UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

ISSN 0103-2569

REFERENCE ARCHITECTURE AND PRODUCT LINE ARCHITECTURE: A COMPARISON

ELISA YUMI NAKAGAWA
PABLO OLIVEIRA ANTONINO
MARTIN BECKER

Nº 401

RELATÓRIOS TÉCNICOS

São Carlos – SP
Ago./2014

Instituto de Ciências Matemáticas e de Computação

ISSN - 0103-2569

Reference Architecture and Product Line Architecture: A Comparison

Elisa Yumi Nakagawa
Pablo Oliveira Antonino
Martin Becker

Nº 401

ICMC - TECHNICAL REPORT

São Carlos, SP, Brazil
September/2014

Abstract

Currently, the size and complexity of software systems, as well as critical time to market, demand new approaches from Software Engineering discipline for building such systems. In this context, the use of reference architectures and product line architectures is becoming a common practice. However, both of these concepts are sometimes mistakenly seen as the same thing; it is also not clearly established how they can be explored in a complementary way in order to contribute to software development. In this perspective, this document makes a clear differentiation of these architectures, by highlighting basic questions like definitions, benefits, and motivation for using each one, when and how they should be used, built, and evolved, as well as stakeholders involved and benefited by each one. As a result, a better understanding of both reference architectures and product line architectures can contribute to promote a more effective reuse in the development of software systems.

Contents

Chapter

1

Introduction

In the face of increasing complexity, diversity, scope, and size of software systems, as well as the necessity of dynamic integration and adaptation of systems and the challenges encountered by managing families of systems, new ways to facilitate the development and evolution of such systems are required. Software Engineering has proposed a number of different ways for that. In particular, software architectures have been increasingly considered as the main artifact that plays a pivotal role in determining system quality, since they form the backbone of any successful software-intensive system (Kruchten et al., 2006; Shaw e Clements, 2006). Motivated by that, the research area of Software Architecture has grown up and has accumulated important knowledge that has contributed to facilitating the achievement of software quality aspects. In this context, two special types of software architecture can be found: Reference Architecture and Product Line Architecture. Both aim at improving software system development by standardizing the architectures of a set of software systems.

In general, a reference architecture has been considered as a structure that, besides aggregating behavior, is the basis from which the software architectures of systems of a given domain are built. Considering the relevance of reference architectures, diverse application domains have proposed and used these architectures. Good examples of domains are automotive (AUTOSAR, 2014), software engineering (Nakagawa et al., 2011b; Oliveira e Nakagawa, 2011), ambient assisted living (OASIS Project, 2014; UniversAAL Project, 2014), and computer games (Delmas et al., 2007).

In parallel, product line architectures have been used in the Software Product Line (SPL) approach to enable reuse in the large. This approach aims to efficiently derive specific products for a given domain based on reusable core assets that have common features and variable parts (Clements e Northrop, 2002). In this context, the product line architecture refers to a structure that also encompasses the behavior from which software products are developed. It is important to highlight that more and more organizations are adopting SPL and, as a consequence, product line architectures, as a solution to improve time to market, productivity, flexibility, and mass customization needs (Clements e Northrop, 2002; Gomaa, 2004; Pohl et al., 2005; Software Engineering Institute, 2014a).

Despite the fact that reference architectures and product line architectures have been widely discussed and used, there exists no consensus within the software architecture community about the effective relationship between them. Besides that, since they seem to be similar, they are sometimes considered to be the same thing. It is also not clear how these architectures can be explored together in order to make an effective contribution to the software development. Moreover, understanding them well seems to be relevant for the establishment and evolution of such architectures.

The main objective of this document is to make a clear distinction between reference architecture and product line architecture. For this, we highlight basic points: definitions of reference architecture and product line architecture, benefits and motivation to use them, when and how they should be used, built, and evolved, as well as the stakeholders involved and benefited by them. We based our work on the more influential, recent works in the software architecture literature, as well as on our experience in proposing, using, and managing these architectures. As main results, we have observed that, despite common characteristics between them, they present a subtle but critical difference. Considering the particular characteristics of each one, we also propose the use of reference architectures as a basis for product line architectures.

The remainder of this document is organized as follows. Chapter 2, we present definitions, purposes, context, and uses of reference architecture and product line architecture. In Chapter 3, we present a perspective about the relationship between reference architecture and product line architecture. In Chapter 4, we summarize our conclusions.

Chapter

2

Discussing Reference Architecture and Product Line Architecture

Reference architecture is a more and more recurrent research topic both in academy and industry. In spite of general understanding about what reference architectures are, there exists no consensus about a definition to these architectures, their benefits and motivations, when and how to manage them, and who are the stakeholders of these architectures. Otherwise, regarding product line architecture, we have observed that these issues — definition, benefits, motivation, management, and stakeholders — are more consolidated. A discussion about definition of reference architecture and product line architecture is presented in (Nakagawa et al., 2011a). Following, we discuss about both architectures.

2.1 Reference Architecture

Both academic and software industry have currently explored reference architectures as basis to the development of software systems. Initiatives can be found to the establishment, representation, evaluation, maintenance, and use of reference architectures. Initiatives in order to understand what these architectures are have been also conducted (Angelov et al., 2009; Cloutier et al., 2010; Muller, 2012). Despite these works, there is not a widely accepted definition of reference architecture and its impact in the software development

process. Following, we discuss then issues related to definitions, benefits, motivations, management, and stakeholders of reference architectures.

What is Reference Architecture? A study of the main works in the software architecture literature allow us to identify some important definitions for reference architecture. One of the first ones was stated by Kruchten (Kruchten, 2000), who said that "A reference architecture is, in essence, a predefined architectural pattern, or set of patterns, possibly partially or completely instantiated, designed, and proven for use in particular business and technical contexts, together with supporting artifacts to enable their use. Often, these artifacts are harvested from previous projects." Another definition was proposed by Bass et al. (Bass et al., 2003) and states that "A reference architecture is a reference model mapped onto software elements (that cooperatively implement the functionality defined in the reference model) and the data flows between them." A reference model can be considered as an abstract framework that presents a minimal set of unifying concepts, axioms, and relationships within a particular problem domain, independent of specific standards, technologies, implementations, or other concrete details. From the perspective of a specific domain, Rosen et al. (Rosen et al., 2007) said that "A reference architecture is a working example of a critical aspect of your enterprise architecture, such as (...) how to work with your organization's message bus or (...) how to work with your business rules engine." In the same perspective, Angelov et al. (Angelov et al., 2009) stated that "A reference architecture is a generic architecture for a class of information systems that is used as a foundation for the design of concrete architectures from this class." To complete these definitions, Reed (Reed, 2002) thinks that "A reference architecture consists of information accessible to all project team members that provides a consistent set of architectural best practices." Even though syntactically different, these definitions present the same essence: the reuse of knowledge about software development in a given domain, in particular with regard to architectural design. Other definitions are also found, but the ones presented above are sufficient for our purpose. Based on these works, we can say that a reference architecture refers to an architecture that encompasses the knowledge about how to design concrete architectures of systems of a given application domain; therefore, it must address the business rules, architectural styles (sometimes also defined as architectural patterns that address quality attributes in the reference architecture), best practices of software development (for instance, architectural decisions, domain constraints, legislation, and standards), and the software elements that support development of systems for that domain. All of this must be supported by a unified, unambiguous, and widely understood domain terminology.

Why to Use? A reference architecture has three basic roles, as also stated in (Gallagher, 2000; Reed, 2002): (i) it generalizes and extracts common functions and configurations of systems of a given domain; (ii) it must be a blueprint for future architectural instantiations; and (iii) it is a basis that supports software projects from the beginning to the end of the projects. It is important to highlight that it captures the essence of the architectures of a collection of systems of a given domain; thus, they can be used as a knowledge repository of that domain from which systems are developed and maintained. As a consequence, reuse can be done more effectively, resulting in more reliably, productivity, and cost reduction. Moreover, Reed (Reed, 2002) argues that a software project that proceeds without reference architecture necessarily will not fail, but it will require much more effort that could be spent better in other parts of the project.

When to Adopt? Reference architectures are adopted when it is mainly intended to reuse knowledge contained in such architectures in order to develop and evolve software systems. Besides that, these architectures could be adopted when it is intended to develop more standardized software systems, since reference architectures provide standardized solution. In another perspective, reference architectures are also used in software development processes, in particular, when designing the software architecture of a given system. For instance, considering RUP, reference architectures could be used in the elaboration phase where the project starts to take shape and the description of the software architecture is created.

How to Create? Considering the relevance of reference architectures, some works have focused on the design of reference architectures. Muller (Muller, 2012) has proposed recommendations in order to create and maintain reference architectures; shortly, references architecture must be understandable, up-to-date, and maintainable. Angelov et al. (Angelov et al., 2012) presented a framework for analyzing and designing reference architectures. Dobrica and Niemela (Dobrica e Niemela, 2008) proposed an approach for reference architecture design for the embedded systems domain. Galster et al. (Galster et al., 2011) proposed a six-step procedure for reference architecture design, including the decision on what type of reference architecture to design, the selection of a design strategy, the empirical acquisition of data, the construction of the reference architecture, enriching the reference architecture with variability, and the evaluation of the reference architecture. Besides these works, Nakagawa et al. (Nakagawa et al., 2014) present a process that systematizes the design, representation, and evaluation of reference architectures.

How to Use? Different software development processes present different perspectives with respect to the use of required assets for building software systems. Reference archi-

tecture has also been inserted as an asset into these processes. Each one presents different way to insert and use reference architecture to support the software development. One hand, in the context of agile processes (also agile methods or methodologies), in general, their adepts consider code as the unique necessary artifact. However, initiative to use reference architecture in the agile context is already found, as summarized in (Zani et al., 2011). For instance, Linden (van der Linden et al., 2003) has explored reference architecture together with XP (eXtreme Programming) to implement a medical system. In another perspective, an initiative suggests that in RUP, the reference architecture should be consulted as part of the "Architectural Analysis" activity, in the Inception and Elaboration phases (Reed, 2002); however, it is observed that tasks and guidelines to perform it are not still available in RUP.

When to Evolve? Independently of the application domain, knowledge about the domain evolves, since new elements, such as processes, methods, and activities are continually proposed and adopted. Therefore, since reference architecture must encompass this knowledge, it must be continually evolved, for not becoming deprecated. It is worth highlighting that evolution of a reference architecture is not a trivial task, since it must encompass the knowledge of a whole domain that it intends to represent. However, evolution have not been widely discussed in the literature. An initiative suggests that, although reference architecture is not currently part of RUP (Rational Unified Process) (Kruchten, 2000), reference architectures could be evolved in the activity "Prepare for Project Close-Out", found in the Transition phase of RUP (Reed, 2002); however, tasks and guidelines to perform it are not defined yet.

Who is Affected? Reference architectures are used by or affect multiple stakeholders, such as software engineers, developers, business managers, customers, and marketing staff. Each one has different perspective and competence on the reference architecture; thus, different architectural views to represent this architecture are required. However, there is not a well-defined group of stakeholders for reference architectures, because they depends on each particular architecture, as well as its goals and its range. In this scenario, we can identify two major groups: management stakeholders and technical stakeholders. In particular, management stakeholders are interested in knowing about cost for building and maintaining a reference architecture, as well as how the reference architecture minimizes the outcomes and maximizes the incomes of the company. Information about time, effort, and risk reduction using this architecture are also required by these stakeholders. According to Muller and Hole(Muller, 2012) and Cloutier et al. (Cloutier et al., 2010), sponsorship of business managers for reference architectures is a prerequisite. Such sponsorship works only if the reference architecture provides value for the business, for

example, as a decision making tool for business leaders. For technical stakeholders, such as software architects, developers, and testers, according to Reed (Reed, 2002), reference architecture is an element that aggregates architectural decisions, for instance, which pattern is the most appropriated to be used when there is a need to separate externalized information from its internal state. Thus, this architecture also support technical stakeholders, aiming at minimizing discussion about making decisions. Furthermore, supported by architectural team and having coherent inputs, the lead software architect is primarily responsible to manage construction and use of the reference architecture.

2.2 Product Line Architecture

In the SPL literature, there is a consensus that the system development based on instances of a common product line architecture can help to exploit the reuse potential and related benefits, such as quality increase, cost reduction, and time to market decrease in a very large extension (Clements e Northrop, 2002; Gomaa, 2004; Pohl et al., 2005). We summarize then what software product architecture is, its benefits and motivations, when and how to manage it, and who are its stakeholders.

What is Product Line Architecture? Various synonyms are used for the term "product line architecture", such as software product line architecture, domain-specific software architecture, domain architecture, and even reference architecture in (DeBaud et al., 1998; Pohl et al., 2005). However, it seems that "product line architecture" is a term that fits best in the SPL context. In parallel, there are also different definitions of this term. DeBaud et al. (DeBaud et al., 1998) say that it is an architecture with a required degree of flexibility and is shared by all the members of a product line, ensuring their conceptual integrity. According to Pohl et al. (Pohl et al., 2005), "product line architecture is a core architecture that captures the high level design for the products of the SPL, including the variation points and variants documented in the variability model." In the same perspective, Gomaa (Gomaa, 2004) stated that "product line architecture is an architecture for a family of products, which describes the kernel, optional, and variable components in the SPL, and their interconnections." In a more complete definition, SEI (SEI, 2014) declares that "The product line architecture is an early and prominent member in the collection of core assets. (...) The architecture defines the set of software components (...) that populates the core asset base. The product line architecture – together with the production plan – provides the prescription (...) for how products are built from core assets". It is worth highlighting that, in general, others definitions are derivations of those presented herein. Moreover, it is important to observe that the common essence

in these definitions is "variability on a common core". Therefore, we can define product line architecture as a special type of software architecture used to build a product line; it explicitly describes commonality and variability and is the basis for the architectures of all product line members.

Why to Use? The literature has presented several successful uses of product line architectures in the SPL context (Clements e Northrop, 2002; Gomaa, 2004; Pohl et al., 2005; Software Engineering Institute, 2014b). By identifying what is common and what is different among software products, the product line architecture can promote a more precise decomposition of a set of systems into manageable pieces. As a direct consequence, a better Separation of Concerns (SoC) both in the SPL common infrastructure and in the software products could be achieved. As also stated initially by Dijkstra (Dijkstra, 1976) and valid until nowadays, an adequate SoC contributes to the quality of software systems. Therefore, the product line architectures will surely collaborate to the quality of the SPL common infrastructure, as well as the software products resulted of this infrastructure. The three different perspectives regarding software quality – developer, organization, and consumer – can be achieved with these architectures. From developer's perspective, these architectures are ideal for promoting reuse and productivity (Clements e Northrop, 2002). From organization's perspective, system development companies have considerable benefits, mainly reduction of development and maintenance costs/efforts, enhancement of quality, and reduction of time to market (Pohl et al., 2005). From consumer's perspective, they can see possible customizations and are certainly benefited by quality issues promoted by product line architecture. Finally, these architectures make possible also communication to and among all stakeholders.

When to Adopt? A product line architecture, together with other SPL artifacts, such as variability model and requirement specification, is adopted in situations where there is a need of developing different variants of a product. In particular, a product line architecture should be adopted when variability and commonality management among similar products are required.

How to Create? In essence, product line architecture is created and used as an essential backbone of an SPL. Different SPL methodologies, such as PuLSE (Product Line Software Engineering) (Bayer et al., 1999; Fraunhofer IESE, 2011) and Pohl's approach (Pohl et al., 2005), deal with the product line architecture in their own way. For instance, PuLSE, a methodology that covers all aspects of product line development and evolution, has a component — the PuLSE-DSSA (Domain-Specific Software Architecture) (DeBaud et al., 1998) — for specifically creating product line architectures.

How to Use? Once created and available, the product line architecture works as a bridge between product line scope and product line implementation. It is basically used to build concrete architectures of software product of an SPL, specifically, during the application design in the application engineering phase.

When to Evolve? Regarding evolution of product line architecture, it must be evolved when it is required the development of new products with new requirements that have not been considered previously in the SPL. However, the evolution of product line architectures is more complex than architectures of single software system, since new and even conflicting requirements originate from the existing products and new products to be built and they must be incorporated in the architecture of the SPL.

Who is Affected? Different stakeholders are involved in the SPL life cycle. Management and technical stakeholders, as well as the consumers, are affected by the product line architecture. Management stakeholders are interested in knowing about time, effort, and risk for adopting SPL and, as consequence, a product line architecture. Furthermore, reduction of the outcomes and rise of the incomes of the company are also interest of these stakeholders. For technical stakeholders, this architecture helps to make assumptions about the architectural context in which core assets will be reused, highlighting the essential product line concepts and suppressing non-essential ones. For application designers, it is used as a common asset that are customized to yield individual product architectures.

Chapter

3

Relationship Between Reference Architecture and Product Line Architecture

As result of our investigation, we can say that while reference architectures should deal with the range of knowledge of an application domain, providing standardized solutions for a broader domain, product line architectures are more specialized, focusing sometimes in a specific subset of software systems of a domain and providing standardized solutions for a smaller family of systems. Moreover, in general, reference architectures are used to standardize the external structure of the systems, i.e., the interfaces and protocols, treating the involved subsystems as black boxes. In product line architecture, where the focus is on building systems from a common set of artifacts, the internal structure is standardized, aiming at reusing existing components that are treated as white boxes. Thus, reference architectures focus on the external standardization (the what), while product line architectures also address the internal standardization (the how).

Another essential difference is that product line architectures contain the variabilities, i.e., information about how the individual product varies in an SPL. Otherwise, currently reference architectures contain what is common in a specific domain, i.e., they contain elements presented in all systems developed from those architectures. For instance, in a reference architecture for e-commerce systems, it is expected that it presents a module

and related functionalities that, for instance, manage the shopping cart. In another perspective, in a recent work (Nakagawa, 2012), the relevance of considering variability in reference architectures was discussed; however, it is not still a current approach widely adopted for the several existing reference architectures.

We can point out that, in general, reference architectures are in a higher level of abstraction if compared to product line architecture and, therefore, they could be basis of product line architectures. In other words, a reference architecture can result in different specializations (i.e., product line architectures), each one for a specific SPL, as illustrated in Figure 3.1. For instance, considering again the reference architecture for e-commerce systems, we can specialize this architecture for a product line architecture of an SPL for bookshops or for online auction. From these specializations, architectural instances (i.e., architectures of specific products) are built. Therefore, reference architectures could be basis of both product line architectures and architectural instances. In (Nakagawa et al., 2013), it is presented a process that systematizes the use of existing reference architectures in the context of SPL; in particular, when building the PLA. Moreover, in (Nakagawa e Oquendo, 2013), we discuss how reference architectures can be explore in the context of Multi Software Product Line (MSPL or simply Multi Product Line - MPL). In brief, MPL has recently emerged as a new approach to develop software systems, mainly those large, complex ones. This approach can be also investigated to the development of Systems-of-Systems (SoS), i.e., a new class of software systems that are resulted by the integration of several operationally independent systems.

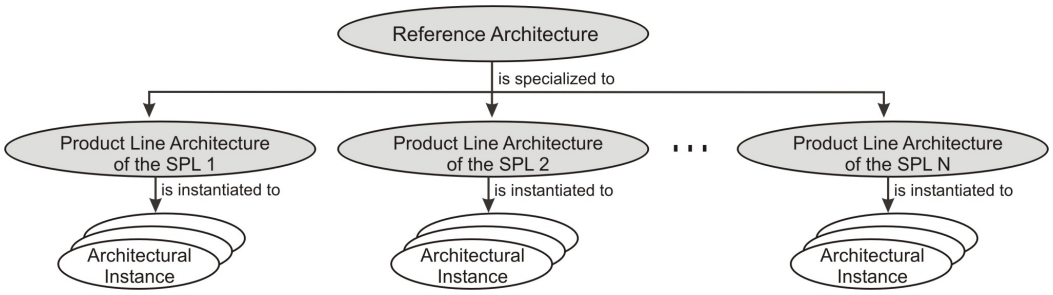

Figure 3.1: Relationship between Reference Architecture and Product Line Architecture

Chapter

4

Conclusion

The contribution of this document is a better understanding what differentiates reference architectures and product line architectures. The main difference is that reference architectures are generally on a higher level of abstraction compared to product line architectures. We have therefore proposed to investigate reference architectures as a basis for product line architectures, pursuing the complementary use of both architectures. We intend that this idea promotes the reuse of the domain knowledge contained in the reference architectures and, as a consequence, reduction in time and effort and increase in productivity as well, when establishing an SPL by using such reference architectures.

Acknowledgments: This work is supported by the Brazilian funding agencies FAPESP, CNPq, and CAPES and partially supported by the OptimAAL-Project (Competence Platform for the development and introduction of AAL-Solutions), funded by the German Federal Ministry of Education and Research (BMBF), and the CESAR-Project (Cost-efficient methods and processes for safety relevant embedded systems), funded by the European ARTEMIS Joint Undertaking.

References

Angelov, S.; Grefen, P.; Greefhorst, D. A classification of software reference architectures: Analyzing their success and effectiveness. In: *8th Working IEEE/IFIP Conference on Software Architecture (WICSA'09)*, Cambridge, UK, 2009, p. 141–150.

Angelov, S.; Grefen, P.; Greefhorst, D. A framework for analysis and design of software reference architectures. *Information and Software Technology*, v. 54, n. 4, p. 417–431, 2012.

AUTOSAR AUTomotive Open System ARchitecture). [*On-line*], in `http://www.autosar.org/` (Access in 02/10/2014), 2014.

Bass, L.; Clements, P.; Kazman, R. *Software architecture in practice*. The SEI Series in Software Engineering, 2 ed. Boston: Addison-Wesley, 2003.

Bayer, J.; Flege, O.; Knauber, P.; Laqua, R.; Muthig, D.; Schmid, K.; Widen, T.; DeBaud, J.-M. PuLSE: A methodology to develop software product lines. In: *Proc. of the 5th Symposium on Software Reusability (SSR'99)*, Los Angeles, USA, 1999, p. 223–234.

Clements, P.; Northrop, L. *Software product lines: Practices and patterns*. The SEI Series in Software Engineering, 1 ed. Boston, MA: Addison-Wesley, 608 p., 2002.

Cloutier, R.; Muller, G.; Verma, D.; Nilchiani, R.; Hole, E.; Bone, M. The concept of reference architectures. *Systems Engineering*, v. 13, n. 1, p. 14–27, 2010.

DeBaud, J.-M.; Flege, O.; Knauber, P. Pulse-dssa - a method for the development of software reference architectures. In: *Proceedings of the Third International Workshop on Software architecture (ISA'98)*, New York, NY, USA: ACM, 1998, p. 25–28.

Delmas, G.; Champagnat, R.; Augeraud, M. Plot monitoring for interactive narrative games. In: *Proceedings of the international conference on Advances in computer entertainment technology (ACE'2007)*, Salzburg, Austria: ACM, 2007, p. 17–20.

Dijkstra, E. W. *A discipline of programming.* Prentice Hall, 1976.

Dobrica, L.; Niemela, E. An approach to reference architecture design for different domains of embedded systems. In: *International Conference on Software Engineering Research and Practice (SERP'2008)*, Las Vegas, USA, 2008, p. 287–293.

Fraunhofer IESE Fraunhofer PuLSE - Product Line Software Engineering. (Internal documentation), 2011.

Gallagher, B. P. *Using the architecture tradeoff analysis method to evaluate a reference architecture: A case study.* Relatório Técnico CMU/SEI-2000-TN-007, Software Engineering Institute, 2000.

Galster, M.; Avgeriou, P.; Weyns, D.; Mannisto, T. Empirically-grounded reference architectures: A proposal. In: *7th ACM Sigsoft International Conference on the Quality of Software Architectures (QoSA'2011)*, Boulder, USA, 2011, p. 153–157.

Gomaa, H. *Designing software product lines with uml.* Object Technology Series. Addison-Wesley, 2004.

Kruchten, P. *The rational unified process: An introduction.* The Addison-Wesley Object Technology Series, 2 ed. Addison-Wesley, 2000.

Kruchten, P.; Obbink, H.; Stafford, J. The past, present, and future for software architecture. *IEEE Software*, v. 23, n. 2, p. 22–30, 2006.

van der Linden, H.; Boers, G.; Tange, H.; Talmon, J.; Hasman, A. PropeR: a multi disciplinary EPR system. *International Journal of Medical Informatics*, v. 70, n. 2–3, p. 149–160, 2003.

Muller, G. A reference architecture Primer. [*On-line*], *World Wide Web*, in `http://www.gaudisite.nl/` (Access in 02/10/2014), 2012.

Nakagawa, E. Y. Reference architectures and variability: Current status and future perspectives. In: *International Workshop on Variability in Software Architecture (VARSA'2012) at the Joint 10th Working IEEE/IFIP Conference on Software Architecture & 6th European Conference on Software Architecture (WICSA/ECSA'2012)*, Helsinki, Finland, 2012, p. 159–162.

Nakagawa, E. Y.; Antonino, P. O.; Becker, M. Reference architecture and product line architecture: A subtle but critical difference. In: *5th European Conference on Software Architecture (ECSA'2011), LNCS 6903*, Essen, Germany, 2011a, p. 207–211.

Nakagawa, E. Y.; Becker, M.; Maldonado, J. C. Towards a process to design product line architectures based on reference architectures. In: *17th International Software Product Line Conference (SPLC'2013)*, Tokio, Japan, 2013, p. 157–161.

Nakagawa, E. Y.; Ferrari, F. C.; Sasaki, M. M.; Maldonado, J. C. An aspect-oriented reference architecture for software engineering environments. *Journal of Systems and Software*, v. 84, n. 10, p. 1670–1684, 2011b.

Nakagawa, E. Y.; Guessi, M.; Feitosa, F.; Oquendo, F.; Maldonado, J. C. Consolidating a process for the design, representation, and evaluation of reference architectures. In: *11th Working IEEE/IFIP Conf. on Software Architecture (WICSA'2014)*, Sydney, Australia, 2014, p. 143–152.

Nakagawa, E. Y.; Oquendo, F. Perspectives and challenges of reference architectures in multi software product line. In: *Int. Workshop on Multi Product Line Engineering (MultiPLE'2013)*, Tokio, Japan, 2013, p. 100–103.

OASIS Project OASIS: quality of life for the elderly. [*On-line*], *World Wide Web*, available in: `http://www.oasis-project.eu/` (Access in 03/03/2014), 2014.

Oliveira, L. B. R.; Nakagawa, E. Y. A service-oriented reference architecture for the software testing domain. In: *5th European Conference on Software Architecture (ECSA'2011) (LNCS: 6903)*, Essen, Germany, 2011, p. 405–421.

Pohl, K.; Böckle, G.; van der Linden, F. *Software product line engineering: Foundations, principles, and techniques.* Berlin: Springer-Verlag, 2005.

Reed, P. Reference architecture: The best of best practices. [*On-line*], *World Wide Web*, `http://www.ibm.com/developerworks/rational/library/2774.html` (Access in 03/03/2014), 2002.

Rosen, M.; Ambler, S. W.; Hazra, T. K.; Ulrich, W.; Watson, J. Enterprise architecture trends. *Enterprise Architecture*, v. 10, n. 1, cutter Consortium, 2007.

SEI A framework for software product line practice, version 5.0. [*On-line*], *World Wide Web*, available in `http://www.sei.cmu.edu/productlines/frame_report/index.html` (Access 03/03/2014), 2014.

Shaw, M.; Clements, P. The golden age of software architecture. *IEEE Software*, v. 23, n. 2, p. 31–39, 2006.

Software Engineering Institute Product line hall of fame. [*On-line*], *World Wide Web*, available in `http://splc.net/fame.html` (Access in 03/03/2014), 2014a.

Software Engineering Institute Software product line: Case studies. [*On-line*], *World Wide Web*, available in `http://www.sei.cmu.edu/productlines/casestudies/index.cfm` (Access in 03/03/2014), 2014b.

UniversAAL Project The UniversAAL Reference Architecture. [*On-line*], *World Wide Web*, in `http://www.universaal.org/` (Access in 02/10/2014), 2014.

Zani, V. A. T.; Feitosa, D.; Nakagawa, E. Y. Current state of reference architectures in the context of agile methodologies. In: *Proc. of the 23th International Conference on Software Engineering and Knowledge Engineering (SEKE'11)*, Miami Beach, USA, 2011, p. 590–595.