Diversity of Mobile Distribution Systems

Haeng-Kon Kim

School of Information Technology, Catholic University of Deagu, Korea
hangkon@cu.ac.kr

Abstract

Modeling of domain-dependent aspects is a key prerequisite for the design of software for mobile applications. Most mobile applications include a more or less advanced model of selected aspects of the domain in which they are used. This paper discusses the creation of such a model and its relevance for technical design of a mobile application. Conventional approaches to modeling of context focus either on the application domain or the problem domain. These approaches are presented and their relevance for technical design of software for mobile systems is discussed. The paper also discuss from an empirical study where a methodology that combines both of these approaches was introduced and employed for modeling of the domain-dependent aspects that were relevant for the design of a software component in a mobile applications. The empirical study was conducted in two companies that produce software for mobile applications. The resulting models of domain-dependent aspects are presented, and the experiences from the modeling process are discussed. It is concluded that a dual perspective based on both of the conventional approaches is relevant for capturing the aspects that are necessary for creating the domain-dependent models that are integrated in a mobile applications system.

Keywords: *Model Driven Development, Mobile Application Development, Domain Dependent Model*

1. Introduction

Mobile applications development challenges the modeling activities that precede the technical design of a software system. The context of a mobile system includes a broad spectrum of technical, physical, social and organizational aspects. Some of these aspects need to be built into the software. Selecting the aspects that are needed is becoming increasingly more complex with mobile systems than we have previously seen with more traditional information systems. The term distributed objects in mobile applications usually refers to software modules that are designed to work together, but reside either in multiple computers connected via a network or in different processes inside the same computer. One object sends a message to another object in a remote machine or process to perform some task.

Deploying mobile applications in the real world implies that the application is deployed in different settings with different constraints. The aforementioned mixed approach implies that each time the application is deployed a great part of the application must be redeveloped, which leads to little software reuse, or even to systematic redevelopment from scratch. If we want mobile applications to leave the laboratory settings and to become truly ubiquitous, software engineering techniques must be created in order to ease the development process and the deployment of applications in different physical and technological settings. In this paper, we propose to adapt the Model-Driven Architecture (MDA) approach to the mobile applications [1] in order to significantly increase software reuse and ease application adaptation to new deployment environments. MDA decouples the applications from their execution platforms via model abstractions and model transformations. An application is initially described by a Platform-Independent Model (PIM) that does not contain any information on the final execution platform and middleware. This model is refined several times via successive transformations, each transformation integrating new constraints like persistency or redundancy. Once the PIM is enough detailed, an execution platform is chosen and the PIM is transformed into a Platform-Specific Model (PSM). This PSM is in turn transformed several times, in order to generate code, compile the application, and package the application. The paper also discuss from an empirical study where a methodology that combines both of these approaches was introduced and employed for modeling of the domain-dependent aspects that were relevant for the design of a software component in a mobile applications. The empirical study was conducted in two companies that produce software for mobile applications. The resulting models of domain-dependent aspects are presented, and the experiences from the modeling process are discussed. It is concluded that a dual perspective based on both of the conventional approaches is relevant for capturing the aspects that are necessary for creating the domain-dependent models that are integrated in a mobile applications system.

2. Related Works

2.1. Key Issues for Mobile Applications

This section presents five issues we consider essential for ubiquitous applications. These five issues are the cornerstones of the proposed application framework [2-3].

Resource-Awareness

Users in ubiquitous computing scenarios are surrounded by hundreds of devices including sensors, displays, and CPUs. In order for applications to exploit these resources, they must be aware of existing resources; the capabilities, availability, and cardinality of these resources. An active meeting room, for example, is aware of existing devices (*e.g.,* plasma displays, projectors, servers, notebooks, PDAs, and sensors), services (*e.g.,* light control, temperature control, and audio control), applications (*e.g.,* collaborative document editor, and slide show manager), and people present in the meeting room, therefore allowing applications to exploit the resources.

Context-Sensitivity

Context is one of the most important properties in ubiquitous computing and affects the behavior of applications. We identify at least three application aspects that are altered by context: data, composition, and logic. Applications require transforming the format of the output data to accommodate changes in the context. For example, an e-mail application using a large monitor to display the e-mail's text may require transforming the text to speech when the user moves to a new location where only speakers are available. Context also affects the internal composition of the application - number of application components and their composition rules. A music application may use a user's laptop to play the music if there are other people present in the room; or may use the audio system of the room, the displays (to present the list of songs), and the room's speech recognition system to control the application when the user is alone. Finally the logic of the application may also be affected by the context situation. For example, a news broadcasting application may select different types of news depending on who is in the room, the time of the day, or the mood of the users.

Multi-Device

In an environment where users are surrounded by hundreds of devices, the notion of logging into a single device becomes inappropriate. Users log into the active space (either implicitly or explicitly) and can take benefit of any entity contained in the space, as long as certain security and availability policies apply. This "post-pc" scenario requires a new model for application construction that allows partitioning applications into different devices as required by users and their associated context (*e.g.,* time of the day, location, current task, and number of people). Application partitioning allows distributing functional aspects of an application (*e.g.,* application logic, output, and input) across different devices. Remote terminal systems allow redirecting the application output and input to different devices. However, they do not provide support to redirect the application output to one device and the input to another device. And for the same application, it is not possible to redirect multiple outputs to different devices. The type of application partitioning we seek is conceptually similar to the one proposed by[5], and provides fine grained control to choose a target device for each individual application functional aspect, as well as support for altering the application partitioning at run-time.

The application partitioning must be: (1) dynamic, so it may vary at run-time according to changes in the active space (*e.g.,* new devices introduced in the space, or new people entering the space), and (2) reliable, in such a way that guarantees application integrity even when the application is distributed across different devices.

For example, a calendar application running in an active office may use a plasma display to present the appointments for the week, a handheld to display the appointments for the day, and may use a desktop PC to enter data. However, the same calendar running in a vehicle may use the sound system to broadcast information about the next appointment, and use an input sensor based on speech recognition to query the calendar or to enter and delete data.

User-Centrism

Environmental-awareness and the multi-device approach convey a third essential property: user-centrism. To accommodate application partitioning into multiple devices that vary over time, we bind applications to users and map the applications to the resources present in the space. Users may start these applications and use them for several days, months, or even years. Applications may be periodically suspended and resumed but not terminated. These applications are bound to users, and take benefit of the resources present in the users' environment. This application behavior requires applications to have permanent access to data (regardless of the users' context such as location), be able to access and take benefit of resources currently available, and be able to adapt to the current environment, either by changing their composition or by modifying the data they manipulate, or both.

User-Centrism requires applications to (1) move with the users, (2) adapt according to changes in the available resources (it may imply data format transformation, or internal application composition, or both), and (3) provide mechanisms to allow users to configure the application according to their personal preferences.

Mobility

Application partitioning and user-centrism require applications to be mobile. We consider two different types of mobility: intra-space mobility and inter-space mobility. Intra-space mobility is related to the migration of application components inside an active space and is the result of application partitioning among different devices. Intra-space mobility allows users and external services to move application components among different devices. Inter-space mobility concerns moving applications across different spaces, and is a consequence of the user-centrism (users are mobile by definition).

2.2. Product line and MDA Approach

Model Driven Architecture (MDA) [4] is a standard produced by the Object Management Group (OMG). The goal of MDA is to separate the design of application or business logic from the implementation platform. Designs are specified in a platform-independent model (PIM) that can then be translated to a platform-specific model (PSM) by an MDA tool, which is then used to generate the implementation. MDA is dependent on and makes use of several other OMG standards including the Unified Modeling Language (UML), Meta-Object Facility (MOF), XML Meta-Data Interchange (XMI), and Common Warehouse Metamodel (CWM). Software product line development involves three essential activities: core asset development, product development, and management. Core asset development involves the creation of common assets that will be used in individual products as well as the evolution of the assets in response to development user feedback, new market needs, *etc*. When combined with managerial strategy, core asset development embodies the domain engineering activity. Product development, or application engineering, creates individual products by reusing common assets, provides feedback to core asset development, and engineers product updates as core assets evolve and new requirements come. Management includes technical management and organization management, where technical management handles new requirements coming from inside and outside the organization, and coordinates core asset development and product development activities. In our work we use both MDA and product-line concepts to enable management and generation of mobile applications specifications and code as in Figure 1.

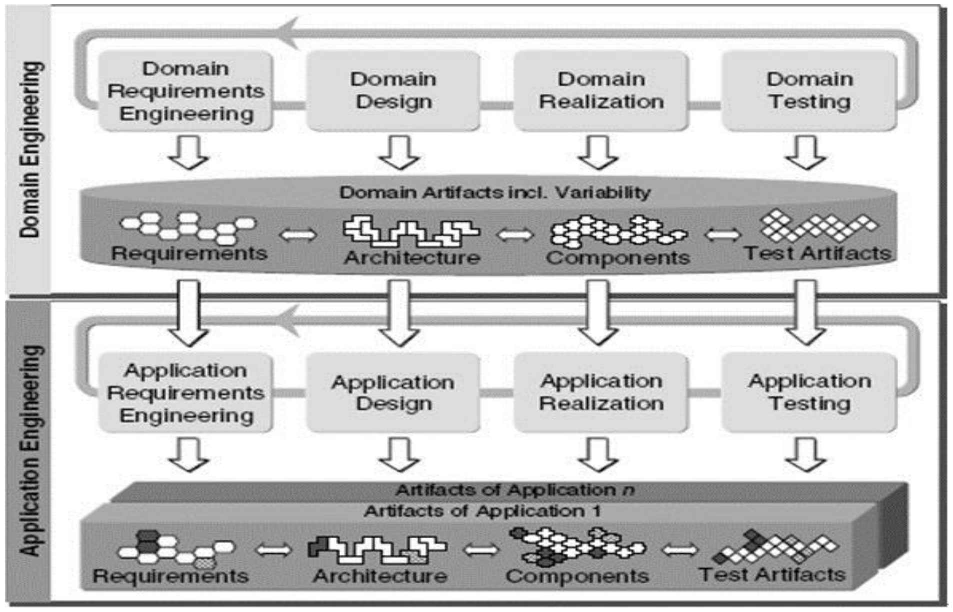

Figure 1. Product Line and MDA Engineering for Mobile Applications

3. Diversity of Mobile Distribution Systems

3.1. Modeling for Mobile Distribution Systems

In this proposal, the MDA ontology-based model follows MOF standardization, using in the data interchange data generated using XMI schema between meta-levels. According with OMG specification, on M2 meta-level is possible to materialize the specified class in M3 (MOF) meta-level. Thus, on level 2 a model can be used to construction of MDA models, being implemented in the M1 meta-level, creating an independent platform model, across of the construction class diagrams and stereotypes for mobile application domain within the UML tool. The generation of the rules mappings for a specific platform occurs on M1 meta-level, using MDA models tools, after the construction of the conceptual model in the M1 meta-level. Figure 2 presents the architecture proposal, starting from the construction of mobile applications distribution on MDA for an application domain until the formal verification model, before of transformations in another one to a specific platform.

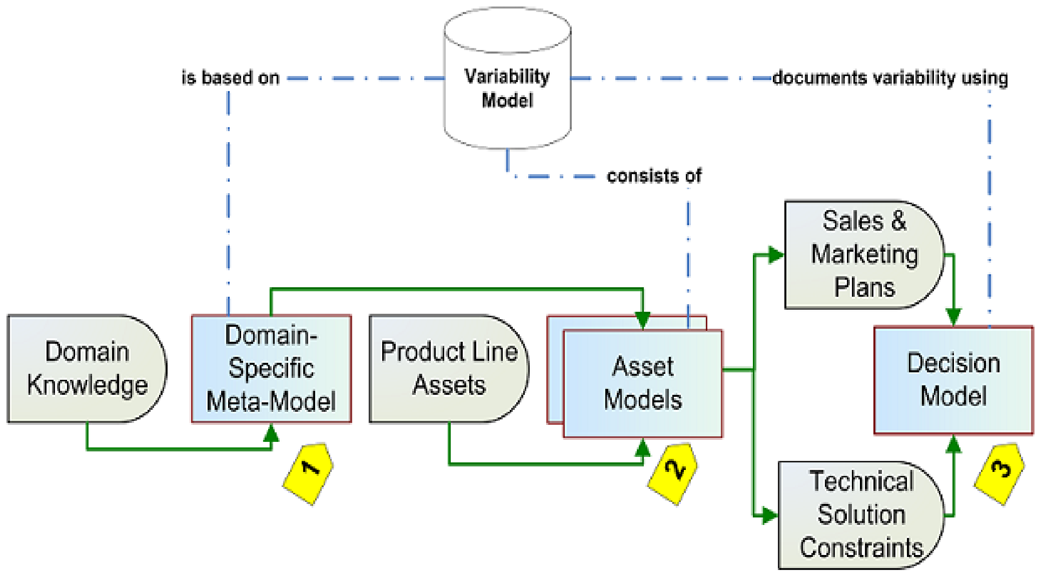

Figure 2. Model for Mobile Applications Distribution

The process modeling approach takes its point of departure in the way users work. This relates more generally to a focus on this domain:

- *Application domain:* The individual persons or roles and the organization that administrates, monitors, or controls a problem domain.

The application domain is where the users are and do whatever they do when they use the system. For an air traffic control system, the application domain is in the control tower where the controllers perform their air traffic control. The controllers monitor the traffic on the screen, decide on interventions, and direct the flights in their air space. With the process modeling approach, the domain-dependent aspects are elicited from the application domain and built into the system through the activities in which the software functions are designed.

The data modelling approach takes a different point of departure by focusing on the data that people work with in the user organization. It has been argued that this was a much more stable foundation for software design than the way in which the users worked. The data modeling approach relates more generally to a focus on this domain:

- *Problem domain:* The part of the context that is administrated, monitored or controlled by a system.

The problem domain is part of what is outside the system (*i.e.,* in the context). For an air traffic control system the problem domain is that part of the context constituted by flights, departures, aircrafts, aircrafts' positions and trajectories, changed altitude, changed speed, *etc*. Everything that the controller in the tower needs to know about to control the air space effectively is in the problem domain. It is fundamental to air traffic control that the controller watches the aircrafts' positions and trajectories on a large monitor displaying data from the radar system rather than looking out the tower's windows with a pair of binoculars. The system creates and maintains the controllers' view of their air space and it is there crucial that the model of the air space (*i.e.,* the problem domain) is in accordance with the controllers' professional language and competence.

With the data modeling approach, the domain-dependent aspects are elicited from the problem domain and built into the system through the activities in which the database and the related software are designed.

3.2. How MDA Apply to Develop Mobile Applications

Mobile applications services have gained a great deal of attention as a way for organizations to use information as a commodity. While many mobile applications services currently exist, very few have semantic descriptions associated with them. Furthermore, from the viewpoint of an "end-user", a given mobile applications service may or may not provide exactly the data that is needed for a given task. However, it may be the case that a group of mobile applications services, when properly composed and integrated, will provide the desired outcomes as in Figure 3. Based on the above, we have identified two primary requirements that are being used to drive our approach: R1 The approach should be able to incorporate the use of both mobile applications services (*e.g.,* services with WSDL specifications only) and semantic mobile applications services (*e.g.,* services with semantic descriptions) R2 The approach should facilitate composition of services to form applications or federations. The intent of requirement *R1* is to leverage existing mobile applications services regardless of the state of their specification. Specifically, the requirement is a recognition of the fact that adoption issues exist in the community and the expectation of widespread use of semantically-rich languages [5]. Requirement *R2* along with the philosophy stated above embody the heart of our approach: a technique for integrating services into composite services by generating an OWLS specification via the use of MDA.

The *Service Model* is a description of how service works (*e.g.,* the semantics of the service).

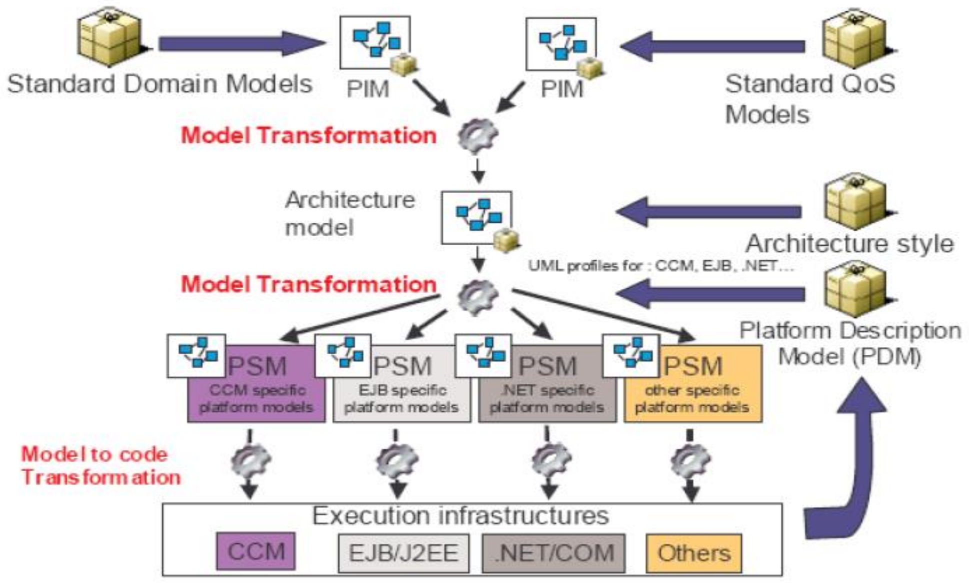
Standard Domain Models
PIM
PIM
Standard QoS Models
Model Transformation
Architecture model
Architecture style
UML profiles for : CCM, EJB, .NET…
Model Transformation
Platform Description Model (PDM)
PSM
CCM specific platform models
PSM
EJB specific platform models
PSM
.NET specific platform models
PSM
other specific platform models
Model to code Transformation
Execution infrastructures
CCM
EJB/J2EE
.NET/COM
Others

Figure 3. MDA Apply to Develop Mobile Applications

Finally, The *Grounding Model* is a description of how to access the mobile applications service. From a WSDL description, an OWL-S description can be generated using XSLT. However, for semantic mobile applications service composition, other information is required to describe the operational behavior of the service (*e.g.,* the service model). We are developing an approach for generating an OWL-S Service Model by taking an XML-based representation of a UML activity diagram and using XSLT to transform the diagram. In this approach, the software developer specifies the operation of a Mobile applications service as a composition of other services using UML. An automated design tool then takes the UML description of the service and converts it into the OWL-S Service Model description. As part of our investigations, we are developing domain models for mobile applications services and mobile applications service composition as a way to facilitate translation via XSLT of WSDL specifications to OWL-S Grounding Models and UML Activity Diagrams to OWL-S Service Models, respectively.

3.4. Process

In our approach we use a product-line strategy to facilitate service composition. Specifically, our approach is based on the development of a product-line architecture that characterizes a family of products that exists in the product line. In a product-line approach, products share a common set of characteristics that are implemented by a set of core assets [5-6]. Individual products in a product line are characterized by variabilities that must be configured or developed to meet the requirements of the individual product as in Figure 4. By reusing assets, which includes processes as well as components, organizations can achieve product differentiation and personalization while benefiting from reuse. Figure 4 shows the context of the approach that we are currently developing. The diagram depicts the activities of the approach as existing on a spectrum of specifications that range from the semantically rich to the syntactically rich. In this context, a series of steps lead to the eventual creation of products that consist of a collection of services. Starting from the left, a domain expert and a software architect work together to develop a number of artifacts based on requirements for a product line including an ontology for the product line, a workflow depicting the series of operations for potential products in the domain, a product-line architecture that specifies the structure of the family of products, and a list of services that meet the requirements of various parts of the product-line architecture.

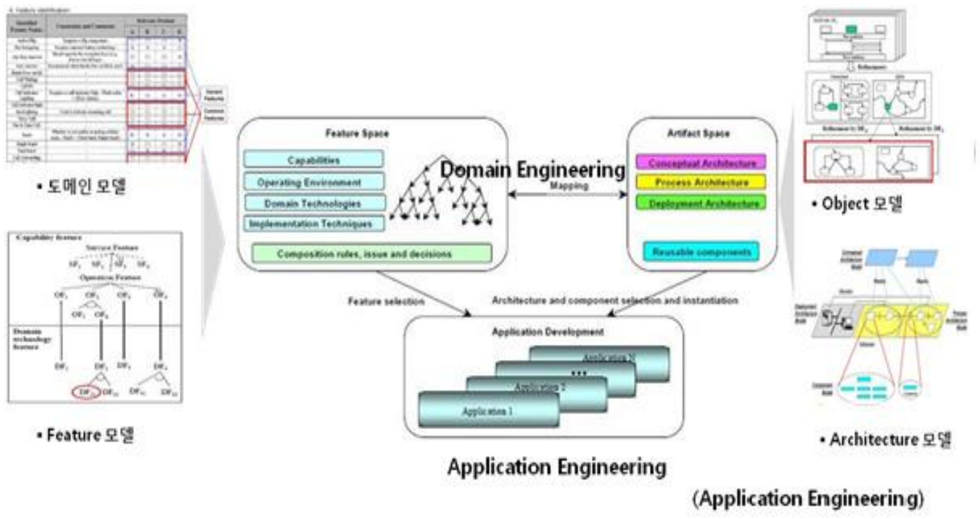

Figure 4. Product Line Apply to Develop Mobile Applications

In this context, all components in the product-line architecture are implemented by services that are either local or distributed. Furthermore, MDA concepts are used to facilitate creation of the appropriate OWL-S specifications. As an example, consider the domain of E-Learning (*e.g.,* delivery and administration of course content in an on-line environment). Requirements for this domain focus on providing content, learning activities (*e.g.*, homework and laboratories), collaboration, and assessment. Within this context, a domain expert and a software architect collaborate to characterize the domain using an ontology and by specifying a workflow. In addition, the architect creates product line architecture for the E-Learning domain while the domain expert identifies potential services that could be used to populate the product line. Using the information provided by the domain expert and software architect, software developers create the software needed to enable a production capability for the product line by developing the software needed to map between different classes of services in the product-line architecture. In addition to creating software that enables the mappings, the software developers create the infrastructure necessary to control the flow of collaborations between classes of services. Since we are using a product-line approach, the mappings are dependent only upon the ontologies for the domain and the collaborations between services well-defined (*e.g.*, content services always provide content). Using the E-Learning example, software developers are responsible for creating the product-line framework for the mobile applications product-line domain by providing software that supports operations like test and quiz management in general as well as the software that will facilitate mapping to specific test and quiz management services. we are developing tools and techniques for generating service based product-line frameworks including using MDA to facilitate creation of mobile applications specifications of a product-line architecture and transformation of the product-line architecture at the PIM-level, to a PSM-level model suitable for generating a framework, where the service mappings provide information on realizing implementations. In addition, we are developing service search technology to support syntactic and semantic service match. Once a product-line framework or generator is developed a domain developer creates new products. At the end of the diagram is the user that utilizes the end products to perform some task. In our approach the domain developer is considered a user of the framework in the sense that we do not require programming knowledge in order to implement a product. That is, the act of product development is an activity of selecting services (*e.g.*, configuration). For instance, in the mobile applications example, the domain developer could be an instructor that wishes to create a course while the domain user is a student.

4. Conclusions and Future Investigations

With the advent of semantic mobile applications services, enterprise applications have the ability of becoming more dynamic and better suited to specific users and customers. However, in order to allow a user to gain control of their application experience, enabling technologies such as model driven architecture are needed to provide the bridge between the technologies being widely used by practitioners and the advanced capabilities of the semantic mobile applications. During the development of this work a set of tools could be perceived to conceive models transformations and a lot of them free for commercial or personal use, however still there is not a mature level for professional use in contrast. With the use of the architecture proposal, the absence of a semantic definition using the UML language has been count out. The mechanism used between creation and validation of the application domain models using description logics, the semantics specifications can be guaranteed for having been used mathematical tests for its formal verification.

Acknowledgements

This work was supported by the Korea National Research Foundation (NRF) granted funded by the Korea Government (Scientist of Regional University No. 2012-0004489).

References

[1] J. S. Mellor, “MDA Destilada: Princípios da Arquitetura Orientada por Modelos”, Editora Ciência Moderna Ltda., **(2003)**, pp. 15-169.
[2] J. Optimal “Model-Driven Development for Java”, http://www.compuware.com/products/optimalj/, Julho, **(2006)**.
[3] S. Brockmans and P. A. Haase “Metamodel and UML Profile for Rule-extended OWL DL Ontologies – A Complete Reference”, AIFB Institute, Karlsruhe Univertity, Germany.
[4] G. C. Gannod and J. T. E. Timm, “An MDA-based Approach for Facilitating Adoption of Semantic Web Service Technology”.
[5] P. Clements and L. M. Northrop, “Software Product Lines – Practices and Patterns”, Addison-Wesley, **(2002)**.
[6] E. W. Clarke and J. M. Wing, “Formal Methods: State of the Art and Future Directions”, Technical Report CMU-CS-96-178, Carnegie Mellon University, August 1996. Group report from the “Strategic Directions in Computing Research Formal Methods Working Group ACM Workshop”, **(1996)**.
[7] W. Ulrich, “A status on OMG architecture-driven modernization task force”, Proceedings EDOC Workshop on Model-Driven Evolution of Legacy Systems (MELS), IEEE Computer Society Digital Library, **(2004)**.
[8] G. C. Gannod and S. Bhatia, “Facilitating automated search for mobile applications services”, Proceedings of the 2004 IEEE International Conference on Mobile applications Services, **(2004)** July.
[9] K. Sycara, M. Paolucci, A. Ankolekar and N. Srinivasan, “Automated discovery, interaction and composotion of semantic mobile applications services”, Journal of Mobile applications Semantics, vol. 1, no. 1, **(2003)**.