Software-Agents for On-Demand Authoring of Mobile Augmented Reality Applications

Rafael Radkowski

Heinz Nixdorf Institute, University of Paderborn
Fürstenallee 11
33102 Paderborn, Germany
Rafael.Radkowski@hni.uni-paderborn.de

Abstract. The paper presents an concept for the automatic authoring of augmented reality (AR) applications. The approach is based on software agents that provide different functions and content on demand for an AR application. Autonomous software agents encapsulate the specific functions of an AR application. It is distinguished between two kinds of software agents: So called provider-agents and user-agents. The user agent is configured by a human user, the provider-agent provides the functionality of an AR application. By communication and cooperation, provider and user agents form an AR application. The AR-based concept has been tested with the agent platform JADE.

1 Introduction

Augmented Reality (AR) is a human-computer-interface, which superimposes the perception of reality with computer-generated information [1]. The information is shown in the right context and with relation to a real world object. This information can be 3D models, text or annotations. To see them, special viewing devices are necessary. A classic viewing device is the head mounted display (HMD), a google-like thing that uses small displays instead of glasses. The user sees the reality as a video stream inside this displays, the computer-generated information superimpose the video stream.

Normally, most of the known AR applications are stand alone applications. These applications have a fixed set of functions as well as a fixed hardware setup at one certain location. These functions and the hardware setup form the configuration of the AR application. Figure 1 shows a schematic overview of a typical configuration of software components of an AR application. There are components for tracking, for interaction and for the content of the applications. To facilitate an AR application, the components exchange information. A tracking system tracks the position and orientation of the user and of real objects, surrounding the user. Content components for AR provide 2D visualizations, 3D models, annotations, and text information, etc. The content represents the information the AR application can present to the user. Interaction devices and the related software components provide the information from the hardware devices. All this information is processed in an application logic component, which manage the application and controls the renderer.

R. Shumaker (Ed.): Virtual and Mixed Reality, LNCS 5622, pp. 225–234, 2009.

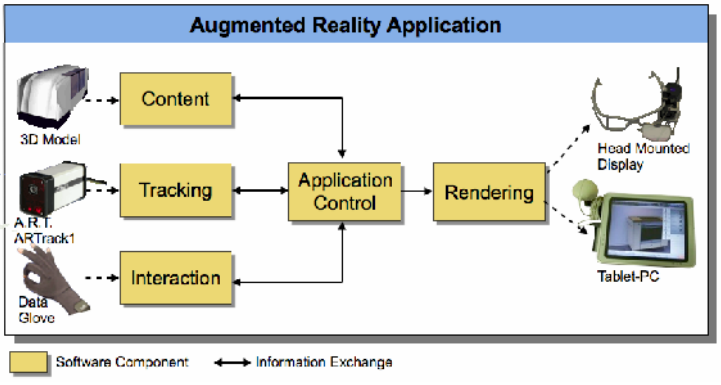

Fig. 1. Common component configuration of an AR application

Because AR applications present information about a localized real world object, a stand-alone application is suitable in many cases. But today, the mobile hardware devices become more and more smaller. In consequence, mobile, location based applications become important. If an AR application should facilitate information about more than on object, or should provide functions for more than one purpose, the configuration has to be changed.

There are two ways to change the configuration of an AR application: a programmer can reprogram the application or an AR-author uses an authoring tool. For a user of an AR application, e.g. a mechanics, an engineer or a medicine, these two ways are not adequate.

To get a dynamic change of the content, the tracking system, etc., different programming techniques exist. Software agents are one solution to achieve this. Software agents are autonomous computer programs that work according to the requirements of a user [2]. One feature of software agents is ability to communicate and cooperate with other software agents. In this paper, an agent-based AR software is described. First, the related work is discussed, and then the concept of the agent-based AR application is presented. Afterwards a software prototype of this concept is described and the first results are presented. At last, the results are summarized and an outlook is given.

2 Related Work

There are two research fields, which are related to the work, presented in this paper. These are software frameworks for AR applications and authoring tools. Both, software frameworks and authoring tools are used change the configuration of an AR application.

Software frameworks and APIs are the common way, because software developers create most of the developed and presented AR applications. They are the most flexible way to integrate new functions into an application, because new functionality is integrated by programming. This is flexible but very complex. At this place, only a

few of them can be presented. One famous framework is Studierstube [3]. It is a framework for the development of mobile, collaborative and ubiquitous AR applications. The standard API is realized as a set of C++ classes built on top of the Open Inventor (OIV) graphics toolkit. It supports camera calibration and uses vision based pattern tracking. Because it is a programming interface it can be extended easily by adding new components. Another software for programming mobile AR applications is Tinmith [4]. Tinmith is a software architecture written in C++. It supports 2D and 3D rendering, tracker abstractions, object extensions etc.. The architecture has been designed and implemented for mobile outdoor augmented reality application. Interaction devices like data gloves can be easily integrated. To extend these two frameworks, it is necessary to program new functions or to integrate other software components by programming, too.

A framework, which can easily be extended by third-party components is DWARF [5]. DWARF is a CORBA based framework that allows the rapid prototyping of distributed AR applications. The framework is based on the concept of collaborating distributed services. The services are interdependent and expose their requirements with the help of service managers. It can be easily extended by integrated new services to the framework. With CORBA the development language can be chosen free from case to case. This framework simplifies the integration of new functions into an AR application and to design or setup a new application. But it doesn't work automatically.

Beside software frameworks and APIs, authoring tools are the second way to program a new application. But authoring tools have a fixed and limited set of functions, which can be configured by the user. If new functions are needed, they have to be programmed by the provider of the authoring tool or a computer scientist. For instance, authoring tools are DART (http://www.cc.gatech.edu/projects/dart/), AMIRE (http://www.amire.net/), and ARBlender (http://www.ai.fh-erfurt.de/arblender).

A work, which has the same aim is presented in [6], where a concept for ubiquitous tracking has been tested. Aim is to get data from widespread and diverse heterogeneous tracking sensors and automatically and dynamically fuse them, and then transparently provide them to applications. For tracking systems, the approach is close to the agent-based approach. But the agent-based approach extends it to content, interaction devices and the entire set of components necessary for an AR application.

3 Agent-Based AR Applications

Software agents have two features that facilitate a dynamic configuration of AR applications. It is their ability to communicate and to cooperate. If software agents encapsulate the functionality for an AR application, they are able to form an application only by communication. For that purpose, a proposal for an agent-based configuration is made and how the on-demand functionality can be achieved. Furthermore, a communication architecture is presented, that facilitates the dynamic connection of different components over TCP/IP or UDP/IP.

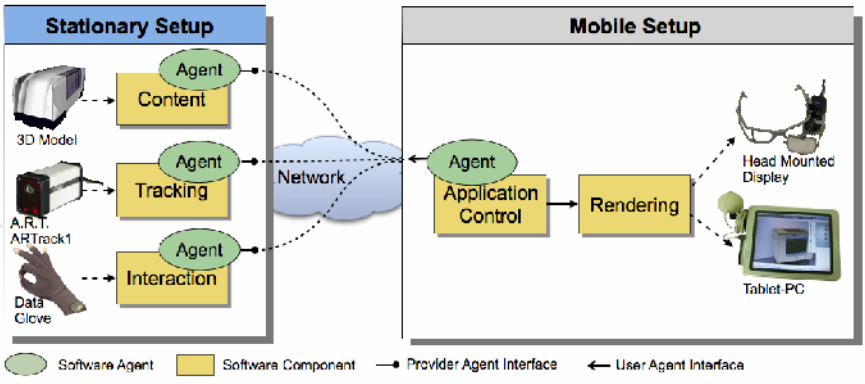

Fig. 2. Schematic overview of the agent-based approach for AR applications

3.1 Agent-Based Configuration

Figure 2 shows a schematic overview of the proposed agent-based configuration. Basic idea is, that software agents encapsulate the software components, necessary for an AR application. There need to be one agent for each component, which have to exchange data to the application control. Each agent knows the functionality of its related software component. Furthermore it knows the application context, in which the functions can be used.

The components of the AR application are separated into two classes: In components for a stationary setup and into components for a mobile setup.

Components of the stationary setup are localized. It is only reasonable to use them with relation to an appropriate place and an appropriate real world object. For instance a tracking system is localized. A hardware tracking system is installed and calibrated for one room. 3D models or text information should that an AR application shows should provide information about a real object. They are only reasonable at an appropriate location, too.

Components of the mobile setup are mobile. These are the application control and the rendering engine. The application control manages the information flow inside the application. For instance it shows or hide 3D models, annotate different real world objects, if they should be visible for the user. This managing component is only once needed. It must provide some basic functions (show/hide models, move models, change color, etc.), which can be selected by the content, tracking and interaction components. A rendering component is reasonable only in relation to a certain hardware device. Furthermore it don't need an agent, because their can be a point-to-point communication to the application control.

An AR application is realized by communication only. The task of the software agents is to found each other, to start an application and to initialize a communication process between the components. Initiate a communication process means, they agree about the values, the semantic of those values as well as about the communication protocol.

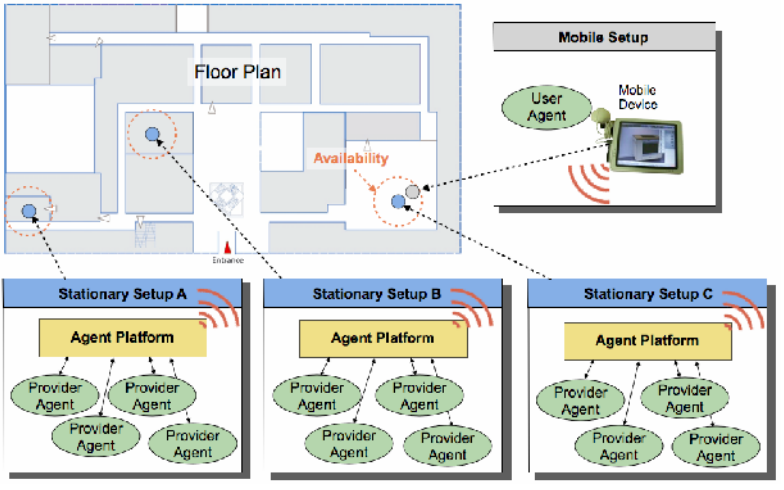

Fig. 3. Different stationary setups provide location-based functions for AR applications. If a mobile device comes close to a stationary setup, the user agent initiates the application.

3.2 On-Demand Authoring

On-demand in the context of AR applications means that a user can use a local referred application, if he is a) at the appropriate place and b) if he liked it. Authoring means that the content and functionality of an application and the necessary devices are changed in order to fulfill new requirements. This requires an infrastructure, that broadcast information about attainable applications and which informs the user, when a suitable application is attainable. To achieve this, two kinds of agents are proposed: the so called *provider agents* and *user agents*.

Provider agents provide information about the functionality of the location-based components (Fig. 3). There are two types of information: General information and communication information. General information provides data about the functionality of the agent and the application context. Communication information provides information about the data itself and about the binding procedure. In order to decide, which application or component is suitable for a user the general information is important.

The developer of the provider agent has to specify this information. Following information is proposed:

- ID: Each agent gets a unique id.
- Name: The name of the agent. The name can be chosen by the programmer of the agent
- Description: This is an annotation that should provide some further information about the agent.
- Keyword: Keywords, those provide a summary of the functions.
- Service Type: Specify the provider agent as provider of tracking functionality, interaction functionality or content.
- Server: Specify the url of the server, where the agent platform is located.

- Connection: Provide some detailed information about the binding, the connection procedure and the query requests.
- Application Class: Tell the application class of the agent. One provider agent should only have one application class. Several application classes like, art, fun, information, service, etc. are specified.
- Reference: References to other provider agents that functions are necessary for that agent in order to provide an entire application.

A user agent represents a human user in the agent community. It fulfills three tasks. First it searches for provider agents and AR applications. If applications are found, it verifies the offered functions with the demands of the user and proposes them to the user. Third, when the user decides to use the application, the agent initiates the AR application and configures the necessary communication process.

To localize a provider agent, a directory service of the agent platform is used. This directory service broadcasts a list of all available agents and their referred service types and keywords. Furthermore, it provides information about how the provider agent is to connect. To verify the application context, the user agent uses a keyword matching. Altogether, the user has to specify four type of information:

- Application Class: These keywords are similar to the applications classes of the provider agent. A user can specify as much keywords, as he wish to use.
- Keywords: These keywords should provide some detailed information about the applications, e.g. workshop, drilling machine, museum, etc. The keywords itself and their number can be chosen by the user.
- Human Factors: The most important human factors are, time, costs and location. By setting a time limit, the user tells the agent, how much time he want to spend for an application. Costs describe the prices that an user have to pay for an application. Location information help to find the application, It gives some detailed information about the location, e.g. a building or a room number.
- Hardware: Describes the hardware of the user. Most important values are the processing power, the available graphic capability, the memory and integrated interaction devices.

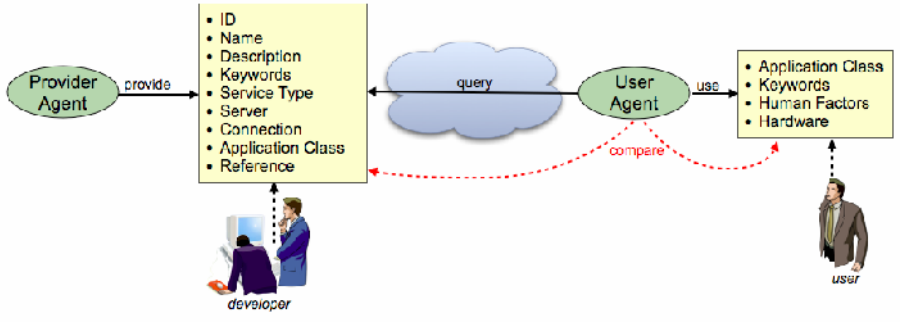

Fig. 4. Provider agent and user agent keep a set of data, the user agent compares this data to identify adequate applications

This keyword matching compares the application class of the provider agent and the keywords with keywords, entered by the user (Fig. 4).

3.3 Communication and Integration Architecture

If the user agent founds provider agents its user want to use, it has to setup an AR application. For that purpose, a communication server is used (Fig. 5). Main task of the communication server is to coordinate the data exchange between different software components. Software components can be tracking systems, software for interaction devices and the content. Because the AR applications should be realized by communication, the communication server is used as integration platform.

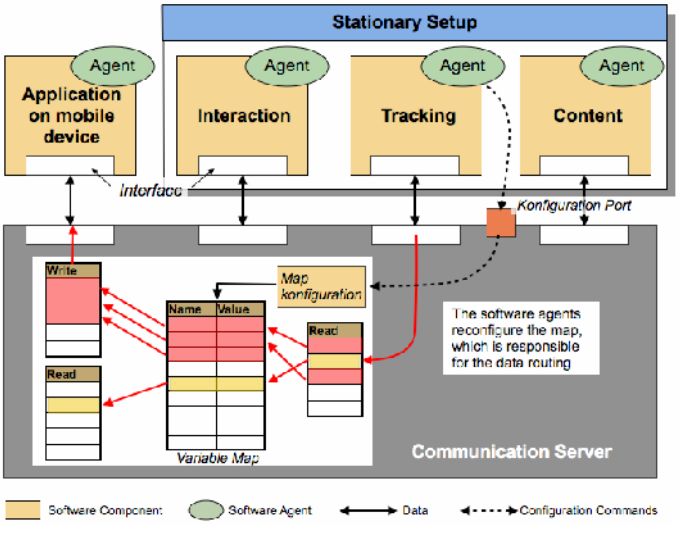

Fig. 5. A communication server is use to exchange the data between the components. The software agents send command to the server in order to configure the data routing.

To use the software agents itself for communication is not necessary. Main reason for that is the static communication between the components. Once, the communication is initialized, the attributes are fixed and will not be changed. An agents should be used only, if the values that are exchanged are not fixed. Furthermore, the time synchronize communication is better controllable over a central server.

Main part of the communication server is a variable map. Every variable that is exchange by the server is stored inside this map. The variables are stored by a unique name. This name is formed by the name of the component and a unique name for each variable

name_of_component.name_of_variable

Each component, which provides a variable, it is stored in the variable map. If a component needs variables, the communication server will send them to the component. Task of the software agents is to configure the read and write maps inside the

communication server. If an user agent initialize an AR application, every provider agent declare the variables of the referred component to the server and start the application. After all variables are declared, the component starts to submit the data to the server. The user agent start to declare the date, its component needs to read.

4 Prototypical Realization

Aim of this work was to verify the agent-based architecture for AR applications and the keyword search for the on-demand configuration. For that purpose, a software prototype has been implemented. For that software prototype, an agent platform JADE (Java Agent DEvelopment Framework) has been used [7]. It simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications and through a set of tools that supports the debugging and deployment phases.

The test the agent-based architecture, two AR applications has been connected to software agents (Fig. 6).

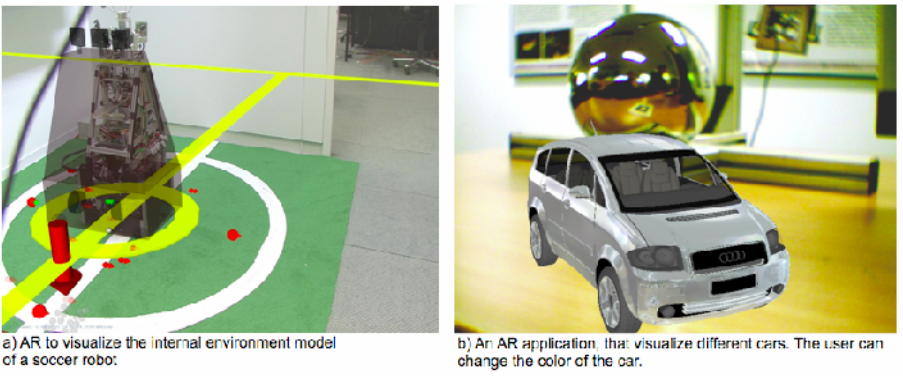

Fig. 6. To test the agent-based architecture, two AR applications have been setup by software agents

The first AR application visualizes the internal environment model of autonomous soccer robots [8]. The second application shows different car models inside the real environment. To simulate different locations, each application has been installed in a different room. The 3D models, which present the environment to the user, and the car model have been encapsulated to software components. Moreover, two different tracking devices have been separated into two software components, which have been connected to a software agent. The soccer robots are tracked by an A.R.T. infrared tracking system [9], the car models are shown on ARToolkit pattern [10]. For the test, these four software components have been executed as stand alone applications, they communicate with the software agent via a TCP/IP socket. This was necessary to keep the existing applications and to allow software agents, written in JAVA to communicate with software, written in C++.

The AR rendering component has been programmed with OpenSceneGraph [11], an open source scenegraph programming library, written in C++. It runs as stand alone application and, for the tests, it communicates to a referred user-agent by a TCP/IP socket to. As hardware device a tablet PC (Compaq) has been used.

4.1 Results and Discussion

The application has been tested by a set of user. This test should show, if the user-agents are able to find the provider agents and if an AR application can be initialized by communication. The test show, that the software agents are able to configure the communication server and to initiate an AR application. Furthermore, they show that it is possible to separate a AR application in different independent running parts, which are integrated to an application, by communication and cooperation only. Existing agent frameworks like JADE allow a flexible cooperation between software agents. Further, they provide a service description and necessary functions to discover software agents on an agent platform.

One advantage of an agent-based concept is, that is separates the provider of the AR applications from the hardware of the user. Today, a provider of an AR application has to provide everything: the hardware, the rendering software, the tracking system, interaction devices and the content. There are no possibilities to separate these parts to different providers. For instance, one person/company could provide a tracking system on different places, and a provider of content, e.g. a museum, provide information to its paintings, based on the available tracking system. A visitor of this museum can use its on smart phone or tablet PC to use this AR application. Because people can user their own hardware theoretically, the provider of an AR application doesn‘t have to hold the hardware ready.

The big advantage is, that the agent-based approach facilitates a flexible framework for AR applications. Different agents form an AR application by communication only. If one component of the application should be exchanged, it is only necessary to change the software agent, encapsulate the needed functions. Of course, the software agents have to be prepared.

To achieve this flexibility, the communication protocol and agent service description must be standardizing. The general information the provider agent and user agent have been used is self-made. For the small controlled test, the keyword matching and the application classes work in order to found an application. But for a real world scenario, the information has to be standardized.

The agent-based AR applications aim to AR application in a workshop environment. Where a mechanics or an engineer use AR to analyze the behavior of a test bench or a mechanic maintenance a machine. This people use one hardware device only, but the need to use different content and different tracking systems. In such a not closed but limited environment, it is possible to have influence to the hardware, the communication protocol, and the service description. If these parts are under control, the agent-based AR application should facilitate flexible AR applications and an easy reconfiguration of the application without any assistance by an engineer of mechanics user.

5 Outlook

It is planed to use the agent-based architecture in an augmented reality-based experimental try-out environment for mechatronic systems. This environment is part of the collaborative research center 614 “Self-optimizing concepts and structure in mechanical engineering”. Engineers use AR to analyze and test intelligent mechatronic systems. For that purpose, different extensions are planed.

First, the prototypical implementation must be extended. To test the architecture, a prototypical, not user-friendly software has been developed. This software serves for testing purposes only. For instance, the TCP based communication between software components of the AR application and the JADE agents is not sufficient. Therefore a wrapper API is planed.

Next, the information, which is exchange between the software agents and the keyword matching, should be extended. It is planed to use more than one model to describe the functionality of a software component. One model should provide information about the function, one model should describe the possible task, where the component can be reasonable used, and one model should describe the data.

In order to localize provider agent, a JADE agent platform needs to now the addresses of other reachable agent platforms. This way, a practical working service discovery cannot be realized. To improve this, the communication protocol, which the agents are used will be changed, of the agent platform will be changed.

References

1. Azuma, R.: A Survey of Augmented Reality. In: Presence: Teleoperators and Virtual Environments 6 (1997)
2. Jennings, N.R.: An agent-based approach for building complex software systems. Communications of the ACM 44(4), 7 (2001)
3. Schall, G., Newman, J., Schmalstieg, D.: Rapid and Accurate Deployment of Fiducial Markers for Augmented Reality. In: Proceed. of the 10th Comp. Vistion Winter Workshop, Zell an der Pram, Upper Austria (2005)
4. Piekarski, W., Smith, R., Thomas, B.H.: Designing Backpacks for High Fidelity Mobile Outdoor Augmented Reality. In: 3rd Int Symposium on Mixed and Augmented Reality, Arlington, Va (2004)
5. Bauer, M., Bruegge, B., Klinker, G., MacWilliams, A., Reicher, T., Riß, S., Sandor, C., Wagner, M.: Design of a Component-Based Augmented Reality Framework. In: Proceedings of The Second IEEE and ACM International Symposium on Augmented Reality (2001)
6. Newman, J., Wagner, M., Bauer, M., MacWilliams, A., Pintaric, T., Beyer, D., Pustka, D., Strasser, F., Schmalstieg, D., Klinker, G.: Ubiquitous Tracking for Augmented Reality. In: International symposium on mixed and augmented reality, Arlington, USA (2004)
7. Webpage of JADE, A open source plattform for peer to peer agent based applications (2009), `http://jade.tilab.com/`
8. Richert, W., Kleinjohann, B., Koch, M., Bruder, A., Rose, S., Adelt, P.: The paderkicker team: Autonomy in realtime environments. In: Proceedings of the Working Conference on Distributed and Parallel Embedded Systems, DIPES 2006 (2006)
9. Homepage of Advanced Realtime Tracing, `http://www.ar-tracking.de/`
10. Kato, H., Billinghurst, M.: Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System. In: Proceedings of the 2nd International Workshop on Augmented Reality, IWAR 1999 (1999)
11. Webpage of OpenSceneGraph, `http://www.openscenegraph.org`