

A Flexible, Interoperable Framework for Active Spaces*

Fabio Kon[†] Christopher Hess Manuel Román
Roy H. Campbell M. Dennis Mickunas

Department of Computer Science
University of Illinois at Urbana-Champaign
{f-kon,ckhess,mroman1,roy,mickunas}@cs.uiuc.edu
<http://choices.cs.uiuc.edu/ActiveSpaces>

1 Introduction

Mobile computing and active spaces are part of the next revolution in information technology. Mobile computing includes systems as diverse as digital cellular telephony, traffic control and information systems, and simple web browsing via wireless connections. Active spaces consist of physical spaces – such as offices, lecture and meeting rooms, homes, hospitals, campuses, train stations, cities – that are augmented with computing devices integrated into the environment. The objective of these devices is to provide information to and obtain information from users of the space, helping them to perform activities they would not be able to perform otherwise, or helping them to perform conventional activities more easily.

Multimedia interfaces will play a fundamental role in human-computer interaction. Applications will gradually abandon dull interfaces based on text and static graphics and will move towards interfaces with a more dynamic appearance based on audio, video, graphics, and animations. We can also expect the migration of services such as radio and TV to personal computers, PDAs, and video walls. Within one decade, there will probably be no distinction among the telephone, radio, TV, and data networks. In the future, there will be no distinction between a computer and dedicated radio and HDTV receivers. Everything will be integrated into an extension of today's Internet. The dynamic nature of multimedia, with its fluctuating, time-sensitive resource requirements, will contribute to the changes towards more dynamic computing environments.

Until recently, these highly-dynamic environments with mobile computers, active spaces, and ubiquitous multimedia were only present in science fiction stories or in the minds of visionary scientists like Mark Weiser [Wei92, WSA⁺95]. But now, they are becoming a reality and one of the most important challenges they pose is the proper management of *dynamism*. Future computer systems must be able to configure themselves dynamically, adapting to the environment in which they are executing. Furthermore, they must be able to react to changes in the environment by dynamically reconfiguring themselves to keep functioning with good performance,

*This research is supported by the National Science Foundation, grants 98-70736, 99-70139, and EIA99-72884EQ.

[†]Fabio Kon is supported in part by a grant from CAPES, the Brazilian Research Agency, proc.#1405/95-2.

irrespective of modifications in the environment. Applications and systems must react to variations in resource availability by dynamically adapting their algorithms, updating parts of the system, and replacing software components when needed.

Unfortunately, the existing software infrastructure is not prepared to manage these highly-dynamic environments properly. Conventional operating systems already have a hard time managing static environments based on components. Even Microsoft researchers say that they are “well aware of how difficult it is to install software on [Windows] NT and get it to work” [MB⁺99] because of the management problems that components induce. In Solaris, a system administrator does not have a clean way of upgrading software that is being executed by the users. When an executable is replaced, the running versions of the old executable crash.

Effective management of the next generation environments for ubiquitous computing requires a novel architectural model for operating systems and middleware. At the University of Illinois at Urbana-Champaign, we are developing Gaia [RC00], an operating system for active spaces. In this paper we describe the requirements faced by such a system and propose an integrated architecture meeting these requirements. The paper focuses on a representation of Active Spaces using standard Naming and Trading mechanisms and on an object-oriented framework for managing heterogeneous devices.

2 Gaia: An Operating System for Active Spaces

As a traditional operating system is the software that manages the resources of a machine, Gaia is the software that manages the resources in an active space. It runs on top of traditional kernels such as Linux, Solaris, Windows, and PalmOS and uses a middleware layer to offer the required distributed services.

We can classify the requirements for an operating system for active spaces in two categories. The first comprises requirements that are also found in most modern distributed environments. These requirements have been addressed by our research on the *2K* operating system and are discussed in Section 2.1. The second includes the requirements that are directly related to the physical nature of active spaces, i.e., to location-awareness. The latter were not addressed by *2K* and is the theme of our ongoing research on Gaia as discussed in Section 2.2.

2.1 Gaia Foundation: The *2K* Operating System

The *2K* operating system [KCM⁺00] was developed to support highly-dynamic, heterogeneous, scalable computing systems. To support an optimal use of such environments, the system must provide support for (1) easy interoperability, (2) naming and trading, (3) distributed resource management, (4) automatic configuration, (5) monitoring, (6) dynamic reconfiguration, (7) security and privacy, (8) fault-tolerance, and (9) quality of service.

2K promotes *interoperability* by using OMG standard services such as Naming and Trading [OMG98] and by defining all its services using OMG IDL. Distributed communication is performed via standard CORBA. Clients can use any commercial or open source ORB to interact with the system. When more flexibility is required in the distributed object mechanisms, a reflective ORB such as *dynamicTAO* can provide the necessary support for dynamically reconfiguring the ORB internal engine to accommodate to changes in the environment [KRL⁺00]. In machines with limited resources and embedded systems, tiny ORBs such as LegORB [RMKC00] can be deployed to achieve memory footprints varying from 6 to 50 KB.

Dynamic software evolution is facilitated by a component-based design and by mechanisms for *Automatic Configuration* [Kon00]. System and application software is assembled at run-time to meet the dynamic requirements imposed by an ever-changing environment. The Automatic Configuration Service in conjunction with the *Distributed Resource Management Service* [Yam00], the *Monitoring Service* [Mao99, KRL⁺00], and the *Environment Service* [CKB⁺00] (the latter currently under development) will provide a solid basis for the construction of adaptive services and applications. Based on the underlying platform, dynamic resource availability, and user preferences, the system will instantiate user applications by selecting the appropriate components, which are then downloaded from network-centric repositories.

Finally, to enhance usability and reliability and to allow the deployment in real-world scenarios, the *2K* design includes mechanisms that can be used to provide *Quality of Service* [NWX00, XWN00], *Fault-Tolerance* [KCN00], and *Security and Privacy* [KRL⁺00, AMAMC00].

2.2 Getting Physical

An operating system controlling active spaces should provide all the functionality contained in a system like *2K*. But, in addition, it must also include services to manage the *physical* spaces in the system. To achieve that, the Gaia design incorporates the following enhancements.

- A federation of CORBA Name Servers to maintain an image of the structure of the physical spaces and references to the objects responsible for each of the different spaces and groups of spaces.
- A federation of CORBA Traders to manage the objects contained in physical spaces (e.g., computing devices, services, real-world objects).
- An object-oriented framework for representing and managing heterogeneous devices in a distributed active space system.
- A *Location Service* to maintain information about the physical location of entities that move across the spaces very frequently (e.g., human users).

In the future, Gaia will also incorporate mechanisms for sensing, visual recognition, adaptive networking and power management for mobile devices, and rendering of realistic 3D models.

2.2.1 Representing Physical Spaces

The physical structure of the active spaces are represented in the system by a federation of Name Servers. Each server is responsible for a partition of the whole system and each partition is further subdivided in smaller units represented as CORBA contexts in the name space. Thus, contexts are used to represent regions of a room, rooms, floors, buildings, city blocks or campuses, cities, counties, etc., which are organized hierarchically.

Each context may contain a reference to a Trader that is responsible for the objects (e.g., devices and services) inside the associated physical space. Traders can be federated and replicated. Figure 1 is a screen shot of the *2K* Name Service browser showing a sample name space for the University of Illinois and its Digital Computer Lab building (DCL).

Figure 1 shows that there is a trader responsible for room 3234 located in the third floor of the DCL building. By browsing the name space, client applications can get a reference to

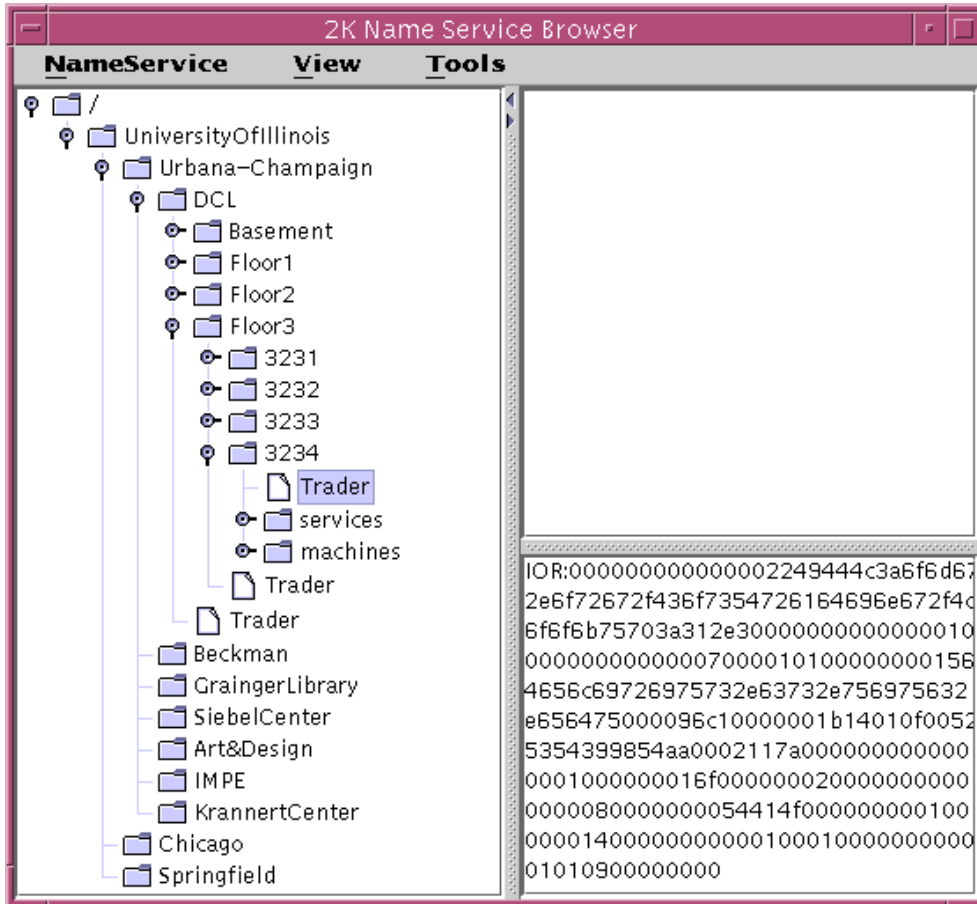


Figure 1: Hierarchical Organization of Active Spaces

the trader and then get information about the objects in that room. Using the OMG standard Trader Constraint Language [HV99, OMG98] a user can request a list of all the workstations in that room that are able to execute Pentium/Linux binaries, have at least 60% of its CPU available and have at least 32MB of free RAM by sending a query like the following.

```
trader->query ("workstation",
               "processor_name == i686'   and
               os_name           == 'Linux' and
               processor_util < 40      and
               RAM_free          >= 32",
               ...);
```

In the example above, `workstation` is the object type. `processor_name` and `os_name` are static properties of objects of this type, while `processor_util` and `RAM_free` are dynamic properties that are updated periodically by the *2K* Resource Management Service [Yam00].

Alternatively, a user may search for devices in a group of rooms. By sending the following request to the trader responsible for the entire DCL building, a client would get a reference to

a postscript printer that prints more than 10 pages per minute and whose price is minimal.

```
DCLtrader->query ("printer",
                  "Postscript == true and PagesPerMinute > 10",
                  "min (Price)",
                  ...);
```

Once the client gets the reference, it can look at its `location` property to see where the printer is physically located and it can use its IOR to request the printing of a postscript file.

The traders are organized hierarchically using the trader federation mechanisms defined in the CORBA standard. Thus, when the DCL trader receives a query like the one above, it forwards the request to the traders responsible for each of its floors and combines the answers, looking for the printer with the lowest price. The DCL trader may also store part of the information locally and cache the results of the most recent queries to lessen response time and decrease system load.

To make the trader information persistent, we are extending the TAO trader with a persistence mechanism based on XML files. All the information maintained by the trader at runtime will be mirrored in an XML format in persistent files. Thus, in addition to adding trader data via the *2K* Trader GUI, system administrators, will also be able to enter new device types and device offers by writing text files containing XML specifications for the devices.

Our group is currently investigating caching strategies for federated traders and the replication of traders within the same context to promote load balancing and fault-tolerance.

2.2.2 A Framework for Managing Heterogeneous Devices

As observed by Ballesteros [BA99], the “file” abstraction has been used by operating systems such as Plan 9 and UNIX as the standard interface for accessing devices. However, the file interface is not necessarily the best one for every device, which, in the case of UNIX, has led to the abusive use of the `ioctl` system call.

Using object-oriented techniques, we are developing a framework for interaction with heterogeneous devices. It allows the representation of device interfaces with different levels of detail and specialization. Gaia device interfaces are defined using IDL, which enables the construction of device drivers and clients in any programming language. A simplified Gaia class diagram is depicted in Figure 2.

Everything in Gaia is an `Entity`, including `Users`, `RealWorldObjects`, `Services`, and `Devices`. `Devices` are divided into input, output, and inout devices. A video camera is an example of an `InDevice`.

The `Entity` interface can be used to get generic information about the entity, its name, location, etc. The `Device` interface has operations for sending and receiving control operations to the device and an operation returning a reference to a `DataContainer`. The data container associated with the device is used to exchange typed data with the device.

An `InDevice` is a subclass of `Device` whose container emits data but does not receive data. A `Camera` is an input device that emits video frames through its container. An `MPEGCamera` is a subclass of `Camera` whose container emits video frames of type MPEG.

Since the data is typed, the system can be extended with several useful features. These include checking for the consistency of data streams, automatically converting data types (e.g., from MPEG to bitmaps), optimizations such as dropping MPEG frames in the occurrence of

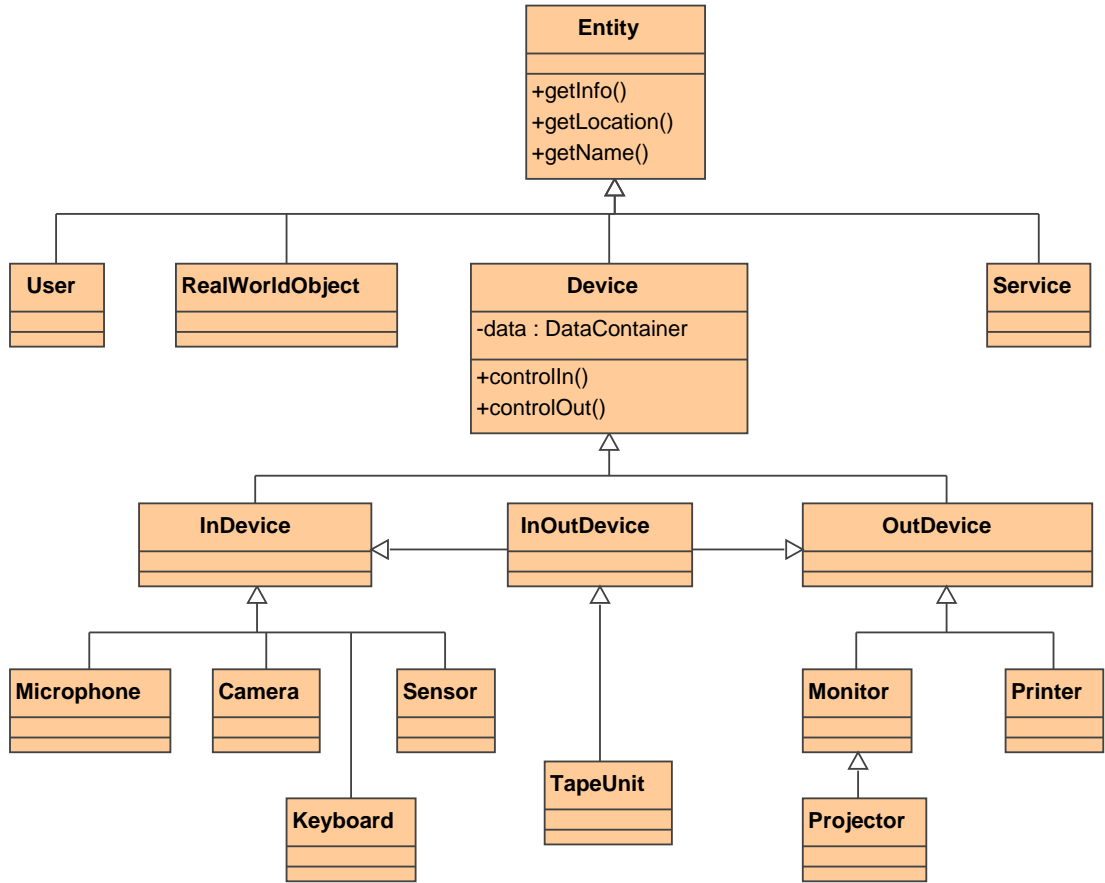


Figure 2: Hierarchical Organization of Active Space Entities

network congestion, etc. This topic has been the subject of our research on the *2K* data management system [HBC00].

Implementation

We have implemented an initial version of the device framework. The interfaces were defined in OMG IDL and a few device drivers and clients were implemented in Java and C++. We are currently implementing new drivers and experimenting with the ones we created in order to evaluate our design. The source code and documentation is available at <http://choices.cs.uiuc.edu/ActiveSpaces/DeviceFramework>.

3 Conclusion

The recent progress in hardware technology is enabling the construction of active spaces with thousands of embedded devices, which augment our perception of reality with new dimensions. Despite recent advances in software technology like Jini [Wal98], existing operating systems and

middleware are not yet prepared to manage such complex environments. They exhibit intricate dependencies and strict requirements with respect to interoperability, automatic configuration, security, privacy, fault-tolerance, and quality of service.

In the *2K* and Gaia projects, we are investigating and developing the required services and mechanisms to manage the complex computing environments we will encounter in future active spaces.

Acknowledgments

The authors thank Brian Ziebart for his prototype implementation of the Location Service. Our ideas for the device management framework were heavily influenced by the work of Francisco Ballesteros on the Box concept [BA99].

References

- [AMAMC00] J. Al-Muhtadi, M. Anand, D. Mickunas, and R. Campbell. Secure Smart Homes using Jini and SESAME. In *Proceedings of the 16th ACSA/ACM Annual Computer Security Applications Conference*, 2000.
- [BA99] Francisco J. Ballesteros and Sergio Arévalo. The Box: A Replacement for Files. In *7th IEEE Workshop on Hot Topics in Operating Systems (HotOS-VII)*, Rio Rico, Arizona, March 1999.
- [CKB⁺00] Dulcinea Carvalho, Fabio Kon, Francisco Ballesteros, Manuel Román, Roy Campbell, and Dennis Mickunas. Management of Execution Environments in 2K. In *Proceedings of the Seventh International Conference on Parallel and Distributed Systems (ICPADS'2000)*, pages 479–485. IEEE Computer Society, July 2000.
- [HBC00] Christopher K. Hess, Francisco J. Ballesteros, and Roy H. Campbell. An Adaptable Distributed File Service. In *Proceedings of the ECOOP PhD Workshop on Object Oriented Systems (PHDOOS'00)*, Cannes, France, June 2000.
- [HV99] Michi Henning and Steve Vinoski. *Advanced CORBA Programming with C++*. Professional Computing Series. Addison-Wesley, 1999.
- [KCM⁺00] Fabio Kon, Roy H. Campbell, M. Dennis Mickunas, Klara Nahrstedt, and Francisco J. Ballesteros. 2K: A Distributed Operating System for Dynamic Heterogeneous Environments. In *Proceedings of the 9th IEEE International Symposium on High Performance Distributed Computing (HPDC'9)*, Pittsburgh, August 2000.
- [KCN00] Fabio Kon, Roy H. Campbell, and Klara Nahrstedt. Using Dynamic Configuration to Manage A Scalable Multimedia Distribution System. *Computer Communication Journal (Special Issue on QoS-Sensitive Distributed Systems and Applications)*, Fall 2000. Elsevier Science Publisher.
- [Kon00] Fabio Kon. *Automatic Configuration of Component-Based Distributed Systems*. PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, May 2000.

- [KRL⁺00] Fabio Kon, Manuel Román, Ping Liu, Jina Mao, Tomonori Yamane, Luiz Claudio Magalhães, and Roy H. Campbell. Monitoring, Security, and Dynamic Configuration with the dynamicTAO Reflective ORB. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware'2000)*, number 1795 in LNCS, pages 121–143, New York, April 2000. Springer-Verlag.
- [Mao99] Jina Mao. Monitoring and Analyzing Method Invocations in the 2K Operating System. Master's thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, May 1999.
- [MB⁺99] Dejan Milojicic, Bill Bolosky, et al. Operating Systems – Now and in the Future. *IEEE Concurrency*, 7(1):12–21, January-March 1999.
- [NWX00] Klara Nahrstedt, Duangdao Wichadakul, and Dongyan Xu. Distributed QoS Compilation and Runtime Instantiation. In *Proceedings of the IEEE/IFIP International Workshop on QoS (IWQoS'2000)*, Pittsburgh, June 2000.
- [OMG98] OMG. *CORBA services: Common Object Services Specification*. Object Management Group, Framingham, MA, 1998. OMG Document 98-12-09.
- [RC00] Manuel Román and Roy H. Campbell. Gaia: Enabling Active Spaces. In *Proceedings of the 9th ACM SIGOPS European Workshop*, Kolding, Denmark, September 2000.
- [RMKC00] Manuel Román, Dennis Mickunas, Fabio Kon, and Roy H. Campbell. LegORB and Ubiquitous CORBA. In *Proceedings of the IFIP/ACM Middleware'2000 Workshop on Reflective Middleware*, pages 1–2, Palisades, NY, April 2000.
- [Wal98] Jim Waldo. Jini Architecture Overview. Available at <http://java.sun.com/products/jini/whitepapers>, 1998.
- [Wei92] Mark Weiser. The Computer for the 21st Century. *Scientific American*, 265(3):94–104, September 1992.
- [WSA⁺95] Roy Want, Bill N. Schilit, Norman I. Adams, Rich Gold, Karin Petersen, David Goldberg, John R. Ellis, and Mark Weiser. An Overview of the ParcTab Ubiquitous Computing Experiment. *IEEE Personal Communications*, pages 28–43, December 1995.
- [XWN00] Dongyan Xu, Duangdao Wichadakul, and Klara Nahrstedt. Multimedia Service Configuration and Reservation in Heterogeneous Environments. In *Proceedings of the 20th International Conference on Distributed Computing Systems (ICDCS'2000)*, Taipei, Taiwan, April 2000.
- [Yam00] Tomonori Yamane. The Design and Implementation of the 2K Resource Management Service. Master's thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, February 2000.