TOWARDS A SERVICE-ORIENTED ARCHITECTURE
FOR MOBILE INFORMATION SYSTEMS

K. Rehrl, M. Bortenschlager, S. Reich, H. Rieser, R. Westenthaler
Salzburg Research, Jakob-Haringer Strasse 5, 5020 Salzburg, Austria

Abstract: Although hardware and networking infrastructures have evolved over the years and people are able to connect their devices to mobile networks and exchange information, we argue that the missing glue to enable the true potential of mobile information systems lies in the seamless integration of wireless infrastructures with existing wired infrastructures. In our paper, we present the service-oriented middleware *Asomnia*, which adapts traditional service-oriented concepts in order to cope with requirements arising from mobile computing challenges.

Key words: service-oriented architecture; mobile service environment.

1. INTRODUCTION

With the emergence of mobile information technologies, we are continuously heading towards Mark Weiser's vision of ubiquitous computing (Weiser, 1991). Mobile end user devices and wireless connection technologies allow for an anytime, anywhere vision (Kleinrock, 1996) and recent developments (Ferscha and Vogl, 2002; Garlan et al., 2002; Strang, 2003; Waldo, 1999) show remarkable approaches towards mobile computing. However, these developments often face the problem to strive towards mobility in a rather isolated way. Mobile technologies are exploited in order to achieve similar functionality as in wired infrastructures, although capabilities of mobile devices or wireless connection technologies are rather limited and the limitations are not considered to disappear in the upcoming years. Thus we argue, that mobile information systems only can reach their expected advantages and user acceptance trough integration with existing

wired infrastructures, i.e. the full value of mobile computing can only be gained by making use of the advantages of a wired infrastructure such as reliability, availability, bandwidth, richness in capabilities in conjunction with the benefits of wireless technologies such as mobility, ad-hoc connectivity or context awareness (Banavar et al., 2000; Banavar and Bernstein, 2002; Raatikainen et al., 2002).

In this paper we argue for a service-oriented middleware (SOM) called *Asomnia* (*A S*ervice-*O*riented *M*iddleware for ambie*N*t *I*nformation *A*ccess) as an approach for bridging the gap between the wired and wireless world. The paper starts with an overview of the main concepts and characteristics of existing service-oriented architectures (Section 2). In Section 3 we continue with a discussion on the characteristics considering pervasiveness and we state requirements for pervasive service-oriented architectures. Section 4 describes the architecture of *Asomnia* and in Section 5 we show two example applications. Section 6 gives an overview of related work and compares existing systems. We finish with a summary and conclusions (Section 7).

2. CHARACTERISTICS OF SERVICE-ORIENTED ARCHITECTURES

In order to describe requirements concerning service-oriented architectures for mobile information systems, in this section we first want to give an overview of characteristics of existing service-oriented architectures. (Bieber and Carpenter, 2001; McGovern et al., 2003; Strang, 2003) describe that services and service-oriented architectures deal with the issue of designing and building systems by using heterogeneous network addressable software components. Thus, service-oriented architectures are architectures that can be characterised by certain properties like being built of loosely coupled components or allowing for broad interoperability. Looking at the historical evolution, the term "service" has been used in many different architectures reaching from transaction monitors in the early 1990s to today's client-server architectures and web service architectures. Following the evolution process, service-oriented architectures have reached a degree of development where the basic concepts have been widely accepted and a set of properties for each concept has been defined to provide a more detailed characterisation of these architectures.

Typically, a service-oriented architecture consists of the following main concepts or any derivations of these (Bieber and Carpenter, 2001; Hashimi, 2003; Pallos, 2001; McGovern et al., 2003; Newmarch, 2000): **Service components (services)**, **Service contracts (interfaces)**, **Service containers**

(contexts), Service connectors (transports), Service discovery (registries).

Service Components or simply services are defined by network addressable software components. One characteristic, that is still open in service-oriented architectures, is the *granularity* of these service components. Fine-grained or micro (Strang, 2003) service components have the advantage to allow for improved reusability because of the atomic nature, but suffer from the disadvantage of being difficult to organise. Coarse-grained or macro services provide good encapsulation of functionality (e.g. information hiding), but are limited in their reusability. Hence, one trend in the conception of service components is to make the service components adaptable, which means that services can either be used in fine-grained or coarse-grained manner. Another important characteristic is the *mobility and on-the-fly deployment* of service components (Waldo, 1999), which also has an impact on the dynamic nature of applications built on the service components. Technically, the mobility of service components has been enabled with the advent of mobile code techniques (Gosling et al., 1996; Meijer and Miller, 2001). Another widely adopted characteristic is the *location transparency* of service components (Foster et al., 2002), which allows for a location independent service provision and hence assists the virtualisation of resources (Kagal et al., 2001).

One characteristic concerning **Service Contracts** or **Interfaces** is whether the service contract is *public or published* (Fowler, 2002). Typical representatives for architectures with published contracts are CORBA (OMG, 1991, JiniWaldo, 1999 or Web Services W3C, 2002), public service contracts are often used together with message or event-based systems like *Hermes* (Pietzuch and Bacon, 2002). Another characteristic of service contracts is how the contract is described (e.g. the description language used for the interface). Examples are the IDL of CORBA, Java interfaces used by Jini or WSDL used by Web Services. The choice of description language has great impact on *interoperability* issues of the architecture.

Service Containers deal with the issue of providing a common execution software environment for service components in order to achieve a high degree of modularity. Service containers can differ in the execution mode for services, meaning that execution of a single service as well as execution of multi services within one container is possible. We characterise containers by their execution modes as *single service containers* and *multi services containers*.

Service Connectors are used for exchanging messages between service components (Hashimi, 2003; Pallos, 2001. According to Stevens, 2002), interoperability is one of the crucial characteristics of service-oriented architectures. Beside the definition of universal interfaces, interoperability can be reached by choosing flexible service connectors. Service connectors are typically capable of various transport protocols and payload formats for

messages (Bieber and Carpenter, 2001; Stevens, 2002). Thus, we consider the capability for *various transport protocols* and *various payload formats* as a main characteristic for service-oriented architectures.

Service Discovery is another important concept for service-oriented architectures and as shown in (Bettstetter and Renner, 2000), various approaches exist. One important characteristic of the different approaches is the aspect of *dynamic service discovery* which is strongly related to the spatial focus of the architecture. Wide area service-oriented architectures like Web Services, CORBA or DCOM rather use more static discovery mechanisms like UDDI or CORBA trader, local area service-oriented architectures like Jini or MOCA (Beck et al., 1999) provide more dynamic discovery mechanisms which allow for services appearing and disappearing dynamically within little time periods. Another characteristic closely related to the dynamic aspects of service discovery can be described by the *hierarchy* of registries. Central registries imply a hierarchical network structure, whereas distributed registries (Beck et al., 1999) provide a flat network structure and are therefore also applicable to P2P networks. Closely related to the distributed mode is the characteristic of *autonomous discovery*, which describes whether the service-oriented architecture is dependent on central registry services or individual services are able to find each other autonomously.

In the next Section we will discuss adaptations of service-oriented architectures for mobile information systems.

3. REQUIREMENTS FOR PERVASIVE SERVICE-ORIENTED ARCHITECTURES

Service-oriented architectures have their origin in the 1990s and have shown their power in wired infrastructures like LANs or WANs. However, since the main characteristics are historically grown and mainly targeted to wired environments, existing service-oriented frameworks often lack the possibility to adapt to non-wired environments (OMG, 1991; Eddon and Eddon, 1998). This is also true for a group of recent developments (van Steen et al., 1999; Foster et al., 2002; W3C, 2002), especially designed for the use in wide area network applications like the WWW. Another group of more recent developments (Kindberg et al., 2002; Beck et al., 1999; Strang, 2003; Waldo, 1999) addresses aspects of mobility (Mattern and Sturm, 2003; Satyanarayanan, 1996); however, the integration with existing infrastructures and the adaptation of proven service-oriented concepts is often neglected. Thus, in this Section we want to describe important requirements addressing the necessary adaptations of service-oriented middleware for the use in mobile information systems.

3.1 R1 Overcome Heterogeneity by using Adaptive SOM

Heterogeneity in pervasive environments can be manifold, it can result in incompatibilities of devices, system platforms, communication networks or applications. Today's applications are typically developed for specific device classes and system platforms (Saha and Mukherjee, 2003), which leads to an increasing number of different versions of the same application. Adaptive middleware can bridge the gap between heterogeneous devices, system platforms and communication networks. As stated in Section 2, service-oriented middleware is traditionally well suited to overcome heterogeneity in wired networks and therefore can also solve incompatibility issues in pervasive environments. Service containers will enable device and platform portability, different service connectors allow for heterogeneous communication networks and universal service contracts reduce incompatibility issues between different applications. New challenges of heterogeneity in pervasive environments arise from mobility issues (Banavar and Bernstein, 2002; Henricksen et al., 2001; Satyanarayanan, 1996) and have to be addressed by the middleware.

3.2 R2 Integration of existing Computing Infrastructure and Computing Components

By now, applications on mobile devices are poorly integrated with existing computing infrastructures and computing components (Mattern and Sturm, 2003. As stated in Banavar et al., 2000), a mobile device should be a portal into an application or data space and not a repository for custom software components. In order to generate real added value for mobile users, existing computing infrastructure and computing components have to be integrated smoothly into mobile environments. For instance, wrappers or proxies can be used in order to integrate existing infrastructure for providing added value. To combine services to a portal view, as stated in (Banavar et al., 2000), is also a widely adopted user interface concept in service-oriented environments.

3.3 R3 Service-Oriented Architectures should incorporate a View on People, Places and Things

According to (Kindberg et al., 2002), nomadic people, places they enter and computerised things in these places are crucial to mobile computing environments and therefore have to be considered in the computing infrastructures. Service-oriented architectures typically only deal with virtual resources such as service components, which have no connection to the physical world. To be adapted to pervasive environments, service

components and service containers have to be semantically modelled (Banavar and Bernstein, 2002) under consideration of the hosting device (things), the location of the hosting device (places) and the current user (people). Thus, service provisioning has to come along with a conceptual model for managing devices in certain locations, groupings of services on devices or at locations and adoption to user profiles.

3.4 R4 Consideration of Mobility Aspects in Service-Oriented Architectures

Traditional service-oriented architectures typically do not consider mobility aspects. According to (Satyanarayanan, 2001), important issues include *mobile networking*, *mobile information access*, *support for adaptive applications* and *location and context sensitivity*. Service-oriented architectures are able to provide assistance in mobile networking and mobile information access through dynamic service discovery (Banavar et al., 2000) and reaching a certain level of autonomy for devices (Strang, 2003) (disconnected operation, caching of data). Support for adaptive applications is provided through on-the-fly deployment, laissez-faire or lazy loading of services (Satyanarayanan, 1996; Beck et al., 1999) and task-centred user interface services (Banavar et al., 2000). Based on the characteristics and requirements, in the following Sections, we give an overview on the system architecture and example applications of *Asomnia*.

4. SYSTEM ARCHITECTURE

The main goal of the system architecture was to overcome the deficiencies of existing service-oriented architectures as described above.

4.1 Network Structure

The network structure is designed as a 2- layered network scheme with hierarchy, where the first layer, the global domain, basically represents a highly reliable and available backbone managed by a central registry service and providing basic system services. The global domain consists of one central authority, a well known central registry, which hosts configuration settings of devices and services in the backbone as well as the current sub domains building up the second layer of the *Asomnia* network. Devices and services in each sub domain have access to a local registry, which is either well known by devices or found via network multicast. This allows mobile devices to register in currently available sub domains and is therefore

considered as a crucial requirement for pervasive environments. Moreover, the sub domains can be organised according to physical places, which allows for inherent location awareness of devices (cf requirement R3). The 2-layered hierarchy allows for consistency as well as scalability of the system structure.

4.2 Service Discovery

The 2-layered hierarchical registry system allows for dynamic registration of mobile devices. Based on the concept of local registries, mobile devices and services in the sub domain can find currently available devices and services in this domain (cf R4). Whereas typical service-oriented architectures do not consider location and characteristics of hosting devices, we consider (mobile) devices and places (sub domains) as important to be modelled in a mobile environment (Kindberg et al., 2002) (cf R3). In contrast to wired infrastructures, where devices are rather homogeneous and differ only little, mobile computing focuses on a huge variety of heterogeneous devices with greatly differing functionality. Moreover, an increasing set of pervasive services will be closely tied to certain devices (e.g. sensor devices) and places, where these devices are available. Thus, we argue, that our registry system allows for location aware discovery of devices as well as services virtually representing the functionality of these devices (cf R3). Moreover, our hierarchical registry system allows to combine the benefits of wired backbone infrastructures and wireless infrastructures within one middleware (cf R2).

4.3 Service Containers

Service containers in *Asomnia* provide a basic runtime infrastructure for services and fulfil the basic characteristics of portability and reuse of common functionality. However, to go a bit further, *Asomnia* uses service containers, which, depending on the executing device, are able to adapt to different roles. On wired devices, the service container adapts to the role of a single service container, i.e. each service on a device is running in its own service container, whereas on wireless, less powerful devices, the container adapts to the role of a multi service container (Fig. 1).

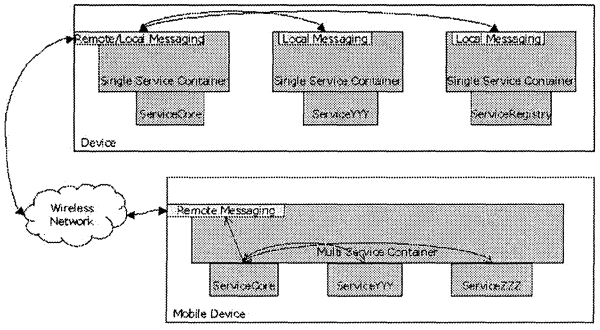

Figure 1. Single Service Container and Multi Service Container

Local and remote communication is handled by a local service broker called the *ServiceCore*. This approach ensures a maximum degree of reliability in wired infrastructures and reduced resource consumption on less powerful devices (cf R2, cf R4).

4.4 Devices and Service Components

Service components in *Asomnia* are typically rather coarse grained software entities. However, in order to allow for reusability, we have developed a concept for coarse grained services to be built of fine grained functional units (FU) (Fig. 2).

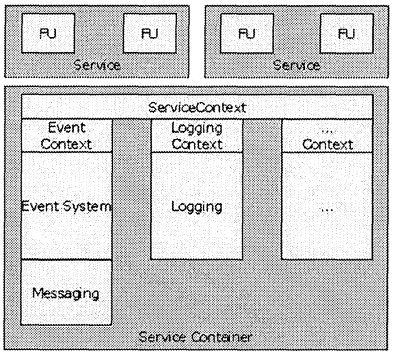

Figure 2. The layered architecture consists of functional units (FU), service components and the service container

Functional units are typically not visible from outside but are encapsulated in the service component to follow object oriented principles. New service components can be composed of a set of functional units at

design time, applications can be composed of service components at runtime leading towards adaptive applications (cf R4).

To address the requirement R3, we not only focus on service components, but also on modelling devices. Physical devices typically provide a collection of services, utilising the device's functionality. A device together with the service components is registered in the registry system. Thus, in order to deal with devices and services, any device needs to be controlled by a controller service i.e. *ServiceCore* (Fig. 1). The *ServiceCore* is the first service which is started for each device, and is used for management functionality. This includes starting and stopping of services on the device, registering the device at the registry system or maintaining the leases (Waldo, 1999) for the device. Additionally, the *ServiceCore* is responsible for remote communication, i.e. it is a kind of communication broker for the device. Registering devices in addition to services and providing one access point to the device has a few advantages:

- Easier management at the overlay network level (easier configuration of network and devices)
- Easier management at the network level (since all devices only communicate via the port of the *ServiceCore*, only this port has to be known to other devices and opened in the firewalls)
- Reaching a certain degree of autonomy of devices (cf R4)
- Services have a strong relation to devices and thus can also be related to places (cf R3).

4.5 Service Contracts

Asomnia uses a non-validated event exchange based on an XML-based event hierarchy together with an event description similar to (Pietzuch and Bacon, 2002). Currently we use the event hierarchy in a public way, future developments will focus on establishing explicit semantics by using a common ontology. Events can easily be serialised to messages and can be delivered either synchronously or asynchronously, which enables store and forward delivery upon network disconnection (cf R4).

4.6 Service Connectors

As described above, *Asomnia* uses event-based communication. Events allow for efficient delivery of information to other services either asynchronously or synchronously. Asynchronous event delivery is mostly used on mobile devices, whereas on wired infrastructures, synchronous delivery is the first choice due to the high level of availability and reliability in case that sender an receiver are both online. For reliable asynchronous communication on mobile devices we use store and forward delivery: For

network communication we use messages, which are events encoded according to an exchangeable payload format (*Formatter*). The messages are sent via exchangeable transport protocols (*Transport*). Both concepts allow for greater flexibility and are typically not provided by other service-oriented systems like Jini or MOCA. In our approach, for example, the default encoding for the wired environment may be XML, which allows for easier interoperability. However, for mobile devices, the XML-conversion may be to cumbersome, thus, an additional binary *Formatter* can be provided, which allows for higher performance in event encoding.

To sum up, since payload format and transport protocols are exchangeable, we are able to provide interoperability between heterogeneous devices on the network layer (Bluetooth, WLAN, infrared connection) as well as on the platform layer. It is possible with *Asomnia* to run each service in a different platform even on the same device.

5. PROTOTYPE AND DEMONSTRATION

According to the system architecture described in the previous Section we have implemented a prototype based on the .net Framework and the .net Compact Framework. In our prototype application scenario, a user with a PDA carries along a Powerpoint presentation and wants to take advantage of an existing video beamer equipment in a meeting room for presentation (cf R3). The user's PDA has access to the *Asomnia* enabled environment and after establishing a personal area network connection via Bluetooth or WLAN, the device automatically finds the local *Asomnia* registry by IP-multicast and starts to register itself. After the registration, the user is informed about available devices and services and uses the presentation control service running on the PDA in order to find the appropriate video beamer device and the proxy service which gives access to the Powerpoint application (cf R1). Upon connection to the service, the user starts uploading the presentation to the beamer device for getting immediately access to the slides from the PDA. During the presentation, the user can randomly jump to the different slides and is able to fully control the presentation on the PDA. If there would be a printer accessible from the computing environment, the presentation control service on the PDA would adapt to the situation (cf R4) and offer a possibility to print out slides. Due to the wireless connection of the PDA and the integration with existing infrastructures (cf R2), the user gets a feeling of pervasiveness and having control over the surrounding environment.

6. RELATED WORK

Our approach towards a mobile service-oriented middleware shares similarities with other work. In the following we will briefly highlight the most relevant ones.

The main goal of the (Heywow framework Strang, 2003), developed by the German Aerospace Center, is to provide location based information to mobile users. Heywow focuses mainly on service discovery and service description for services executed on mobile devices. Discovery and execution take into account the users' context. *Asomnia* differs from Heywow in two main aspects. We use exchangeable service connectors, i.e. transport protocols and payload formatters, since we argue, that HTTP transport and XML-parsers cannot be assumed for all kinds of mobile devices.

(Jini Waldo, 1999) provides a service oriented architecture for building distributed systems. It is based on the distributed computing mechanisms of sockets and remote method invocation (RMI). The intention is to offer a network plug and work mechanism where new services can join a network of other services and where service requestors can search for and use them (Newmarch, 2000). In contrast to *Asomnia*, even local communication between services is done via the lookup service (LUS) and therefore mobile devices can not reach autonomy. Furthermore, Jini assumes a Java Virtual Machine and provides only one type of service connectors, which results in rather few possibilities to integrate mobile devices.

TheWeb Services protocols (W3C, 2002) allow software services in a WWW environment to be deployed, discovered and accessed. Web Services are well suited for wired environments, but due to their rather static nature and resource consuming service containers less viable for low processing power devices in pervasive scenarios. The registry used with Web-Services - UDDI - is kind of static and not well suited for the dynamic requirements of mobile systems.

The *Open Grid Services Architecture* (OGSA) (Foster et al., 2002) by the Global Grid Forum builds on Web Services and provides a higher-level concept of services for grid computing infrastructures. In *Asomnia* we make use of plugable service connectors similar to the Web Services approach, however, we use lightweight service containers and dynamic registries.

CoolTown (Kindberg et al., 2002) offers a web model for supporting nomadic users and does this by tying web resources to physically present objects. The project strongly emphasis on common standards, however, it does not provide an architecture for a service-oriented middleware.

MOCA (Beck et al., 1999) provides a service framework that supports the development and execution of applications with a special focus on mobile computing devices. Basically, the concept is similar to Jini. The

service registry, however, is located and maintained at each participating device, which results in more flexibility with respect to mobility and in independence from a central entity. MOCA is fully implemented in Java and each device requires a JVM to be able to participate, which both restrict this framework regarding technology independence issues.

Some other systems or architectures influenced our work. The most important ones are JXTA and Bluetooth. JXTA (Gong, 2001) is a standard protocol set for a pure Peer-to-Peer (P2P) infrastructures. The mobile extension of JXTA called JXME (JXTA, 2003) seems to be appropriate for pervasive scenarios, however, due to the lack of central authorities in P2P systems, administrative and configuration issues are more complex. Bluetooth (Naveen and Yen, 2002) is a technology specification targeted to establish a standard to interconnect devices with a special focus on mobile devices. The Bluetooth stack spans a great variety of functionality ranging from hardware issues like radio frequency (RF) and coding schemes, basic software issues like addressing and network management, up to a sophisticated service discovery mechanism called the Service Discovery Protocol (SDP). Although SDP and service-oriented concepts are closely related to our work, Bluetooth is a hardware oriented low level protocol and thus considered to be rather an underlying technology compared to *Asomnia*.

A comparative overview of related work according to Section 2 is provided in the characteristics matrix (Table 1).

Table 1. Characteristics matrix (- not supported, + basic support, ++ full support)

	Heywow	Jini	OGSA	CoolT.	MOCA	ASOMNIA
Components: Granularity	+	+	++	-	+	-
Components: Mobility	+	++	-	-	+	++
Components: Transparency	+	++	++	++	+	+
Contracts: Interoperability	-	+	++	+	-	+
Contracts: Published	-	++	++	-	+	-
Containers: Portability	-	+	++	+	+	++
Containers: Service execution	+	+	n.a.	+	+	++
Connectors: Protocols	-	-	++	-	-	++
Connectors: Payload formats	-	-	++	-	-	++
Service discovery: Dynamic	++	+	-	-	++	++
Service discovery: Hierarchy	++	-	+	-	-	++
Service discovery: Autonomy	++	-	-	-	+	+

7. SUMMARY AND CONCLUSION

In this paper we have argued that there is a growing need for seamless integration of wireless and existing wired computing infrastructures. Existing developments in service-oriented middleware have mainly focused

on characteristics important for wired infrastructures (e.g. taking advantage of always available and reliable infrastructures) and do insufficiently cope with the requirements arising from pervasiveness, such as dealing with certain aspects of mobility (e.g. increasing heterogeneity and resource poverty of mobile devices, location awareness or error-prone wireless networks). To overcome these limitations, in our approach we propose a service-oriented middleware targeted to adapt the historically grown and well proven characteristics of service-oriented architectures in order to cope with the forthcoming challenges of pervasive environments. Our architecture addresses the differences of wired and wireless infrastructures using the concept of adaptive service containers and hierarchical registries. In conclusion, we believe that *Asomnia* provides a step towards a service-oriented architecture for seamless integration of wireless and wired computing infrastructures.

REFERENCES

Banavar, G., et.al. (2000). Challenges: an application model for pervasive computing. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 266–274. ACM Press.

Banavar, G. and Bernstein, A. (2002). Software infrastructure and design challenges for ubiquitous computing applications. *Communications of the ACM*, 45(12):92–96.

Beck, J., Gefflaut, A., and Islam, N. (1999). Moca: a service framework for mobile computing devices. In *Proceedings of the 1st ACM international workshop on Data engineering for wireless and mobile access*, pages pp. 62–68, Seattle, Washington, United States. ACM.

Bettstetter, C. and Renner, C. (2000). A comparison of service discovery protocols and implementation of the service location protocol. In *Proceedings of Sixth EUNICE Open European Summer School - EUNICE 2000*, Twente, Nederlands.

Bieber, G. and Carpenter, J. (2001). Introduction to service-oriented programming (rev. 2.1).

Eddon, G. and Eddon, H. (1998). *Inside Distributed Com*. Microsoft Press.

Ferscha, A. and Vogl, S. (2002). Pervasive web access via public communication walls. In *Pervasive Computing, Springer LNCS 2414*, pages pp. 84–97, Zurich, Switzerland.

Foster, I., Kesselman, C., Nick, J. M., and Tuecke, S. (2002). The physiology of the grid: An open grid services architecture for distributed systems integration. Draft 5, Mathematics and Computer Science Division, Argonne National Laboratory and Department of Computer Science, University of Chicago and Information Sciences Institute, University of Southern California and IBM Corporation.

Fowler, M. (2002). Public versus published interfaces. *IEEE Software*, Vol. 19(2):18–19.

Garlan, D., Siewiorek, D., Smailagic, A., and Steenkiste, P. (2002). Project aura: Toward distraction -free pervasive computing. *IEEE Pervasive Computing*, 4:pp. 22–31.

Gong, L. (2001). Project JXTA: A Technology Overview. http://www.jxta.org/project/www/docs/jxtaview 01nov02.pdf.

Gosling, J., et.al. (1996). *The Java Language Specification*. Addison-Wesley.

Hashimi, S. (2003). Service-oriented architecture explained. *O'Reilly ONDotnet.com*.

Henricksen, K., Indulska, J., and Rakotonirainy, A. (2001). Infrastructure for pervasive computing: Challenges. In *GI Jahrestagung Vienna*, pages 214–222.

JXTA, P. (2003). JXTA for J2ME. http://jxme.jxta.org.

Kagal, L., Korolev, V., Chen, H., Joshi, A., and Finin, T. (2001). Centaurus: A framework for intelligent services in a mobile environment. In *Proceedings of the International Workshop on Smart Appliances and Wearable Computing (IWSAWC)*.

Kindberg, T., Barton, J. J., Morgan, J., Becker, G., Caswell, D., Debaty, P., Gopal, G., Frid, M., Krishnan, V., Morris, H., Schettino, J., Serra, B., and Spasojevic, M. (2002). People, places, things: Web presence for the real world. In *MONET 7(5)*, pages 365–376.

Kleinrock, L. (1996). Nomadicity: Anytime, anywhere in a disconnected world. In *Mobile Networks and Applications 1*, pages 351 – 357.

Mattern, F. and Sturm, P. (2003). From distributed systems to ubiquitous computing - the state of the art, trends, and prospects of future networked systems. In Klaus Irmscher, K.-P. F. E., editor, *Proceedings of KIVS 2003*, pages pp. 3–25. Springer-Verlag.

McGovern, J., Tyagi, S., Stevens, M., and Mathew, S. (2003). *Java Web Services Architecture*. Morgan Kaufmann.

Meijer, E. and Miller, J. (2001). Technical overview of the common language runtime. *MSDN*.

Naveen, E. and Yen, D. C. (2002). Bluetooth Technology: A strategic Analysis of its Role in global 3G wireless Communication Era. *Computing Standards and Interfaces*, 24

Newmarch, J. (2000). *A Programmer's Guide to Jini Technology*. Springer-Verlag.

OMG (1991). The common object request broker: Architecture and specification. Technical report, Object Management Group (OMG), OMG Document Number 91.12.1

Pallos, M. S. (2001). Service-oriented architecture: A primer. *eAI Journal*, pages 32–35.

Pietzuch, P. R. and Bacon, J. M. (2002). Hermes: A distributed event-based middleware architecture. In *Proceedings of the 1st International Workshop on Distributed Event-Based Systems (DEBS'02)*.

Raatikainen, K., Christensen, H. B., and Nakajima, T. (2002). Application requirements for middleware for mobile and pervasive systems. *ACM SIGMOBILE Mobile Computing and Communications Review*, 6(4):16–24.

Saha, D. and Mukherjee, A. (2003). Pervasive computing: A paradigm for the 21st century. *IEEE Pervasive Computing*, pages pp. 25–31.

Satyanarayanan, M. (1996). Fundamental challenges in mobile computing. In *Proceedings of the 15th annual ACM symposium on Principles of distributed computing*, pages 1–7. ACM Press.

Satyanarayanan, M. (2001). Pervasive computing: Vision and challenges. IEEE Personal Communications.

Stevens, M. (2002). Service-oriented architecture introduction. *developer.com*.

Strang, T. (2003). Towards autonomous context-aware services for smart mobile devices. In Chen, M.-S., editor, *MDM 2003 (4th International Conference on Mobile Data Management)*, volume LNCS 2574, pages pp. 279–292, Melbourne, Australia. Springer.

van Steen, M., Homburg, P., and Tanenbaum, A. S. (1999). Globe: a wide area distributed system. *IEEE Concurrency*, 7:pp. 70–78.

W3C (2002). Web services activity. http://www.w3.org/2002/ws/.

Waldo, J. (1999). Jini technology architectural overview. Technical report, SunMicrosystems, 901 San Antonio Road, Palo Alto, CA 94303 U.S.A. Available as http://www.sun.com/jini/whitepapers/architecture.html.

Weiser, M. (1991). The computer for the 21st century. *Scientific American*, 265(3):66–75.