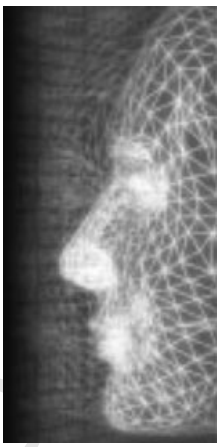
COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2008; **18**: 1–13
Published online in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/cav.220

Augmented reality agents for user interface adaptation



By István Barakonyi and Dieter Schmalstieg*

Most augmented reality (AR) applications are primarily concerned with letting a user browse a 3D virtual world registered with the real world. More advanced AR interfaces let the user interact with the mixed environment, but the virtual part is typically rather finite and deterministic. In contrast, autonomous behavior is often desirable in ubiquitous computing (Ubicomp), which requires the computers embedded into the environment to adapt to context and situation without explicit user intervention. We present an AR framework that is enhanced by typical Ubicomp features by dynamically and proactively exploiting previously unknown applications and hardware devices, and adapting the appearance of the user interface to persistently stored and accumulated user preferences. Our framework explores proactive computing, multi-user interface adaptation, and user interface migration. We employ mobile and autonomous agents embodied by real and virtual objects as an interface and interaction metaphor, where agent bodies are able to opportunistically migrate between multiple AR applications and computing platforms to best match the needs of the current application context. We present two pilot applications to illustrate design concepts. Copyright © 2008 John Wiley & Sons, Ltd.

Received: 4 February 2007; Revised: 2 December 2007; Accepted: 3 December 2007

KEY WORDS: augmented reality; ubiquitous computing; animated agents; autonomous agents; adaptive user interfaces

Introduction

Most augmented reality (AR) applications are primarily concerned with letting a user browse a 3D virtual world registered with the real world.[1] More advanced AR interfaces let the user interact with the mixed environment, but the virtual part is typically rather finite and deterministic. In contrast, autonomous behavior is often desirable in ubiquitous computing (Ubicomp), which requires the computers embedded into the environment to adapt to context and situation without explicit user intervention. To this end, Ubicomp often employs autonomous software agents that are capable of making their own decisions based on their perception of the environment. By sensing physical properties such as pose, velocity, temperature or light as input interaction channels, autonomous agents can react to changes in the environment in accordance with the users' perception.

Animated agents are a special case of autonomous agents with a visual, often anthropomorphic representation. Their strength is that they can appeal to multiple senses and engage a user in a natural conversation. Anthropomorphic agent representations can be particularly useful, where human assistance would normally be required. Previously, animated agents have been mostly used in virtual reality (VR) settings, where they are an integral part of a completely synthetic environment, and consequently all perception and action is virtual. By bringing animated agents into AR, we create two distinctively novel advantages:

- The animated agents can draw their perceptions from the state of both the virtual and the real world, granting them all the perceptual capabilities typically attributed to autonomous agents in Ubicomp. For example, a virtual path planning agent can avoid real world obstacles that it perceives.
- The behavior of the animated agents can affect both the virtual and the real world, extending the agents' scope of communication to include in particular physical objects and properties. For example, an agent

*Correspondence to: István Barakonyi, Graz University of Technology, Inffeldgasse 16, A-8010 Graz, Austria. E-mail: bara@icg.tu-graz.ac.at

may control wireless electronic equipment or home automation systems. Using such physical resources is common in Ubicomp, but not in AR.

Autonomous agents may have another important capability: mobility. The term "mobile agents" is commonly used to describe autonomous software components that have the ability to transfer and reproduce themselves on various networked computing devices. By equipping agents with mobile characteristics, they are no longer bound to a single, statically configured application, and output device, but may opportunistically migrate to and take advantage of other platforms that are more suitable to the agents' current needs. In particular, the agents can follow the user around in the physical world, and manifest themselves in the user's vicinity.

Such agents must be highly adaptive, for example, they must adapt their appearance to a form that suits the current situation and location. However, the desired autonomous behavior of mobile agents extends beyond appearance. The mobile agents must have a persistent knowledge base that is independent of their current location and execution status, in order to survive transportation from one environment to another and recover from transient system failures. Moreover, they have to speak with other components of a Ubicomp system using a network protocol for distributed object communication. Finally, the knowledge base must be extensible with descriptions of new, previously unknown software components encountered by the agents, so that the agents can learn to communicate with these components. With their adaptive capabilities, agents can remain autonomous in a changing, unpredictable environment, and vary their behavior on a per-user basis. We call this intended behavior *user interface adaptation*.

This paper summarizes the past 4 years of research on Ubiquitous Agents,[2,3] or *UbiAgents* for short. UbiAgents have been designed to be capable of user interface adaptation. They have the following noteworthy attributes:

- Animated appearance and information presentation in world registered AR displays;
- Self-determined migration from one AR display to another;
- Capable of sensing environmental state, in particular position and movement of physical artifacts;
- Bidirectional communication with unmodified, existing scene graph-based applications, relying on meta-information introduced via a mark-up mechanism;
- Persistent external knowledge base ensuring agent survival and customizable behavior.

UbiAgent is the first framework that brings animated agents with a high degree of autonomy into a mobile, dynamic environment, drawing from the rich set of AR displays, and interaction capabilities. While we do not claim that the presented agents are intelligent in a cognitive sense, UbiAgents push the boundary of what has been previously possible in terms of adaptive user interfaces.

Related Work

This work draws from a wide range of related work, so for reasons of brevity we limit the discussion to the most directly influential fields, which are AR user interfaces, interface agents, and mobile agents.

Adaptive Augmented Reality User Interfaces

The simplest form of automatic adaptation of AR content to current application context is information filtering based on a spatial or semantic world model. Bane and Höllerer[4] apply spatial filtering based on simple context elements such as distance and visibility to enhance building visualization in a mobile AR setup. The KARMA system[5] employs a rule-based illustration generation system to exploit user viewpoint, object pose, and communicative goals for efficient information visualization in an AR-based machine maintenance scenario. Julier *et al*. developed a hybrid approach[6] for their mobile AR system, where a spatial model is used to prefilter visual elements to reduce input information for a rule-based filter component.

Full or partial migration of user interface elements between devices and displays allows the selection of the most suitable environment for presenting application information.[7] Interface migration examples in AR include SHEEP[8] which employs gestures to transfer virtual sheep characters between multiple displays. Similarly, Schmalstieg *et al*.[9] created a shared collaborative workspace that enables the migration of applications between AR displays to address ad hoc collaboration. Augmented surfaces[10] use interface migration techniques in a spatially continuous augmented physical workspace. EMMIE[11] introduces a hybrid user interface for AR systems enabling room-wide information management.

Autonomous Interface Agents

All adaptive interfaces mentioned above share a significant shortcoming: the world model must be finite to ensure appropriate system reaction. If a novel system

DOI: 10.1002/cav

component produces an unsupported type of information, new filters must be created. We thus argue that AR systems can benefit from software agent technology. Agents are designed to be independent from applications they are embedded into, enabling their employment in diverse application environments without reprogramming the applications. By explicitly modeling application goals and user interest an agent can better cope with the indeterministic nature of augmented physical environments than explicit direct manipulation techniques.

Software agents have been present in AR applications so far in the form of animated anthropomorphic characters. The early ALIVE system[12] exhibits a virtual animated character composited into the user's real environment that responds to human body gestures on a large projection screen. The Welbo project[13] features an immersive setup, where an animated virtual robot assists an interior designer wearing an HMD. MacIntyre *et al*.[14] place prerecorded video-based actors into an AR environment to create an interactive theater experience. Cavazza *et al*.[15] place a live video avatar of a real person into a mixed reality setting, and interact with a digital storytelling system with body gestures and language commands. Balcisoy *et al*.[16] employed virtual humans in mixed reality as collaborative game partners, while Vacchetti *et al*.[17] used a virtual lifelike character in a training scenario for real factory machinery.

Except for Reference 12, all aforementioned research projects feature agents with little or no autonomous behavior, relying on explicit user input for their actions. There is a clear distinction between interface agents[18] operating as assistants for a direct manipulation interface and autonomous agents[19] acting parallel with the user to carry out delegated tasks being uninteresting or time consuming. We combine the advantages of both approaches into an autonomous interface agent[20] that executes tasks and provides feedback without constant attention and explicit commands while monitoring the user's environment and actions.

Mobile Agents

Recent advances in hardware and software technology for portable devices such as PDAs and smartphones have eliminated previously serious constraints on the visual representation of mobile agents.[21] While C-MAP[22] visualizes its context-aware virtual museum guide as a sequence of static images, Wagner *et al*.[23] already present a similar scenario in AR. PEACH[24] uses a cartoon character for animated presentations spanning multiple displays. The mobile agents of Virtual Raft[25] appear to "jump" between tablet PCs. The Agent Chameleons[26] allows agents to seamlessly travel between real and virtual bodies. However, the mobile agents in these examples differ from our work in that they do not use AR displays.

Bringing Animated Agents to Augmented Reality

Augmented Reality raises new challenges and offers novel features that can be exploited by animated agents. These implications can be divided into two categories: representation and behavior.

Representation

Animated agents are embodied as three-dimensional virtual or physical objects. They share users' physical environment, in which they can freely move using all 6 degrees of freedom. Virtual agents in AR scenarios appear to have a solid, tangible body that can be observed from an arbitrary viewpoint, thus becoming integral parts of the physical environment. Virtual objects are typically animated characters but are not necessarily anthropomorphic.

An exciting aspect of UbiAgents is that physical objects like a printer, a digital piano, or an interactive robot can be turned into intelligent, responsive entities that collaborate with virtual characters. If we track and monitor relevant physical attributes, attribute changes can be interpreted by other agents. Using wireless communication, physical objects cannot only be queried for status information but can also be controlled by external commands. The combination of the real and virtual representation results in the augmented representation, as shown in an example displaying an augmented Lego Mindstorms robot in Figure 1. The augmented representation also helps overcome the problem of correct visual occlusion. Physical objects placed between the user's viewpoint and virtual agents appear to cover parts of the virtual objects.

In AR scenarios users are mobile, traveling between different physical locations and hardware setups, therefore requiring cross-platform, mobile assistants. Agents can "live" on several devices and displays, like HMDs, projection screens, PDAs, or more recently mobile phones. Each has its own local coordinate system placed into the global coordinate system of the user's physical environment. UbiAgents are able to smoothly travel between devices and coordinate systems.

DOI: 10.1002/cav

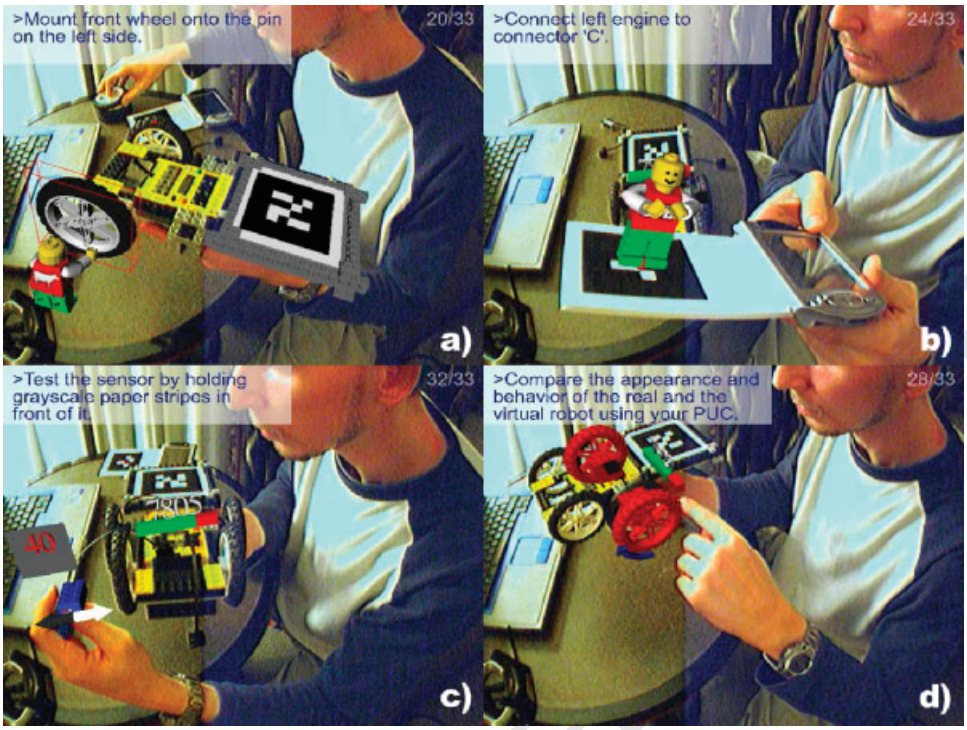


Figure 1. Screenshots from the AR Lego application.

Behavior

A UbiAgent interacts in real-time with other agents, with users and with the applications they are embedded into. In addition to executing scripts, agents react to changes in the properties of AR spatial objects. The agent monitors the physical and virtual world, facilitated by physical and virtual sensors, such as light, push, angular or temperature sensors. For virtual perception, agents can be equipped with object and application-specific sensors that examine application attributes, GUI input, and internal state information of virtual objects (e.g., the emotional state of a virtual human) and physical objects (e.g., an error message of a printer).

With the assistance of an internal simulation model, the agent performs actions in response to input events. AR offers novel, compelling input modalities involving pose, velocity, and status information of objects. The physical location agents inhabit, the direction they are looking into and the object they control all convey important context. For output, the agent uses facial and body gestures, speech synthesis, and playback of multimedia content. These new modalities enable a wide range of new behavioral patterns, such as the following:

- The user places a character into the physical working volume of an application. The character receives an event with the identity of the user and the application, and loads the user's application-specific profile and the state in which she last left the application. The character continues to work with this application.
- A virtual presenter is working with a user in an immersive AR setup and wears an HMD. She decides to work in another room with a projection screen suitable for a larger audience. She takes a pose-tracked PDA, moves it close to the character and "picks it up". The character continues to "live" on the PDA screen until it is carried over to the projection screen in the other room. It then becomes aware of the new environment and jumps to the projection screen, where the same application is running, maintaining the state of the user's work.
- A machine in a large PC cluster starts malfunctioning. Firstly, a virtual repairman character identifies the computer in the cluster room, then leads the human operator to the physical location. Once in the vicinity, the repairman points out the possible sources of error on the machine itself. An explanation is only begun once the operator looks at the repairman, in order to ensure appropriate attention and focus. The machine

DOI: 10.1002/cav

sends feedback to the repairman when it is back to its normal state.

Mobility for Animated Agents

As long as the agent is concerned with interfacing its host's application, it may remain embedded in the host environment. However, external conditions may make it necessary to migrate to another host. For example, an agent with an objective of remaining close to its user must migrate whenever the user wanders away. In general, the agent needs to monitor its host environment to satisfy its needs, or search for another environment providing more favorable conditions to complete its job. If such an environment is found, the agent opportunistically migrates to it and executes actions in its new "home."

Migration is signaled to the user by an animation sequence, text warning or sound alert to make the user aware of the migration action or to instruct the user to prevent migration within a certain grace period. For example, the agent may suspend its current activities and advise the user to charge her PDA or set the screen resolution of the PC monitor to match minimum requirements. After the grace period expires without appropriate changes in the local workspace, the agent migrates to its new preferred environment and resumes its actions. Although migration causes an interruption in the application flow, this temporary break is invoked in favor of a more efficient work environment.

Migratable user interfaces demand that dominant interface properties are preserved during and after migration to bridge the spatial and cognitive gap between disjoint workspaces. The user has to create a mental link between the old and new workspaces, thinking that the same virtual assistant continues to aid her work, even if it migrated to a projection screen from a local display to increase its public exposure.

The agent should appear to continue its task exactly at the point where it left off before migration. Beside temporally continuous agent behavior, visual agent appearance also frequently needs to remain unchanged across multiple agent environments by migrating respective 3D models, textures, color schemes, spatial arrangement etc. together with the mental state. Nevertheless, this is not a general requirement, since in some cases the agent may deliberately choose a different visual representation. For example, a simple arrow may replace the pointing gesture of the full-body animated repairman of the robot maintenance application to avoid occluding details of tiny mechanical robot parts. Similarly, the agent may choose to occupy a physical body taking advantage of sensors and actuators affecting the real world.

Agent migration and the preservation of agent attributes and mental state during migration necessitate the use of an external knowledge base. We call this component the agent brain, being in charge of controlling multiple agent representations or agent bodies. The agent brain relies on a persistent information storage, where agent and workspace attributes can be saved and recalled.

Expect the Unexpected

Let us imagine that we buy a new microwave oven and want to employ an AR system to explain its operation. With current classic AR software design we would use a standalone application tailored to the explanation of our specific microwave oven model. We cannot get our favorite animated agent—say, a repairman—to introduce us our new household item, as this agent has never "seen" a microwave oven before and thus it does not know how to explain it.

UbiAgents can adapt to unknown applications. If we enhance the aforementioned household scenario with UbiAgent components, the microwave oven application becomes part of the dynamic world model of the new UbiAgent-based repairman character. If the application generates a request calling for an animated presentation of its typical features, the repairman agent migrates to the new environment and starts the explanation.

UbiAgents communicate with AR applications via input and output attributes described by a schema, comparable to an interface definition language. Any agent "understanding" this schema is able to automatically establish communication with the application, deduce application state by monitoring these attributes, and influence application behavior by modifying attribute values. Due to the standard application interface described by the schema, the agent does not have to be aware of the fact that it is working with a microwave oven, a photocopier, or a factory machine as long as these objects properly map parameters describing their appearance and behavior to interface attributes well known to the agent. Any parameter changes not communicated through schema attributes are ignored by the agent.

The diversity of AR applications demands the creation of multiple schemata. Systems acting in the fashion of

DOI: 10.1002/cav

digital manuals need a schema enabling presentation, while navigation systems necessitate a schema for encapsulating parts of the physical environment such as floors, offices, streets, and buildings.

Multi-purpose agents need to understand several schemata to let the same agent work as a virtual tour guide or animated technician on demand. When a hitherto unknown application appears in the system and a UbiAgent wants to communicate with it, the agent first checks its schema. If the agent "speaks the language" of the schema, it includes the application in its control loop and reacts accordingly to attribute changes.

UbiAgents' capability of working with previously unknown applications is centered around the presence of well-designed schemata for all applications that may become part of the UbiAgents' world model, as a schema maps the internal state of unknown and undeterministic objects to a deterministic set of attributes familiar to the agent. Without exposing an appropriately formed schema, UbiAgents cannot become aware of an application. UbiAgents are therefore not capable of "learning" entirely new things in a strict Artificial Intelligence sense but rather of exploiting previously unknown resources such as in UbiComp.

User Interface Adaptation

Users favor customizable interfaces over fixed ones. People have diverse preferences for the color, size, spatial arrangement, and numerous other style elements of user interface components, including accessibility features for the disabled.

Present AR systems offer tweaking interface appearance, but customization information is only considered in the current session without being stored persistently. As illustrated by Figure 2, the UbiAgent framework includes a persistent knowledge base to store user preferences observed and accumulated by a learning

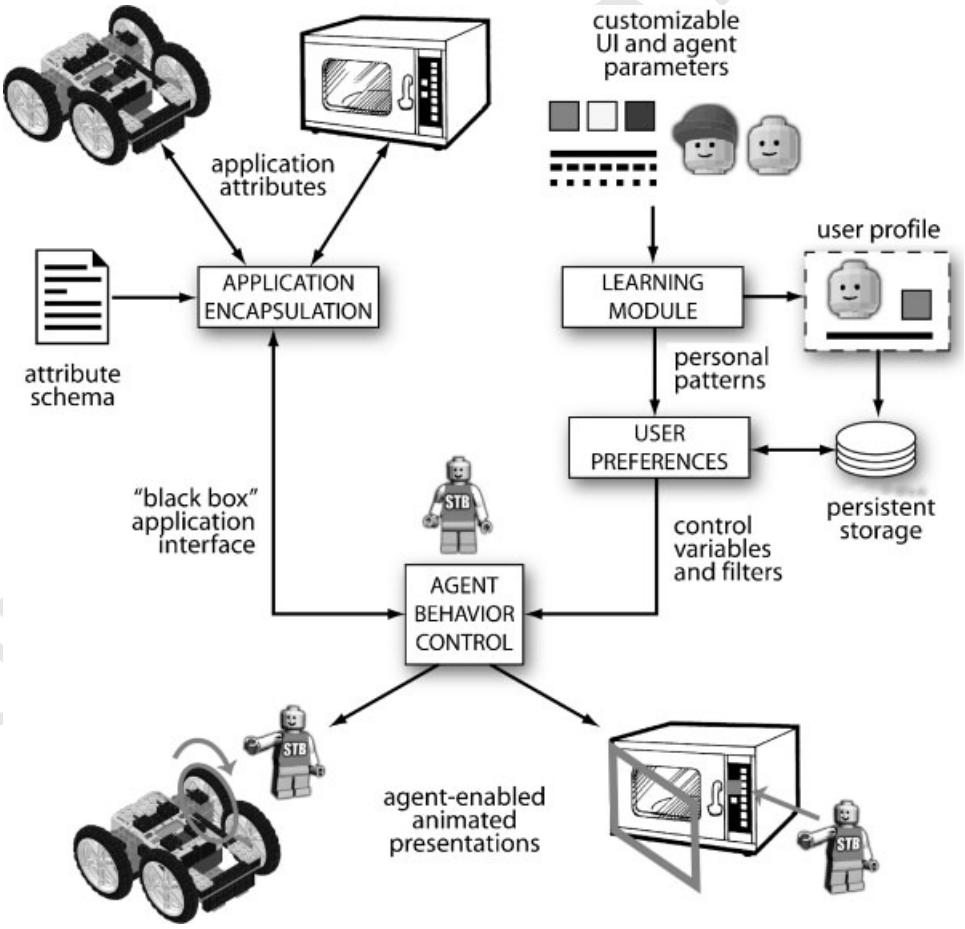


Figure 2. Application encapsulation with schema and adaptive user interface personalization.

DOI: 10.1002/cav

module for future application sessions. The learning module captures typical patterns in the way users set interface customization parameters with statistical methods and stores them in a personal user interface profile in the knowledge base. This personalization profile follows users around while they are working with multiple distributed applications running on various computing devices.

The UbiAgent framework is based on a fast and robust knowledge base that enables storing and recalling preferences on demand for a large number of users, thus enhancing AR systems with user interface adaptation capabilities. Identification of individual users relies on a unique user ID associated with personal devices such as PDAs or tablet PCs, or based on user accounts for shared public computers.

Beliefs, Desires, Intentions

Our framework follows the belief-desire-intention (BDI) model[27] for the implementation of the agent's reasoning mechanism. This model is not only one of the most well-known approaches for practical reasoning agents with a substantial research corpus, but is also highly suitable for dynamic and uncertain environments such as AR systems. Figure 3 depicts the BDI model-based structure of UbiAgents.

Beliefs represent the agent's current knowledge of the real world (such as the estimated pose and internal attributes of application objects) mapped to an internal world model. Since the world model represents only a potentially imperfect local view of the physical and virtual world, it needs to be regularly updated by measurements coming from sensors in the real and virtual environment. The knowledge base caches the current world model state between measurements and stores persistent information such as user preferences, application attributes, and agent properties.

Desires stand for agent goals associated with a desired end system state. They represent high-level concepts in the UbiAgent's brain subordinating user interface components to adapt their behavior to achieve goals as quickly as possible. UbiAgents work towards their goals by carrying out tasks or intentions using actuators in the real and virtual world. The currently executed tasks are constantly re-evaluated to verify whether they are efficiently advancing the system towards the end state. The system may reconsider its decisions in case of inadequate progress, and kill suboptimal tasks while starting new, more promising ones.

According to Georgeff *et al*.[28] adaptive, goal-oriented systems offer a superior performance compared to task-oriented systems in dynamic environments requiring automatic recovery from erroneous situations. In task-oriented systems each task strives to achieve a local optimum without remembering the purpose of its execution. In ubiquitous AR environments, where failures and suboptimal working conditions are inevitable, such an adaptive software architecture can tackle issues such as computer crashes, load balancing, and resource discovery.

In UbiAgents, goals represent a combination of desired application and agent states, for instance "the repairman presents the operation of the oven's alarm clock." This high-level goal is decomposed into subgoals or plans

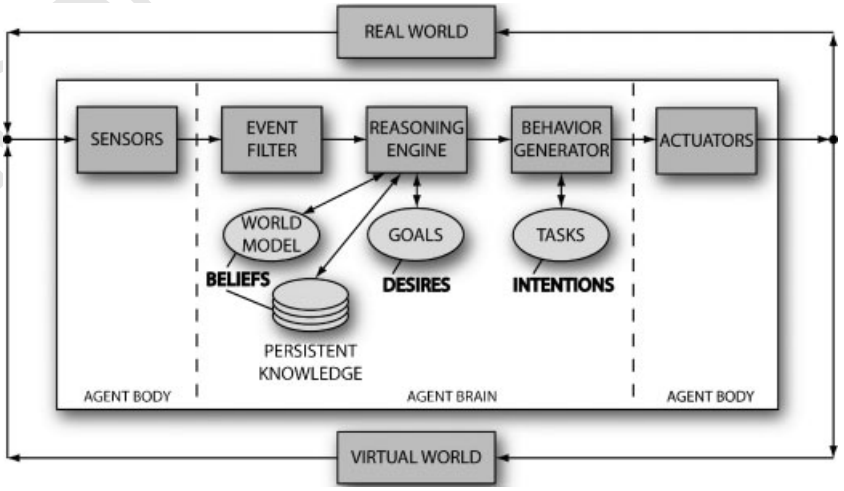


Figure 3. UbiAgent structure based on the BDI model.

equivalent to application attribute changes and animated agent action sequences such as "explain how to set hours, then explain how to set minutes."

Plans are converted into concrete agent tasks that are executed by actuators available in the current agent environment. In AR actuators can be physical as well as virtual. Typical examples for virtual actuators include animation engines controlling 3D models and virtual characters, 2D text messages, sound, text-to-speech engines etc. Common physical actuators involve stationary and mobile computers, fixed and portable displays, audio speakers, electric motors, and control systems of mechanical engines.

Before task execution, the UbiAgent framework checks whether actuators required for the next task are available on the current agent platform. If the desired sensors and actuators are missing or fail to meet the minimum requirements in the current agent environment, the UbiAgent consults its beliefs in the knowledge base storing persistent knowledge about all available agent platforms, and looks for a more suitable environment where it can migrate to and complete its current job.

Implementation

In this section we present the UbiAgent framework components and explain their functions with references to the design principles described in the previous section. Figure 4a shows a diagram with all UbiAgent entities and their relationships.

Agent Body, Brain, and Habitat

Each UbiAgent consists of an agent brain and one or more agent bodies. The agent brain serves as a control logic and reasoning engine controlling global agent behavior. It can be scripted by a propriatery scripting language composed of commands with a hierarchical grammar. Non-terminal symbols in the grammar represent abstract agent instructions (such as "go to next door"), which will be gradually decomposed into terminals or executable low-level commands with exact coordinate positions, speed, path, and other animation parameters. These low-level commands are then passed on the agent bodies for execution.

The agent body is a local representation of the agent brain and an embodiment of the agent. As UbiAgents operate in AR environments, they are allowed to possess real as well as virtual bodies, and thus appear to be integral parts of the user's physical surroundings. The agent body contains sensors observing the agent's real and virtual environment, and actuators affecting the physical and virtual world.

UbiAgents do not exist on their own, they need a host environment. In our framework we call this host environment a habitat, which is characterized by its hospitability attribute, combining diverse parameters into a single heuristic value.

The hospitability value hides irrelevant internal technical details of diverse computing platforms such as CPU load, available memory, or remaining battery power, yet allows agents to identify erroneous situations

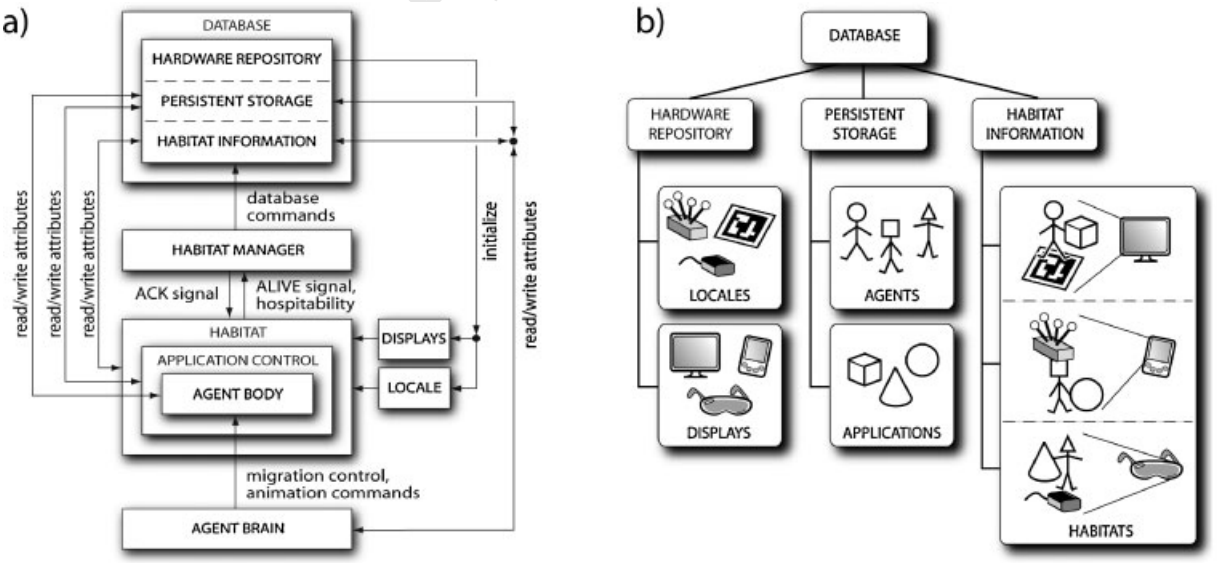


Figure 4. (a) Entities and their relationships in the UbiAgent framework, (b) Structure of the UbiAgent knowledge base.

DOI: 10.1002/cav

such as a crashing or overloaded device by their low hospitability value. Poor habitat conditions endangering agent operation trigger survival agent behavior that usually forces migration to another habitat. Graceful degradation of agent services is also possible as the availability of resources declines.

Computing devices serving as agent habitats in AR must provide one or more displays and a network of tracking systems called a locale, to support the superimposition of virtual information over the real world. Some parameters such as display size or the degrees of freedom of tracking data rarely change, therefore they are stored in a hardware repository.

Habitats constantly update their current attributes in a habitat information storage, including references to applications they are hosting, references to agents currently embedded in the applications, and information about currently associated locales and displays. Display and locale parameters are loaded from the repository, while dynamic parameters such as screen resolution are updated by the respective embedded display or locale component.

The UbiAgent components and AR applications are built on the Studierstube AR framework,[29] and implemented as scene graphs based on Open Inventor, a multi-platform high-level 3D graphics API. In Inventor all scene graph objects interact with one another via input and output attributes or "fields," which also provide control variables for the C++-based control logic of the scene graph objects. The brain implementation of UbiAgents in the demo applications of Section "Applications" has also been based on C++ code, however, dynamic scripting approaches for scene graphs such as Pivy[30] enable the dynamic uploading of procedural code, making the use of agents more flexible.

The application control marks fields relevant to the application state with a special tag, which allows constant observation of their values. A disadvantage of this scene graph approach is the lack of support for legacy applications. Interfaces to legacy applications must be implemented on a case-by-case basis.

Shared Knowledge Base

The dynamic nature of AR environments makes constant and reliable communication between framework objects crucial. Thus the knowledge base plays an eminent role for UbiAgents. The knowledge base not only stores the hardware repository, current habitat information, and persistent UbiAgent component attributes, but also serves as a shared memory between agents and applications. Components can register as observers in the knowledge base, which are requests for notifications about changes in certain elements. When a particular application or agent writes a message into the shared memory, all applications and agents having registered as observers for that particular element receive a notification about the update. This mechanism makes the knowledge base an effective communication medium in our ubiquitous AR framework.

To avoid losing synchronization when a UbiAgent has multiple bodies, communication between agents and applications happens at a higher level: the application control and the agent brain exchange messages through the knowledge base, controlling actions of application instances, and agent bodies. An application control object is embedded into the application, which maps internal application state to public knowledge base elements. The mapping function implements the schema concept described in the "Expect the Unexpected" Section. The application control creates a transparent interface between multiple agents and applications to hide private implementation details. This interface enables already existing complex AR systems to exploit agent services without any modifications in structure and code. By employing multiple application controls, different schemata can be supported, allowing a versatile use of the application with various agents.

Agent Migration

We use Muddleware[31,32] to implement the UbiAgents' knowledge base and shared application and agent memory. Muddleware provides fast and robust access to the UbiAgent knowledge base, which has a well-defined hierarchical structure (see Figure 4b). The Muddleware technology uses XPath-based queries and observers. The XPath language syntax suits the hierarchical structure of the UbiAgent knowledge base and enables the use of complex queries and observers to receive information about UbiAgent components. With XPath expressions agents can quickly identify habitats matching a set of infrastructure requirements such as hospitability, display parameters, tracking data, and application and agent attributes.

The agent brain controls a finite state machine. Each state defines a set of desired actions for agent bodies. When entering an agent state, an observer is registered to represent minimum expectations about the ideal environment for the agent bodies' actions. If the observer

DOI: 10.1002/cav

reports the appearance of a more suitable habitat, the agent brain instructs the currently embedded agent body to migrate to the new location.

Migration can happen in two ways: either by serialization techniques of distributed shared scene graphs[33] to create a new agent body, or by activating already existing "sleeping" agent bodies with remote commands sent over the network while deactivating previous ones. We also created a GUI-based UbiAgent browser to issue custom queries for debugging purposes and to trigger forced agent migrations for simulations.

Applications

We present two applications to illustrate the UbiAgents' capabilities. The first application shows how animated agents can enhance AR applications. The second application focuses on application encapsulation and agent mobility.

LEGO Robot Maintenance

Two UbiAgents are employed to educate an untrained user to assemble, test and maintain a LEGO Mindstorms robot: a physical, augmented LEGO Mindstorms robot, and a virtual repairman. By monitoring the attributes of the LEGO assembly application's interface, the UbiAgent framework is able to automatically generate behavior for the animated virtual repairman (see Figure 1).

The assembly application's interface provides the agent with all the relevant information about the next building block to be mounted. It outputs the current construction step so that the agent is aware of the user's progress, i.e., what was and needs to be constructed. Based on relevant information, the appropriate LEGO block is generated, which is then linked to the agent's hands. The position of the next block allows the movement from the agent's current location to the tile's target location to be planned without bumping into the already constructed model. The block's suggested orientation instructs the agent to put the block into the correct pose before mounting. Finally, the virtual block is added to the real robot using an appropriate gesture.

Engines and sensors (push, light, rotation, temperature etc.) are important components of the robot since misconfiguration would lead to erroneous behavior. To let the user quickly verify whether they are functioning correctly, the physical robot is augmented with a UbiAgent, which accepts commands from the assembly application. This means that as soon as the user finishes mounting an engine, the robot attempts to turn the engine on and lets the user inspect if it is working. In case of incorrect behavior, the current or previous construction steps have to be repeated.

A wireless PDA serves as a tangible interface to move the virtual repairman and virtual LEGO tiles around in the physical environment. It also shows a 2D user interface to navigate the assembly sequence and control the LEGO robot's engines and sensors. Since the PDA is tracked, it can be used to pick up the repairman and drop it at a desired task location.

Ubiquitous Technician

In this scenario a technician is assisted by a UbiAgent on his daily tour of duty in our lab. The agent follows the technician and takes on various forms that are helpful in carrying out the necessary maintenance tasks. We re-use existing AR applications to demonstrate how UbiAgents are capable of migrating from location to location and task to task. Figure 5 provides illustration.

The UbiAgent's knowledge base contains a list of maintenance tasks and their associated task locations. The UbiAgent uses Signpost, an AR navigation system[34] to guide the technician to the next task's location. During navigation, the UbiAgent resides on the technicians handheld computer, a Sony VAIO U70 equipped with optical, intertial, and ultra-wideband (UWB) tracking sensors.

At the target location, the agent migrates to the host of the AR application associated with the current maintenance task. The first job is to calibrate a cell of a UWB tracking system. An AR application[35] visualizes angle-of-arrival sensor measurements by virtual rays emanating from the physical sensors, and helps overcome problematic situations such as multi-path signals caused by reflections from metal ceilings and doors. After the calibration procedure, the technician is guided to the next task, the LEGO maintenance scenario described above.

In the Ubiquitous technician scenario three previously independent AR applications—LEGO robot, UWB calibration, Signpost—communicate with one another using the shared knowledge base. New applications can be dynamically added to the to-do list by encapsulating them with appropriate markup, containing the physical location as well as start and completion criteria for the task.

DOI: 10.1002/cav

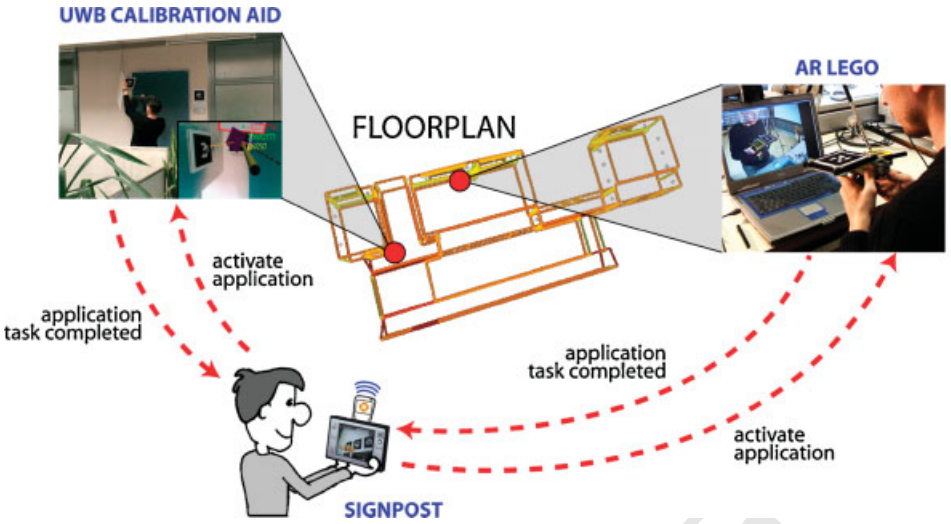


Figure 5. Going through a technician's to-do list with the ubiquitous technician application. This application seamlessly combines the Signpost indoor AR navigation system, an ultra-wideband calibration aid, and the AR Lego machine maintenance application with the help of an application schema.

Concluding Remarks

Weiser[36] questions the usefulness of embodied interface agents by juxtaposing them with Ubicomp systems. He argues that assistant-like interfaces increase the seam between humans and computers, which conflicts with the fundamental goals of Ubicomp. We believe that empowering our interface agents with autonomous behavior minimizes the required explicit user input to ensure correct agent operation, which is important in particular in AR applications.

User preference for agent representations spans a wide spectrum between lifelike and non-anthropomorphic embodiments. The robot maintenance application uses a human-like animated character, while the calibration aid employs simple geometrical shapes to visualize application state. Animated gestures can be very instructive, but in some AR systems a simple arrow may prove more useful than a full-body animated character. A possible solution to match preferences and purposes of a wide range of users and applications may be the employment of multiple agent bodies with varying behavior, appearance, and style. The appropriate agent body would either be explicitly selected by the user, or automatically chosen by the application matching the amount of information currently shown on the display to avoid clutter.

Another important variable in agent systems is the amount of proactivity ranging between submission and aggression. Humans are normally suspicious about systems that exclude users from the decision making. On the other hand, the complexity of computing systems including AR systems will soon reach a level where direct manipulation interfaces become so saturated with controllable parameters that users will have no other choice than delegating interface manipulation tasks. We also share Lieberman's view that agents are rather suited for making uncritical decisions,[20] therefore we let agents make suggestions instead of taking immediate actions. A typical UbiAgent example is the grace period allowing appropriate user response before agent migration.

For the work presented in this paper we have created our own ad hoc ontology for adaptive AR systems without the intention of completeness. However, an exhaustive ontology is highly desirable, in particular if based on standards such as WSDL.[37] This would allow information sharing between UbiAgents and other AR and agent systems, supporting resource sharing between diverse computing systems. Another important issue for future work is to eliminate vulnerability of a single central knowledge base, which makes the system prone to network and computer failures.

ACKNOWLEDGEMENTS

This project has been sponsored by the Austrian Science Fund FWF (contract No. Y193). The authors wish to thank Joseph Newman for valuable discussions on UbiAgent concepts, Daniel Wagner for his Muddleware and FPK software packages, and Gerhard Schall for his help with the Signpost system.

References

1. Poupyrev I, Tan DS, Billinghurst M, Kato H, Regenbrecht H, Tetsutani N. Tiles: a mixed reality authoring interface. In *Proceedings of Interact 2001*, Zurich, Switzerland, 2001; pp. 334–341.
2. Barakonyi I, Psik T, Schmalstieg D. Agents that talk and hit back: animated agents in augmented reality. In *Proceedings of International Symposium on Mixed and Augmented Reality (ISMAR'04)*, Arlington, VA, USA, 2004; pp. 141–150.
3. Barakonyi I, Schmalstieg D. Ubiquitous animated agents for augmented reality. In *Proceedings of International Symposium on Mixed and Augmented Reality (ISMAR'06)*, Santa Barbara, CA, USA, 2006.
4. Bane R, Höllerer T. Interactive tools for virtual X-ray vision in mobile augmented reality. In *Proceedings of International Symposium on Mixed and Augmented Reality (ISMAR'04)*, Arlington, VA, USA, 2004; pp. 52–62.
5. Feiner S, MacIntyre B, Seligmann D. Knowledge-based augmented reality. *Communication of the ACM* 1993; **36**(7): 52–62.
6. Julier S, Lanzagorta M, Baillot Y, Brown D. Information filtering for mobile augmented reality. *Computer Graphics and Applications* 2002; **22**(5): 12–15.
7. Molina Massó PJ, Vanderdonckt J, López PG. Direct manipulation of user interfaces for migration. In *Proceedings of International Conference on Intelligent User Interfaces (IUI'06)*, Sydney, Australia, 2006; pp. 140–147.
8. MacWilliams A, Sandor C, Wagner M, Bauer M, Klinker G, Brügge B. Herding sheep: live system development for distributed augmented reality. In *Proceedings of International Symposium on Mixed and Augmented Reality (ISMAR'03)*, Tokyo, Japan, 2003; pp. 123–132.
9. Schmalstieg D, Reitmayr G, Hesina G. Distributed applications for collaborative three-dimensional workspaces. *Presence: Teleoperators and Virtual Environments* 2003; **12**(1): 52–67.
10. Rekimoto J, Saitoh M. Augmented surfaces: a spatially continuous work space for hybrid computing environments. In *Proceedings of Conference on Human Factors in Computing Systems (CHI'99)*, Pittsburgh, PA, USA, 1999; pp. 378–385.
11. Butz A, Höllerer T, Feiner S, MacIntyre B, Beshers C. Enveloping users and computers in a collaborative 3d augmented reality. In *Proceedings of International Workshop on Augmented Reality (IWAR'99)*, San Francisco, CA, USA, 1999; pp. 35–44.
12. Maes P, Darrell T, Blumberg B, Pentland A. The alive system: wireless, full-body interaction with autonomous agents. *ACM Multimedia Systems* 1997; **5**(2): 105–112.
13. Anabuki M, Kakuta H, Yamamoto H, Tamura H. Welbo: an embodied conversational agent living in mixed reality space, extended abstracts. In *Proceedings of Conference on Human Factors in Computing Systems (CHI'00)*, The Hague, The Netherlands, 2000; pp. 10–11.
14. MacIntyre B, Bolter JD, Vaughan J, Hannigan B, Moreno E, Haas M, Gandy M. Three angry men: dramatizing point-of-view using augmented reality. In *Proceedings of SIGGRAPH 2002 Technical Sketches*, San Antonio, TX, USA, 2002.
15. Cavazza M, Martin O, Charles F, Mead SJ, Marichal X. Interacting with virtual agents in mixed reality interactive storytelling. In *Proceedings of Intelligent Virtual Agents*, Kloster Irsee, Germany, 2003.
16. Balcisoy S, Kallmann M, Torre R, Fua P, Thalmann D. Interaction techniques with virtual humans in mixed environments. In *Proceedings of International Symposium on Mixed and Augmented Reality (ISMAR'01)*, Tokyo, Japan, 2001.
17. Vacchetti L, Lepetit V, Papagiannakis G, Ponder M, Fua P. Stable real-time interaction between virtual humans and real scenes. In *Proceedings of International Conference on 3D Digital Imaging and Modeling (3DIM'03)*, Banff, AL, Canada, 2003; pp. 449–457.
18. Laurel B. Interface agents: metaphors with character. In *The Art of Human-Computer Interface Design*, Laurel B (ed.). Addison-Wesley: Reading, MA, USA, 1990.
19. Franklin S, Graesser A. Is it an agent or just a program? A taxonomy for autonomous agents. In *Agent Theories, Architectures and Languages*. Springer Verlag: Berlin, Germany, 1996; 21–95.
20. Lieberman H. Autonomous interface agents. In *Proceedings of Conference on Human Factors in Computing Systems (CHI'97)*, Atlanta, GA, USA, 1997; pp. 67–74.
21. Kotz D, Gray RS. Mobile agents and the future of the internet. *SIGOPS Operating Systems Review* 1999; **33**(3): 7–13.
22. Mase K, Sumi Y, Kadobayashi R. The weaved reality: what context-aware interface agents bring about. In *Proceedings of Asian Conference on Computer Vision (ACCV'00)*, Taipei, Taiwan, 2000.
23. Wagner D, Billinghurst M, Schmalstieg D. How real should virtual characters be? In *Proceedings of Conference on Advances in Computer Entertainment Technology (ACE'06)*, Los Angeles, CA, USA, 2006.
24. Kruppa M, Krüger A. Concepts for a combined use of personal digital assistants and large remote displays. In *Proceedings of Simulation and Visualization (SIMVIS'03)*, Magdeburg, Germany, 2003; pp. 349–361.
25. Tomlinson B, Yau ML, Baumer E. Embodied mobile agents. In *Proceedings of International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'06)*, Hakodate, Japan, 2006.
26. Duffy BR, O'Hare GMP, Martin AN, Bradley JF, Schön B. Agent chameleons: agent minds and bodies. In *Proceedings of International Conference on Computer Animation and Social Agents (CASA'03)*, New Brunswick, NJ, USA, 2003.
27. Bratman ME. In *Intentions, Plans, and Practical Reason*. Harvard University Press: Cambridge, MA, USA, 1987.
28. Georgeff M, Pell B, Pollack M, Tambe M, Wooldridge M. The belief-desire-intention model of agency. In *Proceedings Of International Workshop on Intelligent Agents*, Heidelberg, Germany, 1999; pp. 1–10.
29. Schmalstieg D, Fuhrmann A, Hesina G, Szalavári Z, Encarnação M, Gervautz M, Purgathofer W. The Studierstube augmented reality project. *PRESENCE—Teleoperators and Virtual Environments* 2002; **11**(1): 32–45.
30. Pivy website. http://pivy.tammura.at/
31. Handheld AR libraries website. http://www.studierstube.org/handheld_ar/
32. Wagner D, Schmalstieg D. Muddleware for prototyping mixed reality multiuser games. In *Proceedings of IEEE Virtual Reality 2007 Conference (VR'07)*, Charlotte, NC, USA, 2007.

DOI: 10.1002/cav

33. Hesina G, Schmalstieg D, Fuhrmann A, Purgathofer W. Distributed open inventor: a practical approach to distributed 3d graphics. In *Proceedings of ACM Virtual Reality Software and Technology (VRST'99)*, London, UK, 1999; pp. 74–81.
34. Kalkusch M, Lidy T, Knapp M, Reitmayr G, Kaufmann H, Schmalstieg D. Structured visual markers for indoor pathfinding. In *Proceedings of the First IEEE International Workshop on ARToolKit (ART02)*, Darmstadt, Germany, 2002. IEEE Computer Society.
35. Newman J, Schall G, Barakonyi I, Schürzinger A, Schmalstieg D. Wide area tracking tools for augmented reality. *Advances in Pervasive Computing 2006*, 207, 2006.
36. Weiser M. Does ubiquitous computing need interface agents? In *MIT Media Lab Symposium on Interface Agents*, Cambridge, MA, USA, 1992.
37. WSDL website. http://www.w3.org/TR/wsdl/

Authors' biographies:

István Barakonyi is a software developer at Imagination and Verdandi Gmbh in Austria developing handheld augmented reality applications. He obtained his PhD from the Vienna University of Technology and worked as a researcher at the Graz University of Technology on augmented reality user interfaces. He obtained a Masters Degree in computer science at the Budapest University of Technology and Economics. His research interests include stationary and mobile augmented reality applications, embodied autonomous agents, affective computing, and man-machine interfaces.

Dieter Schmalstieg is full professor of Virtual Reality and Computer Graphics at Graz University of Technology, Austria, where he directs the "Studierstube" research project on augmented reality. His current research interests are augmented reality, virtual reality, distributed graphics, 3D user interfaces, and ubiquitous computing. He received Dipl.-Ing. (1993), Dr. techn. (1997) and Habilitation (2001) degrees from Vienna University of Technology. He is author and co-author of over 100 reviewed scientific publications, member of the editorial advisory board member of computers & graphics, member of the steering committee of the IEEE International Symposium on Mixed and Augmented Reality, chair of the EUROGRAPHICS working group on Virtual Environments, and advisor of the K-Plus Competence Center for Virtual Reality and Visualization in Vienna. In 2002, he received the START career award presented by the Austrian Science Fund.

DOI: 10.1002/cav