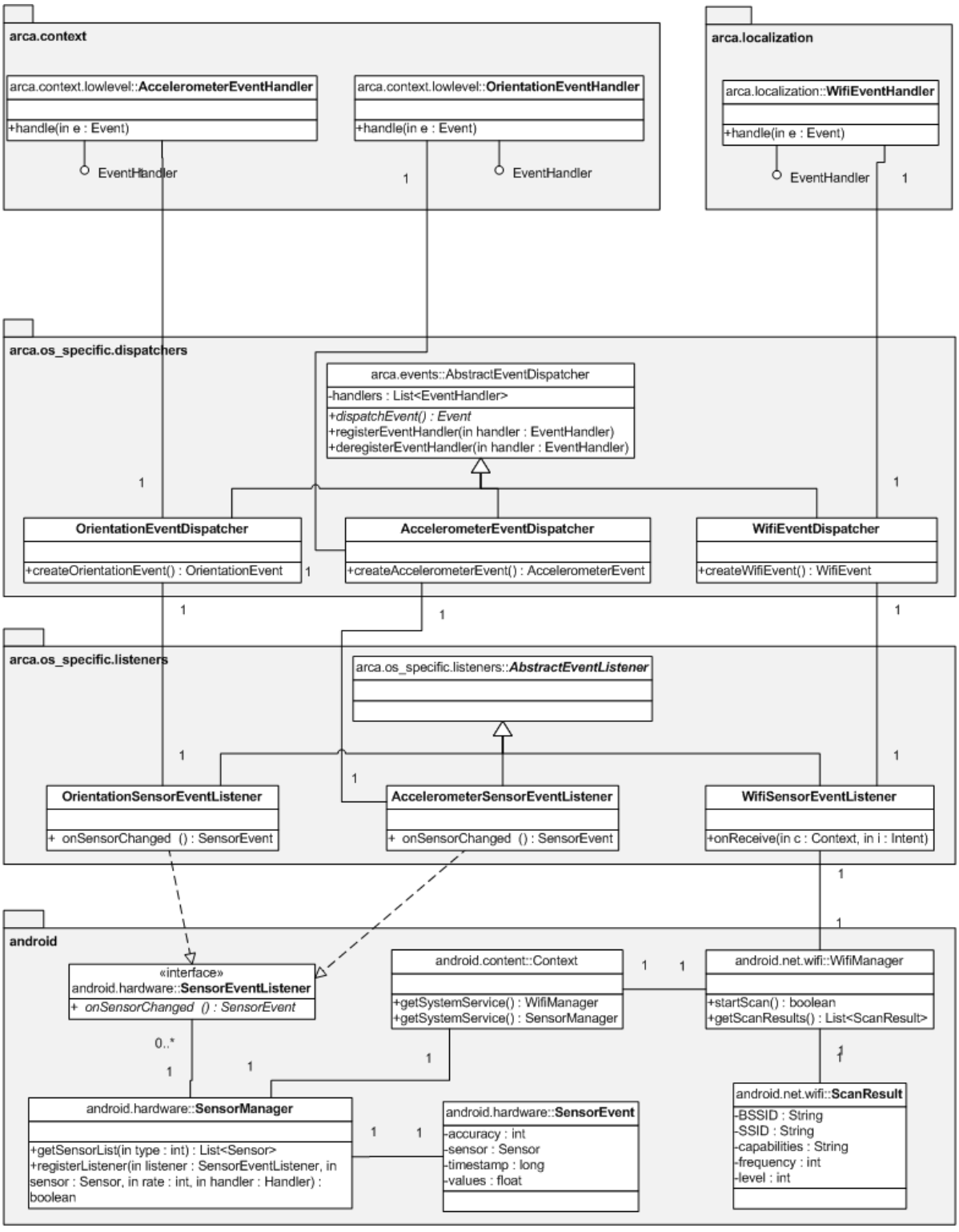

Figure A.1.: Class diagram of interaction between the operating system and the framework

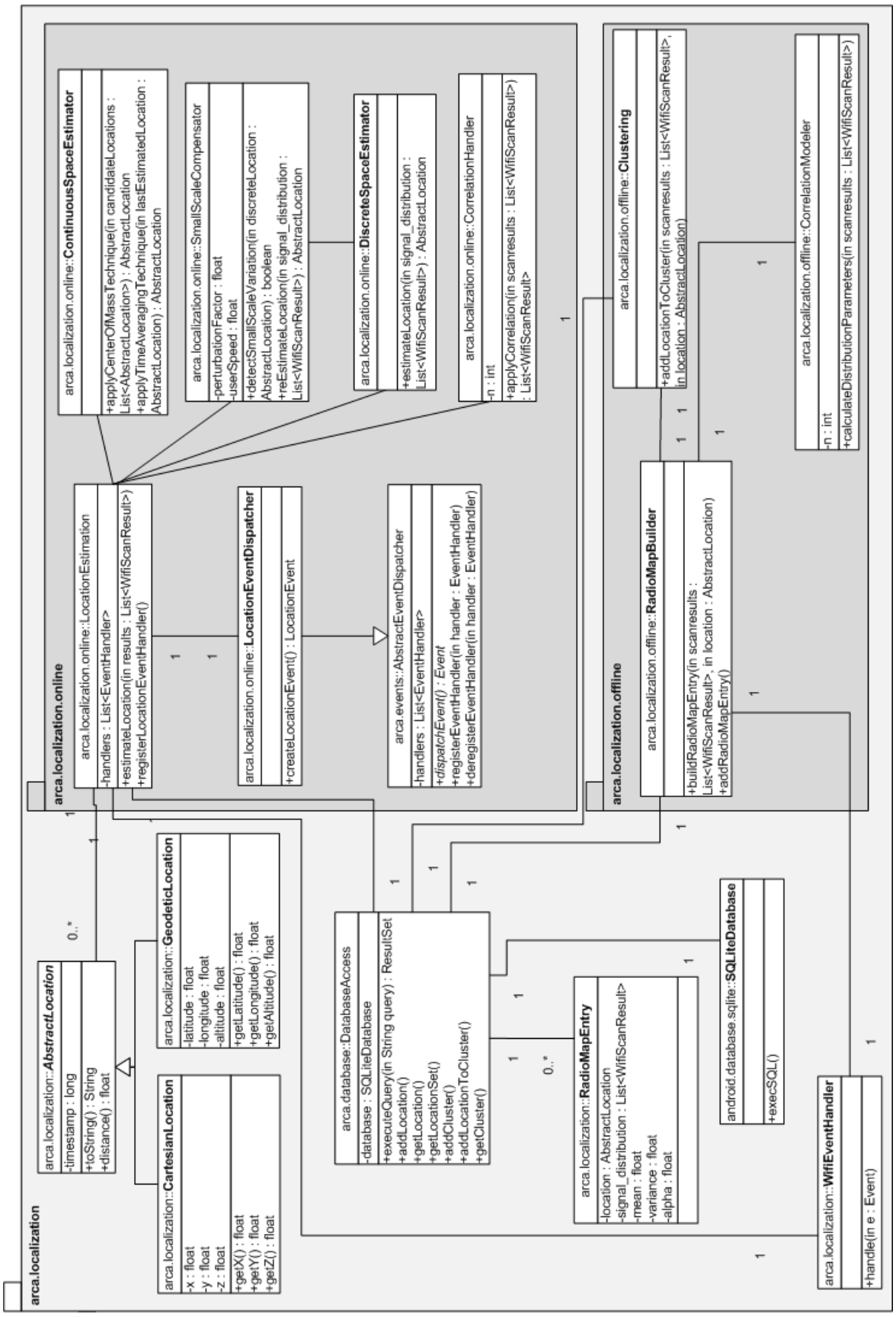

Figure A.2.: Class diagram of the localization component

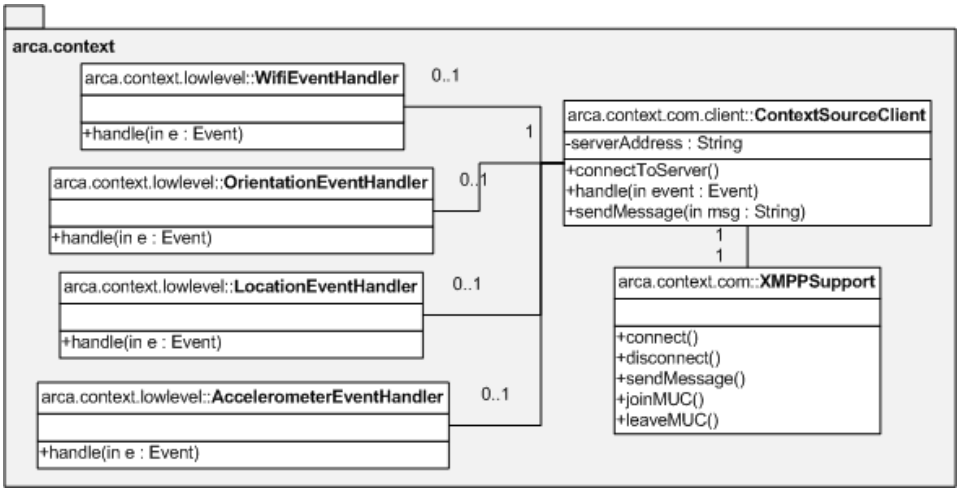

Figure A.3.: Class diagram of the low-level context component

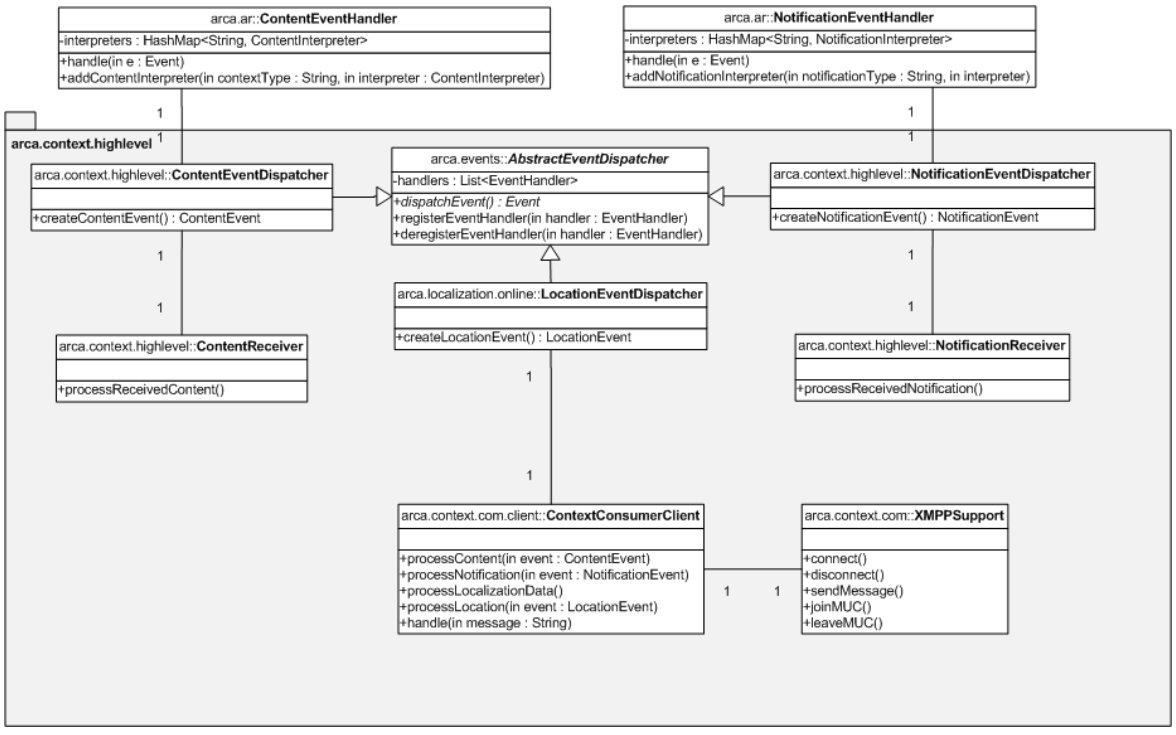

Figure A.4.: Class diagram of the higher-level context component

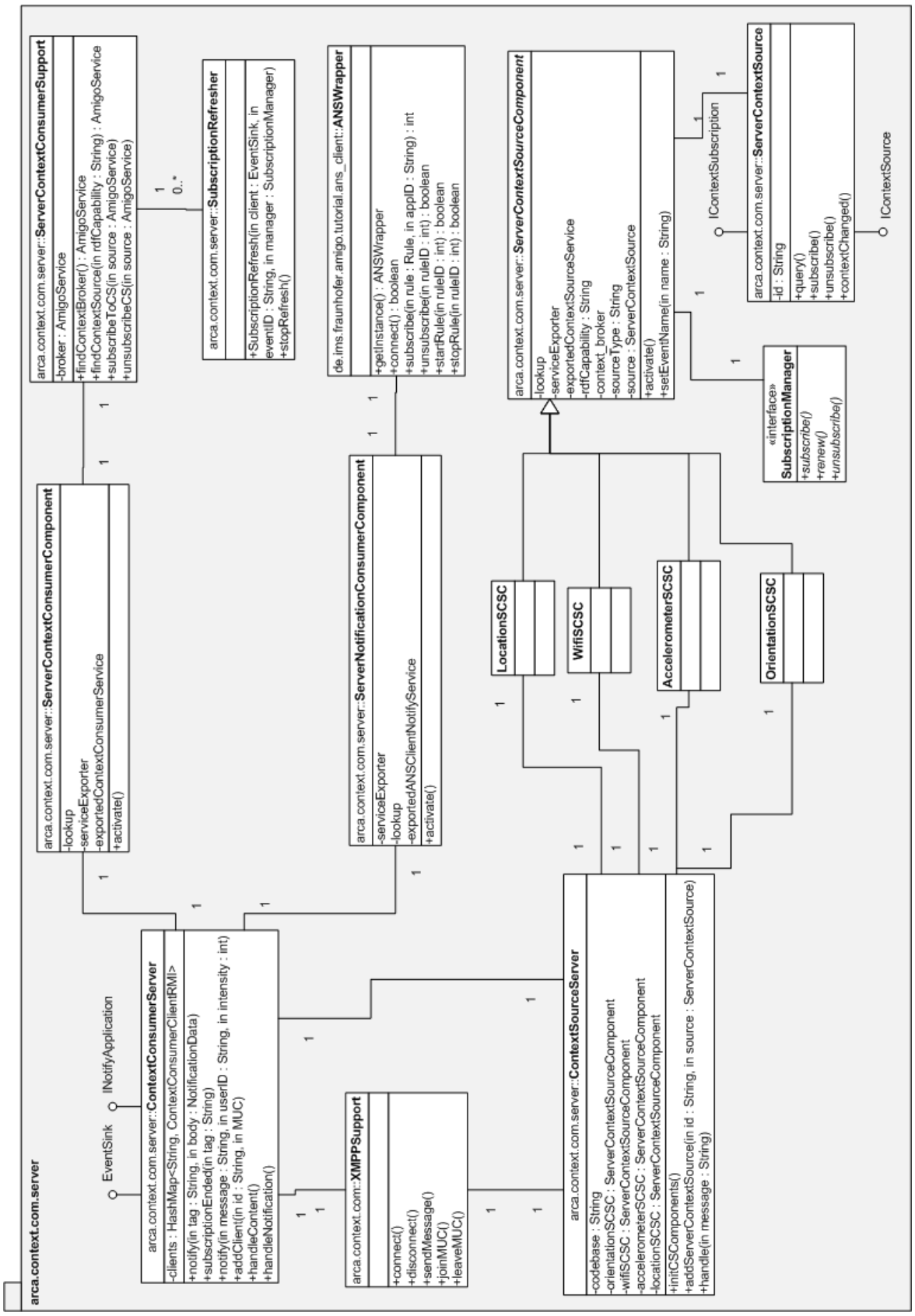

Figure A.5.: Class diagram of the context server

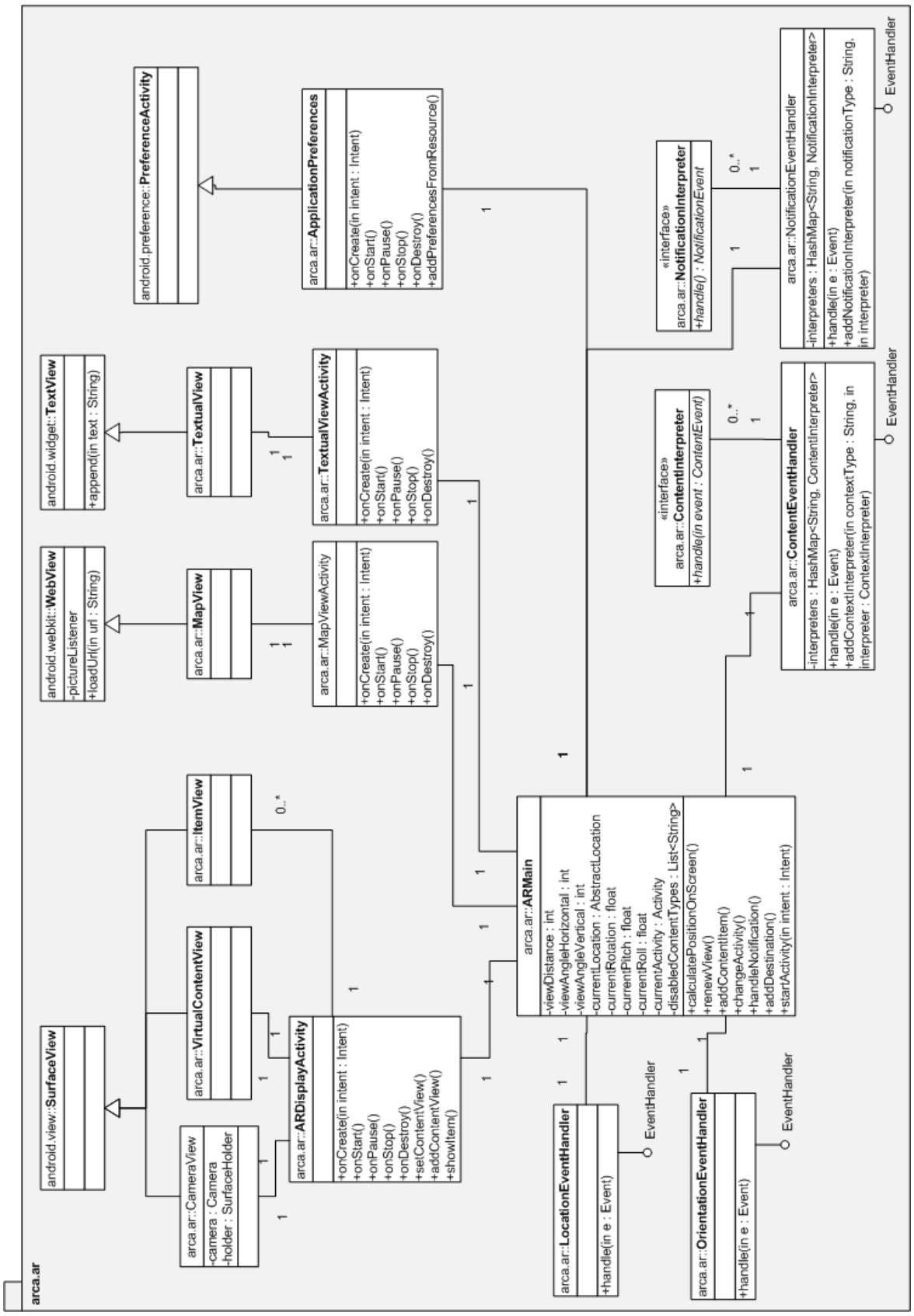

Figure A.6.: Class diagram of the augmented reality component

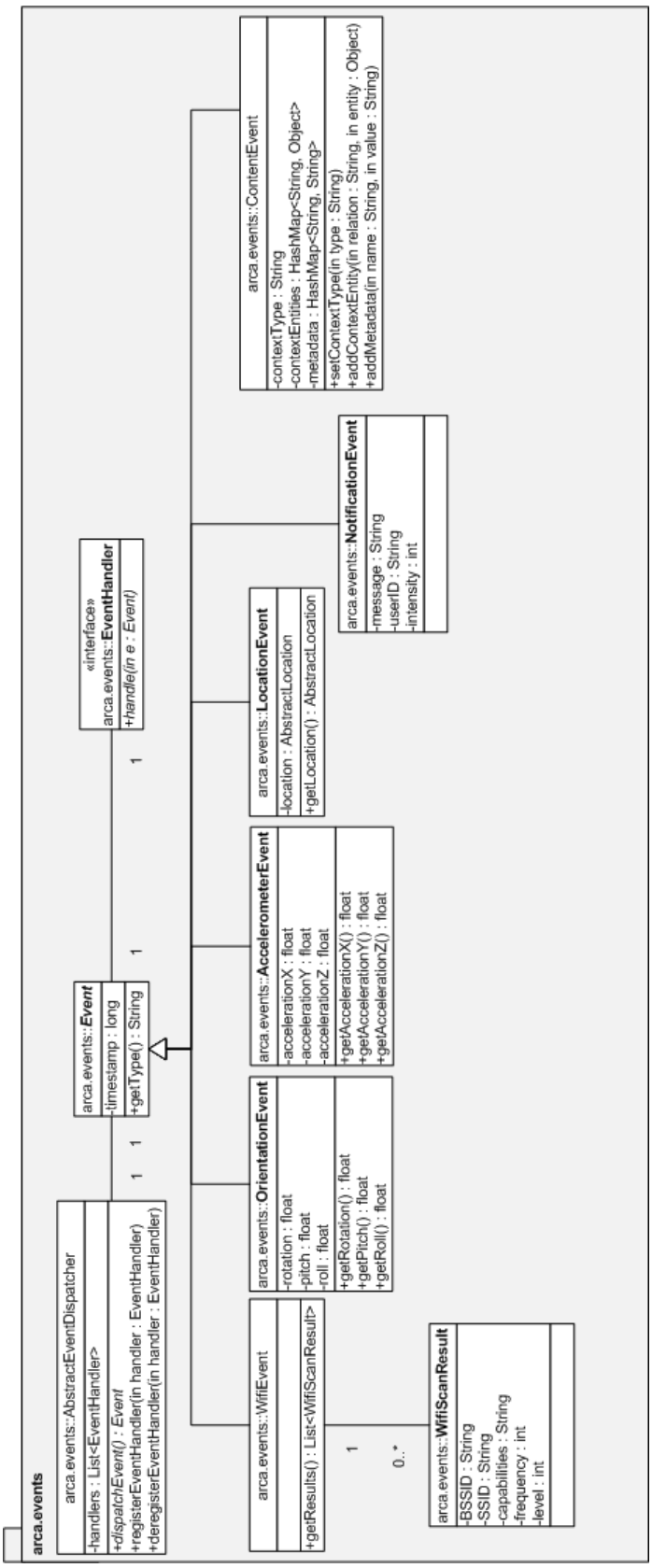

Figure A.7.: Class diagram overview of the different events

Bibliography

[1] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pp. 304–307, Springer-Verlag, 1999.

[2] P. Milgram and F. Kishino, "A taxonomy of mixed reality visual displays," *EICE Transactions on Information Systems*, vol. E-77D, no. 12, 1994.

[3] R. Azuma, "A survey of augmented reality," *Presence: Teleoperators and Virtual Environments,*, vol. 6, pp. 355–385, August 1997.

[4] J. Vallino, "Augmented reality page," September 2006. `http://www.se.rit.edu/~jrv/research/ar/`.

[5] S. Ong, M. Yuan, and A. Nee, "Augmented reality applications in manufacturing: a survey," *International Journal of Production Research*, vol. 46, pp. 2707 – 2742, May 2008.

[6] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 2, no. 4, 2007.

[7] G. U. of Technology and V. U. of Technology, "Studierstube augmented reality project." `http://studierstube.icg.tu-graz.ac.at/`.

[8] B. Bruegge and G. Klinker, "Dwarf: Distributed wearable augmented reality framework," 2005.

[9] N. R. Center, "Mobile augmented reality applications project," 2007. `http://research.nokia.com/research/projects/mara/index.html`.

[10] ARToolworks, "Artoolkit," 2009. `http://www.artoolworks.com/`.

[11] A. H. Behzadan, B. W. Timm, and V. R. Kamat, "General-purpose modular hardware and software framework for mobile outdoor augmented reality applications in engineering," *Adv. Eng. Inform.*, vol. 22, no. 1, pp. 90–105, 2008.

[12] S. Goose and G. Schneider, "Augmented reality in the palm of your hand: A pda-based framework offering a location-based, 3d and speech-driven user interface." `http://www.ainformatik.fh-trier.de/~schneider/papers/graz.pdf`.

[13] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *In Proceedings of the Workshop on Mobile Computing Systems and Applications*, pp. 85–90, IEEE Computer Society, 1994.

[14] N. S. Ryan, J. Pascoe, and D. R. Morse, "Enhanced reality fieldwork: the context-aware archaeological assistant," in *Computer Applications in Archaeology 1997* (V. Gaffney, M. van Leusen, and S. Exxon, eds.), British Archaeological Reports, (Oxford), Tempus Reparatum, October 1998.

[15] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, (London, UK), pp. 304–307, Springer-Verlag, 1999.

[16] T. Buchholz, "Context-aware services for umts-networks." Summer School on Ubiquitous and Pervasive Computing, 2002. `http://www.vs.inf.ethz.ch/events/dag2002/program/ws/Buchholz.pdf`.

[17] O. Riva and C. di Flora, "Contory: A smart phone middleware supporting multiple context provisioning strategies," p. 68, 2006.

[18] P. Dockhorn Costa, "Towards a services platform for context-aware applications," Master's thesis, University of Twente, Aug. 2003.

[19] D. Salber, A. K. Dey, and G. D. Abowd, "The context toolkit: Aiding the development of context-enabled applications," in *Proceedings of CHI'99*, (Pittsburgh, PA, USA), ACM Press, 1999.

[20] T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, B. Serra, and M. Spasojevic, "People, places, things: web presence for the real world," *Mob. Netw. Appl.*, vol. 7, pp. 365–376, October 2002.

[21] P. Pawar, A. Halteren, van, and K. Sheikh, "Enabling context-aware computing for the nomadic mobile user: A service oriented and quality driven approach," in *WCNC 2007*, pp. 2531–2536, 2007.

[22] H. Truong and S. Dustdar, "A survey on context-aware web service systems," *International Journal of Web Information Systems*, vol. 5, no. 1, pp. 5–31, 2009.

[23] K. Kjaer, "A survey of context-aware middleware," in *SE'07: Proceedings of the 25th conference on IASTED International Multi-Conference*, (Anaheim, CA, USA), pp. 148–155, ACTA Press, 2007.

[24] N. Gray, "Comparison of web services, java-rmi, and corba service implementation," in *Fifth Australasian Workshop on Software and System Architectures*, 2004.

[25] J. Vallino, *Interactive Augmented Reality*. PhD thesis, University of Rochester, Rochester, New York, 1998.

[26] D. Wagner, *Handheld Augmented Reality*. PhD thesis, Graz University of Technology, Oct. 2007.

[27] D. Schmalstieg, "An introduction to augmented reality," 2001. http://www.iswc.ethz.ch/events/tutorials/slides_schmalstieg.pdf.

[28] ARToolworks, "How does artoolkit work," 2009. http://www.hitl.washington.edu/artoolkit/documentation/userarwork.htm.

[29] GE, "Plug into the smart grid: Augmented reality." http://ge.ecomagination.com/smartgrid/#/augmented_reality.

[30] Topps, "Topps 3d live," Mar. 2009. http://www.toppstown.com/UserSite/TotalImmersion/Info.aspx.

[31] Sony, "The eye of judgement," 2007. http://www.us.playstation.com/EyeofJudgment/.

[32] Wearable Computer Lab, University of South Australia, "Tinmith augmented reality project," Dec. 2007. http://www.tinmith.net/.

[33] Lego, "Lego digital box." http://www.psfk.com/2009/01/legos-digital-box-adds-augmented-reality-toy-previews.html.

[34] Toyota, "Toyota iq: Augmented reality." http://www.youtube.com/watch?v=1_7NW_u3VFo.

[35] Ubergizmo, "Lego digital box kiosk uses augmented reality in retail stores," Jan. 2009. http://www.ubergizmo.com/15/archives/2009/01/lego_digital_box_kiosk_uses_augmented_reality_in_retail_stores.html.

[36] Mobilizy, "Wikitude," 2009. "http://www.mobilizy.com/wikitude.php".

[37] R. Fontana, E. Richley, and J. Barney, "Commercialization of an ultra wideband precision asset location system," *IEEE Conference on Ultra Wideband Systems and Technologies*, pp. 369–373, 2003.

[38] Y. Gwon, R. Jain, and T. Kawahara, "Robust indoor location estimation of stationary and mobile users," *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 2, pp. 1032– 1043, 2004.

[39] Y. Zhao, "Standardization of mobile phone positioning for 3g systems," *IEEE Communications Magazine*, July 2002.

[40] S. Feng and C. Law, "Assisted gps and its impact on navigation in intelligent transportation systems," *Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on*, pp. 926– 931, 2002.

[41] G. Djuknic and R. Richton, "Geolocation and assisted gps," *Computer*, 2001.

[42] J. Hurn, *Differential GPS Explained*. Trimble Navigation, 1993.

[43] S. Wang, J. Min, and B. Yi, "Location based services for mobiles :technologies and standards," 2008. IEEE International Conference on Communication (ICC).

[44] D. Jonsson and J. Olavesen, "Estimated accuracy of location in mobile networks using e-otd," Master's thesis, Agder University College, 2002.

[45] B. Bing, *Wireless Local Area Networks: The New Wireless Revolution*, vol. 41. Wiley-Inter-Science, 2003.

[46] B. Li, A. Dempster, C. Rizos, and J. Barnes, "Hybrid method for localization using wlan," *Networks*, 2005.

[47] B. Li, Y. Wang, H. K. Lee, A. Dempster, and C. Rizos, "A new method for yielding a database of location fingerprints in wlan," *Communications, IEEE Proceedings*, vol. 152, pp. 580–586, 2005.

[48] D. Pandya, R. Jain, and E. Lupu, "Indoor location estimation using multiple wireless technologies," *Personal, Indoor and Mobile Radio Communications, 2003. PIMRC 2003. 14th IEEE Proceedings on*, vol. 3, pp. 2208–2212, 2003.

[49] P. Bahl and V. Padmanabhan, "Radar: An in-building rf-based user location and tracking system," in *IEEE Infocom 2000*, (Tel Aviv, Israel), pp. 775–784, 2000.

[50] P. Bahl, A. Balachandran, and V. Padmanabhan, "Enhancements to the radar user location and tracking system," 2000.

[51] A. Smailagic, D. Siewiorek, J. Anhalt, D. Kogan, and Y. Wang, "Location sensing and privacy in a context aware computing environment," *IEEE Wireless Communications*, vol. 9, pp. 10–17, 2001.

[52] Z. Li, W. Trappe, Y. Zhang, and B. Nath, "Robust statistical methods for securing wireless localization in sensor networks," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pp. 91–98, 2005.

[53] A. M. Ladd, "Robotics-based location sensing using wireless ethernet," in *MOBICOM 2002*, (Atlanta, Georgia, USA), ACM, 2002.

[54] H. Tirri, P. Misikangas, and J. Sieva, "A probabilistic approach to wlan user location estimation," *International Journal of Wireless Information Networks*, vol. 9, 2002.

[55] M. Youssef and A. Agrawala, "The horus wlan location determination system," in *International Conference On Mobile Systems, Applications And Services*, pp. 205 – 218, ACM, 2005.

[56] P. Castro, P. Chiu, T. Kremenek, and R. Muntz, *A Probabilistic Room Location Service for Wireless Networked Environments*, pp. 18–34. Springer Berlin / Heidelberg, volume 2201/2001 ed., 2001.

[57] A. Haeberlen, E. Flannery, and A. Ladd, "Practical robust localization over large-scale 802.11 wireless networks," *Wireless Networks*, 2004.

[58] D. Madigan, E. Elnahrawy, R. P. Martin, W.-h. Ju, P. Krishnan, and A. S. Krishnakumar, "Bayesian indoor positioning systems," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, pp. 1217– 1227 vol. 2, 2005.

[59] E. Inc., "Ekahau," 2009. `http://www.ekahau.com/`.

[60] K. Inc., "Placeengine," 2009. `http://www.placeengine.com/en`.

[61] S. Wireless, "Skyhook," 2009. `http://www.skyhookwireless.com/`.

[62] Z. E. Solutions, "Rtls," 2009. `http://zes.zebra.com/products/rtls/index.jsp`.

[63] M. Hossain and W.-S. Soh, "A comprehensive study of bluetooth signal parameters for localization," in *Proc. IEEE PIMRC*, (Athens, Greece), Sep. 2007.

[64] G. Takacs and N. Gelfand, "Outdoors augmented reality on mobile phone using loxel-based visual feature organization," *Pattern Recognition*, 2008.

[65] K. Virrantaus, J. Markkula, A. Garmash, V. Terziyan, J. Veijalainen, A. Katanosov, and H. Tirri, "Developing gis-supported location-based services," vol. 2, pp. 66–75 vol.2, 2001.

[66] eMarketer, "Mobile location-based services on the move," Oct. 2008. `http://www.emarketer.com/Article.aspx?R=1006609`.

[67] J. Pereira, "E-112 and location-based vas." `ftp://ftp.cordis.europa.eu/pub/ist/docs/directorate_d/cnt/wshop/ws040308/e112_and_privacy_concerns_en.pdf`.

[68] G. Association, "Location based services," 2003. context-awareness.

[69] S. Steiniger, M. Neun, and A. Edwardes, "Foundations of location based services," 2006. context-awareness.

[70] R. Miller, "Response time in man-computer conversational transactions," *Proc. AFIPS Fall Joint Computer Conference*, vol. 33, pp. 267–277, 1968.

[71] Amigo, "Amigo project," 2009. `http://www.hitech-projects.com/euprojects/amigo/index.htm`.

[72] L. O. Bonino da Silva Santos, R. P. van Wijnen, and P. Vink, "A service-oriented middleware for context-aware applications," in *Proceedings of the 5th International Workshop on Middleware for Pervasive and Ad-hoc Computing (MPAC 2007), Newport Beach, CA, USA*, ACM International Conference Proceedings Series, (New York), pp. 37–42, ACM Press, November 2007.

[73] OSGi Alliance, "Osgi," 2009.

[74] Oscar, "Oscar bundle repository," 2009. `http://oscar-osgi.sourceforge.net/`.

[75] Google, "Android os," 2009. http://www.android.com/.

[76] S. Pandey and A. Prathima, "a survey on localization techniques for wireless networks," *Journal of the Chinese Institute of Engineers*, vol. 29, no. 7, pp. 1125–1148, 2006.

[77] M. Youssef and A. Agrawala, "Continuous space estimation for wlan location determination systems," pp. 161–166, 2004.

[78] M. Youssef and A. Agrawala, "Small-scale compensation for wlan location determination systems," vol. 3, pp. 1974–1978 vol.3, 2003.

[79] M. Youssef and A. Agrawala, "Handling samples correlation in the horus system," vol. 2, pp. 1023–1031 vol.2, 2004.

[80] M. Youssef, M. Abdallah, and A. Agrawala, "Multivariate analysis for probabilistic wlan location determination systems," pp. 353–362, 2005.

[81] W3, "Resource description framework," 2009. http://www.w3.org/RDF/.

[82] W3C, "Owl web ontology language." http://www.w3.org/TR/owl-features/.

[83] XMPP Standards Foundation, "Xmpp," 2009. http://xmpp.org/.

[84] Google, "Android api - activiy," 2009. http://developer.android.com/reference/android/app/Activity.html.

[85] SQL.org, "Sql." http://www.sql.org/.

[86] Ignite Realtime, "Smack api," 2009. http://www.igniterealtime.org/projects/smack/.

[87] Ignite Realtime, "Openfire," 2009. http://www.igniterealtime.org/projects/openfire/.

[88] W3, "Sparql query language for rdf," 2009. http://www.w3.org/TR/rdf-sparql-query/.

[89] IST Amigo Project, "Deliverable awareness and notification service software developers guide," January 2008. http://www.hitech-projects.com/euprojects/amigo/deliverables/amigo_4_d4.7_final.pdf.

[90] IST Amigo Project, "Amigo code repository," 2008. http://gforge.inria.fr/frs/?group_id=160.

[91] L. M. Daniele, P. D. Costa, and L. F. Pires, "Towards a rule-based approach for context-aware applications," in *Proceedings of the 13th Open European Summer School and IFIP TC6.6 Workshop, EUNICE 2007, Enschede, The Netherlands* (A. Pras and M. J. van Sinderen, eds.), vol. 4606 of *Lecture Notes in Computer Science*, (Berlin Heidelberg), pp. 33–43, Springer Verlag, July 2007.

[92] JESS, "Jess, the rule engine for the java platform," 2009. http://www.jessrules.com/.

[93] Novoda, "Motion sensors and orientation in android," 2009. http://www.novoda.com/blog/?p=77.

[94] Sun, "Creating a gui with jfc and swing."

[95] R. Khalil, "Digital compass, is it error free?," *CERM*, 2004.