Applying Utility Functions to Adaptation Planning for Home Automation Applications

Pyrros Bratskas, Nearchos Paspallis, Konstantinos Kakousis & George A. Papadopoulos

Department of Computer Science, University of Cyprus, P.O.Box 20537, Nicosia, Cyprus
{bratskas, paspalli, kakousis, george}@cs.ucy.ac.cy

Abstract A pervasive computing environment typically comprises multiple embedded devices that may interact together and with mobile users. These users are part of the environment, and they experience it through a variety of devices embedded in the environment. This perception involves technologies which may be heterogeneous, pervasive and dynamic. Due to the highly dynamic properties of such environments, the software systems running on them have to face problems such as user mobility, service failures, or resource and goal changes which may happen in an unpredictable manner. To cope with these problems, such systems must be autonomous and self-managed. In this paper we deal with a special kind of a ubiquitous environment, a smart home environment, and introduce a user-preference based model for adaptation planning. The model, which dynamically forms a set of configuration plans for resources, reasons automatically and autonomously, based on utility functions, on which plan is likely to best achieve the user's goals with respect to resource availability and user needs.

1. Introduction

Ubiquitous computing environments are characterized by frequent and unpredictable changes. To retain their usability, usefulness, and reliability in such environments, systems must adapt to the changing context conditions. Consequently, there is a growing demand for software systems to be deployed in such environments. In this case, many constraints and requirements must be taken into consideration in order to provide a fair utility to the users. Furthermore, mobile computing environments include a huge spectrum of computation and communication devices that seamlessly aim to augment peoples' thoughts and activities with information, processing and analysis. Devices such as Personal Digital Assistants (PDAs) and smart-phones have gained a lot of popularity and are increasingly being networked. On the other hand, people use different software development and deployment platforms to design and create applications for such devices. These applications must be context-aware to meet the requirements of the highly dynamic and distributed environment. These types of context-aware systems adapt

not only to changes in the environment, but also to the user requirements and needs.

But even though the device capabilities become more and more powerful, the de-sign of context-aware applications is constrained not only by physical limitations, but also by the need to support a plethora of features such as distribution, scalability, fault tolerance, etc. Indeed, mobile devices will continue to be battery-dependent and operate in an environment where more and more devices will be present and will need to communicate or share resources for the foreseeable future.

A special kind of a ubiquitous environment is a smart home environment where a large number of devices are used for a wide set of purposes. Examples include lighting control modules, heating control panels, light sensors, temperature sensors, gas/water leak detectors, motion detectors, video surveillance, healthcare systems and advanced remote controls. As in ubiquitous environments, a smart home environment faces challenges like adaptability and context-aware reconfiguration, mobility (user mobility, device mobility and information mobility ([14], [21]), etc. However unlike them the smart home environment is very much user-oriented and thus sensible to user preferences and needs.

In this paper we describe a self-adaptive distributed approach that automates the configuration and reconfiguration in a ubiquitous computing environment. This approach provides home users with the ability to set up an advanced home environment taking into account user preferences and needs. We focus on a home automation application and we present a model of adaptation in such an environment. In this respect, utility functions are used to choose from a set of dynamically constructed configuration plans. The primary aim is to choose the one that best meets the user preferences and needs while respecting the limitations imposed by the resources availability.

The main contributions of this work include that it makes explicit representations of user preferences and needs with respect to quality dimensions offered by the devices, so that the system can automatically determine what service qualities are required for any given configuration. The system is dynamic and performs reactive adaptation: the application defines which aspects of context are of interest to the application itself, identify dynamically which context changes are relevant, and choose the best configuration to execute. Some of these configurations could be predefined and some others could be created on-demand during execution. Furthermore, the system enables the decoupling of user preferences from the lower level mechanisms that carry out those preferences, which in result provides a clean separation of concerns (from an engineering perspective) between what is needed and how it is carried out.

This paper is organized as follows: Section 2 describes a motivation scenario that illustrates our idea. Then, in Section 3 we discuss about the architecture of the sys-tem, how the user's preferences are considered in our paper and how we use utility functions to map them into numerical values. Section 4 presents a case study scenario and Section 5 describes the related work and outlines the research

challenges of the preference and task-driven computing, but also the use of utility function in some existing middleware systems in the area of ubiquitous computing. Finally we provide conclusion and outlook for future work in Section 6.

2. Motivating Scenario

In order to better illustrate the motivation for this work, this section describes an application scenario to demonstrate the problem aspects. We have chosen a home automation system which is composed of devices and services operated by users. This includes the air-conditioning system, a multimedia home entertainment system, a digital lighting system, additional electrical devices, etc. This environment also includes user-carried devices like smart-phones, PDAs, laptops, etc. It is assumed that these devices discover and communicate with each other, using technologies such as Jini [19] and UPnP [18].

When a user returns back to his home, he would like an automatic adjustment of the heating, cooling, and lighting levels in the living room, or the control of the home entertainment system. In this case, the devices can sense the presence of a user, her or his identity, and thus set appropriate values to the different features of the room based on a set of factors (e.g. the day of the week or the time of the day). The values of the different features may also vary given the preferences of the individual user.

In the scenario we consider 4 home devices: the stereo, the TV, the air-conditioning and the digital lighting system. Related to these systems, we also consider the following characteristics: the volume of the stereo system, the room temperature, the TV brightness, and the room luminosity. These devices offer information to the users about their characteristics and functionalities so that the users can configure them. This can be done using any technology such as an electronic house key, a mobile phone or a PDA.

When the user enters the room, he can activate a command to tune the lights and the temperature in the room, and the volume of the multimedia entertainment system. Several decisions have to be made though. For example, if there is enough natural light, the digital light system can be switched off. Due to this decision, the TV contrast and the room temperature have to be tuned properly during summer or winter months. The TV volume has to be tuned considering the state of the stereo for example; if the stereo is switched on the TV volume should be muted. On the other hand, if privacy is needed, the user can shut the drapes which influences all other device settings, i.e. the lighting systems must be adjusted again, etc. The scenario is described by the user's point of view and we must map all these operations into system's services. As these environments are user-oriented, the user preferences are of high importance, the services must satisfy them taking into account the context constraints and the QoS required by the user.

Our approach suggests the use of a user-preference management application that adapts the home automation environment to the user. To apply adaptation, we

need a model which takes into account the user preferences during reasoning and decision making. For instance, there could be a situation with the user leaving home. The system will detect if the user is leaving and pass the control from its remote control or PDA to the home control system. When leaving, all lights should turn-off automatically (which saves the user from having to go in each room and switch off the lights manually), and the video surveillance system will be enabled. In the case of detecting movements in the home, the home control system has two choices depending on the available bandwidth: it will send a video stream to the user's PDA via Internet if there is enough bandwidth, otherwise it will store the video stream to the home control computer.

The adaptation model we introduce in this paper attempts to make the best choice among all possible ones, by applying a utility function and taking into account the preferences of the user and resource availability. In this model the user specifies his preferences and then the system maps them onto the services offered in the ubiquitous environment.

3. System overview

The main idea of our approach is to enable users to make requests for tasks to be achieved, which must implicitly take into consideration the user preferences. These preferences guide the selection of a plan capable of executing the task.

When a user enters a room, he selects his preferences based on a set of options. This set can be predefined depending on a common set of user tasks. The selection of preferences also depends on a set of constraints, which can be divided to logical and context constraints. For example a logical constraint is that the TV and the stereo can not be switched on and tuned to different inputs at the same time (i.e. another media is played by the TV and another by the stereo system). A context constraint is related to the resources i.e. if there is not enough bandwidth, some internet-based TV channels might not be available.

In this section we first discuss device and service discovery protocols, as they are a basic requirement for enabling synergies. Then, we discuss about the configuration plans and how the user preferences are taken into account by the system. We also present how the utility functions are constructed, and how the system operates based on the selection of the best plan using the utility function.

3.1 Device and Service Discovery

In a home automation environment with devices that provide services without requiring any user attention, service discovery is essential to achieving such sophistication. It enables devices and services to discover, configure, and communicate with each other. Device interaction protocols like UPnP, Bonjour [5], SLP [10],

Bluetooth SDP [4], and Jini allow the dynamic discovery of devices in a home environment network without any need for user interaction. These technologies allow interested clients to dynamically discover available services in a network and also they supply the needed mechanisms for browsing through services, as well as for detecting and using the desired service. Dynamic and automatic service discovery is particularly challenging in ad hoc communication networks, where no fixed infra-structure exists. Such networks are characterized by devices which connect to each other, spontaneously offer and acquire services and then disconnect [3].

3.2 Configuration Plans

Our system uses configuration plans which are applied to achieve the user goals. A configuration plan is a plan which defines how the components are connected to each other in order to provide the functionality required by the application. Thus, a plan can be formally thought of as being like a protocol which defines the communication of the user with the environment.

The resources in the environment are subject to dynamic changes concerning their values and/or properties. Also, new devices can be added and others can be deactivated, the luminosity in the environment may increase or decrease, batteries discharge, etc. Taking that into account, the choice of configuration plans can make the adaptation process easier. In order to perform adaptations, we decouple the preference specification from the middleware specification which provides a separation of concerns. In this way, the "adaptation logic" is one level higher than the middle-ware. That means that the users do not deal with the values and names of resources, but its viewpoint over the system are the configuration plans. Some of these plans can be predefined based on a set of user preferences. On the other hand, other plans can be redefined if changes occur in the system, as well as new configuration plans that can be added.
We make the following assumptions:

- Plans Vs Utility: For any particular adaptation, there may be multiple configuration plans that can achieve it. The choice of a particular plan is based on the utility it offers to the system.
- Plans Vs Resources: Each plan requires a set of resources with values defined by user preferences in order to provide a certain QoS to the user.
- Utility Vs User goals: The utility offered by a plan may not satisfy the user at a certain moment. For example the user in a room may want to prefer natural light to the privacy by opening the drapes. It is up to the user to decide his preference priorities and guide the plan execution accordingly.
- User needs Vs offered utility: Each device defines its domain of the offered value. The utility of a plan will be evaluated with respect to user needs for a

certain utility and to the utility offered by the configuration plan. The difference between them will be weighted in order to prefer one plan to another.

- Plans Vs Distributed, dynamic planning: Plans are not predefined but rather they are dynamically generated at run-time by discovering and coordinating with the set of available, distributed devices. To achieve this, we assume a component-based approach, where applications are formed as component compositions, and where components might become available or unavailable dynamically [9].

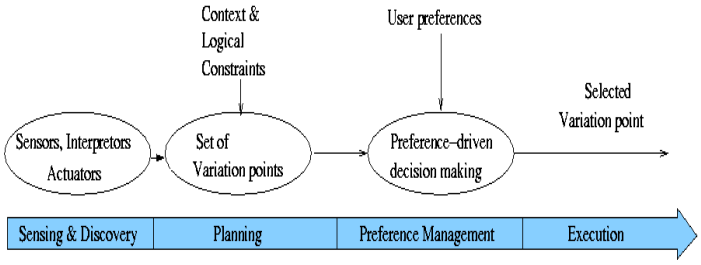

Figure 1 The system lifecycle

Based on these assumptions the system operates as follows: the user-attached device performs device discovery as the user enters the home. Then a set of variation points is defined based on context and logical constraints. The decision on which variation point is the best one is driven by the user preferences. As a result, the system performs adaptation after the selection of a new variation point. This operation is depicted in Figure 1.

3.3 User Preferences

The problem of modelling preferences has been widely research in others fields. For instance, Agrawal and Wimmers in [1] present a preference model that allows users to supply preferences and be combined, using preference functions, for decision making. Hafenrichter and Kießling in [11] represent preferences as strict partial orders for the efficient integration of preference querying. Both approaches are used in the field of database research.

In our case, we use a simplified version of the preference model introduced by Henricksen et al. in [12]. This preference model employs a scoring mechanism, which is loosely based on the scheme proposed by Agrawal and Wimmers. Each preference is assigned with a score which is a numerical value in the range of [0, 1] where larger score indicates higher preference Four special scores represent prohibition, indifference, obligation and error conditions.

Applying this preference model to our scenario yields the following preferences:

p_1 = when Privacy (public) and Light (natural, 30)

$w_1 = 0.7$

p_2 = when Privacy (public) and Light (natural, 10)

$w_2 = 0.2$

p_3 = when Privacy (public) and Light (artificial)

$w_3 = 0.7$

The value w_i is used by the utility function in order to compute the utility of a configuration. In fact, w_i is a weight that reflects how much the user cares about the user preference p_i .

3.4 Utility Functions

To perform adaptation, the selected plan must meet the user's preferences in order to receive a good utility and also to minimize the use of shared resources. To achieve this, we use utility functions to map the user preferences for QoS to a function that defines how a selected plan satisfies the user preferences. The aim of the utility functions is to express the quality of the adaptation for the user. Its input includes the user preferences taking into account the current context and the available resources, while its output is the degree to which a variation point satisfies the user goals. Utility functions are in general *n*-dimensional functions taking as arguments values from an *n*-dimensional utility space [2]. In our work, we adopt a simple approach, which defines overall utility as a weighted sum of the set of utility functions.

Let $\mathrm{P} = \{cp_1,...,cp_n\}$ be the set of configuration plans and let q_i for $i = 1..n$ be the corresponding utility of the adaptation when the plan p_i is selected. Then $Q = \{q_1,...,q_n\}$ is the set of utility dimensions related to these plans. This utility depends on the availability of resources and on the logical and context constraints. To express this with a mathematical formula let *R* be the set of resources and *C* the set of constraints. We assign to each resource a weight *w* which expresses its availability as well as its value. As user preferences express user constraints and needs, they can be thought of as predicates that must be maximized.

Thus, the utility function *F* related to the QoS q_i is expressed as:

$$F(q_i) = f_{user\_need}(x_1,...,x_m) \text{ where } x_i \in R \cup C \quad \forall i \in 1...m$$

Then, the utility function associated to a configuration plan cp_i is expressed as the normalized, weighted sum of the dimensional utilities, as follows:

$$U\ (cp_i) = \frac{\sum_{i=1}^{n} w_i F(q_i)}{\sum_{i=1}^{n} w_i} \quad \forall cp_i \in P$$

The utility function $U : P \to [0,1]$ is a function that maps and transforms a set of configuration plans into a numerical values and the weight w_i reflects how much the user values the particular user preference p_i. This process also takes into account the availability of resources and context constraints.

For any particular adaptation, there may be multiple configuration plans that can achieve it. The choice of a plan variant is based on the utility it offers to the system. Each configuration plans requires a set of resources with values defined by user preferences in order to provide a certain QoS to the user. On the other hand, its utility may not satisfy the user at a certain moment. It is up to the user to decide his preference priorities and guide the plan selection accordingly.

4. Evaluation

Existing solutions for self-adaptation consider all configurations of applications and choose the ones that best fit the user's preferences and needs. In a highly dynamic environment where the number of such configurations is quite numerous, the task of managing them can be time consuming.

In this section we show that our approach offers some benefits as opposed to other approaches. For example, in the comparison with the MADAM approach [7], the main advantage is that the plans can include adaptation at the server side (assuming a client-server model), whereas in MADAM adaptations were limited to local only. This is important because it enables adaptations in ubiquitous computing environments, which otherwise (i.e. in MADAM) it is very difficult.

Case study example

Let us revisit the application scenario of Section 2. Given the smart-home environment, we consider three context types: *temperature* which corresponds to the room temperature, *luminosity* which corresponds to the room light, and privacy which express the user's need for privacy. Let w_t, w_l and w_p be the weights indicating how the user may specify the relative importance of the quality dimensions for the context types, temperature, luminosity and privacy respectively. Table 1 illustrates the importance that the user gives to any of these dimensions for three different configurations and Table 2 the dimensional utility function for

the user, defined as a set of coefficient values where each coefficient specifies the utility value for a quality layer of a QoS-dimension.
The temperature and luminosity properties take their values in the ranges [10, 40] and [0, 40] respectively, while the privacy property is a binary variable taking the values *true* or *false* according to the user's preference for privacy. If we consider that the user goal is to illuminate the room then the first service variant will be whether the light is natural or artificial. This is closely related with privacy since opening the drapes will have an impact to the privacy property but also to the temperature property because the sunny light will impact it as well.

Table 1 Dimensional weights

Configuration	w_t	w_l	w_p
1st	0.7	0.3	.0.5
2nd	0.3	0.8	0.5
3rd	0.2	0.2	0

Table 2 Dimensional utility functions

Configuration	$F(t)$	$F(l)$	$F(p)$
1st	0.8	0.6	0.5
2nd	0.7	0.5	0.4
3rd	0.6	0.4	0.3

As can be seen from the Table 1, the first configuration perceives the temperature dimension as the most important ($w_t > w_p > w_l$) while the second configuration gives more importance to the luminosity dimension. We assume that privacy is needed for these two configurations which is not the case for the third one. The utility for each configuration is shown in Table 3.

Table 3 Utility for each configuration

Configuration	1st	2nd	3rd
Utility	0.66	0.5	0.5

5. Related work

The notion of task-driven computing was first introduced by Wang and Garlan in [20]. The approach is based on two basic concepts, tasks and sessions, and that it

is possible to let users interact with their computing environments in terms of high level tasks and free them from low level configuration activities.

Implementing this notion, the Aura infrastructure [16] performs automatic configuration and reconfiguration in ubiquitous computing environments according to the users' tasks. For that, the infrastructure needs to know what the user needs from the environment in order to carry out his tasks. It also needs mechanisms to optimally match the user's needs to the capabilities and resources in the environment. Aura infrastructure addresses two principles of autonomic computing as they were introduced in [13]: self-optimization and self-healing from the point of view of user's task in an ubiquitous environment.

In [17], the authors describe an approach to self-configuring in computing environments. Their adaptation architecture allows explicit representation of user's tasks, preferences and service qualities.

Henricksen et al. present an approach involving the use of preference information as a basis for making flexible adaptation decisions in [12]. Their work focuses on generic preference and programming models that can be used for arbitrary context-aware applications and can facilitate preference sharing among applications. They introduce a preference model and a programming model to support a common form of context-dependent choice problem. The authors consider preferences as a link between the context and appropriate application behaviors placing them in a layer of separation between the application and its context model allowing them to evolve independently of one another.

In MADAM, utility functions were used to enable self-adaptive behavior in mobile applications. Basic composition plans were provided by the developer and were dynamically used to from the set of possible configurations (variants). Then, the MADAM Middleware was used to evaluate them at runtime, based on the contextual and resource conditions, and automatically select the most suitable option. Naturally, the MADAM approach is very similar to this approach, as our approach builds on it and attempts to extend it. The main limitation that we attempt to overcome is that of limited support for ubiquitous computing. While MADAM enabled the adaptation of locally hosted applications, it fails to support adaptation of remotely hosted services (i.e. an application running on a different host). This limits the domain of possible applications to locally deployed software only, with apparent limitations concerning ubiquitous computing.

MUSIC middleware builds on the legacy of MADAM, and attempts to extend its scope to ubiquitous computing environments. As described in [15], the MUSIC planning framework is an extension of the MADAM planning framework, which supports the adaptation of component-based architectures. The extension proposed supports self-adaptation of ubiquitous applications to changes in the service provider landscape. The planning middleware evaluates discovered remote services as alternative configurations for the functionalities required by an application. In the case of services, the planning framework deals directly with SLA protocols supported by the service providers to negotiate the appropriate QoS for the user while our approach deals with utility functions and uses an explicit representation of

user preferences allowing users to supply their own preferences and providing flexibility and spontaneity, in response to changes in user needs and operating conditions. As MUSIC is still work in progress, there has been no real-world evaluation of it yet.

6. Conclusions

In this paper we address the self-adaptation in a home automation environment. The adaptation is performed through the use of configuration plans which are selected using a utility function mechanism. The choice of a plan, which best meets the user requirements and needs, is made by taking into account the user preferences which are represented explicitly during the calculation of the utility. This makes the system dynamic and offers transparency to the users.

A future direction of this work will be the study of issues like diagnosis and recovery by introducing a mechanism supporting fault-tolerance. On the other hand, as a good starting point over the adaptation for a home automation application, another future direction of this work could be the context and service discovery routing in such a system. Home automation applications represent a special segment of networked wireless devices with its unique set of requirements related to the set of home networking applications and the perceived operation of the system.

Acknowledgments

The authors of this paper would like to thank their partners in the IST-MUSIC project and acknowledge the partial financial support given to this research by the European Union (6th Framework Programme, IST 035166).

References

1. Agrawal, R., Wimmers, E. L. 2000. A framework for expressing and combining preferences. *ACM SIGMOD Conference on Management of Data*, Dallas, TX, 2000. ACM Press, New York.
2. Alia, M., Eide, V. S. W. , Paspallis, N., Eliassen, F., Hallsteinsen, S., Papadopoulos, G.A. 2007. A Utility-based Adaptivity Model for Mobile Applications. *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*, Niagara Falls, Ontario, Canada, May 21-23, 2007, IEEE Computer Society Press, pp. 556-563
3. Bettstetter C., Renner C. 2000. A Comparison of Service Discovery Protocols and Implementa-tion of the Service Location Protocol. EUNICE 2000, Sixth EUNICE Open European Sum-mer School, Twente, Netherlands, September 2000.
4. Bluetooth Special Interest Group, Bluetooth Core Specification – Version 2.1. 2007
5. Bonjour Protocol Specification. http://developer.apple.com/networking/bonjour/specs.html

6. Edwards, W. K. 2006. Discovery Systems in Ubiquitous Computing. *IEEE Pervasive Computing* 5, 2, 70-77, April 2006.
7. European IST-FP6 project MADAM (Mobility And aDaptation enabling Middleware) http://www.ist-madam.org
8. European IST-FP6 project MUSIC (Self-adapting applications for Mobile Users In ubiquitous Computing environments) http://www.ist-music.eu
9. Floch, J., Hallsteinsen, S., Stav, E., Eliassen, F., Lund, K., Gjorven, E. 2006. Using Architecture Models for Runtime Adaptability. *IEEE Software* 23, 2, pp. 62-70, March 2006.
10. Guttman, E., Perkins, C., Veizades, J., and Day, M. 1999. Service Location Protocol, Version 2. RFC. RFC Editor
11. Hafenrichter, B., Kießling, W. 2005. Optimization of relational preference queries. 16th Australasian Database Conference - Volume 39 (Newcastle, Australia). H. E. Williams and G. Dobbie, Eds. *ACM International Conference Proceeding Series*, vol. 103. Australian Computer Society, Darlinghurst, Australia, 175-184.
12. Henricksen, K., Indulska, J., Rakotonirainy, A. 2006. Using context and preferences to implement self-adapting pervasive computing applications. *Journal of Software Practice and Experience, Special Issue on Experiences with Auto-Adaptive and Reconfigurable Systems*, volume 36(11-12), pages 1307-1330. Wiley, 2006
13. Kephart, J., Chess, D. M. 2003. The vision of autonomic computing. *IEEE Computer Magazine* vol. 36, no. 1, pp. 41–50, January 2003.
14. Labrinidis, A.., Stefanidis, A. 2005. Panel on Mobility in Sensor Networks. *6th International Conference on Mobile Data Management (MDM'05)*, pages 333-334. Ayia Napa, Cyprus, May 2005.
15. Rouvoy, R., Eliassen, F., Floch, J., Hallsteinsen, S., Stav, E. 2008. Composing Components and Services using a Planning-based Adaptation Middleware. *7th International Symposium on Software Composition (SC'08)*. p. 52–67 of LNCS 4954 (Springer). Budapest, Hungary. March 29–30, 2008.
16. Sousa, J. P. and Garlan, D. 2002. Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments. *IFIP 17th World Computer Congress - Tc2 Stream / 3rd IEEE/IFIP Conference on Software Architecture: System Design, Development and Maintenance* (August 25 - 30, 2002). J. Bosch, W. M. Gentleman, C. Hofmeister, and J. Kuusela, Eds. IFIP Conference Proceedings, vol. 224. Kluwer B.V., Deventer, The Netherlands, 29-43.
17. Sousa, J. P., Poladian, V., Garlan, D., Schmerl, B., Shaw, M. 2006. Task-Based Adaptation for Ubiquitous Computing. IEEE Transactions on Systems, Man, and Cybernetics, Part C: *Applications and Reviews, Special Issue on Engineering Autonomic Systems*, Vol. 36(3), May 2006.
18. UPnP Forum 2007. Universal Plug and Play. http://www.upnp.org
19. Waldo, J. 2000 The Jini Specifications. *2nd. Addison-Wesley Longman Publishing Co*., Inc.
20. Wang, Z., Garlan, D. Task-driven computing. Technical Report CMU-CS-00-154 School of Computer Science Carnegie Mellon University http://reports-archive.adm.cs.cmu.edu/anon/2000/CMU-CS-00-154.pdf
21. Zachariadis, S., Mascolo, C., Emmerich, W. 2002. Exploiting Logical Mobility in Mobile Computing Middleware. *22nd International Conference on Distributed Computing Systems - Workshops (ICDCS 2002 Workshops)*. July 2002, Vienna, Austria.