Towards a System of Patterns for Augmented Reality Systems

Thomas Reicher, Asa MacWilliams, and Bernd Bruegge
Institut für Informatik
Technische Universität München
D-85748 Garching bei München, Germany
(reicher,macwilli,bruegge)@in.tum.de

***Abstract*— The discussion and comparison of software architectures of different Augmented Reality systems is often difficult because of the heterogeneous ways the developers document them. They use different notations, different abstraction levels, and have different intentions. On the other hand, there are terms that are quite well-known and understood among the developers of Augmented Reality systems that describe an employed abstract pattern or technique, such as scene graphs or OpenGL for rendering. Such patterns can be found at different abstraction levels.**

We propose to describe Augmented Reality systems by a system of such patterns. The base is an abstract generic architecture which describes the common subsystems found in most Augmented Reality systems. This approach allows to compare and discuss Augmented Reality systems by comparing the patterns they employ to implement a particular subsystem. We present a set of patterns that we identified in study on software architectures for Augmented Reality systems. As an example we describe the DWARF Pathfinder system by its employed patterns.

I. Introduction

The discussion and comparison of different software architectures for Augmented Reality (AR) is often difficult because of the heterogeneous ways the developers document them. They use different notations, different abstraction levels, and have different intentions.

A common ground for the comparison of software architectures is the analysis object model [1, pp 510] of the particular application. Often an application belongs to a class of applications, the domain. For each domain, there are specific functional and non-functional requirements which are mapped to common functions. In each architecture, these functions are implemented by subsystems. And for the implementation of a subsystem a developer uses a particular approach. In a given domain such as Augmented Reality, similar or identical approaches are used by various developers. Often this stems from the common use of software components or libraries that implement the same functionality, for example OpenInventor. The result is a vocabulary of common terms that are understood by most Augmented Reality developers. This enables discussion of the software architectures on the base of such terms. To classify the approaches, we extracted an abstract generic architecture for Augmented Reality systems [2] from the descriptions of existing systems. A software architecture for AR can be described by the set of approaches used in the system.

While this approach allows us to discuss Augmented Reality systems, it is only of little use for the design of new systems. For this, we must measure each approach within a certain context. The catalogue of known approaches then can mature to a system of known *patterns* for Augmented Reality systems. Each pattern must state the context where it is used, the problem it solves and the solution. In software architecture, patterns are structured descriptions of successfully applied problem-solving knowledge. Each approach is described by name, goal, motivation, a description, usability, consequences, and project usage. This follows the scheme of describing architectural or design patterns, e.g. as used in [3], [4].

II. A System of Patterns for Augmented Reality

A. A Generic Software Architecture for Augmented Reality

For the comparison of different software architectures, we developed an abstract generic architecture for Augmented Reality Systems on the base of the model-view-controller pattern (MVC) [5]. The MVC pattern separates interactive systems into subsystems for data and control code, user input and user output. We extend this model with specific extensions for Augmented Reality and ubiquitous computing, in particular tracking, a world model, and context. This divides an Augmented Reality system into a set of six subsystems: application (MVC model), interaction (MVC control), rendering (MVC view), tracking, context, and world model. These subsystems collaborate with each other and consist in turn of several components. The architectures of Augmented Reality systems can be mapped to this generic architecture [2].

B. Patterns Sorted by Subsystems

On a subsystem level the Augmented Reality system developers use different techniques and building blocks to implement the subsystems, e.g. tracking or presentation. An analysis of existing systems reveals that several techniques and building blocks recur in various existing systems—sometimes explicitly, such as when two systems use a common library, and sometimes implicitly, when different developers apply the same basic techniques. The selection depends on the non-functional requirements and the design goals.

These techniques and building blocks can be extracted from existing systems and described as abstract reusable patterns for Augmented Reality systems design. This is heavily based on

the idea of *design patterns* in software architectures. Patterns are structured descriptions of successfully applied problem-solving knowledge: *A software architectural pattern describes a specific design problem, which appears in a particular design context, and presents a generic solution scheme. The solution scheme specifies the involved components, their responsibilities, relationships and the way they cooperate* [6, pp 8].

The following list gives an overview of the patterns we found ordered by subsystems. We cannot discuss all of the identified patterns, instead we give some examples of interesting Augmented Reality patterns.

a) Application: The application subsystem is where developers can add application-specific logic. Various solutions are possible with different advantages and disadvantages. We identified the following approaches:

Main Executable Write the application in a high-level programming language, explicitly describing what happens when.

Scripting Use a scripting wrapper around all components that have performance constraints. These components are written in compiled languages such as C++.

Node in Scene Graph Model the world around a user as a tree of nodes, including non-graphical objects that include control code.

Part of Event Loop Provide hooks that can be called within a rendering library's update loop and that react to changes in the scene.

Web Service Keep control flow on a web server, publish AR content to an AR-enabled web client.

Multimedia Flow Description Use a high-level markup language for domain specific content such as workflow information and AR content such as repair tasks.

Application Component In a component-based system, encapsulate all application logic in a separate component that communicates with the others.

b) Tracking: Without tracking, Augmented Reality is impossible. Here, we concentrate on architectural approaches of gathering tracking data, not on the tracking devices or algorithms themselves.

Tracking Server Offload the processing of raw tracking data to a server in the user's environment and only transfer the result to the client system.

Networked Trackers For each tracking device, provide a middleware wrapper with an interface to the tracker. Consumers find trackers through middleware services and then communicate transparently.

Operating System Resources The tracking devices are accessed directly through operating system drivers.

c) Interaction: Augmented Reality systems tend to concentrate more on output than on input; however, the interest in new user input techniques and architectures is growing. We concentrate on architectural approaches of combining user input, not on the input devices themselves:

Direct Access Include input handling code in the application code, with explicit references to the types of input devices.

Browser Input Functions Use VRML browser events sent out through the External Authoring Interface (EAI) interface when the user clicks on on-screen objects with the mouse or when the gaze direction coincides with certain objects.

Networked Input Devices Provide an abstraction layer for input devices and a description of how the user input can be combined; interpret this description using a controller component.

Input Manager Coordinate several lower-level input devices to create higher-level input.

d) Presentation: This subsystem deals mainly with the presentation of three-dimensional information; thus, most of the approaches here are geared specifically to rendering. There are several approaches:

VRML Browser Use a third-party VRML browser, often designed as a web browser plugin, to display 3D information; access it using the EAI.

OpenGL Use low-level OpenGL 3D constructs.

Scene Graph Use scene graph library such as (Open) Inventor, OpenSG, Open Scene Graph.

Proprietary Scene Graph Use own scene graph for graphics rendering on top of OpenGL.

Video Transfer A server augments video images and sends them to the client, which shows them on the HMD.

Multiple Viewer Classes Provide an abstraction layer for different types of viewers (AR, speech, text etc.) that can handle different document types.

e) Context: The context information must be gathered, processed and distributed to interested components. Possible approaches are:

Blackboard Information producers write information to the Blackboard, a central component; consumers read data, process it and may write new, higher abstract data to the Blackboard.

Repository Components that produce context information write to the repository; components that are interested into context information read from the repository.

Publisher/Subscriber Context providers connect as publishers to a central messaging service, context consumers as subscribers.

Ad hoc An interested component directly queries the context producer component or it registers itself as subscriber.

f) World model: The World Model is used to describe the world around the user, particularly the virtual objects and their position. Besides that, the world model must also store information about the marker positions or any other features required for tracking. Approaches are:

OpenGL Code The developer creates OpenGL code and calls the OpenGL rendering engine to display it.

Scene Graph Format With an authoring tool, a content developer creates the model of a virtual scene.

Object Stream The runtime environment allows serializing and deserializing objects to and from disk.

Configuration File Load a file, e.g. for marker positions, at startup time or upon request at run time.

Database Instead of loading a particular scene from a file, the system has access to a database system with information

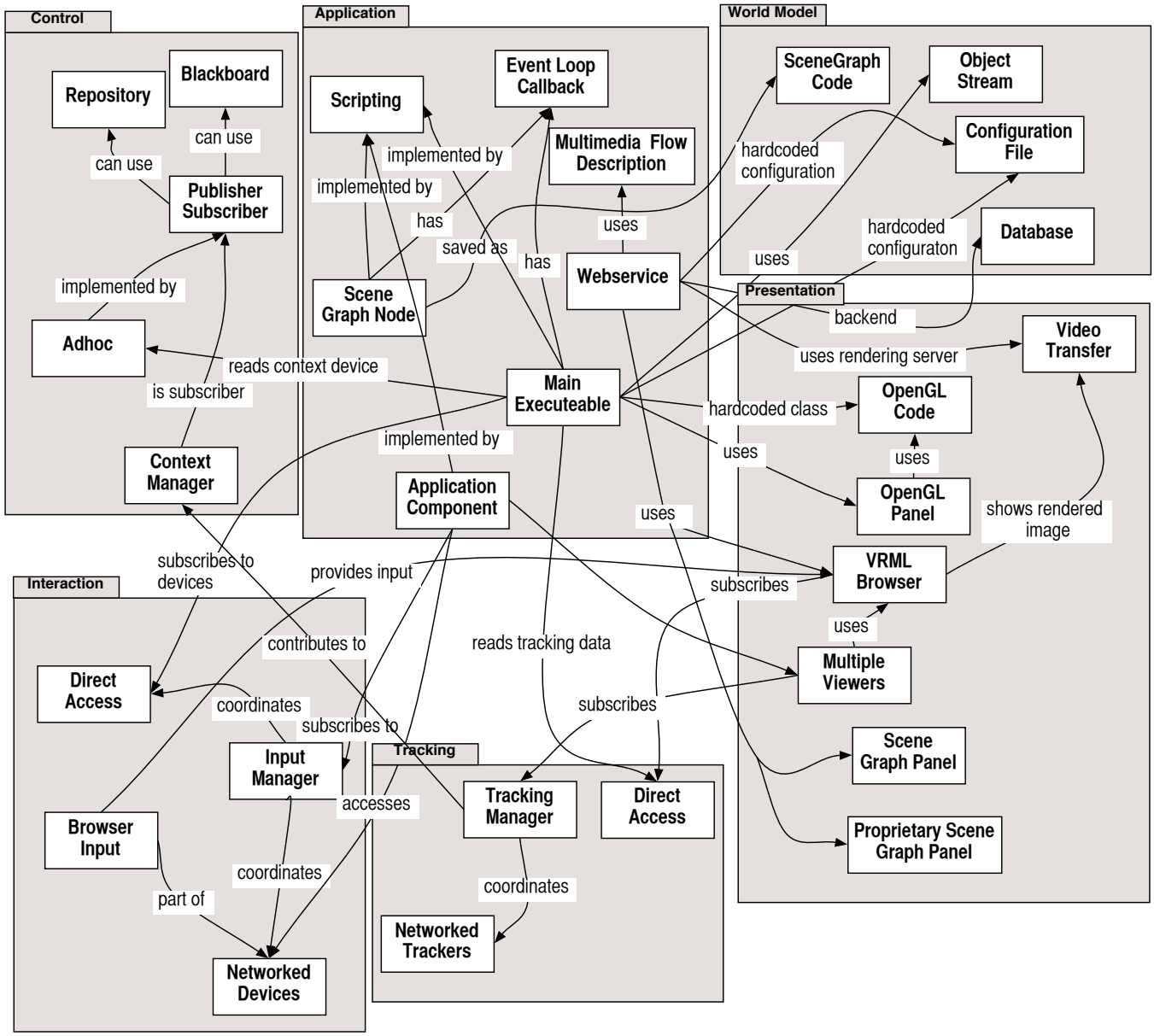

Fig. 1. Relationships between approaches for subsystem implementation. Several approaches are used in combination within an AR system. One approach might require the use of another approach or prevent its usage.

about the environment, e.g. in a geographical schema.

Figure 1 shows all identified approaches and links between approaches that are related to each other. For example, a webserver based application subsystem is often connected with a browser-based interaction and presentation subsystem. Each patterns is shown within the subsystem it belongs to.

III. Exemplary Approaches in Detail

We cannot discuss all the identified approaches here. Instead we present some examples. An approach is described by name, goal, motivation, a description, usability, consequences, and known project usage. This follows the scheme of describing architectural or design patterns, e.g. as used in [3], [4]. We see this as a first step towards the development of a pattern language for Augmented Reality system composition.

Name: Node in Scene Graph (Application):

Goal:	Embed application in world model.
Motivation:	In Augmented Reality, user interaction is connected with the physical environment. Consequently, applications are often linked to places in the real world. With this approach, the application is seamlessly embedded in the environment.
Description:	A scene graph models the world around a user as a tree of nodes. Each node can be any type of object, usually graphical ones.

But there are also non-graphical objects that include control code.

Usability: Together with a scene graph-based rendering approach.

Consequences: The scene graph-based approach for an application handles the control flow to the underlying scene graph platform, e.g. Open Inventor. On the other hand, this approach offers a relatively simple possibility for the implementation of shared applications for locally nearby users. One 3D interface can be shared among several users but displayed for each user from a different view.

Known use: Studierstube [7], Tinmith [8], MARS [9]

Name: Web Service (Application):

Goal: Treat Augmented Reality as one type of media among others.

Motivation: For content-based applications, the web-based approach has been proven to be a reasonable approach. Augmented Reality scenes and world model information can be seen as an Augmented Reality document. A scene such as an arrow that points to a particular button in front of the user is then described in document that is loaded from a web server.

Description: The control flow is situated on a web server and implemented within a web service. This web service is published under a particular web address and the answer of the service is rendered on a web client. If the answer contains Augmented Reality content, the Augmented Reality component is activated to display it.

Usability: This approach can be used where the focus is on displaying various types of content and loading them dynamically from a server.

Consequences: The client and the server must be connected. If a connection cannot be guaranteed, there must be a proxy available locally that emulates the server. Alternatively, a smaller instance of the server component may be deployed on the client machine. This approach should be combined with a scene-based rendering component, e.g. a VRML or custom Augmented Reality browser.

Known use: ARVIKA [10]

Name: VRML Browser (Presentation):

Goal: Use a rendering component that can display simple virtual scenes.

Motivation: The usage of a VRML browser is a simple way to display virtual scenes. The standardized VRML format, a markup language for the description of virtual worlds, allows the use of tools for authoring and rendering virtual worlds.

Description: Use a third-party VRML browser, often designed as a web browser plugin, to display 3D information. Use the External Authoring Interface (EAI) that is part of the VRML standard to modify the scene and set the viewpoint based on tracking data.

Usability: A VRML browser component can be used if the complexity of the scenes is relatively low and the browser is only used as a rendering engine.

Consequences: The advantages of using a VRML browser are the standardized format and the reuse of tools for authoring and the reuse of existing components. This allows rapid prototyping of Augmented Reality system based on VRML scenes. The disadvantages are that the EAI is restricted to relatively simple operations and that tying the VRML browser to the rest of the system may be tedious. Also, the rendering performance of VRML browsers is not as high as that of native OpenGL.

Known use: STAR [11], DWARF Pathfinder [12]

Name: Scene Graph (Presentation):

Goal: Use a rendering component that allows more complex and dynamic scenes.

Motivation: For the representation of 3D environments, scene graphs have shown to be a reasonable choice. The level of abstraction is higher than for OpenGL, but they are much more powerful and flexible than VRML browsers with their limited application programming interface. Most scene graph components can read VRML based descriptions of scenes.

Description: A scene graph is a structure that is based on a 3D scene database and includes objects typically used in 3D graphics such as various bodies, materials, lights, and cameras. Additionally, any other classes can be stored in a scene graph. A typical feature of a scene graph is the ability to traverse it and update the linked objects. Examples are (Open) Inventor, OpenSG, Open Scene Graph.

Usability: Use a scene graph if you don't need the low-level graphics access that OpenGL provides but want to render more complex scenes and need more dynamic access than a VRML browser offers.

Consequences: Can restrict the possibilities for modeling the application.

Known use: ARVIKA [10], Studierstube [7], Tinmith [8], DWARF Sheep [13]

We are aware of the fact that the above list is not complete and should be seen as a starting point. We collected the

approaches on a discussion page that is open to interested parties[1].

IV. Example: DWARF Pathfinder

As a case study for describing the software architecture of an existing Augmented Reality system using a system of architectural approaches, we present the DWARF Pathfinder system [12].

DWARF is a representative of the group of peer-to-peer systems. Goal of this architectural approach is the seamless integration of DWARF-based systems deployed on wearable computers into a ubiquitous computing environment. The base for this approach is a middleware designed for ubiquitous computing applications [14]. Note that the peer-to-peer approach is a pattern itself, but on a lower architectural level.

Application. DWARF Pathfinder uses the *Application Component* approach. This is a typical approach for distributed systems. The application logic is encapsuled into a distributed component that collaborates with other distributed components. The Application Component is responsible for the bootstrapping and provides the glue code for the application. Internally it uses a *Web Service* approach that provides the application content over Internet. The content itself is executed by an interpreter for a multimedia flow description language (*Multimedia Flow Description* approach).

The advantage of this approach is that the content for the Augmented Reality system is described in a high-level language. This allows a faster development of new content.

Tracking. DWARF Pathfinder used a combination of different tracking modalities. There was no hybrid tracking, each modality covered a specific part: GPS tracking for outdoors, room tracking to track the location within a building, and optical tracking for near-range. Each tracker worked independently and as part of a distributed system.

As a distributed system Pathfinder could use the approach of *Networked Trackers* combined with the approach to use a *Tracking Manager* that coordinates the trackers.

Interaction and Presentation. The goal for Pathfinder's interaction and presentation subsystems was to support different types of viewers depending on the content and to reuse existing viewer components. Pathfinder was developed to show simple augmentations.

The solution found was to use a web-based approach and write adaptors to integrate third-party web components with DWARF. The central component was a web browser with several plugins. Pathfinder employs a VRML plugin for 3D graphics controlled over the External Authoring Interface (EAI), a plugin for speech recognition, and an HTML frame for text and untracked graphics. Thus, Pathfinder uses the *VRML Browser* and *Multiple Viewers* approach for output and *Browser Input* and *Input Manager* for input. Technically, a User Interface Engine combined input and output control.

World model. Pathfinder used a simple file-based world model. The tracking information was based on a proprietary format for saving information about the environment such as the building and the area, and VRML files for the representation of 3D scenes. The approach is therefore called *Configuration File*.

[1] `http://wwwbruegge.in.tum.de/projects/lehrstuhl/twiki/bin/view/DWARF/ARPatterns`

V. Emerging Approaches

The list of approaches above identifies solutions for subsystems that we found in studying Augmented Reality systems. Besides these approaches in current systems, we identified several trends that we expect will result in new architectural approaches, but are not systematically employed yet.

Use scene graph for view and model. Most projects use a scene graph only as the viewer component of the system. Accordingly, only graphical objects are saved in the scene graph, often in VRML format that can be created by many authoring tools. But similar to HTML for web content, VRML has no means to transport the semantics of the objects in the model. Even feature information is not included in the model but saved separately. Some projects store every object that are related to some place in the world in the scene graph. Even applications become part of it, for example used in Tinmith and Studierstube. But there is no open format for world model information, Tinmith and Studierstube use object serialization mechanisms.

Use distributed trackers over middleware. Most commercial tracking systems are boxed and must be connected using low level communication means such as Unix sockets. Many projects try to encapsulate these low level access methods in higher level approaches from distributed computing, for example, the ACE [15] library or CORBA [16].

Provide multiple views on world. Most Augmented Reality systems are single-user systems. Some support multiple users, such as Studierstube, AR Boarderguard [17] (based on MR Platform) or DWARF Sheep [18]. Nevertheless, these solutions still lack a general concept based on a world model that is application independent. The approach that comes nearest to that goal is Studierstube with an extension of OpenInventor, called Distributed OpenInventor.

Adapt to tracking quality. Most current Augmented Reality systems assume a homogeneous tracking quality at any time. However, this is more a goal than reality. Due to technical compromises such as range of tracking devices or price, there will be the need to adapt the overall system to a varying tracking quality. Particularly the presentation subsystem must be adaptable. An existing example is AIBAS [19].

Model integration. There is a break between the 3D coordinate system of the tracking and the rendering subsystem, the network-oriented model of ubiquitous computing resources in the environment, and the relational databases used in industry [10]. We expect the integration of these models to be *the* research question to be solved for Augmented Reality to leave the labs.

VI. Conclusion

A system of abstract approaches for subsystems is a practical way to discuss the software architectures of existing

applications and prototypes. It provides a common vocabulary of well-known approaches for developers in the Augmented Reality domain. Of course this requires an agreement among the developers on the chosen names for the approaches.

In a next step, approaches that describe existing systems must mature into full patterns that can be used as guidelines for the development of new systems. Each approach must be examined for its usability in a specific context.

In this work, we have only considered architectural patterns. In several subsystems such as tracking, it would be worthwhile to establish more technical or algorithmic patterns for complex tasks such as sensor fusion. Also, existing virtual reality systems can provide a rich source for additional patterns that can be applied not only to VR, but to AR as well.

ACKNOWLEDGMENTS

This work was supported by the German Federal Ministry for Education and Research (BMBF) in the ARVIKA project, the High-Tech-Offensive Zukunft Bayern of the Bavarian Government, and the compound project Forsoft 2 within the Softnet subproject supported by the Bavarian research foundation.

REFERENCES

[1] B. Bruegge and A. H. Dutoit, *Object-Oriented Software Engineering: Conquering Complex and Changing Systems*. Upper Saddle River, NJ: Prentice Hall, 2000.

[2] T. Reicher, A. MacWilliams, B. Bruegge, and G. Klinker, "Results of a study on software architectures for augmented reality systems," in *Poster Session of IEEE and ACM International Symposium on Mixed and Augmented Reality ISMAR 2003*, Tokyo, Japan, 2003.

[3] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley, 1995.

[4] D. C. Schmidt, M. Stal, H. Rohnert, and F. Buschmann, *Pattern-Oriented Software Architecture, Vol. 2: Patterns for Concurrent and Networked Objects*. New York, NY: Wiley, 2000.

[5] G. E. Krasner and S. T. Pope, "A description of the model-view-controller user interface paradigm in the smalltalk-80 system," ParcPlace Systems, Inc., Mountain View, USA, Tech. Rep., 1988.

[6] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-Oriented Software Architecture. A System of Patterns*. John-Wiley & Sons, 1996.

[7] "Studierstube Augmented Reality Project," www.studierstube.org, 2003.

[8] W. Piekarski and B. H. Thomas, "Tinmith-evo5 - an architecture for supporting mobile augmented reality environments," in *Proceedings of the 2nd International Symposium on Augmented Reality (ISAR 2001)*, New York, USA, 2001.

[9] "Mobile augmented reality," www.cs.columbia.edu/graphics/projects/mars/mars.html, 1999.

[10] W. Friedrich, D. Jahn, and L. Schmidt, "Arvika - augmented reality for development, production and service," Proceedings of the International Status Conference HCI, 2001.

[11] STAR consortium, "Star," www.realviz.com/STAR/, 2002.

[12] M. Bauer, B. Bruegge, G. Klinker, A. MacWilliams, T. Reicher, S. Riss, C. Sandor, and M. Wagner, "Design of a component-based augmented reality framework," in *Proceedings of ISAR 2001*, 2001. [Online]. Available: citeseer.nj.nec.com/bauer01design.html

[13] C. Sandor, A. MacWilliams, M. Wagner, M. Bauer, and G. Klinker, "Herding sheep: Live system development for distributed augmented reality," in *IEEE and ACM International Symposium on Mixed and Augmented Reality ISMAR 2003*, Tokyo, Japan, 2003.

[14] A. MacWilliams, T. Reicher, and B. Bruegge, "Decentralized coordination of distributed interdependent services," in *Proceedings of Middleware 2003 Work-in-Progress*, Rio de Janeiro, Brazil, 2003.

[15] D. C. Schmidt, "The adaptive communication environment (ace)," www.cs.wustl.edu/~schmidt/ACE.html, 2003.

[16] Object Management Group (OMG), "Common object request broker: Architecture and specification, corba 2.6.1," www.omg.org/cgi-bin/doc?formal/02-05-08, 2002.

[17] T. Oshima, "Rv-border guards: A multiplayer entertainment in mixed reality space," in *Poster session of IEEE Internation Workshop on Augmented Reality*, San Francisco, USA, 1999.

[18] C. Sandor, A. MacWilliams, M. Wagner, M. Bauer, and G. Klinker, "Sheep: The shared environment entertainment pasture," in *IEEE and ACM International Symposium on Mixed and Augmented Reality ISMAR 2002*, Darmstadt, Germany, 2002.

[19] C. Robertson and B. MacIntyre, "Adapting to registration error in an intent-based augmentation system," in *ACM User Interface Software and Technology 2002 (UIST 2002)*, Paris, France, Oct. 2002, presented as a poster.