Support for Mobility in Active Spaces[†]

Renato Cerqueira[1‡], Cristina Ururahy[1‡], Christopher Hess[2], Dulcineia Carvalho[2], Manuel Román[2], Noemi Rodriguez[1‡], and Roy Campbell[2]

[1] Department of Computer Science – PUC-Rio
Rio de Janeiro, RJ – Brazil
{rcerq,ururahy,noemi}@inf.puc-rio.br

[2] Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL – USA
{ckhess,dcarvalh,mroman1,rhc}@cs.uiuc.edu

Abstract In this paper, we present an overview of our research project with *GaiaOS*, a middleware operating system that provides a generic computational environment for ubiquitous computing. In addition to an outline of the *GaiaOS* architecture, we describe how we address some mobility issues in this infrastructure.

1 Introduction

We envision a world of mobile users in an unobtrusive ubiquitous computing environment that couples a computational model, digital media, and virtual representations of the physical world. Hundreds of embedded computers support the information and computational needs of each user. Users, applications, and computing devices move. The location of users and devices drives applications and resource management. Users have anytime/anywhere access to information, the network, and computational resources. Within this world, applications that make effective use of resources to support the activities of users must be simple and efficient to construct. Changes to the physical environment alter the computational model and information space of the users. Similarly, changes to the computational model and information space may alter the physical environment. We call such environment an *active space*.

We argue that a computing environment for a physical room, operating theater, building, or city must have some form of operating system that organizes that environment to simplify the management of resources, the coding of applications, the identification and authentication of users, the provision of services, and the reuse of software. An active space is analogous to traditional computing systems; just as a computer is viewed as one object, composed of input/output devices, resources and peripherals, so is an active space.

An active space and its appropriate libraries should run on a ubiquitous infrastructure. A home or office application should be movable from one physical location to another, overlaying the actual physical devices and geometry of the physical space. Users should be able to interact with their office or home whether or not they are physically present. The active space should permit virtualization of resources so that a user may access her home from her laptop, work, or a conference.

However, the heterogeneity, mobility, and sheer number of devices makes the system vastly more complex. Applications may have the choice of a number of input devices, such as mouse, pen, or finger; and output devices, such as monitor, PDA screen, wall-mounted display, or speakers. An operating system for

[†] This research is supported by a grant from the National Science Foundation, NSF CCR 0086094 ITR and NSF 99-72884 EQ.

[‡] Authors on leave at UIUC, supported by grants from CAPES-Brazil and CNPq-Brazil.

such a space must be able to locate the most appropriate device, detect when new devices are spontaneously added to the system, and adapt content when data formats are not compatible with output devices.

Traditional operating systems manage the tasks common to all applications; the same management is necessary for physical spaces. Our research is focused on the study of issues related to the design and implementation of such an operating system and the construction of applications that run in the context of a space. We have developed an operating system, called *GaiaOS*, for physical spaces that creates an active space. *GaiaOS* provides a user-centric and application-oriented computational environment. Section 2 presents an overview of *GaiaOS*. Section 3 describes how we have been working on providing *GaiaOS* with support for mobile users and mobile hosts. Finally, Section 4 offers some concluding remarks.

2 GaiaOS

Several approaches have been proposed for interacting with ubiquitous computing environments that are customized for particular scenarios or targeted towards a specific type of application. We argue for a general model, which exports and coordinates the resources contained in a physical space. *GaiaOS* defines a generic computational environment and converts physical spaces and the ubiquitous computing devices they contain into a programmable computing system or active space [RHR+01].

2.1 Gaia Architecture

GaiaOS is a component based meta-operating system, or middleware operating system [2K 98], that runs on top of existing systems, such as Windows2000, WindowsCE, and Solaris. *GaiaOS* is composed of four main parts. Fundamental to the system is the *Unified Object Bus*, which provides tools to manipulate uniformly heterogeneous components running in the system and is responsible for the life cycle of components. The *Gaia Kernel* includes essential services that implement the core functionality of the system, including entity discovery, an object repository, event distribution, naming, data storage and manipulation, and security. The *Gaia Application Model* defines a standard framework for creating ubiquitous applications, which run in the context of a space. Finally, the *Active Space Execution Environment* comprises the "user level" of our operating system. It consists of all application components that run in the context of a particular active space, but are not part of *GaiaOS*.

2.2 Gaia Kernel Services

The kernel contains the minimum required services to bootstrap *GaiaOS* in any arbitrary space. A fundamental service is the *Event Manager*, which is used to distribute information among components, while maintaining loose coupling. Applications can register to specific event channels to be notified of information or changes in the environment. The *Discovery Service* uses the event manager to track software components, people, and physical entities present in a space.

Entities (i.e., devices, services, applications, and users) currently active in a space are stored in the *Space Repository*. The Space Repository exports an interface to search for specific entities based on constraints, such as location.

Applications access data through the *Data Object Service*, a dynamically typed file system that supports content adaptation, customized data access, and location awareness. Personal storage may reside on remote desktop machines or mobile special-purpose devices. A user may define personal storage that can be incorporated into a space when she is physically present.

Security concerns in *GaiaOS* include authentication, access control, secure dynamic loading of components, secure tracking, and location privacy. In addition to general access control and authentication issues, a user may desire her location and activity be kept private. *GaiaOS* employs an *Authentication Service*, which issues credentials for user identity verification. These credentials enable the *Access Control Service* to provide discretionary, mandatory and role-based access control.

3 Mobility in *GaiaOS*

GaiaOS already offers some support for mobility through its kernel services. For example, the Discovery Service tracks people and physical devices present in a space, and sends notification events to interested components when some entity enters or leaves the space. The Space Repository uses this monitoring mechanism in one of its most basic forms. It uses the events that the Discovery Service produces to keep itself updated with the currently available entities in an active space.

We have also been using this monitoring mechanism to launch different adapting procedures in active spaces, based on which and how many entities and resources are available. For instance, an active space can be used as an office, where all resources can be allocated to only one user. However, if one or more people arrive at this space and start to discuss some work, this space can become a meeting room, where all participants can have access to the available resources. In this section, we describe some extensions to *GaiaOS* to support adaptation to mobile users and mobile hosts.

3.1 Mobile Users

When a new user enters an active space, she wants to have access to the space's resources, what is possible through the input and output devices available in the space. But this user might want to configure the space with her own preferences. In this case, the supporting infrastructure has to provide some mechanism that allows space configuration for a particular user. In fact, the infrastructure should be able to support more than one user configuration at same time, using some policy to solve conflicts between preferences of different users.

The user preferences can be specific to a particular space, or can be shared among different spaces. Therefore, there must be a mechanism that allows different active spaces to have access to the user preferences.

We defined a set of components that provides *GaiaOS* with support for space configuration for incoming users. First, we implemented the *User Proxy* component, which is responsible for holding, in a persistent basis, the location of the user, as well as her personal configuration and any other data she would like to have access from anywhere; and for providing this information to applications and services. The user's personal configuration is composed of a set of preference definitions plus a login script, that defines which applications should be started to set the space to her.

Because the user may not know in advance what kind of configuration and devices she will find in the active space she is entering, she must be able to specify more abstract descriptions of her preferences and scripts. The more abstract the user's definitions are, the more flexibility the active space will have to fulfill her needs.

Since the user should have access to her personal configuration from each of the different active spaces she enters, it is important to have a global proxy locator. We defined the *Provider* service, which works like a naming service, holding references to different user proxies and providing them to other services and client applications.

The *User Proxy* and the *Provider* components are global services; that is, they might not belong to any particular active space. Using these two components, we have developed two new *GaiaOS* components and defined a *login process* for *GaiaOS*. The *GaiaOS*'s login process is depicted in Figure 1. We have implemented a *Login* component for *GaiaOS*, which registers itself to listen to the events that the *Discovery Service* sends to notify applications when users enter or leave the space. In step 1, the *Login* component receives an event from the Discovery Service, meaning that a user has entered the active space. This event must provide the *Login* component with the user id and her Provider address. In step 2, the *Login* component connects to the user's Provider and gets a reference to her user proxy. In step 3, the *Login* component connects to the user proxy and gets the user's personal configuration. In step 4, the *Login* component combines the user configuration with the active space configuration, and sends the resultant configuration to another

component that is responsible for running the login script (this component is called *Shell*). The *Login* component can reject some of the user preferences either because the space does not have the required resources or because of some security restriction. Finally, in step 5, the Shell component configures the active space according to the new user's preferences and launches the applications she needs.

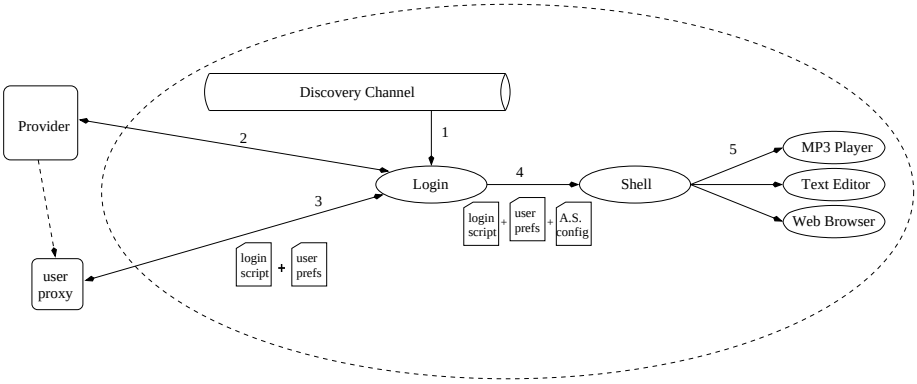

Figure 1. The login process.

3.2 Mobile Hosts

If an incoming user carries a device with her, such as a laptop or a PDA, and this device is using services provided by another active space, she might want to integrate this device with the new active space, reconfiguring the device to use the services available in the new space. Therefore, the applications that are running in the user's device should be able to adapt to the new services, avoiding the reinitialization of the user's environment.

To support this kind of adaptation, we started the development of a mechanism for *GaiaOS* that tracks where the user is, notifies the user's applications when the user moves to a new active space, and launches reconfiguration procedures to adapt these applications to use the services in the new space.

We developed a component, called *Location Notifier*, to track the user location. This component is similar to the *Login* component: It registers itself to listen to the Discovery Service's events, and gets a reference to the proxy of an incoming user, through the user's *Provider*. Then, this component notifies the user proxy that the user entered the new space. Moreover, this component also notifies the user proxy when the user leaves the space.

To notify the user's applications about changes in the user location, and to launch the proper reconfiguration procedures, we have integrated a mechanism that provides support for dynamic adaptation of distributed applications, with *GaiaOS*. This mechanism allows distributed applications to adapt dynamically to nonfunctional properties of their components [MUCR01]. This mechanism, based on the programming language Lua [IFC96,Luaa] and on CORBA, allows applications to select dynamically the components that best suit their requirements, to verify whether the system is satisfying these requirements, and to react, when appropriate, to variations in the nonfunctional properties of the services in use. In the integration of this mechanism with *GaiaOS*, we are exploring the user's location as a nonfunctional property to trigger dynamic adaptation procedures.

4 Final Remarks

In this work, we have used the Lua language [IFC96,Luaa] as a scripting language to define the user's preferences and her login scripts. We have also used the LuaOrb system [CCI99,Luab] to have access to CORBA and COM components from Lua scripts. LuaOrb is a programming tool that implements dynamic bindings between Lua and the systems CORBA, COM, and Java. LuaOrb provides support for dynamic application composition, and for interoperability between different component systems.

The reconfiguration mechanism described in Section 3.2 is also based on the LuaOrb system, and it is restricted to applications that are composed using LuaOrb. Despite this restriction, this mechanism has allowed us to do some experiments with dynamic adaptation based on the user's location.

Up to now, we have implemented some simple applications to validate the support for mobile users and mobile hosts that we are developing to *GaiaOS*, and we still have many mobility issues to investigate in active spaces.

We are planning to extend the mechanisms presented in this paper in many directions. We are studying mechanisms to guarantee the application's consistency when it is adapted from one active space to another. As an evolution of this study, we intend to provide a full support for the architecture for the management of ubiquitous execution environments described in [Car01].

Another on-going study is how to provide higher level programming mechanisms to describe user and space configurations, and reconfiguration procedures. For instance, we want to be able to specify user applications using templates with more abstract component descriptions, based on functional and nonfunctional properties. We are investigating extensions to the Lua language, in order to use Lua as a domain specific language to coordinate active spaces.

References

2K 98. 2K Research Team. 2K: A Component-Based Network-Centric Operating System for the Next Millennium. `http://choices.cs.uiuc.edu/2k`, 1998.

Car01. Dulcineia Carvalho. A framework for the management of ubiquitous execution environments. A Proposal. Department of Computer Science, University of Illinois at Urbana-Champaign, June 2001. `http://devius.cs.uiuc.edu/~dcarvalh/environmentsProposal.ps`.

CCI99. R. Cerqueira, C. Cassino, and R. Ierusalimschy. Dynamic Component Gluing Across Different Componentware Systems. In *International Symposium on Distributed Objects (DOA'99)*, Edinburgh, Scotland, 1999. IEEE Press.

IFC96. R. Ierusalimschy, L. Figueiredo, and W. Celes. Lua - an extensible extension language. *Software: Practice and Experience*, 26(6):635–652, 1996.

Luaa. The Programming Language Lua. `http://www.lua.org/`.

Luab. LuaOrb Online. `http://www.tecgraf.puc-rio.br/luaorb/`.

MUCR01. A. Moura, C. Ururahy, R. Cerqueira, and N. Rodriguez. Dynamic Support for Nonfunctional Requirements in Distributed Applications. Available at `http://www.tecgraf.puc-rio.br/luaorb/`, 2001.

RHR+01. Manuel Romàn, Christopher K. Hess, Anand Ranganathan, Pradeep Madhavarapu, Bhaskar Borthakur, Prashant Viswanathan, Renato Cerqueira, Roy H. Campbell, and M. Dennis Mickunas. GaiaOS: An infrastructure for active spaces. Technical Report UIUCDCS-R-2001-2224 UILU-ENG-2001-1731, University of Illinois at Urbana-Champaign, May 2001.