An Architecture for Distributed Spatial Configuration of Context Aware Applications

Martin Wagner Gudrun Klinker

Technische Universität München, Institut für Informatik
Boltzmannstr. 3, 85748 Garching bei München, Germany*

Abstract

This paper discusses an architecture for spatially distributed storage of contextual configuration information in ubiquitous computing environments. Based on the assumption that we want to integrate arbitrary mobile clients in ubiquitous computing environments, we derive the requirements for the spatial distribution of data, transparent access to context aware configuration data, and separation of context estimation algorithms. We developed a highly distributed architecture that fulfills these requirements. Using DWARF, we implemented and successfully demonstrated a mobile demonstration setup that incorporates all key concepts of our architecture.

CR Categories: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities; C.2.4 [Computer-Communication Networks]: Distributed Systems—Distributed applications

Keywords: Augmented Reality, Context Awareness, Ubiquitous Computing, Distributed Data Storage

1 Introduction

This paper deals with an architecture for spatially distributed storage of contextual configuration information in ubiquitous computing environments. *Ubiquitous Computing* [Mattern 2001; Weiser 1991] aims at providing computers invisibly at all places one may go to such that access to information gets ubiquitous.

In particular, we explore the possibilities of Augmented Reality based applications in ubiquitous computing environments. *Augmented Reality* (AR) is a new technology that seamlessly integrates virtual information in a user's physical environment. Using a Head-Mounted Display (HMD), the user's view of the world is augmented with virtual objects that are spatially fixed in relation to the physical objects. For example, a refrigerator AR application would display virtual objects that depict the content of the refrigerator or objects that are missing and should be refilled on the display, thus giving the impression to the user that the door of the refrigerator is transparent. When the user moves or turns his head, the virtual objects are displayed so that the user has the impression that they are "attached" to the refrigerator. Hence, one of the core requirements of AR is the correct three-dimensional registration of the user's viewing direction and all relevant objects in real time.

Many AR applications handle registration by a combination of several trackers, each specialized for different situations (e.g., tracking the user outdoors, tracking the general location of the user indoors, precise tracking of the user's head motion) [Klinker et al. 2000]. All these devices need to be configured. *Configuration data* is data that a generic software or hardware device needs to work correctly in a ubiquitous computing environment. Hence, an optical tracker worn by a mobile user needs descriptions of markers in the current room to correctly compute the room's position relative to the user; a tracker mounted in a room needs information about the properties of a marker on the user to register the user's position relative to the room. Moreover, many ubiquitous computing applications have the requirement to accomodate arbitrary *mobile clients* potentially composed of multiple, configurable devices.

The usual approach of storing all configuration data in a central database turns the environment into a complex monolith that is hard to understand for both developers and users. A strictly central data structure is contradictory to the intention of ubiquitous computing to provide usually simple applications that are specifically tailored to the who, where, when and how. In this paper, we propose a highly distributed architecture that stores the configuration data where it belongs to. We think that every device in a ubiquitous computing environment should be responsible for its own configuration. This yields more flexible applications in context aware environments, as the environment can be partially reconfigured dynamically and with minimal side effects to the rest of the system. By this assumption, the architecture we propose naturally allows to incorporate arbitrary new mobile clients worn by users that come to ubiquitous computing environments they have never been to before.

Although this architecture results from the need to configure mobile tracking devices, it is suitable to store all types of configuration information that have a clear mapping on spatial entities. These entities may be rooms as well as mobile users carrying their own information with them on their *mobile clients*.

This paper is structured as follows. Section 2 illustrates the problem using a visionary scenario. Section 3 derives requirements for our architecture from this. Section 4 presents the core concepts of the DWARF framework that allow us to implement a distributed configuration. The key ideas of this architecture are described in section 5. We implemented the scenario with a mobile marker-based tracker and describe some details of this implementation in section 6. Section 7 discusses work related to ours. Section 8 concludes this paper with proposals for future enhancements.

2 An Example Scenario

The following scenario illustrates the typical features of a ubiquitous computing environment.

A user, Joe, walks around an intelligent building (see figure 1). The building is equipped with many sensors registering what happens inside. All sensors are integrated in a ubiquitous computing environment providing access to information and applications at

*e-mail: {wagnerm,klinker}@in.tum.de

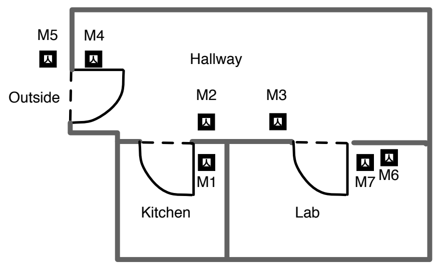

Figure 1: Example scenario.

every point in the building. Most users of the building wear mobile clients, a set of computing devices having widely varying functionality and configuration, depending on the user's personal preferences. Joe is wearing a mobile Augmented Reality system consisting of computers that access the environment using a Wireless Local Area Network (WLan), a head mounted display, an input device on his hand and a camera that detects markers for tracking Joe's current position. A prototype of this mobile client is depicted in figure 5.

Many applications are available in the building, and most are tied to specific rooms. For example, the refrigerator in the kitchen has a virtually transparent door, the table in the AR lab is augmented with shared views of three-dimensional construction plans Joe is currently working on. The AR system enables him to collaborate on these with some remote colleagues.The hallway provides short speech output about what is behind all the doors. To execute the matching applications at every point in time, the mobile client has to know which room Joe is currently in. This is handled by optical tracking software analyzing the video stream from Joe's camera. For example, the inner side of the kitchen door carries a marker with the semantics "User leaving the kitchen". We do not assume that the mobile client has any knowledge about the building, as we would like to incorporate arbitrary new users. As such, our environment provides the semantical information correlating with certain markers automatically to the mobile clients.

Let us assume Joe is in the hallway. His mobile client has a service that can play arbitrary sounds and gets therefore connected to the hallway's "What's behind the door?" application. Whenever either the environment or Joe's mobile client realize that he is standing in front of a door, the application gets notified and sends an audio stream to Joe's mobile client. If Joe now leaves the hallway and goes to the lab, his audio service gets disconnected from the hallway application and connected to the collaboration program in the lab. By this dynamic reconfiguration, Joe is now able to listen to his colleagues' comments on his work using the same mobile client services as in the hallway. If Joe had another service allowing streaming video in his mobile client, he could see his colleagues as well.

3 Requirements for Distributed Spatial Configuration

Ubiquitous computing environments are typically made out of many small applications using a small number of basic components and a large number of users and sensors. Both applications and sensors need to be partially reconfigured very often, as the environment serves many users with different background, preferences, access rights and capabilities of their mobile clients. The simple approach of using a central static configuration database for our setup has several major drawbacks. On the user level, the whole environment would have to be treated as a single complex application, making it hard for both developers and end users to understand the effects of changes in the configuration. This may easily break existing applications if the central system is adapted to new applications. Moreover, the centralized approach introduces a single point of failure.

Configuration information that only is of interest in certain spatial regions is organized best according to these regions rather than in a centralized structure. This facilitates keeping the data up to date and allows simple partial reconfiguration. In addition, it seems reasonable to let every user with a mobile client store all configuration information that the environment needs to correctly communicate with the user at this mobile client. This strategy allows users to dynamically change their clients, depending on personal preferences and the availability of hardware or software components, without the need to tell the ubiquitous environment of these changes before actually using the reconfigured client. Our architecture supports spatial distribution of configuration data and automatic reconfiguration of both the environment and the mobile clients.

This leads to the following requirements on a distributed configuration architecture.

Context aware configuration data. In our work, we follow Dey's [Dey and Abowd 2000] separation of context in *location, identity, activity* and *time*. Our architecture should support configuration based on all these contextual attributes. Location is obviously handled by the spatial distribution, as is partially the identity of the user. Some configuration may depend on the current time of day, for example, an optical tracker may have some parameters telling it the current lighting situation. Finally, the user's current activity, in our example the application he currently uses, may influence the choice of tracking accuracy.

Transparent access to configuration data. Components that need to be configured should not have to bother about where to get the necessary data. The architecture should support dynamic reconfiguration based on the current context. This should happen transparently to the components getting configured. The transparency may be used to centralize some logically separate configuration information on a single database server for security or reliability reasons.

Separate context estimation component. If the mobile client and the ubiquitous infrastructure have to work together to provide its users full access to all information available, they need to communicate using clear protocols. This holds especially true for those parts of the system that derive the current context out of sensor data. For example, an optical tracker has to get information about its environment to determine the user's current location. If the environment is equipped with new trackable features that can only be handled by an algorithm not available to the mobile client beforehands, we either have to implement this algorithm at the mobile client, dynamically load executable code to the mobile client or have to send the mobile client's video image to a stationary compute server. In all three cases, we must encapsulate the tracking algorithm in a component that takes video images and provides location information. The same holds true for all other components deriving context of the current situation.

4 DWARF

The *Distributed Wearable Augmented Reality Framework* [Bauer et al. 2002b] is a research platform that explores the possibilities of Augmented Reality based applications in distributed ubiquitous computing environments. The framework contains services for tracking, visualization of three-dimensional data, calibration of objects, multimodal input, and modeling of user tasks. Several systems have been built so far [Bauer et al. 2001; Echtler et al. 2002; Klinker et al. 2002; MacWilliams et al. 2003], consisting of between 10 and 50 services.

The DWARF services are accessed through a CORBA-based middleware. On each network node of a DWARF system, there is one *service manager*. There is no overall central component. The service manager controls its local services and maintains descriptions of them. Each service manager cooperates with the others in the network to set up connections between services. A DWARF *service* is the basic building block of a running system. It either encapsulates a hardware device like a tracker, performs some reusable functionality like controlling a taskflow or handles some application-specific task. Each service is running within a separate operating system process or thread. Its functionality is described in terms of *abilities*, the functionality it requires from other services is described in terms of *needs*. Needs and abilities are matched using *connectors*.

An *ability* is the abstract description of a service's functionality, e.g. location data for optical trackers. A service can have multiple abilities, e.g. the tracker may track multiple objects simultaneously. Abilities are typed, an optical tracker delivers `PoseData` for location information.

A *need* describes the functionality a service requires from its counterparts to be able to work. Again, a service may have multiple needs. The optical tracker needs a stream of video data and descriptions of markers to be able to find these markers in the video image. Needs are typed, too, and a need can only be satisfied by an ability of the same type.

A *connector* is a description of the communication protocol, e.g. shared memory for video data, CORBA object references or CORBA notification events for event-based communication of location data.

Attributes enhance the description of an ability. The optical tracker may therefore specify which object it tracks, using e.g. `Thing=UserHead`. Needs can be refined using *predicates*, e.g. `(&(User=Joe)(Room=Lab))`. When matching needs with abilities, the service managers ensure that the ability's attributes satisfy the need's predicate. Attributes may be specified for the entire service, in this case all abilities of that service have these attributes.

```
<service name="OpticalTracker">
    <need name="video" type="VideoStream">
        <connector protocol="SharedMemory"/>
    </need>
    <need name="marker" type="MarkerData"
          predicate="(&(Room=*)(User=*))">
        <connector protocol="ObjectReference"/>
    </need>
    <ability name="poseData" type="PoseData"
             isTemplate="true">
        <attribute name="Room"
                   value="$(markerData.Room)">
        <attribute name="User"
                   value="$(markerData.User)">
        <connector protocol="NotificationPush"/>
    </ability>
</service>
```

Figure 2: Sample XML description of an optical tracker service having two needs of type *VideoStream* and *MarkerData* and an ability of type *PoseData*. If the need for *MarkerData* gets satisfied by another service with attributes *Room* and *User*, the optical tracker offers the *PoseData* ability with the same attributes.

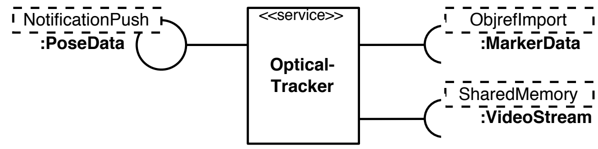

Figure 3: Sample UML description of an optical tracker service. Needs are depicted as semicircles, abilities as circles.

Each DWARF service installed on a system may either describe itself at startup to the service manager or use an XML file format to give the service manager the possibility to start and stop the service on demand. Our example of an optical tracker is shown in figure 2. A UML-based notation is shown in figure 3.

MacWilliams and Reicher describe in [MacWilliams and Reicher 2003] how a service's ability may change at runtime according to how its needs are satisfied. In our example, if and only if the optical tracker gets `MarkerData` for user *Joe* and the room *Lab*, it can provide an ability of type *PoseData* with the attributes `User=Joe` and `Room=Lab`. As such, the optical tracker has a *template* ability of type `PoseData` that can run in multiple instances depending on the available configurations.

5 Automatic Context Aware Configuration

The DWARF framework provides the basic building blocks for our distributed configuration architecture. In this section, we describe this architecture based on a simple example setup consisting of the following components. A UML description of the setup is shown in figure 4. This rather simple setup is prototypical for most ubiquitous computing applications.

Video Grabber. A digital camera is read out and the resulting video image is sent to other components.

Optical Tracker. This component takes a video image and marker descriptions detailing the marker's appearance as well as their 3D location in a room. It then finds the marker in the image and reconstructs the camera's current position [Tsai 1987]. In our setup, the user is wearing the camera, therefore the tracker yields the user's current position. Finally, this component has the need for contextual changes and can therefore be told the current context.

Application. There is one instance of this component for each application, such as the transparent refrigerator door discussed in section 2. Applications may either run on the user's mobile client or in the ubiquitous environment. A typical application takes the location information from the tracker and displays augmentations to the user's view using the mobile client's rendering component [MacWilliams et al. 2003].

Context Estimation. This component analyzes the location data from the tracker and triggers context switches whenever it concludes that a user has moved to a new room. Note that this component may exist in multiple instances, each specifically adopted to a certain contextual situation.

Configuration Data. This component actually stores configuration data. There is one instance of this component for each contextual state. As the marker descriptions of the tracker are nothing more than configuration information, it is this component that reconfigures the optical tracker.

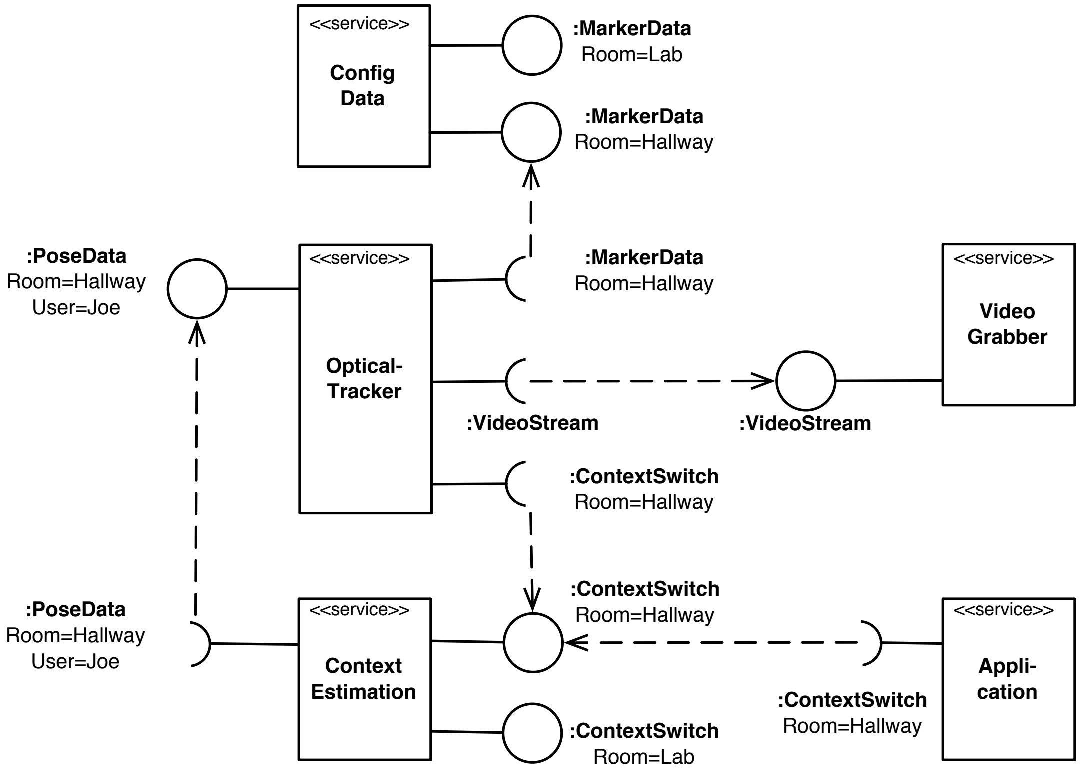

Figure 4: UML model of the mobile setup. Needs are depicted as semicircles, abilities as circles. Note that needs may satisfy two other service's abilities (*ContextSwitch*). Both the *ContextEstimation* and the *ConfigData* services offer abilities for two contextual situations.

Context is modeled using DWARF attributes and predicates. According to [Dey and Abowd 2000], we split context in *identity, location, activity* and *time*. We assume that *Joe* is moving around the *Lab* at *9 a.m.* doing *NothingSpecial*. As such, the optical tracker's need for marker descriptions should have the predicate `(Room=Lab)` and its ability to send location data should have the attributes `User=Joe`, `Room=Lab` and `Time=9am`. Our optical tracker does not care about Joe's current activity, so we leave out this attribute.

Let us now discuss how the automatic configuration proceeds. We assume the user starting in the *Hallway* and moving to the *Lab* (see floor plan in figure 1). The transition from the contextual state `Room=Hallway` to `Room=Lab` is triggered by the marker *M3*, therefore the optical tracker is configured initially to recognize marker *M3* (and some others). Once the user is in the lab, the optical tracker has to be configured in a way that allows it to recognize markers *M6* and *M7* in order to detect other context switches. The flow of events is as follows:

1. The user enters the building's hallway with his mobile setup. He starts the *Video Grabber*, *Application* and *Optical Tracker* components. This may also be performed automatically (see [MacWilliams and Reicher 2003]).

2. Initially, the optical tracker's need for marker data has the predicate `(Room=Hallway)`. As there exists a *Configuration Data* service with a matching attribute, the DWARF service manager connects both. The configuration service gives the information necessary to detect markers *M2*, *M3* and *M4* to the optical tracker.

3. The service manager recognizes that there exists a *Context Estimation* component that has the need for marker data with the predicate `(Room=Hallway)` and the ability to trigger contextual changes, attributed again by `Room=Hallway`. The component matches the optical tracker in its current state and gets therefore connected.

4. The user now moves towards the door and has marker *M3* for quite a while in his camera's viewing frustum. This information is sent to the context estimation that concludes that the user must have left the hallway and went to the lab. It therefore tells the optical tracker (and all other components of the mobile setup that may be interested) that there was a change in context.

5. The optical tracker service changes its service description at the service manager, setting the marker data need's predicate to `(Room=Lab)`.

6. The service manager disconnects the hallway's configuration and context estimation services from the optical tracker. In consequence, the optical tracker unloads all configuration data it received from these services. As there is a new configuration service available in the lab with an attribute `Room=Lab`, the optical tracker is connected with the new configuration and context estimation services.

7. Again, the optical tracker gets information about markers, this time *M6* and *M7*, and outputs location information to the lab's context estimation component until the latter concludes that the user left the room again.

Up to now, we always assumed that the user has an initial contextual state. This state is either provided by explicit user input, i.e. "I am at the TUM campus." or by using the same configuration mechanisms with special attributes.

For each network broadcast area, there must be one configuration and context estimation service that provides the initial state. For example, if a user walks towards a campus building and enters its WLan area, the current `Room` predicate should be set to `(Room=global)`. The campus building may now provide a context estimation service with matching attributes that interprets the user's current GPS information in order to trigger contextual switches at the user's mobile client, such as setting the `Room` attribute to `Room=FMIBuilding`.

6 Implementation Status

The architecture just described was implemented within a larger project, ARCHIE (Augmented Reality Collaborative Home Improvement Environment), evaluating some new AR technologies for collaborative support of architectural planning. Further information about ARCHIE can be found on our web site[1].

The system runs on several stationary and mobile computers running Linux (see figure 5). The components are implemented in C++ and Java. We use an iBot IEEE 1394 camera that delivers its data via the *libdc*[2] library. The optical tracker uses the AR Toolkit library[3], an easy to use marker-based tracking library. More information on how we incorporate AR Toolkit in our setup can be found in [Wagner 2003]. The application consists of a simple *Speaker* service that plays prerecorded texts matching the current location. Context estimation is rather simplistic, once the system realizes that a single or multiple markers have been "seen" several times by the user's camera, a context switch is triggered. As such, our system allows the user to implicitly change the context by moving to another room. The configuration data is stored by a service wrapping a MySQL database offering separate abilities for every configuration context stored. We successfully demonstrated the context aware configuration architecture incorporating multiple instances of the *Context Estimation* and *Configuration Data* services presented in this paper.

7 Related Work

There exist several systems supporting ubiquitous computing environments. The Gaia project [Hess and Campbell 2003; Román et al. 2002] introduced the concept of *Active Spaces*, ubiquitous computing environments similar to our spatial entities. It proposes a context aware filesystem that stores the user's data. In contrast, our system stores the user's data on the user's mobile client. Project Aura [Garlan et al. 2002] uses the coda file system [Satyanarayanan 2002] to allow a mobile user nomadic file access. Again, the data storage solution is central. The Ninja system [Gribble et al. 2001] uses distributed data structures to provide the high load that a central storage solution has to handle. Riché and Brebner propose a user-centric replication mechanism [Riché and Brebner 2003] to speed up access to contextual information, however, this approach is based on a centralized data storage as well. The NEXUS project [Hohl et al. 1999] aims at providing an open platform for context-aware applications with a special focus on location. The CANU subproject [Bauer et al. 2002a] proposes to obtain model data by the next network node in a mobile ad-hoc network and is therefore similar to our basic assumption of spatially organized data. However, the spatial model is still represented as a single central graph and does not support the separation of context estimation according to the current spatial situation.

The configuration problem of trackers with large tracking areas has been treated by Reitmayr and Schmalstieg using an XML based configuration framework called OpenTracker [Reitmayr and Schmalstieg 2001], later this work was extended to allow reuse of configuration information [Kalkusch et al. 2002]. Although using OpenTracker is an easy way of configuring setups of many tracking devices, the framework does not support dynamic reconfiguration of trackers. The Bat system [Addlesee et al. 2001] provides building wide location tracking, but suffers from a central data storage, not allowing dynamic addition of mobile clients.

The configuration data in our architecture is structured along the four dimensions of context defined by Dey [Dey and Abowd 2000]. Lieberman and Selker [Lieberman and Selker 2000] point out that context aware applications can simplify the interaction with computers without reducing functionality. This simplification can be done via automatic configuration as discussed in this paper. The GUIDE Project's data storage architecture [Efstratiou et al. 2001] discusses strategies to resolve ambiguities in contextual information that could be incorporated in our architecture, whereas Dearle et al. propose an architecture for global smart spaces [Dearle et al. 2003], leading to universally available information that may be enhanced locally by ubiquitous computing environments as discussed in this paper.

8 Conclusion and Future Work

In this paper, we presented an architecture for distributed spatial configuration of context aware applications that allows spatial distribution of data, separation of configuration data according to its contextual use, transparent access to the data and an encapsulation of algorithms estimating the context. This architecture is built on the DWARF framework, thus incorporating a wide array of already existing other components for location tracking, 3D rendering and modeling of data necessary for Augmented Reality.

Although we discussed our concepts only for a rather simple marker-based tracking setup, we think it is well suited for all mobile systems acting in ubiquitous environments. However, clear interfaces need to be defined to allow flexible yet simple configuration for other areas than tracking.

While the ideas presented in this paper allow the spatial separation of data, it is left to the application to define spatial entities. In our setups single rooms proved well, for other applications we might have to choose larger or smaller contextual entities. An interesting question is how to structure context in a way that allows a "natural" design of new applications. Problems arising include how to provide architectural support for restructuring legacy information and whether it is possible to let the system learn context boundaries automatically. A similar area of future work lies in developing concepts on how to actually store the spatially organized data. For security or reliability reasons, it might make sense to store all spatially organized configuration data of a building in a single database system, if the network infrastructure in this building is dense and reliable.

Up to now, we have not systematically investigated security and privacy issues, although storing all user data at the user's mobile client will serve as a good starting point for implementing privacy policies.

Finally, we might incorporate some of the storage facilities discussed in related work in order to allow efficient access not only to local but to global data as well. In our concept, global data is a context device that has no `Room` attribute and is therefore available

[1] `http://www.augmentedreality.de`

[2] `http://sourceforge.net/projects/libdc1394/`

[3] `http://www.hitl.washington.edu/research/shared_space/download/`

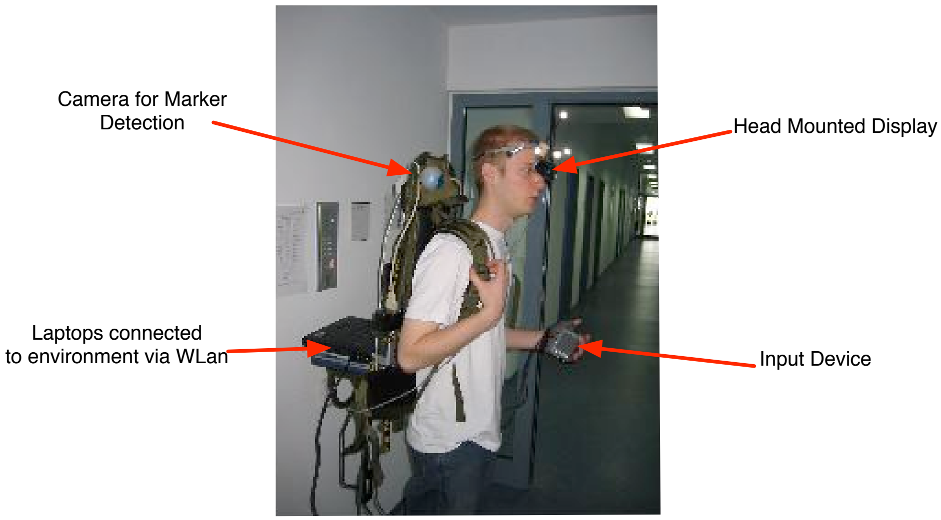

Figure 5: The mobile client.

everywhere. To access such information efficiently, caching and prefetching services should be added.

Acknowledgments

The work described in this paper was partially supported by the High-Tech-Offensive of the Bayerische Staatskanzlei.

The authors would like to thank all people involved in the ARCHIE project, in particular Felix Löw and Marcus Tönnis, and all members of the DWARF project for their collaboration on the framework. Special thanks to Allen Dutoit, Thomas Reicher and Christian Sandor for helpful comments and valuable dicussions.

References

ADDLESEE, M., CURWEN, R., HODGES, S., NEWMAN, J., STEGGLES, P., AND WARD, A. 2001. Implementing a Sentient Computing System. *IEEE Computer* (August).

BAUER, M., BRUEGGE, B., KLINKER, G., MACWILLIAMS, A., REICHER, T., SANDOR, C., AND WAGNER, M. 2001. Design of a Component-Based Augmented Reality Framework. In *Proceedings of the International Symposium on Augmented Reality*.

BAUER, M., BECKER, C., AND ROTHERMEL, K. 2002. Location Models from the Perspective of Context-Aware Applications and Mobile Ad Hoc Networks. *Personal and Ubiquitous Computing 6*, 5–6 (December), 322–328.

BAUER, M., BRUEGGE, B., KLINKER, G., MACWILLIAMS, A., REICHER, T., SANDOR, C., AND WAGNER, M. 2002. An Architecture Concept for Ubiquitous Computing Aware Wearable Computers. In *International Workshop on Smart Appliances and Wearable Computing*.

DEARLE, A., KIRBY, G., MORRISON, R., MCCARTHY, A., MULLEN, K., YANG, Y., CONNOR, R., WELEN, P., AND WILSON, A. 2003. Architectural Support for Global Smart Spaces. In *Proceedings of International Conference on Mobile Data Management*.

DEY, A. K., AND ABOWD, G. D. 2000. Towards a better understanding of context and context-awareness. In *Workshop on the What, Who, Where and How of Context-Awareness, affiliated with CHI 2000*.

ECHTLER, F., NAJAFI, H., AND KLINKER, G. 2002. FixIt. In *Demonstration at the International Symposium on Augmented and Mixed Reality (ISMAR 2002)*.

EFSTRATIOU, C., CHEVERST, K., DAVIES, N., AND FRIDAY, A. 2001. An Architecture for the Effective Support of Adaptive Context-Aware Applications. In *Proceedings of International Conference on Mobile Data Management (MDM 2001)*, Springer, K.-L. Tan et al., Eds., vol. 1987 of *LNCS*, 15–16.

GARLAN, D., SIEWIOREK, D., SMAILAGIC, A., AND STEENKISTE, P. 2002. Project Aura: Toward Distraction-Free Pervasive Computing. *IEEE Pervasive Computing 1*, 2.

GRIBBLE, S. D., WELSH, M., VON BEHREN, J. R., BREWER, E. A., CULLER, D. E., BORISOV, N., CZERWINSKI, S. E., GUMMADI, R., HILL, J. R., JOSEPH, A. D., KATZ, R. H., MAO, Z. M., ROSS, S., AND ZHAO, B. Y. 2001. The Ninja Architecture for Robust Internet-Scale Systems and Services. *Computer Networks 35*, 4, 473–497.

HESS, C. K., AND CAMPBELL, R. H. 2003. A Context-Aware Data Management System for Ubiquitous Computing Applications. In *Proceedings of the 4th International Conference on Mobile Data Management*.

HOHL, F., KUBACH, U., LEONHARDI, A., ROTHERMEL, K., AND SCHWEHM, M. 1999. Next century challenges: NEXUS – an open global infrastructure for spatial-aware applications. In *Proceedings of the Fifth Annual International Conference on Mobile Computing and Networking (MobiCom)*.

KALKUSCH, M., LIDY, T., KNAPP, M., REITMAYR, G., KAUFMANN, H., AND SCHMALSTIEG, D. 2002. Structured Visual Markers for Indoor Pathfinding. In *The First IEEE International Augmented Reality Toolkit Workshop*.

KLINKER, G., REICHER, T., AND BRUEGGE, B. 2000. Distributed User Tracking Concepts for Augmented Reality Applications. In *Proceedings of the International Symposium on Augmented Reality*.

KLINKER, G., DUTOIT, A., BAUER, M., BAYER, J., NOVAK, V., AND MATZKE, D. 2002. Fata Morgana – A Presentation System for Product Design. In *International Symposium on Aumgented and Mixed Reality ISMAR 2002*.

LIEBERMAN, H., AND SELKER, T. 2000. Out of context: Computer systems that adapt to, and learn from, context. *IBM Systems Journal 39*, 3&4.

MACWILLIAMS, A., AND REICHER, T. 2003. Decentralized Coordination of Distributed Interdependent Services. *IEEE Distributed Systems Online* (June). Accepted for publication as Middleware Works in Progress Paper.

MACWILLIAMS, A., SANDOR, C., WAGNER, M., BAUER, M., KLINKER, G., AND BRUEGGE, B. 2003. Herding Sheep: Live System Development for Distributed Augmented Reality. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*.

MATTERN, F. 2001. The Vision and Technical Foundations of Ubiquitous Computing. *Upgrade 2*, 5 (October), 2–6.

REITMAYR, G., AND SCHMALSTIEG, D. 2001. OpenTracker – An Open Software Architecture for Reconfigurable Tracking based on XML. In *Proceedings of the ACM Symposium on Virtual Reality Software & Technology (VRST)*.

RICHÉ, S., AND BREBNER, G. 2003. Storing and Accessing User Context. In *Proceedings of the 4th International Conference on Mobile Data Management*.

ROMÁN, M., HESS, C. K., CERQUEIRA, R., RANGANATHAN, A., CAMPBELL, R., AND NAHRSTEDT, K. 2002. Gaia: A Middleware Infrastructure to Enable Active Spaces. *IEEE Pervasive Computing 1*, 4, 74–83.

SATYANARAYANAN, M. 2002. The evolution of Coda. *ACM Transactions on Computer Systems 20*, 2 (May), 85–124.

TSAI, R. 1987. A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses. *IEEE Journal on Robotics and Automation 3*, 323–344.

WAGNER, M. 2003. Configuration Strategies of an AR Toolkit-based Wide Area Tracker. In *Proceedings of The Second IEEE International Augmented Reality Toolkit Workshop*.

WEISER, M. 1991. The computer of the twenty-first century. *Scientific American* (Sep.), 94–100.