Later published as: Fernandes, J. E., R. J. Machado, João A. Carvalho, A Case Studies Approach to the Analysis of Profiling and Framing Structures for Pervasive Information Systems, International Journal of Web Portals 4(2), 1-18, April-June 2012

A Case Studies Approach to the Analysis of Profiling and Framing Structures for Pervasive Information Systems

José Eduardo Fernandes[1], Ricardo J. Machado[2], João Á. Carvalho[2]

[1] Polytechnic Institute of Bragança, School of Technology and Management, Dept. of Informatics and Communications
5301-854 Bragança, Portugal
jef@ipb.pt

[2] Universidade do Minho, Escola de Engenharia, Centro ALGORITMI
4800-058 Guimarães, Portugal
{rmac, jac}@dsi.uminho.pt

Abstract. Model-Based/Driven Development (MDD) constitutes an approach to software design and development that potentially contributes to: concepts closer to domain and reduction of semantic gaps; automation and less sensitivity to technological changes; capture of expert knowledge and reuse. The widespread adoption of pervasive technologies as basis for new systems and applications, lead to the need of effectively design pervasive information systems that properly fulfil the goals they were designed for. This paper presents a profiling and framing structure approach for the development of Pervasive Information Systems (PIS). This profiling and framing structure allows the organization of the functionality that can be assigned to computational devices in a system and of the corresponding development structures and models, being. The proposed approach enables a structural approach to PIS development. The paper also presents two case studies that allowed demonstrating the applicability of the approach.

Keywords: MDD, PIS, pervasive, ubiquitous, software engineering, process, information systems, architecture, framework.

Introduction

The dissemination of computing and heterogeneous devices and platforms, the high pace of technological innovations and volatile requirements, the size and complexity of software systems characterize the software development context today. This context challenges the way software is developed for emerging forms of information systems. Software Development Processes (SDPs), as well as generalized adoption of models, are fundamental to efficient development efforts of successful software systems.

Pervasive Computing, also called Ubiquitous Computing (Weiser, 1993b; Weiser, Gold, & Brown, 1999), represents a new direction on the thinking about the integration and use of computers in people's lives. It aims to achieve a new computing paradigm, one in which there is a high degree of pervasiveness and availability of interconnected computing devices in the physical environment. Widespread availability of affordable and innovative information technologies represents a potential opportunity for improvement/innovation on business processes or for enhancement of life quality of individuals. Among other things (such as social concerns), this opportunity promotes the attention to the efficiency and effectiveness of information management regarding to the way they acquire, process, store, retrieve, communicate, use, and share information. To take full benefits of the opportunities offered by modern information technologies, these devices need to be "appropriately integrated within organizational frameworks" (Sage & Rouse, 1999). Therefore, Pervasive Information Systems (PIS) (Fernandes, Machado, & Carvalho, 2008) orchestrate these devices in order to achieve a set of well-established goals. In this way, PIS not only provide a solid basis to sustain the needed information to achieve effectiveness at both individual and organizational levels, but also leverages the investment on those information technologies or other organizational resources. In order to explore the potential offered by pervasive computing and to maximize the revenue of these kinds of systems, a PIS, as any other

information system, must be designed, developed and deployed attending to its nature (these systems may potentially accommodate a large quantity of heterogeneous devices and be subject of frequent updates/evolutions).

Software engineering has been, since its existence, subject of research and improvement in several areas of interest, such as software development processes (SDPs) whose process models evolved from waterfall and nowadays may assume several forms (Ruparelia, 2010). The development of large software systems is another area of interest that has been, for decades, subject of research work; several topics can be pointed out such as the exploration of issues related to the management of large scale software development (Benincasa, Daneels, Heymans, & Serre, 1985; Kay, 1969), software architecture (Gorton & Liu, 2010; Laine, 2001; Mirakhorli, Sharifloo, & Shams, 2008), model-driven development (Heijstek & Chaudron, 2009; Mattsson, Lundell, Lings, & Fitzgerald, 2007), among others. Not directly related with large projects, Medvidovic (2005) points the relevance of software architecture in leveraging the pervasive and ubiquitous area. Model-Based/Driven Development (hereafter in this document, unless otherwise stated, simply referred as MDD) is another area that gains an increasing focus. MDD constitutes an approach to software design and development that strongly focuses and relies on models (Fernandes, Machado, & Carvalho, 2004). It automates, as much as possible, the transformation of models and the generation of the final code. This enables higher independence from the technological platform that supports the realization of the system.

This paper, further exploring the topic of software development for PIS, proposes an approach for profiling and framing functional profiles for PIS development, and presents a case study used for its applicability. This document structures its content as follows: section 1 introduces pervasive information systems, its issues and the benefits of a model-based/driven development based approach; section 2 gives insight into related research works and gives an overview of a development framework for PIS; section 3 presents the suggested approach; section 4 presents a case study wherein this approach is demonstrated; section 5 presents the conclusions and finishes this document.

Pervasive Information Systems

Ubiquitous (computing embodies a philosophy different of that inherent to the personal computers of the 70s. In essence, it sustains that computing technology should not be the focus of attention of the user activity. It even does not require the need of carrying around any personal computer or PDA to access information; in this world, fully of connected devices, information is available and accessible everywhere (Weiser, 1993a). The data, once entered in a computing system, is readily available whenever and wherever needed (Ark & Selker, 1999), being accessible in an intuitive way through the use of devices eventually different from that one through which the data was entered.

Decreasing emphasis of focus on the personal computer has already occurred with the emergence of the World Wide Web. For many users the computer is just a machine that provides a portal to the digital world where they have presence through their homepage, their email, or chat. In this way, computers are 'disappearing' and the focus goes beyond them (Davies & Gellersen, 2002). Ubiquitous computing brings then "the end of dominance of the traditional computing" (Ark & Selker, 1999), being computing embedded in more things than just our personal computer.

Considering the vision about ubiquitous computing, there are key characteristics of ubiquitous computing systems that differentiate these from traditional computing systems. Among these are: decentralization (autonomous small devices, taking over specific tasks and functionality, cooperate and establish a "dynamic network of relationships"), diversification (there is a move from universal computers to diversified devices for specific purposes), connectivity (different type of devices connect among themselves to exchange data and applications) and simplicity (pervasive devices, being specialized tools, should be easy and intuitive to use − "complex technology is hidden behind a friendly user-interface") (Hansmann, Merck, Nicklous, & Stober, 2003).

In ubiquitous computing, the environment take a relevant place in computing: in "contrast with most traditional computing, in which the environment is mostly irrelevant, the environment plays a fundamental role for ubiquitous computing; the environment has influence on the 'semantics' of computing" (Ciarletta & Dima, 2000). There is a need of perceptual information about the environment (Saha & Mukherjee,

2003) and about the location of people and devices: such information enables for an enhanced interaction with users, allowing applications to adapt themselves to their environment, and constitutes an enabler element for the so-called invisible computing.

Beyond the traditional media, the web has emerged as a new fundamental and valuable global information system, being widely adopted not only by organizations but also by people. Today, the web is easily accessible in all developed countries, in schools, in private and public organizations, at home, and inside or outside buildings. Also notable has been the widespread adoption of cellular phones that, along with increasing computing resources, have acquired improved communication capabilities and new multimedia features. They allowed a new and quick way to contact and interchange information with people, to access to the World Wide Web everywhere, and to interconnect computing devices all around the world (even in the most inhospitable places).

The advent of accessible commercial wireless networks and communications systems further contributed to dissemination of computing. The embedding of computing devices in objects or places for monitoring or control, enabled us to envision a "real" physical world enhanced with information and computing capabilities. These capabilities can be used to facilitate and pleasure human life in its diverse facets (as the personal or social) or to improve businesses or other organizational processes. Want, Pering, Borriello and Farkas (2002) consider that the "four most notable improvements in hardware technology" during the last decade that directly affected ubiquitous computing are: wireless networking, processing capability, storage capability, and high quality displays.

These factors, among others, contributed for a culture characterized not only by having an easy access to information, but also by demanding for information availability; consequently there is an implicit acceptance of surrounding and permanent computing or other IT devices. Nowadays, there is an increasing feeling that information is omnipresent (we just need an IT device to access it) and that computing devices or applications are naturally part of our daily lives.

From a business perspective, ubiquitous computing brings the opportunity to introduce changes in the way business and consumers interact with each other (Fano & Gershman, 2002). It allows for an improvement on mutual intercommunications, richer and innovative

interactions, and closer relationships. People become able to interact with services not only through telephone or PC but also through products.

What was initially confined in developing technology to make pervasive computing out of a vision (Lyytinen & Yoo, 2002), surpassed the initial restricted frontiers to reach the development of applications for organizational domains, enabling for enhancements of current business processes or even to assist the development of new business models (Langheinrich, Coroama, Bohn, & Rohs, 2002).

Business benefits and ubiquitous computing technologies have a mutual influence in each other: ubiquitous computing technologies are seen has offering support for potential business benefits to organization efficiency, and those potential benefits constitute a driving force and key factors to further research and deployment of ubiquitous computing technologies (Bohn, Coroamã, Langheinrich, Mattern, & Rohs, 2004); this leads to a permanent, vigorous, and rapid proliferation of information technology. Aware of those business benefits potentially offered by ubiquitous computing technologies, the industry has set their attention to the deployment of those technologies in supporting applications in diverse domains, pursuing imagined business benefits. Government agencies, insurance companies, organizations of several domains have been developing projects aiming to collect the potential gains of deployment of ubiquitous computing.

A world full of smart devices and the widespread adoption of pervasive technologies as basis for new systems and applications, lead to the need of effectively design information systems that properly fulfil the goals they were designed for. These pervasive information systems and the applications that constitute them need to be able to accommodate the permanent technological evolutions/innovations of the heterogeneous devices and the requirements changes that result from a faster and intense world of business competition.

Model-Driven Development

Albeit some opinions consider that there is no “universally accepted definition of MDD is and what support for it entails” (Atkinson & Kuhne, 2003), it can be said that MDD carries the notion that it can be possible to build, with modelling languages, a model that entirely

represents the intended software system. This model can then be transformed, through well-defined transformation rules, into the "real thing" (Mellor, Clark, & Futagami, 2003). Nonetheless, it's noteworthy to point out that, to achieve or undertake model-driven development, "not all models need to be executable or even formal, but those that are can benefit from automation" (Mellor et al., 2003) and models do not need to be complete, as "it incompleteness or high degree of abstraction do not equate to imprecision" (Mellor et al., 2003).

Since antiquity engineering disciplines have the activity of modelling as a fundamental technique to cope with complexity ("The use of engineering models is almost as old as engineering itself." (Selic, 2003)). Modelling provides a way to facilitate the understanding, reasoning, construction, simulation, and communication about complex systems (usually composed by smaller parts) (Thomas, 2004). Software engineering, in comparison with other forms of engineering, is on a privileged position to attain benefits from modelling, as it is one whereby an "abstract high-level model can be gradually evolved into the final product without requiring a change in skills, methods, concepts, or tools" (Selic, 2003).

There have been, and there will always be, several efforts in order either to improve the way and the cost of development of sosftware systems, or to achieve a better satisfaction on accomplishment of systems requirements and expectations. One area of these efforts of improvement is on raising the abstraction at which software developers mainly work.

Several examples of such rising of abstraction are the movements from binary languages to assembly languages and from assembly languages to higher-level languages. The new abstractions, initially introduced as novel concepts, were later adopted and supported, and tools were developed "to map from one layer to the next automatically" (Miller et al., 2004). Nowadays, there is a promotion of another rising of abstraction at which development occurs: this one is based on changing of the main development efforts from code and programming to models and modelling. This raise of abstraction at which software is written (the shift of the level of abstraction from code and programming languages, to models and model languages (Sendall & Kozaczynski, 2003)) implies that a software system will be mainly and fully (as possible) expressed by models. The models are the main

artefacts of the development effort rather than computer programs (Selic, 2003). The raise of abstraction subjacent to the use of models allows for productivity improvement: “it’s cheaper to write one line of Java than write 10 lines of assembly language. Similarly, (…) it’s cheaper to build a graphical model in UML, say, than to write in Java” (Mellor et al., 2003).

Synthetically, models, in a descriptive or a prescriptive form, can then be used to: (i) understand or communicate a problem, a existing system, or a proposed solution; (ii) analyse, or predict on changes, systems properties or risk failures; (iii) productivity improvement; (iv) reduction of system’s development costs.

As models are the primary artefact in model-driven development approach, it is necessary that “a clear, common understanding of the semantics of our modelling languages is at least as important as a clear, common understanding of the semantics of our programming languages.” (Seidewitz, 2003). The Unified Modelling Language (UML) specifies the primary notation used in the current practice of modelling. UML allows for the creation of models that capture different perspectives of the system.

Regarding to the development of software systems, the Object Management Group (OMG, 2005) introduced in 2001 the Model Driven Architecture (MDA) (OMG, 2003), an open and vendor neutral architectural framework to the construction of software systems. MDA constitutes a software development approach that, through the focus on models and defined standards, separates the specification of the functionality of a system from the specification of its implementation on target technological platforms, providing a set of guidelines framing these specifications (Appukuttan, Clark, Reddy, Tratt, & Venkatesh, 2003). It enables the detachment of business-oriented decisions from technological issues of eventual specific platforms into which the system could be targeted, allowing for “a greater flexibility on the evolution of the system” (Brown, 2004). Model-driven architecture is considered a “model-driven” approach in the sense “code is (semi-) automatically generated from more abstract models, and which employs standard specification languages for describing those models and the transformations between them.” (Brown, 2004).

MDD has the potential to offer key pathways that enable software developers to cope with complexity inherent to PIS. A proper PIS

construction demands an approach that recognizes particularities of PIS and that benefit from MDD orientation.

Research has been performed (Fernandes, Machado, & Carvalho, 2007) to bring the application of MDD concepts and techniques to software of PIS. Fernandes et al. (2008) suggest a conceptual development framework able to sustain an approach for software development of PIS that take into account MDD potential and PIS characteristics, particularly, heterogeneity and functional variability. The following paragraphs present a brief overview of this development framework

The *development framework* (Fernandes et al., 2008) for PIS introduces and describes new conceptions framed on three perspectives of relevance to the development, called *dimensions*. Based in these dimensions, the development framework considers two additional main perspectives of development: one concerning the overall development process, and a second concerning to individual development processes. Fig. 1 illustrates a schema of the framework. The following paragraphs give an overview of these dimensions and development perspectives.

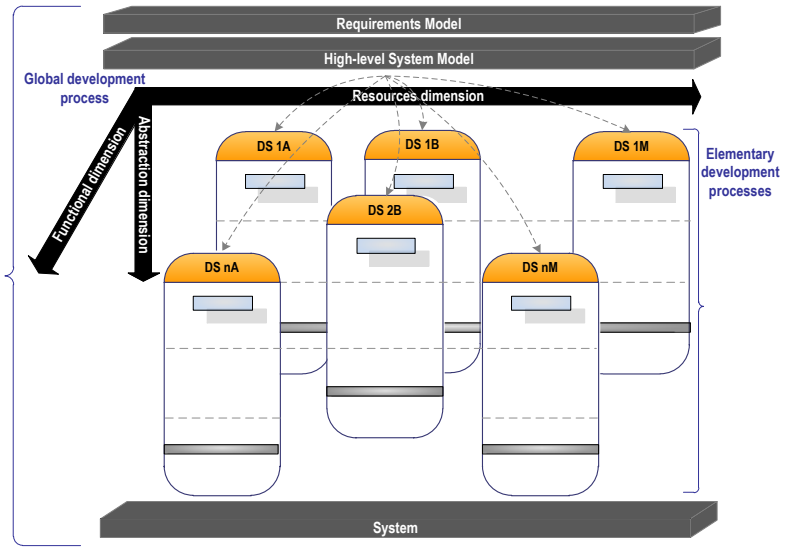

Fig. 1. Development framework for PIS.

The three dimensions considered are: resources, functional, abstraction. The *resources dimension* sets up the several categories of devices with similar characteristics and capabilities. The *functional dimension* sets up the different functionality needed by the system and

that can be assigned to resources in the system for its concretization. The assignment of a specific functional profile to a specific resource category results in a specific *functional profile instance* that is realized by devices in that resource category. Each functional profile instance has a corresponding *development structure* which embodies an elementary development process aiming to realize that instance. The *abstraction dimension* respects, in an MDD context, to the levels of abstraction that elementary development process may have (from platform-independent model (PIM), passing by platform-specific model (PSM), to generated code). The development framework structures the development in a global development process and several elementary development processes. The *global development process* is responsible for modeling requirements and for establishing high-level and global system models. Based on these models, it sets up functional profiles and categories of resources, as well as, high-level PIM for each functional profile instance that shall exist. The global development process has the responsibility for making all the necessary arrangements for integration of the several artifacts that result from elementary development processes and for final composition, testing, and deployment of the system. *Elementary development processes* are responsible for the software development of parts of the system that realize specific functionalities for specific categories of resources. For each of the development structures, an adequate software development process can be chosen, as long as it respects the principles of the approach globally adopted. MDD concepts and techniques may be applied in order to improve the development and the quality of those resulting parts of the system.

The implicit strategy to this development framework enables the adoption of development process and techniques most suitable to development of that individual development structure. It also eases the assignment of those structure units to different collaborating teams and, eventually, the outsourcing of the development.

Besides the traditional documentation, the development approach should provide documentation for each development structure. Among this documentation, it is expected to be found information about the platform independent models (PIMs) at the top model-level, the PSMs at the intermediate model-level, the PSM at the bottom model-level, the mappings (either vertical or horizontal) and inherent transformation techniques used on the model's transformations, as well as information

regarding to code generation. It becomes clear that it is convenient the use of suitable CASE tools to support global and individual development process developments as herein proposed. It is also expected the use of well-established standards on languages and techniques for modelling (models and transformations models), support for code generation, change management, and documentation of all artefacts and design decisions.

The global process and the elementary process are not prescribed to be performed by any particular existent development process, being the choice of process development left to the developer.

In (Booch et al., 2007), the concepts of "macro process" and "micro process" are used in the framework proposed for the software development process. They represent perspectives of the overall software development cycle (the macro process) and of the analysis and design process (the micro process). Whilst the macro process aims to guide the overall development of the system and its scope is "from the identification of an idea to the first version of the software system that implements that idea" (Booch et al., 2007), the micro process cover the analysis and design activities. Activities of analysis focus on behaviour and not on form, and produce an initial solution from system requirements. In the develomnet framework for PIS, the global process can be see can be seen as being similar to macro process, as it respect to the overall development process, and feeds the elementary processes as the macro also does for the micro process. The elementary process is somehow different from the micro process as it has a distinct scope: it respects to a whole development structure, which can be seen by itself as a system for which it can be applied a development process that can inclusively include the strategy of development associated with the macro and micro concepts presented.

Profiling and Framing Structures

In the context of the previously presented development framework, this section aims to provide a way to effectively and consistently apply it in PIS development projects, independently of its size. The section starts by taking some considerations regarding functional profile instantiation, modeling levels in development structures; then it

illustrates the concept of framing structure, giving emphasis on the way of using it in the context of large projects.

The assignment of a functional profile to a resource corresponds to an instantiation of the functional profile, carrying the meaning of responsibility assignment to that resource. Fig. 2 illustrates an example of instances resulting from the assignment of functional profiles to resource categories.

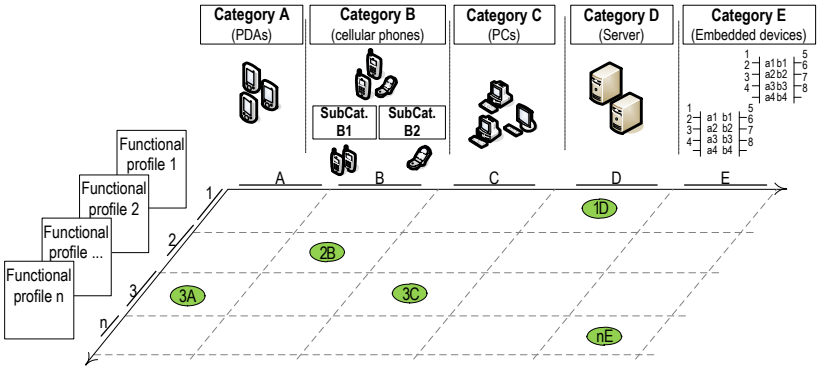

Fig. 2. Functional profile instances.

The result of an instantiation process is an instance profile that has subjacent a kind of platform independent model (or depending of the perspective, it may be seen as a PSM) as it is expected to be later subject of possible model transformations into intermediate platform specific models (or eventually directly subject to code generation). Further development takes place based on this model, giving origin to a specific development structure related to that specific functional profile instance. Each development structure reflects a pathway of software development in order to realize a functional profile assigned to a category of resources. Fig. 3 illustrates these development structures as well, as the modeling levels that can be found inside them. These modeling levels respects to the abstraction dimension, one of the three dimensions previously exposed. Depending from the point of view, an intermediate model can be seen as a PIM or a PSM: a model can be seen as a PSM when looking from a preceding higher abstraction model level, and can be seen as a PIM when looking from lower abstraction model level. For some development structures these levels may eventually not exist, as it is possible to directly generate the bottom-level PSM or even the code itself.

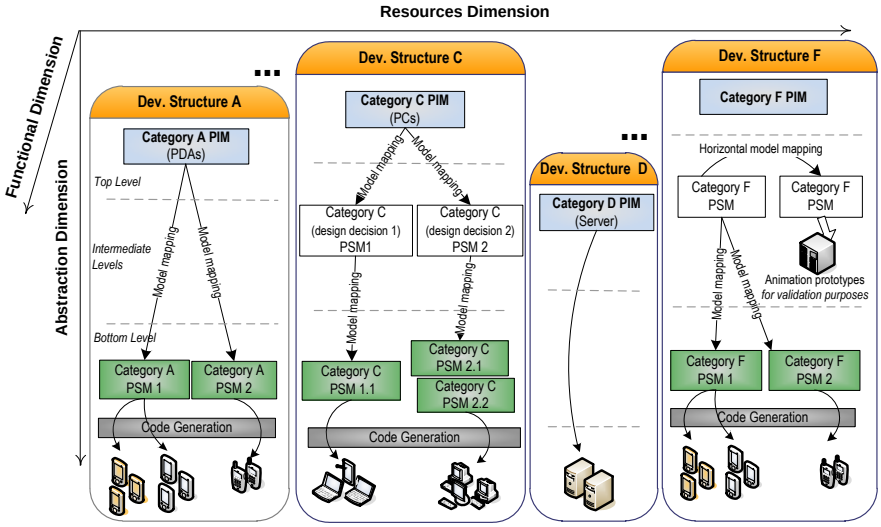

Fig. 3. Modeling levels in development structures (abstraction dimension).

Considering the schema of the development framework and the schemas related to functional profiles instantiation, an overall conceptual representation of conceptions involved in the development framework can be schematized into a conceptual framing structure that allows the definition and framing of functional profile instances. This conceptual structure can be expressed by a schema similar to the one presented in Fig. 4.

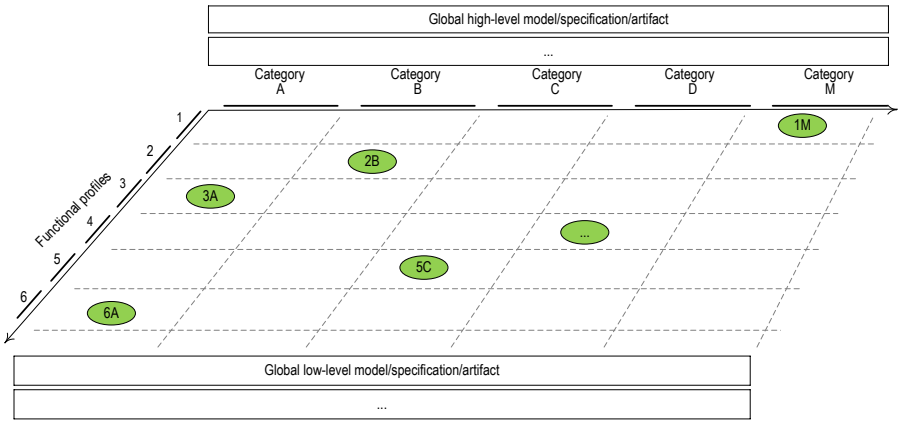

Fig. 4. Framing structure for a project.

Fig. 4 illustrates the high-level and low-level models/specifications/artifacts produced by starting and ending activities of the global development process (it is important to notice that in parallel with the elementary development process activities, there may be in course other global development processes activities). All relevant functional profiles are listed at the left side of the framing structure, and the resources categories identified are listed at the middle top. The definition of functional profile instances are signaled in the proper intersections of lines of functional profile with the columns of resource categories. For each functional profile instance there is an associated development framework (as depicted in Fig. 3); for each of these development frameworks there will be a corresponding elementary development process (as depicted by Fig. 1).

Considering that systems vary in size and complexity, there may be large projects of systems involving the definition of large subsystems, for which there is the interest to define their own functional profiles and resources categories. For such cases, the framing structure has an extended way of use. A framing structure is defined for the system and, for each of the identified subsystems, there is an additional framing structure; this will bring to existence nested framing structures.

The system framing structure will contain elements (functional and resources) with a system level granularity, while each of the subsystem framing structures will have its own suitable subsystem level granularity. This situation may be recursive and a subsystem may be composed by its own subsystems; is this case, for each of the subsystems, there will be again a corresponding framing structure that, at a certain point, will be a leaf framing structure containing final functional profiles and resource categories.

The recursive nesting of framing structures allows dealing with any system size. In this process, each of the framing structures implicitly defines its own namespace for naming its constituent elements. Fig. 5 shows an example of the nesting of the framing structures to deal with the size of large projects.

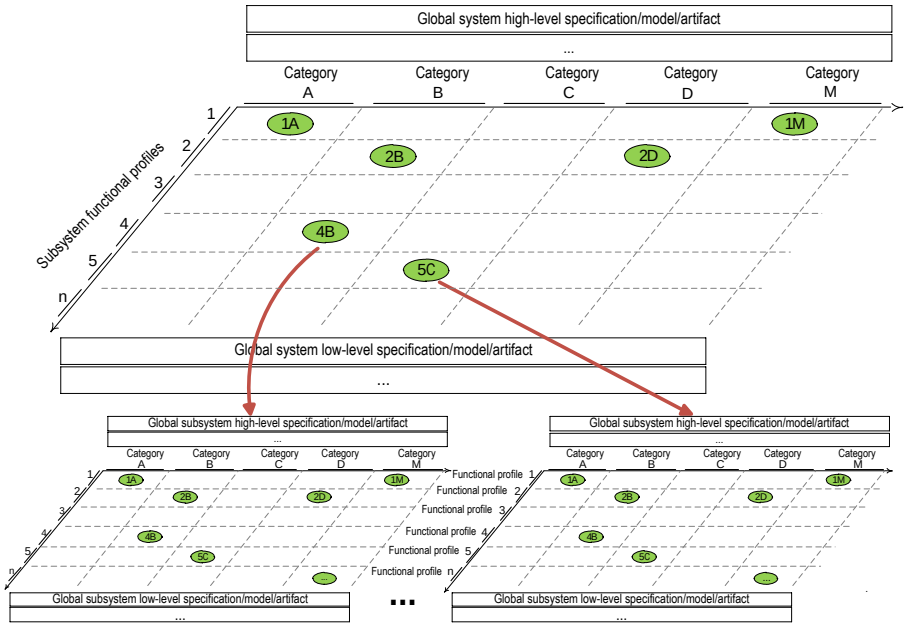

Fig. 5. Nesting of framing structures for large projects.

Case Studies

This section starts by briefly introducing the USE-ME.GOV (USability-drivEn open platform for MobilE GOVernment), a project that aimed to create an open platform for mobile government services, and the uPAiN (Ubiquitous Solutions for Pain Monitoring and Control in Post-Surgery Patients) project, conceived to create a an information system for anaesthesiology services of healthcare centres. Then, it illustrates the application of the development framework on these projects. Attending to the project dimensions, only a part of the model (where appropriate) will be used for illustration purposes (this does not affect the rationale to be taken for the whole model). This section ends by exposing some issues pertinent to a proper project definition for PIS.

The USE-ME-GOV Case Study

The USE-ME.GOV project (USE-ME.GOV, 2003) focused on the development of an open platform for mobile government services. This

platform facilitates the access of authorities to the mobile market by allowing them to share common modules of the platform and to deal with multiple mobiles operators independently of each one's interface. USE-ME.GOV system general architecture is illustrated by Fig. 6.

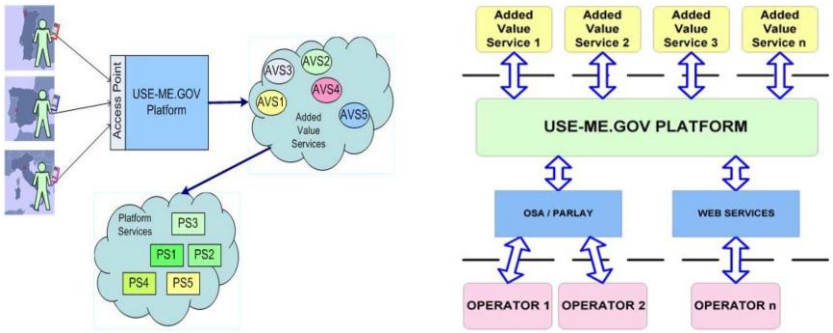

Fig. 6. USE-ME.GOV System General Architecture (from (USE-ME.GOV, 2006)).

The USE-ME.GOV Platform basically consists of two separate application system: (i) Core Platform, which is responsible for user's platform access, user and terminal management; (ii) Service Repository, which is a central registry of services. The USE-ME.GOV system also contains what is designated by "platform services". Platform services included in the USE-ME.GOV system are: (i) Context Provision and Aggregation Services; (ii) Localization Service; (iii) Content Provision and Aggregation Service. These services enable the use of user's context, user's localization, and access and aggregation of data form external sources.

The USE-ME.GOV project is extensive and includes several subsystems services. In the light of the approach proposed, these subsystems can be seen as a system for which a whole development process can be applied. As such, the project will have a contextual system framing structure identifying the major subsystem's functional profiles and subsystem's resource category groupings. Then, for each of the subsystem functional profile instances (the crossing of subsystem's functional profile with subsystems' resources category grouping) is developed a new framing structure, at a subsystem level. In this framing structure the high-level model corresponds to the one regarding to the specific subsystem's functional profile instance in the preceding framing structure. In each subsystem's functional profile instance related framing structure, there will be functional profiles and

resources categories, as expected (unless there is another level of subsystems, in which case, the rationale is applied again).

The following paragraphs show the system framing structure of USE-ME.GOV. For one of the identified subsystem's functional profile instances, the respective nested framing structure is illustrated. Further nested framing structures of this last one will not be presented here.

Fig. 7 illustrates the framing structure at the system level. It shows the subsystem's functional profile instances that get existence in the project. As it can be seen in Fig. 7, the framing structure has two major subsystem functional profiles: "Platform" and "Pilot Services". The resource categories related to subsystem functional profiles (as it also happens at the system level), have symbolic names of "Category group A", "Category group B", and so on. In these cases, it is acceptable to make no explicit identification/characterization of the resources categories. The framing structure assigns each of the subsystem functional profiles to only one resource group, giving origin to a single subsystem functional profile. The "Platform" and "Pilot Services" functional profile instances have also corresponding framing structures.

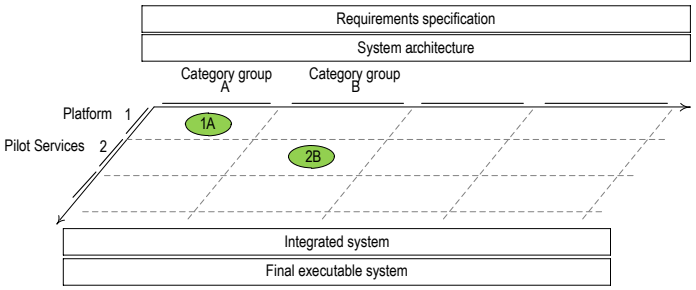

Fig. 7. Framing structure at system level for USE-ME.GOV project.

Fig. 8 illustrates the framing structure related do "Pilot Services". The Pilot Services has several subsystems, one for each of the services of "Complaint Information Broadcasting", "Mobile Student", "Healthcare Information", and "Citizen Complaint". Again, as before in the preceding framing structure, there are resource category groups; for each of the subsystems, there will be again a corresponding framing structure. Symbolic names identify the several elements of the framing structure. Note that there is no conflict on the names used for resource categories groupings, functional profiles, or functional profiles instances as the framing structure implicitly defines a namespace.

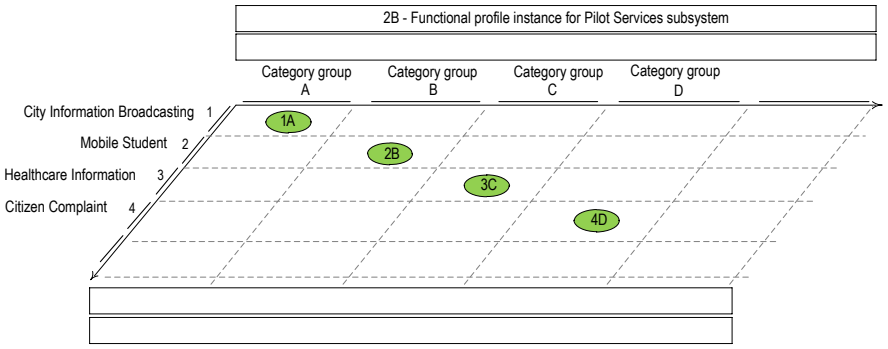

Fig. 8. Framing structure for Pilot Services subsystem of USE-ME.GOV.

The uPAIN Case Study

The uPAIN project was conceived with the purpose to create a networked informational computing system (see Fig. 9) that, making using of current wireless and mobile communication technologies, allowed to enhance hospital's anaesthesiology services on the control and monitor at pain level on post-surgery (uPAIN, 2003). It aimed to enable for better assessment and treatment of the pain phenomena by the hospital staff.

The uPAIN project was developed for the anaesthesiology services of hospitals. It consisted of an information system conceived to assist in monitoring and controlling pain of patients that stay in a relatively long period of recovery after being submitted to a surgery. During this period, analgesics are administered to them in order to minimize the pain that increases as the effects of the anaesthesia gradually disappear. This administration of analgesics is controlled by means of specialized devices called PCAs (patient controlled analgesia) based in the personal characteristics of the patient and the kind of surgery to which the patient has been submitted. The PCA can be described as "a medication- dispensing unit equipped with a pump attached to an intravenous line, which is inserted into a blood vessel in the patient's hand or arm. By means of a simple push-button mechanism, the patient is allowed to self-administer doses of pain relieving medication (narcotic) on an 'as need' basis" (Machado, Lassen, Oliveira, Couto, & Pinto, 2007).

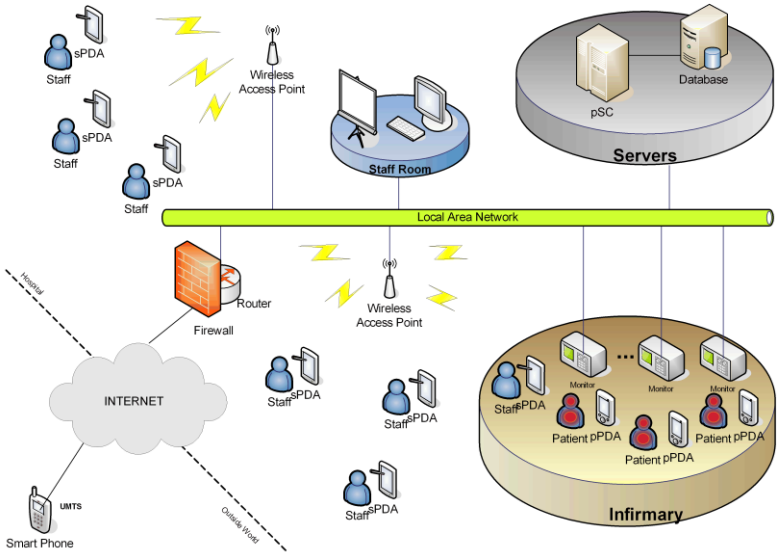

Fig. 9. General architecture for the uPAIN system.

Regarding to the framing structures, this system did not required the consideration of subsystems. Fig. 10 shows the framing structure for uPAIN. It shows the functional profiles instances that come to existence in the project. Related to each of these intances exists an elementary process development structure.

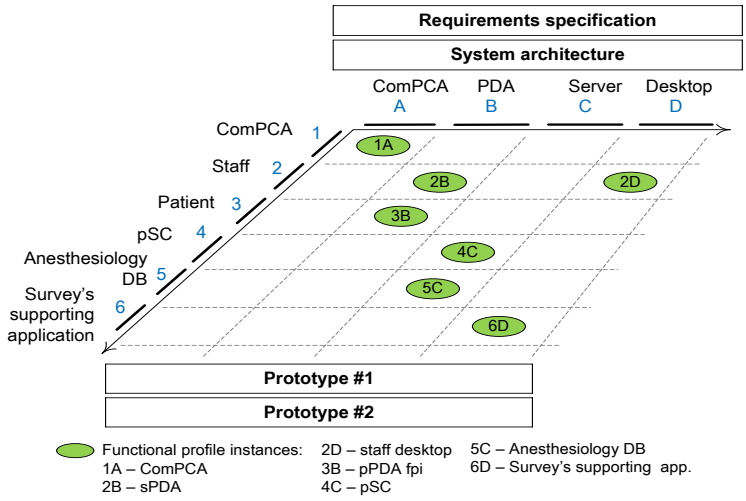

Fig. 10. Framing strcuture for uPAIN.

Synopsys of the Case Studies Analysis

The case studies USE-ME.GOV and uPAIN promoted the reasoning about the design of project structures for the model-driven development of PIS. In this context, several factors/needs emerged as being pertinent to the design of project structures to accommodate MDD for PIS in order to achive a proper, efficient, and resilient development and final system. The following paraghaps state some of these factors/needs of influence.

Project structures should be designed to support PIS. The way the elements are structured can have a positive impact in coping with heterogeneity in devices and in functionalities of PIS. The development structure approach, jointly with the profiling and framing structures herein presented, provides assisting techniques to deal with pervasive characteristics such as heterogeneity of devices and changing functionalities.

Projects need an explicit manifestation of a model-driven approach. The projects must have, in the project design, a clear strategy to accommodate a model-driven approach making use of models beyond of schematic or documentatinal purposes in the several phases of the project.

Project elements must be properly defined. It is important to pay attention to several issues that may occur in the definition of project elements. Among these, are the lack of explicit artefacts, activities, or relationships; the inconsistent or improper naming; the incoherent sequence activities; or the misused of conceptions. The attention given to them is important as they are at a core level where it is fundamental to assure its correctness in order to pursuit, at higher levels of abstraction, the goals of model-driven development.

Projects should formalize activites as model transformations and other elements with semantic correcteness. Without having a coherent, consistent, and clear formalization of the several projects constituents' elements, it will not be possible to establish, with an acceptable quality, a model-based/driven process development. Without the existence of coherently interconnected and precise process elements, it is hard, even impossible, to achieve a model-based/driven development orientation at

a large extent and depth of the process. This is the consequence of the difficulties in: (i) incorporating new activities or optimizing the existing ones with model transformations techniques; (ii) reorganizing or redefining the process in order to pursuit a clearer and enhanced model-based/driven quality.

Projects should seek for model-driven semantic continuity/visibility. How much model-based/driven is a software development process? When does a software development project go from being model-"based" to being model-"driven"? It is important to reason about the robustness of process and the suitability of activities and artefacts regarding its use on a model-based/driven orientation. The usability of an artefact is related to its expression and ability to be consumed/reused on subsequent modelling tasks. The suitability of an activity is related to its ability to incorporate formal/explicit model transformation techniques that (optionally) consume models and produce models. The robustness of the process is related to the degree of the modelling semantic continuity provided by the chains of activities, from the beginning to the end of the development process. The links among model artefacts and model transformation activities (or well-structured and formalized activities) of the process define the visibility. The longer the path, the more model-driven is the development process. So, to enhance model-based/driven visibility, it is needed to pay attention to activities (or tasks) and the realization and flow of models.

The activities and artifacts of development process, either at global of elementary process, can be described using the Software & Systems Process Engineering Meta-model Specification (SPEM) 2.0 (OMG, 2008). SPEM provides to process engineers a conceptual framework for modelling method contents and processes, and as such, it is used to define software and systems development processes and their components. SPEM can be an important auxiliary tool for the definition (or diagonosis or optimization) of processes. SPEM 2.0 specification provides, not only the metamodel, but also a set of corresponding stereotypes of the metamodel concepts that can be used to easier illustrate the process elements.

Conclusion

Pervasive forms of information system are increasingly predominating on landscape of software systems development. Among others, resources heterogeneity, increased number of functionalities that may be simultaneously accomplished by distinct resources, high pace of changes on resources and requirements characterizes PIS. These have to be taken into account by a suitable approach to software development for PIS. Some properties of process structures should be seek in order to achieve robustness of a development process definition, such as the comprehensiveness and depth of the structure of the process, semantic correctness, naming coherency and consistency, activity flows and input/output clearness, work unit's robustness, overall rationale, and model-based/driven visibility. Satisfaction of these properties contributes for the perception of a solid ground for project development.

This paper presents a profiling and framing structure approach for the development of PIS. This profiling and framing structure allows the organization of the functionality that can be assigned to computational devices in a system and of the corresponding development structures and models. The proposed approach allows accommodating the profiling of functionalities that can be assigned to several resource categories and enables a structural approach to PIS development. Analysing two real cases studies, we have concluded that the strategy inherent to this profiling and framing structure reveals as being able to cope with systems composed of several subsystems, while keeping the capacity to deal with heterogeneous devices and to accommodate model-based/driven approaches.

SPEM, besides being useful for the analysis and design of processes, can also be extended to represent the concepts used in this profiling and framing structure, and can also be used to represent a process structure pattern for application of these concepts. These developments shall be subject of further work.

References

Appukuttan, B., Clark, T., Reddy, S., Tratt, L., & Venkatesh, R. (2003). A model driven approach to model transformations.

Proceedings of Model Driven Architecture: Foundations and Applications (CTIT Technical Report TR–CTIT–03–27, University of Twente).

Ark, W. S., & Selker, T. (1999). A look at human interaction with pervasive computers. *Ibm Systems Journal, 38*(4), 504-507.

Atkinson, C., & Kuhne, T. (2003). Model-driven development: a metamodeling foundation. *Software, IEEE, 20*(5), 36-41. doi: 10.1109/ms.2003.1231149

Benincasa, G. P., Daneels, A., Heymans, P., & Serre, C. (1985). Engineering a Large Application Software Project: The Controls of the CERN PS Accelerator Complex. *Nuclear Science, IEEE Transactions on, 32*(5), 2029-2031.

Bohn, J., Coroamã, V., Langheinrich, M., Mattern, F., & Rohs, M. (2004). Living in a world of smart everyday objects - social, economic, and ethical implications. *Human and Ecological Risk Assessment, 10*(5).

Booch, G., Maksimchuk, R. A., Engle, M. W., Young, B. J., Conallen, J., & Houston, K. A. (2007). *Object-Oriented Analysis and Design with Applications*: Addison-Wesley.

Brown, A. W. (2004). Model driven architecture: Principles and practice. *Software and Systems Modeling, 3*, 314-327.

Ciarletta, L., & Dima, A. (2000). *A conceptual model for pervasive computing.* Paper presented at the International Workshops on Parallel Processing, 2000.

Davies, N., & Gellersen, H.-W. (2002). Beyond prototypes: challenges in deploying ubiquitous systems. *Pervasive Computing, IEEE, 1,* 26-35.

Fano, A., & Gershman, A. (2002). The future of business services in the age of ubiquitous computing. *Communications of ACM, 45*(12), 83-87.

Fernandes, J. E., Machado, R. J., & Carvalho, J. Á. (2004). *Model-Driven Methodologies for Pervasive Information Systems Development*. Paper presented at the MOMPES'2004, Hamilton, Ontario, Canada.

Fernandes, J. E., Machado, R. J., & Carvalho, J. Á. (2007). *Model-Driven Software Development for Pervasive Information Systems Implementation*. Paper presented at the QUATIC 2007 (SEDES Workshop), Lisbon, Portugal.

Fernandes, J. E., Machado, R. J., & Carvalho, J. Á. (2008). Model-Driven Development for Pervasive Information Systems. In S. K. Mostefaoui, Z. Maamar & G. M. Giaglis (Eds.), *Advances in Ubiquitous Computing: Future Paradigms and Directions* (pp. 45-82): IGI Publishing.

Gorton, I., & Liu, Y. (2010). *Advancing software architecture modeling for large scale heterogeneous systems*. Paper presented at the FSE/SDP workshop, FoSER 2010, Santa Fe, New Mexico, USA.

Hansmann, U., Merck, L., Nicklous, M. S., & Stober, T. (2003). *Pervasve Computing* (2nd ed.): Springer.

Heijstek, W., & Chaudron, M. R. V. (2009). *Empirical Investigations of Model Size, Complexity and Effort in a Large Scale, Distributed Model Driven Development Process*. Paper presented at the SEAA '09.

Kay, R. H. (1969). *The management and organization of large scale software development projects*. Paper presented at the AFIPS '69 (Spring), Boston, Massachusetts.

Laine, P. K. (2001). *The role of SW architecture in solving fundamental problems in object-oriented development of large embedded SW systems*. Paper presented at the Working IEEE/IFIP Conference on Software Architecture, 2001.

Langheinrich, M., Coroama, V., Bohn, J., & Rohs, M. (2002). As we may live – Real-world implications of ubiquitous computing: Distributed Systems Group, Institute of Information Systems, Swiss Federal Institute of Technology, ETH Zurich, Switzerland.

Lyytinen, K., & Yoo, Y. (2002). Introduction [Issues and challenges in ubiquitous computing]. *Communications of ACM, 45*(12), 62-65.

Machado, R. J., Lassen, K. B., Oliveira, S., Couto, M., & Pinto, P. (2007). Requirements Validation: Execution of UML Models with CPN Tools. *International Journal on Software Tools for Technology Transfer (STTT), 9*(3), 353-369.

Mattsson, A., Lundell, B., Lings, B., & Fitzgerald, B. (2007). *Experiences from Representing Software Architecture in a Large Industrial Project Using Model Driven Development*. Paper presented at the SHARK-ADI'07.

Medvidovic, N. (2005). Software architectures and embedded systems: a match made in heaven? *Software, IEEE, 22*(5), 83-86.

Mellor, S. J., Clark, A. N., & Futagami, T. (2003). Model-driven development - Guest editor's introduction. *Software, IEEE, 20*(5), 14-18.

Miller, G., Ambler, S., Cook, S., Mellor, S., Frank, K., & Kern, J. (2004). *Model driven architecture: the realities, a year later*. Paper presented at the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications, Vancouver, BC, CANADA.

Mirakhorli, M., Sharifloo, A. A., & Shams, F. (2008). *Architectural challenges of ultra large scale systems*. Paper presented at the 2nd international workshop on Ultra-large-scale software-intensive systems, Leipzig, Germany.

OMG. (2003). OMG's MDA Guide Version 1.0.1, from http://www.omg.org/docs/omg/03-06-01.pdf

OMG. (2005). OMG's Home Page, from http://www.omg.org

OMG. (2008). SPEM v2.0 - Software & Systems Process Engineering Meta-Model Specification v2.0: OMG.

Ruparelia, N. B. (2010). Software development lifecycle models. *SIGSOFT Softw. Eng. Notes, 35*(3), 8-13. doi: 10.1145/1764810.1764814

Sage, A. P., & Rouse, W. B. (1999). Information Systems Frontiers in Knowledge Management. *Information Systems Frontiers, 1*(3), 205-219.

Saha, D., & Mukherjee, A. (2003). Pervasive computing: a paradigm for the 21st century. *Computer, 36*(3), 25-31.

Seidewitz, E. (2003). What models mean. *Software, IEEE, 20*(5), 26-32.

Selic, B. (2003). *Model-driven development of real-time software using OMG standards.* Paper presented at the Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 2003 (ISORC'03).

Sendall, S., & Kozaczynski, W. (2003). Model transformation: the heart and soul of model-driven software development. *Software, IEEE, 20*(5), 42-45.

Thomas, D. (2004). MDA: revenge of the modelers or UML utopia? *Software, IEEE, 21*(3), 15-17.

uPAIN. (2003). uPAIN - Candidatura de Submissão.

USE-ME.GOV. (2003). Consortium Agreement - Annex I - Description of Work.

USE-ME.GOV. (2006). D3.1 Recommendations.

Want, R., Pering, T., Borriello, G., & Farkas, K. I. (2002). Disappearing hardware [ubiquitous computing]. *Pervasive Computing, IEEE, 1,* 36-47.

Weiser, M. (1993a). Hot topics-ubiquitous computing. *Computer, 26*(10), 71-72.

Weiser, M. (1993b). Some computer science issues in ubiquitous computing. *Communications of ACM, 36*(7), 75-84. doi: http://doi.acm.org/10.1145/159544.159617

Weiser, M., Gold, R., & Brown, J. S. (1999). The origins of ubiquitous computing research at PARC in the late 1980s. *Ibm Systems Journal, 38*(4), 693-696.