

Building Ubiquitous Business Process following an MDD approach*

Pau Giner

Departamento de Sistemas
Informáticos y Computación
Universidad Politécnica de Valencia
Camí de Vera, s/n 46022
Valencia, España
pginer@dsic.upv.es

Victoria Torres

Departamento de Sistemas
Informáticos y Computación
Universidad Politécnica de Valencia
Camí de Vera, s/n 46022
Valencia, España
vtorres@dsic.upv.es

Vicente Pelechano

Departamento de Sistemas
Informáticos y Computación
Universidad Politécnica de Valencia
Camí de Vera, s/n 46022
Valencia, España
pele@dsic.upv.es

Abstract

Information Systems supporting Business Processes can be improved by the use of Ubiquitous Computing (UbiComp) technologies. However, the dynamism of Business Processes and the complexity in the construction of UbiComp systems requires an adequate method for its development. This paper proposes a Model Driven Development method to obtain this kind of systems in a systematic way, abstracting technological details and automating its implementation. The method is presented in depth with a case study of the supply chain of a pharmaceuticals company where UbiComp is used to improve the cold chain and ease the inventory tasks.

1. Introduction

Ubiquitous computing (UbiComp) represents a step forward in the conception of computing systems. Weiser vision [18] presents a paradigm where computing devices are integrated seamlessly into the physical environment and the interaction between the user and the system is performed in a natural way.

Although UbiComp can be applied in different areas [13], this work deals particularly with the enablement of UbiComp as a support for Business Processes (BPs); hence, Ubiquitous Business Processes (UBPs).

The introduction of UbiComp to build this kind of systems has been proven successfully

obtaining valuable benefits after its application. Elements involved in the process that behave as passive objects turn now into active and intelligent elements. This change reduces the gap between the real world and its computational representation thus reducing the media breaks and increasing the quality of information.

The technological evolution in aspects such as automatic identification, connectivity, localization and sensor technologies enable the development of such systems [15]. However, these systems have been implemented completely ad-hoc or using implementation frameworks that support the specific requirements of this kind of systems. Considering that (1) the complexity of the system arises to satisfy UbiComp promises and (2) Business Processes tend to be very dynamic, it is obvious that the quality of the final system becomes difficult to assure.

Model Driven Development (MDD) proposes the use of models as the basis for software development. Models describe, in relevant terms to the problem domain, the aspects of the system to build. This permits a separation of concerns and allows the generation of the final system in an automatic way.

This article presents in a detailed way a method for the development of UBPs following an MDD approach. A case study is presented where the method is applied detailing the models that take part in the process. The rest of the paper is structured as follows. Section 2 presents the context of the work, giving some notions about BPs and UbiComp and some research about these areas in conjunction. Section 3 presents an overview of the method designed for the development of UBPs. Section 4 puts a case study into practice through the application of this method. Finally, section 5 provides some conclusions and further work.

*This work has been developed with the support of MEC under the project DESTINO TIN2004-03534 and cofinanced by FEDER

2. Related Work

Business Processes (BPs) are constantly evolving according to the organization needs, thus adding or changing the requirements of the underlying Information System. Traditional systems have been an obstacle to the evolution of Business Processes as its monolithic conception hinders their adaptation.

Business Process Management (BPM) promotes the modeling, automation, integration, monitorization and optimization of the Business Processes of an organization. To accomplish these objectives different initiatives have emerged.

Business Process Modelling Notation [19] (BPMN) is the OMG standard notation for the modelling of BPs. Despite BPMN being high in its level of abstraction, a mapping to WS-BPEL exists to obtain an executable process definition. WS-BPEL is a XML based language for the orchestration of Web Services Products such as *Oracle BPEL Process Manager*, *Microsoft BizTalk Server* or *Active BPEL Engine* offer execution capabilities for it.

Information Systems supporting BPs are generally based on the desktop metaphor. This fact introduces a gap that maintains the system far from the activities of the organization and that can be filled by Ubicomp.

The reduction in size and costs of computer devices turns Ubiquitous technologies into a reality. New forms of interaction and cooperation among users can be put into practise thanks to auto identification (Auto-ID), localization, sensor technology and the like.

The benefits of applying Ubicomp to support BPs have been studied in economic [8] and process improvement terms [2][12][15]. Ubicomp allows obtaining information very close to the point where it is generated by means of sensors and transferring it automatically to the system avoiding the use of humans as information carriers –which tends to be expensive and error prone–. The rate at which information could be acquired is also increased and big populations can be measured in an individual basis. All this increases information detail and –if managed correctly– quality. For the people involved in the BP, Ubicomp supposes a more natural interaction paradigm. Interaction with the system can happen in a transparent way –the system detects that the

user has performed some task– or an appropriate device could be used to interact with the system – or collaborate among other users–.

Some prototypes have been developed to experiment those benefits in contexts as grocery retail [11], aircraft maintenance [7] and vineyard control [1]. Ubicomp systems are difficult to develop due to its heterogeneous nature in terms of devices and interaction modes. The development of these systems requires a methodological approach to assure the quality in the development. From the Requirements Engineering area some proposals have appeared (Jørgensen and Bossen [5], Kolos-Mazuryk [6] and Sutcliffe [14]). Muñoz in [9] presented a method for the development of pervasive systems based on MDD. However, no method considers the modeling of BPs to construct this kind of systems in a methodological way.

3. Method Overview

In [3] we stated the requirements introduced by UBPs and how these could be fulfilled within a MDD method. Taking into account these results we have defined a method that comprises three groups of models. These groups represent (1) the Business Process, (2) the Ubiquitous System and (3) the Interaction between the users and the system. The method has been elaborated with the idea of reducing the dependencies between these different aspects of the system.

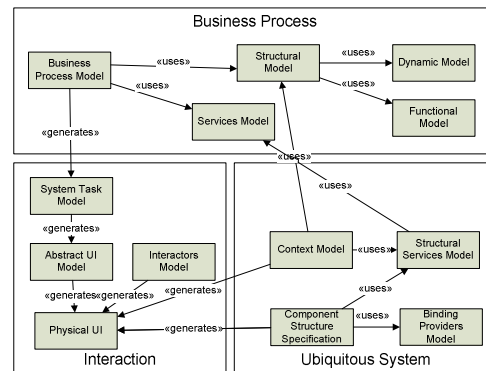


Figure 1 Models involved in the method

Figure 1 depicts the different models involved in the proposal and their dependency relations. Relationships labelled as «uses» indicate that the

source model of the relationship references elements included in the target one. On the other hand, the label «generates» expresses that the target model can be completely or partially obtained from the source one. The three groups depicted in Figure 1 are explained next.

In the first place, models included into the Business Process group define the conceptual model of the process supported by the system and its partners in a technologically agnostic manner. The *Business Process Model* allows us to describe BPs in terms of their activities/tasks and the participants in charge of performing them. Some of these tasks can be performed by functionality provided by external systems (functionality that is defined in the *Services Model*) or by operations from the domain objects involved in the process (defined in the *Structural Model* and which behaviour is specified by means of *Dynamic* and *Functional Models*¹).

Then, regarding the Ubiquitous aspects of the system, these are represented by (1) a set of models (*Structural Services Model*, *Binding Providers Model* and *Component Structure Specification*) in charge of binding the services consumed by the system with the devices that provide those services and by (2) the *Context Model* which characterizes how the information is obtained by the system. This latter model was introduced in [4], while the rest of the models are part of PervML [9].

Finally, the interaction aspect of the system is described following the framework proposed in [16]. This framework defines three more models which deal with user tasks, environment and platform issues. All these models are already considered in our proposed method as follows. In the one hand, user tasks are defined in the Business Process Model. On the other hand, environment and platform issues are described by means of the models related to the Ubiquitous System.

Once it is clear all the models included in the method it is necessary to define the process in which these models are built. The following subsection presents this process and the roles involved in it.

3.1. Development Process

The method involves four roles: Business Analyst, System Analyst, System Architect and Driver Developer. Figure 2 shows the different models involved in the method and the roles which are responsible of their elaboration. The arrows shown in the model represent dependencies between models. The source model needs the target model to be completed in order to be totally defined.

The development process will involve the four roles. First, the Business Analyst defines the Business Process related models. A Business Process Model could be developed first and then describe the objects of the domain and the services used. If the domain is well known, maybe it could be preferred to define the Structural Model first. An iterative approach could be also valid, describing the process and the domain objects as the analyst discovers them in the analysis phase. Dynamic and Functional model define the semantics of operations and will be defined at the end of this phase when the operations to specify are identified.

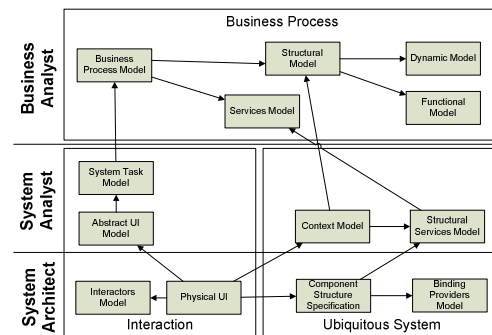


Figure 2 Roles in the development process

The System Analyst task is to decide how information is going to be acquired and presented and how the users are going to interact with the system.

The System Architect defines which devices are used to offer the functionality needed by the system in terms of information acquisition, representation and interaction with the system.

Finally there is a need for a Driver Developer; her mission is to make the software adapters to enable the different devices to interoperate with

¹ Dynamic and Functional models are based in OO-Method [10]

the system. For instance, if the system is implemented using Web Services, a Web Service wrapper should be created for the devices that are not based on this technology.

3.2. Benefits of the method

Systems are not built from scratch in order to use the full potential of Ubicomp. Ubiquitous technology is incorporated in a gradual way instead. The system should not be a burden to the inclusion of technology. A system originally conceived as a traditional Information System, should be easily migrated to incorporate Ubicomp technologies or change its interaction mechanisms. On the other hand, a system supporting UBPs should also be able to change the devices that implement some services with the minimum impact on the system.

The presented method reduces the impact in the system produced by the evolution of requirements and technical aspects. To achieve this, technological aspects are not considered in BPs description and the linkage with technological solutions is specified gradually decoupling the required functionality from its providers.

Considering the modeling of the BPs as the central part of the development of the system involves the centralization of the organization knowledge. Following an MDD approach the

changes in the process can be translated directly into the system automatically, which facilitates the evolution of the system.

The method decouples the conceptual model from the technological aspects related to Ubicomp or interaction. This allows the separation of concerns and helps to determine which models are affected for each kind of change. The method also decouples the services from the devices that implement them, so the reuse is promoted.

4. Case Study

The case study developed in this paper introduces a scenario where a Pharmaceuticals distribution company serves petitions of products made by its clients. The process comprises from the product request reception to the product delivery. Ubicomp has been introduced in several parts of the process to ease the packaging and to control the temperature of the products during the transportation. Moreover, the process includes also manual tasks and others suitable to be improved by ubiquitous technologies.

In this work we focus on the models relative to the Business Process and the Ubiquitous System. The interaction models to define human computer interaction are out of the scope of this paper.

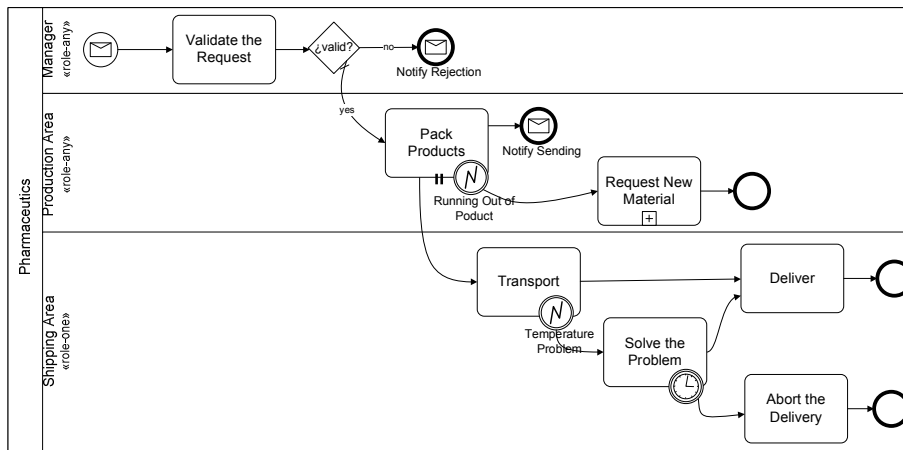


Figure 3 Business Process Model

4.1. The Business Process

The following models express the tasks involved in the process, the objects that take part in it and the services used independently of the ubiquitous nature of the system.

The *Business Process Model* is a BP diagram (BPD) expressed in the BPMN notation. It shows the tasks performed during the process and its temporal order. As it is shown in Figure 3, the process involves several agents (Management, Production and Shipping area) which are represented by lanes. The process is triggered by a request, which is validated by the Management area. If it is considered valid then the requested products are packaged by some member of the Production area. During the packaging, the stock of one type of product can reach its minimum level, in that case the *Requesting New Material* sub-process will be started (the full description of the sub-process is not shown for the shake of simplicity). After the packing, the client will be notified that the transportation phase is going to start and her products will be delivered soon (this allows the client to follow the position of her package through a web system).

During the transportation task the temperature of the products is constantly measured. If some of the products reach an inadequate temperature level (raised above the maximum level specified for that kind of product), there is a limited amount of time to solve and evaluate the situation. Only in case the problem could be solved (or just part of the cargo is affected) it will finally be delivered. In contrast, if the damage is serious it has no sense to keep on with the delivery and the operation should be aborted.

The *Solve the Problem* task has been modeled as a manual task because circumstances can vary too much in this situation to be automated.

The process combines automatic tasks with others which need human intervention. In addition, as explained in [17], we have introduced the *role-one* and *role-any* qualifiers to the roles defined in BPMN. Qualifying a lane with *role-one* means that all the tasks assigned to a certain group should be performed by the same member for an instance of the process. In the case study, the member of the Shipping area who performed the transportation task will be the same that the one that performs the delivery task. The *role-any*

qualifier, in contrast, allows any member of the group to perform each task contained in the lane.

BPMN notation defines eight types of tasks (Service, Receive, Send, User, Script, Manual, Reference y None). In this work only five (arranged in four groups) of those types are considered, (1) manual tasks, (2) user tasks (a human participant interacts with the system), (3) message interchange tasks (Send and Receive) and (4) service tasks (automatic tasks).

In order to confer semantics to the model, tasks of groups 2 and 4 (user and service tasks) will be linked with functionality defined in the Structural and Services Model. For instance, the *Validate the Request* task will be associated with the *validate* operation defined in the Request class, and the event *Temperature Problem* has also its event definition in the Structural Model as it will be explained later.

The *Structural Model* is a UML 2.0 Class diagram that presents the domain object classes showing their attributes and operations. In Figure 4 it is shown that a *Request* is made by a *Client*, and can contain several product requests (indicating the number of units per product type). A *Box* of products will be conscious of its temperature and a *Package* will know its location and will contain several boxes to deliver.

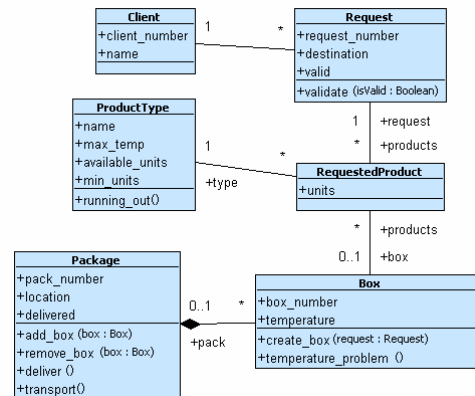


Figure 4 Structural Model

The *Functional Model* defines the semantics of operations. It is based on dynamic logic, a formalism that allows the specification of actions (express events offered or consumed by objects) with a solid formal basis. A formula like $\phi[a]\phi'$ expresses that ϕ should be satisfied if a occurs and

the next state after the occurrence of a will satisfy ϕ' . Being ϕ and ϕ' logical formulas, and a an action. The operation *validate* could be specified as:

```

true
[validate(isValid)]
self.valid = isValid;

```

The event *temperature_problem* arises when the temperature of a box has increased over the maximum temperature supported by any of the products included in it. Its functional specification corresponds to the following formula:

```

self.temperature >
min(self.products.type.maxTemp)
[not temperature_problem()]
false;

```

The previous statement indicates that is not possible (notice the false as ϕ') to reach the problem temperature state without the triggering of *temperature_problem* event.

The *Dynamic Model* expresses by means of a UML 2.0 State Machine, the valid events that can occur in the lifetime of an object. For example, the package object will accept modifications of its content (*addBox* and *removeBox* events) during the packing stage, and not until the transportation phase has begun.

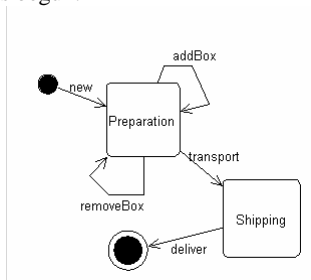


Figure 5 Dynamic Model

The *Service Model* abstracts functionality provided by external systems (like sensors, web services or GPS receivers). By means of a UML 2.0 Class Diagram, as Figure 6 shows, the case study includes services to perform element identification, temperature measurement or object location. Moreover, a service to count the instances of a class (*UnitCounter*) is also included. The *UnitCounter* service will be used to

provide the information needed by the *available_units* attribute of the *ProductType* class.

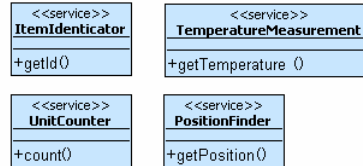


Figure 6 Service Model

Operations from the Structural or the Services Model can be associated with tasks of the Business Process Model. Manual tasks will have no associated operation, while User or Service tasks will.

4.2. The Ubiquitous System

The Ubiquitous nature of the system is specified with the models included in this group. The reason of these models is in the one hand to determine how information will be obtained by means of the Context Model, and on the other hand, how services are related with the devices that offer them.

The *Context Model* refines the Structural Model to classify the associations contained in it (relations between classes and between a class and its attributes). This model is based in [4]. Associations are qualified as static (fixed during the whole lifecycle of an object) or dynamic. Dynamic associations are classified according to source as sensed (obtained through a hardware or software sensor), derived (obtained from existing associations expressed by a formula) or profiled associations (information supplied by users).

Figure 7 includes an excerpt of Context Model related to the *Box* class included in the Structural Model and it also defines the information sources. As this figure shows, the *box_number* attribute will be obtained through a sensor (so an Auto-ID mechanism will be needed). Similarly, the temperature of the box will be also obtained with a sensor. However, the requested product contained in the box or the package in which the box is contained is information introduced by the user.

It's worth mentioning that with this configuration, each time a box is entered or

extracted from a package this action should be informed by the user. In order to automate that, from the modeling point of view the solution would be to mark that association as sensed. This change illustrates the facilities of the method to support the evolution of the system. As a consequence, box identification would be enforced to be sensed and a receiver able to detect a box entering and exiting a package should be added to each package to automate *addBox* and *removeBox* operations.

In addition to the previous classification, some structural constraints can be added to note the cardinality of associations (simple or composite). Composite associations can be a collection, alternative or temporal association. Collections permit the association with multiple objects at the same time, alternative associations represents that only one of the possible associations is required, temporal associations ensures that only one of the possible associations exist at a time.

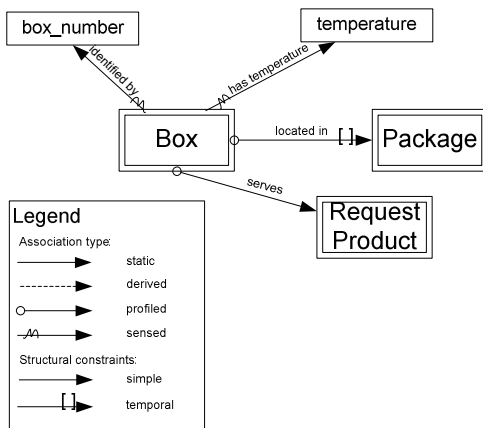


Figure 7 Context Model

The information quality could be affected by the acquisition mode. The Context Model also allows expressing context quality, defining some quality parameters and associating different quality metrics. In the example, Accuracy (measured by the standard error of the temperature measurement system) or Freshness (determined by the time the temperature information is obtained and the refresh rate of the measure) could be considered as two parameters of quality.

The Context Model will also indicate which service realizations (defined in the Structural

Services Model) are used to obtain the information in case of dynamic associations.

The rest of the models presented try to decouple the services used by the system from the devices that implement them using several layers of abstraction. These models are part of PervML [9].

The *Structural Services Model* defines concrete realizations of the services defined in the Services Model. It's expressed by means of a UML 2.0 Component Diagram as shown in Figure 8. In this model different instances of the services can be defined, all of them providing the same service, but in different contexts. For example, the service *PositionService* has two realizations: *PackagePosition* (to allow the tracking of a package) and *DestinationPosition* (to indicate where a package should be delivered). The service idea is the same (obtaining the coordinates of an object) but the context are different because the package is in the organization control and the destination not. And a positioning chip can be attached to the package, but not to the recipient location.

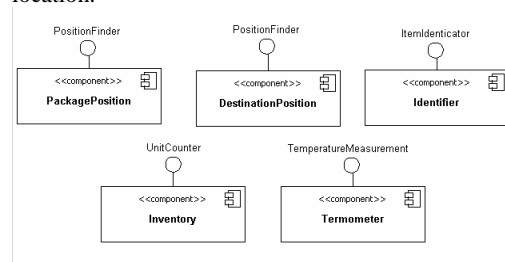


Figure 8 Structural Services Model

The *Binding Providers Model* represents the providers of the services. These providers can be any kind of devices or even software services offered by third parties. The example shown in Figure 9 includes a barcode reader. These components are highly reusable and maintaining a repository is a must.

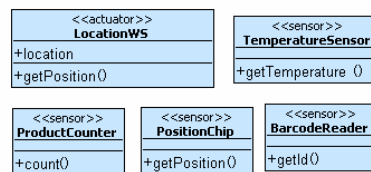


Figure 9 Binding Providers Model

The *Component Structure Specification Model* serves to specify which binding provider is used to implement a service realization. For example, the Identifier service realization will be implemented by the barcode reader. The *PositionFinder* service will have two implementations: the *PackagePosition* realization will use a *PositionChip* (using GPS information, for example) whereas the *DestinationPosition* realization will be implemented using a Web Service. The *LocationWS* gives the geographical coordinates but needs the postal address of the element to locate. So this attribute should be mapped for the clients of this service (indicating the destination attribute of Request object has the location function).

5. Implementation details

To validate the proposed method, a prototype of the presented case study has been developed. The MDD approach followed by the method is supported in the prototype by the Eclipse modeling tools.

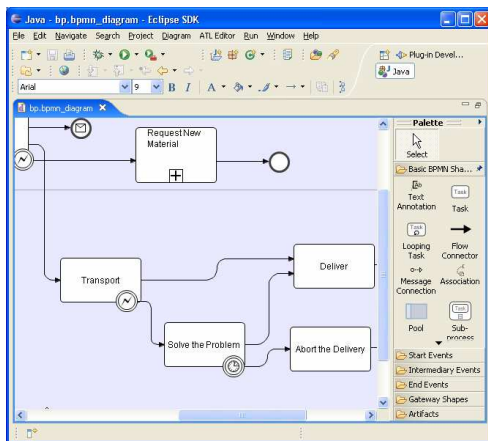


Figure 10 BPMN Editor

The *Eclipse Modeling Framework*² (EMF) is the basis of several modeling projects developed by the Eclipse community. EMF includes tools for the generation, edition and serialization of models conforming to Ecore metamodels (an implementation of the OMG's Essential MOF to

² <http://www.eclipse.org/modeling/emf/>

represent metamodels). the *Atlas Transformation Language*³ (ATL) was used to define transformations between models.

The Business Process Model was based on the SOA Tools Platform BPMN metamodel (Figure 10 provides a snapshot of the graphical editor). The mappings between BPMN and WS-BPEL have been defined extending and integrating the BABEL BPMN2BPEL tool into the prototype. The resulting WS-BPEL definition was executed using the ActiveBPEL⁴ Engine. Finally, to support human tasks we have implemented an extra module (the task manager module) to maintain user to-do lists.

All models based in UML 2.0 Diagrams were defined using the Topcased⁵ editor which offers graphical edition for the Ecore metamodel defined by the Eclipse UML2 Project. However, to define the Context Model, an Ecore metamodel was created.

The Structural, Dynamic and Functional Models were developed using ONME⁶, as its support of OO-Method (in which this models are based) offers full code generation of structure and behaviour for the application.

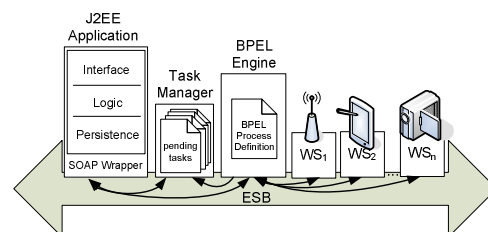


Figure 11 System architecture

Devices and external services were implemented by means of Web Services that simulate them. Thus adapters should be necessary to connect the real elements.

To integrate ONME Application, Process Execution Engine, Task Manager and the different web services that represented devices, Mule⁷, an open source Enterprise Service Buss, was used. The different components of the architecture are

³ <http://www.eclipse.org/m2m/atl/>

⁴ <http://www.activebpel.org>

⁵ <http://www.topcased.org>

⁶ <http://www.care-t.com>

⁷ <http://mule.codehaus.org>

depicted in Figure 11. Although a Web Services based implementation was chosen, this approach allows different kind of services to be used.

Thanks to the code generation capabilities of ONME, only WS-BPEL definition, WSDL and Mule configuration files were needed to be generated.

The separation of concerns proposed by the method eases the code generation strategy by reducing the number of models needed to generate each artifact. On one hand, models describing the BP are the source of the executable definition of the process (WS-BPEL definition and its related service description using WSDL). On the other hand, Mule configuration files are generated from the models that deal with Ubiquitous aspects of the system. Following this approach, a change in the UbiComp related models only supposes the generation of UbiComp related artifacts affecting in no way the ones derived from the Business Process Definition models.

Regarding user interaction, UbiComp systems require complex interaction mechanisms (aspect that is out of the scope of this paper). However, a web based user interfaces could be generated following the approach introduced in [17].

6. Conclusions

Ubiquitous systems require a systematic approach in their development, especially when they are built to support BPs in an organization. In this work a method for the development of UbiComp systems to support BPs has been presented. Based in solid research in the areas of Business Process Management, UbiComp and Human Computer Interaction, the method combines them to provide an MDD method. The method introduces several abstraction levels to decouple services from the elements that offer them. This separation of concerns helps to cope with the evolution and interoperability of systems.

In this paper we have not taken into account interaction issues. As a future work a detailed description of interaction modes supported by the method should be introduced.

The extension of ONME tool to fully support the method will allow an automatic code generation facility which constitutes the final goal of this research.

References

- [1] T. Brooke and J. Burrell, "From ethnography to design in a vineyard", in Proceedings of the Design User Experiences (DUX) Conference, June 2003, case Study.
- [2] Fleisch, E. Business Perspectives on Ubiquitous Computing. M-Lab Working Paper No. 4. University of St. Gallen, Switzerland, 2001.
- [3] Pau Giner, Victoria Torres: Una Propuesta Basada en Modelos para la Construcción de Sistemas Ubicuos que den Soporte a Procesos de Negocio. In: IDEAS'07: X Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software.
- [4] Henricksen, K., Indulska, J., and Rakotonirainy, A. Modeling context information in pervasive computing systems. In LNCS 2414: Proceedings of 1st International Conference on Pervasive Computing (Zurich, Switzerland, 2002), F. Mattern and M. Naghshineh, Eds., Lecture Notes in Computer Science (LNCS), Springer, pp. 167–180.
- [5] J. B. Jorgensen and C. Bossen. Requirements Engineering for a Pervasive Health Care System. In 11th IEEE International Requirements Engineering Conference (RE'03), pages 55–64. Springer Verlag, 2003.
- [6] L. Kolos-Mazuryk. Development of a requirements engineering method for pervasive services. In RE'05 Doctoral Consortium, 2005.
- [7] Lampe, M., Strassner, M., Fleisch, E.: A ubiquitous computing environment for aircraft maintenance. In: ACM Symposium on Applied Computing, Nicosia, Cypress (2004).
- [8] M. Langheinrich, V. Coroama, J. Bohn, M. Rohs. As we may live - Real world implications of ubiquitous computing. Technical Report, Institute of Information Systems, Swiss Federal Institute of Technology, Zurich, Switzerland, 2002. <http://www.inf.ethz.ch/vs/publ/papers/ucimplifications.pdf> (13 July 2003).
- [9] J. Muñoz and V. Pelechano. Building a Software Factory for Pervasive Systems Development. In CAISE 2005, Porto, Portugal, June 13-17, volume 3520 of LNCS, pages 329–343, May 2005.

- [10] Pastor, O., Gomez, J., Insfran, E., Pelechano, V. The OO-Method Approach for Information Systems Modelling: From Object-Oriented Conceptual Modeling to Automated Programming. *Information Systems* 26, pp 507–534 (2001).
- [11] Roussos, G., L. Koukara, P. Kourouthanasis, J. Touminen, O. Sappala and J. Frissaer (2002). "A Case Study in Pervasive Retail." *ACM*: 90-94.
- [12] Uwe Sandner, Jan Marco Leimeister, Helmut Krcmar. Business Potentials of Ubiquitous Computing. In: *Proceedings of the Falk Symposium No. 146 bGut-Liver Interactions: Basic and Clinical Concepts*. Innsbruck, Austria, 2005.
- [13] Spinola, Rodrigo, Massolar, Jonson, Travassos, G. H. Towards a Conceptual Framework to Classify Ubiquitous Software Projects. In: *SEKE 2006 - International Conference on Software Engineering and Knowledge Engineering, 2006, San Francisco. Proceedings of the SEKE 2006, 2006.*
- [14] Sutcliffe, S. Fickas, and M. M. Sohlberg. Personal and contextual requirements engineering. In *13th IEEE International Conference on Requirements Engineering (RE'05)*, pages 19–30, Paris, 2005.
- [15] Strassner, M., Schoch, T. (2002): Today's Impact of Ubiquitous Computing on Business Processes. In: Mattern, F. Naghshineh, M. (Eds.): *Pervasive Computing. First Int. Conf. Pervasive Computing 2002*, Zurich, Switzerland.
- [16] D. Thevenin and J. Coutaz, "Plasticity of User Interfaces: Framework and Research Agenda", *Proc. of INTERACT'99 (Edinburgh, August 1999)*, IOS Press, 1999.
- [17] Torres, V., Pelechano, V., Giner, P.: *Generación de Aplicaciones Web basadas en Procesos de Negocio mediante Transformación de Modelos*. In: Riquelme, J.C., Botella, P. (eds.): *Ingeniería del Software y Bases de Datos. JISBD 2006*. 443-452
- [18] M. Weiser. The Computer for the 21st Century. *Scientific American*, 265(3):94–104, Sept. 1991.
- [19] S. A. White. *Business Process Modeling Notation (BPMN) Version 1.0*. Business Process Management Initiative, BPMI.org, May 2004.