

Pervasive Systems Development with the Model Driven Architecture

Javier Muñoz *

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camí de Vera s/n, E-46022, Spain
jmunoz@dsic.upv.es

1 Introduction: Pervasive Systems

Computing based systems growth is arriving to all environments of our daily life. Pervasive systems live around us providing services to the inhabitants of a home, the workers of an office or the drivers in a car park. We know that requirements for current and future pervasive systems involve a great diversity of types of services [16]. Such different services as multimedia, communication or automation services need hardware devices that different manufacturers provide. These devices live in several networks running on different platforms.

The development of such systems is very hard because they have to achieve devices interoperability in an heterogeneous environment in order to satisfy system requirements. This situation requires solid engineering methods for developing robust systems. For many years, software engineers have used conceptual models for providing structured and high level views of software systems which are used as a guide for the coding task. Current trends propose that models must play a more important role. These approaches claim that models must be the most important artifacts in the development cycle, therefore systems developers build and transform systems models in order to achieve automatic code generation. The Object Management Group (OMG) has reflected these trends in its Model Driven Architecture (MDA) [11] proposal, that is being adopted as a new industrial strategy. MDA guidelines application to pervasive systems development can help to build better systems in an easy way.

The thesis presented in this paper aims to improve current state of the art of pervasive systems development techniques by means of proposing a **MDA based method for pervasive systems development**.

2 Current Problems in Pervasive Systems Development

Currently, technologies for pervasive systems development use *low level abstraction concepts*: I/O ports, data flows, routers, etc. Therefore pervasive systems developers spend a lot of effort in solving problems about technology instead

* Advisor: Vicente Pelechano

of focusing on satisfying systems requirements. Developers have to battle with data format conversions or protocol adaptations.

Modeling methods for real-time and embedded systems are being applied to pervasive systems development [13, 8, 12]. Some parts of a pervasive system can be seen as embedded or real-time systems but, usually, a complete pervasive system have to integrate many subsystems of different nature for providing heterogeneous services (devices control, multimedia, communications, etc.). The techniques applied currently provide to the developer low-level abstraction constructs that directly represent hardware entities. Following this approach the pervasive system description has a *strong dependence on the hardware system*. Moreover, any change in the system requirements usually affects to a wide portion of the system model. Another weak point of these approaches is the *lack of well defined and automated transformations* from system models to implementation. Finally, most of these techniques assume that developers should program system devices wich is a very strong assumption in current pervasive systems which make extensive use commercial off-the-shelf (COTS) devices ¹.

In order to improve this situation, this work proposes to increase the abstraction used in the development of pervasive systems through the use of conceptual modelling and automatic code generation techniques following the Model Driven Architecture (MDA) proposal. This approach provides many benefits to the development of pervasive systems. The pervasive system developers focus on describe the system using high abstraction level primitives. This is more intuitive than in traditional methods and, therefore, the development time falls dramatically due to the developers do not spend time solving technological problems.

3 Model Driven Architecture for Pervasive Systems Development

MDA is a proposal for developing software using models. So far, software development only use models for documentation or as a guide for impelementing. The goal of MDA is to promote that models become the kernel of the development process. In order to reach that goal, MDA proposes to start the process by building high level abstraction models (Platform Independent Models, PIM) and refine them until obtain models that directly represents the final software (Platform Specific Model, PSM).

The MDA proposal defines the building blocks for constructing model driven methods, but it does not specify concrete techniques to use in each building block. In order to make MDA more useful, a MDA method should provide techniques for each MDA step. Our approach proposes the following techniques for applying MDA (see Fig. 1) to pervasive system development:

1. **A precise language for building Platform Independent Models (PIMs).**
Pervasive system developers use this language for precisely describing the

¹ We extend the definition of COTS to include hardware devices

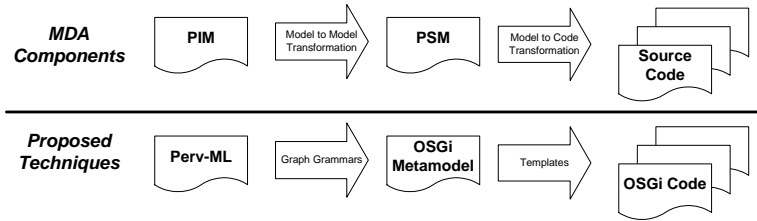


Fig. 1. MDA building blocks and our proposed techniques for pervasive computing

system with high-level constructs. We have defined the Pervasive Modeling Language (Perv-ML) in order to build PIMs.

2. One or many **modelling languages for building Platform Specific Models** (PSMs). The constructs of these languages must be direct representations of constructs of the technology they model. This means that, for instance, a language for modelling an object oriented language must have elements like *class*, *attribute*, *reference*, etc. Currently, we are working in the development of a language for modelling an OSGi system. OSGi is a Java middleware initially created for hosting software of residential gateways.
3. **PIM to PSM transformations**. These transformations define how a PIM can be converted to a PSM. Currently, model transformations is a hot research topic. We apply graph grammars for defining the transformations from Perv-ML to OSGi.
4. **PSM to source code transformations**. Finally, the code generation from the PSMs is the last step of the development method. We are applying templates to the elements of models in order to obtain the source code.

3.1 Pervasive Modelling Language (Perv-ML): The PIM Language

Our approach considers that the development of pervasive systems consists in the the selection of the suitable commercial off-the-shelf (COTS) devices and the software system that integrates them in order to provide the services that users require.

Pervasive Modelling Language (Perv-ML) is a language designed in the context of this thesis with the aim of providing the system analyst with a set of constructs that allow to precisely describe the pervasive system. Perv-ML promotes the separation of roles where developers can be categorized as analysts and architects. Systems analysts capture system requirements and describe the pervasive system at a high level of abstraction. Analysts use three diagrams (models) that constitute what we call the Analyst View. On the other hand, system architects specify what COTS devices and/or software systems realize system services. Architects build other three models that constitute what we call the Architect View. Fig. 2 shows the language organization. The dashed arrow of Fig. 2 defines the construction order of the conceptual models that our approach proposes.

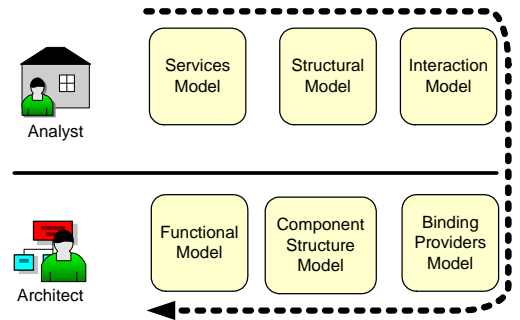


Fig. 2. The six models of Perv-ML

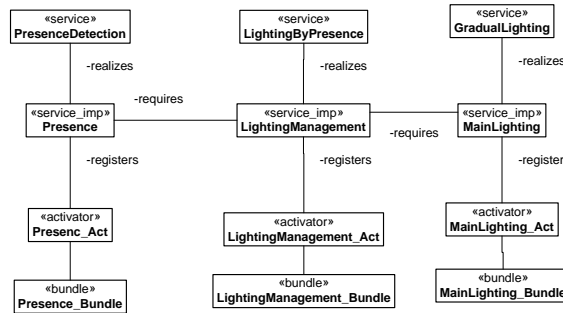
For the specification of the language semantics we have developed a MOF metamodel with a textual description of every metamodel element. In order to provide the language with a graphical representation we have developed an UML profile.

3.2 OSGi Metamodel: the PSM Language

There are a lot of implementation technologies for developing pervasive systems. Using only a low-level technology for control (LonWorks, EIB, UPnP) , data (Ethernet, Bluetooth, WiFi) or multimedia (IEEE1394, HAVi) networks is not possible because of the diversity of services required, therefore we have selected a middleware platform that has bridges to many of them and provides high-level constructs for building pervasive systems.

The Open Service Gateway Initiative (OSGi) [9] is an association of companies, that includes Sun Microsystems, IBM, Oracle and Nokia, created with the aim of developing an open standard for service gateways. A service gateway is the platform where resides in the software for providing home services. It manages home devices and it communicates with external networks. The standard defines Java APIs for libraries that the OSGi platform provides. These libraries build a framework that offers high level abstraction constructs in order to build OSGi-based software. Therefore, it is a very suitable platform for developing pervasive systems.

In order to integrate OSGi in our development method, we have to be able to create models that are built using OSGi constructs. We are developing an OSGi metamodel for defining these constructs and their relationships. The models are represented using an UML profile. Fig. 3 shows a simple OSGi PSM that is depicted with a piece of the code that it generates. The mappings between the elements to the PSM and the OSGi code are one-to-one.



```
import org.osgi.framework.*;

public class LightingManagement_Activator implements BundleActivator {

    private ServiceRegistration servReg = null;
    private ServiceReference lighting,presence;

    public void start(BundleContext bc){
        lighting =
            bc.getServiceReferences("xxx.yyy.zzz.GradualLighting","(PervMLid=MainLighting)") [0]
        presence =
            bc.getServiceReferences("xxx.yyy.zzz.PresenceDetection","(PervMLid=Presence)") [0]
        Activation PC(lighting,presence);
        Properties props = new Properties();
        props.put("description","Manages Room Lighting");
        props.put("HAMONid","LightingManagement");
        servReg = bc.registerService("xxx.yyy.zzz",PC,props);
    }
    .....
}
```

Fig. 3. A simple OSGi PSM an its related source code

3.3 Graph Transformations. Defining the Model Transformation Engine

The definition of transformations between PIM and PSM involve jumping a wide gap between abstraction levels. Currently standards for the definition of transformations do not exist [1]. OMG published a *Request For Proposal* [10] in order to achieve a language for defining transformation between metamodels built with its Meta Object Facility (MOF). In the meantime, we are using graph transformations and graph grammars [2] as the model transformation engine. There exist many works [3, 14, 15] that propose graph grammars as a suitable technique for model transformation. From a mathematical point of view, a model can be seen as a graph where model elements are labeled nodes and the relationships between model elements are edges. In this way we can apply all the existing knowledge for defining graph transformations in order to achieve model transformations in the MDA context. Graph grammars have many advantages over other proposed techniques: a formal mathematical sound, algorithms for applying them and a graphical representation for defining intuitively transformations.

Fig. 4 shows two rules for model transformation from Perv-ML models to OSGi-based models. Every rule is composed by a Left Hand Side (LHS), that defines a pattern to be matched in the source graph, and a Right Hand Side that defines the replacement for the matched subgraph. For instance, first rule says that when a Perv-ML Component element is found it must be transformed

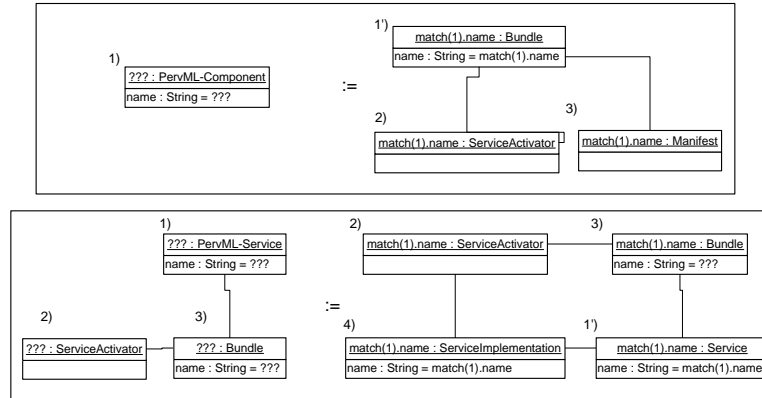


Fig. 4. Two rules that define models transformation

into a **Bundle** element and references to a **ServiceActivator** and **Manifest** elements have to be created and linked to the **Bundle**. Following this approach, transformation from Perv-ML models to OSGi-based models is defined following a set of rules like those defined in this section.

4 Main Thesis Contributions

This work contributes to several research fields: pervasive systems development, model driven development, conceptual modelling, model transformations, meta-modelling, domain specific modelling languages, automatic code generation, ... Specifically, the thesis develops:

- a complete application of the MDA proposal to an specific domain.
- a modelling language for the specification of pervasive systems, what supposes defining a MOF metamodel, a textual description, an UML profile, a method of application and cases of study.
- a modelling language for the specification of OSGi systems, what supposes defining a MOF metamodel, a UML profile and the generation of OSGi code. Other MDA-based methods will be able to use this language for specifying its PSMs.
- a graph grammar for the transformation of Perv-ML models to OSGi models.
- a prototype tool for supporting the complete method (the building of Perv-ML models and its automatic transformations to OSGi code).

It's planned to validate the whole method developing real-world projects in an industrial environment. We have the support of SOSY Inc. and CARE Technologies, a company that implements and markets an industrial CASE tool for building informations systems. This tool gives support to a model driven method with automatic code generation capabilities that has been developed in our research group.

When this work finishes the experiences achieved with the development of this method will help in improving new model driven methods. On the other hand, the research community in pervasive systems will have available a method for developing robust systems in an intuitive way. In general, the thesis will be a good example of the application of the technologies that builds the MDA proposal (UML, MOF, UML Profiles, etc.).

Several publications have been already released as result of this work. The position paper [4] outlines our initial goals and strategy which is focused on home automation systems. In [5] a first version of a modelling language for home automation systems is presented. The work introduced in [6] describes our methodological approach to the development of pervasive systems. This work applies the MDA and its is focused on the description of Perv-ML. The work introduced in [6] describes our methodological approach to the development of pervasive systems. This work applies the MDA and its is focused on the description of Perv-ML. Finally, the paper [7] briefly shows the global strategy and the proposed techniques for each MDA building block.

References

1. Krzysztof Czarnecki and Simon Helsen. Classification of Model Transformation Approaches. In *2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture*, 2003.
2. H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors. *Handbook on Graph Grammars and Computing by Graph Transformation*, volume 2 Applications, Languages and Tools. World Scientific Publishing Co., Inc., 1999.
3. G. Csertan and G. Huszerl and I. Majzik and Z. Pap and A. Pataricza and D. Varro. VIATRA: Visual Automated Transformations for Formal Verification and Validation of UML Models. In *Proceedings of ASE 2002: 17th IEEE International Conference on Automated Software Engineering*. IEEE Computer Society, 2002.
4. Javier Muñoz and Joan Fons and Vicente Pelechano and Oscar Pastor. Hacia el Modelado Conceptual de Sistemas Domóticos. In *VIII Jornadas de Ingeniera de Software y Base de Datos*, pages 369 – 378, November 2003.
5. Javier Muñoz and Joan Fons and Vicente Pelechano and Oscar Pastor. Desarrollo de sistemas domóticos guiado por modelos. In *VII Workshop Iberoamericano de Ingenieria de Requisitos y Ambientes Software (IDEAS)*, pages 228 – 233, May 2004.
6. Javier Muñoz and Vicente Pelechano and Joan Fons. Model Driven Development of Pervasive Systems. In *I International Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MOMPES)*, pages 3 – 14. Turku Centre for Computer Science, June 2004.
7. Javier Muñoz and Vicente Pelechano and Joan Fons. Model driven development of pervasive systems. *ERCIM News*, (58):50 – 51, July 2004.
8. R. Machado, J. Fernandes, and H. Santos. A Methodology for Complex Embedded Systems Design: Petri Nets within a UML Approach. *Architecture and Design of Distributed Embedded Systems*, pages 1–10, 2001.
9. D. Marples and P. Kriens. The Open Services Gateway Initiative: An Introductory Overview. *IEEE Communications Magazine*, 39(12):110–114, 2001.

10. Object Management Group. OMG MOF 2.0 query, views, transformations request for proposals.
11. Object Management Group. Model Driven Architecture Guide, 2003.
12. Mor Peleg and Dov Dori. Extending the Object-Process Methodology to Handle Real-Time Systems. *Journal of Object-Oriented Programming*, 11(8):53–58, 1999.
13. Bruce Powel. *Real-Time UML: Developing Efficient Objects for Embedded Systems (2nd Edition)*. Addison-Wesley, 2001.
14. R. Heckel and J. Küster and G. Taentzer. Towards automatic translation of UML models into semantic domains. In *Proc. AGT 2002: Workshop on Applied Graph Transformation*, pages 174–188, 2002.
15. Shane Sendall. Combining Generative and Graph Transformation Techniques for Model Transformation: An Effective Alliance? In *2nd OOPSLA Workshop on Generative Techniques in the context of Model Driven Architecture*, 2003.
16. R. Want, T. Pering, G. Borriello, and K.I. Farkas. Dissapearing Hardware. *Pervasive Computing*, 1(1), 2002.