A Preliminary Outline for a Ubiquitous Computing Software Infrastructure

Cristiano André da Costa
Universidade Federal do Rio Grande do Sul - UFRGS
Universidade do Vale do Rio dos Sinos - UNISINOS
cacosta@inf.ufrgs.br, cac@unisinos.br

Luciano Cavalheiro da Silva
Universidade Federal do Rio Grande do Sul - UFRGS
lucc@inf.ufrgs.br

Adenauer Corrêa Yamin
Universidade Católica de Pelotas - UCPel
adenauer@ucpel.tche.br

Cláudio Fernando Resin Geyer
Universidade Federal do Rio Grande do Sul - UFRGS
geyer@inf.ufrgs.br

Abstract

The proliferation of various types of computational devices (many of which include the possibility of wireless interconnection) allows a glimpse into a new area, which transcends the characteristics of the majority of distributed systems in use today. This area, called ubiquitous (or pervasive) computing, presupposes a strong integration with the real world, keeping high transparency and focus on the user. For the development of applications in this scenario, we need an appropriate software infrastructure. In this way, this article presents an outline of Continuum, a software architecture designed to support ubiquitous computing. The proposal applies follow-me semantics to let the user access applications and data at any time and place. The focus of our work is also on addressing context awareness and context management issues.

1. Introduction

In the classic and visionary article about computation for the 21st Century, Mark Weiser [11] summarizes what is expected from pervasive or ubiquitous computing (ubicomp): user access to the computational environment, everywhere and at all times, by means of any device. The difficulty lies in how to develop applications that will continually adapt to the environment and remain working, as people move or change devices [10]. The development of this area, however, is still hindered by the limited number of available languages and tools [9]. Besides, context-aware applications are still being executed in laboratories rather than present in everyday real-world environments [6]. Ubiquitous applications need middleware to interface between many different devices and end-user applications [7]. The aim is to hide environment complexity, by isolating applications from the explicit management of protocols, distributed memory access, data replication, communication faults, etc. Middleware can also solve heterogeneity problems related to architectures, operating systems, network technologies, and even programming languages, promoting their interoperation. On the other hand, a framework is an environment, composed of APIs, user interfaces, and tools, which simplifies software development and management in a specific domain [3]. We can use a framework to build software that runs on middleware, which can be developed by using existing frameworks. This middleware must allow the user to access her computational environment (applications and data) at any time and place. One possible solution is to apply follow-me semantics [1] [13]. The idea behind this concept is that applications and data go along with users, providing a virtual environment and adapting to the current context. This adaptation is fundamental to the ubiquitous computing vision, and involves the perception of context (context awareness) and the proper adjustment of the system based on this perceived information (context management). We defend that follow-me semantics can reduce the distance between the ubiquitous computing vision, as proposed by Weiser, and the current distributed computing scenario. To achieve this goal, our work focuses on the development of a software infrastructure for ubiquitous computing that addresses context awareness and context management issues. This proposal differs from other works, such as Aura [4], Gaia [9], and One.World [5], because the fo-

cus here is on keeping the computational environment near the user. In this article, we propose a software infrastructure, targeting ubicomp, named Continuum. It makes use of ISAM (Infra-estrutura de Suporte às Aplicações Móveis – Mobile Applications Support Infrastructure) [1] especially of EXEHDA middleware (Execution Environment for Highly Distributed Applications) [12]. Continuum infrastructure uses framework and middleware, and redesigns the current ISAM project, modifying the middleware EXEHDA to better support follow-me semantics, and the context awareness and management issues involved. The text is organized as follows. In the next section, we review essential concepts of the area and suggest challenges that must be addressed. In section 3, we describe follow-me semantics and the integration of grid in the ubiquitous computing field. Section 4 briefly presents the ISAM project. In section 5, we show an outline of the Continuum infrastructure based on ISAM redesign and context awareness issues. Section 6 briefly presents the Continuum pluggable services design. Finally, in section 7, we show some conclusions and suggestions for future work.

2. An Appraisal of Ubiquitous Computing

We should begin by defining ubiquitous computing (also called ubicomp). Mark Weiser created this term, so he is considered one of the area's fathers. He presents computer ubiquity as the idea of integrating computers seamlessly, invisibly enhancing the real world. Weiser [11] formulates a "new way of thinking about computers in the world, one that takes into account the natural human environment and allows the computers themselves to vanish into the background". Computers will vanish as a consequence of human psychology: when people use things without consciously thinking about it, they focus beyond them. This is a phenomenon defined by some philosophers and psychologists [11]: people cease to be aware of something when they use it sufficiently well and frequently. To achieve the physical integration of computers into the world, as a background, we must apply some conceptual changes. In this perspective, Weiser also defines embodied virtuality in opposition of virtual reality because computers cannot be limited to their devices and software installed. Moreover, it is inadequate to consider the Internet or distributed file systems access as an example of seamless integration. Weiser points out that the power of ubiquitous computing does not steam from the capacity of a particular device, but rather from the interaction of all devices. Besides computer interaction, Scale and Location are two important topic highlighted by Weiser. There will be many computers per room, in different sizes, with different user's interfaces, and suitable for specific jobs. Computers must also know where they are and with this they can adapt to the environment. Adaptation is then currently one of the crucial concerns in pervasive computing. We must understand and support everyday practices of people to reach Weiser's vision, offering different forms of interactive experiences through heterogeneous devices connected via integrated network components.

3. Follow-me Semantics

We propose a personal virtual environment that goes along with the user, in her movement, to achieve a wide ubiquitous scenario. This movement can involve both logical mobility (of data, code, and computation) and physical mobility (of resources and devices in use). This environment feature is known as follow-me semantics [1]. To support this concept, we propose the use of grid infrastructure, which allows us to easily provide global mobility of user, terminal, data, code, and session. Grid strategies are employed to manage the user's displacement and to accomplish the migration of her virtual environment. The integration of grid and ubiquitous computing is complex, due to the contrast between the dynamism of pervasive environment and the static resource management of traditional grid systems. We believe that we can improve both ubiquitous and grid computing with this convergence. In our infrastructure model, grid applications should adjust themselves to currently available resources, instead of looking for more, in order to satisfy their needs. We should bear in mind that such adaptation goes beyond the traditional grid solutions, which consist in adjusting the system to the number of available resources of a given kind. As an alternative, we propose that applications adapt to the kind of resource available at each moment. In terms of ubiquitous computing, the changes proposed are such that infrastructure manages a wider environment and makes decisions based on many context dimensions: temporal, spatial, personal, or even social. In addition to the grid, we propose the use of mobile and context-aware computing. The joint use of these three concepts allows us to manage a large-scale environment through the use of follow-me semantics. Moreover, this kind of environment makes it easier to develop ubiquitous applications.

4. The ISAM Project

ISAM is a Brazilian acronym for Infra-estrutura de Suporte às Aplicações Móveis (Mobile Applications Support Infrastructure), developed by researchers from Federal University of Rio Grande do Sul (UFRGS). The project aims at integrating the concepts of context-awareness, grid, and mobile computing [13]. The idea behind ISAM is to build a pervasive computing infrastructure, integrating a programming language and middleware to support its execution. Differently from other proposals, ISAM focuses on appli-

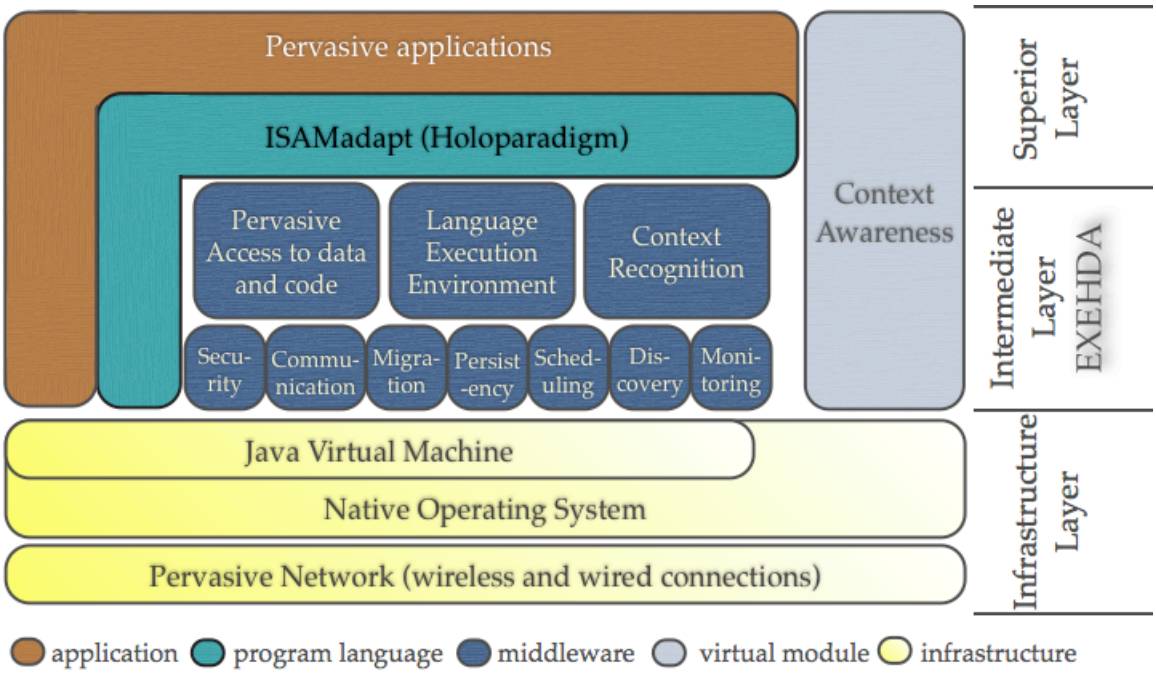

Figure 1. Continuum architecture

cation development rather than on the environment and services. Because of that, the project encompasses a model, a language, and a runtime support to build and execute pervasive applications. There is a prototype available , built mainly in Java, with some modules in C. The prototype is fully functional and bundled to a Linux live CD, in order to facilitate its use. Programmers can develop applications utilizing Java or ISAMadapt. ISAMadapt [1] is a programming language that facilitates the development of pervasive applications. It provides some means for expressing dynamic adaptation and context-awareness in design-time. ISAMadapt is based on a mulitparadigm model named Holoparadigm (hereafter simply referred as Holo) [2]. In Holo, a logic blackboard, called history, implements the coordination mechanism, and a new programming entity, called being, organizes several encapsulated levels of beings and histories (multi-domains). These "beings" are the main Holo abstractions. They represent the logical or physical components of the system that is modeled. The architecture of ISAM 1 is organized in three components: the infrastructure layer, the intermediate layer, and the superior layer. The infrastructure consists of the network, the operating system, and the Java Virtual Machine (JVM). Currently, ISAMadapt programs are converted to Java source code, then compiled, and executed in the JVM.

The intermediate layer is the Execution Environment for Highly Distributed Applications (EXEHDA). Designed as middleware, it consists of a collection of services, such as naming, communication, migration, replication, interoperability, location, and monitoring. On top of these basic services, EXEHDA executes the User Virtual Environment, a container for user applications and sessions; the Scheduler, for migration and remote execution of objects; and the Context Server, for context-aware adaptive behavior. The superior layer has the ISAMadapt and the distributed mobile applications. Context awareness is represented as a virtual module, since it is present in the conception of all other ISAM components [12]. Applications run on the ISAM pervasive environment (ISAMpe), which uses cellular hierarchy. Each cell has a specific host, called base, responsible for communications among cells. Devices belonging to the same cell can directly communicate with each other and are identified as nodes. The hierarchy allows a cell to recursively contain other cells.

5. Overview of Continuum Software Infrastructure

Continuum software infrastructure integrates the concepts of context-awareness, grid, and mobile computing. The idea behind Continuum is to build a ubiquitous computing infrastructure, integrating a framework and middleware to support its execution. Figure 2 illustrates the development process using Continuum. We decide on this name for the software infrastructure, in order to represent the idea of continuous execution obtained by the use of follow-me

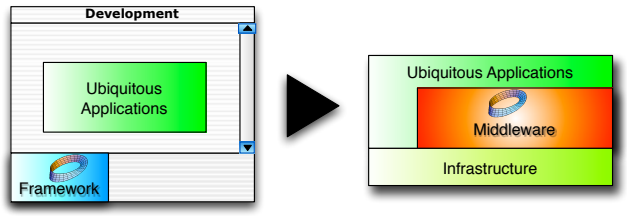

Figure 2. Development Process using Continuum

semantics. We choose the Möbius strip as the logo of our project because it represents a continuous curve.

The project is based on ISAM, acronym for Infraestrutura de Suporte às Aplicações Móveis (Mobile Applications Support Infrastructure). Differently from other proposals, ISAM focuses on application development rather than on the environment and services [1]. Because of that, the project encompasses a model, a language (named ISAMAdapt), and a runtime support to build and execute pervasive applications. Although Continuum is an ISAM evolution, there are conceptual differences between both. ISAMAdapt was discontinued, and Java is the only language currently supported. Programmers usually prefer to use languages they are already familiar with, allowing the use of legacy codes. Additionally, we want to support the context awareness without excessively burdening the programmer and of the software development process. We propose the use of a framework, instead of a language, to achieve these goals. The Continuum framework maintains all the characteristics we consider important to support the design-time development. Furthermore, the framework inherits some important characteristics of ISAMAdapt, in an independent language approach. Neither do we intend to be bound to Java. We aim to be as independent as possible, making it easier to port Continuum to other languages, such as C# or C++, in the future. The focus of Continuum is on context awareness, and also on sustaining follow-me semantics. As a result, we rebuilt EXHEDA middleware considering these aspects. The original proposal [12] focuses only on partial context awareness, i.e. the acquisition of raw information, its distribution, and the conversion of raw information into abstract context elements, guided by an XML description. Another important element in ISAM is informality in the treatment of context. Instead of treating context awareness as a virtual module, as in ISAM, we consider it as a real subsystem, which must additionally encompass the following characteristics: formal representation of context information; storage of context data; distribution and placement of context information; and the mix of user preferences with context information. In ISAM, context awareness was a virtual module mainly because it was used to define which services should be available in each node, and what the implementation used by those would be. Continuum architecture is presented in Figure 3. The architecture is divided in layers: infrastructure, pluggable services, subsystems, and user space. The infrastructure comprises the execution environment and support, including the network, operating system, and, currently, the Java Virtual Machine (JVM). The pluggable services provide basic functionalities to Continuum subsystems. As the name implies, this services can be loaded on demand, as needed. The subsystems layer constitutes the core of Continuum middleware and supplies the main functionalities during execution. Finally, the user space layer has user applications and the Continuum Framework. Applications can use the JVM directly and interact also with the middleware towards the framework.

The framework incorporates Execution Profiler support, which helps the user choose the best implementation for each service. It also helps the user choose what services will be available in each node. The execution profiler parameterizes the deployment process during load-time. This reconfiguration process is needed each time a node is bootstrapped. During execution, applications could also need to load services on-demand. This is done by the Adaptation Management subsystem, which is also an improvement over ISAM context services. Not only is it targeting at the adaptation process itself, but also at the management of the adaptation process, which includes agility aspects and the maintenance of system stability. On one side, we have to address the delay between the perception of a new context state and the execution of actions to adapt the system to this new environment condition, demanding for agility. On the other hand, the execution of adaptation actions has a computational cost and competes with the application itself. In an extreme case, adaptation actions can be very frequent leading the system to an instability state, in which the majority of resources are being consumed by the execution of these adaptation actions. This demands for stability maintenance in the environment. Another redesigned subsystem is Ubiquitous Access. This subsystem reinforces follow-me semantics and invisibility issues, giving special consideration to user attention and intent. The main features of ubiquitous access subsystem are keeping a application database that manages applications in the environment, maintaining the user application environment, supporting persistent storage, managing user sessions, and interfacing to components external Continuum. We should design device-neutral applications, i.e., we should not start with the presentation and then build up the programming logic from that. Conse-

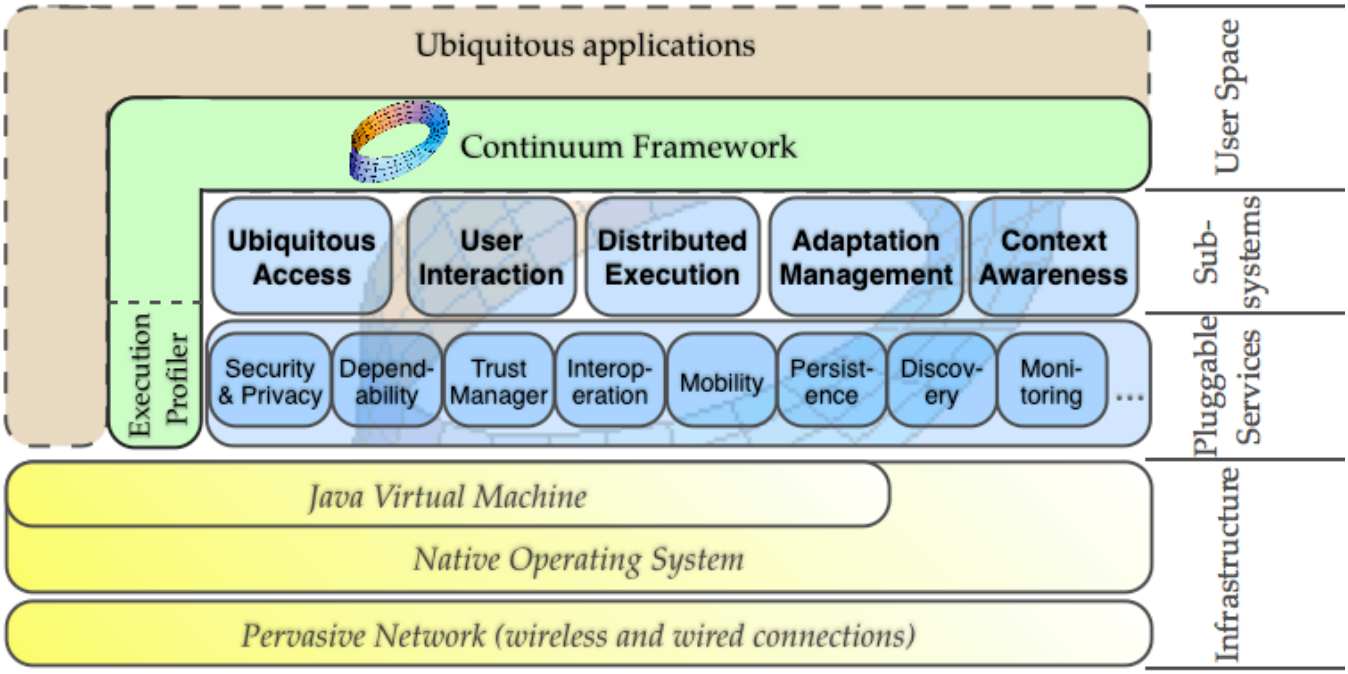

Figure 3. Continuum architecture

quently, Continuum has a subsystem specifically aimed at transparent User Interaction, which deals with people interaction and interfaces suitable to each type of device or environment. To accomplish this, during design-time we can define abstract user interfaces and predict different types of interaction, with aid of the framework, so that the decision of which interface to use can be postponed to execution-time. Another option is to dynamically generate the interfaces during execution, based on the abstract definitions, specific devices features, and contextual information. This option requires less effort during design, and tends to consume more processor power and communication latency during execution. However, it facilitates the use of contextual data. Currently, dynamic generation of interfaces is not supported by Continuum, but it is planned to be available in the future. The Distributed Execution subsystem is responsible for the distributed processing support in Continuum. In this, applications are spawned, codes are deployed on-demand, and their objects created and migrated among nodes. Furthermore, this subsystem keeps the environment physical organization, by storing attributes related to the management of the grid, i.e. resources, users, and applications. Finally, the Context Awareness subsystem deals with a variety of contextual information. It provides a formal representation for context, in an independent application manner. The subsystem also considers user preferences, i.e. requirements that vary from user to user and over time, which until recently, very little research has addressed this problem [6]. Context awareness subsystem is also in charge of storing context, along with points in time these data were created at, and distributing and localizing those.

6. Pluggable on-demand Services

We use in Continuum, as in ISAM, a service-based organization, which loads services on demand depending on what functionalities the applications need. These pluggable services add an adaptive behavior, which is important due to the high heterogeneity of the many different resources. In addition, Continuum proposes the use of SOC (Service-Oriented Computing) [8]. In SOC, we have a service layer according to the service-oriented architecture (SOA). The purpose of SOA is to support critical applications, which require the management and deployment of services and applications; it is also targeted at providing support for open services [8]. The application of SOC on the web is obtained by the use of web services. SOC, SOA, and web services create a general interface, which makes interaction easier in Continuum, and in a more ad-hoc approach, enables many applications to make effortless use of its services. Besides being loaded on demand, the services are context adaptive, i.e., the infrastructure is able to use the implementation that is better tuned to each device. Furthermore, we reduce resource consumption by loading only ser-

vices that are in fact needed. Such scheme is possible because services are defined by their semantics and interface, instead of a specific implementation. We propose the following services in Continuum: security & privacy, responsible for security mechanisms, such as authorization and privacy protection; dependability, aimed at handling fault, error, and failure; trust manager, accountable for the establishment of trust; interoperation, targeting at communication; mobility, to support logical and physical mobility; persistence, intended to store data; discovery, used to dynamically locate resources; and monitoring, for the purpose of interacting with sensors. Moreover, it is easy to add other services to Continuum, since we make use of SOA architecture.

7. Conclusion and Future Work

We believe that it is still difficult to find a software infrastructure that has all the necessary challenges presented by ubiquitous computing. Projects such as Aura, Gaia, One.World, and ISAM tried to accomplish many of these aspects. However, it is very hard to address several different open research topics in one project. The tendency today is providing middleware or frameworks for specific issues. In spite of this tendency, we think that a comprehensive software infrastructure can help the development of pervasive software. In this article, we presented Continuum software infrastructure, an evolution in the original ISAM project. The proposal, which is currently under development, applies follow-me semantics and deals with context awareness and context management in the ubiquitous computing field. As a future work, we intend to investigate and propose innovative solutions to deal with context. Particularly, we are interested in reduce the programmer burdening and the software development process, to support context awareness. Another concert is to contemplate the quality of context information and deal with the boundary problem, considering the physical limits (or other criteria) when defining the environment scope.

References

[1] I. Augustin, A. Yamin, J. Barbosa, L. Silva, R. Real, G. Frainer, G. Cavalheiro, and C. Geyer. *Mobile Computing Handbook*, chapter ISAM, Joining Context-Awareness and Mobility to Building Pervasive Applications, pages 73–94. CRC Press, 2004.

[2] J. Barbosa, C. Costa, A. Yamin, and C. Geyer. Gholo: A multiparadigm model oriented to development of grid systems. *Future Generation Computer Systems*, 39(2):86–98, 2005.

[3] P. A. Bernstein. Middleware: a model for distributed system services. *Commun. ACM*, 39(2):86–98, 1996.

[4] D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste. Project aura: Toward distraction-free pervasive computing. *IEEE Pervasive Computing*, 01(2):22–31, 2002.

[5] R. Grimm, J. Davis, E. Lemar, A. Macbeth, S. Swanson, T. Anderson, B. Bershad, G. Borriello, S. Gribble, and D. Wetherall. System support for pervasive applications. *ACM Trans. Comput. Syst.*, 22(4):421–486, 2004.

[6] K. Henricksen and J. Indulska. Developing context-aware pervasive computing applications: models and approach. *Pervasive and Mobile Computing*, 2(2):37–64, 2006.

[7] A. Mukherjee, D. Saha, and C. Biswas. Present scenarios and future challenges in pervasive middleware. In *Communication System Software and Middleware, 2006. Comsware 2006. First International Conference on*, pages 1–5, 08-12 Jan. 2006.

[8] M. P. Papazoglou and D. Georgakopoulos. Introduction: Service-oriented computing. *Commun. ACM*, 46(10):24–28, 2003.

[9] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt. A middleware infrastructure for active spaces. *IEEE Pervasive Computing*, 01(4):74–83, 2002.

[10] T. Weis, M. Knoll, A. Ulbrich, G. Muhl, and A. Brandle. Rapid prototyping for pervasive applications. *Pervasive Computing, IEEE*, 6(2):76–84, April-June 2007.

[11] M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):94–104, 1991.

[12] A. Yamin. *Arquitetura para um ambiente de grade computacional direcionado às aplicações distribuídas, móveis e conscientes de contexto da computação pervasiva*. PhD thesis, Univerisidade Federal do Rio Grande do Sul, 2004.

[13] A. Yamin, J. Barbosa, I. Augustin, L. Silva, R. Real, C. Geyer, and G. Cavalheiro. Towards merging context-aware, mobile and grid computing. *International Journal of High Performance Computing Applications*, 17(2):191–203, 2003.