Providing Service in a Changing Ubiquitous Computing Environment

Simon Schubiger-Banz, Sergio Maffioletti, Beat Hirsbrunner, Amine Tafat Bouzid *
IIUF -- Institute for Informatics
University of Fribourg
Chemin du Musée 3
1700 Fribourg, Switzerland
*{Simon.Scubiger|Sergio.Maffioletti|Beat.Hirsbrunner|Amine.Tafat.Bouzid}@unifr.ch

Abstract

Ubiquitous computing (UbiComp) is a highly dynamic and heterogeneous environment. These aspects increase the complexity of the development of UbiComp applications. We are currently working on an UbiComp middleware that aims at shielding the application developer from these aspects. This paper presents the design criteria for an UbiComp middleware as well as its classifiers that interface an application with a changing context (the logical and physical environment). The applications based on our middleware state their resource requirements not in terms of specific instances but through concepts that are part of the application's ontology. Classifiers that also provide the information to perform context dependent service instantiation in a process called "addressing by concept" give concept semantics.

Keywords: Ubiquitous Computing, Middleware, Interactive Environment, Addressing by Concept, Knowledge Representation, Classifier System, Resource Classification, Ontology.

1 Introduction

Research in ubiquitous computing (UbiComp) is towards the development of an application environment able to deal with the mobility and interactions of both users and devices. The vision of UbiComp relies on the presence of environments enriched by computers embedded in everyday objects (blackboards, table, walls, etc.) and by sensors able to acquire information from the context. An UbiComp application is described by a set of services and is realized by the interaction between a group of devices called a federation. Every service may be associated to a specific device that supplies its implementation. UbiComp applications usually depend on resources provided by the context. Therefore, describing these resources is an important issue. The classical approach is standardizing interfaces for resource access and selection. Unfortunately, standardization is slow, usually involves several compromises and the resulting specifications are seldom open enough to allow innovation.

Instead of enforcing a standardized view for the application, we propose a middleware that decouples the high-level concepts (abstractions) from the instances implemented by each context. The concept "nearest printer" [1] for instance may be used no matter how a context supplies the corresponding implementation.

Every UbiComp application relying on our middleware will provide its own view of the world through *ontology*. Ontology is basically a collection of *concepts* an application depends on. This means that an application expresses its resource requirements in terms of its concepts instead of addressing specific resources directly for example by an URL.

2 Interactive Environments

It is an important requirement for UbiComp applications to provide an environment in which specialized computing instruments[1] can be accommodated and integrated into existing application contexts. Interactive environments are introduced to formalize the base execution level for the abstraction layers UbiComp applications will rely on. They model physical regions of the world enriched by computing instruments.

[1] With "computing instrument" we refer to both devices and sensors. It identifies the abstract idea of computers integrated in our everyday environment.

They have a large number of hardware and software components that need to cooperate; they tend to be highly dynamic and require reconfiguration and resource management on the fly as their components and inhabitants change and as they adjust their operation to suit the learned presence of their user.
Intelligent Environments defined in the Metaglue [6] project are an example.

3 Common Middleware

The few existing integrated multi-device computer environments today tend to be highly specialized and based on application-specific software. Applications developed for interactive environments should be able to interconnect and manage large numbers of disparate hardware and software components.
They should operate in real-time; dynamically add and remove components to a running system without interrupting its operation; control allocation of resources; and provide a means to capture persistent state information.
In order to model applications in this domain we need to define a common design methodology based on new paradigms independents from the technology. We are investigating a model to abstract the main components of an UbiComp system in order to formalize the development of Interactive Environment applications.
Thank to these abstractions our middleware will present a uniform access abstraction for different ubiquitous devices, allowing them to interact and cooperate. This allows us to write applications scaling both on services offered and on devices composing the system.
Existing projects like Oxygen [5], Nexus [3], Beach [7] and Metaglue [6] address the main functionalities of a UbiComp middleware but do not consider higher level service classification, so applications can not rely on a suitable abstraction layer for describing their functionalities.
Our approach allows applications to define their own ontology for the resources they may offer and require. These high level concepts will be instantiated with implementations depending on each application context. This allows both the application and the instruments to use a high-level service description for interaction. In such a way each computing instrument is also able to roam from one application context to another (even if the other is using another ontology) without changing its service description.

4 UBIDEV: The Software Abstraction

Any computing instruments should be able to communicate with any other instrument no matter of its origin or manufacturing.
The concept UBIDEV has been introduced to describe the computing instruments abstraction in an application context where different devices have to interact with each other supplying a group of coordinated services. Thanks to this abstraction we can define a generic interaction scheme for UbiComp applications.
The idea behind UBIDEV is to provide a common framework, which allow existing computing instruments to cooperate and carry out their particular information, data and processing tasks. UBIDEV presents an abstract reference model to describe computing instruments interactions. It divides the functions into distinct yet connected layers. The model is divided in five levels of interaction whose functionalities are similar to those of the Open System Interconnect (OSI) [4] model.

UBIDEV Layers

The UBIDEV layers have been conceived to specify the sequence of interaction each computing instrument has to follow when belonging to a specific application context.

- **The medium** layer is concerned with all the physical communication capabilities of the device. It specifies the medium used by the device in order to exchange information with the application. The heterogeneity of devices and networks asks for an integrated, seamless communication framework.

- **The data management** layer takes care of all constrains in a communication abstraction (error-free transfer, logical to physical address mapping, etc.). This layer should ensure reliable communication connections when instruments communicate within an application context and across two different application contexts.

- **The federation management** layer allows session establishment in the application context between the device and the existing federation. This layer supplies the suitable abstraction for describing software entities able to roam from one application context to another.

- **The service** layer describes how services may be loaded, discarded and organized. It also allows the devices to specify and classify the services they are able to supply to the application. According to the specifications in this layer the middleware may choose to delegate to a specific device one or more services according with its capabilities.

- **The application** layer represents the user level abstraction; it is concerned with the user interface for the specific application context.

5 Ontologies and Concepts

UbiComp applications as well as the different UBIDEV layers express their resource requirements in terms of concepts that are matched with services available in the current application context.
For that purpose, each application describes its view of the world by ontology. Figure 1 gives an overview of the interaction of the different parts.

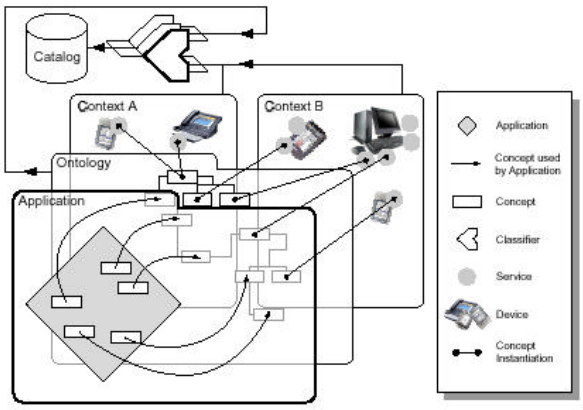

Figure 1: An overview of the relations between application, ontology, services and devices. An application states its requirements in terms of concepts. Concepts are organized in ontology. Classifiers analyse the context and store the device and service classification in a catalogue. The catalogue is used to associate concepts with services during concept instantiation.

The following informal definition of ontology will be used throughout this paper that is more directed towards a usage for UBIDEV:

Definition 1 *An* ontology *is an agreed vocabulary of common terms and their meaning within a context of communicating entities.*

In a more practical sense, entities that agree with an ontology, will use the same terms for making assertions and queries with respect to the same meanings of the terms.
This definition opens several questions that are addressed in the following sections. According to definition 1, an ontology is a vocabulary of terms.
A *concept* has a semantic that is given by a classifier. Classifiers are usually built on top of existing services; for example a c-compiler is a good classifier for resources representing c-programs. This means that a classifier will associate the concept "c-program" to a resource if it can be compiled by a c-compiler. The semantic of the concept "c-program" is given implicitly by the c-compiler used for the classification by the classifier. One may read this implicit definition of the concept "c-program" as "a resource is an instance of the concept *c-program* if and only if a compilation with a c-compiler is successful."
A resource is said to be an *instance* of a concept if a classifier associated that concept with the resource in question.

6 Semantics of Concepts

In order to give meaning to a *concept*, terms either state explicitly the relation of that concept with other concepts or meaning is implicitly given by *classifiers*[2]. Classifiers are services[3] that given a resource and an ontology, output concepts of that ontology. This basically means that a classifier associates one or more concepts it knows about with a resource, therefore marking the resource as an instance of these concepts.

[2] The use of classifiers in the knowledge representation domain is not new. For example the LOOM System [2] includes a classifier but for a different reason and there is only *one* single classifier in contrast to the multiple classifier approach presented here.
[3] Classifiers are considered a special kind of resources; these are JAVA classes in the current prototype that implement the "Classifier" interface.

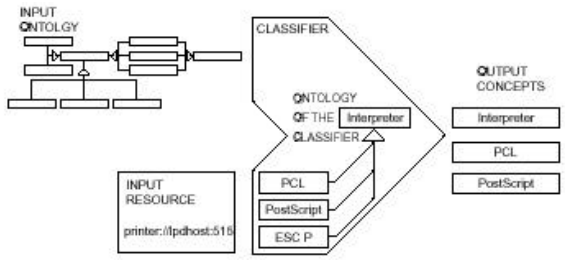

Figure 2: An example of a classifier

Figure 2 gives an example of a classifier that classifies different pages description languages.
The classifier takes an ontology and a resource as input. The input ontology is either a superset of the classifiers ontology or contains concepts that are equivalent with the classifiers concepts. The classifier outputs zero or more concepts of the input ontology that the resource is an instance of. The classifier limits its output to concepts defined in the input ontology because other concepts, even if known to the classifier, will not make any sense for the application providing the input ontology[4]. In the example depicted in figure 2, the printer is an instance of the concepts "Interpreter", "PCL" and "PostScript" but not "ESC P". Classifiers have their own ontology, usually the domain they are able to classify. Classifications of resources are stored and used as a cache when an instance of a concept is requested. The process of requesting an instance of a concept is called "addressing by concept" because the instance is referred to by a concept instead of specific resource identification such as a memory address, a name or an URL. Addressing by concept occurs at different points in the middleware and the applications.
Even the lowest UBIDEV layers use classifiers and addressing by concept. For example a PDA equipped with an IrDA, a Bluetooth and an Ethernet interface will continually classify its environment by sending polling packets on these interfaces. If a connection over one of these interfaces can be established, the corresponding concepts like the medium and the data layer can be instantiated.
This process continues, classifying for example the protocols used over the Ethernet interface and can go up to instantiating high level concepts such as "Nearest Printer", "Blackboard" in the service layer and "Word Processor" in the application layer.

[4] An application can only refer to concept of its own ontology; only for these concepts the application has classifiers and therefore a semantic. Concepts outside the ontology of the application thus do not have a meaning for the application.

7 Conclusion

In this paper we have presented the general idea of a middleware that provides the basic functionalities for modelling interactive environment applications.
The notion of UBIDEV has been designed for answering the need to provide a unified vision of the different computing functionalities issuing from convergence between information and communication technologies.

References

[1] Alan Kaminskz. Jini Print Service Design. http://developer.jini.org/exchange/users/jpgwg/JiniPrintService/design20000215/index.html, Februarz 2000

[2] David Brill, *Loom Reference Manual, Version 2.0*. University of Southern California, December 1993.

[3] Fritz, Hohl and Uwe Kubach and Alexander Leonhardi and Kurt Rothermel. Next Century Challenges: Nexus – An Open Global Infrastructure for Spacial Aware Applications. ACM, 1999.

[4] J.D. Day and H. Zimmermann. The OSI Reference Model. Volume 71, pages 1334-1340. IEEE, December 1983.

[5] Laboratory for Computer Science and Artificial Intelligence Lab. MIT Project Oxygen. http:// www.oxygen.lcs.mit.edu/, June 2000.

[6] Michael H. Coen and Brenton A. Philips and Nimrod Warshawshy and Luke Weisman and Stephen Peters and Peter Finin. Meeting the Conputational Needs for Intelligent Environments: The Metaglue System. Submitted to MANSE99, 1999.

[7] Norbert A. Streitz and Jörg Geissler and Torsten Holmer. Roomware for Cooperative Buildings: Integrated Design of Architectural Spaces and Information Spaces. In N. A. Streitz, S. Konomi, H. J. Burkhardt, editor, *LNCS 1370, Proceedings of the First International Workshop on Cooperative Buildings*, pages 4-21, Darmstadt, February 1998.