Fernandes, José Eduardo, Ricardo J. Machado, João Á. Carvalho, "Profiling and Framing Structures for Pervasive Information Systems Development" in Goran D. Putnik and Maria Manuela Cruz-Cunha (Eds.), Virtual and Networked Organizations, Emergent Technologies and Tools, Communications in Computer and Information Science Series, Vol. 248 (Revised Selected Papers from "ViNOrg 2011 - First International Conference on Virtual and Networked Organizations, Emergent Technologies and Tools", Ofir, Portugal, July 6-8, 2011), Springer-Verlag, Berlin Heidelberg, Germany, January, 2012, pp. 283-293.

Profiling and Framing Structures for Pervasive Information Systems Development

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, Dept. de Sistemas de Informação
4800-058 Guimarães, Portugal
{rmac, jac}@dsi.uminho.pt

Abstract. Pervasive computing is a research field of computing technology that aims to achieve a new computing paradigm. Software engineering has been, since its existence, subject of research and improvement in several areas of interest. 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. 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 a case study that allowed demonstrating the applicability of the approach.

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

1 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 [1, 2], 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" [3]. Therefore, Pervasive Information Systems (PIS) [4] 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).

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.

2 Related Work

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 [5]. 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 [6, 7], software architecture [8-10], model-driven development [11, 12], among others. Not directly related with large projects, Medvidovic [13] 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 [14]. 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.

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 [15] to bring the application of MDD concepts and techniques to software of PIS. Fernandes et al. [4] 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* [4] 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.

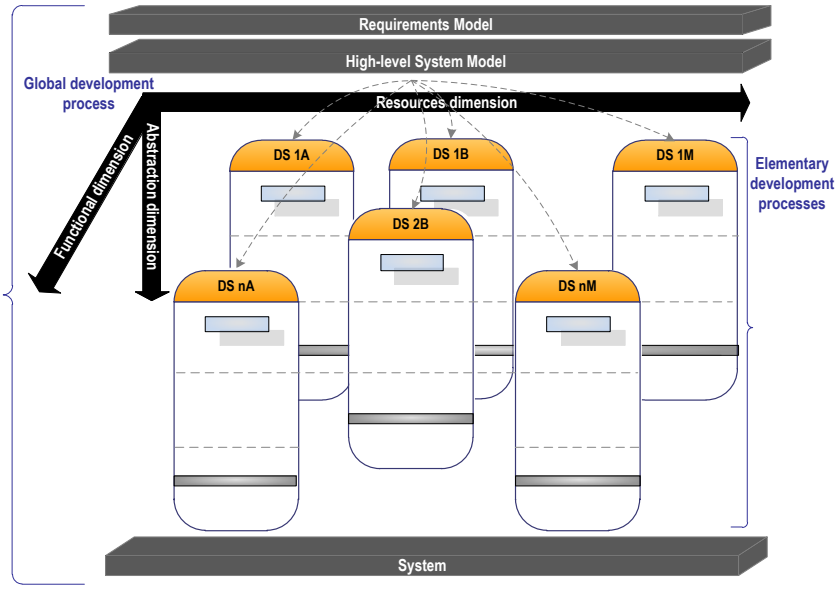

Fig. 1. Development framework for PIS.

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.

3 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.

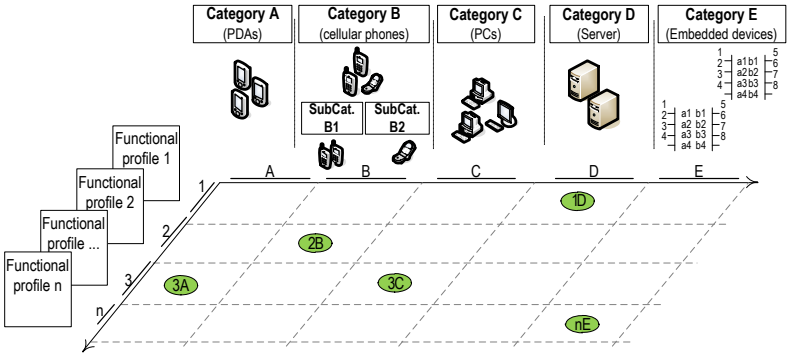

Fig. 2. Functional profile instances.

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. 5 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 tree 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.

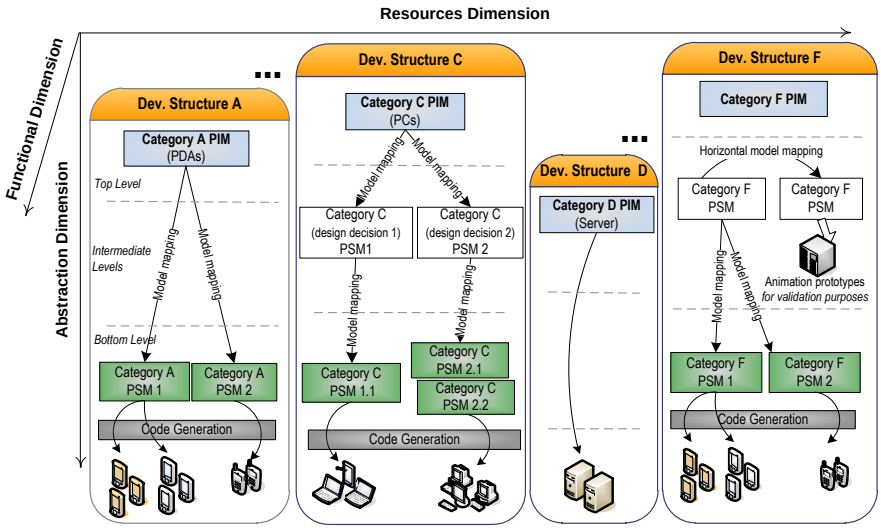

Fig. 3. Modeling levels in development structures (abstraction dimension).

Considering the schema of the development framework (Fig. 1) and the schemas related to functional profiles instantiation (Fig. 2 and Fig. 3 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 illustrates the high-level and low-level models/specifications/artifacts (produced by starting and ending activities of the global development process). 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. 2 and Fig. 3; for each of these development frameworks there will be a corresponding elementary development process (as depicted by Fig. 1).

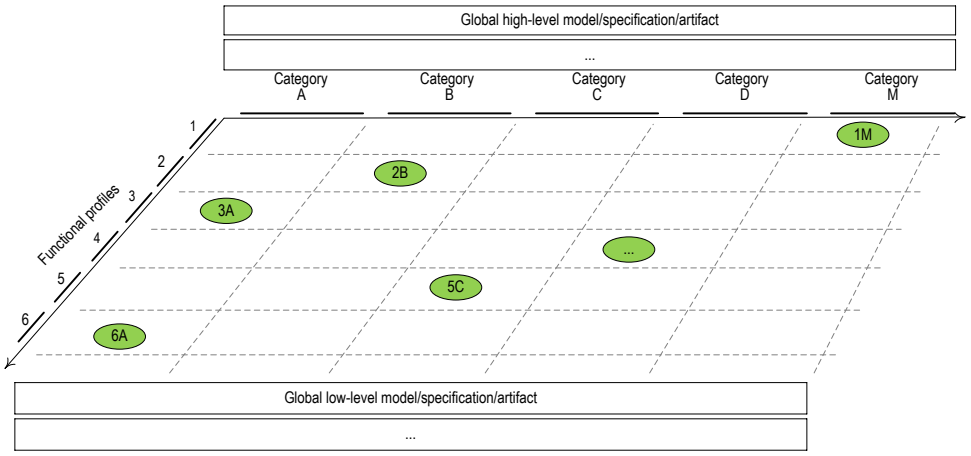

Fig. 4. Framing structure for a project.

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.

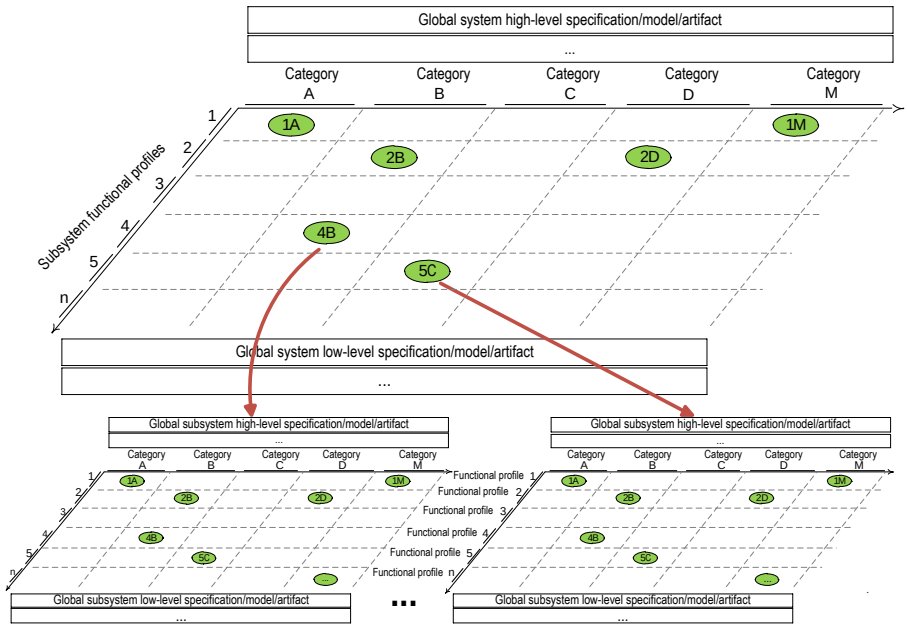

Fig. 5. Nesting of framing structures for large projects.

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.

4 The USE-ME-GOV Case Study

This section starts by briefly introducing the USE-ME.GOV (USability-drivEn open platform for MobilE GOVernment) project that aimed to create an open platform for mobile government services. Then, it illustrates the application of the development framework on this project. Attending to the project dimension and purpose/size of this paper, 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).

The USE-ME.GOV project [16] 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.

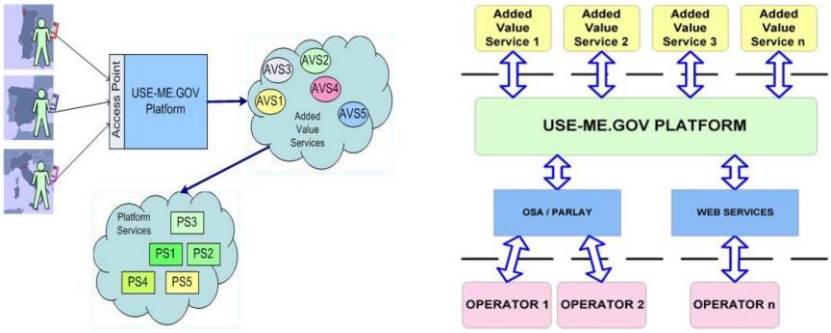

Fig. 6. USE-ME.GOV System General Architecture (from [17]).

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.

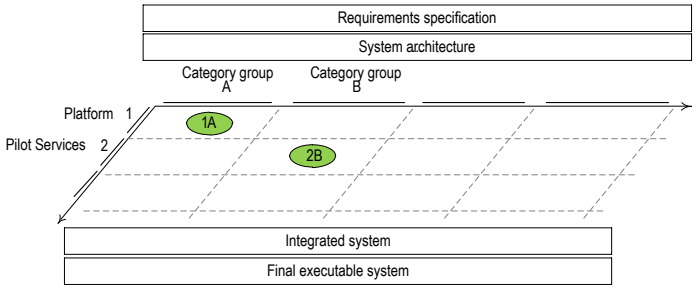

Fig. 7. Framing structure at system level for USE-ME.GOV project.

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.

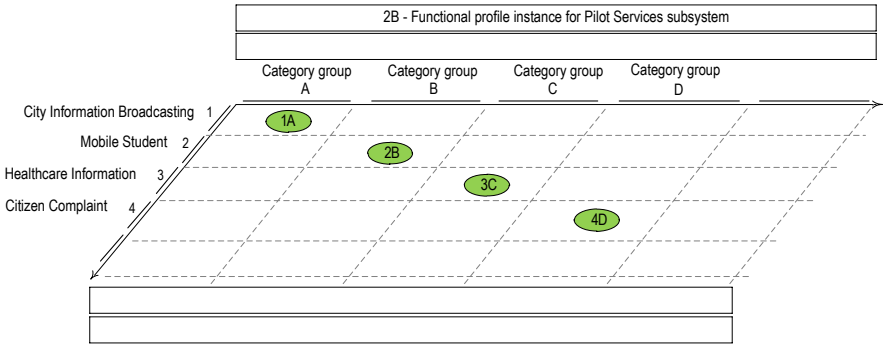

Fig. 8. Framing structure for Pilot Services subsystem of USE-ME.GOV.

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. 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.

5 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. 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. 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. This paper also introduces a case study that allows demonstrating this approach.

References

1. Weiser, M.: Some computer science issues in ubiquitous computing. Communications of ACM 36, 75-84 (1993)
2. Weiser, M., Gold, R., Brown, J.S.: The origins of ubiquitous computing research at PARC in the late 1980s. Ibm Systems Journal 38, 693-696 (1999)

3. Sage, A.P., Rouse, W.B.: Information Systems Frontiers in Knowledge Management. Information Systems Frontiers 1, 205-219 (1999)
4. Fernandes, J.E., Machado, R.J., Carvalho, J.Á.: Model-Driven Development for Pervasive Information Systems. In: Mostefaoui, S.K., Maamar, Z., Giaglis, G.M. (eds.) Advances in Ubiquitous Computing: Future Paradigms and Directions, pp. 45-82. IGI Publishing (2008)
5. Ruparelia, N.B.: Software development lifecycle models. SIGSOFT Softw. Eng. Notes 35, 8-13 (2010)
6. Kay, R.H.: The management and organization of large scale software development projects. Proceedings of the May 14-16, 1969, spring joint computer conference, pp. 425-433. ACM, Boston, Massachusetts (1969)
7. Benincasa, G.P., Daneels, A., Heymans, P., Serre, C.: Engineering a Large Application Software Project: The Controls of the CERN PS Accelerator Complex. Nuclear Science, IEEE Transactions on 32, 2029-2031 (1985)
8. Gorton, I., Liu, Y.: Advancing software architecture modeling for large scale heterogeneous systems. Proceedings of the FSE/SDP workshop, FoSER 2010, pp. 143-148. ACM, Santa Fe, New Mexico, USA (2010)
9. Mirakhorli, M., Sharifloo, A.A., Shams, F.: Architectural challenges of ultra large scale systems. Proceedings of the 2nd international workshop on Ultra-large-scale software-intensive systems, pp. 45-48. ACM, Leipzig, Germany (2008)
10. Laine, P.K.: The role of SW architecture in solving fundamental problems in object-oriented development of large embedded SW systems. Software Architecture, 2001. Proceedings. Working IEEE/IFIP Conference on, pp. 14-23 (2001)
11. Mattsson, A., Lundell, B., Lings, B., Fitzgerald, B.: Experiences from Representing Software Architecture in a Large Industrial Project Using Model Driven Development. Proceedings of SHARK-ADI'07. IEEE Computer Society (2007)
12. Heijstek, W., Chaudron, M.R.V.: Empirical Investigations of Model Size, Complexity and Effort in a Large Scale, Distributed Model Driven Development Process. SEAA '09, pp. 113-120 (2009)
13. Medvidovic, N.: Software architectures and embedded systems: a match made in heaven? Software, IEEE 22, 83-86 (2005)
14. Fernandes, J.E., Machado, R.J., Carvalho, J.Á.: Model-Driven Methodologies for Pervasive Information Systems Development. In: Fernandes, J.M., Machado, R.J., Lilius, J., Porres, I. (eds.) MOMPES'2004, pp. 15-23. TUCS General Publication, Hamilton, Ontario, Canada (2004)
15. Fernandes, J.E., Machado, R.J., Carvalho, J.Á.: Model-Driven Software Development for Pervasive Information Systems Implementation. In: Machado, R.J., Abreu, F.B.e., Cunha, P.R.d. (eds.) QUATIC 2007 - pp. 218-222. IEEE Computer Society, Lisbon, Portugal (2007)
16. USE-ME.GOV: Consortium Agreement - Annex I - Description of Work. (2003)
17. USE-ME.GOV: D3.1 Recommendations. (2006)