ulm university universität
uulm

Engineering an Advanced Location-Based Augmented Reality Engine for Smart Mobile Devices

Philip Geiger, Rüdiger Pryss, Marc Schickler, Manfred Reichert

Ulmer Informatik-Berichte

Nr. 2013-09
Oktober 2013

Engineering an Advanced Location-Based Augmented Reality Engine for Smart Mobile Devices

Philip Geiger, Rüdiger Pryss, Marc Schickler, and Manfred Reichert
Institute of Databases and Information Systems
University of Ulm, Germany
Email: {philip.geiger, ruediger.pryss, marc.schickler, manfred.reichert}@uni-ulm.de

I. Introduction

Daily business routines more and more require to access information systems in a mobile manner, while preserving a desktop-like feeling at the same time. However, the design and implementation of sophisticated mobile business applications constitutes a challenging task. On the one hand, a developer must cope with limited physical resources of mobile devices (e.g., limited battery capacity, or limited screen size) as well as with non-predictable user behaviour (e.g., mindless instant shutdowns). On the other, mobile devices provide new technical capabilities like motion sensors, a GPS-sensor, or a potent camera system [1], and hence new types of applications types can be designed and implemented. Integrating sensors and using data recorded by them, however, is a non-trivial task when considering requirements like robustness and scalability. In this work, we present the engineering of such an application, which provides location-based augmented reality, and discuss the various challenges to be tackled in this context.

A. Problem Statement

The overall purpose of this work is to outline the engineering process of a sophisticated mobile service running on a smartphone. More precisely, we show how to develop the core of a *location-based augmented reality engine* for the *iPhone 4S* based on the operating system *iOS 5.1 (or higher)*. We denote this engine as *AREA*[1]. In particular, we develop concepts for coping with limited resources on a mobile device, while providing a smooth user augmented reality experience at the same time. We further present and develop a suitable application architecture in this context, which easily allows integrating augmented reality with a wide range of applications. However, this architecture must not neglect the characteristics of the underlying type of mobile operating system. While in many cases, the differences between mobile operating systems are not crucial when engineering a mobile application, for more sophisticated mobile applications this does no longer hold. Consequently, mobile cross development frameworks [1] are usually not suitable in this context; e.g., they can not provide functions for accessing sensors.
It is noteworthy that there already exist several augmented reality frameworks and applications for smartphone operating systems like Android or iOS. This includes proprietary and commercial engines as well as open source frameworks and applications (e.g., [2], [3]. To the best of our knowledge the proposals do neither allow for deeper insights into the engineering and functionality of such an engine nor for customizing it to a specific purpose.

As a particular challenge in our context, the augmented reality engine shall be able to display *points of interest (POIs)* from the surrounding on the screen of the smartphone. Thereby, POIs shall be drawn based on the *angle of view*, the current *attitude* of the smartphone, and its *position*. This means that the real image captured by the camera of the smartphone will be augmented by *virtual objects (i.e., the POIs)* relative to the current position and attitude.

The development of such an augmented reality engine constitutes a non-trivial task. In particular, the following challenges need to be tackled. The overall goal is to draw POIs on the camera view of the smartphone.

- In order to enrich the image captured by the smartphone's camera with virtual information about POIs in the surrounding, basic concepts enabling geographic calculations need to be developed,
- an efficient and reliable method must be identified to calculate the distance between two positions based on data of the GPS-sensor,
- numerous sensors must be queried correctly to determine the attitude and position of the smartphone,
- the angle of view of the smartphones camera lens must be calculated to display the virtual objects on the corresponding position of the camera view.

[1]AREA stands for Augmented Reality Engine Application. A video demonstrating AREA can be viewed at: http://vimeo.com/channels/434999/63655894

B. Contribution

Fig. 1 presents a screenshot of *AREA*. In particular, when designing and implementing this engine presented challenges have been tackled. Fig. 1 shows a radar, compass, slider to adjust the radius, and virtual object in front of the corresponding POI.

Fig. 1: AREA Screenshot

To deal with the above discussed, issues related to three different areas must be addressed. First, general issues of realizing augmented reality on mobile devices must be considered. Second, issues related to the underlying mobile operation system are crucial. Finally, issues concerning the engineering of such a sophisticated mobile service must be properly addressed. Fig. 2 summarizes the various issues within the overall setting of AREA.

As illustrated by Fig. 2, the engineering process not only concerns the design and implementation of AREA itself, but also the realization integration of applications on top of AREA. In this paper, we present one such AREA application, namely *LiveGuide Ditzingen*. It uses the full functionality provided by AREA and has proven its practical use in the field. Along this application, we further illustrate how to integrate AREA, to communicate with AREA, and to make the application ready to use all features of AREA correctly.

The remainder of this paper is structured as follows: Section 2 discusses fundamental requirements of the augmented reality engine and main topics of its engineering process. In Section 3, the formal foundation and mathematical basis of the developed augmented reality engine as well as the architectural design of AREA are presented. In Section 4, major issues related to the engineering and implementation of AREA are introduced. In turn, Section 5 presents results of a user survey, in which we evaluated the core of AREA. Section 6 describes how to realize and integrate *LiveGuide Ditzingen* using AREA and illustrates the communication between AREA, this application, and a remote server. Section 7 summarizes the process of engineering an application based on AREA and the tasks to be performed in this context. Section 8 then presents a validation of our sample application (i.e., *LiveGuide Ditzingen*) and discusses problems identified when implementing and using this application. Section 9 discusses related work and Section 10 summarizes this paper and concludes with an outlook.

II. Requirements

This section introduces the requirements for providing a fully functional and usable augmented reality engine like AREA. In particular, these requirements also yield issues concerning the engineering process for such advanced mobile service.

A. AREA

The augmented reality engine *AREA* shall basically show *points of interest* (POIs) inside the camera view relative to the current position of the user and the POIs. As a prerequisite for this, the GPS as well as the POIs' coordinates must be available. On the screen, the POIs shall be only displayed if they are inside a visible view of the user, particularly inside the field of view of the device's built-in camera. Hence, it becomes necessary to determine (1) the altitudes of the device and POIs, (2) the bearing between them relative to the north pole, and (3) the attitude of the device based on three axes of the accelerometer. Reading these attributes must be accomplished in realtime and with high accuracy during all possible movements of the user or his device. In this case, neither efficiency nor stability can be neglected. Furthermore, it shall be possible to set up a radius, such that only POIs being inside this radius are displayed on the screen. Thus, points with a longer distance to the user as defined by the radius are ignored. To analyze all POIs independent of the field of view inside the camera view, we provide a radar feature. Independent of the radius and the field of view, a map view shall display the user's current position and the surrounding POIs.

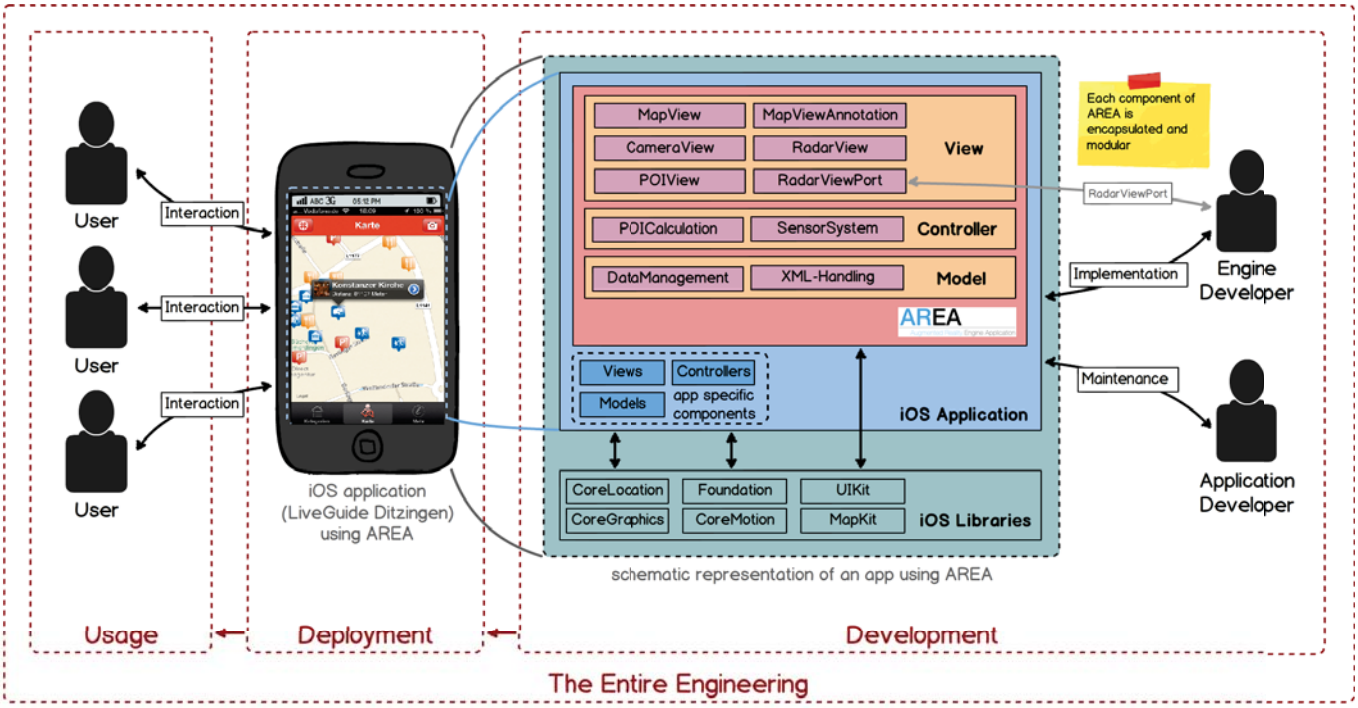


Fig. 2: Overall AREA Setting

Thereby, all visible POIs on the camera view and the map view are linked with touch events, thus, further information about these POIs can be obtained. Both, using the smartphone in portrait mode and in landscape mode shall to be possible. Fig. 3 illustrates, which POIs shall be displayed within the camera view.

For the augmented reality engine, it shall be possible to integrate it into other applications without big efforts. To enable this, the engine shall offer public interfaces and ensure a high modularity. Accordingly, a consistent and simple specification of POIs must be provided, i.e., it should be possible to add and remove them statically or dynamically if required. In particular, the POIs shall be implemented in a way such that extensions to their internal structure have no effects on the engine's functionality.

Tab. I summarizes the requirements of AREA. In particular, these requirements need to be addressed to ensure that the augmented reality engine works well. Note that comparable requirements exist for many other sophisticated mobile applications, i.e., our results should be of high interest for any engineer of sophisticated mobile services. Modern mobile devices nowadays have high amount of resources like fast processors and big internal memory, but it is crucial to implement such an application or engine in a resource-saving manner to save battery time. It is also very important to provide a user-friendly and suitable user interface for the relatively small screen of mobile devices.

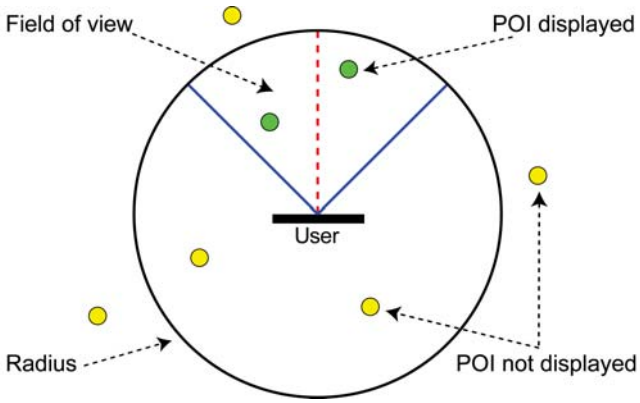


Fig. 3: Schematic illustration of visible POIs

TABLE I: Requirements of AREA

Requirement	Type
Provide POIs on camera view	functional
Provide POIs on map view	functional
Enable POIs on camera view only if they are inside the visible field of the user's view	functional
Provide that POIs on camera view and on map view react to touch events	functional
Read sensors to determine position of the iPhone (i.e. accelerometer, GPS sensor, and magnetic field sensor)	functional
Update data and POI in realtime while all possible movements	functional
Provide adjustable radius for the distance of showable POIs	functional
Provide a radar in camera view showing POIs in the environment and inside the radius	functional
Provide additional information according to touch events	functional
Support portrait mode and landscape mode	functional
Provide efficiency of calculations	non-functional
Provide accuracy of calculations	non-functional
Provide efficiency of screen drawing	non-functional
Provide stability	non-functional
Support maintainability	non-functional
Provide consistent specification of POIs	implementation
Provide that POIs can be easily extended	implementation
Provide easy integration into other applications (modularity)	implementation

B. Engineering Process

Section 2.1 has presented the requirements of the AREA engine. In turn, these also refer to the main issues of the corresponding engineering process, which we split into four categories. First, engineering aspects related to the *architecture* must be covered, which mainly concern modularity and extendability of AREA. Second, aspects related to the *performance* are crucial. To meet the requirement of realtime updates of POIs and related data, in addition, a quick, resource-efficient, and reliable way of calculating and redrawing the screen must be realized. Third, aspects related to *usability* are crucial. Hence, a suitable and intuitive user interface must be designed. Fourth, aspects related to the *communication* between AREA, AREA applications, and remote servers storing all POIs must be properly addressed. Tab. II summarizes these four categories. Thereby, each row is divided into *question*, *answer*, and *validation*. In turn, a question refers to a specific aspect - AREA's requirements - and must be considered when engineering an augmented reality engine and similar applications. Corresponding answers and validations of our approach are subject of the following sections.

III. Foundation and Architecture of AREA

This section presents two fundamental issues of the engineering process of AREA. First, we present the formal foundation and mathematical basis. Second, we discuss the architectural design of AREA.

TABLE II: Engineering Process Aspects

Topic	Questions	Answers	Validation
Architecture	How to design the architecture of AREA making it extendable and modular?	-	-
	How to easily implement applications based on AREA?	-	-
Performance	How to implement an efficient way to update POIs in realtime?	-	-
	How to provide a quick redraw process?	-	-
Usability	How to make the user interface intuitive and usable?	-	-
Communication	How perform communication between AREA and a remote server?	-	-

"-" Answers and Validation will be given in Section 7

A. Formal Foundation of AREA

This section presents the formal and mathematical basis necessary to develop the augmented reality engine AREA. First, we must be able to calculate the distance between user and POI location. Second, to determine whether a POI is inside the current visible field of view, the point of compass of a POI according to the user's location and the altitude difference between the user and POI must be calculated. Finally, the field of view of the smartphone's internal camera must be determined.

1) Calculating the Distance: Great-circle distance is a method known from spherical geometry [4] to calculate the distance between two points on a curved surface like the earth. Thereby, the distance is not measured based on a line through the sphere, instead, the distance is related to the surface of the sphere. Formula (1) offers a first way to calculate this distance.

$$\theta = \arccos(\sin\phi_A \sin\phi_B + \cos\phi_A \cos\phi_B \cos(\Delta\lambda)) \tag{1}$$

$$D = \theta * 6371km \tag{2}$$

Thereby, A corresponds to the position of the user and B to the one of the POI. Further, ϕ denotes the latitude and λ the longitude of one of these positions. Finally, $\Delta\lambda = \lambda_B - \lambda_A$ corresponds to the difference between the two longitudes. Note that parameters ϕ, λ, and $\Delta\lambda$ must be available in radians to calculate the angle θ between these two points. To determine distance D, θ must be multiplied by the radius of the sphere, i.e, the radius of the earth (cf. Formula (2)).

However, this method has a significant drawback. If both points are located close to each other, there might be a rounding error, while executing Formula (1) on a computer. This is due to the fact that the value inside the parentheses will then be close to 1, e.g., 0.99999. Thus, a numerical instability occurs calculating $arccos$. To improve this method, the *Haversine*-formula (cf. Formula (3)) had been developed; it allows for a better numerical stability [4] and meets the requirements of our engine.

$$\theta = 2\arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos\phi_A \cos\phi_B \sin^2\left(\frac{\Delta\lambda}{2}\right)}\right) \tag{3}$$

Note that the numerical stability problem still exists if the points are located almost oppositely to each other on the sphere [5]. However, such big distances will not occur in our engine, Formula (3) is suitable and, therefore, used to calculate the distance between two points. In order to calculate the distance, θ from Formula (3) is inserted in Formula (2). The difference between the great-circle distance, the haversine formula, and the numerical instability is shown in Listing 1. In lines 8 to 16, two locations are initialized and necessary variables are defined. In lines 18 to 20, the great-circle distance and in lines 23 to 24 the distance based on haversine are calculated and, afterwards the results are printed. The value of angle θ corresponding to locations initialized before is close to zero ($0.99999999999988883...$). Thus, the CPU rounds it to 1.000000, and the great-circle distance calculates an inaccurate distance (0.000134 km). The correct distance is calculated by haversine and results in 0.000079 km.

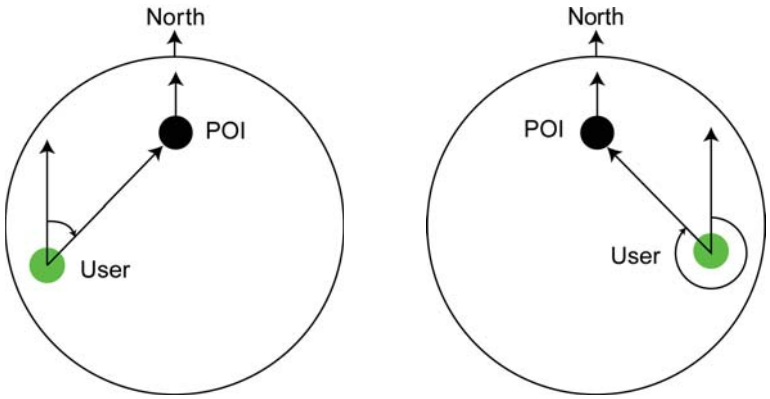
Listing 1: Comparison between great-circle distance and haversine formula

```c
#include <stdio.h>
#include <math.h>
#define toRad(x) ((x)*M_PI/180.0)

int main()
{
   // location one
   double lat1 = 48.4042981;
   double lon1 = 9.979349;
   // location two
   double lat2 = 48.40429881;
   double lon2 = 9.979349;

   double dLat = toRad(lat2 - lat1);
   double dLon = toRad(lon2 - lon1);
   double radius = 6371;

   // Great-Circle Distance
   double angle = sin(toRad(lat1)) * sin(toRad(lat2)) + cos(toRad(lat1)) * cos(toRad(lat2)) * cos(dLon);
   double great_circle_distance = acos(angle) * radius;

   // Haversine Formula
   double res1 = pow(sin(dLat/2), 2) + cos(toRad(lat1)) * cos(toRad(lat2)) * pow(sin(dLon/2), 2);
   double haversine = 2*asin(sqrt(res1)) * 6371;

   printf("Angle: \t\t %f \nGreat Circle:\t %f km\nHaversine:\t %f km\n\n", angle, great_circle_distance, haversine);

   return 0;
}

Angle:         1.000000
Great Circle:  0.000134 km
Haversine:     0.000079 km
```

2) Calculating the Bearing: Only POIs being inside the visible field of view shall be displayed on the camera view. Hence, the bearing between user and POI's positions relative to the north pole must be calculated (cf. Fig. 4). While the POI is located at the same position in both figures (cf. Fig 4), the user is located at different positions in (a) and (b). Hence, a different bearing between the POI and the user results based on the following calculation. Formula (4) is used to calculate this bearing [4], [6].

$$\theta = \arctan 2(sin(\Delta\lambda)\cos\phi_B, \cos\phi_A \sin\phi_B - \sin\phi_A \cos\phi_B \cos(\Delta\lambda)) \tag{4}$$

The identifiers A, B, ϕ, and λ have same meanings as in Formula (1). Since the result, in which θ is converted to degrees, maps the interval between $-180°$ and $+180°$, it has to be transformed with $(\theta + 360°)\ mod\ 360°$ to finally map the interval to $0°...360°$. Using this result, it becomes possible, in combination with the smartphone's compass, to determine whether a POI is inside the *horizontal field of view* and where it must be drawn on the screen.



(a) A POI located to the user's east (b) A POI located to the user's west

Fig. 4: Schematic representation of the calculated bearing

3) Calculating the Elevation Angle: The visible field of view of the smartphone's is not only limited in its width, but also in its height. Therefore, the altitude differences between the user and POIs must be calculated to determine whether or not a

particular POI is inside the *vertical field of view*. For this reason, we are using an approach that yields an angle between $-90°$ and $+90°$ as result. Thus, it can be determined what area shall be visible on the display related to the pitch of the smartphone. Fig. 5 illustrates Formula (5), which calculate this angle.

$$\theta = \sigma * \frac{180}{\pi} \arctan\left(\frac{\Delta h}{d}\right), \sigma \in \{-1, 1\} \tag{5}$$

Thereby, Δh corresponds to the altitude difference between the locations of user and POI, and d to the distance between them. Attention should be paid to the fact that σ is dependent on this altitude difference. We obtain $\sigma = 1$ if the POI is located higher than the user, otherwise we obtain $\sigma = -1$. Using term $\frac{\Delta h}{d}$, the angle between the hypotenuse and adjacent, respectively the elevation angle, is calculated (cf. Fig. 5).

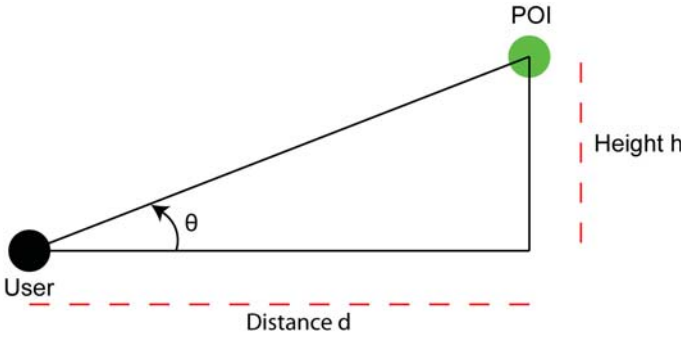


Fig. 5: Illustration of Formula (5)

4) Calculating the Field of View: Based on the previous considerations, it is possible to determine whether a point is inside the vertical and horizontal field of view as well as to calculate the distance. Additionally, it must be known how the size of the field of view of the smartphone - in this case the iPhone - and its camera can be specified (cf. Formula (6). [7])

$$\alpha = 2 \arctan\left(\frac{B}{2f}\right) \tag{6}$$

Thereby, B corresponds to the image size, more precisely to the size of the *image sensor*, and f to the focal length of the camera lens. The image sensor has a size of 4.592 x 3.45 mm^2 and the focal length corresponds to 4.28 mm [8]–[11].

$$2 \arctan\left(\frac{4.592}{2 * 4.28}\right) * \frac{180}{\pi} \approx 56.4225 \tag{7}$$

$$2 \arctan\left(\frac{3.45}{2 * 4.28}\right) * \frac{180}{\pi} \approx 43.9026 \tag{8}$$

Consequently, the iPhone has a horizontal field of view of $56°$ (cf. Formula (7)) and a vertical field of view of $44°$ (cf. Formula (8)) when used in landscape mode.

Now the size of the field of view of the iPhone camera is known, by additionally using Formulas (3)-(5), it becomes possible to determine whether a POI is inside the vertical and horizontal field of view, and in what distance a POI is located to the user. These calculations play a crucial role during the implementation and the functionality of AREA.

B. Architectural Design of AREA

In the following we introduce the architectural design of AREA, which is characteristic for any sophisticated mobile application. As mentioned in Section 2, it must be possible to easily integrate AREA into other applications. Furthermore, AREA must be highly modular making it possible for developers to exchange and extend modules as well as to modify specific behavior of the engine. We present these requirements in this section.

1) Overall Architectural Design: The architecture of AREA comprises four main modules. First, a *controller* uses the mathematical formulas introduced in this section. On one hand, the controller is responsible for reading the sensors of the smartphone. This includes determining the user's position based on the GPS-sensor and the horizontal point of view based on the compass sensor. To calculate the vertical point of view, an acceleration sensor is used. On the other hand, the controller determines whether a POI is inside the field of view and on which position on the screen it shall be drawn. Second, a model realizes the data management of POIs providing a unified interface, which consists of an XML-schema and appropriate XML-parser. Thus, it becomes possible to extend and exchange POIs independent of the platform used. Third, a view contains various elements displayed on the screen. These include the POIs, a radar, and further elements for user interactions. Fourth, another important module, which is not directly related to AREA, provides a collection of libraries and frameworks offered by the mobile operation system and required to use functions like *reading sensors* and *drawing respective results on the screen*.

The four modules are arranged in a multi-tier architecture (cf. Fig. 6) and comply to the *MVC* pattern [12]. Lower tiers offer their services and functions by interfaces to upper tiers. Based on this architectural design, modularity can be ensured, which means that both the data management and various elements of the view can be customized and extended on demand. Furthermore, the compact design of AREA enables us to build new applications based on it, or to easily integrate AREA into existing applications.

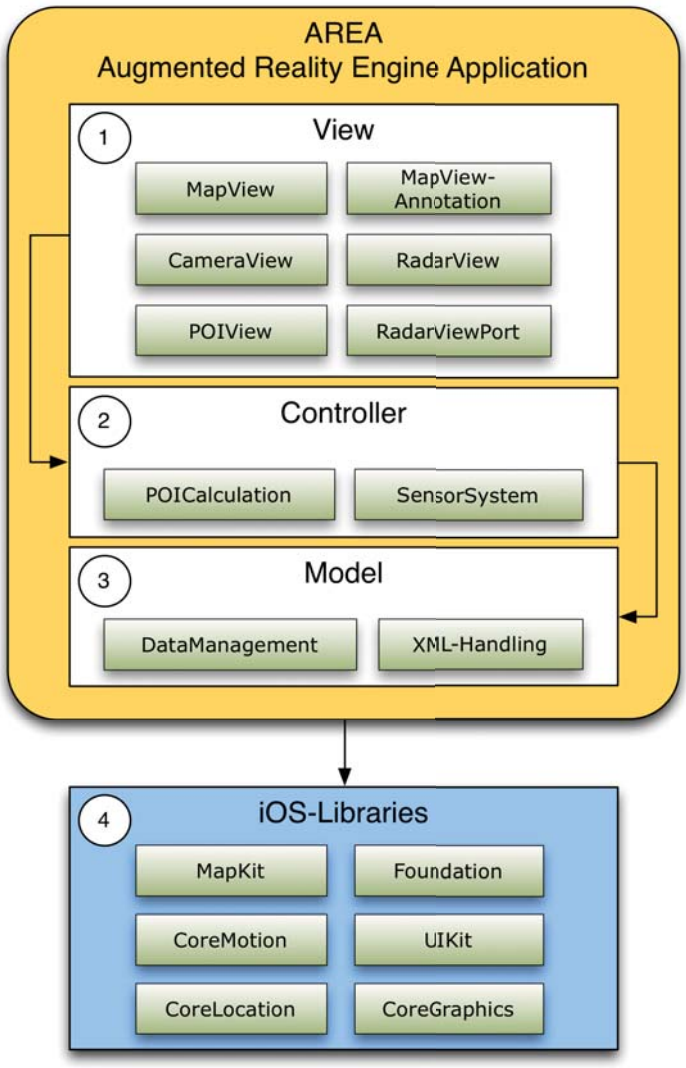


Fig. 6: Multi-tier architecture of AREA

2) Class Structure: In the class diagram of AREA (cf. Fig. 7), the different modules are depicted. The class structure has been designed to easily integrate applications into AREA, or to use AREA in already existing applications. For this purpose, only a reference to class `ViewController` is needed. Initializing the components and modules will be accomplished by AREA autonomously.

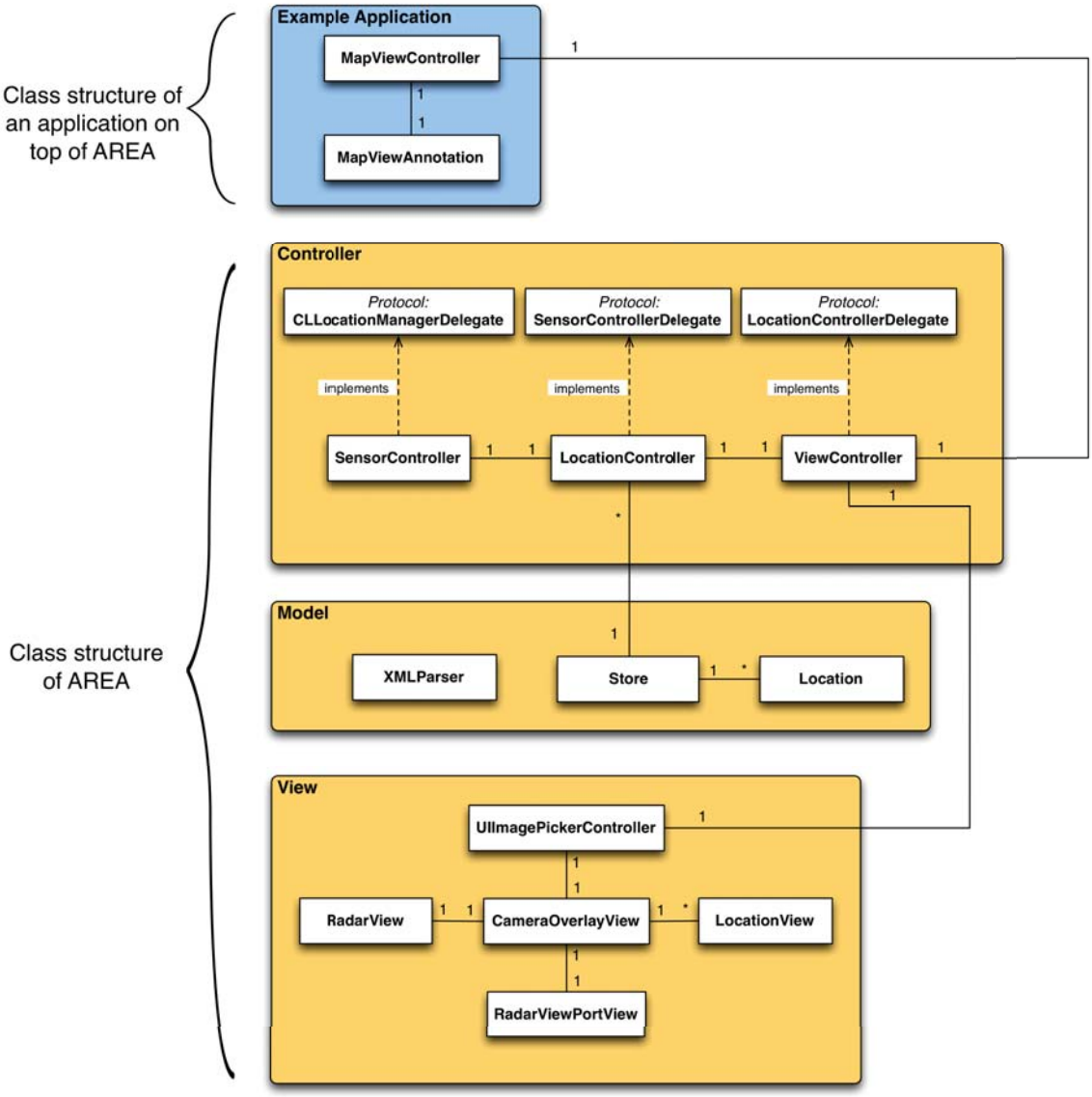


Fig. 7: Class diagram of AREA and an application on top

As already described above, the controller is responsible for reading sensors calculating POIs. The class `SensorController` implements the protocol `CLLocationManagerDelegate` in order to query data of the GPS-sensor and compass sensor. This class also implements an application loop in which the data of the acceleration sensor is polled, and both the polled data and the data of the GPS as well as acceleration sensor, are passed to the `LocationController`. For this reason, the class must implement protocol `SensorControllerDelegate`. Then, it is determined whether a POI is inside the camera's field of view and - if yes - where on the screen it shall be drawn. Class `ViewController` is informed about these calculations and the POIs to be drawn are informed by protocol `LocationControllerDelegate`. Furthermore, the `ViewController` is responsible for preparing and loading the complete view. For this purpose, the camera must be accessed by using the `UIImagePickerController`. Based on its property `cameraOverlayView`, it becomes possible to place

custom sub-views like RadarView, RadarViewPort, and locationView on the camera view. The latter constitutes an important view containing the POIs. Besides a Store and Location, the model component encompasses an XMLParser, which is is responsible for loading POIs and manipulating them independently of a platform.

AREA has the requirement to provide a map view. This is one example for the easy integration of functionality into AREA. The MapViewController is responsible for the map view and is delivered by the mobile operating system. To change to the camera view starting by the map view, just the ViewController has to be referenced. Everything else is accomplished by AREA autonomously.

IV. Engineering and Implementing AREA

The prototype of AREA was implemented using the programming language *objective-C* (v. iOS 5.1) on *Apple iPhone 4S*. For the development, the *Xcode* [13] environment in Version 4.4.1 has been used.

In the following section, the most important engineering aspects of AREA are discussed. This includes manipulating the camera view and drawing POIs on the screen. Inside the controller, the core of AREA, proper querying of sensors, interpreting data, calculating the field of view, and determining the location of POIs are elucidated.

A. Manipulating the Camera View

On the camera view, both a radar and the POIs shall be displayed as customized UIViews. For this purpose, a UIImage-PickerController must be initialized and customized. The latter is responsible for controlling the camera and is provided by the *iOS* operating system. Listing 2 shows a code snippet of the AREAViewController's init-method. As mentioned in Section 3, this class is responsible for controlling the view components. The camera view of the iPhone usually has a UINavigationBar. Since the camera view should use the entire screen, the UINavigationBar is hidden. This results in a blank black bar at the upper end of the device to be removed by scaling up the camera view. This happens in Line 7 in which constants CAMERA_TRANSFORM_X and CAMERA_TRANSFORM_Y are set to 1.24299. In order to be able to draw on the camera view, a customized overlay must be given to the UIImagePickerController (Line 10). On this overlay, the radar, a slider to adjust the radius, and the POIs can be drawn.

Listing 2: Initializing the iPhone's camera and creating a customized overlay

```
1     self.picker = [[UIImagePickerController alloc] init];
2     self.picker.sourceType = UIImagePickerControllerSourceTypeCamera;
3     self.picker.showsCameraControls = NO;
4     self.picker.navigationBarHidden = YES;
5     self.picker.wantsFullScreenLayout = YES;
6     // let the camera take the entire screen
7     self.picker.cameraViewTransform = CGAffineTransformScale(self.picker.cameraViewTransform, CAMERA_TRANSFORM_X,
          CAMERA_TRANSFORM_Y);
8
9     // initialize the overlay for the UIImagePickerController
10    self.overlayView = [[UIView alloc] initWithFrame:CGRectMake(0, 0, 320, 480)];
11    self.overlayView.opaque = NO;
12    self.overlayView.backgroundColor = [UIColor clearColor];
13    ...
14    self.locationView = [[UIView alloc] initWithFrame:CGRectMake((IPHONE_WIDTH-VIEW_SIZE)/2, (IPHONE_HEIGHT-VIEW_SIZE)/2,
          VIEW_SIZE, VIEW_SIZE)];
15   ...
16    [self.overlayView addSubview:self.locationView];
17   self.picker.cameraOverlayView = self.overlayView;
```

The POIs being inside the camera's field of view are displayed on this overlay. Instead of drawing them directly on this overlay, a second view called locationView with a size of $580x580$ pixels is placed centrally on the overlay. Choosing this particular approach has specific reasons. First, AREA even shall display POIs correctly when the device is hold obliquely. This requires that the POIs, depending on the device's attitude, must be rotated by a certain angle and moved relatively to the rotation. Instead of rotating and shifting every single POI separately to the rotation, therefore, it is possible to rotate only the locationView (containing the POIs) to the desired angle. Thus, the POIs rotate automatically by rotating the locationView. In particular, resources needed for complex calculations can be saved.

The size of $580x580$ pixels is needed to draw POIs visible in portrait mode, landscape mode, as well as in a obliquely position between those modes. Fig. 8 presents the locationView as a white square illustrating that the locationView is bigger than the iPhone's display with a size of $320x480$ pixels. Therefore, the field of view must be increased by a certain factor such that all POIs, which are either visible in portrait mode, landscape mode, or any rotation in between, are drawn on the locationView. Figure 8 illustrates how a POI is drawn on the locationView, but not yet visible in landscape mode. Not before the device is rotated to the position shown in the middle figure, the POI will be visible for the user. When the device is rotated from the position in the middle figure to portait mode, the POI on the left moves out of the field of view, but remains on the locationView.

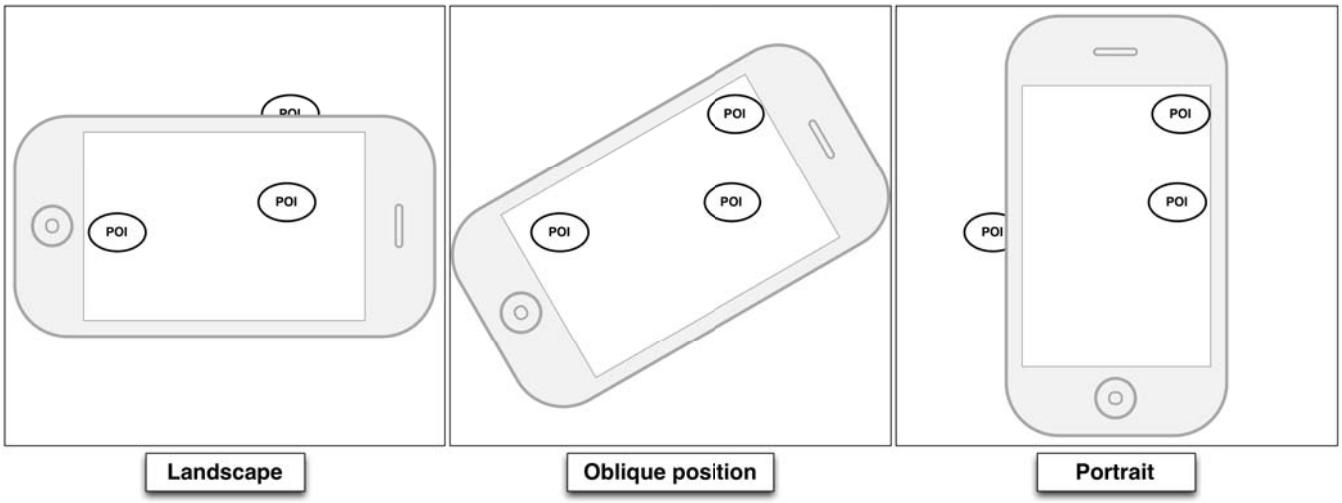


Fig. 8: Representation of the `locationView` (white square)

Second, a recalculation of the camera's field of view does not have to be considered when the device is in an oblique position. The vertical and horizontal field of view is scaled proportionally to the diagonal of the screen, such that a new maximum field of view results in the size of $580x580$ pixels. Since the `locationView` is placed centrally on the screen, the camera's actual field of view is not distorted and can be customized by rotating it contrary to the device's rotation.

The last reason for using the `locationView` concerned with performance and is shown in Listing 3. When the display has to be redrawn, the POIs already drawn on the `locationView` can be easily queried and reused. Instead of first clearing the entire screen and afterwards initializing and drawing already visible POIs again, POIs that shall still be visible after the redraw, can be moved, POIs outside the field of view be deleted, and only POIs being new inside the field of view, be initialized. Tab. III summarizes all reasons for using the `locationView`. Note that the reasons for using the approaches of the `locationView` are not iPhone-specific and, therefore, can be also applied to other platforms.

Listing 3: Resource-saving redraw process

```
-(void)didUpdateHeadingLocations:(NSArray *)locations {
    // array containing new visible locations
    NSMutableArray *array = [NSMutableArray arrayWithArray:locations];
    // iterate over the subviews (the AREALocationViews) of the locationView
    for(AREALocationView *view in self.locationView.subviews) {
        if([array containsObject:view.location]) {
            // if the location (subview) exists also in the new locations, just update its position on the screen
            [array removeObject:view.location];
            [view updateFrame];
        } else {
            // otherwise remove the subview from its superview
            [view removeFromSuperview];
        }
    }
    // the locations that were not yet visible (no subview on the locationView) must be initialized
    for(AREALocation *loc in array) {
        AREALocationView *view = [[AREALocationView alloc] initWithLocation:loc];
        [self.locationView addSubview:view];
    }
}
```

B. Sensors and their Data

The correct reading of sensors is done by the `AREASensorController`. Listing 4 shows its `init`-method. First, a `CMMotionManager` is initialized, which is necessary to read the acceleration sensor. Second, a `CMLocationManager` is initialized, which is responsible to read the GPS-sensor and compass.

The acceleration sensor is required to calculate the current attitude of the device based on three axes (cf. Fig. 9) [14]. The attitude is needed to correctly rotate the `locationView`, to adjust the compass, and particularly to determine the vertical field of view. Since its data has to be polled, a time interval for the query is defined. This interval should be as small as possible.

TABLE III: Reasons for using the `locationView`

Reasons	Benefit
View size of *580x580* pixels	Drawing POIs independently of the rotation
Rotating the POI	Altogether instead of each separately
Calculating whether POI is inside the field of view	Independently of the device's rotation
Direct access to POI	Reusing POI during the redraw process

Otherwise, calculations might result in smearing, bucking, and inaccuracies. Furthermore, the compass data is required to calculate the angle of view and the GPS-sensor to determine the current location. These sensors push their data to methods defined in protocol `CLLocationManagerDelegate`. Since AREA must provide high accuracy and reliability, some adjustments in the `CMLocationManager` had to be made. First, all compass data, no matter how small the difference to the previous measured value is, must be queried. Hence, the `headingFilter` is set to the constant `kCLHeadingFilterNone` in Line 10, which conforms to the value zero degree. Second, the `distanceFilter` of the GPS-sensor is set to the constant `kCLDistanceFilterNone` in Line 12, whereby all changes in position are received. Finally, `desiredAccuracy` is adjusted to get the most accurate data from the GPS-sensor.

Listing 4: `init`-method of `AREASensorController`

```
1  -(id)init
2  {
3      if(self = [super init]) {
4          _motionManager = [[CMMotionManager alloc] init];
5          // for reliable acceleration data, the update frequency must be really high.
6          // 1/90.0 seconds is high enough, so no lagging will be visible
7          self.motionManager.accelerometerUpdateInterval = 1.0/90.0;
8          _locationManager = [[CLLocationManager alloc] init];
9          // no heading filter, therefore all heading updates will be received
10         self.locationManager.headingFilter = kCLHeadingFilterNone;
11         // location updates every 10 meters
12         self.locationManager.distanceFilter = kCLDistanceFilterNone;
13         // with the highest possible accuracy. ATTENTION: High battery usage!
14         self.locationManager.desiredAccuracy = kCLLocationAccuracyBestForNavigation;
15         self.locationManager.delegate = self;
16     }
17     return self;
18 }
```

In Listing 5, the reading of the sensors is started. Since the data of the acceleration sensor must be polled, an `NSOperationQueue` is created in Line 7 and the data is polled in a separate thread. Since only the gravitation is interesting for calculating the device's attitude, a *low-pass filter* is implemented inside this queue [14], and the result is stored in the instance variables `_xAcc`, `_yAcc`, and `_zAcc`. During the design of AREA, we described that `AREASensorController` implements an application loop. In turn, this loop is initialized with a time interval and started in Line 17. Inside the application loop, all data, including data of the acceleration sensor, the GPS-sensor, and the compass, are gathered and sent to a `delegate`. In this particular case, the delegate corresponds to `AREALocationController`, which is responsible for the calculation. Therefore, the delegate must implement the protocol shown in Listing 6. The first method will be called when the position based on the GPS-sensor has changed, the second will be executed in every iteration of the application loop. In the context of these methods, the calculations of POIs are performed. We will describe these in the next section.

Listing 5: Start querying sensors and the application loop

```
1  -(void)startSensoring
2  {
3      [self.locationManager startUpdatingLocation];
4      [self.locationManager startUpdatingHeading];
5
6      // this queue is polling the acceleration data.
7      self.motionQueue = [[NSOperationQueue alloc] init];
8      [self.motionManager startAccelerometerUpdatesToQueue:self.motionQueue withHandler:^(CMAccelerometerData *data, NSError *
           error) {
9          _xAcc = (data.acceleration.x * 0.1) + (_xAcc * (1.0 - 0.1));
10         _yAcc = (data.acceleration.y * 0.1) + (_yAcc * (1.0 - 0.1));
```

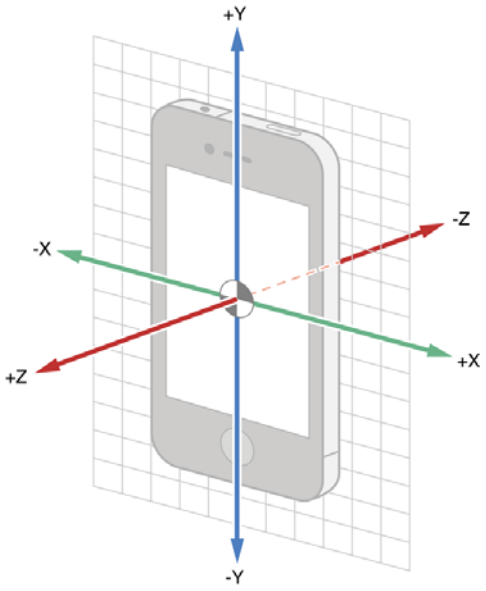


Fig. 9: The three axes of the iPhone's acceleration sensor [14]

```
        _zAcc = (data.acceleration.z * 0.1) + (_zAcc * (1.0 - 0.1));
    }];

    // this is the main loop of the sensor controller. Every 1/30.0 seconds the sensor data will be
    // collected and sent to its delegate.
    // In addition, every 1/30.0 seconds the screen will be redrawn
    self.timer = [NSTimer scheduledTimerWithTimeInterval:1.0/30.0 target:self selector:@selector(updateSensors) userInfo:nil
         repeats:YES];
}
```

Listing 6: Protocol `AREASensorControllerDelegate`

```
// Protocol to delegate updates of sensors to a delegate
@protocol AREASensorControllerDelegate
-(void)didUpdateToLocation:(CLLocation *)newLocation;
-(void)didUpdateBearingX:(double)newBearingX andBearingY:(double)newBearingY andBearingZ:(double)newBearingZ andHeading:(
    double)newHeading;
@end
```

C. Calculations inside the Controller

This section introduces the calculations inside the `AREALocationController`. Particularly, we discuss the gathering of POIs from the user's surrounding, the calculation of the field of view based on sensor data, implementing the formulas from Section 3, and the correct placement of POIs on the device's screen.

1) Calculating the Surrounding of POI.: When `AREASensorController` has received a new position, the POIs inside a certain radius around the user must be gathered and calculated. `AREASensorController` is informed about the new position by the `didUpdateToLocation:`-method, as defined in Listing 6, whereby the new position is received by a parameter. This approach is shown in Listing 7.

Listing 7: Calculating surrounding POIs and their vertical and horizontal heading

```
-(void)didUpdateToLocation:(CLLocation *)newLocation
{
    self.currentLocation = newLocation;
    NSMutableArray *locations = [NSMutableArray array];

    for(AREALocation *loc in self.store.store) {
        double distance = [loc.location distanceFromLocation:newLocation];
        if(distance <= self.maxDistance) {

            // the horizontal heading with values between 0 and 359 degrees
```

```
11              double horizontalHeading = [self calculateHorizontalHeadingFromLocation:newLocation toLocation:loc.location];
12              // the vertical heading with values from +90 (top) to -90 (bottom) degrees
13              double verticalHeading = [self calculateVerticalHeadingFromLocation:newLocation toLocation:loc.location
                    withDistance:distance];
14              loc.distance = distance;
15              loc.horizontalHeading = horizontalHeading;
16              loc.verticalHeading = verticalHeading;
17              [locations addObject:loc];
18          }
19      }
20
21      ...
22
23      // calculation is finished. Afterwards, call the delegate method with the current distance, the surrounding locations
            and the current user location
24      [self.delegate didUpdateLocations:self.surroundingLocations inDistance:self.maxDistance fromLocation:newLocation];
25  }
```

For each POI stored in `AREAStore`, at first, it is checked whether it is contained in a certain radius around the new position. This happens in Line 7 based on the *haversine formula*, which the iOS already implements internally. If the distance between the POI and the current position is less or equal to the radius (`maxDistance`), the horizontal (Line 11) and vertical heading (Line 13) based on the current position are calculated and stored together with the determined distance in the POI's data object. Finally, the `didUpdateLocations:inDistance:fromLocation:`-method from protocol `AREASensorControllerDelegate` is called on the `delegate`, in this case implemented by `AREAViewController`, informing it about the needed redraw. Further, all POIs inside the radius are cached in Line 17. They are later required to calculate their positions on the screen.

In Listings 8 and 9, the calculations of the vertical and horizontal heading of POIs are shown. Primarily, they constitute the implementations of Formulas (4) and (5) from Section 3. The former formula returns a value between $0°$ and $359°$, while the latter delivers a value between $-90°$ and $+90°$. Since results are floating-point numbers, the one the horizontal heading needs to be adapted to the mentioned interval by a modulo calculation, performed by the `fmod(double, double)` function (cf. Line 11). With this function, it becomes possible to apply the modulo operator on floating-point numbers.

Listing 8: Calculating the horizontal heading

```
1  -(double)calculateHorizontalHeadingFromLocation:(CLLocation *)loc1 toLocation:(CLLocation *)loc2
2  {
3      double lat1 = radians(loc1.coordinate.latitude);
4      double lon1 = radians(loc1.coordinate.longitude);
5      double lat2 = radians(loc2.coordinate.latitude);
6      double lon2 = radians(loc2.coordinate.longitude);
7      double deltaLon = lon2-lon1;
8      double x = sin(deltaLon) * cos(lat2);
9      double y = cos(lat1) * sin(lat2) - sin(lat1) * cos(lat2) * cos(deltaLon);
10     double heading = atan2(x, y) * 180.0 / M_PI;
11     return fmod(heading + 360.0, 360.0);
12 }
```

Listing 9: Calculating the vertical heading

```
1  -(double)calculateVerticalHeadingFromLocation:(CLLocation *)loc1 toLocation:(CLLocation *)loc2 withDistance:(double)distance
2  {
3      double locHeight = loc2.altitude;
4      double myHeight = loc1.altitude;
5      double dHeight = 0.0;
6      int sign = 0;
7      if(locHeight >= myHeight) {
8          dHeight = locHeight - myHeight;
9          sign = 1;
10     } else {
11         dHeight = myHeight - locHeight;
12         sign = -1;
13     }
14     double verticalHeading = dHeight/distance;
15     return sign * degrees(atan(verticalHeading));
```

2) Calculating the Field of View.: After each cycle of the application loop, the most recent data provided by the compass and acceleration sensor is sent to `AREASensorController`. This means that heading and attitude of the device might have changed and, thus, a re-calculation of the field of view must be performed.

Therefore, it is necessary to determine the maximal angle of view in height and width, since, as already shown above, the POIs are drawn on the `locationView` with a size of $580x580$ pixels. This means that the actual angle of view of the iPhone's camera of $56°$ in width and $44°$ in height must be increased proportionally to the size of the new area. Since it does not matter whether this conversion is done based on the horizontal or vertical angle of view, the following calculation is performed.

$$\theta = \frac{width_{new}}{width_{old}} * FOVwidth_{old} = \frac{580px}{480px} * 56° = 67,6° \tag{9}$$

Thereby, $FOVwidth_{old}$ corresponds to the field of view calculated in Section 3 (cf. Formula (6)). Using Formula (9), the maximal angle of view has a height and width of $67.6°$. As illustrated by Fig. 10, the user still can only use an angle view of $56°$ and $44°$. The new maximal angle of view is used only for internal calculations as well as the rotation of POIs.

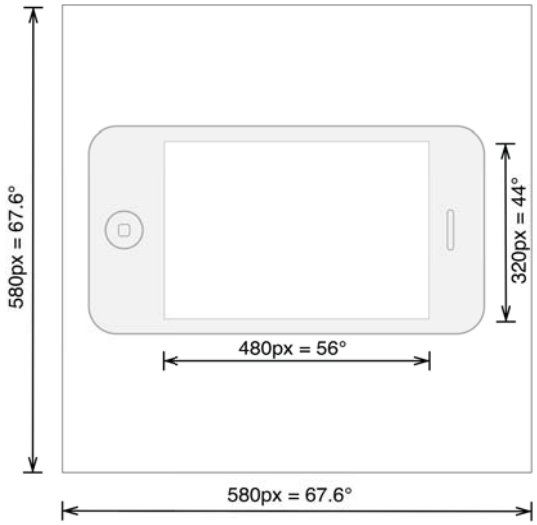


Fig. 10: Illustration of the new maximal angle view and the real one

With the maximal angle of view, it now becomes possible to calculate the current field of view. Listing 10 shows the code snippet for this purpose. The compass data and the horizontal heading will be only correct if the device is in portrait mode. If the device is rotated, for example to landscape mode, the compass data is still handled as it is in portrait mode by iOS. To handle this issue, adding a value of $90°$ to the heading might improve the situation in this particular case. However, since continuous values are required (i.e., the rotation of the device is continuous), the current rotation is calculated by the gravity values of the `x`- and `y`-axes of the acceleration sensor [15] in Lines 4 and 5. Fig. 11 illustrates how the real heading should look like, whereby the red arrow indicates heading without adaption and the blue arrow with adaption. This value is converted to radians and normalized in Line 7. The new real heading, according to the device's current rotation, is then defined in Line 8. The resulting horizontal field of view can then be calculated in Lines 9 and 10, whereby the constant `DEGREES_IN_VIEW` has the value from the result of Formula (9). The left boundary of the current field of view is calculated by the current heading and decreased by the half of the maximal angle of view ($heading - \frac{67.6°}{2}$), defined as `leftAzimuth`. The right boundary is calculated by adding half of the maximal angle of view to the current heading ($heading + \frac{67.6°}{2}$). Both values are normalized to values between $0°$ and $359°$. Since POIs have also a vertical heading, a vertical field of view must be calculated as well. This happens in analogy to the calculation of the horizontal field of view, except that the data of the acceleration sensor's `z` axis is required and values between $-90°$ and $+90°$ are calculated (cf. Lines 11 to 13).

Listing 10: Calculating the actual field of view

```
1  -(void)didUpdateBearingX:(double)newBearingX andBearingY:(double)newBearingY
2  andBearingZ:(double)newBearingZ andHeading:(double)newHeading {
3      // calculate the rotation of the device in degrees referenced to portrait
4      mode double deviceRotation = atan2(-newBearingY, newBearingX);
5      deviceRotation = deviceRotation - M_PI/2;
6      // calculate real heading relative to top, and in whatever orientation the device currently is
7      double headingOffset = fmod((deviceRotation*180.0/M_PI)+360.0, 360.0);
8      double heading = fmod(newHeading + headingOffset, 360.0);
9      double leftAzimuth = fmod(heading - DEGREES_IN_VIEW/2.0 + 360.0, 360.0);
10     double rightAzimuth = fmod(heading + DEGREES_IN_VIEW/2.0 + 360.0, 360.0);
11     double verticalHeading = degrees(asin(newBearingZ));
12     double topAzimuth = fmod(verticalHeading+DEGREES_IN_VIEW/2.0, 180.0);
13     double bottomAzimuth = fmod(verticalHeading-DEGREES_IN_VIEW/2.0, 180.0);
14 }
```

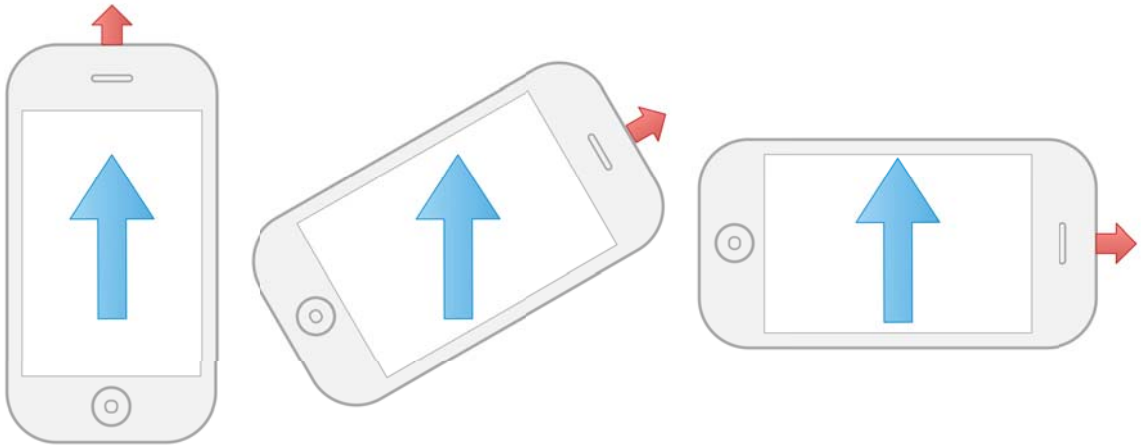


Fig. 11: Adjusting the compass data to the device's current rotation

3) Placement of POIs on the Camera View.: Once the horizontal and vertical fields of view have been calculated, the next step is to determine whether a POI is inside or not. Therefore, the horizontal heading of a POI is compared with the right and left boundary of the current field of view. If the POI's heading is greater than the left boundary and smaller than the right one, the POI is located in the horizontal field of view. However, several cases must be distinguished since the compass has a modulo transition from $359°$ to $0°$. This means that separate considerations must be made when the left boundary of the field of view is greater than $291.4°$, or the right boundary is smaller than $67.6°$. This value indicates the maximal field of view of $67.6°$. Fig. 12 illustrates the different cases. In order to display a POI on the camera view, it also must be in the vertical field of view. As opposed to the horizontal field of view, only one case must be considered, i.e., it must be determined whether the POI's vertical heading is smaller than the upper and greater than the lower boundary. Listing 11 presents this procedure in a code snippet. In Lines 4 and 14, the different cases of the horizontal field of view calculations are shown. This approach is also applicable to other operating systems and platforms. Thus, the calculations in Listing 11 can be used as a blue print to specify POIs' positions on the screen.

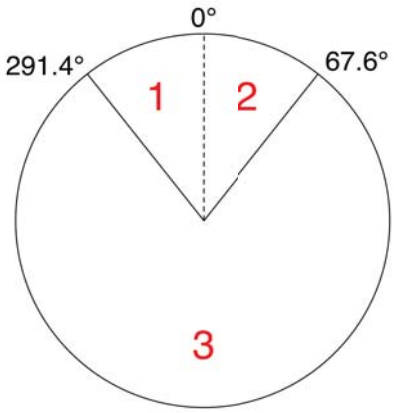


Fig. 12: The three cases of the field of view's case differentiation

Listing 11: Determining the POI's x- and y-coordinate on the screen

```
1  -(void)didUpdateBearingX:(double)newBearingX andBearingY:(double)newBearingY andBearingZ:(double)newBearingZ andHeading:(
       double)newHeading {
2      ...
3      for(AREALocation *loc in self.surroundingLocations) {
4          if ((rightAzimuth <= DEGREES_IN_VIEW || leftAzimuth >= 360-DEGREES_IN_VIEW) && loc.verticalHeading <= topAzimuth &&
                loc.verticalHeading >= bottomAzimuth) {
5              if (leftAzimuth <= loc.horizontalHeading) {
6                  loc.point = CGPointMake((PIXEL_DEGREE) * (loc.horizontalHeading-leftAzimuth), (PIXEL_DEGREE) * (topAzimuth -
                       loc.verticalHeading));
```

```
7                   [locations addObject:loc];
8               }
9               else if((rightAzimuth >= loc.horizontalHeading) && loc.verticalHeading <= topAzimuth && loc.verticalHeading >=
                     bottomAzimuth) {
10                  loc.point = CGPointMake((PIXEL_DEGREE) * (360 - leftAzimuth + loc.horizontalHeading), (PIXEL_DEGREE) * (
                        topAzimuth - loc.verticalHeading));
11                  [locations addObject:loc];
12              }
13          }
14          else if(leftAzimuth <= loc.horizontalHeading && loc.horizontalHeading <=rightAzimuth && loc.verticalHeading <=
                 topAzimuth && loc.verticalHeading >= bottomAzimuth){
15              loc.point = CGPointMake((PIXEL_DEGREE) * (loc.horizontalHeading-leftAzimuth), (PIXEL_DEGREE) * (topAzimuth - loc
                    .verticalHeading));
16              [locations addObject:loc];
17          }
18      }
19      [self.delegate didUpdateDeviceOrientationToAngle:deviceRotation];
20      [self.delegate didUpdateHeadingLocations:locations];
21      [self.delegate didUpdateHeadingWithRotation:newHeading];
22  }
```

When a POI is inside the field of view, x- and y-coordinates must be calculated to draw it correctly on the device screen afterwards. Again, different cases must be distinguished. Depending on whether the POI's horizontal heading is inside the first, second, or third area (cf. Fig. 12), the conversion of the heading to an x-coordinate is performed in different ways. If the POI is inside the first or third area, the difference between its horizontal heading and the left boundary must be calculated and multiplied by the constant PIXEL_DEGREE (cf. Lines 6 and 15). The constant defines how many pixels match to exactly one degree on the screen and is calculated by $\frac{480}{56}$, whereas the screen width is divided by the real horizontal angle of view. With this difference, a number of degrees is calculated. The POI is then shifted by this number from the left boundary to the right. If the POI is inside the third area, the zero point between the left boundary and the POI's horizontal heading will be leaped. Hence, at first, the value of the left boundary is subtracted from $360°$. Thus, the distance between the left boundary and the zero point is obtained. Afterwards, the POI's horizontal heading is added and multiplied by PIXEL_DEGREE. By calculating the y-coordinate, no different cases must be considered, since the interval reaches from $-90°$ to $+90°$. The coordinate is determined by the difference of the upper boundary and the POI's vertical heading, and is again multiplied by PIXEL_DEGREE.

After these calculations have been performed, AREAViewController is informed. Therefore, the controller must implement protocol AREALocationControllerDelegate (cf. Listing 12). The method in Line 2 is invoked, when the position of the device based on the GPS-sensor has changed and the POIs around the device have been gathered within a certain radius. The other methods are called whenever the field of view is calculated, i.e., after each cycle through the application loop, which is defined in AREASensorController. These methods provide the AREAViewController with information about by how many degrees the locationView must be rotated, which POIs have to be drawn on the locationView, and by how many degrees the compass and radar must be rotated in order to display them correctly, even when the device is hold obliquely.

Listing 12: AREALocationControllerDelegate

```
1  @protocol AREALocationControllerDelegate
2  -(void)didUpdateLocations:(NSArray *)locations inDistance:(CLLocationDistance)distance fromLocation:(CLLocation *)location;
3  -(void)didUpdateHeadingLocations:(NSArray *)locations;
4  -(void)didUpdateHeadingWithRotation:(double)rotation;
5  -(void)didUpdateDeviceOrientationToAngle:(double)radianAngle;
6  @end
```

V. Survey

This section presents the results of a user survey that was conducted to evaluate AREA. A total of 26 people from four areas and different ages have been interviewed:

- (1) People working in computer science,
- (2) people who work in the IT business domain,
- (3) people from other business domains, for which mobile business applications are useful, and
- (4) people which use sophisticated mobile applications in their leisure time have been involved in the survey.

The survey contained general questions about the use of their smartphones and particularly their knowledge about augmented reality. Moreover, it contains specific questions regarding the usability and quality of the individual functions of AREA.

To increase the significance of the results, a heterogenoeus sample of participants has been chosen. This is illustrated by Fig. 13.

In order to evaluate the usability of the most important features of AREA (e.g., guidance with the radar), participants have been asked about their perception to these features. The overall result has been very positive, since almost all main AREA features have been conceived positive and intuitive (cf. Figure 16). In particular, the radar feature and the interaction style with POIs have been perceived very positive. Moreover, for the radar and map feature some more considerations for improve their usability have to be taken into account.

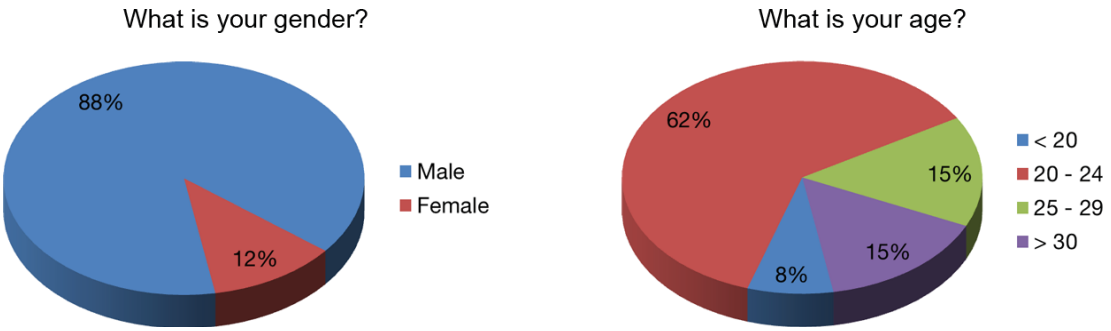


Fig. 13: Statistical Questions

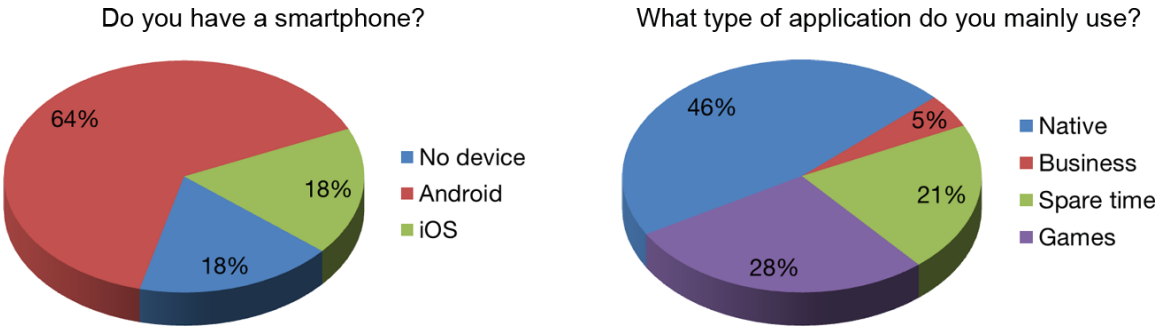


Fig. 14: General Smartphone Questions

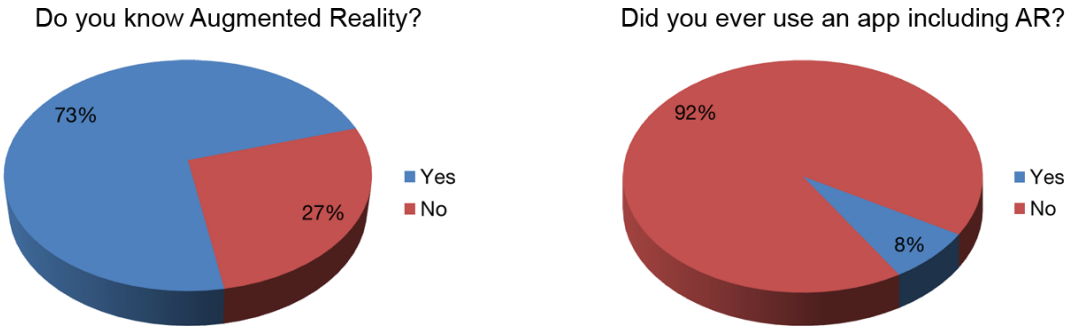


Fig. 15: General Augmented Reality Questions

VI. Architecture of an Application Realized on top of AREA

The overall architecture of an application on top of AREA, the connections between engine and application, as well as managing POIs stored in a database on a remote server or file, are shown in Fig. 17. The model of AREA must link to a file containig the POIs or to a remote server running a script to query POIs stored in a database. The script queries the database and returns corresponding POIs back to the requestor. The returned POIs have to be in a valid and known format that can be processed by AREA. AREA then processes the returned POIs and stores them in a local database. Querying the remote database to get the an initial set of POI must only be accomplished at the very first start of AREA. Hence, it is possible to use AREA and an application on top even without network connection after loading the initial set of POI. Since AREA is implemented on iOS, we use the *Core Data* database. The application based on AREA has a connection to AREA's model and consequently can use the stored POIs to display them also on a map, or inside a list. Thus, AREA and applications based on AREA can use the same model, which is then queried and processed by the engine. The application on top uses functions of AREA. To use the camera view of AREA, the application on top has a connection to the main controller of AREA. By using this connection, in turn, the camera opens and shows the POIs that were previously stored in the model and inside a local database.

In Section 8, we will introduce *LiveGuide Ditzingen*. LiveGuide is an application that makes use of AREA's features and is built right on top of AREA. The architecture of LiveGuide complies with the architecture mentioned in this section and can be seen as a blue print to integrate AREA in a third party application.

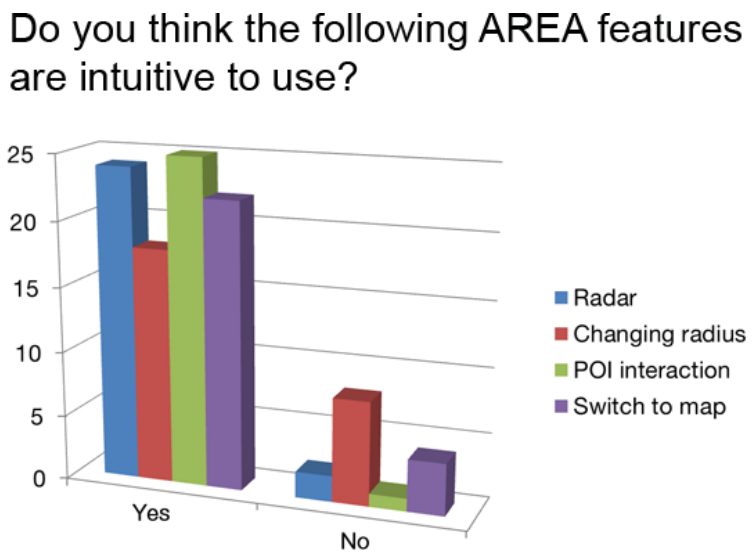


Fig. 16: Perception of AREA Features

VII. AREA Engineering Process

In this section, the engineering processes of the different parties involved in the development of an augmented reality application are discussed.

A. Involved Developer Parties

The engineering process of developing and implementing an application containing augmented reality functionality like AREA can be divided according to the two groups of developers involved (cf. Fig. 18).

The *engine developer* is responsible for developing and implementing a robust, efficient, modular, and extendable core engine. The engine developer must provide interfaces to realize a customizable engine (e.g. to adjust the minimal and maximal radius), and to define the model to different other applications based on it.

In turn, the *application developer* is responsible for developing and implementing applications on top of AREA that uses its functionality. Independent of AREA, she also develops and implements the features of the application that do not rely on the engine's functionality. She further coordinates the unified data format of POIs with the engine developer. Therefore, it must be elaborated in which data format (e.g., *XML*, or *JSON*) the POIs shall be saved and which fields and attributes are required. The *engine developer* must implement the model and also may alter the overall data processing. Furthermore, if it becomes necessary to import previously defined POIs into the model, a connection to a web server, or a file containing these POIs, must be considered. After defining the final model and bringing in the POIs, the model can be referenced and accessed from the application and the data can be used. Following this, the next step treats with the handling of the augmented reality view, i.e., the camera view of AREA. Therefore, the *application developer* only needs a reference to AREA's main controller. The engine then opens the camera and starts processing the POIs fully autonomously. To integrate AREA into the application, the *application developer* may customize the behavior. For example, he may adjust minimal and maximal radius or the appearance of the user interface elements used by the camera view.

B. Answers to the Engineering Process Topics

In Section 2, we introduced the engineering process comprising the different addressed aspects (cf. Tab. II). We now complete Tab. II by giving answers based on the previous sections. The first question of the architecture's topic has been how to design the architecture. Our approach proposes to use the MVC pattern for AREA. Thus, it becomes possible to exchange the model without necessarily changing the controller or view. Furthermore, by separating the controller, changes related to sensor querying, e.g., adjusting the frequency of polling sensors or redrawing the screen, can be easily realized. It further becomes possible to replace the view, e.g., to realize an individual look and feel of the application based on AREA. The second architectural question can be answered by simply defining a model and referencing the main controller of AREA to start the camera view.

The answer to the first question of the performance's topic is to use the previously mentioned mathematical formulas and poll the sensors' recorded data every $\frac{1}{30}$ seconds instead of letting the sensors push their data in a much higher frequency to the controller. Furthermore, by only calculating those coordinates of POIs on the screen, which are located inside a specific radius,

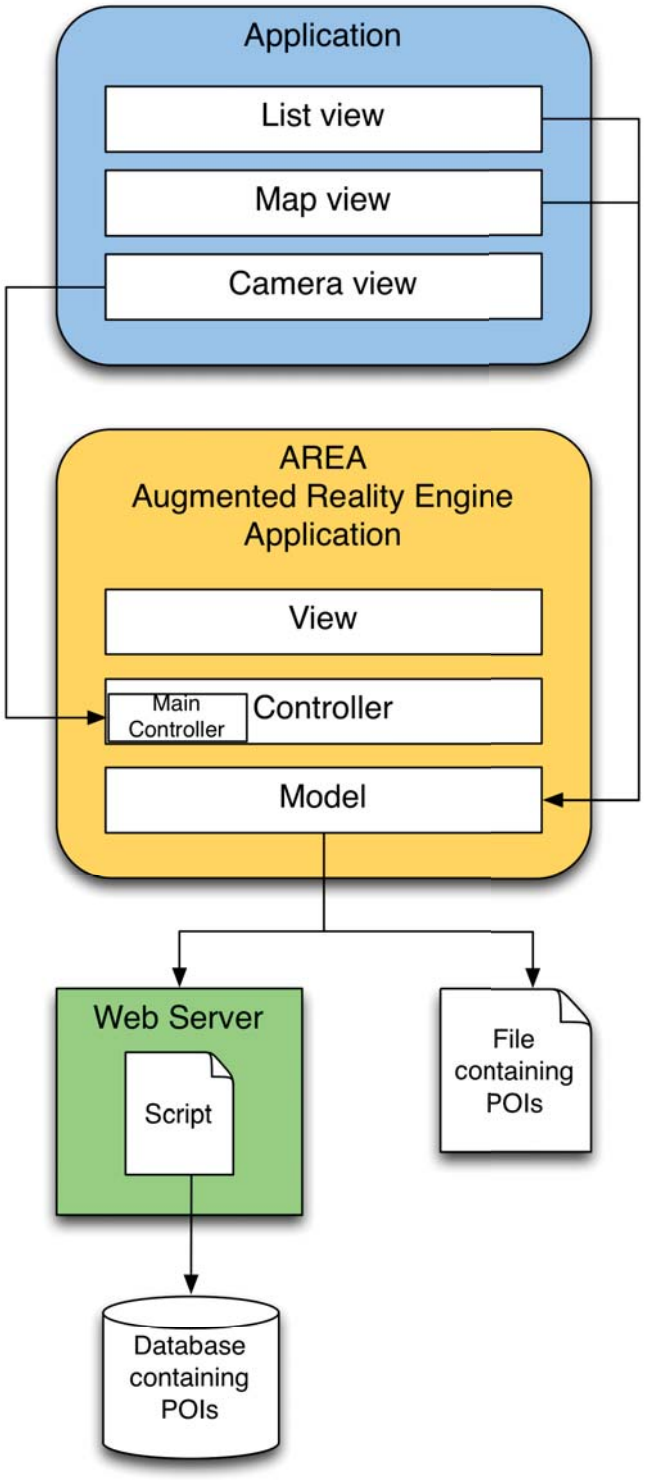


Fig. 17: Overall Architecture of AREA and an application on top

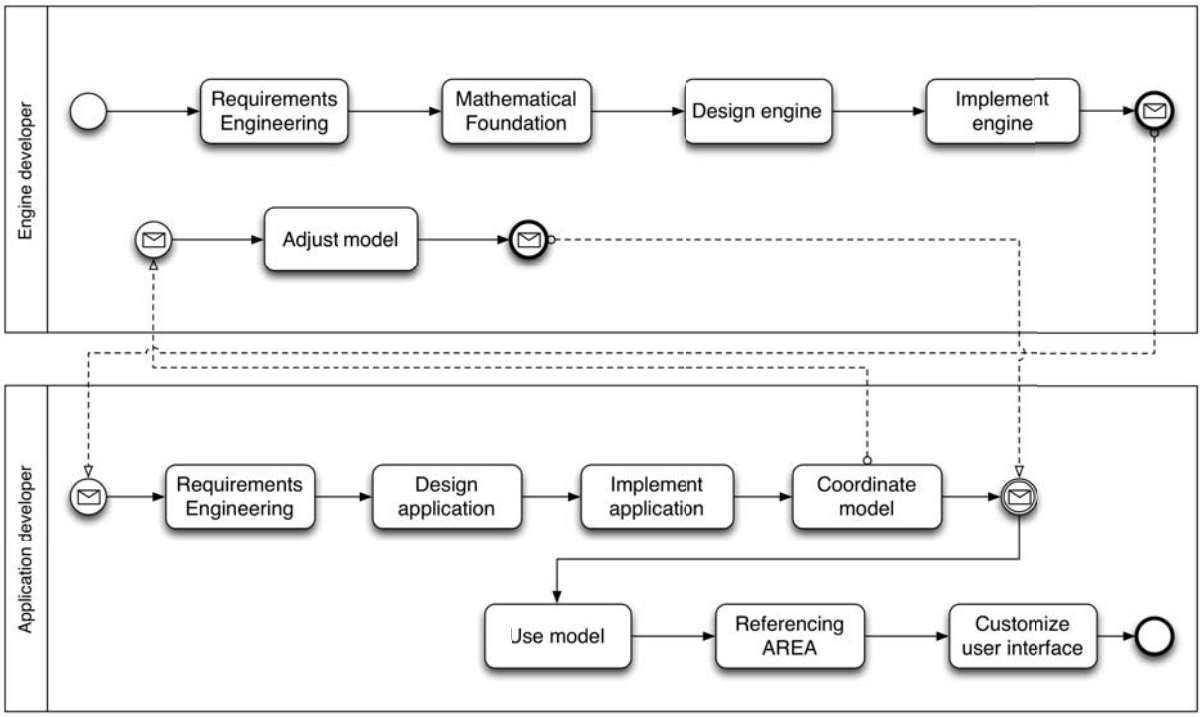


Fig. 18: Engineering process of engine and application developer in BPM notation [16]

this contributes to the first performance question. The answer to the second question then is to use the `locationView` to rotate POIs, calculate the rotation aware angle of view, and reuse POIs on the camera view instead of initializing new ones.

To keep the camera view understandable and usable, a radar as well as a compass are added (cf. Figure 1). Thus, a user gets help in which direction she must turn, to make POIs visible. POIs, in turn, are transparent and are not fully opaque. Thus, the background (i.e., the recorded reality) is not completely overlayed by POIs.

The communication problem can be answered as follows. AREA constitutes a modular design. Therefore, only three connections must be considered when using it: (1) the connection to a data set, which may be stored on a remote server or in a file, (2) a reference to the engine for integrating AREA into other applications, and (3) a reference must be made to directly access the data model. In summary, AREA constitutes an approach in which the main components are loosely coupled.

C. Evaluation of the Engineering Process Topics

Our approach to design AREA as a layered architecture has turned out to be very useful. Furthermore, all layers may be easily replaced. For example, to change the appearance of the user interface elements presented in the camera view, only the layer containing the view must be replaced.

The reuse of visible POIs inside the camera view, the advantages of using the `locationView`, and the efficient implementations of calculations and readings of sensors, emphasize that AREA is both efficient and reliable. However, we identified additional challenges when implementing the engine as well as the application on top of it. First, the compass data of the iPhone and the underlying iOS is partly rough. For example, it is not possible to get notified about changes smaller than 1 degree. Therefore, the POIs inside the camera view seem to jump when moving the device horizontally. This effect is caused by the ratio between the camera's horizontal angle of view and the width of the device's display of $\frac{480}{56} \approx 8$. It means that for every degree changed, the POI is moved 8 pixels from its previous position. Another challenge has been the required transparency of POIs. During the development of LiveGuide Ditzingen (i.e., the application on top of AREA, cf. Section 8) it has turned out that a high amount of very close POIs causes overlapping in a bad manner. The internal calculations of each POI's transparency require a high amount of time and the engine starts slowing down. To prevent the engine of slowing down while drawing the screen, *OpenGL ES* may be used as a solution. By using OpenGL, the *GPU* is used for drawing the screen instead of the *CPU*, which is then not blocked anymore during the redraw process.

AREA provides a radar on the camera view. This is a two-dimensional user interface element and shows all POIs of the user's surrounding independent of their altitude, i.e., if no POI is visible on the actual camera view, but the radar indicates that POIs

are located in the current user's direction, the POI is located above or below the users vertical heading. To support users, small arrows may be used to indicate the vertical direction of not yet visible POIs. With the use of arrows the usability of finding not yet visible POIs can be improved.

Tab. IV summarizes the engineering process topics, their answers and evaluations ($++$ indicates *very good* and $--$ indicates *very bad*).

TABLE IV: Topics of the engineering process, their questions, answers, and validations

Topic	Questions	Answers	Validation
Architecture	How to design the architecture of AREA making it extendable and modular?	Use the MVC pattern for AREA.	++
	How to easily implement applications based on AREA?	Define a model and make a reference to the main controller of AREA.	++
Performance	How to implement an efficient way to update POIs in realtime?	Use well-known formulas, poll data of sensors, only calculate POI inside the radius.	++
	How to provide a quick redraw process?	Use the locationView, reuse POIs.	+
Usability	How to make the user interface intuitive and usable?	Add a radar, compass, and field of view; make POIs transparent.	+
Communication	How perform communication between AREA and a remote server?	Only three main connections are used. One to the dataset, one to the model, and one to the main controller.	++

VIII. Case Study

This section illustrates an application realized on top of AREA. The application is called *LiveGuide Ditzingen*. Its implementation is based on AREA and the overall implementation procedure is accomplished as described in Section 7. In this section, we present the requirements and features of the LiveGuide application.

A. Requirements

The application shall display points of interest like public buildings, bus stations, or parks in the city of Ditzingen as POIs inside the camera view of AREA. Ditzingen is a city located in the south of Germany. Its population is about $25,000$ and it encompasses an area of $30.4\ km^2$. The POIs of Ditzingen shall be stored on a remote server. These POIs shall be displayed on a map view as well as in a list view. Since the application is mainly used by tourists, who might not have an Internet connection while using the app, it should be further possible to store the POIs locally on the user's smart device. Furthermore, information of POIs may change and additional POIs be added. Hence, an option is needed to update POIs with the remote server.

Overall, the data set of Ditzingen's POIs comprises 250 points of interests grouped in 15 different categories. These groups must be particularly considered when displaying the POIs. For example, POIs might have a picture of the corresponding location, which must be depicted in the application as well.

B. LiveGuide Ditzingen

LiveGuide Ditzingen is available on the Apple App Store [17]. It is noteworthy that all requirements and features of AREA (cf. Section 2), and *LiveGuide Ditzingen* have been used and implemented. In the following, we discuss requirements which posed particular considerations.

Our first consideration is related to the remote communication requirement of the *LiveGuide Ditzingen* application. The application shall only use a network connection when updating existing POIs by using a remote server. Therefore, to load POIs without a network connection with AREA, these POIs had to be queried from a database, stored in a file, and put into the project folder. This file is then used to import POIs at the very first start of the application into a local database on the device. Since these POIs may change on the remote database over time, a script had to be developed and deployed on a remote server. In turn, this script is used by the application to find new POIs, download them from the database onto the device, and store them again in the device's local database. The update schedule proceeds as follows: the POIs have a unique *id* that is the same on the remote and the local database. The application queries the script on the remote server with the currently highest *id* in the local database. If the remote database has POIs with a higher *id*, the script will send all POIs with a higher *id* than the queried one back to the device. This approach only works, if POIs in the database do not change their properties over time or are removed. To get also notified about changes of POIs' properties, each POI must be saved with a timestamp in the database. The timestamp can then be used to verify whether a POI's property has been changed by comparing the remote POI's timestamp and the local POI's timestamp. To get informed about removed POIs, an additional table can be used to store the *id* of removed POIs. This database must then be queried to get informed about removed POIs. To synchronize the data set of POIs, the returned *id*s of removed POIs must also be removed from the local database.

LiveGuide Ditzingen further allows displaying POIs in a list and additionally on a map. Since AREA provides a public interface to its model, *LiveGuide* can use the model to show the POIs in a list and on a map easily. The calculation of the distance between a POI and the user's current location, in turn, is only executed inside AREA when the camera view is visible. Consequently, another important consideration must be addressed. If the application shall show the distance between the user's current location and the POI also in the list and the map, the application developer must implement a function to calculate the distance. Without this function, the distance of each POI would only be updated when AREA's camera view is visible.

Fig. 19 presents screenshots of the *LiveGuide Ditzingen* application as deployed to the Apple App Store. Shown are (from left to right) the list view, the map view, and the camera view powered by AREA. All three views present POIs on a different way.

Fig. 19: Screenshots of LiveGuide Ditzingen

IX. Related Work

Previous research on engineering and developing a location-based augmented reality application based on GPS-coordinates, and sensors running on *head-mounted* displays are described in [18] and [19]. A simple mobile device, extended by additional sensors, has been used by [20] to develop an augmented reality system. An application using augmented reality has also been

developed in [2] with the purpose of sharing media and information in a real world environment and letting users interact with this data by augmented reality. However, none of these approaches addresses location-based augmented reality on mobile devices in particular. Moreover, they do not provide basic insights into how to develop such applications.

The increasing growth of the smartphone market and the technical maturity of mobile devices has driven commercial vendors to develop *augmented reality software development kits* as well, e.g., *Wikitude* [3], *Layar* [21], and *Junaio* [22]. Moreover, very popular applications, e.g., *Yelp* [23], are using the additional features of augmented reality to support users in interacting with their surrounding. Besides, only little research can be found, which addresses the development of augmented reality systems in general. For example, [24] validates existing augmented reality browsers. Again, both commercial vendors and general research results related to augmented reality provide no insights how to develop a location-based mobile augmented reality engine.

Furthermore, there exist approaches focusing on sophisticated mobile applications not related to augmented reality [1], [25], [26]. Most of them present similar challenges to be considered like user acceptance and security. But they do not focus on the entire engineering process for developing sophisticated mobile applications.

Finally, from the research field of business process management, many approaches address the development of mobile applications [25], [27]. This research field offers a wide range of flexibility concepts [25], which are particularly useful for a proper exception handling. Since such handling is important for mobile appplications, the latter frequently use these concepts as well. Again, none of them focus on the entire process to develop sophisticated mobile applications.

X. Summary

The purpose of our paper has been to present the engineering process for developing the core framework of an augmented reality engine for mobile devices. We have further shown how applications can be implemented based on its functionality. As discussed, such development is complex and challenging. First of all, a basic knowledge about mathematical calculations is required, i.e., formulas to calculate the distance and heading of points of interest on a sphere. Furthermore, it is necessary to know the various sensors of the smartphone, particularly how to determine and process the data provided by them. Another important issue concerns resource and energy consumption. Since smartphones have limited resources and performance capabilities, the points of interest should be displayed without delay. Therefore, the calculations to handle sensor data and the general screen drawing must be implemented as efficient as possible. The latter has been realized by implementing the `locationView`, increasing the field of view, and reusing already drawn points of interest. The suitable increased field of view realized the advantage to easily determine whether a point of view is inside the field of view without considering the smartphone's current rotation. In addition, all displayed points of interest can be rotated easily.

We argue that an augmented reality engine must particularly provide modularity to ensure a comprehensive and easy integration in existing applications as well as to implement new applications. Finally, it is necessary to design a proper architectural and class design, whereas the communication between the components must not be neglected.

In this paper, we have also shown how to integrate AREA in a real world application, namely LiveGuide Ditzingen, and how to make use of its functionality. The application, in turn, is available in the Apple App Store. The application has proven its robust functionality using AREA.

Future research work addresses the challenges we identified during the development of *AREA* and *LiveGuide Ditzingen*. Currently, AREA is only available on iOS and, therefore, only usable in applications running on Apple smart mobile devices. Thus, to address applications and developers of other platforms, we transfer AREA to run on the *Android* operating system. In addition, to use AREA in indoor scenarios where GPS is inaccurate or unavailable, the engine will be extended by features of *Computer Vision* as *Marker-based* and *Markerless Augmented Reality*.

References

[1] J. Schobel and M. Schickler and R. Pryss and H. Nienhaus and M. Reichert, "Using Vital Sensors in Mobile Healthcare Business Applications: challenges, Examples, Lessons Learned," *Int'l Conference on Web Information Systems and Technologies*, pp. 509–518, 2013.

[2] Lee, Ryong and Kitayama, Daisuke and Kwon, Yong-Jin and Sumiya, Kazutoshi, "Interoperable augmented web browsing for exploring virtual media in real space," p. 7, 2009.

[3] "Wikitude," [Online; accessed 11.06.2013].

[4] Sinnott, R.W., "Virtues of the Haversine," *Sky and telescope*, vol. 68:2, p. 158, 1984.

[5] Bullock, R, "Great Circle Distances and Bearings Between Two Locations," 06 2007, [Online; accessed 04.09.2012].

[6] Veness, C., "Calculate distance, bearing and more between Latitude/Longitude points," [Online; accessed 04.09.2012].

[7] Hohner, M., "Formelsammlung Kameras," [Online; accessed 05.09.2012].

[8] hotpaw2, "iPhone 4 Camera Specifications - Field of View / Vertical-Horizontal Angle," [Online; accessed 05.09.2012].

[9] Lumo, F., "Apple iPhone 4 camera specs," [Online; accessed 05.09.2012].

[10] Bautch, M., "Digitale bildgebende Verfahren," [Online; accessed 05.09.2012].

[11] cujo30227, "I Need iPhone 4 Camera Specifications - Field of View / Vertical-Horizontal Angle," [Online; accessed 05.09.2012].

[12] Krasner, Glenn E and Pope, Stephen T and others, "A description of the model-view-controller user interface paradigm in the smalltalk-80 system," *Journal of object oriented programming*, vol. 1, no. 3, pp. 26–49, 1988.

[13] Apple, “Xcode 4 Downloads and Resources - Apple Developer,” [Online; accessed 10.09.2012].

[14] “Apple Developer Guide,” [Online; accessed 11.06.2013].

[15] Alasdair Allan, “Basic Sensors in iOS: Programming the Accelerometer, Gyroscope, and More,” p. 47, 2011.

[16] “Business Process Model and Notation,” [Online; accessed 11.06.2013].

[17] “LiveGuide Ditzingen,” [Online; accessed 29.05.2013].

[18] Feiner, Steven and MacIntyre, Blair and HÖllerer, Tobias and Webster, Anthony, “A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment,” *Personal Technologies*, vol. 1, no. 4, pp. 208–217, 1997.

[19] Kooper, Rob and MacIntyre, Blair, “Browsing the real-world wide web: Maintaining awareness of virtual information in an AR information space,” *International Journal of Human-Computer Interaction*, vol. 16, no. 3, pp. 425–446, 2003.

[20] Kähäri, Markus and Murphy, David J, “Mara: Sensor based augmented reality system for mobile imaging device,” 2006.

[21] “Layar,” [Online; accessed 11.06.2013].

[22] “Junaio,” [Online; accessed 11.06.2013].

[23] “Yelp,” [Online; accessed 11.06.2013].

[24] Grubert, Jens and Langlotz, Tobias and Grasset, Raphaël, “Augmented reality browser survey,” *Tehnical report, Institute for Computer Graphics and Vision, Graz University of Technology, Austria*, 2011.

[25] R. Pryss and D. Langer and M. Reichert and A. Hallerbach, “Mobile Task Management for Medical Ward Rounds - The MEDo Approach,” *Proc. BPM’12 Workshops*, no. 132, pp. 43–54, 2012.

[26] Andreas Robecke and Rüdiger Pryss and Manfred Reichert, “DBIScholar: An iPhone Application for Performing Citation Analyses,” no. Vol-73, June 2011.

[27] R. Pryss and J. Tiedeken and U. Kreher and M. Reichert, “Towards Flexible Process Support on Mobile Devices,” *Proc. CAiSE’10 Forum*, pp. 150–165, 2010.

Liste der bisher erschienenen Ulmer Informatik-Berichte

Einige davon sind per FTP von `ftp.informatik.uni-ulm.de` erhältlich
Die mit * markierten Berichte sind vergriffen

List of technical reports published by the University of Ulm

Some of them are available by FTP from `ftp.informatik.uni-ulm.de`
Reports marked with * are out of print

91-01 *Ker-I Ko, P. Orponen, U. Schöning, O. Watanabe*
Instance Complexity

*91-02** *K. Gladitz, H. Fassbender, H. Vogler*
Compiler-Based Implementation of Syntax-Directed Functional Programming

*91-03** *Alfons Geser*
Relative Termination

*91-04** *J. Köbler, U. Schöning, J. Toran*
Graph Isomorphism is low for PP

91-05 *Johannes Köbler, Thomas Thierauf*
Complexity Restricted Advice Functions

*91-06** *Uwe Schöning*
Recent Highlights in Structural Complexity Theory

*91-07** *F. Green, J. Köbler, J. Toran*
The Power of Middle Bit

*91-08** *V.Arvind, Y. Han, L. Hamachandra, J. Köbler, A. Lozano, M. Mundhenk, A. Ogiwara, U. Schöning, R. Silvestri, T. Thierauf*
Reductions for Sets of Low Information Content

*92-01** *Vikraman Arvind, Johannes Köbler, Martin Mundhenk*
On Bounded Truth-Table and Conjunctive Reductions to Sparse and Tally Sets

*92-02** *Thomas Noll, Heiko Vogler*
Top-down Parsing with Simulataneous Evaluation of Noncircular Attribute Grammars

92-03 *Fakultät für Informatik*
17. Workshop über Komplexitätstheorie, effiziente Algorithmen und Datenstrukturen

*92-04** *V. Arvind, J. Köbler, M. Mundhenk*
Lowness and the Complexity of Sparse and Tally Descriptions

*92-05** *Johannes Köbler*
Locating P/poly Optimally in the Extended Low Hierarchy

*92-06** *Armin Kühnemann, Heiko Vogler*
Synthesized and inherited functions -a new computational model for syntax-directed semantics

*92-07** *Heinz Fassbender, Heiko Vogler*
A Universal Unification Algorithm Based on Unification-Driven Leftmost Outermost Narrowing

*92-08** *Uwe Schöning*
On Random Reductions from Sparse Sets to Tally Sets

*92-09** *Hermann von Hasseln, Laura Martignon*
Consistency in Stochastic Network

92-10 *Michael Schmitt*
A Slightly Improved Upper Bound on the Size of Weights Sufficient to Represent Any Linearly Separable Boolean Function

92-11 *Johannes Köbler, Seinosuke Toda*
On the Power of Generalized MOD-Classes

92-12 *V. Arvind, J. Köbler, M. Mundhenk*
Reliable Reductions, High Sets and Low Sets

92-13 *Alfons Geser*
On a monotonic semantic path ordering

*92-14** *Joost Engelfriet, Heiko Vogler*
The Translation Power of Top-Down Tree-To-Graph Transducers

93-01 *Alfred Lupper, Konrad Froitzheim*
AppleTalk Link Access Protocol basierend auf dem Abstract Personal Communications Manager

93-02 *M.H. Scholl, C. Laasch, C. Rich, H.-J. Schek, M. Tresch*
The COCOON Object Model

93-03 *Thomas Thierauf, Seinosuke Toda, Osamu Watanabe*
On Sets Bounded Truth-Table Reducible to P-selective Sets

93-04 *Jin-Yi Cai, Frederic Green, Thomas Thierauf*
On the Correlation of Symmetric Functions

93-05 *K.Kuhn, M.Reichert, M. Nathe, T. Beuter, C. Heinlein, P. Dadam*
A Conceptual Approach to an Open Hospital Information System

93-06 *Klaus Gaßner*
Rechnerunterstützung für die konzeptuelle Modellierung

93-07 *Ullrich Keßler, Peter Dadam*
Towards Customizable, Flexible Storage Structures for Complex Objects

94-01 *Michael Schmitt*
On the Complexity of Consistency Problems for Neurons with Binary Weights

94-02 *Armin Kühnemann, Heiko Vogler*
A Pumping Lemma for Output Languages of Attributed Tree Transducers

94-03 *Harry Buhrman, Jim Kadin, Thomas Thierauf*
On Functions Computable with Nonadaptive Queries to NP

94-04 *Heinz Faßbender, Heiko Vogler, Andrea Wedel*
Implementation of a Deterministic Partial E-Unification Algorithm for Macro Tree Transducers

94-05 *V. Arvind, J. Köbler, R. Schuler*
On Helping and Interactive Proof Systems

94-06 *Christian Kalus, Peter Dadam*
Incorporating record subtyping into a relational data model

94-07 *Markus Tresch, Marc H. Scholl*
A Classification of Multi-Database Languages

94-08 *Friedrich von Henke, Harald Rueß*
Arbeitstreffen Typtheorie: Zusammenfassung der Beiträge

94-09 *F.W. von Henke, A. Dold, H. Rueß, D. Schwier, M. Strecker*
Construction and Deduction Methods for the Formal Development of Software

94-10 *Axel Dold*
Formalisierung schematischer Algorithmen

94-11 *Johannes Köbler, Osamu Watanabe*
New Collapse Consequences of NP Having Small Circuits

94-12 *Rainer Schuler*
On Average Polynomial Time

94-13 *Rainer Schuler, Osamu Watanabe*
Towards Average-Case Complexity Analysis of NP Optimization Problems

94-14 *Wolfram Schulte, Ton Vullinghs*
Linking Reactive Software to the X-Window System

94-15 *Alfred Lupper*
Namensverwaltung und Adressierung in Distributed Shared Memory-Systemen

94-16 *Robert Regn*
Verteilte Unix-Betriebssysteme

94-17 *Helmuth Partsch*
Again on Recognition and Parsing of Context-Free Grammars:
Two Exercises in Transformational Programming

94-18 *Helmuth Partsch*
Transformational Development of Data-Parallel Algorithms: an Example

95-01 *Oleg Verbitsky*
On the Largest Common Subgraph Problem

95-02 *Uwe Schöning*
Complexity of Presburger Arithmetic with Fixed Quantifier Dimension

95-03 *Harry Buhrman,Thomas Thierauf*
The Complexity of Generating and Checking Proofs of Membership

95-04 *Rainer Schuler, Tomoyuki Yamakami*
Structural Average Case Complexity

95-05 *Klaus Achatz, Wolfram Schulte*
Architecture Indepentent Massive Parallelization of Divide-And-Conquer Algorithms

95-06 *Christoph Karg, Rainer Schuler*
Structure in Average Case Complexity

95-07 *P. Dadam, K. Kuhn, M. Reichert, T. Beuter, M. Nathe*
ADEPT: Ein integrierender Ansatz zur Entwicklung flexibler, zuverlässiger kooperierender Assistenzsysteme in klinischen Anwendungsumgebungen

95-08 *Jürgen Kehrer, Peter Schulthess*
Aufbereitung von gescannten Röntgenbildern zur filmlosen Diagnostik

95-09 *Hans-Jörg Burtschick, Wolfgang Lindner*
On Sets Turing Reducible to P-Selective Sets

95-10 *Boris Hartmann*
Berücksichtigung lokaler Randbedingung bei globaler Zieloptimierung mit neuronalen Netzen am Beispiel Truck Backer-Upper

95-11 *Thomas Beuter, Peter Dadam*:
Prinzipien der Replikationskontrolle in verteilten Systemen

95-12 *Klaus Achatz, Wolfram Schulte*
Massive Parallelization of Divide-and-Conquer Algorithms over Powerlists

95-13 *Andrea Mößle, Heiko Vogler*
Efficient Call-by-value Evaluation Strategy of Primitive Recursive Program Schemes

95-14 *Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
A Generic Specification for Verifying Peephole Optimizations

96-01 *Ercüment Canver, Jan-Tecker Gayen, Adam Moik*
Formale Entwicklung der Steuerungssoftware für eine elektrisch ortsbediente Weiche mit VSE

96-02 *Bernhard Nebel*
Solving Hard Qualitative Temporal Reasoning Problems: Evaluating the Efficiency of Using the ORD-Horn Class

96-03 *Ton Vullinghs, Wolfram Schulte, Thilo Schwinn*
An Introduction to TkGofer

96-04 *Thomas Beuter, Peter Dadam*
Anwendungsspezifische Anforderungen an Workflow-Mangement-Systeme am Beispiel der Domäne Concurrent-Engineering

96-05 *Gerhard Schellhorn, Wolfgang Ahrendt*
Verification of a Prolog Compiler - First Steps with KIV

96-06 *Manindra Agrawal, Thomas Thierauf*
Satisfiability Problems

96-07 *Vikraman Arvind, Jacobo Torán*
A nonadaptive NC Checker for Permutation Group Intersection

96-08 *David Cyrluk, Oliver Möller, Harald Rueß*
An Efficient Decision Procedure for a Theory of Fix-Sized Bitvectors with Composition and Extraction

96-09 *Bernd Biechele, Dietmar Ernst, Frank Houdek, Joachim Schmid, Wolfram Schulte*
Erfahrungen bei der Modellierung eingebetteter Systeme mit verschiedenen SA/RT–Ansätzen

96-10 *Falk Bartels, Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
Formalizing Fixed-Point Theory in PVS

96-11 *Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
Mechanized Semantics of Simple Imperative Programming Constructs

96-12 *Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
Generic Compilation Schemes for Simple Programming Constructs

96-13 *Klaus Achatz, Helmuth Partsch*
From Descriptive Specifications to Operational ones: A Powerful Transformation Rule, its Applications and Variants

97-01 *Jochen Messner*
Pattern Matching in Trace Monoids

97-02 *Wolfgang Lindner, Rainer Schuler*
A Small Span Theorem within P

97-03 *Thomas Bauer, Peter Dadam*
A Distributed Execution Environment for Large-Scale Workflow Management Systems with Subnets and Server Migration

97-04 *Christian Heinlein, Peter Dadam*
Interaction Expressions - A Powerful Formalism for Describing Inter-Workflow Dependencies

97-05 *Vikraman Arvind, Johannes Köbler*
On Pseudorandomness and Resource-Bounded Measure

97-06 *Gerhard Partsch*
Punkt-zu-Punkt- und Mehrpunkt-basierende LAN-Integrationsstrategien für den digitalen Mobilfunkstandard DECT

97-07 *Manfred Reichert, Peter Dadam*
$ADEPT_{flex}$ - Supporting Dynamic Changes of Workflows Without Loosing Control

97-08 *Hans Braxmeier, Dietmar Ernst, Andrea Mößle, Heiko Vogler*
The Project NoName - A functional programming language with its development environment

97-09 *Christian Heinlein*
Grundlagen von Interaktionsausdrücken

97-10 *Christian Heinlein*
Graphische Repräsentation von Interaktionsausdrücken

97-11 *Christian Heinlein*
Sprachtheoretische Semantik von Interaktionsausdrücken

97-12 *Gerhard Schellhorn, Wolfgang Reif*
Proving Properties of Finite Enumerations: A Problem Set for Automated Theorem Provers

97-13 *Dietmar Ernst, Frank Houdek, Wolfram Schulte, Thilo Schwinn*
Experimenteller Vergleich statischer und dynamischer Softwareprüfung für eingebettete Systeme

97-14 *Wolfgang Reif, Gerhard Schellhorn*
Theorem Proving in Large Theories

97-15 *Thomas Wennekers*
Asymptotik rekurrenter neuronaler Netze mit zufälligen Kopplungen

97-16 *Peter Dadam, Klaus Kuhn, Manfred Reichert*
Clinical Workflows - The Killer Application for Process-oriented Information Systems?

97-17 *Mohammad Ali Livani, Jörg Kaiser*
EDF Consensus on CAN Bus Access in Dynamic Real-Time Applications

97-18 *Johannes Köbler,Rainer Schuler*
Using Efficient Average-Case Algorithms to Collapse Worst-Case Complexity Classes

98-01 *Daniela Damm, Lutz Claes, Friedrich W. von Henke, Alexander Seitz, Adelinde Uhrmacher, Steffen Wolf*
Ein fallbasiertes System für die Interpretation von Literatur zur Knochenheilung

98-02 *Thomas Bauer, Peter Dadam*
Architekturen für skalierbare Workflow-Management-Systeme - Klassifikation und Analyse

98-03 *Marko Luther, Martin Strecker*
A guided tour through *Typelab*

98-04 *Heiko Neumann, Luiz Pessoa*
Visual Filling-in and Surface Property Reconstruction

98-05 *Ercüment Canver*
Formal Verification of a Coordinated Atomic Action Based Design

98-06 *Andreas Küchler*
On the Correspondence between Neural Folding Architectures and Tree Automata

98-07 *Heiko Neumann, Thorsten Hansen, Luiz Pessoa*
Interaction of ON and OFF Pathways for Visual Contrast Measurement

98-08 *Thomas Wennekers*
Synfire Graphs: From Spike Patterns to Automata of Spiking Neurons

98-09 *Thomas Bauer, Peter Dadam*
Variable Migration von Workflows in *ADEPT*

98-10 *Heiko Neumann, Wolfgang Sepp*
Recurrent V1 – V2 Interaction in Early Visual Boundary Processing

98-11 *Frank Houdek, Dietmar Ernst, Thilo Schwinn*
Prüfen von C–Code und Statmate/Matlab–Spezifikationen: Ein Experiment

98-12 *Gerhard Schellhorn*
Proving Properties of Directed Graphs: A Problem Set for Automated Theorem Provers

98-13 *Gerhard Schellhorn, Wolfgang Reif*
Theorems from Compiler Verification: A Problem Set for Automated Theorem Provers

98-14 *Mohammad Ali Livani*
SHARE: A Transparent Mechanism for Reliable Broadcast Delivery in CAN

98-15 *Mohammad Ali Livani, Jörg Kaiser*
Predictable Atomic Multicast in the Controller Area Network (CAN)

99-01 *Susanne Boll, Wolfgang Klas, Utz Westermann*
A Comparison of Multimedia Document Models Concerning Advanced Requirements

99-02 *Thomas Bauer, Peter Dadam*
Verteilungsmodelle für Workflow-Management-Systeme - Klassifikation und Simulation

99-03 *Uwe Schöning*
On the Complexity of Constraint Satisfaction

99-04 *Ercument Canver*
Model-Checking zur Analyse von Message Sequence Charts über Statecharts

99-05 *Johannes Köbler, Wolfgang Lindner, Rainer Schuler*
Derandomizing RP if Boolean Circuits are not Learnable

99-06 *Utz Westermann, Wolfgang Klas*
Architecture of a DataBlade Module for the Integrated Management of Multimedia Assets

99-07 *Peter Dadam, Manfred Reichert*
Enterprise-wide and Cross-enterprise Workflow Management: Concepts, Systems, Applications. Paderborn, Germany, October 6, 1999, GI–Workshop Proceedings, Informatik ’99

99-08 *Vikraman Arvind, Johannes Köbler*
Graph Isomorphism is Low for ZPP^{NP} and other Lowness results

99-09 *Thomas Bauer, Peter Dadam*
Efficient Distributed Workflow Management Based on Variable Server Assignments

2000-02 *Thomas Bauer, Peter Dadam*
Variable Serverzuordnungen und komplexe Bearbeiterzuordnungen im Workflow-Management-System ADEPT

2000-03 *Gregory Baratoff, Christian Toepfer, Heiko Neumann*
Combined space-variant maps for optical flow based navigation

2000-04 *Wolfgang Gehring*
Ein Rahmenwerk zur Einführung von Leistungspunktsystemen

2000-05 *Susanne Boll, Christian Heinlein, Wolfgang Klas, Jochen Wandel*
Intelligent Prefetching and Buffering for Interactive Streaming of MPEG Videos

2000-06 *Wolfgang Reif, Gerhard Schellhorn, Andreas Thums*
Fehlersuche in Formalen Spezifikationen

2000-07 *Gerhard Schellhorn, Wolfgang Reif (eds.)*
FM-Tools 2000: The 4th Workshop on Tools for System Design and Verification

2000-08 *Thomas Bauer, Manfred Reichert, Peter Dadam*
Effiziente Durchführung von Prozessmigrationen in verteilten Workflow-Management-Systemen

2000-09 *Thomas Bauer, Peter Dadam*
Vermeidung von Überlastsituationen durch Replikation von Workflow-Servern in ADEPT

2000-10 *Thomas Bauer, Manfred Reichert, Peter Dadam*
Adaptives und verteiltes Workflow-Management

2000-11 *Christian Heinlein*
Workflow and Process Synchronization with Interaction Expressions and Graphs

2001-01 *Hubert Hug, Rainer Schuler*
DNA-based parallel computation of simple arithmetic

2001-02 *Friedhelm Schwenker, Hans A. Kestler, Günther Palm*
3-D Visual Object Classification with Hierarchical Radial Basis Function Networks

2001-03 *Hans A. Kestler, Friedhelm Schwenker, Günther Palm*
RBF network classification of ECGs as a potential marker for sudden cardiac death

2001-04 *Christian Dietrich, Friedhelm Schwenker, Klaus Riede, Günther Palm*
Classification of Bioacoustic Time Series Utilizing Pulse Detection, Time and Frequency Features and Data Fusion

2002-01 *Stefanie Rinderle, Manfred Reichert, Peter Dadam*
Effiziente Verträglichkeitsprüfung und automatische Migration von Workflow-Instanzen bei der Evolution von Workflow-Schemata

2002-02 *Walter Guttmann*
Deriving an Applicative Heapsort Algorithm

2002-03 *Axel Dold, Friedrich W. von Henke, Vincent Vialard, Wolfgang Goerigk*
A Mechanically Verified Compiling Specification for a Realistic Compiler

2003-01 *Manfred Reichert, Stefanie Rinderle, Peter Dadam*
A Formal Framework for Workflow Type and Instance Changes Under Correctness Checks

2003-02 *Stefanie Rinderle, Manfred Reichert, Peter Dadam*
Supporting Workflow Schema Evolution By Efficient Compliance Checks

2003-03 *Christian Heinlein*
Safely Extending Procedure Types to Allow Nested Procedures as Values

2003-04 *Stefanie Rinderle, Manfred Reichert, Peter Dadam*
On Dealing With Semantically Conflicting Business Process Changes.

2003-05 *Christian Heinlein*
Dynamic Class Methods in Java

2003-06 *Christian Heinlein*
Vertical, Horizontal, and Behavioural Extensibility of Software Systems

2003-07 *Christian Heinlein*
Safely Extending Procedure Types to Allow Nested Procedures as Values
(Corrected Version)

2003-08 *Changling Liu, Jörg Kaiser*
Survey of Mobile Ad Hoc Network Routing Protocols)

2004-01 *Thom Frühwirth, Marc Meister (eds.)*
First Workshop on Constraint Handling Rules

2004-02 *Christian Heinlein*
Concept and Implementation of C+++, an Extension of C++ to Support User-Defined Operator Symbols and Control Structures

2004-03 *Susanne Biundo, Thom Frühwirth, Günther Palm(eds.)*
Poster Proceedings of the 27th Annual German Conference on Artificial Intelligence

2005-01 *Armin Wolf, Thom Frühwirth, Marc Meister (eds.)*
19th Workshop on (Constraint) Logic Programming

2005-02 *Wolfgang Lindner (Hg.), Universität Ulm , Christopher Wolf (Hg.) KU Leuven*
2. Krypto-Tag – Workshop über Kryptographie, Universität Ulm

2005-03 *Walter Guttmann, Markus Maucher*
Constrained Ordering

2006-01 *Stefan Sarstedt*
Model-Driven Development with ACTIVECHARTS, Tutorial

2006-02 *Alexander Raschke, Ramin Tavakoli Kolagari*
Ein experimenteller Vergleich zwischen einer plan-getriebenen und einer leichtgewichtigen Entwicklungsmethode zur Spezifikation von eingebetteten Systemen

2006-03 *Jens Kohlmeyer, Alexander Raschke, Ramin Tavakoli Kolagari*
Eine qualitative Untersuchung zur Produktlinien-Integration über Organisationsgrenzen hinweg

2006-04 *Thorsten Liebig*
Reasoning with OWL - System Support and Insights –

2008-01 *H.A. Kestler, J. Messner, A. Müller, R. Schuler*
On the complexity of intersecting multiple circles for graphical display

2008-02 *Manfred Reichert, Peter Dadam, Martin Jurisch,l Ulrich Kreher, Kevin Göser, Markus Lauer*
Architectural Design of Flexible Process Management Technology

2008-03 *Frank Raiser*
Semi-Automatic Generation of CHR Solvers from Global Constraint Automata

2008-04 *Ramin Tavakoli Kolagari, Alexander Raschke, Matthias Schneiderhan, Ian Alexander*
Entscheidungsdokumentation bei der Entwicklung innovativer Systeme für produktlinien-basierte Entwicklungsprozesse

2008-05 *Markus Kalb, Claudia Dittrich, Peter Dadam*
Support of Relationships Among Moving Objects on Networks

2008-06 *Matthias Frank, Frank Kargl, Burkhard Stiller (Hg.)*
WMAN 2008 – KuVS Fachgespräch über Mobile Ad-hoc Netzwerke

2008-07 *M. Maucher, U. Schöning, H.A. Kestler*
An empirical assessment of local and population based search methods with different degrees of pseudorandomness

2008-08 *Henning Wunderlich*
Covers have structure

2008-09 *Karl-Heinz Niggl, Henning Wunderlich*
Implicit characterization of FPTIME and NC revisited

2008-10 *Henning Wunderlich*
On span-P^{cc} and related classes in structural communication complexity

2008-11 *M. Maucher, U. Schöning, H.A. Kestler*
On the different notions of pseudorandomness

2008-12 *Henning Wunderlich*
On Toda's Theorem in structural communication complexity

2008-13 *Manfred Reichert, Peter Dadam*
Realizing Adaptive Process-aware Information Systems with ADEPT2

2009-01 *Peter Dadam, Manfred Reichert*
The ADEPT Project: A Decade of Research and Development for Robust and Fexible Process Support
Challenges and Achievements

2009-02 *Peter Dadam, Manfred Reichert, Stefanie Rinderle-Ma, Kevin Göser, Ulrich Kreher, Martin Jurisch*
Von ADEPT zur AristaFlow® BPM Suite – Eine Vision wird Realität "Correctness by Construction" und flexible, robuste Ausführung von Unternehmensprozessen

2009-03 *Alena Hallerbach, Thomas Bauer, Manfred Reichert*
Correct Configuration of Process Variants in Provop

2009-04 *Martin Bader*
On Reversal and Transposition Medians

2009-05 *Barbara Weber, Andreas Lanz, Manfred Reichert*
Time Patterns for Process-aware Information Systems: A Pattern-based Analysis

2009-06 *Stefanie Rinderle-Ma, Manfred Reichert*
Adjustment Strategies for Non-Compliant Process Instances

2009-07 *H.A. Kestler, B. Lausen, H. Binder H.-P. Klenk. F. Leisch, M. Schmid*
Statistical Computing 2009 – Abstracts der 41. Arbeitstagung

2009-08 *Ulrich Kreher, Manfred Reichert, Stefanie Rinderle-Ma, Peter Dadam*
Effiziente Repräsentation von Vorlagen- und Instanzdaten in Prozess-Management-Systemen

2009-09 *Dammertz, Holger, Alexander Keller, Hendrik P.A. Lensch*
Progressive Point-Light-Based Global Illumination

2009-10 *Dao Zhou, Christoph Müssel, Ludwig Lausser, Martin Hopfensitz, Michael Kühl, Hans A. Kestler*
Boolean networks for modeling and analysis of gene regulation

2009-11 *J. Hanika, H.P.A. Lensch, A. Keller*
Two-Level Ray Tracing with Recordering for Highly Complex Scenes

2009-12 *Stephan Buchwald, Thomas Bauer, Manfred Reichert*
Durchgängige Modellierung von Geschäftsprozessen durch Einführung eines Abbildungsmodells: Ansätze, Konzepte, Notationen

2010-01 *Hariolf Betz, Frank Raiser, Thom Frühwirth*
A Complete and Terminating Execution Model for Constraint Handling Rules

2010-02 *Ulrich Kreher, Manfred Reichert*
Speichereffiziente Repräsentation instanzspezifischer Änderungen in Prozess-Management-Systemen

2010-03 *Patrick Frey*
Case Study: Engine Control Application

2010-04 *Matthias Lohrmann und Manfred Reichert*
Basic Considerations on Business Process Quality

2010-05 *HA Kestler, H Binder, B Lausen, H-P Klenk, M Schmid, F Leisch (eds):*
Statistical Computing 2010 - Abstracts der 42. Arbeitstagung

2010-06 *Vera Künzle, Barbara Weber, Manfred Reichert*
Object-aware Business Processes: Properties, Requirements, Existing Approaches

2011-01 *Stephan Buchwald, Thomas Bauer, Manfred Reichert*
Flexibilisierung Service-orientierter Architekturen

2011-02 *Johannes Hanika, Holger Dammertz, Hendrik Lensch*
Edge-Optimized À-Trous Wavelets for Local Contrast Enhancement with Robust Denoising

2011-03 *Stefanie Kaiser, Manfred Reichert*
Datenflussvarianten in Prozessmodellen: Szenarien, Herausforderungen, Ansätze

2011-04 *Hans A. Kestler, Harald Binder, Matthias Schmid, Friedrich Leisch, Johann M. Kraus (eds):*
Statistical Computing 2011 - Abstracts der 43. Arbeitstagung

2011-05 *Vera Künzle, Manfred Reichert*
PHILharmonicFlows: Research and Design Methodology

2011-06 *David Knuplesch, Manfred Reichert*
Ensuring Business Process Compliance Along the Process Life Cycle

2011-07 *Marcel Dausend*
Towards a UML Profile on Formal Semantics for Modeling Multimodal Interactive Systems

2011-08 *Dominik Gessenharter*
Model-Driven Software Development with ACTIVECHARTS - A Case Study

2012-01 *Andreas Steigmiller, Thorsten Liebig, Birte Glimm*
Extended Caching, Backjumping and Merging for Expressive Description Logics

2012-02 *Hans A. Kestler, Harald Binder, Matthias Schmid, Johann M. Kraus (eds):*
Statistical Computing 2012 - Abstracts der 44. Arbeitstagung

2012-03 *Felix Schüssel, Frank Honold, Michael Weber*
Influencing Factors on Multimodal Interaction at Selection Tasks

2012-04 *Jens Kolb, Paul Hübner, Manfred Reichert*
Model-Driven User Interface Generation and Adaption in Process-Aware Information Systems

2012-05 *Matthias Lohrmann, Manfred Reichert*
Formalizing Concepts for Efficacy-aware Business Process Modeling

2012-06 *David Knuplesch, Rüdiger Pryss, Manfred Reichert*
A Formal Framework for Data-Aware Process Interaction Models

2012-07 *Clara Ayora, Victoria Torres, Barbara Weber, Manfred Reichert, Vicente Pelechano*
Dealing with Variability in Process-Aware Information Systems: Language Requirements, Features, and Existing Proposals

2013-01 *Frank Kargl*
Abstract Proceedings of the 7th Workshop on Wireless and Mobile Ad-Hoc Networks (WMAN 2013)

2013-02 *Andreas Lanz, Manfred Reichert, Barbara Weber*
A Formal Semantics of Time Patterns for Process-aware Information Systems

2013-03 *Matthias Lohrmann, Manfred Reichert*
Demonstrating the Effectiveness of Process Improvement Patterns with Mining Results

2013-04 *Semra Catalkaya, David Knuplesch, Manfred Reichert*
Bringing More Semantics to XOR-Split Gateways in Business Process Models Based on Decision Rules

2013-05 *David Knuplesch, Manfred Reichert, Linh Thao Ly, Akhil Kumar, Stefanie Rinderle-Ma*
On the Formal Semantics of the Extended Compliance Rule Graph

2013-06 *Andreas Steigmiller, Birte Glimm*
Nominal Schema Absorption

2013-07 *Hans A. Kestler, Matthias Schmid, Florian Schmid, Dr. Markus Maucher, Johann M. Kraus (eds)*
Statistical Computing 2013 - Abstracts der 45. Arbeitstagung

2013-08 *Daniel Ott, Dr. Alexander Raschke*
Evaluating Benefits of Requirement Categorization in Natural Language Specifications for Review Improvements

2013-09 *Philip Geiger, Rüdiger Pryss, Marc Schickler, Manfred Reichert*
Engineering an Advanced Location-Based Augmented Reality Engine for Smart Mobile Devices

Ulmer Informatik-Berichte
ISSN 0939-5091

Herausgeber:
Universität Ulm
Fakultät für Ingenieurwissenschaften und Informatik
89069 Ulm