Premier Reference Source

Systems and Software Development, Modeling, and Analysis

New Perspectives and Methodologies

Mehdi Khosrow-Pour

Development Framework Pattern for Pervasive Information Systems

José Eduardo Fernandes[1], Ricardo J. Machado[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@dsi.uminho.pt

Abstract

During last decade the world watched to a social acceptance of computing and computers, enhanced information technology devices, wireless networks, and Internet, they gradually became a fundamental resource for individuals. Nowadays, people, organizations, and the environment are empowered by computing devices and systems; they depend on services offered by modern Pervasive Information Systems supported by complex software systems and technology. Research on software development for PIS brought information on issues and challenges on software development for them, and provided several contributions. Among these contributions are a development framework for PIS, a profiling and framing structure approach, and a SPEM 2.0 extension. This chapter, revisiting these contributions, provides an additional contribution: a pattern to support the use of the development framework and profiling approach on software development for PIS. This contribution completes a first series of contributions for the development of PIS. This chapter also presents a case study that allowed demonstrating the applicability of these contributions.

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. 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 (PIS) 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. Software Development Processes (SDPs), as well as generalized adoption of models, are fundamental to efficient development efforts of successful software systems.

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 document structures its content in the following sections: an introduction section that synthesizes pervasive information systems, its issues and the benefits of a model-based/driven development based approach; a software development for PIS section that provides a background into related research works for PIS; a development framework pattern section that presents the pattern as the contribute in this work; a section case study section that presents a case study wherein the contributions are demonstrated; and a last section that presents the conclusions and finishes this document.

Pervasive Information 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. 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.

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).

Model-Driven Development

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 software 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, Clark, & Futagami,

2003). Synthetically, models, in a descriptive or a prescriptive form, can then be used to: (i) understand or communicate a problem, an existing system, or a proposed solution; (ii) analyze, or predict on changes, systems properties or risk failures; (iii) productivity improvement; (iv) reduction of system's development costs.

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 et al., 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).

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.

SOFTWARE DEVELOPMENT FOR PIS

Research performed on pervasive information systems and model-driven development (Fernandes, Machado, & Carvalho, 2007) brought several contributions on the application of MDD concepts and techniques to software of PIS. Among these contribution are a 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), a profiling and framing structure approach, and a SPEM extension. The following subsections present a brief overview of these contributions.

Development Framework for PIS

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.

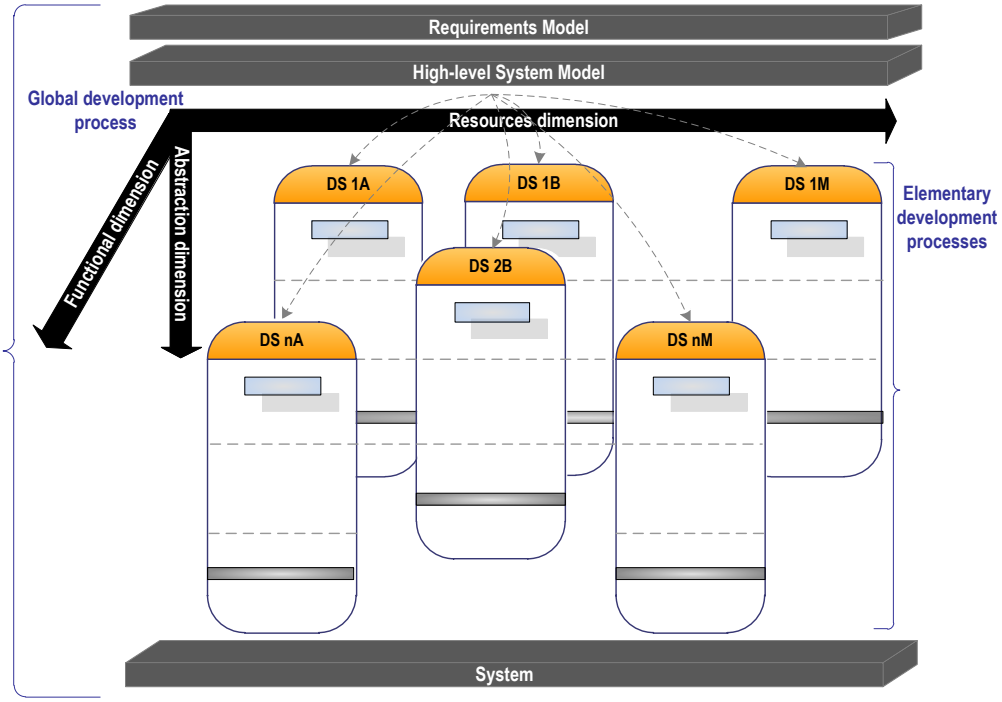

Fig. 1. Development framework for PIS.

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. Therefore, 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. 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.

Profiling and Framing Structures

In the context of the previously presented development framework, Fernandes et al. (Fernandes, Machado, & Carvalho, 2012a, 2012b) provided an approach to effectively and consistently apply it in PIS development projects, independently of its size. The next paragraphs start by presenting some considerations regarding functional profile instantiation, and modeling levels in development structures; then they illustrate 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. 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.

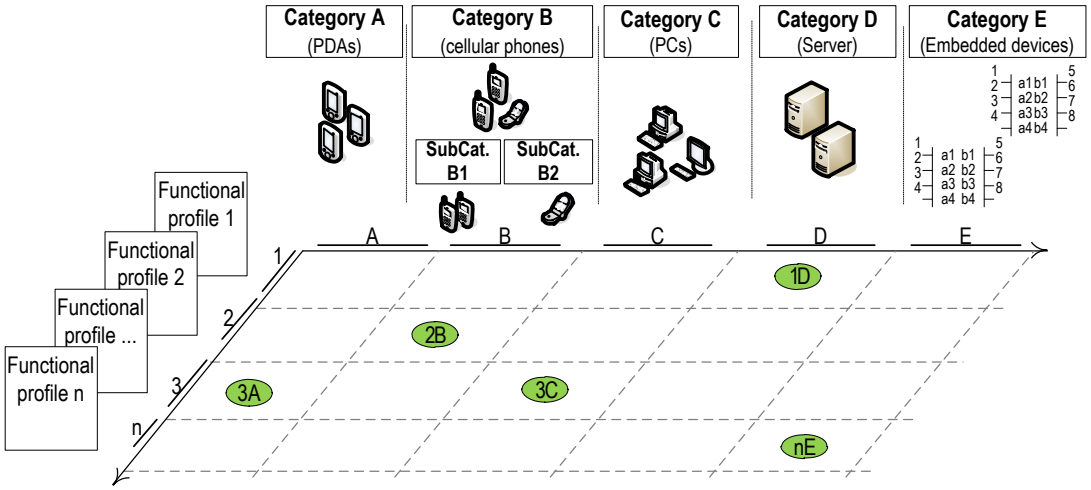

Fig. 2. Functional profile instances.

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.

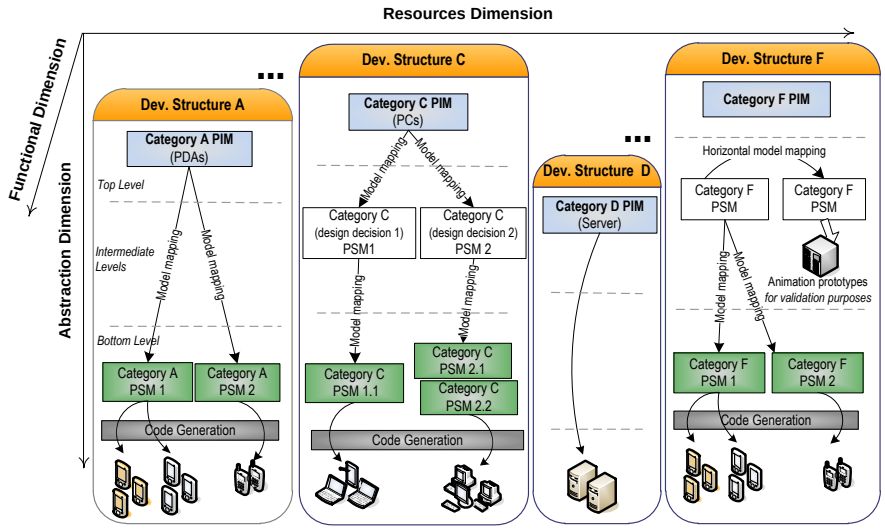

Fig. 3. *Modeling levels in development structures (abstraction dimension).*

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.

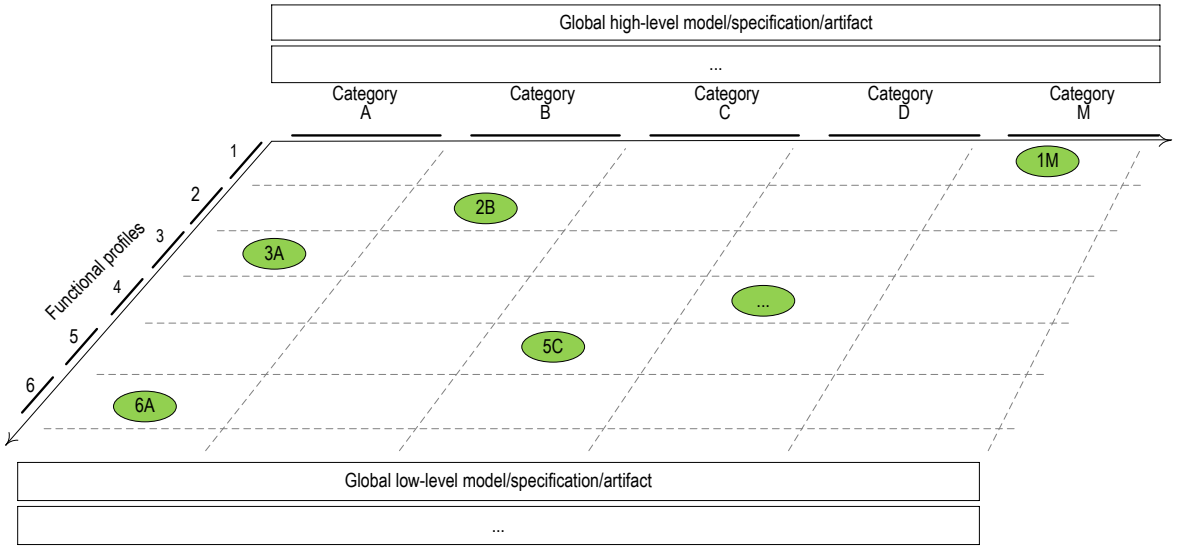

Fig. 4. *Framing structure for a project.*

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.

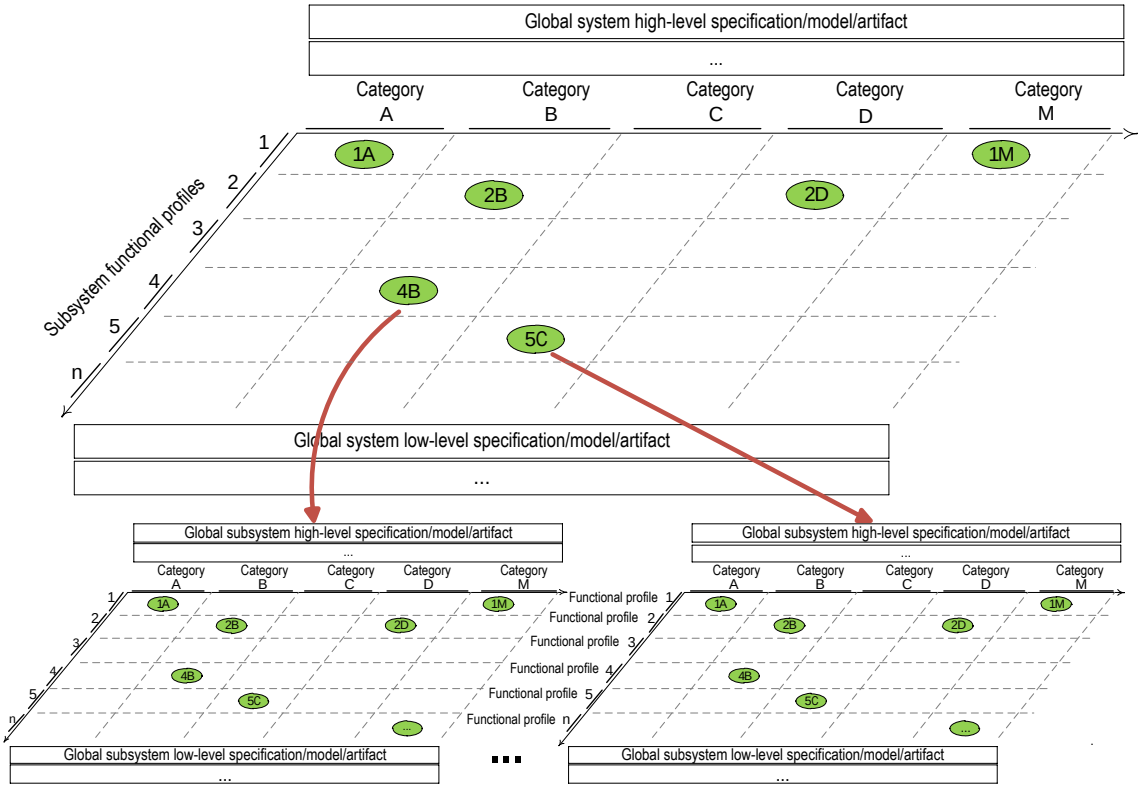

Fig. 5. *Nesting of framing structures for large projects.*

SPEM 2.0 Extension for PIS

Software & Systems Process Engineering Meta-Model Specification (SPEM) (OMG, 2008) is a current standard published by the OMG (OMG) for the description of systems and software processes. SPEM provides, to process engineers, conceptions for modeling method contents and processes. SPEM 2.0 is defined as a meta-model as well as a UML 2 Profile (concepts are defined as meta-model classes as well as UML stereotypes). This section briefly introduces the extension to SPEM (version 2.0) Base Plug-In Profile proposed by Fernandes and Machado (Fernandes & Machado, 2012) that includes stereotypes needed to support a suitable structural process organization for MDD approaches aiming to develop software for PIS.

The stereotypes proposed extend two main groups of stereotypes defined in SPEM 2.0 Base Plug-in: the “ActivityKind” and “WorkProductKind” stereotypes. Fig. 6 and Fig. 7 illustrate, respectively, the new "ActivityKind" and "WorkProductKind" stereotypes (white boxes contain the predefined kinds; grey boxes contain the proposed additional kinds). The following paragraphs describe these new stereotypes, grouped by each of those kinds.

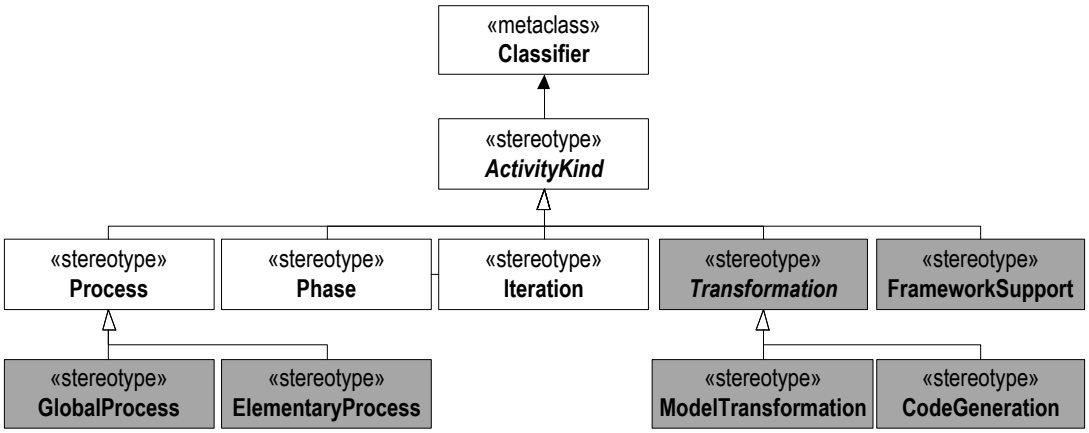

Fig. 6 - New "ActivityKind" stereotypes.

Regarding to "ActivityKind" stereotypes (Fig. 6), in addition to the predefined “Process”, “Phase”, and “Iteration” stereotypes, there are the stereotypes “FrameworkSupport” and “Transformation” along with its specializations “ModelTransformation” and “CodeGeneration. Additionally, to the “Process” of "ActivityKind", come to existence as specializations the “GlobalProcess” and “ElementaryProcess” stereotypes. The purpose of each of these "ActivityKind" stereotypes is explained in the following paragraphs.

The “GlobalProcess” stereotype allows the representation of the global process that encompasses the overall development of the system (as considered in the development structure). The convenience of this stereotype arises since an approach consistent to the development structure will have two major types of processes: a global process and several elementary development processes. This stereotype allows representing such overall process and also to relate with the overall main activities. The “ElementaryProcess” stereotype allows the representation of an elementary development process that exists for a Development structure associated with each functional profile instance. The convenience of this stereotype is analogous to the “GlobalProcess” stereotype: this stereotype allows the representation of an elementary development process and of the relationships of its inherent main activities. The “Transformation” stereotype is an abstract generalization that represents the activities of transformation models or other artifacts. Model-based/driven approaches are rich on these transformations; the specializations of this abstract stereotype allow to represent such transformations and to give further emphasis on its formalization. Specializations of this the “Transformation” stereotype are “ModelTransformation” and “CodeGeneration” stereotypes, which are next described. The “ModelTransformation” stereotype, a specialization of the “Transformation” stereotype, intends to represent activities that transform models into other kinds of model. The “CodeGeneration” stereotype intends to represent activities that transform models into code, or any other suitable artifact into code (for example, source code into executable code). The “FrameworkSupport” stereotype has a particular use. It does not map into any element of the development structure, but is essential for the overall structuring of development structure. The “FrameworkSupport” stereotype intends to represent any special activity related to the organization and deployment of the development framework, such as assisting the definition of the resources categories, functional profiles, functional profile instances, or elementary development processes.

Regarding to the "WorkProductKind" stereotypes (Fig. 7), in addition to “Artifact”, “Deliverable”, and “Outcome” stereotypes, there are the “FunctionalProfile”,

"ResourceCategory", and "FP_Instance" stereotypes. These stereotypes aim to represent work products directly related to the development structure. These stereotype are convenient as they allow the representation of structural elements of the development structure.

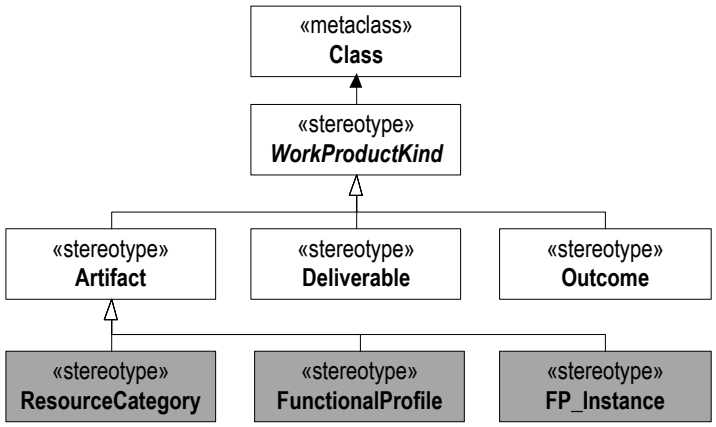

Fig. 7 - New "WorkProductKind" stereotypes.

DEVELOPMENT FRAMEWORK PATTERN

Designing software for complex systems is not easy, and reuse of workable solutions for recurrent problems is fundamental: "One thing expert designers know not to do is solve every problem from first principles. Rather, they reuse solutions that have worked for them in the past" (Gamma, Helm, Johnson, & Vlissides, 1995). Patterns, as general solutions for common problems, are important when crafting complex systems and can also be used to perceive the quality of system constructions. Being used in several disciplines, the use of patterns in software development has been promoted for years through conferences, books, and groups such as the Hillside Group (HillSide, 2013).

The pattern herein presented represents a technique and a set of actions to assist the process of software development for pervasive information systems. It facilitates the organization and documentation of requirements/analysis and design decisions regarding the (de)composition of functional responsibilities/profiles among the several sub-systems and devices of a PIS. The pattern is to be adopted at initial phase of the software development process, and is not meant to give origin to code directly. The foundational concepts, semantics, notational elements needed for its adoption were already set by the previous contributions of the development framework, the profiling and framing approach, and SPEM 2.0 extension for PIS.

In order to build a consistent knowledge on patterns and to facilitate their adoption, patterns are organized into pattern catalogues containing groups of patterns set through classification schemes. There are several classification schemes, each using well-defined criteria setting-up the relevant characteristics that sustain the classification and grouping of patterns. One of these classification schemes is proposed by Azevedo *et al.* (Azevedo, Machado, Bragança, & Ribeiro, 2011) that takes a multilevel and multistage classification approach based on Rational Unified Process (RUP) and OMG Four-Layer Architecture. This scheme considers a discipline dimension, a stage dimension, a level dimension, and a domain nature attribute; it also considers a set of pattern type. Observing this scheme, this PIS pattern is considered as follows: regarding the *Discipline dimension*, this pattern can be adopted both within RUP's Requirements and Analysis and Design Disciplines; regarding the *Stage dimension* this pattern can be adopted at both RUP's

Inception and Elaboration Stages); regarding the *Level Dimension*, this pattern is at M2-Level (see Fig.8); regarding the *Domain nature* this patterns is agnostic, as it is applicable to both vertical and horizontal domains; regarding the *Pattern type*, none of the pattern types presented are applicable (it is a process pattern type, whose main target is not the system or part of it, but the development process itself).

Pattern Structure

- As state previously, the PIS Development Framework Pattern assumes the context PIS and the conceptions aforementioned, such as the development framework, the profiling and framing approach and the SPEM 2.0 Plug-In extension for PIS. This pattern assist in the task of defining or restructuring a SPEM-based software development process for PIS. Fig. 8 illustrates the pattern for organizing the process elements according to the development framework for PIS; as it can be seen, the pattern makes use of the extending SPEM stereotypes. This pattern arranges the extending stereotypes in the following way:
- An activity kind instance stereotyped as «global process», named with project's name. It includes, besides all the major activities of the project that are deemed as being global process activities, a special activity named "Framework creation" that is stereotyped as «FrameworkSupport».
- An activity kind instance stereotyped as «FrameworkSupport», named as "Framework Creation". This is instance central to the incorporation of the several concepts of development framework. It includes activities for resources categories definition, functional profiles definition, functional profiles instances definition, and elementary process creation.
- An activity instance stereotyped as «FrameworkSupport», named as "Resources Categories Definition". This activity instance has the responsibility to provide the definition of the resources categories, which is materialized by the artefact "Resource Categories" that includes a work product instance stereotyped as «ResourceCategory» for each of the resource categories (symbolically named as A, ..M).
- An activity instance stereotyped as «FrameworkSupport», named as "Functional Profiles Definition". This activity instance has the responsibility to provide the definition of the functional profiles, materialized by the artefact "Functional Profiles", which includes a work product instance stereotyped as «FunctionalProfile» for each of the functional profile (symbolically named as 1, ...n).
- An activity instance stereotyped as «FrameworkSupport», named as "FP Instances Definition". This activity instance has the responsibility to provide the definition of the functional profiles instances, materialized by the artefact "Functional Profile Instances", which includes a work product instance stereotyped as «FP_Instance» for each of the functional profile instance (symbolically named in the pattern as 1A…, ...nM). This activity has as input the artefacts "Functional Profiles" and "Resource Profiles".
- An activity instance stereotyped as «FrameworkSupport», named as "Elementary Processes Creation". This activity instance has the responsibility to provide the creation of the elementary processes. It includes one activity instance stereotyped as «ElementaryProcess» for each of the defined functional profile instances, and each giving

origin to a conceptual development structure (these elementary processes are symbolic named in the pattern as 1A…, …nM). This activity has as input the artefact "Functional Profiles Instances".

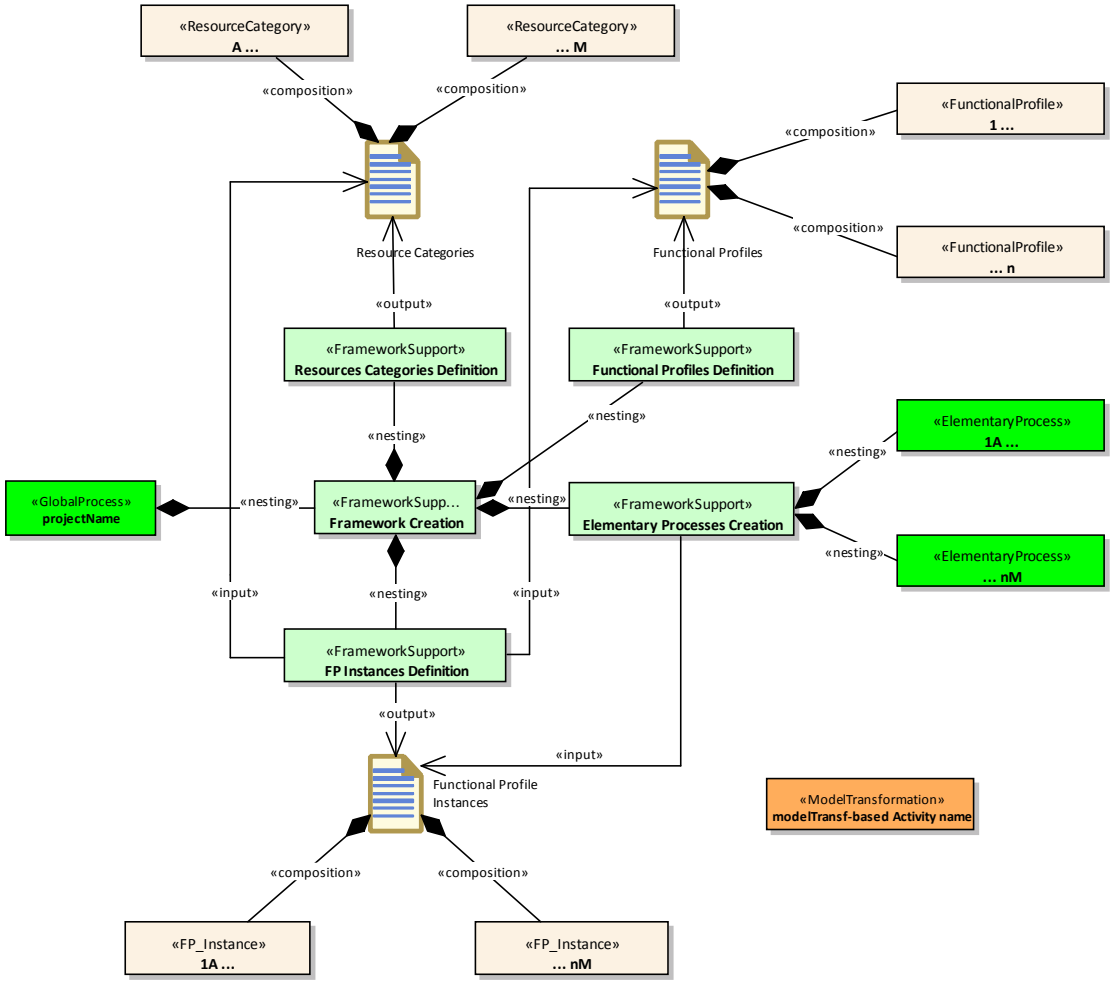

Fig. 8. Structural Development Pattern for PIS.

Complementing the presentation, and belonging to the definition of this pattern, some notes are stated about the use of the pattern application and, additionally, some guidance actions are also provided. First, it is possible to intersperse activities among the patterns activities. For example, in the redefinition of a SPEM diagram of a development process, it is possible to put an existing major development activity between the «GlobalProcess» activity and the «FrameworkSupport» activity named "FrameworkCreation". Nonetheless, the structural organization of the pattern should be present and recognized in the overall SPEM model. Second, the realization of this pattern is considered complete after the identification of the transformations. Third, after the complete realization of the pattern, other actions can be undertaken, being these ones out of the scope of this pattern definition.

Guidelines for Applying the Pattern

For the realization of the pattern, some guidance actions are provided. Table 1 list these steps, grouped in four tasks; each of this task has a variable number of steps.

Table 1. Guidance steps for pattern application.

Task Id.	Task name / Step id. – Step name. Step description
1	**Elements identification** S1.1 – Identify global process activities. Identify the activities that should be considered as belonging to the scope of global process. S1.2 – Identify elementary process activities. Identify the activities and respective structure that shall be considered as belonging to elementary processes. S1.3 - Identify resources categories. S1.4 - Identify functional profiles. S1.5 – Define functional profile instances. Define the functional profile instances that shall come to existence. You may use a framing structure to visualize the functional profile instances.
2	**Pattern creation** S2.1 – Create the pattern structure. Create in the SPEM diagram, the raw pattern structure, materializing the «GlobalProcess», «ResourceCategory», «FunctionalProfile», «FP_Instance», and «ElementaryProcess» stereotype instances with the information available. All «FrameworkSupport» can be materialized with the name used in the pattern (nonetheless, if deemed relevant, other names can be given).
3	**Pattern framing** S.3.1 – Include global process activities. Include in the «GlobalProcess» activity instance, all the identified global process activities. S3.2 - Include elementary process activities. Include each of the «ElementaryProcess» activity instances, as well as any corresponding activity structures of them. Make the adaptions and rearrangements needed.
4	**Transformations identification** S4.1 – Identify transformations. After the (re)arrangements for proper pattern framing, it should be identified the existing activities that are susceptible to be classified as a transformation; for the ones that are validated as such, replace the activity by and activity instance stereotyped as «ModelTranformation» or «CodeGeneration» as appropriate. S4.2 – Formalize the transformations. For each of the transformations, formalize, in a separate SPEM diagram, the transformation.

These guidance actions for pattern realization can be also formalized in a SPEM 2.0 model, using the concepts of activity, tasks, and steps. Fig. 9 depicts the pattern realization SPEM 2.0 model.

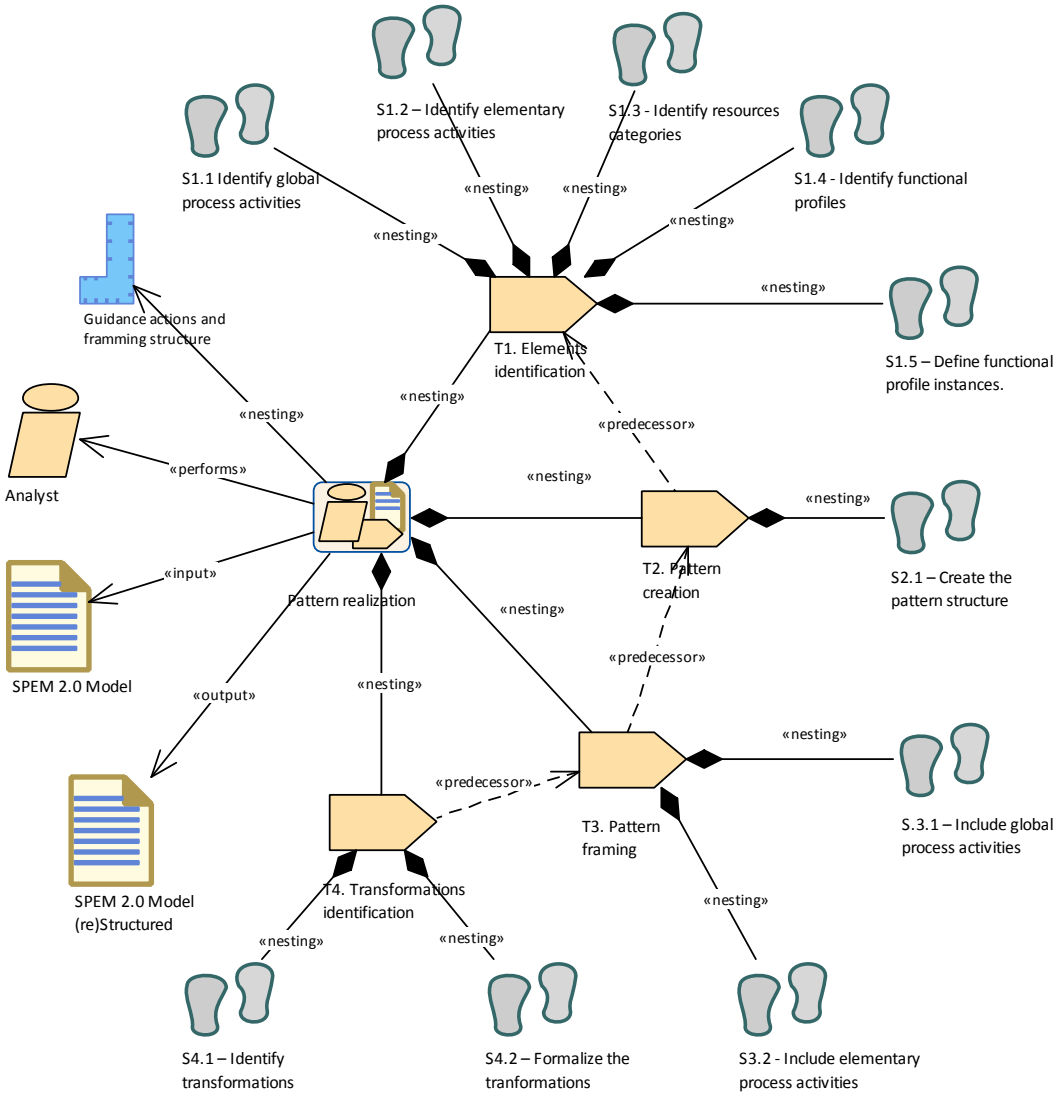

Fig. 9. SPEM diagram of pattern realization.

CASE STUDY

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. Then, it illustrates the application of the development framework in this project. Attending to the project dimension, 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 Project

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. 10.

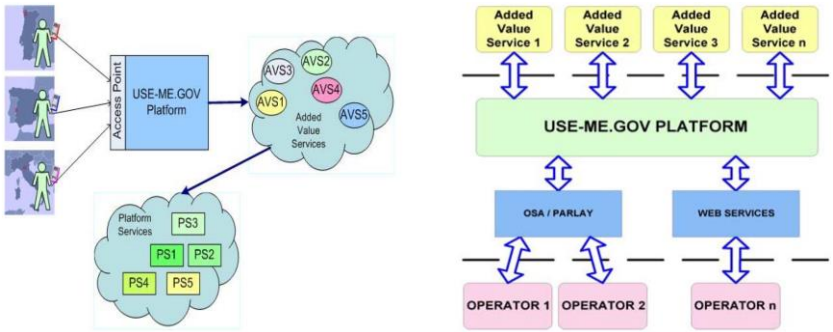

Fig. 10. *USE-ME.GOV System General Architecture (from (USE-ME.GOV, 2006)).*

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 profiling and framing approach, 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. 11 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. 11, 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.

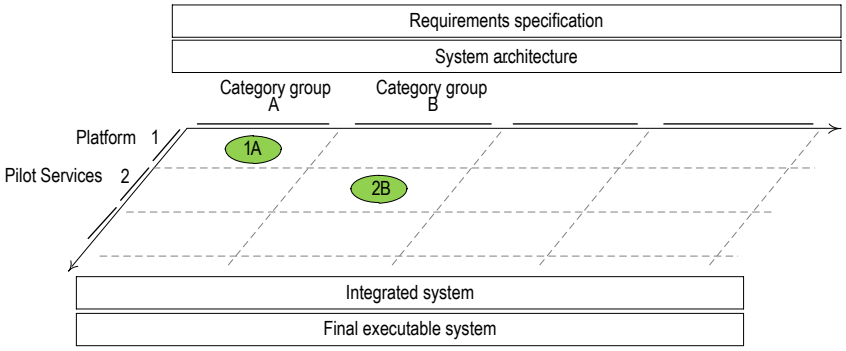

Fig. 11. Framing structure at system level for USE-ME.GOV project.

Fig. 12 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.

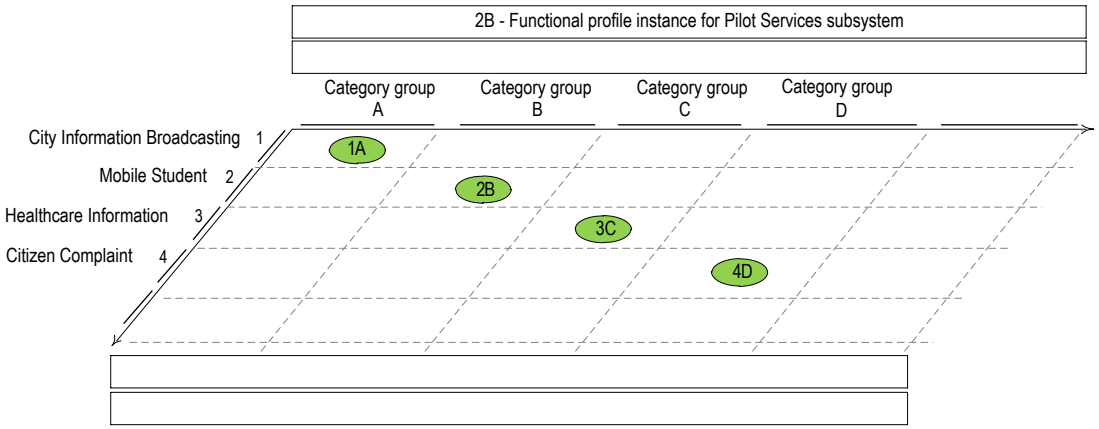

Fig. 12. Framing structure for Pilot Services subsystem of USE-ME.GOV.

The Pattern in USE-ME.GOV

The following paragraphs present the use of the pattern related to the framing structure at system level (Fig. 13). For the remaining framing structures, the rationale of the corresponding realizations of the development framework patterns is identical to this pattern or to the one already presented in the earlier project. For the realizations of the patterns in the SPEM model, the approach reorganizes, modifies, or properly creates the existing activities.

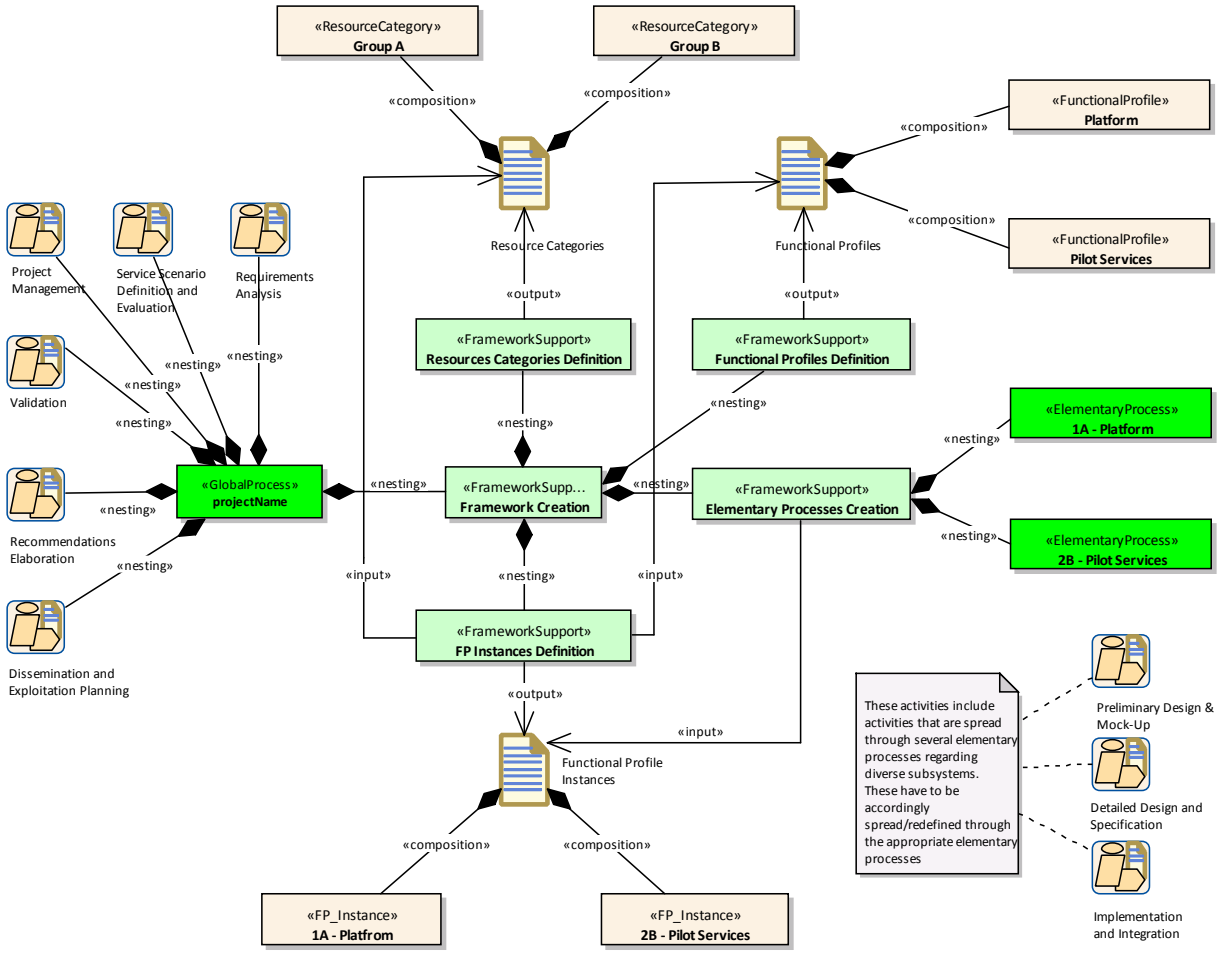

Fig. 13. USE-ME.GOV's after pattern adoption (major structuring elements at system level).

The global process activity of "Project Management" includes "Project Coordination" and "Project Assessment" activities. "Recommendations" global process activity includes "Business Planning", "Implementation Planning", and "Standardization and Regulations" activities. The restructuration has to distribute (and eventually to redefine) the activities of "Preliminary Design & Mock-Up", "Detailed Design ad Specification", and "Implementation and Integration" by the several elementary processes regarding diverse subsystems.

Fig. 14 illustrates the nesting of a framing structure achieved through new realization of the development framework pattern in connection to an elementary process of a higher framing structure. This elementary process assumes in the pattern the place usually allocated to the global development process. Note that there is a namespace established by the framing structure, reflected in this case by the symbolic designation of the element that is representing the global process, in this case the elementary process "1A-Platform". As such, the absolute name of another relative identified element in the diagram will be, its element's type reference followed by a tag in the form 1A.relative_id_of_diagramElement. For example, "elementary process 1A.3C" refers to the elementary process "3C-Platform Services".

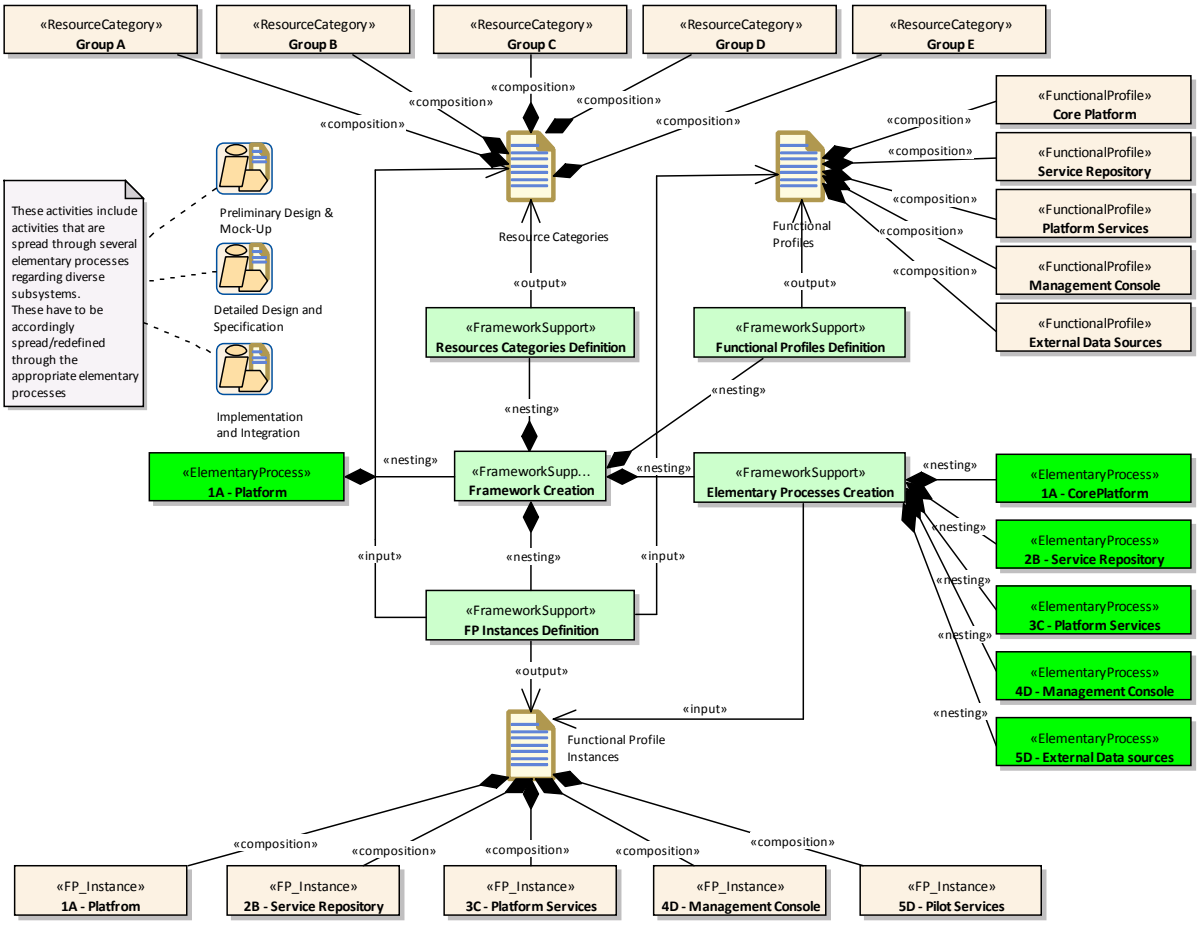

Fig. 14. USE-ME.GOV after pattern adoption (major structuring elements at Platform subsystem level).

Fig. 15 illustrates another nesting of a framing structure, through further realization of the development framework pattern in connection to an elementary process of a higher framing structure. In this case, it respects to the "Pilot Services" subsystem.

Synopsys

The case study USE-ME.GOV 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 achieve a proper, efficient, and resilient development and final system. The following paragraphs 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 framework (jointly with the profiling and framing approach) and the pattern herein presented, provide techniques to deal with pervasive characteristics such as heterogeneity of devices and changing functionalities.

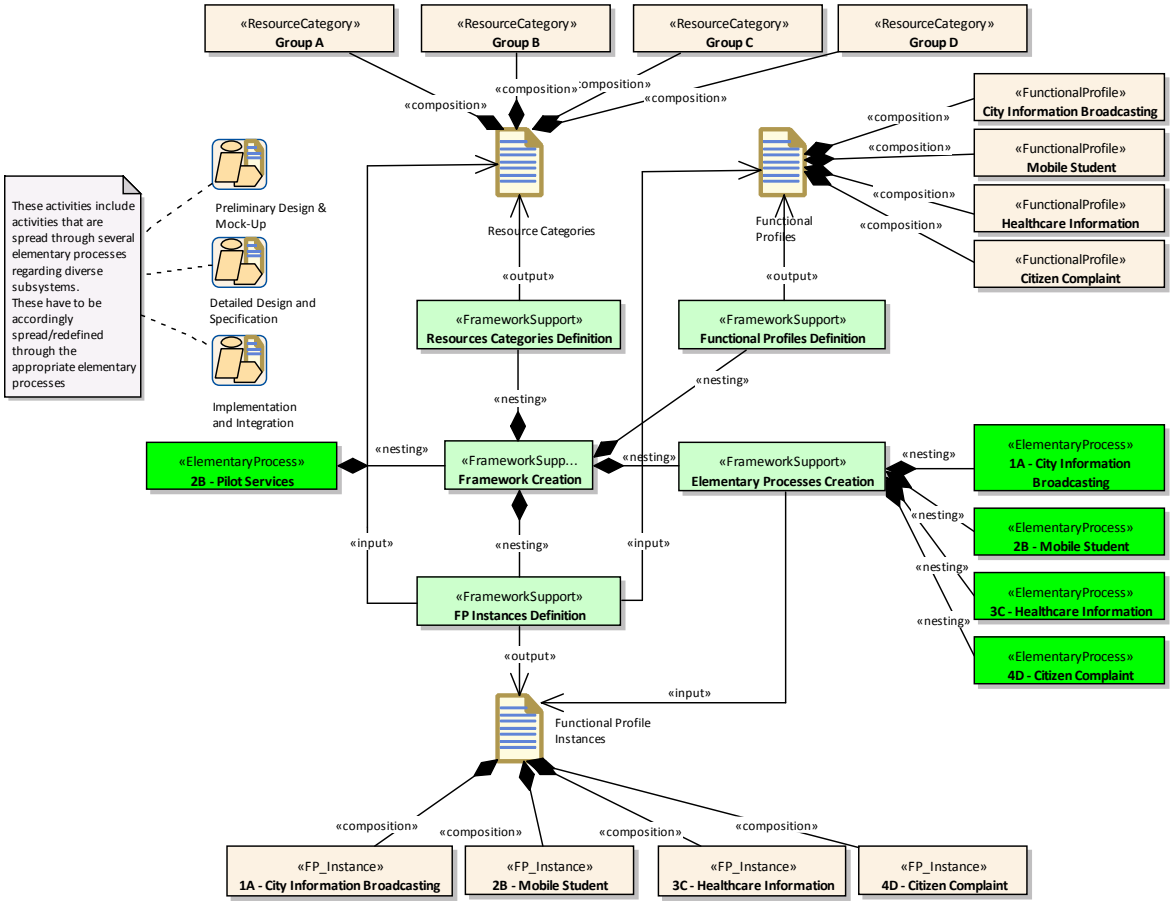

Fig. 15. USE-ME.GOV after pattern adoption (major structuring elements at Pilot Services subsystem level).

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 documentation 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 activities as model transformations and other elements with semantic correctness. 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 diagnosis or optimization) of processes.

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 document revisited development structures and techniques for the development of PIS and presented a development framework pattern for the software development of PIS. Those structures, techniques, and pattern allow 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. Through a real case study, we have concluded that the strategy inherent to this development pattern and its inherent conceptions 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.

References

Appukuttan, B., Clark, T., Reddy, S., Tratt, L., & Venkatesh, R. (2003). A model driven approach to model transformations. In *Proceedings of Model Driven Architecture: Foundations and Applications,* (pp. 1-12). Enschede, Netherlands: 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

Azevedo, S., Machado, R. J., Bragança, A., & Ribeiro, H. (2011). Systematic use of software development patterns through a multilevel and multistage classification. In J. Osis & E. Asnina (Eds.), *Model-Driven Domain Analysis and Software Development: Architectures and Functions,* (pp. 304-333). IGI Global.

Benincasa, G. P., Daneels, A., Heymans, P., & Serre, Ch. (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).

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.* In Proceedings of Parallel Processing, 2000, (pp. 9-15). Washington, DC: IEEE Computer Society.

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. (2012). SPEM 2.0 Extension for pervasive information systems. *WSEAS Transactions on Computers, 11*(9), 10.

Fernandes, J. E., Machado, R. J., & Carvalho, J. (2004). Model-driven methodologies for pervasive information systems development. In *Proceedings of MOMPES'04, 1st International Workshop on Model-Based Methodologies for Pervasive and Embedded Software,* (pp. 15-23). Turku, Finland: TUCS.

Fernandes, J. E., Machado, R. J., & Carvalho, J. (2007). Model-Driven Software Development for Pervasive Information Systems Implementation. In *Proceedings of the 6th International*

Conference on Quality of Information and Communications Technology (pp. 218-222).Washington, DC: IEEE Computer Society.

Fernandes, J. E., Machado, R. J., & Carvalho, J. (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.

Fernandes, J. E., Machado, R. J., & Carvalho, J.. (2012a). A Case studies approach to the analysis of profiling and framing structures for pervasive information systems. *International Journal of Web Portals, 4*(2), 18.

Fernandes, J. E., Machado, R. J., & Carvalho, J. (2012b). Profiling and framing structures for pervasive information systems development. In G. Putnik & M. Cruz-Cunha (Eds.), *Virtual and Networked Organizations, Emergent Technologies and Tools* (vol. 248, pp. 283-293): Springer Berlin Heidelberg.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns - Elements of reusable object-oriented software*. Boston, MA: Addison-Wesley.

Gorton, Ian, & Liu, Yan. (2010). Advancing software architecture modeling for large scale heterogeneous systems. In *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, (pp. 143-148). New York, NY: ACM.

Hansmann, U., Merck, L., Nicklous, M.S., & Stober, Th. (2003). *Pervasive computing* (2nd ed.). New York, NY: Springer-Verlag Berlin Heidelberg.

Heijstek, W., & Chaudron, M. R. V. (2009). Empirical investigations of model Size, complexity and effort in a large scale, distributed model driven development process. In *Proceedings of the 2009 35th Euromicro Coonference on Software Engineering and Advanced Applications,* (pp. 113-120). Washington, DC: IEEE Computer Society.

HillSide. (2013). *The Hillside Group*. Retrieved from http://hillside.net/

Kay, R. (1969). The management and organization of large scale software development projects. In *Proceedings of the May 14-16, 1969, Spring Joint Computer Conference,* (pp. 425-433). New York, NY: ACM.

Laine, P. K. (2001). The role of SW architectures in solving fundamental problems in object-oriented development of large embedded SW systems. In *Proccedings of the Working IEEE/IFIP Conference on Software Architecture,* (pp. 14-23). Washington, DC: IEEE Computer Society.

Langheinrich, Marc, Coroama, Vlad, Bohn, Jürgen, & Rohs, Michael. (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, Kalle, & Yoo, Youngjin. (2002). Introduction [Issues and challenges in ubiquitous computing]. *Communications of ACM, 45*(12), 62-65.

Mattsson, A., Lundell, B., Lings, B., & Fitzgerald, B. (2007). Experiences from representing software architecture in a large industrial project using model driven development. In *Proceedings of the Second Workshop on SHAring and Reusing Architectural Knowledge Architecture, Rationale, and Design Intent* (pp. 6-6). Washington, DC: IEEE Computer Society.

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. In *Companion to the 19th Annual ACM SIGPLAN Conference on Object-oriented Programming Systems, Languages, and Applications*, (pp. 138-140). New York, NY: ACM.

Mirakhorli, M., Sharifloo, A., & Shams, F. (2008). Architectural challenges of ultra large scale systems. In *Proceedings of the 2Nd International Workshop on Ultra-large-scale Software-intensive Systems*, (pp. 45-48). New York, NY:ACM.

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

OMG (2005). *Object Management Group Home Page*. Retrieved from http://www.omg.org

OMG (2008). *SPEM v2.0 - Software & Systems Process Engineering Meta-Model Specification v2.0*. Retrieved from www.omg.org/spec/SPEM/2.0/.

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

Sage, Andrew P., & Rouse, William 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. In *Proceedings of the Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, (pp. 4-6). Washington, DC: IEEE Computer Society.

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.

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.