Developing Pervasive Systems as Service-Oriented Multi-Agent Systems

Jorge Agüero, Miguel Rebollo, Carlos Carrascosa, and Vicente Julián

Departamento de Sistemas Informáticos y Computación,
Universidad Politécnica de Valencia,
Camino de Vera S/N 46022 Valencia (Spain)
{jaguero,mrebollo,carrasco,vinglada}@dsic.upv.es

Abstract. The development of *Pervasive Systems* is an emerging research topic due to the high heterogeneity of involved technologies and the changing nature of the existing platforms/devices, which make it hard to develop this kind of systems. This work presents a Model Driven Development approach to develop *agent-based* software for Pervasive Environment in order to design and implement application prototypes in an easy and productive way. Our approach provides a method for the specification of *Pervasive Systems*, which allows to face the development of such systems from a higher abstraction level. The deployment over different execution platforms is achieved by means of automatic transformations among models that described entities and the environment (UML-like). The result is a simplified and homogeneous deployment process for *Agent-Based Pervasive Systems*.

Keywords: Multi-Agent Systems, Pervasive Systems, Model Driven Development.

1 Introduction

Pervasive Systems is a paradigm in which technology is virtually invisible in our environment. *Pervasive Systems* are very common nowadays and will be even more in the next years. This is due to the appearance of *new objects* (of daily usage) with different technological capabilities, because they incorporate different electronic devices [14]. So, it is easy to think that this paradigm requires, from a designer point of view, the development of applications in different software and hardware platforms depending on the diversity of the objects in the environment. This raises big challenges.

In this way, the development of *Pervasive Systems* is a complex task, with multiple actors, devices and different hardware environments; where it is difficult to find a compact view of all components. The requirements of this kind of systems are very different[15]. However some of them are basic: (i) integration of external devices and software systems, the services that are provided by *Pervasive System* can be supplied by physical devices and also by existing software systems, and it is essential that the system supports these issues; (ii) the isolation of the technology and the manufacturer-dependent devices, in order to facilitate the development of this kind of systems, the manufacturer dependent devices must be well encapsulated in independent and generic functionalities.

Software engineering based on Multi-Agent Systems (MAS), particularly Open MAS, has the capability to fulfill these requirements[4]. This approach supports the integration of highly heterogeneous platforms, where agents work together to support complex tasks, in a collaborative and dynamic way, with the ability to adapt, coordinate and organize each other[20]. Therefore, it seems appropriated a MAS-based development process to facilitate the orchestration and integration of the functionalities of physical devices.

Moreover, Model Driven Development (MDD) approach can facilitate and simplify the design process and the software quality in the development process of a MAS-based *Pervasive System*. It allows to re-use software and the transformation between models[23]. This methodology can be applied in the development of embedded agents for *Pervasive Systems*, where different technologies and execution platforms coexist. That is, to design applications automatically, where the toolkits guide the developer in the design process, using unified models to apply transformations allowing to obtain specific code for the deployment platforms.

This work proposes to use a MDD approach to facilitate the development process of *Agent-Based Pervasive Systems*, providing the user with a set of abstractions that ease the implementation of *Pervasive Systems* and the deployment of a platform for their execution. To sum up, this work presents: (i) an Environmental meta-model, which allows to incorporate in the modeling process the different environment devices of a *Pervasive System*; and (ii) a layered deployment architecture. This allows to design pervasive applications using high-level abstractions, avoiding the low-level implementation details and, after that, the *Pervasive System* deployment (with embedded agents and devices) is generated by using automatic transformations. In this way, a non-expert programmer will be able to develop *Agent-Based Pervasive Systems*, reducing the gap between the design and the implementation phases.

This document is structured as follows. Section 2 explains the MDD approach and some related works. Section 3 explains how we use MDD for *Agent-Based Pervasive Systems*. Section 4 presents the proposed Environment meta-model. Section 5 shows how to implement the MDD approach in order to develop *Pervasive Systems* and explains the deployment architecture. Finally, some conclusions are presented in section 6.

2 Model Driven Development

The purpose of MDD is to create models legible by computers that can be understood by automatic tools to generate code templates and proof models, integrating them in multiple platforms[5]. Model driven methods make a clear distinction between the problem space (centered on what the system is) and the solution space (centered on how it is implemented as a software product).

2.1 MDD for MAS

From the viewpoint of the design of agent-oriented systems, application development consists of how to obtain the agent code that could be executed in different platforms. That is, to concentrate the development of the application from a unified agent meta-model and, after that, to apply different transformations to obtain implementations for different platforms.

In MAS literature, researchers have formulated a set of typical meta-models that guide the process of MAS development using a model-driven approach. Some works have concentrated their efforts on creating a generic unified model for analyzing and modeling the system using different methodologies. Some of the most significant proposals are: INGENIAS[17], PIM4AGENT[18] FAML[9], Agent UML(AUML)[7] and AML[13]. However, although these proposals use similar components in their meta-models, none of them focus in the development of the MAS as a *Pervasive System*. Those approaches have a limited scope in order to design how the agents perceive/act on the physical world.

The difference of our proposal with respect to the existing approaches is that we propose specific Environment meta-model which provides access to physical devices located in the real world. Furthermore, some of the proposals (FAML, AUML and AML) just define high-level meta-models, and they do not arrive to the implementation phase, and difficulting enormously the developer work when trying to obtain executable code. Moreover, even in those methodologies that include the implementation phase, there exist a gap between design and implementation models. Since there exist notable differences between the high-level agent definition and the implemented agent class.

2.2 MDD for Pervasive Systems

Although the application of model-driven approaches has no been widely adopted in the Pervasive System field, some heterogeneous efforts can be identified that follow this development paradigm. The most important proposals include: CML[19], pervML[22], VRDK[21], among others [11][19]. These approaches using a modeling language to model *Pervasive Systems*, focused on the description of context-aware data (or framework system), and to provide visual tools to facilitate the system development. Some proposals (such as CML and VRDK) focus directly on the device as the main system component and the implementation of the functionality using this device.

Our proposal provides high-level abstractions to decouple the dependencies of the devices with the functionality that the device provides. So, it is based on the encapsulation of the device in an independent service controlled by agents. Similarly as pervML and other[11], we have the aim to decouple the system functionality from the implementation software, using a Service Oriented Architecture (OSGi environment).

The OSGi[1] framework, (Open Service Gateway Initiative) is a standard technology that defines an environment for the execution of services and their

[1] `http://www.osgi.org/`

life-cycle management. The OSGi is originally focused in networked devices market, using services that provide access to different existing device networks (EIB, UPnP, X10, etc.). These features are very interesting for developing *Pervasive Systems*. The OSGi proposal uses a component called *Bundle*, which is Java-base technology that provides a mechanism for releasing and deploying applications.

In summary, our approach allows agents to manage, interact and control physical worlds and thereby create a versatile *Pervasive System* model, since the agents (and developer) interacts with the environment using independent services, which can be combined, composed, and so on.

3 πVOM Approach

Models is a MDD approach are defined through meta-models[23]. One fundamental challenge when defining a meta-model is selecting which concepts or components will be included in order to model the system. To achieve this objective, some of the most well-known approaches in the area of MAS and *Pervasive Systems* were studied (mentioned in Section 2). The purpose of this analysis was to extract the common features from the methodologies studied and adapt them to the current proposal, specifying a generic platform-independent meta-model of an *Agent-Based Pervasive System.*

This set of meta-models is created by the detection of common concepts in an iterative cycle consisting of a bottom-up analysis. Common elements in existing MAS and *Pervasive Systems* methodologies, have been identified and incorporated to the Computation Independent Model (CIM) level (see Figure 1). These models can be adjusted as MDD models that specify the concepts of the system, as roles, behaviors, tasks, environment, interactions or devices. The models can be used to describe an *Agent-Based Pervasive System* without focus on platform-specific details and requirements, as a Platform Independent Model (PIM). After that, it is possible to transform PIM models into Platform Specific Models (PSM). Figure 1 shows relationships between the concepts of different MDD models and their transformations.

The proposed set of meta-models integrate different MAS modeling approaches, and it mainly focuses on the integration of Services and MAS techniques for supporting dynamical and open MAS societies[4]. This set of meta-models is called πVOM (*Platform-Independent Virtual Organization Model*). The main views of πVOM are the *Structure*, *Functionality*, *Normative*, *Agent*, and its *Environment.* Therefore, to model the characteristics of these components in our approach, five key concepts are used: *Organizational Unit*, *Service*, *Environment*, *Norm*, and *Agent*[4]. According to this, πVOM is structured in five meta-models or views, which cover the above mentioned key aspects.

These five elements describe those members (entities) that form the organization: the topology of the organization; the services and features that the organization offer; the evolution of the organization over time; the environment where the organization is situated; and the rules about the behavior of members respectively. These five elements are described in more detail in [3]. In this paper, an extension of the Environment meta-model is described in detail. This

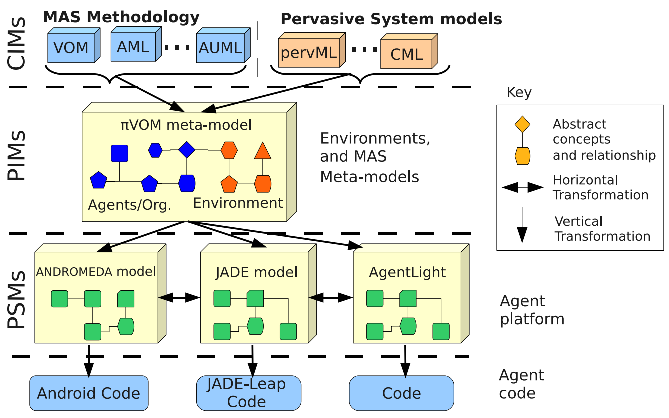

Fig. 1. MDD for MAS

meta-model allows agents to easily interact with the different physical devices in the environment, which enables agents to manage, perceive and be located in a *Pervasive System*. This new Environment meta-model is able to capture the details and requirements of a *Pervasive Systems* (whereas the old meta-model no). Next section explains in detail this Environment meta-model.

4 Environment Meta-model: An Ubiquitous Meta-model

This new meta-model is focused on describing environmental components, representing perceptions and acts of the devices. Moreover, it defines permissions for accessing them. The proposed Environment meta-model is showed in Figure 2. The *Environment* concept represents the physical world where the agent is located. The *Environment* can be perceived using the *Perceive* relationship. The *Environment* is a recursive model, which allows the creation of sub-worlds (workplaces) contained one inside another. These sub-worlds may be connected with other neighboring worlds using the relationship *neighborhood*.

The *Environment* consists of *Resources* and *Devices*. A *Resource* is a software-oriented environment component and its access is performed by a standard protocol, not requiring an adjustment of the device drivers. A *Device* represents a physical component, which access is made directly through a physical interface (sensor or actuator). In this case, it is required to bind the low-level driver (firmware) to a software component.

The *EnvironmentService* concept allows to use the functionalities of the physical devices through a service, decoupling the low-level abstraction. *Resources* and *Devices* are accessed through an *EnvironmentPort*. An *entity* (agent or organization) is in charge of managing the access permissions to these elements using the *Port* abstraction. Each *Port* is controlled by an entity and it can be used by one or more roles (played by the entities). A *Port* represents a point of interaction between the *Entity* and physical devices, and it serves as an interface

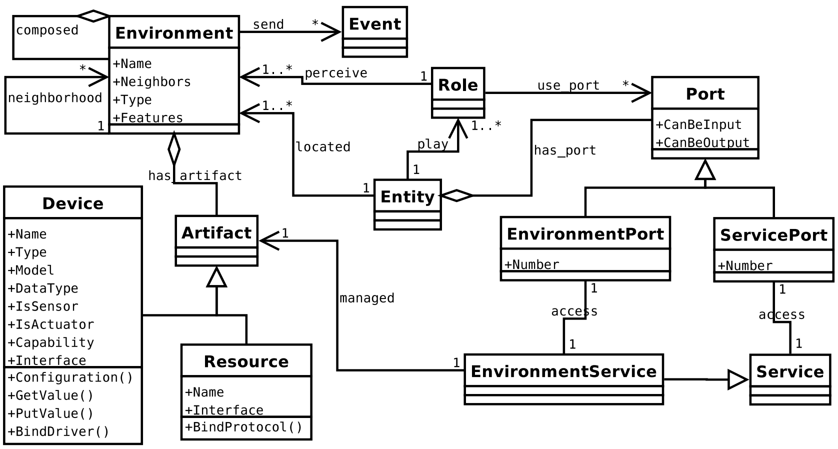

Fig. 2. Environment meta-model of πVOM

Table 1. Main concepts used in the Environment meta-model

πVOM concepts	Description
Entity	Specification of something that has an individual existence in the MAS.
Role	Specification of a behavioral pattern expected from some MAS entities.
Service	A single activity (or complex block of activities) that represents a functionality of the agents/devices/resources.
Environment	Physical world, workspace where the agents are situated.
Resource	Specification of a software artifact, that has a reasonable representation in the environment, which can be perceived and shared using data protocols.
Device	Specification of a hardware artifact, which we can perceive and act through low-level interfaces (using proprietary firmware).
Port	This abstraction is an interface to the service, that allows the input/output of data.
EnvironmentPort	Access point to interact with the environment (with the physical world).
ServicePort	Access point to use an agent-based service.
EnvironmentService	High-level functionality, which decouples the protocol or firmware of the environmental artifacts.

to the physical world. Finally, all these basic concepts and relationships will enable agents (and users) to create new high-level representations of the *Pervasive Systems*, as context-aware models[6], Service discovery/composition models[10], and adaptation models[24]. Table 1 summarizes the main concepts used in the Environment meta-model.

Finally, in order to illustrate the use of this meta-model, a typical case study of a *Pervasive System* is analyzed. The case study presents a little office where some pervasive services are provided. The office is structured in three locations: the reception desk, the head office and meeting room. Every location provides a lighting service which is activated when is detected in the space the presence of a user. The head office and the meeting room provide services for multimedia content (audio, video, and presentations). When employees make a presentation or use multimedia devices in the meeting room, the light intensity is decreased and the blinds are automatically closed to improve the visibility. The ubiquitous office provide others services, such as: security and recording; but, due to the space limitations of the paper, these functionalities are not described.

In order to model the ubiquitous office, the developer must specify the different components which model the different devices, resources, environments, services and agents, of the *Pervasive System*. Figure 3 shows the partial model (focused on meeting room) of the office using πVOM Environment meta-model. Figure 3 shows that the ubiquitous office uses different devices, such as: smart bulbs (using X10 protocol), gradual light bulbs and blinds automatic, are used to control the lighting room. Also, the ubiquitous office uses infrared sensors that are motion detectors and cameras. Furthermore, the developer can create new services such as Security and Recording, using the basic services (*EnvironmentService*) offered by each device.

5 Implementing Agent-Based Pervasive Systems

This work proposes to use a homogeneous and unified model for implementing *Agent-Based Pervasive Systems* permitting its translation into different execution MAS platforms through MDD, in which agents act/perceive about the environment and thereby manage/control the *Pervasive System*. This means that the user can design a *Pervasive Systems* with a unified, intuitive, visual model, i.e., with a high level of abstraction. Then, the user can get the agent code automatically using MDD with minimal user intervention. Finally, the drivers or firmware must be added to support environmental devices. These drivers will be encapsulated within a service (an OSGi service), to export their functionality as a high-level abstraction (which will be managed by agents). Finally, this code should be compiled for execution over an OSGi framework, as is showed in Figure 4.

5.1 Development Process

The design process starts trying to modelize the agents and the environmental devices using the abstract components of the proposed meta-models (commented in the previous section). The *Pervasive System* design process is formed by a set transformations that finally will obtain the OSGi-Java code. In order to do these steps, a set of tools, which support the process, are required. The tools used at each stage of the design can be summarized as follows (see Figure 4):

Step 1: At the beginning, the developer must create the different diagrams (using the EMFGormas toolkit[16]) which model the different devices, resources or services of the agents. To perform this step, an Eclipse IDE with a set of *plugins* is employed[16]. These plug-ins are mainly *EMF*, *Ecore*, *GMF* and *GEF*, which allow the user to draw the models that represent the *Pervasive System*.

Step 2: Once the model has been developed, it is necessary to select in which platforms the user wants to execute the agents. This phase corresponds with the PSM model definition of each agent. To do this, it is necessary to apply a model-to-model transformation (PIM-to-PSM). This is done using the Eclipse IDE and the ATL *plug-in* incorporating the appropriated set of transformation rules. It

Table 2. Transformation rules between agent meta-model and JADE-Leap model

Rule	Concept	Transformation
1	Agent	πVOM.Agent $\Rightarrow$ JADE.Agent
2	Behaviour	πVOM.Behaviour $\Rightarrow$ JADE.ParallelBehaviour
3	Capability	πVOM.Capability $\Rightarrow$ JADE.OneShotBehaviour
4	Task	πVOM.Task $\Rightarrow$ JADE.Behaviour

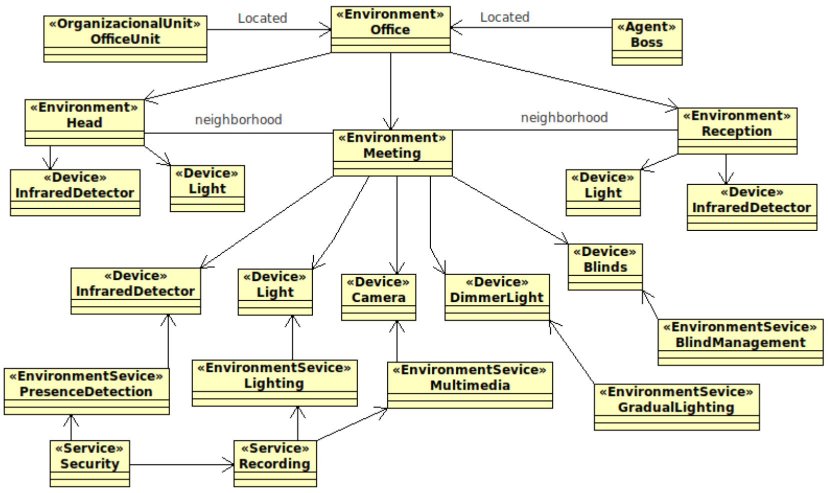
«OrganizacionalUnit» OfficeUnit
Located
«Environment» Office
Located
«Agent» Boss
«Environment» Head
neighborhood
«Environment» Meeting
neighborhood
«Environment» Reception
«Device» InfraredDetector
«Device» Light
«Device» Light
«Device» InfraredDetector
«Device» InfraredDetector
«Device» Light
«Device» Camera
«Device» DimmerLight
«Device» Blinds
«EnvironmentSevice» BlindManagement
«EnvironmentSevice» PresenceDetection
«EnvironmentSevice» Lighting
«EnvironmentSevice» Multimedia
«EnvironmentSevice» GradualLighting
«Service» Security
«Service» Recording

Fig. 3. Partial view of the office using Environment meta-model of πVOM

is important to remark that the same agent model can be transformed into different specific agent platforms. Table 2 illustrates the agent transformations, from agent meta-model of π *VOM* to JADE-Leap[8]. These rules are a subset of the transformation rules needed in this phase, which are explained in detail in [2]. In this way, agent concepts are mapped from source models to target models, and agent components are transferred or changed from one model to another.

Step 3: After the second step, the developer must apply a transformation to convert the models into the MAS code (OSGi-based). To do this, we must use a PSM-to-code transformation. In this case, we use *MOFScript* which is an Eclipse plug-in that uses templates to do the translation. These templates have been developed for two MAS platforms: JADE-Leap and ANDROMEDA[1]. Figure 5 illustrates how one rule is implemented using *MOFScript*. Part of the code of the rule shows the transformation of the *agent* concept.

Step 4: Finally, the process finishes by adding the necessary drivers for the different environment devices needed in the *Pervasive System*. All the functionality of physical devices are encapsulated as OSGi services, which allow agents to use them without worrying about low-level features. However, it is necessary to provide the driver/firmware/protocol of the new devices that are not in the OSGi *bundle* library. This application has a *bundle* library, which store the *EnvironmentService* (service devices) for frequent use or re-use.

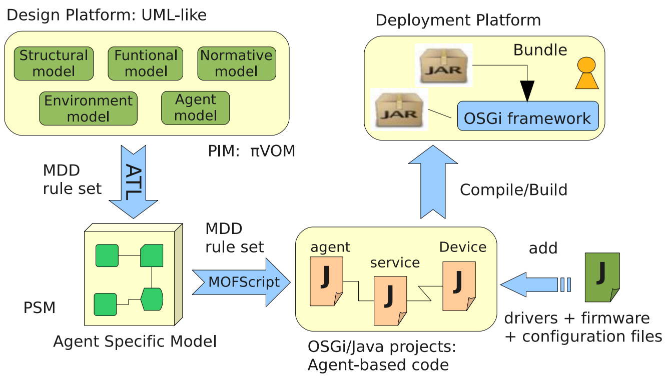

Fig. 4. Implementing Agent-based Pervasive System using MDD transformations

```
texttransformation AGENT2ANDROMEDA (in myAgentModel:uml2)
//Rule1: Agent transformation
uml.Package::mapPackage () {
  self.ownedMember->forEach(c:uml.Class)
     if (c.name != null)
       if (c.name = Agent) c.outputGeneralization()   }
uml.Class::outputGeneralization(){
   file (package_dir + self.name + ext)
   self.classPackage()
   self.standardClassImport ()
   self.standardClassHeaderComment ()
   <% public class %> self.name <% extends Agent { %>
      self.classConstructor()
      <% // Attributes  %>
      self.ownedAttribute->forEach(p : uml.Property) {
            p.classPrivateAttribute()
      } newline(2) ...
```

Fig. 5. *Agent* translation using MOFScript

5.2 Deployment Platform

Implementing *Pervasive Systems* is a challenging and exciting task, since a solid background knowledge about how to implement this kind of systems do no exist. Many research efforts are currently being developed on prototype implementations[15]. However, some *Pervasive System* prototypes share a common architectural style, which correctly fits the requirements of this systems.

As discussed above, the requirements of this kind of systems are basically two: (i) is essential that the *Pervasive System* must support the integration of services provided by external devices and software systems (as another services); (ii) the isolation of the low-level abstraction of the devices (manufacturer dependent), the environmental devices must be well encapsulated in independent and generic functionalities. Therefore, an architecture style that meets these requirements is the well-known layered architecture[15]. By means of this architecture, the sys-

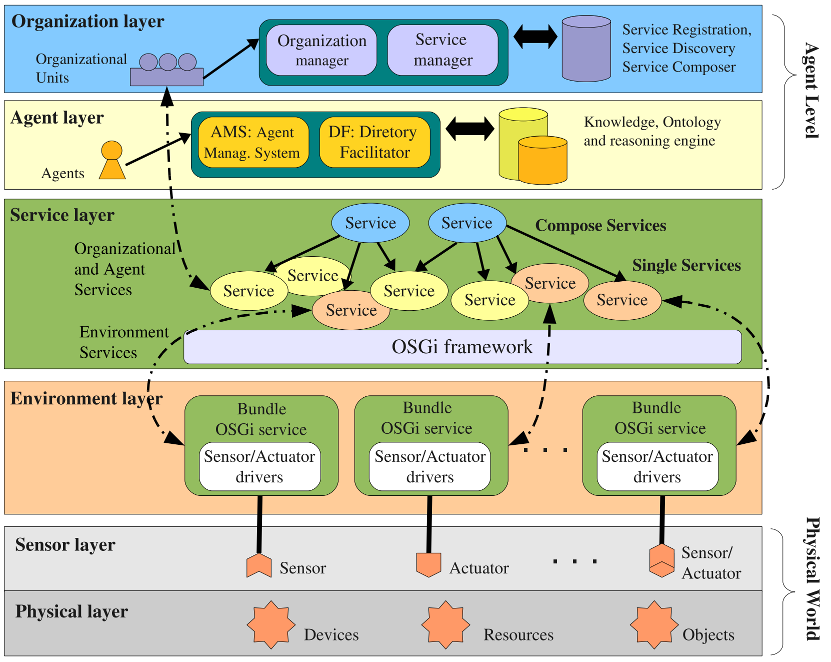

Fig. 6. Proposal architecture of Pervasive System

tem elements are organized in different levels with well-defined responsibilities. Our proposal follows this architecture style. Figure 6 shows our deployment platform, which is a framework based on OSGi technology. The main layers of this deployment platform are:

Physical layer, has the resources/devices that are perceived in the environment. Fully represents the real world, where the *Pervasive System* is located.

Sensor layer, has the responsibility of accessing to physical devices, through actuators and sensors, which allow to change or read the state of the devices.

Environment layer, has the responsibility of encapsulating the manufacturer dependent technology of the environment devices. The drivers that conform this level directly export their functionalities through a *bundle*. The *bundles*, which manage similar devices of software systems from different technology or vendors, are implemented as a common interface, in order to provide a uniform way of communicating within the environment devices.

Service layer, provides the system functionality, offering services that the *Pervasive System* must supply. The services are provided by the devices located in the physical world, and by the MAS entities (agent or organizational unit). Also, at this level the services can be *single* or *composite services*, which are formed by the composition of other services.

Agent layer, supports the mechanisms in order to register, de-register and discovery of agents. In this layer the agents work together through different interactions to support complex tasks in a collaborative and dynamic way. This layer also supports the information management (*knowledge*) and the needed *knowledge models*, including the *reasoning engine* and *ontologies* needed by the agents. Furthermore, this layer provides the necessary mechanisms to support the communications and needed languages used by agents, such as FIPA ACL.

Organizational layer, is used as a regulatory framework for the coordination, communication, and interaction among different computational entities. This layer is formed by a set of individuals and institutions that need to coordinate resources and services across institutional boundaries. This layer supports high level interoperability to integrate diverse information systems in order to share knowledge and facilitate collaboration among entities. This layer is an open system formed by the grouping and collaboration of heterogeneous entities. From a technical view, these functionalities are obtained using the THOMAS platform[12], which consists basically of a set of modular services that enable the development of agent-based organizations in open environments.

6 Conclusions

This work presents the application of the ideas proposed by the MDD for the design of *Agent-Based Pervasive Systems*. Although the use of MDD refers primarily to methodologies of object-oriented software, it was verified that the approach can be adopted in the development of *Agent-Based Pervasive Systems*.

The application of these technologies to the development of *Pervasive Systems* can provide many benefits: (i) the *Pervasive System* can be adapted to new devices technologies in an easy way, because the approach uses high-level abstraction that are not so hardware dependent; (ii) the development of a *Pervasive System* is more intuitive using this model-driven method than using classical approaches, because the developer models the *Pervasive System* as an UML-like model avoiding technical details; (iii) the implementation of *Pervasive System* over a service-based framework (as OSGi) using agent-oriented software engineering, allows agents to manage the services, the context, adapting the environment, giving the possibility to create more advanced and powerful models.

Future work of this research will focus on developing an explicit support to Context-Awareness specification, using ontologies to describe the contextual information, allowing new knowledge in the environment to be inferred.

Acknowledgment. This work was partially supported by TIN2009-13839-C03-01 and PROMETEO/2008/051 projects of the Spanish government and CONSOLIDER-INGENIO 2010 under grant CSD2007-00022.

References

1. Agüero, J., Rebollo, M., Carrascosa, C., Julián, V.: Towards on embedded agent model for Android mobiles. In: Proceedings of Mobiquitous 2008, pp. 1–4 (2008)

2. Agüero, J., Rebollo, M., Carrascosa, C., Julián, V.: Agent design using Model Driven Development. In: 7th Int. Conf. on PAAMS 2009, vol. 55, pp. 60–69 (2009)
3. Agüero, J., Rebollo, M., Carrascosa, C., Julián, V.: MDD for Virtual Organization design. In: Trends in Int. Conf. on PAAMS 2010, vol. 71, pp. 9–17 (2010)
4. Argente, E., Julian, V., Botti, V.: MAS Modeling Based on Organizations. In: Luck, M., Gomez-Sanz, J.J. (eds.) AOSE 2008. LNCS, vol. 5386, pp. 16–30. Springer, Heidelberg (2009)
5. Atkinson, C., Kuhne, T.: Model-driven development: a metamodeling foundation. IEEE Software 20(5), 36–41 (2003)
6. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. International Journal of Ad Hoc and Ubiquitous Computing 2(4), 263–277 (2007)
7. Bauer, B.: UML Class Diagrams Revisited in the Context of Agent-Based Systems. In: Wooldridge, M.J., Weiß, G., Ciancarini, P. (eds.) AOSE 2001. LNCS, vol. 2222, pp. 101–118. Springer, Heidelberg (2002)
8. Bergenti, F., Poggi, A.: LEAP: A FIPA Platform for Handheld and Mobile Devices. In: Meyer, J.-J.C., Tambe, M. (eds.) ATAL 2001. LNCS (LNAI), vol. 2333, pp. 436–446. Springer, Heidelberg (2002)
9. Beydoun, G., Low, G., Henderson-Sellers, B., et al.: FAML: A Generic Metamodel for MAS Development. IEEE Trans. on Software Engineering, 841–863 (2009)
10. Brønsted, J., Hansen, K., Ingstrup, M.: A survey of service composition mechanisms in ubiquitous computing. In: RSPSI 2007 at Ubicomp (2007)
11. Cano, J., Madrid, N., Seepold, R., Aguilar, F.: Model-driven development of embedded systems on OSGi platforms. In: FDL 2007, pp. 1–6 (2007)
12. Carrascosa, C., Giret, A., Julian, V., Rebollo, et al.: Service oriented multi-agent systems: An open architecture. In: AAMA 2009, pp. 1–2 (2009)
13. Cervenka, R., Trencansky, I.: The Agent Modeling Language – AML. Whitestein Series in Software Agent Technologies and Autonomic Computing (2007)
14. Davidsson, P., Boman, M.: Distributed monitoring and control of office buildings by embedded agents. Inf. Sci. Inf. Comput. Sci. 171(4), 293–307 (2005)
15. Endres, C., Butz, A., MacWilliams, A.: A survey of software infrastructures and frameworks for ubiquitous computing. Mobile Inf. Syst. 1(1), 41–80 (2005)
16. Garcia, E., Argente, E., Giret, A.: A modeling tool for service-oriented Open Multiagent Systems. In: Yang, J.-J., Yokoo, M., Ito, T., Jin, Z., Scerri, P. (eds.) PRIMA 2009. LNCS, vol. 5925, pp. 345–360. Springer, Heidelberg (2009)
17. Garca-Magario, I., Gómez-Sanz, J., Fuentes, R.: INGENIAS Development Assisted with Model Transformation By-Example. In: PAAMS 2009, pp. 40–49 (2009)
18. Hahn, C., Madrigal-Mora, C., Fischer, K.: A platform-independent metamodel for multiagent systems. In: AAMAS 2008, vol. 18(2), pp. 239–266 (2008)
19. Henricksen, K., Indulska, J.: Developing context-aware pervasive computing applications: Models and approach. Pervasive and Mobile Comp. 2(1), 37–64 (2006)
20. Huhns, M., Singh, M., Burstein, M., et al.: Research directions for service-oriented multiagent systems. IEEE Internet Computing 9(6), 65–70 (2005)
21. Knoll, M., Weis, T., Ulbrich, A., Brändle, A.: Scripting your home. In: Location and Context-Awareness, pp. 274–288 (2006)
22. Munoz, J., Pelechano, V., Fons, J.: Model driven development of pervasive systems. In: International Workshop MOMPES 2004, pp. 3–14 (2004)
23. OMG: Object management group. MDA guide version 1.0.1 (June 2008), `http://www.omg.org/docs/omg/03-06-01.pdf`
24. Poladian, V., Sousa, J., et al.: Task-based adaptation for ubiquitous computing. IEEE Trans. on System, Man, and Cybernetics 36(3), 328–340 (2006)