HKU CS Tech Report TR-2008-14

To appear in *Proceedings of the 2008 IEEE Asia-Pacific Services Computing Conference* (*APSCC 2008*), IEEE Computer Society Press, Los Alamitos, CA (2008)

A Tale of Clouds: Paradigm Comparisons and Some Thoughts on Research Issues*†

Lijun Mei
The University of Hong Kong
Pokfulam, Hong Kong
ljmei@cs.hku.hk

W.K. Chan ‡
City University of Hong Kong
Tat Chee Avenue, Hong Kong
wkchan@cs.cityu.edu.hk

T.H. Tse
The University of Hong Kong
Pokfulam, Hong Kong
thtse@cs.hku.hk

Abstract

Cloud computing is an emerging computing paradigm. It aims to share data, calculations, and services transparently among users of a massive grid. Although the industry has started selling cloud-computing products, research challenges in various areas, such as UI design, task decomposition, task distribution, and task coordination, are still unclear. Therefore, we study the methods to reason and model cloud computing as a step toward identifying fundamental research questions in this paradigm. In this paper, we compare cloud computing with service computing and pervasive computing. Both the industry and research community have actively examined these three computing paradigms. We draw a qualitative comparison among them based on the classic model of computer architecture. We finally evaluate the comparison results and draw up a series of research questions in cloud computing for future exploration.

Keywords*:* cloud computing, paradigm comparison.

1. Introduction

Cloud computing is a paradigm that focuses on sharing data and computations over a scalable network of nodes. Examples of such nodes include end user computers, data centers, and Web Services. We term such a network of nodes as a *cloud*. An application based on such clouds is taken as a *cloud application*.

This paradigm is increasingly popular in the industry, where industrial leaders such as Microsoft [26], Google [2], and IBM [5] strongly promote the paradigm in recent years. An early attempt to formulate cloud computing dates back to at least 1997 [8]. However, to our best knowledge, the adoption and promotion of cloud computing has been slow until 2007 [9].

We observe that the history of early industrial adoptions of cloud computing share some common milestones with that of service computing [4]. For example, it took service computing [27] a long time (ten years or so) to receive worldwide support from leading companies like IBM, Microsoft [25], BEA, and Oracle. Similarly, it has been many years since the early formalization effort [8] toward cloud computing.

Besides, the wide adoption of a computing paradigm usually depends highly on the maturity of supporting technologies and industry recognitions. Service computing has become much more popular since the success of Web services, although a Web service is only one of the technologies to fulfill the notion of service orientation [4]. Similarly, the distributed computing community has pointed out that many distributed computing techniques for cloud computing have been mature [7][10][11]. Many companies such as Dell and IBM have begun to ship cloud computing machines [5][10].

Last but not the least, in either service computing or cloud computing, research developments lag behind industrial adoptions. For instance, *COSCON*, a leading international container shipper, has a successful adoption of service computing. It successfully used service-oriented architecture to improve the business responsibility to customers in 2004 [3]. Yet, research studies in service-oriented architecture from the software engineering community [19] are still inadequate.

Despite our survey over the Internet, to our best knowledge, there are few articles to pinpoint research

† This research is supported in part by the General Research Fund of the Research Grant Council of Hong Kong (project nos. 111107, 717308, and 717506).

‡ Corresponding author.

issues in cloud computing. This would slow down the next research advances. We will alleviate this problem in the present paper.

In this paper, we use the classic computer architecture model [15] to provide a qualitative comparison framework to compare cloud computing with pervasive computing and service computing. The qualitative comparison framework includes three features: input-output (I/O), storage, and calculation. For each feature, we draw the comparison using multiple characteristics. Through such comparisons, we identify the connections between cloud computing and the other two computing paradigms from the perspective of software engineering. Based on the connections, we draw up a few research issues and discuss them in the paper to promote future exploration.

The main contribution of the paper is twofold: (i) To our best knowledge, we provide the first qualitative comparison on cloud computing, service computing, and pervasive computing. (ii) We present a series of research issues in cloud computing on top of the comparison framework. These issues promote future explorations.

The rest of the paper is organized as follows: Section 2 presents the preliminaries of cloud computing, service computing, and pervasive computing. Section 3 introduces our qualitative framework to compare the above three computing paradigms and present our efforts to identify research issues in cloud computing. Finally, we review related work in Section 4 and draw a conclusion in Section 5.

2. Preliminaries

This section reviews the preliminaries of cloud computing, service computing, and pervasive computing.

2.1. Cloud computing

As we have introduced in Section 1, a computing cloud is a massive network of nodes. Thus, scalability should be a quality feature of the computing cloud. It has at least two dimensions, namely *horizontal cloud scalability* and *vertical cloud scalability* (adapted from [9]).

- *Horizontal cloud scalability* is the ability to connect and integrate multiple clouds to work as one logical cloud. For instance, a cloud providing calculation services (*calculation cloud*) can access a cloud providing storage services (*storage cloud*) to keep intermediate results. Two calculation clouds can also integrate into a larger calculation cloud.
- *Vertical cloud scalability* is the ability to improve the capacity of a cloud by enhancing individual existing nodes in the cloud (such as providing a server with more physical memory) or improving the bandwidth that connects two nodes. In addition, to meet increasing market demand, a node can be gradually upgraded from a single power machine to a data center.

Scalability should be transparent to users. For instance, users may store their data in the cloud without the need to know where it keeps the data or how it accesses the data.

For simplicity, we will refer to horizontal and vertical cloud scalability, respectively, as *horizontal scalability* and *vertical scalability* in this paper.

2.2. Service computing

Service computing (or service-oriented computing) is an emerging paradigm to model, create, operate, and manage business services. In this paradigm, services publish themselves in public registries, discover peer services, and bind to the latter services to form service compositions using standardized protocols [6]. To create a service composition, engineers may use a specification, such as WS-BPEL [30], to model the collaborative need in workflows. To carry out individual workflow steps, software developers may use Web services, the most popular way to fulfill service-oriented architecture in the industry. A set of service-oriented applications over the Web services thus creates a network of services.

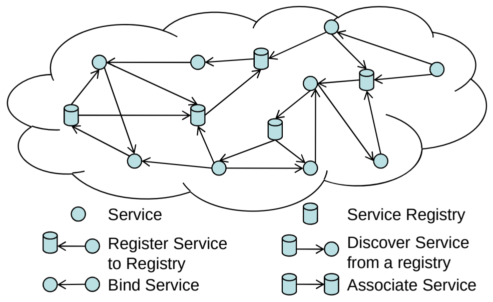
Service
Service Registry
Register Service to Registry
Discover Service from a registry
Bind Service
Associate Service

Figure 1. Service-oriented network [18].

We briefly describe a service-oriented network [18] to facilitate the comparison in the rest of the paper. An element in such a network is a service registry, service consumer, or service provider. A service provider registers itself in a service registry. A service consumer first discovers the service from a registry, and then binds to the service. A service provider may register itself to more than one registry. A registry may also associate its registered services to other registries, and acts as a service itself. Such a treatment on a registry provides a generic view among elements in service-oriented modeling.

2.3. Pervasive computing

Pervasive computing (or ubiquitous computing) [23][24] is another emerging computing paradigm. Software (often referred as *pervasive software*) can be embedded in a constantly changing computing environment. Therefore, pervasive software users do not need to be concerned about how to adjust the software to adapt to the surrounding computing environment. A well-developed environment will enable users to use pervasive software everywhere without extra effort.

To understand and react to a user, applications use environmental features, known as *contexts*, extensively. Sensors can capture these contexts. To allow ubiquitous support to end users, smart sensors are placed around users to preserve different information, such as the locations, contexts, and user-relevant data.

Figure 2 shows a pervasive computing example. Sensors, mobile phones and PDAs, desktop computer, and servers are interconnected logically to form an application. Suppose a nomadic user at the top left corner of Figure 2 moves from using a laptop to using a desktop computer. The laptop and the desktop computer both serve as UI portals to the tuple space maintained by the pervasive software. The remarked information from various display portals (such as the PDAs on the right-hand part) may need adapting. For example, a desktop computer may be equipped with a high-definition webcam. Thus, a presentation display portal may display the contents with a camera image kept in the tuple space of the application when using a laptop.

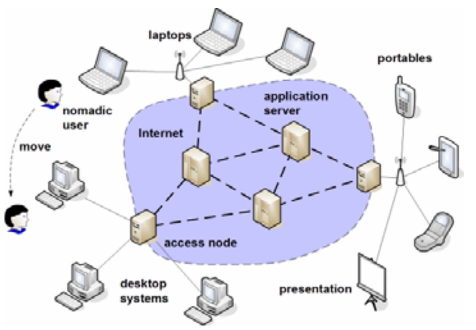

Figure 2. Pervasive computing environment [22].

3. Comparison of cloud, service, and pervasive computing paradigms

This section presents a qualitative comparison among the cloud, service, and pervasive computing paradigms.

Many researchers consider cloud computing as derived from grid computing [12] and have provided many comparisons between them [21]. To identify more issues for cloud computing, we choose to compare it with service computing and pervasive computing for the following reasons. Service computing is useful in modeling functionality and providing flexible services. Pervasive computing enables users to use software everywhere and provides self-adaptive capacity to the software with respect to environmental contexts. Cloud computing needs both functionality modeling and context-sensitivity. Through comparison with service computing and pervasive computing, therefore, we can gain insights on cloud computing.

Researchers (such as [13]) have applied the notion of virtual computers to model various computing entities and their interconnections. Such a treatment motivates us to analyze the key features of software applications (or services) of cloud computing. We thus compare cloud computing with service computing and pervasive computing from the perspective of computer architecture.

In classic computer architecture [15], a computer has three features: Input-Output (I/O), Storage, and Calculation. The descriptions of these features are as follows: (i) The typical computer-Input entities include the keyboard and mouse, and the computer-Output entities include, for instance, the monitor, printer, and speaker. (ii) In the Storage feature, there are storage entities such as the hard disk (internal storage) and USB (external storage). (iii) The key entity in the Calculation feature is the CPU.

We then show the representative characteristics in each feature of the three computing paradigms from the perspective of software engineering. Our understanding of service computing and pervasive computing is mainly based on our software engineering research [17][18][19] in these two paradigms. Our understanding of cloud computing is mainly based on our survey over the Internet.

We summarize the comparison results in Table 1. The key findings are as follows. We note that at least three notable likenesses of cloud computing from Table 1:

- The I/O feature of cloud computing resembles that of service computing.
- The storage feature of cloud computing is closer to that of pervasive computing than service computing.
- The calculation features of the three computing paradigms are similar.

Table 1. Comparisons in the framework of the classical model of computer architecture

	Model Dimension	General Characteristics
Cloud Computing	I/O	User requests and cloud responses
	Storage	Stored in the clouds collectively
	Calculation	Both intra-cloud calculations and inter-cloud calculations
Service Computing	I/O	Service requests and service responses
	Storage	Stored in specific service hosts
	Calculation	Performed by individual service compositions
Pervasive Computing	I/O	Situation detections and setup
	Storage	Stored in the tuple space of the application
	Calculation	Mainly performed by the entities embedded or connected to the surrounding environments

We thus use the above-identified likenesses to extract the main properties of service computing and pervasive computing to study cloud computing. Although the three paradigms are similar at a high-level, they still show differences in the details, as we will present below.

Tables 2, 3, and 4 show the comparisons of the key properties (that is, subfeatures) in the I/O, storage, and calculation features, respectively. Owing to the page limit, we will leave the comparisons of other properties in further publications. We only pick the key points of the main properties for discussions in this paper.

In the sequel, we will discuss the research questions indexed in Tables 2, 3, and 4.

Table 2. Comparisons in the *I/O* feature

	I/O Attributes	Description (Research Question Index)
Cloud Computing	Interface	Cloud interface (**Q1**, **Q2**) (not yet formally defined)
	Data Type	Cloud data type (**Q2**) (not yet formally defined)
	Synchronization	Synchronous or asynchronous I/O communication? (Not yet formally defined) (**Q2**)
Service Computing	Interface	Service interface
	Data Type	XML data which can be transferred using certain protocols (e.g., SOAP)
	Synchronization	Providing both synchronous and asynchronous I/O communications
Pervasive Computing	Interface	Interfaces with various devices in the environments (e.g., PDAs, mobile phones, and laptops)
	Data Type	Various data types (e.g., XML, WAP, GPRS, and Bluetooth)
	Synchronization	Providing both synchronous and asynchronous I/O communications

Table 3. Comparisons in the *Storage* feature

	Storage Attributes	Description (Research Question Index)
Cloud Computing	Location	Encapsulated in clouds. No explicit distinction between local and remote storage entities (**Q2**)
	Scale	The scale of intra-cloud storage and the inter-cloud storage (**Q1**)
	Access	Through cloud access (**Q2**)
Service Computing	Location	Encapsulated within individual services. Online storage is not the focus of service computing
	Scale	Depending on the storage scales of individual service hosts
	Access	Service requests
Pervasive Computing	Location	Explicit storage in the surrounding environments
	Scale	Depending on the storage scales in the environment or inter-connected to the environment
	Access	Through context communications

Q1. How do computing entities dynamically plug into a computing cloud?

In service computing, service providers dynamically register their services into the public service registries. Service consumers discover services from the registries and dynamically bind or unbind themselves to these services [18]. In pervasive computing, a mobile entity can move from one place to another and embed into different environments [17].

Similarly, in cloud computing, computing entities should be able to plug into a cloud dynamically. For example, when a large cluster of computer workstations and business services are attached to a cloud, the availability of computing entities in the cloud may change radically. How can a cloud application be entity-aware to plug-in heterogonous computing entities (which may also be different in modeling) with respect to various application parts? Techniques to alleviate this problem will strengthen the vertical scalability of applications.

Table 4. Comparisons in the *Calculation* feature

	Calculation Attributes	Description (Research Question Index)
Cloud Computing	Location	Encapsulated in clouds. No explicit distinction between local and remote calculation entities. (**Q1**, **Q2**)
	Context	Environment of individual entities in clouds. Unclear location of contexts. (**Q3**)
	Granularity	The entire cloud. (**Q4**)
Service Computing	Location	Encapsulated within individual services. Discovery of services via explicit service publishing, discovering, and binding mechanisms.
	Context	Environment of individual services and service compositions. Contexts are kept in services.
	Granularity	Usually, a small subset of all available services in a SON.
Pervasive Computing	Location	Explicit calculation discoveries via context communications.
	Context	Environment of individual computing entities. Contexts are kept in tuple space.
	Granularity	Ad hoc aggregations due to the changing nature of contexts and networks.

We thus need to address Q1 under the interface characteristic of the I/O feature, the scale characteristic of the Storage feature, and the location characteristic of the Calculation feature.

Q2. How do computing clouds store and access large-scale data?

Because a mobile entity in pervasive computing often has limited storage (see [23][24], for example), the entity usually stores its data in the surrounding environment. Similarly, every cloud has only a finite amount of physical storage entities. Therefore, a cloud c_1 may seek help from another cloud c_2 for shared storage entities to fulfill some demands on storage. Such sharing may result in storing a dataset using distributed data storage among multiple clouds.

Nevertheless, the cloud user should not be aware of the distributed storage of the data [14]. For instance, when the stored data needs to be accessed, the user may directly retrieve it from the cloud c_1. Then c_1 is responsible for gathering the data from both c_1 and c_2, and returns the collected data to the user. The cloud provides location transparency to applications.

From this scenario, we note that cloud data storage and access may need not only intra-cloud communications, but also inter-cloud communications. To do so, we also need a cloud-readable interface that supports extensible data type (such as XML Schema). The WSDL approach in

Web services may not work because it codes the entities explicitly in URL, which may not be compatible with the notion of location transparency in cloud computing. The scenario also raises the question of how a cloud application or service may publish its interface (such as where to publish). Moreover, since a calculation does not know the location of its required data, such data distribution among clouds may impose a huge performance penalty.

Thus, we should address Q2 under the three characteristics of the I/O feature, the location and access characteristics of the Storage feature, and the location characteristic of the Calculation feature.

Q3. How does a computing cloud become adaptive to both external and internal changes?

Both service computing and pervasive computing may meet environmental changes [16][18][24]. Besides, a service-oriented application is concerned with the evolving quality of individual services [18]. Pervasive software also needs to address the quality of the mobile entities involved [24].

Whenever a cloud detects a change in the environment, it should respond and adjust itself to achieve a better performance in the new environment. Besides, the vertical scalability of a node should evolve following the advancement of technology.

Q3 entails a few subquestions: (i) What are the suitable metrics to decide the effective adaption in cloud computing? (ii) What is a good strategy to model a cloud where the internal composition is intentionally blurry? (iii) How may cloud application designers specify a blurry calculation unit and its blurry environment? Traditional state machines or process models that detail the internal composition of a cloud may be inappropriate in addressing such vertical scalability issues.

Handling Q3 allows a cloud and a cloud application to strengthen their qualities and emergent properties in a more scalable manner. Besides, Q3 also leads to question Q4 below. Therefore, we need to address Q3 under the context characteristic of the Calculation feature.

Q4. How can computing clouds self-discover their quality?

The quality of either a service in service computing or a mobile entity in pervasive computing can change constantly (see [18][24], for example). Therefore, the quality of the resultant service-oriented application or the pervasive software may also change over time (see [17][18][27], for instance).

In cloud computing, each cloud may involve different computing entities. Because of the various types and qualities of the involved entities, the qualities of clouds can be different. For example, a cloud involving large computer workstations provided by large business companies are usually higher in quality than a cloud built on personal computers by college students. Both the internal status and external environment of a cloud change constantly. The ability of self-discovery of the quality features of the clouds provides opportunities for users to use computing entities efficiently and effectively. Suitable quality discovery mechanisms would help strengthen the vertical scalability of a cloud application.

Thus, we need to address Q4 under the granularity characteristic of the Calculation feature.

In summary, we present a few research issues that cover the key characteristics of all three features in the comparison framework. We note that the Calculation feature is exciting. We have successfully identified questions that are specific to it. Questions for the other two features are still mixed. The fusion of features may suggest that a good model of cloud computing application may need blurry I/O and Storage modeling.

4. Related work

This section reviews the literature related to our work.

First, we briefly review grid computing in general. The paradigm of grid computing is close to that of cloud computing [29]. Foster and Kesselman [12] present their understanding of grid computing. They [12] show how grids can solve research problems such as diagnostic problems and Aero-engine DP problems. Other researches on grid computing, such as [7][11], focus on the computing entity organization and computing task distribution. It would be difficult for an ordinary user to make use of such grid services. On the contrary, cloud computing highlights user experience in cloud services and encourages any users to use cloud services as if they were using their own computing laptops or PCs.

Second, we review context-aware approaches in general. Context-aware approaches are important in providing adaptive behaviors to software applications. Lu et al. [17] discuss testing pervasive software surrounded by different services. Mokhtar et al. [20] explain the problem of composition in the environment of pervasive computing. Lee et al. [16] propose to use a smart middleware architecture to hide the complexity involved with context-aware and automated service composition. Besides, Anhalt et al. [1] provide a general solution to address context awareness.

In cloud computing, large-scale computing entities will be placed on various host machines with different environmental contexts. The environmental contexts may heavily affect the performance of clouds. To guarantee the quality of cloud services, therefore, it is critical to develop techniques to address the environmental effect.

Our previous work [18] proposes to solve the service selection problem by link analysis techniques. In cloud computing, different computing entities also need evaluating and ranking. In this way, computing clouds will only use qualified entities. Such filtering process will increase the quality of computing clouds. Members in our research group have been studying both pervasive computing (in [17], for example) and service computing (in [18][19], for instance). We look forward to applying our privileged knowledge, achieved in studying pervasive computing and service computing, to solve the research problems in cloud computing.

5. Conclusion

Cloud computing is an emerging computing paradigm that is increasingly popular. Leaders in the industry, such as Microsoft, Google, and IBM, have provided their initiatives in promoting cloud computing. However, the public literature that discusses the research issues in cloud computing are still inadequate. To identify the research questions thus warrants more efforts.

In this paper, we have drawn up a qualitative comparison framework to compare cloud computing with pervasive computing and service computing. This framework positions on the classic model of computer architecture, which includes three features: input-output, storage and calculation. We further present a detailed comparison result for each feature and achieve the following identifications: (i) The input-output feature of cloud computing resembles that of service computing. (ii) The storage feature of cloud computing is closer to that of pervasive computing than that of service computing. (iii) The calculation features of the three paradigms are similar. Based on the comparison, we have drawn up a few research issues, such as pluggable computing entities to cloud applications, data access transparency, adaptive behavior of cloud applications, and automatic discovery of application quality. We have found the calculation feature to be interesting from the software engineering perspective.

In the future, we will continue along the direction of comparing cloud computing with other computing paradigms, and identify other emerging research issues. We also plan to provide a sound framework for reasoning and analyzing computing clouds and explore technical practices using our models.

References

[1] J. Anhalt, A. Smailagic, D. P. Siewiorek, F. Gemperle, D. Salber, S. Weber, J. Beck, and J. Jennings. Toward context-aware computing: experiences and lessons. *IEEE Intelligent Systems*, 16 (3): 38–46, 2001.

[2] S. Baker. Google and the wisdom of clouds. Available at http://www.businessweek.com/magazine/content/07_52/b4064048925836.htm. (Last accessed on Sept. 15, 2008.)

[3] M. Barnes. COSCON improves business responsiveness via service-oriented architecture. Gartner Research, 2006. Available at http://www-128.ibm.com/developerworks/blogs/resources/SOA_Off_the_Record/coscon_improves_business_res_139656.pdf. (Last accessed on Sept. 15, 2008.)

[4] B. Benatallah, R. M. Dijkman, M. Dumas, and Z. Maamer. Service-composition: concepts, techniques, tools and trends. In *Service-Oriented Software System Engineering: Challenges and Practices*, pages 48–66. Idea Group, 2005.

[5] Big blue goes for the big win. IBM. Available at http://www.businessweek.com/magazine/content/08_10/b4074063309405.htm. (Last accessed on Sept. 15, 2008.)

[6] M. Broy, I. H. Kruger, and Meisinger, M. A formal model of services. *ACM Transactions on Software Engineering and Methodology*, 16 (1): Article No. 5, 2007.

[7] R. Buyya. *Economic-based Distributed Resource Management and Scheduling for Grid Computing*. PhD Thesis, Chapter 2. Monash University, Melbourne, 2002.

[8] R. Chellappa. Cloud computing: emerging paradigm for computing. In *INFORMS 1997*. Dallas, TX, 1997.

[9] Cloud computing. Wikipedia. Available at http://en.wikipedia.org/wiki/Cloud_computing. (Last accessed on Sept. 15, 2008.)

[10] Dell cloud computing solutions. Available at http://www.dell.com/cloudcomputing (Last accessed on Sept. 15, 2008.)

[11] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: enabling scalable virtual organization. *International Journal of High Performance Computing Applications*, 15 (3): 200–222, 2001.

[12] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Elsevier, Amsterdam, 2004.

[13] A. S. Grimshaw and Wm. A. Wulf. The legion vision of a worldwide virtual computer. *Communications of the ACM*, 40 (1): 39–45, 1997.

[14] K. Hartig. What is cloud computing? *SOA World Magazine*. Available at http://soa.sys-con.com/read/579826.htm. (Last accessed on Sept. 15, 2008.)

[15] J. P. Hayes. *Computer Architecture and Organization*. McGraw-Hill, New York, 1998.

[16] C. Lee, S. Ko, S. Lee, W. Lee, and S. Helal. Context-aware service composition for mobile network environments. In *Ubiquitous Intelligence and Computing*, volume 4611 of Lecture Notes in Computer Science, pages 941–952. Springer, Berlin, Germany, 2007.

[17] H. Lu, W. K. Chan, and T. H. Tse. Testing pervasive software in the presence of context inconsistency resolution services. In *Proceedings of the 30th International Conference on Software Engineering* (*ICSE 2008*), pages 61–70. ACM Press, New York, 2008.

[18] L. Mei, W. K. Chan, and T. H. Tse. An adaptive service selection approach to service composition. In *Proceedings of the IEEE International Conference on Web Services* (*ICWS 2008*). IEEE Computer Society Press, Los Alamitos, 2008.

[19] L. Mei, W. K. Chan, and T. H. Tse. Data flow testing of service-oriented workflow applications. In *Proceedings of the 30th International Conference on Software Engineering* (*ICSE 2008*), pages 371–380. ACM Press, New York, 2008.

[20] S. B. Mokhtar, D. Fournier, N. Georgantas, and V. Issarny. Context-aware service composition in pervasive computing environments. In *Rapid Integration of Software Engineering Techniques*, volume 3943 of Lecture Notes in Computer Science, pages 129–144. Springer, Berlin, 2006.

[21] P.S. Narayanan. From grid computing to cloud computing: the IBM approach. Garuda Partner Meet, Bangalore, India, March 4, 2008.

[22] Nomadic computing example. Available at http://www.uni-marburg.de/fb12/verteilte_systeme/forschung/crossware. (Last accessed on Sept. 15, 2008.)

[23] D. Saha and A. Mukherjee. Pervasive computing: a paradigm for the 21st century. *IEEE Computer*, 36 (3): 25–31, 2003.

[24] M. Satyanarayanan. Pervasive computing: vision and challenges. *IEEE Personal Communications*, 8 (4): 10–17, 2001.

[25] Service orientation and its role in your connected systems strategy. Microsoft Corporation. Available at http://msdn.microsoft.com/en-us/library/ms954826.aspx. (Last accessed on Sept. 15, 2008.)

[26] Software via the Internet: Microsoft in 'cloud' computing. Microsoft Corporation. Available at http://www.nytimes.com/2007/09/03/technology/03cloud.html. (Last accessed on Sept. 15, 2008.)

[27] M. Stevens. Service-oriented architecture introduction. Available at http://www.developer.com/services/article.php/1010451. (Last accessed on Sept. 15, 2008.)

[28] T. von Eicken. The three levels of cloud computing. Available at http://pbdj.sys-con.com/read/581961. (Last accessed on Sept. 15, 2008.)

[29] P. Wallis. Cloud computing: is the cloud there yet? — a brief history. Available at http://soa.sys-con.com/read/581838.htm. (Last accessed on Sept. 15, 2008.)

[30] Web services business process execution language version 2.0. Available at http://docs.oasis-open.org/wsBPEL/2.0/wsBPEL-v2.0.html. (Last accessed on Sept. 15, 2008.)