Infrastructure for Pervasive Computing: Challenges[1]

Karen Henricksen[†‡], Jadwiga Indulska[†‡] and Andry Rakotonirainy[‡]
[†]School of Computer Science and Electrical Engineering, The University of Queensland
[‡]Distributed Systems Technology Centre
Email: {karen, jaga}@csee.uq.edu.au, andry@dstc.edu.au

Abstract: *As mobile and embedded computing devices become more pervasive, it is becoming obvious that the nature of interactions between users and computers must evolve. Applications need to become increasingly autonomous and invisible, by placing greater reliance on knowledge of context and reducing interactions with users. Moreover, applications must cope with highly dynamic environments in which resources, such as network connectivity and software services, frequently vary over time. This paper presents our vision of pervasive computing and enumerates the software engineering challenges involved in realizing this vision. It also evaluates the current state of research and presents an agenda for future investigations in pervasive computing.*

1. Vision

In the future, the computing landscape will evolve into an environment in which computers are autonomous devices that provide largely invisible support for tasks performed by users. Networked computing devices will proliferate in this landscape, being embedded in objects ranging from home appliances to clothing, and users will no longer be tethered to a single computing device. The nature of devices will change to form augmented environments in which the physical world is sensed and controlled in such a way that it becomes merged with the virtual world. Norman suggests that computing devices will also become more specialized in purpose, with each device being designed to solve a narrow set of well defined and interrelated tasks [1].

Applications will have greater awareness of context, and thus will be able to provide more intelligent services that reduce the burden on users to direct and interact with applications. The context of an application may include "any information that can be used to characterize the situation of an entity", where an entity is "a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves" [2]. Many applications will resemble agents that carry out tasks on behalf of users by exploiting the rich sets of services available within computing environments.

This paper addresses the research issues associated with this vision of the future computing landscape. The structure of the paper is as follows. Section 2 presents our assumptions about the pervasive computing environment. Next, Section 3 describes the requirements and challenges associated with our vision, while Section 4 presents a survey of existing approaches and evaluates them with respect to the requirements identified in Section 3. Section 5 describes an agenda for research towards pervasive computing and Section 6 presents some concluding remarks.

2. Assumptions

Our requirements for the pervasive computing infrastructure are centered on a high-level conceptual model consisting of devices, users, software components and user interfaces. The distinction between software components and user interfaces is an important one. While software components are programming units that are dynamically composed to form complete applications, user interfaces are conceptual entities that are responsible for interaction with the user, and which may be distributed over multiple software components and devices.

In the remainder of this paper, we outline software engineering issues associated with the four components of our model. Other important concerns in pervasive computing, such as security and privacy, communication infrastructure, network protocols for PANs, sensor networks and energy efficiency, fall outside the scope of

[1] The work reported in this paper has been funded in part by the Co-operative Research Centre Program through the Department of Industry, Science & Tourism of the Commonwealth Government of Australia.

the paper. For a discussion of communication issues in pervasive computing, see Esler et al. [3], who advocate an approach based on data-centric networking, and Arnold et al. [4], who describe an alternative approach using undirected, content-based routing.

The pervasive computing landscape will involve vast numbers of the four component types. A scalable supporting infrastructure will be required, in order to enable the dynamic discovery of software components and information; the dynamic interconnection of components; the sensing, interpretation and dissemination of context; the mobility and adaptation of components; and the rapid development and deployment of large numbers of software components and user interfaces.

3. Challenges

The four component types that make up our conceptual model of pervasive computing each present challenges, which place requirements on both the supporting infrastructure and the manner in which software components and user interfaces are constructed. These challenges are characterized in this section.

3.1 Devices

Two device-related challenges must be addressed by the pervasive computing infrastructure; these are the wide differences between heterogeneous device types and the problems caused by device mobility.

3.1.1 Device heterogeneity. We believe that heterogeneity in computing systems will not disappear in the future, but instead will increase as the range of computing devices widens. Devices in a pervasive computing environment will include sensors and actuators that mediate between physical and virtual environments; embedded devices in objects such as watches and shoes; home and office appliances such as videos, toasters and telephones; mobile devices, such as handheld organizers and notebooks; and traditional desktop machines.

Heterogeneous devices will be required to interact seamlessly, despite wide differences in hardware and software capabilities. This will require an infrastructure that maintains knowledge of device characteristics and manages the integration of devices into a coherent system that enables arbitrary device interactions (for example, between a mobile phone and a desktop workstation).

3.1.2 Device mobility. Mobility introduces problems such as the maintenance of connections as devices move between areas of differing network connectivity, and the handling of network disconnections. While protocols for wireless networking handle some of the problems of mobility, such as routing and handovers, some problems cannot be solved at the network level, as they require knowledge of application semantics. It should be the role of the computing infrastructure to cooperate with applications in order to perform tasks related to device mobility, such as management of replicated data in cases of disconnection.

3.2 Software components

The responsibility of the pervasive computing infrastructure with respect to applications includes supporting application requirements such as context awareness, adaptation, mobility, distribution and interoperability; facilitating the rapid development and deployment of software components; providing component discovery services; and providing scalability. This section addresses the challenges involved in meeting these requirements.

3.2.1 Mobility and distribution. As users can be mobile and able to exploit the capabilities of several devices simultaneously, mechanisms will be required to enable the mobility and distribution of software. These mechanisms should be largely transparent to component developers, who should not be concerned with program and data migration or synchronization and coordination of distributed components. The support for mobility will need to go beyond the current support for code migration provided by platforms such as the Java virtual machine (JVM), as run-time migration in heterogeneous execution environments will be required. Similarly, the support for distribution will need to surpass that provided by platforms such as CORBA, which only offer transparency of distributed communication, and typically do not address mobility, synchronization or coordination.

3.2.2 Context awareness. Invisibility of applications will be accomplished in part by reducing input from users and replacing it with knowledge of context. Context-aware software components will exploit information such as the activities in which the user is engaged, proximity to other devices and services, location, time of day and weather conditions. Knowledge of context will also be required to enable adaptation to changing environmental conditions, such as changing bandwidth and input and output devices, which can be brought about by mobility.

The infrastructure for pervasive computing should support context awareness by facilitating the gathering of information from sources such as sensors and resource monitors; performing interpretation of data; carrying out dissemination of contextual information to interested parties in a scalable and timely fashion; and providing models for programming context-aware applications. A very challenging aspect is interpretation, which involves steps such as integration of data from different sources (for example, combining height and horizontal position into a three dimensional position); inference (for example, "Bob is in the meeting room and Alice is in the meeting room, therefore a meeting between Bob and Alice is taking place"); prediction based on context history; resolution of inconsistencies between context data from different sources; and provision of estimates of the accuracy of contextual information.

3.2.3 Adaptation. Adaptation is required in order to overcome the intrinsically dynamic nature of pervasive computing. Mobility of users, devices and software components can occur, leading to changes in the physical and virtual environments of these entities. Moreover, applications can be highly dynamic, with users requiring support for novel tasks and demanding the ability to change requirements on the fly.

It should be the role of the infrastructure for pervasive computing to facilitate adaptation, which may involve adapting individual software components and/or reconfiguring bindings of components by adding, removing or substituting components. Adaptation may be done in an application-aware or application-transparent manner, as described by Noble et al. [5].

Dynamic adaptation can involve complex issues such as managing the adaptation of software components that are used simultaneously by applications with different (and possibly conflicting) requirements, and maintaining a consistent external view of a component that has behavior that evolves over time.

3.2.4 Interoperability. Today, application developers use a wide range of programming models, languages and development environments, and we foresee this heterogeneity continuing into the future, particularly as the range of uses for computing technology expands. Thus, the infrastructure for pervasive computing must support diverse types of software component. The infrastructure will be required to integrate software components, which may reside in fundamentally different environments (such as home or office computing environments), into compositions that can successfully interact and cooperate to achieve common tasks.

As applications in pervasive computing environments will be required to respond to novel tasks and situations, applications will increasingly be formed dynamically from available software components. This will require dynamic interoperability at the component level, in addition to interoperability that overcomes the heterogeneity of the environment and of components. Components will need to be capable of dynamically acquiring knowledge of each other's interfaces and behavior, in order to learn how to interact with previously unknown components.

3.2.5 Component discovery. The issue of discovery of software components has been addressed in various research areas. In open distributed computing, resource discovery is supported by a type management repository, which maintains descriptions of service interface types, and a trader, which is aware of instances of service types. Resource discovery is also addressed by network directory protocols such as LDAP, and in other technologies such as Jini and Bluetooth. As the problem of resource discovery has been solved differently in each of these domains, the main challenge in pervasive computing environments, which are characterized by their heterogeneity, will be to integrate the different approaches into a single scalable resource discovery system by mapping requests between resource discovery domains.

3.2.6 Development and deployment. The number and diversity of software components that will be required in pervasive computing environments will necessitate methods for their rapid development and deployment. Rapid development will be, in part, enabled by feature-rich infrastructure that obviates the need for application developers to be concerned with tasks such as adaptation, context gathering and management, resource discovery, distribution management and communication between distributed application components. Rapid development of specialized applications, such as agents, can be enhanced further by the development of special-purpose languages that enable applications to be specified at a very high level of abstraction. Several attempts to create such languages are already underway; one example is the XML-based Mobile Document Application Language (MoDAL), developed by IBM [6], for specifying small document-based applications for information appliances.

Infrastructural support for rapid application deployment can be achieved through the provision of execution environments into which applications can be placed without regard for configuration or adaptation. Rapid deployment of applications in distributed environments is already supported in a limited fashion by platforms such as the JVM, which can dynamically load and execute programs. However, these platforms do not yet meet the needs of heterogeneous environments, which require support for a broad range of component types, scalability, and dynamic configuration and adaptation of components.

3.2.7 Scalability. One of the features of pervasive computing is the increasing ubiquity of devices and software. Thus, the infrastructure, the interactions between components, and the software services provided in the pervasive computing environment must all be scalable. A powerful software platform on which scalable, fault-tolerant, distributed components can be built will be essential. Such platforms are already being developed; one such example is the Ninja service architecture [7] which is discussed in Section 4.2.

3.3 Users

Users in pervasive computing environments can be mobile and have computing sessions distributed over a range of devices. The infrastructure's role with respect to users should be to maintain knowledge of their context and to manage tasks related to their mobility.

3.3.1 Context. The infrastructure should maintain context data related to users, including their capabilities, preferences, current activities and active computing sessions. The uses of user-related context include allowing applications to provide adaptation to user requirements, and enabling the amount of input that applications require from users to be reduced. For example, with knowledge that a user is engaged in driving a car, an application can ensure interaction is carried out through a speech-based interface, rather than a screen-based one, in order to enable the user to focus on the road.

Information about users' computing sessions, including details about applications and the devices on which these applications reside, can be used to manage the application migration and adaptation that frequently must occur when a user is mobile. This is discussed in the following section.

3.3.2 Mobility. User mobility between devices should be supported by enabling automated migration (or re-instantiation in a new location) of application components. The tasks of identifying the need for application migration and then carrying out the migration, as described in Section 3.2.1, should ideally be performed by the computing infrastructure in a manner that makes the migration as transparent as possible to the applications concerned.

3.4 User interfaces

Users in pervasive computing environments will demand ubiquitous access to their computing applications, which will create a requirement for universally available user interfaces. Device heterogeneity will introduce a further requirement for user interfaces that are highly adaptable. Finally, the diminishing amount of user interaction with applications (brought about in part by the increasing ratio of applications to people) and the changing nature of the interactions (brought about by computing becoming situated in mobile and other novel situations) will mandate the creation of new types of user interfaces.

3.4.1 Universal Interfaces. The need for universally available user interfaces will create a requirement for new methods of programming user interfaces that do not make assumptions about the available input and output devices. GUI interfaces that are designed for use with a screen, pointing device and keyboard will no longer be broadly useful in future computing environments, as devices with novel input and output mechanisms (for example, touch screens and gesture recognition) will become increasingly common.

In order to provide scalable support for universal interfaces, it will be necessary for application programmers to write generic interfaces that allow the semantics of user interaction to be specified without reference to rendering or input modalities. Early efforts in this area are already emerging, such as MoDAL [6] which enables graphical user interfaces to be specified independently of platform and instantiated with regard for the context (including user preferences) at execution time.

3.4.2 Adaptation. User interfaces for pervasive computing environments must be highly adaptable in order to respond to changes in the available input and output devices caused by mobility; to other changes in the context in which the application is used (for example, when the user switches from working at a desk to driving a car); and to novel application behaviors that are created in a dynamic fashion from available components.

One of the challenges of providing user interface adaptation lies in ensuring that adaptation preserves a consistent view of an application. The user should have a uniform mental model of an application regardless of whether the user is interacting with a speech-based interface or a graphical one. Consistent adaptation is particularly challenging when the interaction paradigms are completely different. For example, interactions in gesture-based interfaces occur over a continuous period, whereas interactions in a mouse-driven interface occur at discrete points in time, which makes it difficult to map between the two forms of input [8].

Another challenge lies in dynamically coordinating the use of heterogeneous collections of input and output devices to form a single user interface (for example, when a control interface for a video conferencing system is formed using a user's PDA as an input device and the videoconference screen as the output device).

3.4.4 Usability. User interfaces for pervasive computing must be carefully designed with several factors in mind. First, the ergonomics of the interface must be designed to keep the user's attention focused on the task at hand rather than on peripheral matters; that is, the interface should not be distracting. Second, the user interface should be rewarding and enjoyable to use. Third, the user interface must allow novel types of interaction that will become more common as computing tasks become increasingly ubiquitous, such as delegation of tasks and provision of guidance to software agents. Finally, user interfaces should be designed for ordinary people, rather than just for technologists [1].

Table 1. Primary areas of focus

Challenge	***Issue***	*Context Models*	*Service Platforms*	*Appliance Environments*	*Pervasive Computing Environments*
1. Support for devices	*1.1 Support for device heterogeneity*				
	1.2 Support for device mobility				
2. Support for software components	*2.1 Management of application mobility and distribution*				
	2.2 Support for context-aware components	●			
	2.3 Support for adaptation		●		●
	2.4 Support for dynamic interoperability of components		●	●	●
	2.5 Support for component discovery		●	●	●
	2.6 Support for rapid component development and deployment		●		●
	2.7 Support for scalability		●		
3. Support for users	*3.1 Management of user context*	●			
	3.2 Support for user mobility				
4. Support for user interfaces	*4.1 Support for universal interfaces*			●	
	4.2 Management of user interface adaptation				●
	4.3 Support for usability				

4. Discussion of existing approaches

We have described four classes of challenges involved in realizing pervasive computing. These are summarized in the leftmost two columns of Table 1. The challenges have been partially addressed by existing research, a sample of which is surveyed and evaluated in this section. The discussion is partitioned according to the primary aims of the research efforts; note that overlaps exist between some of the categories (notably, between service platforms and appliance environments).

4.1 Context Models

Many context models have been developed to support context-aware and adaptive systems and applications. These primarily address challenges 2.2 and 3.1 in Table 1 by providing context representation, interpretation and dissemination.

The Sentient Computing project is concerned with supplying context information to applications, with particular focus on location information [9]. The location of mobile objects, such as people and equipment, is tracked by devices known as Bats (a successor to the earlier active badges), which communicate with base stations by ultrasound. Other context information is gathered by resource monitors, which track resources such as CPU, memory and bandwidth.

Context is associated with a logical model of the physical world. In this model, real world entities are captured as objects that have types, names, capabilities and properties, including static and dynamic context. Objects are stored in a persistent database and queried through a CORBA proxy. Additionally, applications can receive notifications of location-related events from a spatial monitoring service, which performs interpretation of location data and detects important events defined by containment rules.

One of the drawbacks of the Sentient Computing framework is its focus on location. While the framework provides interpretation and event notification of location changes, its support for other types of context is limited to the ability to query the information via the proxy server. In order to support the rich context requirements of pervasive computing software, the means to apply context interpretation to arbitrary types of context are required.

Hewlett-Packard’s Cooltown project proposes a Web-based model of context. In this model, entities (people, places and things) have Web representations that can be retrieved using a URL [10]. An entity’s Web representation captures both static and dynamic aspects of context, including relationships with other entities and sets of services associated with the entity. One of the primary aims of the model is to enable adaptation of Web content according to user context. However, the potential uses of the framework are much broader.

Location awareness is based around the concept of a space. Beacons wirelessly transmit URLs corresponding to spaces, enabling devices near the beacons to discover and access their local spaces. Spaces are accessed through portals, which are responsible for providing access control and a gateway to the space's services. A space manager performs tracking of the devices located within the space at any point in time and generation of dynamic Web pages that reflect the current context.

The Cooltown context model has several limitations. First, it does not address the means of specifying context, but instead allows arbitrary Web descriptions, which renders machine processing of context difficult. Additionally, interpretation of context and subscription to context events are outside the scope of the model.

Unlike the Sentient Computing and Cooltown projects, the Context Toolkit project [11] focuses on programming with context rather than context representation. The Context Toolkit has the aim of providing abstractions for separating the gathering and processing of context from the use of context. The toolkit comprises three types of component: context widgets, which acquire context data from sensors; interpreters, which perform processing of context data, such as abstracting high-level information about a person's location from raw location coordinates; and aggregators, which combine context data from multiple sources.

None of the work carried out on context to date is adequate to satisfy the requirements of pervasive computing. The ideas of context modeling found in the Sentient Computing and Cooltown approaches, and those of context processing found in the Context Toolkit must be united into a scalable framework, and better programming models for context-aware applications, which support rich types of context-awareness and adaptation, must be created.

4.2 Service platforms

Service platforms typically aim to facilitate the rapid creation and deployment of services (challenge 2.6 in Table 1), while also offering dynamic service discovery (2.5), including the ability for clients to learn the capabilities of services (partially addressing 2.4). Some platforms also address issues of scalability (2.7) and adaptation (2.3).

Jini [12] is a service framework based on Java and RMI, which supports flexible, easily administered environments in which services can be added and removed dynamically. It offers a service model based on three components: an infrastructure for federating services in a distributed environment, a programming model for distributed services, and a set of system services, including a lookup service used by clients to locate required services.

MOCA [13], like Jini, aims to provide a dynamic service environment; however, it focuses on satisfying the requirements of mobile computing environments. MOCA provides dynamic service discovery (2.5), limited forms of adaptation to changes caused by mobility, such as disconnection from the mobile network (1.2, 2.3), support for device heterogeneity (1.1), and location-transparent access to services. The framework consists of two components that reside on the mobile device: the service registry, which is a repository of information about available services, and a set of core services that provide local file caching, file loading and application management.

MOCA's model of locating both service discovery and essential services on the mobile device was designed with disconnected operation in mind. Unfortunately, this model places considerable resource demands on the mobile device, rendering the framework unsuitable for extremely resource-poor devices.

Jini and MOCA address service provision within relatively small service environments, and ignore scalability. The Ninja service framework, in contrast, is designed for large-scale Internet services [7]. It provides a service platform that delivers scalability, fault-tolerance, distribution and composability of services. Services are written in Java, as in Jini and MOCA, according to a well-defined programming model.

Services are executed on top of the vSpace platform within cluster computing environments known as bases. Dynamic deployment of a service is performed by uploading the service into a base. Service discovery is supported by a hierarchical arrangement of service discovery services.

In addition to the service platform, Ninja offers adaptation to meet the needs of heterogeneous client devices. Adaptation is performed by active proxies that lie between the Internet server and the client, and can overcome problems such as disconnection, limited bandwidth, limited processing capacity on the client device and disparities in protocols or data formats used by clients and servers. Thus, Ninja is able to address challenges 1.1, 1.2 and 2.3.

While, Ninja and MOCA, and to a lesser extent, Jini, satisfy some of the requirements of pervasive computing, many other issues remain beyond their scopes, including context-awareness and user and user-interface issues.

4.3 Appliance Environments

The aim of models for appliance computing is to support interoperability among collections of appliances. HAVi [14] is a standard for home appliances (designed by a consortium of industry players in the appliance market) consisting of a set of APIs, services, and a standard for communication. HAVi's primary goal of providing a dynamic service environment in which software components can discover and interact with services closely resembles the goals of service frameworks such as Jini.

HAVi provides mechanisms for devices to discover, query and control other appliances on the home network, and provides system services such as message and event transfer. Application interfaces are specified in a programming language-independent IDL, and applications can be programmed in a variety of languages, with Java being the language of choice for applications that require portability (such as device controllers that can be uploaded and executed on a range of devices).

HAVi supports universal appliance controls in the form of Java applets (havlets). Control applets can be loaded from appliances into a device employed by the user as a remote control. This provides the user with uniform access to the appliance regardless of which device is used for control.

Universal user interfaces are also addressed by an IBM research project aiming to enable a user to employ a single PDA-like device, called a Universal Information Appliance, to interact with all services [6].

Instead of using Java applets to provide remote interfaces, the IBM approach relies on MoDAL, a new high-level application and user interface description language based on XML. MoDAL applications are uploaded dynamically into a user's device from their corresponding services, and are tailored to the user's device and preferences.

MoDAL applications are supported by an infrastructure comprising a MoDAL engine (the execution environment for applications, similar in purpose to the JVM), a local database responsible for storing data such as user preferences and passwords, and a communication middleware based on a shared tuple space (TSpaces). The benefits of the communication model are that it offers distribution transparency, can support a range of interaction types, including event and stream interactions, and removes the need for resource discovery. Unfortunately, unless communication based on distributed tuple spaces can scale to large systems, which remains to be demonstrated, it is unsuitable for invisible computing environments.

The primary challenges addressed by IBM's universal information appliance infrastructure and HAVi are dynamic interoperability of components (2.4) and dynamic component discovery (2.5). Additionally, they address the provision of appliance interfaces that are universally available regardless of computing device (4.1). At present, however, they consider only graphical user interfaces, whereas pervasive computing will demand a much broader range of interface types.

4.4 Pervasive Computing Environments

This section surveys other work that broadly addresses the pervasive computing goal of providing "anytime, anywhere" computing by decoupling users from devices and viewing applications as entities that perform tasks on behalf of users. There are numerous ongoing projects in this area, including PIMA [15], Aura and Portolano [3].

PIMA is founded on the idea that the application model for pervasive computing must decouple application logic from details that are specific to the run-time environment, such as specific services and user interface renderings. Application functionality is modeled in a generic fashion as tasks and sub-tasks joined together by navigation mechanisms.

The PIMA project has several ongoing research thrusts. These include the creation of an application development environment that supports the device-independent application model described above (challenge 2.6) and the construction of service environments that provide applications with access to local services (2.5). PIMA also aims to address application adaptation (2.3), including rendering for specific devices (4.2) and dynamic application apportioning (that is, determining the split of functionality between client and server at execution time according to context). Finally, PIMA aims to create mechanisms that allow applications to learn how to interact with previously unknown services at execution time (2.4).

The Aura pervasive computing project is the successor to the earlier Coda and Odyssey projects on adaptation. Like PIMA, Aura proposes a programming model for task-based computing [16]. In this model, tasks are viewed as compositions of services. Both tasks and services have explicit representations. Services are described by virtual service types, which define functional, state and configuration interfaces and dependencies upon other services. Virtual service types can be related through inheritance, and can also be composed to form new virtual services. Tasks are top-level compositions of services that are specified as flows that decompose tasks into steps of subtasks or primitives (actions carried out by services).

Tasks are instantiated by a protocol that is responsible for gathering information about available services, selecting suitable services to carry out tasks and binding them together, and, finally, performing configuration and initialization of services. A coordination protocol manages the plugging and unplugging of services in response to resource changes. Tasks are also managed by a third protocol responsible for task migration, obtaining consistent snapshots of task state, and managing replication and consistency.

Like PIMA, Aura addresses the development and deployment of applications (2.6), application adaptation (2.3), and dynamic service discovery (2.5).

The Portolano project [3], in contrast to PIMA and Aura, primarily addresses issues of infrastructure rather than of software development. The Portolano group proposes the use of data-centric networks, an approach based on active networks in which data packets are responsible for traversing the network and obtaining required resources inside the network. The group is also considering infrastructural issues such as service discovery and proxy architectures that support resource-poor devices, and has an interest in applications such as location tracking of objects, gathering of data from sensors and applications of embedded Web servers. The Portolano research currently remains in its early stages.

4.5 Discussion

This section has surveyed four active research areas that address some, but not all, of the requirements of pervasive computing. The main goals and focuses of each of the research areas are summarized in Table 1. The table demonstrates that, while some of the ingredients for pervasive computing infrastructures are present in existing research, a considerable challenge remains in producing solutions to problems that have so far been little addressed, and in constructing a computing infrastructure that integrates solutions to all of the challenges we have described.

5. Research Agenda

In summary, pervasive computing has the following requirements:

1. the ability to dynamically discover and compose software components in frequently changing environments
2. the ability to support increasingly autonomous and invisible applications through the provision of rich context information that is gathered from a wide range of sources, interpreted, and disseminated in a scalable fashion to interested parties
3. the ability to rapidly develop and deploy flexible software components that are adaptive and context-aware and, additionally, satisfy special requirements such as scalability and fault-tolerance
4. the ability to integrate heterogeneous computing environments, which have differing communication protocols and services (such as discovery mechanisms), into coherent pervasive computing systems that enable the formation of dynamic interactions between components
5. the ability to construct novel types of user interfaces that are universally available, regardless of the input and output capabilities of the available devices, that are sensitive to situation, and that are non-distracting

These requirements are being addressed individually in a broad range of research disciplines, including mobile and distributed computing, software engineering, collaborative work, context-awareness, wearable computing and HCI; a sample of this work was characterized in Section 4. However, there remain considerable hurdles to realizing the pervasive computing vision. We believe that several research directions, in particular, must be pursued.

First, there is a need for new approaches to software design and development. Banavar et al. have described a programming model for pervasive computing in which generic programs are created without reference to device or service characteristics [15]. We believe that this model is not sufficient for invisible computing, in which applications are reliant on rich knowledge of context in order to function autonomously. Therefore, we advocate the development of new models for programming with context. Previous work on the gathering, interpretation and representation of context must be extended by deeper investigation into the use of context data. Abstractions of application context-awareness and adaptation must be formed which can serve as the basis for new application programming models and languages. These issues are currently the focus of our own research group, which is constructing a comprehensive framework for context-aware computing that supports concerns through from context gathering and interpretation to programming issues. This framework is being developed in conjunction with *Mercury*, an application supporting context-aware communication in pervasive computing environments, which serves as a testbed for our ideas about context-aware computing.

Second, a scalable framework for overcoming heterogeneity and enabling dynamic interactions between software components is required. This framework must build upon and broaden the scope of existing work on creating interoperability between computing environments such as CORBA and DCOM. It must address device and software heterogeneity, as well as heterogeneity of components such as networking infrastructures, middleware platforms and service discovery mechanisms. In addition, mechanisms are needed that enable software components to respond to novel situations by dynamically learning the interfaces and behaviors of other components.

Finally, further investigation into the user interface and usability issues of pervasive computing is required. Current research in the HCI arena is addressing the construction of user interfaces for innovative devices, including information appliances and mobile computers. From this work must emerge new models of user interaction to replace the models, centered on desktop computing, that have been widely used in the past.

6. Concluding Remarks

In this paper, we have presented a vision of a future computing landscape characterized by the ubiquity of computing devices, the autonomy, dynamicity and context-awareness of computing applications and the heterogeneity of system components. We have provided a discussion of the challenges associated with such a vision, framed around our conceptual model of pervasive computing which encompasses devices, users, software components and user interfaces. Additionally, we have evaluated the current state of research and the degree to which it satisfies the requirements of pervasive computing. We have concluded with an agenda for future research that highlights the need for further investigations into context-awareness and adaptation; integration frameworks that overcome system heterogeneity; and user interface models for future computing environments.

7. References

[1] Norman, D., "The invisible computer: why good products can fail, the personal computer is so complex, and information appliances are the solution", MIT Press, 1998.

[2] Dey, A. and Abowd, G., "Towards a Better Understanding of Context and Context-Awareness", *Workshop on the what, who, where, when and how of context-awareness at CHI 2000*, April 2000.

[3] Esler, M. et al., "Next Century Challenges: Data-Centric Networking for Invisible Computing", *Proceedings 5th Annual Intl. Conference on Mobile Computing and Networking (MobiCom'99),* August 1999.

[4] Arnold, D. et al., "Discourse with Disposable Computers: How and Why you will talk to your tomatoes", *Usenix Workshop on Embedded Systems,* March 1999.

[5] Noble, B. et al., "Agile Application-Aware Adaptation for Mobility", *Proceedings 16th ACM Symposium on Operating Systems and Principles,* October 1997.

[6] Eustice, K. et al., "A Universal Information Appliance", *IBM Systems Journal,* Vol. 38, No. 4, 1999.

[7] Gribble, D. et al., "The Ninja Architecture for Robust Internet-Scale Systems and Services", *Computer Networks, Special Issue on Pervasive Computing,* June 2000.

[8] Benyon, D., "The new HCI? Navigation of Information Space", To appear, *Special Issue of Knowledge-Based Systems*.

[9] Harter, A. et al., "The Anatomy of a Context-Aware Application", *Proceedings 5th Annual Intl. Conference on Mobile Computing and Networking (MobiCom'99),* August 1999.

[10] Kindberg, T. et al., "People, Places, Things: Web Presence for the Real World", http://www.cooltown.hpl.hp.com, Hewlett-Packard Labs Technical Report HPL-2000-16, 2000.

[11] Dey, A., Salber, D. and Abowd, G., "A Context-based Infrastructure for Smart Environments", *Proceedings 1st Intl. Workshop on Managing Interactions in Smart Environments (MANSE'99),* December 1999.

[12] Waldo, J., "Jini Technology Architectural Overview", White Paper, Sun Microsystems, Inc., January 1999.

[13] Beck, J., Gefflaut, A, and Islam, N., "MOCA: A Service Framework for Mobile Computing Devices", *Proceedings International Workshop on Data Engineering for Wireless and Mobile Access,* August 1999.

[14] Lea, R., Gibbs, S., Dara-Abrams, A. and Eytchison, E., "Networking Home Entertainment Devices with HAVi", *Computer,* Vol. 33, No. 9, September 2000.

[15] Banavar, G. et al., "Challenges: An Application Model for Pervasive Computing", *Proceedings 6th Annual Intl. Conference on Mobile Computing and Networking (MobiCom 2000),* August 2000.

[16] Want, Z. and Garlan, D., "Task-Driven Computing", Technical Report, CMU-CS-00-154, School of Computer Science, Carnegie Mellon University, May 2000.