1 Problem Description

The confluence of ubiquitous computing, anywhere/anytime access to information resources, and scalable computing enables the construction of Active Spaces. In such a Space, a spectrum of computation and communication devices seamlessly augment human thought and activity with digital information, processing, and analysis to provide an observed or imagined world that is automated and enhanced by the behavioral context of its users. Large numbers of inexpensive computing devices provide new functionality, enhance user productivity and ease everyday tasks. In home, office and public spaces, ubiquitous computers will unobtrusively augment work or recreational activities with information technology that optimizes the environment for people's needs. The power of such a computer infrastructure has three contributing factors: the translating of information to and from physical properties, the computers and their ability to transform data, and the cooperative computational environment that results from embedding these devices in a network. Given the impact of the Internet, this last factor, the computational capabilities of an Active Space, is the likely long-term benefit of the current information technology revolution. However, the benefit cannot be achieved without devising a new form of operating system that enables applications to be built and run in Active Spaces. The operating system must manage the resources of a physical space and its devices for a user and his applications. This proposal addresses the fundamental issues in the design of such a system.

The power of the computer as a tool to enhance thought and action encourages its broad application to human activities, endeavors and enterprises. Future "smart" cities, buildings, offices, homes, and vehicles will contain many hundreds of thousands of devices and services that join thousands of users through a hierarchy of ubiquitous networks, ranging from personal "body area" networks to the global Internet. In this ubiquitous network, there will be enormous opportunities for information appliance software to gather, organize, and analyze data and then provide sophisticated services that enhance and improve human activities, experience, learning and knowledge. The Active Space must scale and must support the behavioral contexts of diverse groups of mobile users. In particular, the Active Space must support application programs written by or for these users.

Industry is poised to build huge numbers of embedded devices specialized in specific tasks. Already Phillips electric razors come with 7k bytes of code and the automobiles include from 20 to 200 microprocessors. Sun and Microsoft compete to produce software for the next generation "smart" devices. These devices communicate through a spectrum of networking schemes; from point to point infrared, through very local area wireless communications and body networks, to local area wireless and wired networks, to wide-area networks like the Internet and cellular communication networks. Many different approaches are used to support smart devices communication over these networks including web-servers, Jini and RMI, CORBA, and DCOM. Some of these communication schemes are directed towards PDAs and others are targeted towards communicating PCs and server machines. Some schemes assume stateless server models, while others assume stateful servers. Some schemes assume mobility and others permanent installations. We argue that after the inevitable rush of new ubiquitous computing products, there must follow an intensive period of research to build a software architecture for these approaches that integrates them into a computing environment.

In direct analogy with building a computing environment for users to run applications on a dedicated machine, we propose that a computing environment for a physical room, operating theatre, 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 of authenticate users, the provision of services, the reuse of software, and offer portability and flexibility. An active office, home, classroom, operating theatre, building, or city application and its appropriate libraries should run on a ubiquitous infrastructure. A home or office application should be moveable from one physical location to another, overlaying the actual physical devices and geometry of a home of office. 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, from work, or a conference. Entering an Active Space should not require a user login; yet users must be authenticated and user Spaces must be secure.

The end result of this effort will be uniquely user-centric and application-oriented computational environment. Simply put, a user wants perfect toast, irrespective of which toaster or device is used to produce it. Users do not want to program a "VCR-like" gadget that produces toast to specification. Our goal is to let the user run his anywhere/anytime application to produce good toast. Our research is aimed at investigating appropriate software architectures so that there will be a foundation upon which industry can build. The next section introduces example problems in an application setting that our architecture is intending to solve.

Example

In order to convey the problems and issues with building a smart space for computer science education, we review scenarios of its possible uses in the new Siebel Hall building at UIUC. Siebel Hall has a number of different rooms that support Active Spaces. A typical room contains a display wall, rendering engines, speakers, wire and wireless network infrastructure, wireless microphone, several wired microphones spread across the room, directional cameras and directional lights. The devices are controlled through a variety of different schemes including Jini, CORBA, and DCOM.

The computer science curriculum contains courses. The classes of a course are assigned to meet in rooms in the building for lectures, discussions, and laboratories. Each activity may require a different room configuration. For the scenario, the class may have a teacher and students, including a hearing impaired student and a remote student attending the class from her house. Students have different devices (PDAs, Laptops) and carry them to class. These devices communicate over a variety of different networks; infrared, X10, BlueTooth, wireless Ethernet, Cellular communication and wired Ethernet.

For a lecture, the slides and notes are retrieved before starting the class, while the graphics, audio and the video are generated on the fly. The video cameras detect the teacher in the class, and the active room authenticates the teacher, activates his microphone and configures the audio parameters according to the geometrical properties of the classroom. Video cameras tracking the teacher provide gesture control and speech recognition to provide last minute adjustments to the presentation. For example, the teacher may point to a display and verbally command the video to be displayed there. The active classroom uses its video cameras to match the students against the class roster and helps the network authenticate any new portable devices that it hadn't expected. At this point the class is ready.

Gaia OS bootstraps the rooms using underlying mechanisms like Jini discovery and publish and subscribe. The maintenance service looks for failed devices like cameras or lights, readjusts lighting and display intensity and color to compensate for temperature, use, and user adjustments. Any physical problems are emailed to the lab technician before the classroom schedule begins. In preparation for the class, the system examines the schedule and caches information for the classes for which it is scheduled.

Students may customize their view of the classroom according to their particular requirements. Some students drop in late and quickly read a summary to catch up. During the lecture, students with laptops or PDAs may interact with the system to download a view of part of or all of the slides or video. Gaia uses adapters to customize the slides or video to the hardware properties of the student machines (for example, downscaling the size, reducing the color depth, and adapting the color model of the displays). The hearing impaired student registers his PDA to view the audio. Gaia creates an adapter to convert the audio to text so that the student can read what the teacher is saying. A transcript of the lecture is generated and inserted in the notes. To preserve the audio-slides-notes synchronization, the adapter uses the quality of service to enforce real-time transcription deadlines.

The remote student registers her home with Siebel Hall in order to establish a collaborative active classroom in her house. Sound equipment is mapped to sound sources, her TV is mapped to the video, and her computer monitor becomes the view for the slides and notes.

When a student asks a question, the active classroom locates the position of the student and activates a microphone in that area. Using the position of the student asking the question, the room points the camera towards him and modifies the lighting patterns by increasing the intensity in that region so the quality of the recorded image is bright enough.

In the interactive design session, a student may take control of all or part of the Active Space. The student may provide video, slides, graphics or a display of his computation. The student and lecturer may share a screen to edit slides or control rendering.

The teacher may run an Active Room application that demonstrates the impact of the principles being taught in a simulation. For example, the instructor runs an application that provides a visual replay of the numerical computations that must be performed by parallel processors simulating the rocket motors involved in the launching of a Saturn rocket. As the simulation evolves, students and the teacher may guide the simulation to study specific portions of the activity.

However, day-to-day events in Siebel Hall are never dull. Students may have to leave a discussion early, the instructor may have to give his lecture from Washington, and a power failure may make a room unavailable requiring the students and instructor to improvise with laptops on the lawn. Reconfiguration and dynamic adaptability are key if Active Spaces are to work in practice. Applications written for Active Spaces must be able to operate in a variety of conditions requiring careful design of interfaces and system support.

2 Related Work

In recent years, ubiquitous computing has become a serious research topic. Some of the earliest and most influential work occurred at Xerox Parc and introduced a Tab, a low-capacity networked terminal, and Active Badges, a mechanism for tracking users. MIT's Oxygen project envisions a world where embedded computers will be so pervasive that it becomes a part of everyday life. Speech, vision, and movement drives interaction with these machines. The Endeavour project from Berkeley focuses on integrating sensors, actuators, position locators, displays, and mobile handheld devices. A key component is the processing and management of information as it flows through the system. The Classroom2000 project from Georgia Tech is an application of Active Spaces to the classroom. The architectural scheme records and timestamps events associated with a class, supports development of interfaces to integrate and associate related streams, and provides universal information access via the web. CoolTown [1] from Hewlett Packard leverages existing Web technologies to create "web presence". Physical spaces are accessible from the web. Web sites are no longer simply virtual, but can consist of physical sites. Beacons are used to determine when a user has entered a space. The web site for the space is a collection of relevant information for the particular location. Easy Living from Microsoft Corp. focuses on home and work environments. Preferences are associated with users to customize physical spaces. A geometric model is used to track physical object in the space and fine-grained location detection is determined with video cameras in conjunction with object detection software. Portolano from the University of Washington strives to provide invisible computing through transparent user interfaces, universal connectivity, and intelligent services. The project is investigating determining user interfaces based on user location and movement, and data-centric networking, and novel models for distributed services.

Jini provides an architecture for discovering services and allowing devices to communicate. Devices can offer services that others can access without complicated configuration. Devices may be introduced on the network and the services it provides are immediately available to others. CORBA offers a distributed object architecture that is platform independent. A variety of services are available to facilitate the construction of applications. DCOM is a proprietary alternative to CORBA. Although these architectures provide infrastructure and services, they in themselves do not define an active space; much is needed in addition for their realization.

The common denominator for these projects is that they provide services and applications for ubiquitous computing. However, they do not define a common model of the space where computation takes place. Our proposed operating system called Gaia exports such a model in order to provide the same interface and basic services for diverse physical spaces. This model describes the physical space including the contained entities (e.g., devices, services and people) and their properties (e.g., interfaces, QoS characteristics, security issues and location). The model supports sophisticated applications that exploit the attributes of the available computational resources for its users. It is possible then to write applications in terms of a generic active space, which shields applications from the specific details of particular physical spaces.

3 Results from previous NSF research

Previous grants from NSF have supported several research initiatives in our group and in collaborating groups in our Department of Computer Science.

3.1 Internet, ATM, and Multimedia

Professor Campbell, in collaboration with Professors Reed, Chien, and Kaplan, was awarded an NSF CISE Infrastructure grant CDA 94-01124 ($1.7M) to study multimedia and video servers and collaborative environments using ATM technology. It has led to the development of Web Video, the VDP adaptive bandwidth protocol, the Vosaic continuous media client/server technology, and a scalable multimedia distribution system [2] [3] [4] [5] [6] [7] [8].

Prof. Nahrstedt's currently ongoing NSF Career grant, called Time-variant QoS Management, supports research in the area of time-variant Quality of Service(QoS) management. The award (NSF CCR 96-23867) totals approximately $200K for a period of four years. She investigates the necessary QoS services in distributed resource management to provide bounded control on timing QoS requirements such as jitter and synchronization skew for multimedia applications. Current results from this grant include (1) the QoS-aware resource management, called QualMan, which comprises a low level middleware and consists of CPU, memory and communication resource servers with low level and precise timely QoS control [9] , and (2) the QoS translator [10] [11] [12] with extended translation service between application QoS and system QoS for MPEG/MJPEG-compressed multimedia data residing on top of the QualMan.

3.2 Object-Oriented Operating Systems

In the fall of 1988, the Department of Computer Science received its second Institutional Infrastructure award (PIs Roy H. Campbell and Daniel A. Reed) that totaled approximately $2.4M. The goals of this project, called Tapestry, were to explore mechanisms and policies for parallel systems software. Mechanisms, as embodied in the object-oriented Choices operating system, were designed to permit reconfiguration of operating system components to support new parallel architectures and applications. By replacing operating system policies (e.g., task scheduling, message management or file management) one could study the effects of those policies in a consistent framework.

Choices demonstrated an object-oriented system architecture that embodies the notion of customizing operating systems to support particular hardware configurations and particular applications. Choices has, as its kernel, a dynamic collection of objects. It supports an object-oriented application interface based on local and distributed objects, inheritance, and polymorphism. System resources, mechanisms, and policies are represented as objects that belong to a class hierarchy. Subsystems of Choices are designed as frameworks of classes. The Choices research provided significant contributions in the areas of adaptive file systems [13] and fast process migration [14]. An architecturally-aware interactive visualization tool called OSView [15] [16] [17] allows users to navigate and explore the Choices operating system environment, examining dynamic interactions, manipulating system objects, and evaluating system performance. A system redesign, following the microkernel philosophy, led to MicroChoices, which provided support for dynamic loading of native code [18] and of Java bytecode[19], real-time networking [20], and multimedia applications [21].

Profs. Campbell, Nahrstedt, and Mickunas were awarded the NSF grant 9870736 ($970K) and the NSF/CNPq Collaborative grant 9970139 ($100K), which have supported the development of the 2K Operating System for distributed heterogeneous environments. This included a novel model for dependence management in component-based distributed systems [22] [23], a reflective ORB [24] that supports dynamic security policies [25] [26] [27] and monitoring [28], a model for supporting user environments [29] and resource management [30] in heterogeneous systems, an wide-area multimedia gateway protocol [31], and an infrastructure for multimedia service configuration and reservation [32].

Recently, the NSF CISE grant NSF EIA 99-72884 ($2M) is providing the infrastructure support for our ongoing and future research on smart information spaces. The grant requested in this proposal will provide the required human resources for carrying out this research.

3.3 3-D Geometry, Computer Vision, and Pattern Recognition

Prof. Kriegman is currently a PI and Co-PI on two NSF Grants: Domain Independent Vision-Based navigation (IRI-9711967 by D. Kriegman and G. Hager) and An Intelligent Microscope for Transmission Electron Microscopy (DBI-9904547 by C. Potter, B. Carragher and D. Kriegman). However, this proposal is most closely related to research under an NSF Young Investigator Award (NYI IRI-9257990, $313K, 8/30/92-9/30/99).

The NYI supported research in robot planning [33] [34], mobile robot navigation [35] [36] [37], and multi-frame, perspective projection structure-from-motion [38], [39]. In early work, he developed the first method for relating the 3-D geometry of curved objects (represented by algebraic surfaces) to the occluding contours detected as linked edges in images [40] and the first implemented methods for constructing the aspect graph (a representation of all topologically distinct line drawings) of curved 3-D objects [41]. To recognize 3-D curved objects with more complex shape, he discovered and used invariants of viewpoint-dependent features to index a model database [42] [43] [44]; this indexing mechanism was used in the first-published algorithm for estimating both object structure and camera motion of curved 3-D objects from the deforming silhouettes detected in video sequences [45]. With P. Belhumeur at Yale, Prof. Kriegman have modeled lighting to produce effective methods for object recognition, for reconstructing surface geometry, and for image-based rendering [46]. This work began with a new face recognition method called Fisherfaces [47], which had an order of magnitude lower error rates than conventional methods. Since this data-driven method does not model shadowing or multiple/extended sources, we developed an explicit model of the set of images (the illumination cone) of an object under all possible lighting conditions [48] and applied it to face recognition [49] [50]. Prof. Kriegman received the best paper awards in the 1996 IEEE Conference on Computer Vision and Pattern Recognition and in the 1998 European Conference on Computer Vision.

3.4 Graphics

Prof. Garland's research is centered on the adaptive representation of shape and the efficient display of geometrically complex, visually realistic environments. Much of his recent work has been focused on the design and implementation of effective algorithms for the automatic simplification of highly detailed polygonal surface models into faithful approximations containing fewer polygons. His QSlim [51] [52] simplification system has been recognized as one of the fastest available algorithms capable of producing high-quality results. It has been used in a number of real-world applications, including medical imaging, scientific visualization, CAD packages, and game development systems. His research has also produced a closely related clustering algorithm which forms the basis for an efficient new multiresolution radiosity system [53]

3.5 Adaptive Communication and Power Management

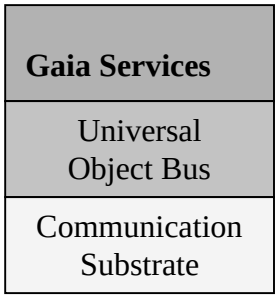
Prof. Kravets has had experience in the areas of adaptive communication and power management. Her key contribution in the area of adaptive communication is a cooperative solution to dynamic resource management, implemented by a communication layer that uses both application resource requirements and network resource availability to determine how to best configure the application's communications [54]. This communication layer provides flexible network support through the use of a configurable protocol subsystem. In the area of communication-based power management, she designed and implemented a power management protocol that allows the application to conserve battery power during idle times in communication [55] and researched for communication time power-aware communication for mobile computers [56]. Central to her work is the notion of communication actions, which captures both the energy consumed for data transmission by the network interface device (e.g., wireless Ethernet, packet radio modem) and the energy consumed by the host when it generates or consumes this data.

4 Proposed research

Several approaches that provide models for interacting with Active Spaces have been proposed [57] [58] [1]. However, the solutions they provide are customized for particular scenarios or are targeted towards a specific type of functionality. In this proposal, we describe a generic model, called Gaia, that exports and coordinates the resources contained in a physical space thereby defining a generic computational environment. The model we present requires a revolutionary form of operating system that manages heterogeneous computational resources and exports them uniformly, facilitating the control of physical spaces, and the development of applications that run in the context of a space. Gaia converts physical spaces and the ubiquitous computing devices they contain into a programmable computing system or Active Space. We will investigate the supporting infrastructure to build applications that exploit ubiquitous communicating computer devices embedded within a physical space.

We argue that Active Spaces are 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. 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; output devices, such as monitor, PDA screen, wall-mounted display, or speakers. An operating system for 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 will focus on the study of issues related to the design and implementation of such an operating system, called Gaia OS. Applications running on this system require the development of a novel application model that defines the rules to build applications that run in the context of a space.

As a part of Gaia OS, we propose a new application model inspired from the Model-View-Controller (MVC) model. The traditional MVC model is composed of software components. We expand the notion to include hardware and physical entities. Models may be ambient room conditions, human vital signs, or a person's schedule; views may be projection screens, a phone, lights, or audio speakers; controllers may be a mouse, keyboard, hand motion, a person entering a room, or voice. Different views may be registered with the same model to render data; multiple controllers may be used for applications and may choose which is most appropriate as environmental conditions change. The infrastructure must manage all these dependencies and seamlessly adapt to device types and characteristics.



While traditional distributed systems are concerned with resources such as CPU, memory, and bandwidth, Gaia must deal with spaces augmented with a vast assortment of other resources, include sensors, actuators, displays, stores, processors, networks and users. Gaia OS is composed of an adaptive communication substrate, a unified object bus, and a collection of services. The physical environment may include high-speed network connections to low-bandwidth unreliable wireless links and devices may include a variety of network interfaces. We will research mechanisms to efficiently and easily enable the simultaneous use of multiple communication channels to provide seamless connectivity. The unified object bus will provide a uniform method for devices to communicate. Existing distributed object implementations are not well suited for devices possessing a limited amount of resources, due to their size and complexity. Often, much of the functionality is unused and not needed for these devices; they only require a minimal amount of infrastructure to support interaction with surrounding devices. This requires an understanding of how to construct composable software that can adapt to device characteristics. The Gaia OS services will provide the additional infrastructure required by applications to execute in an Active Space. They include quality of service (QoS), resource management, security, application configuration, code distribution, data management, sensing and rendering.

4.1 Adaptive Networking

Wireless connectivity is vital to ubiquitous computing to support mobile, untethered, and easy to deploy computing. Gaia is built on an adaptive communication substrate that enables seamless communication for active spaces, incorporating wired and wireless connectivity. Connectivity for active spaces is governed by the communication technology on specific devices and the coverage area of that technology. Many devices are normally built with multiple communication technologies and the capacity for expansion. Laptops commonly come with built in infrared and with multiple PCMIA slots, allowing for multiple communication cards. With the growing demand for mobility support, coverage areas for wireless connectivity are growing and overlapping. Due to different administrative authorities and different underlying link layer technologies, the current infrastructures only support coordination and cooperation between homogenous support stations ("horizontal handoffs"). In order to support better handoffs, communication quality and cost optimization, the mobile node's operating system can coordinate the simultaneous use of multiple diverse communication channels, providing seamless connectivity as the mobile node migrates between coverage areas ("vertical handoffs"). If the mobile node maintains multiple communication channels, the application can be guided as to which channels is currently the most appropriate [59]. Such support is limited to the use of one communication channel per application data stream. The adaptive communication substrate in Gaia exposes the abstraction of a communication channel to the upper layer, namely the object bus. This abstraction transparently supports the aggregation of multiple communication channels, enabling the seamless use of multiple channels for multiple and individual data streams.

Successful techniques for the use of multiple communication channels must have an understanding of the effect of such use on the mobile node's performance and available resources. While adding communication channels certainly increases available bandwidth, it may also increase the node's energy consumption. Furthermore, the use of this extra bandwidth may have unexpected effects on the transmission of the applications data. For example, the transmission of rate-base multimedia traffic may incur longer delays, due to the introduction of more bandwidth on a channel with a longer propagation delay. The goal of this research is to provide intelligent support for the use of multiple communication channels based on knowledge of the needs of the application, the communication resources available from the channels and the energy resources available at the node.

The adaptive communication substrate has three major components: the resource manager, the protocol suite, and the upper layer interface. The resource manager is responsible for discovering and monitoring available communication channels. For channel discovery, the resource manager must determine when and if it is necessary to search for new channels. These decisions are based on information about the current usage patterns of the application and current energy levels of the mobile node. Usage patterns can be gathered from statistical monitoring of the application data stream or specified directly by the application. Channel discovery algorithms must balance the tradeoff between timely discovery of communication channels and the amount of resources used to find such channels.

Traditional protocols used over multiple communication channels may not be able to efficiently use the bandwidth from all channels. For example, packets send over different channels will often arrive in a different order than they were sent. Although TCP can tolerate some messages arriving out of order, a protocol designed for a high degree of out-of-order packets will increase performance. Additionally, a protocol that knows that there are multiple communication channels will be able to perform better resource estimations for each channel. In order to efficiently

support the use of multiple channels, we propose the design and implementation a suite of transport level protocols to support different classes of applications.

The upper layer interface provides a channel for exchanging resource information between the framework and the application or operating system. By providing information about resource availability, the application can adapt its configuration to the current availability. We specifically focus on bandwidth and energy. We will define an interface that allows the framework to inform the application of its current options. This will include information about the bandwidth and delay parameters for available channels and the energy consumption associated with the use of each channel. The application can use this information to determine what quality of data to transmit based on the effect of the transmission of battery lifetime. An example of the parameters of such an interface would include bandwidth, delay, and energy characteristics for various combinations of the available communication channels. By aggregating the information before it is passed up to the application, the resource manager can hide specifics about the number and characteristics of individual channels.

In order to provide transparent lower layer connectivity and location management for mobility, our communication substrate integrates network-level location management techniques from MobileIP [60] and the newly proposed micro-mobility protocols [61] It has been suggested that the paradigm of locating a host is too low-level, and that what needs to be located is the actual person, or their intelligent proxy. The introduction of multiple communication channels changes the paradigm for mobile communications, defining communication with any of the interfaces on a device to be successful communication. Successful mobility management must provide the ability to find the device wherever it is currently connected to the Internet. The challenge lies in expanding the current support for devices with a single interface to include multiple interfaces.

4.2 Unified Object Bus

This research provides a framework for the aggregation of diverse distributed object and web based communication schemes into a common mechanism to access objects within an Active Space. The Unified Object Bus provides the basic mechanisms for packaging, loading, and deploying software components into the system. It will be built as an abstraction layer over specific component architectures such as DCOM, CORBA, and Enterprise Java Beans. This will allow the construction of complex software systems by integrating the components implemented using any of these different architectures. Interoperability will be achieved by instantiating inter-architecture bridges either statically [62] or dynamically [63].

The heterogeneity and dynamism inherent to active spaces leads to high levels of heterogeneity at the link and transport layers. Thus, the Object Bus must be able to implement seamless communication over a wide variety of networks (e.g., Fast Ethernet, ATM, wireless LAN, wireless WAN, IR), using a wide variety of protocols (e.g. UDP, TCP, IP-Multicast, WTCP, VDP, MMTP [Magalhaes, 2000 #283). One of the goals of the Unified Object Bus is to hide the heterogeneity at the lower layers while still allowing the selection of the most efficient protocols at any time. Thus, our object bus will utilize a reflective middleware layer [64] that will be based on our previous research results in reflective ORBs [24]. This technology allows applications to select dynamically the most efficient transport protocols and the most suitable strategies for aspects such as security, scheduling, demultiplexing, marshalling, and concurrency. As many of the devices present in an Active Space will possess limited resources, we will research the requirements for extending the Unified Object Bus to support PDAs and embedded devices, using technologies such as LegORB [65].

4.3 Services

This research provides a framework for defining, developing, and deploying a standard infrastructure and a common model to support Active Space applications and users. Gaia services must interoperate, support critical functions, discover resources, and provide an execution environment. For example, critical services include security, environment, resource management, and object streaming. Dynamic systems require configuration and code distribution services. The enhanced services, sensing and rendering, provide additional functionality that is necessary to support our key applications for the classroom and operating theatre.

4.3.1 QoS-Aware Resource Management

This research provides solutions to satisfy the demanding QoS requirements of Active Spaces. Active Spaces employ heterogeneous networking and require seamless communication among all software and physical objects within the distributed computing environment. The physical network infrastructure can consist of both wireless and wired networks. Devices may be connected through a diverse array of networks possessing different bandwidth. The algorithms, services, protocols and architectures complying with application QoS requirements must deliver

service location-awareness, adaptive behavior, availability, scalability, and stability [66]. We propose to develop novel approaches to QoS-aware service location, discovery, and configuration management that coexist with hybrid communication protocols. The QoS-aware communication and computing support will strongly support Gaia components, including the operating system, automatic configuration, security, mobile communication, and adaptive protocols.

We propose a hybrid control model to satisfy the demanding QoS requirements of the system. This model consists of a three-layer scheme to control resources and services in an end-to-end fashion. The lowest layer of control provides resource reservation, supporting minimal quality of service requirements. Especially challenging will be the coordination of distributed heterogeneous resource reservations. The middle layer uses adaptive control theory and includes controllers such as Proportional Integral Derivative (PID) and feedback control to adjust the application qualities in case of small bursts of resource availability fluctuation [67]. Such fluctuations occur when requests for resources exceed the reserved resource allocation due to a desire for higher QoS. Adaptive control offers stability ranges, as well as agility conditions, to react to small changes in resource allocation above the reservation level. Furthermore, the interaction between distributed control and multiple adaptive controllers must be examined. Currently, the exact effects of various distributed controllers on each other are unknown. We will investigate these relations analytically as well as experimentally within the Active Space framework. The top layer will use fuzzy control to apply linguistic rules to adjust application quality of service for large bursts and large changes in resource allocation above the reservation [67] These rules will provide functional adaptation and enforce dynamic QoS-aware reconfiguration of services and applications. As new resources and end-to-end paths are discovered, components may be reconfigured, based on decisions that take into account various trade-offs [68].

The functionality provided by this model will be exported to Active Space applications by using an advanced version of our distributed Resource Management (RM) Service [30] [69] [68] and new entities of the QoS-aware Resource Management. The current RM service is composed of simple Local Resource Managers (LRMs) [70] [66], present in each networked machine and a Global Resource Manager (GRM) per Active Space. The LRM's task exports the state of the hardware resources in that machine to the whole distributed system. LRMs send periodic updates of the state of their resources to the GRM, which maintains an approximate view of the Active Space resource utilization state. The GRM then utilizes this information as a hint for performing QoS-aware load distribution [32] within the space. Groups of GRMs can be combined hierarchically to provide hardware resource sharing across multiple spaces connected through the Internet [31]. The new challenges consist of bringing the three layer model, supporting minimal reservation, adaptive control and fuzzy control, into the distributed Resource Management architecture in such a way that we achieve scalable, adaptive and high quality delivery of information and services to applications in Active Spaces.

Furthermore, our QoS-aware Resource Management Service will leverage the services provided by standard trading services, providing powerful mechanisms for resource and service discovery and translation based on specifications of application QoS requirements [71]. Here, new problems of QoS compilation and instantiation will be studied to design and formalize new algorithms for resource and service discovery, distribution, profiling, translation, compilation, and instantiation with the goal of end-to-end quality provision.

4.3.2 Automatic Configuration Service and Code Distribution

The Automatic Configuration Service's objective is to ease system and application configuration by managing two kinds of inter-component dependencies: *prerequisites* [72], which specify the requirements for loading an inert component into the runtime system; and *dynamic dependencies* [22] [23], which refer to the dependencies among loaded components in a running system. As long as the system has access to the requirements for installing and running a software component, the installation and configuration of new components can be automated [72]. As a byproduct of this knowledge, component performance can be improved by analyzing the dynamic state of system resources, by analyzing the characteristics of each component, and by configuring them in the most efficient way.

The prerequisites for a particular inert component specify any special requirement for properly loading, configuring, and executing that component. Prerequisites specify 1) the type and share of hardware resources that a component needs and 2) the software services (i.e., other components) it requires. The first kind of prerequisites lets the QoS-aware Resource Management Service determine where, how, and when to execute each component. It uses this data to enable proper admission control, resource negotiation, reservation, and scheduling. The second kind of prerequisites determines which auxiliary components must be loaded and which other software services must be located. Prerequisite specifications can be created manually by application developers but we will also investigate mechanisms for automating its creation using, for example, software tools for QoS profiling such as QualProbes [73].

As the Automatic Configuration Service parses the software prerequisite specifications, it verifies whether it is necessary to create new instances of the required components in the system runtime. If necessary, it contacts the a repository of components, fetches the component binary code compiled for that specific platform, and dynamically loads it. This approach is reminiscent of the Jini pull-based mechanism for code distribution [74]. Using the Automatic Configuration Service, a client running in a certain network node *pulls* the code and configuration information from a *Component Repository* available in the Active Space. Our approach takes the Jini mechanism to a new level, enabling the construction of sophisticated component-based systems in a language-independent way. To support that, we will develop a flexible Component Repository that will be able to (1) classify components according to their categories, (2) store multiple implementations of the same component for different kinds of software and hardware platforms, and (3) support security by providing mechanisms for access control and for digitally signing components.

In large-scale systems, this pull-based approach may not be enough. To support efficient and scalable code distribution, we also need a mechanism to allow system administrators to *push* code and configuration information into the network. Our architecture will achieve that by using mobile agents for distributing software components and reconfiguration scripts across Active Spaces in a scalable way [75].

4.3.3 Object Stream Service

This research provides typed access to distributed data storage and a uniform mechanism to perform automatic data transformations in an Active Space. Active Spaces require the exchange of large amounts of information among stores, devices and services. Due to the heterogeneous environment, some devices may not be able to render data in its original format. This service uses *adaptors* to match content to the particular device characteristics and application requirements.

The object stream service supports application access to abstract containers that are structured data aggregates, extending the concept of data as files, with blocks of data on disk. In the most basic form, containers are simply wrappers for native file data or directories. However, in general, they are much more interesting and useful objects [76]. Containers are *not* traditional persistent objects. They may represent a data transformation, pieces of data collected from various sources, or data that is grouped in a particular way. Considering data in this way allows type and format specific operations to be performed on these objects. Containers simplify the interface that applications use to access the wide assortment of data types in an Active Space.

Applications gain access to data in a particular format by opening the data source as a specific container type. The system automatically adapts content to the desired format. For example, an application may open a container holding lecture notes in text format and paste them to a display. However, a different application can retrieve the data from the same source in an audio format and write the objects to an audio device. The lecture notes are automatically converted to audio translations.

The system can transparently instantiate a sequence of adaptor containers to construct appropriate *object streams*. The object streams may alter, convert, or filter data objects as they flow through the system, in order to perform the proper transformations to present the data in a format that suits the application requirements. Devices with limited resources may require external support to assist in computationally intense data conversions or manipulation operations. Containers can act as proxies to support these devices. The system can place container objects on different nodes, based on knowledge of surrounding devices, to distribute load. Thus, for example, graphics images may be displayed on a networked PDA using an appropriate distributed rendering object stream. The research issues include adaptation mechanisms, how to apply adaptation, where to locate adaptors, how to create the correct sequence of adaptors according to dynamic requirements, caching, and how to maintain the consistency of different views of stored data.

4.3.4 Security Service

This research provides security services that support authentication, confidentiality, integrity, access control, interoperability, and protection from attacks for Active Spaces. An Active Space may be populated with a large number of embedded and mobile devices as well as a number of mobile users requiring new approaches to assure security and privacy. Access to Active Spaces, smart devices, and information requires varying degrees of authentication. Gaia supports various authentication mechanisms including passwords, badges, smart cards, face recognition, and biometric techniques by providing the developer with a general API for authentication.

Different cryptographic techniques must be employed to ensure that information is accessible only to authorized principals, while protecting it from tampering. Gaia will wrap different cryptographic mechanisms with the standard GSS-API (General Security Services API) [77], allowing developers to select appropriate mechanisms without

affecting existing applications. Furthermore, in many scenarios involving mobility there is a limit on the processing power, energy, and memory that a security mechanism can utilize, requiring the utilization of different *Quality of Protection* (QoP) levels. *Tiny SESAME* [78] is a lightweight security mechanism that we plan to integrate in Gaia to provide the various components of an Active Space with a subset of both the SESAME [79] and Kerberos [80] security and authentication services. Additionally, we plan to extend Tiny SESAME to incorporate support for different QoP levels.

Different principals must be granted different privileges according to their roles in the Space at a particular time. Gaia will incorporate the GAA-API (General Authorization and Access Control API) [81] to allow applications to make authorization decisions. The Gaia security service will support RBAC technology [82] to facilitate access control in a flexible and dynamic way. Mobility and remote access requires authentication of principals from Active Spaces with foreign security domains. We will use our model for secure interoperability [83] to research secure interactions among Active Spaces in different security domains.

The Active Space environment with mobile devices and users provides many opportunities for implanting "rogue devices" that could attempt to SPAM principals or launch Denial of Service attacks. Our research will investigate methods for avoiding these attacks. Security policies and mechanisms will be customized for mobile users, smart devices, applications, QoS requirements, and physical spaces.

4.3.5 Environment Service

This research provides context and state for the execution of user applications in an Active Space [29]. An environment is represented by an aggregation of structures that embody entities such as services, devices, users, and programs currently involved in the computational context of the application. The structures include events, profiles, preferences, configuration parameters, and sub-environments. Environments must provide a level of indirection and a mechanism for manipulating groups of entities, for example, by providing support for changing the devices used by an application of a mobile user. Thus, it should support mapping a presentation of audio, video, and slides from a mobile user's laptop to the devices in a room as he enters the room. The research issues include event handling, environment support for dynamic resource mapping, the definition mechanism for profiles and preferences, and the mechanisms for performing operations on groups of entities.

4.3.6 Application Management

This research provides libraries, frameworks, and foundation classes to construct applications for Active Spaces. Many of the applications in Active Spaces are interactive. These applications involve dependencies between sensing, computation, and physical space, views or displays. In Gaia, we provide an application management service that supports the ability to create, register, modify, list, delete, and schedule interactive applications and deliver their events. The application management service is inspired by the Model-View-Controller [84]. In a MVC, the model contains application data; the view is a visual representation of the model, the controller provides information to update the view and the model, and the system maintains consistency between the view and the model as they change. In Gaia, this framework is reinvented to support adaptive interactive applications in the distributed, dynamic, and heterogeneous nature of Active Spaces. The framework introduces a new element called the object streams *model adapter* that transforms, at runtime, the type of data exported by the model to the type of data expected by a view. Creating an interactive application in the context of an Active Space requires providing a model, attaching one or more views, and associating one or more controllers to each of the views. Views are attached to the model using model adapters that format the data according to the requirements of the view. Finally, controllers, reacting to sensing services, modify views and their corresponding models. For example, students and teacher in an active classroom may observe a rocket engine simulation. Reacting to a view of the model of the engine, they may change parameters of the simulation through a sensing device configured to act as a controller and see the impact of those changes on the view. In an Active Space, the view can broadly be interpreted to include not only the displays in the physical space but also the effects of any embedded actuator devices in the physical space. The research issues include reinventing the MVC for Active Spaces, developing the frameworks and libraries, defining a consistency model for interactions and building support for dynamic reconfiguration.

4.3.7 Sensing Service

This research provides a rich set of sensing devices for Active Spaces while making ubiquitous computing less intrusive. While some user interactions with an active information space are intentional and are provided through conventional input devices such as keyboards, mice, and voice, a space will also be aware of and react according to the collective users' intentions, objectives and physical/task states. Accordingly, physical and concomitant computing resources can then be deployed according to the sensed physical state of the persons and objects within

the space. For example, by sensing the identity, location, and gaze of individuals, cameras can be moved (pan/tilt/gantry) to provide the best vantage points in telecollaboration. As identified by the space, multiple cameras may be focused upon "important" participants while a single low-resolution video source may be allocated to a group of secondary participants. Gestures can be used like a pointing device to control display content, lighting, and other physical devices. As a person moves throughout the room, relevant content and context may follow her. On the other hand, sensitive content may be automatically censored from a screen when an unauthorized individual is within view of the screen.

To realize these capabilities, the Active Space will be equipped with video cameras, which are both passive (permitting many sensors to cover one space) and high bandwidth. While other biometric methods (e.g. fingerprint or iris) provide more secure identification, they cannot provide dynamic information about the space's state. The challenge of using vision is that it provides an indirect measurement of state with many confounding factors.

We will provide the infrastructure to enable the rapid deployment and reconfiguration of sensor/display devices within a space, on top of which vision-based HCI techniques can be applied. We will also explore how active control of the space can facilitate these HCI modalities. Imagine bringing a newly constructed space on-line, temporarily bringing new cameras or displays into an existing space, or rearranging the furniture in a multi-use space. We would like an Active Space to determine the mapping between perceptual services and devices along with their operating parameters. We would also like capabilities akin to "plug-in-play" where new sensors and displays can be seamlessly incorporated or removed from the space. Unlike a computer's operating system, a table of critical device parameters/drivers cannot characterize the device's performance in-situ. While one may be able to probe a digital camera to determines its model number and hence its intrinsic characteristics, the space will also need to know its extrinsic characteristics (i.e., its relationship to the physical environment, the displays, and to other cameras).

We have all experienced the effect of moving a color image from one display (e.g., a CRT) to another display (e.g., projection screen), only to be disappointed that the colors appear different. Even for identical monitor types, hue and saturation controls may be set differently. Within an Active Space where graphical/video objects will be moved between displays, this effect will be particularly disturbing. Pairs of color cameras observing a collection of scene points can ascertain the relation between their relative color responses (e.g., a linear relation between the RGB values). If the scene points happen to be on displays, then a mapping between display colors and observed colors can be determined. Since these mappings are invertible, colors can be balanced across the displays using color lookup tables on the outputs of the graphics engines, much like gamma correction [85]. The video input can be similarly transformed.

Though the lights are controllable, the space may not know which light illuminates which area of the space. However, the lighting can be actively varied, and the affect can be measured with video cameras. While the lights themselves may be outside a camera's field of view, their effect on the scene will be observable. We will relate the effects of controlled lighting variation on the set of video streams, and also on the 3-D world.

The geometric configuration of surfaces (floor, doors, steps, tables, chairs, benches, white boards, etc.) can be determined from video images and controlled lighting. We will use a combination of multi-view stereopsis coupled with controllable lighting akin to photometric stereo. We have developed very accurate shape and albedo reconstruction methods up to this transformation, though only applied them to faces [50] [46]. We will consider the issues in how active lighting control (intensity but not location) can be used to facilitate scene reconstruction from multiple video streams.

4.3.8 Rendering Service

The visual display of information is an essential component of almost all collaborative and computational work engaged in by users. Many advanced applications, such as collaborative architectural design and visualization, will need to display very complex 3-dimensional scenes. To enable this kind of sophisticated application, our system will provide support for the real-time display and interactive manipulation of synthetic 3-D models within the Active Space.

Active Spaces present a significant challenge for the display of 3-D scenes. Numerous display devices will be distributed throughout the environment. They will also vary substantially in their display capacity, both in terms of resolution and rendering capacity. Although there is a single representation of an object, it must be displayed on systems with very high resolution (e.g., HDTV) and very low resolution (e.g., palm computers). It must also be rendered in real time on systems ranging from high-end servers with many processors to embedded devices with little processing capacity. Our approach for providing real-time display of complex 3D objects in this environment consists of two primary components: an adaptive object representation and a distributed rendering architecture.

Adaptive Object Representation. When attempting to provide real-time display of complex 3-D scenes, we are faced with a number of capacity constraints, including available CPU power, graphics throughput, network bandwidth, and storage space. And because of the extremely heterogeneous nature of the Active Space, these constraints will vary considerably within the environment. These physical limitations demand that we use methods for transmitting and rendering 3-D models that would automatically adapt both the representation and complexity of the models based on available resources. Building upon the QoS and object streams mechanisms provided by the Gaia OS services, the rendering system will support the adaptive delivery of 3-D content to all the client displays, while satisfying the resource constraints.

This need for adaptive models has been the primary motivation for recent work on multiresolution surface models [86]. Indeed, the new MPEG-4 standard already incorporates protocols for the compression and progressive transmission of geometric models. However, current methods provide for only limited adaptation of the surface model. We propose a considerably more powerful system, one that would accommodate multiple representations of the object, as well as multiple levels of geometric detail. The proposed system would be built upon a new appearance-based framework for analyzing surface models. Using appearance samples to guide multiresolution analysis of a surface has a number of important advantages. It focuses the method on the precise property of the object that we wish to preserve: its appearance. More importantly, it makes possible a more natural transition between geometric surface representations and image-based [87] [88] representations of the object. It is often convenient to convert very small-scale surface features (e.g., the pattern of indentations in brick) into textures [89]. Since the rendering system will be built on these continuum models, which combine both image-based and geometric representations, incoming image-based models of the physical environment, acquired via cameras, can easily be incorporated.

Distributed Rendering. The heavily distributed nature of the active space naturally suggests distributing the rendering process throughout the environment. For example, graphic workstation clients might perform all rendering locally, while palm devices would perform little or no rendering locally, instead relying on rendering servers. This requires both the distribution of graphical object models and the reintegration of partially rendered segments of the final image. For very complex static scenes, significant performance improvements can be gained by caching rendered pixels for later reuse [90] [91]. These pixel caches, from which similar views can be synthesized largely without re-rendering [92], are essentially partial light fields [93] [94] constructed at run time. Unlike similar prior systems, these pixel caches would be distributed throughout the environment. Distributed rendering relies on the automatic configuration and resource management services to automate those rendering processes on the right platforms.

5 Schedule of Milestones

The research will be carried out in five phases of one year each. In each phase, the groups working under the guidance of each of the co-Pis and PI will produce at least one research paper describing their achievements and one technical manual describing the software resulting from the research.

Fall 2000

Using the Unified Modeling Language (UML), produce an initial detailed design of the overall architecture of the Gaia OS and its constituent services.

Implement an initial prototype of the Gaia OS skeleton integrating minimal, simplified versions of its services.

Document the difficulties in the integration of the various services and suggest improvements in the design.

Design rendering and tracking services.

Fall 2001

Refine the initial Gaia design and prototype implementation with the experience from the first year.

Start the integration of Gaia services among themselves (for example, provide security for the QoS-aware resource management service and control the QoS of the security services).

Start development of test applications for the active classroom including support for distance learning.

Build prototype rendering and tracking services.

Construct a prototype Active Space for Siebel Hall.

Fall 2002

Extend the functionality of the individual Gaia services while keeping them integrated.

Enhance the test applications to benefit from the additional service functionality.

Develop a stable version of Gaia and the test applications, analyze their performance in an experimental production environment, and refine the whole system, preparing it for deployment.

As possible, start the deployment of parts of the Active Space infrastructure into offices and classrooms in Siebel Hall, the new Computer Science building.

Fall 2003

Deploy the complete Gaia OS and applications in Siebel Hall.

Collect feedback from the people involved in the deployment of our software. Analyze the usability of the software infrastructure and applications from the point of view of (1) administrators responsible for the deployment of our system into the new building, (2) administrators responsible for operating the software, and (3) end-users.

Monitor the production system and collect data about system and application usage. Analyze the data comparing to previous results in the development environments.

Based on the human feed-back and on the analysis of the collected data, refine the software infrastructure and application design.

Fall 2004

Continue with system refinement, optimizing it for performance and usability.

Collect final data for performance evaluation and investigate opportunities for technology transfer and future research.

6 Role of team members

Professor Campbell will lead the overall research direction of the project. Professor Nahrstedt and Professor Kravets will research quality of service and seamless wired and wireless networking. Professor Mickunas will lead security research. Professor Kriegman and Professor Garland will research sensing and tracking respectively. Professor Reed will lead application research.