(http://alarcos.esi.uclm.es/ginseng2016/home)

GInSEng 2016

2 nd Green in Software Engineering Workshop

29 th August 2016, Amsterdan

ICT4S
ICT for Sustainability

(http://ict4s.org/)

Agenda

Ginseng Tentative Program	
Time	**Activity**
8:15 - 9:00	Registration Workshop
8:30 - 9:00	Welcome with refreshment
9:00 - 10:30	Invited Speech: Robert Decker
10:30 - 11:00	**Coffee break**
11:00 - 11:25	**Title:** GreenITAudit: A Tool to Audit the Green IT. Authors: J. David Patón-Romero and Mario Piattini.
11:25 - 11:50	**Title:** How to cast the approaches on green software engineering upon t Authors: Eva Kern.
11:50 - 12:15	**Title:** Static Energy Consumption Analysis in Variability Systems. Authors: Marco Couto, Jácome Cunha, Joao Fernandes, Rui Pereira an
12:30 - 13:30	**Lunch break**
13:30 - 13:55	**Title:** In search of energy efficient architectural patterns. Authors: Gianantonio Me and Coral Calero.
13:55 - 14:20	**Title:** Five years of green in software engineering: the Number of the Be Author: Coral Calero.
14:20 - 14:35	Warm up: Converging Research and Practical Green IN Software Engin
14:35 - 16:00	Working groups – 15:00 Tea break
16:00 - 17:00	Wrap up
17:00	Closing

Last updated: **July 26, 2016**

In search of energy efficient architectural patterns

Gianantonio Me
Faculty of Science, Vrije Universiteit, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
Universidad de Castilla-La Mancha, Paseo de la Universidad 4, 13071 Ciudad Real, Spain
g.me@vu.nl gianantonio.me@uclm.es

Coral Calero
University of Castilla-La Mancha, Department of Technologies and Information Systems, ALARCOS Research Group, Paseo de la Universidad, 4 13071 Ciudad Real, Spain
coral.calero@uclm.es

***Abstract*— Nowadays software pervasively support human life. That massive software presence poses environmental challenges, due to the increasing resources consumption. Sustainability becomes a major concern for software engineering too. Software engineers should deal with sustainable aspects and green quality attributes. Software Architecture is an interesting domain where to introduce green design decisions. Indeed, in the specific context of software architecting, we primarily deal with quality issues. Quality attributes should be early addressed in the design of systems. Therefore the main challenge for architects is to evaluate the architecture fitness respect to competing quality attributes. This work specifically focuses on Energy Efficiency in the context of architectural patterns. Due to the lack of evidence so far, we started from the assumption (literature-based) that Maintainability is a quality attribute that shows an implication with Energy Efficiency. We outline some hypothesis about potential relationship between Maintainability and Energy Efficiency in the context of architectural patterns. This work represents an attempt on assess how established knowledge on patterns selection might be challenged and eventually re-shaped by introducing in the architecture evaluation process sustainable and green quality dimensions.**

***Index Terms*—Sustainability, Software Architecture, Architecture Patterns, Quality Attributes, Green Software Engineering.**

I. Introduction

Sustainability is a major issue for our society. One aspect of implementing sustainability is the need of cutting down energy consumption trends, which requires new technical solutions. Information Technology (IT) is on the front-line of this perspective. Indeed IT can support green behaviors (*Green by IT*) or sustainability can be intrinsic in the process of building or running IT artifacts (*Green in IT*) [1]. This work focuses on a specific IT artifact, software, and in particular on the context of software architecture. Indeed, software itself is responsible of energy consumption trends and we are interested in increasing the knowledge on how build up green software. Software Energy Efficiency (EE) would reduce the usage of electrical energy by software artifacts [2]. Although Energy Efficiency does not entirely represent sustainability, its specifications and characteristics make this quality attribute as a good starting point for bringing green issues in software engineering.

Software architecture builds the basis for quality software systems. Defining this term is a potential dangerous activity and it has been surrounded by a long debate (3). For instance, some definitions identify software architecture in its structural shape (components and connectors) and the connections that coordinate the activities of those components [4]. Other definitions go beyond structural elements such as algorithms, responsibilities of design elements, protocol communication and data access and structure. More, software architecture can be defined as the output of a selection among design alternatives [5]. Another important definition put the stress on the architecture as set of design decisions [6]. This last definition highlights the role of decision-making in software architecture.

Chiefly, decisions have been taken regarding the quality targets. Architects should take decision regarding overall architectures by balancing the different quality attributes which the system needs to exhibit. In that evaluation, we have competing quality attributes and it is not possible to address all the QAs at the same degree. In practice we have to sacrifice some properties (or the level of them) for other properties. We observe trade-offs among quality attributes. As outlined before an important quality dimension of sustainability is Energy Efficiency. Energy Efficiency can be considered as a system quality attribute. Our interest is in an early identification on how Energy Efficiency can re-shape the knowledge we have on the interaction quality- architectural patterns. One quality attribute that already shows interaction with green quality aspect is Maintainability [7]. We will compare the scarce knowledge available on the relationship between Maintainability and Energy Efficiency to the established knowledge on Maintainability and architectural patterns. The remained paper is organized as following: Section II shortly describes related work, Section III focused on the background, while Section IV shows the potential interaction between Maintainability and Energy Efficiency in the context of architectural patterns. Finally, Section V explains threats to validity and Section VI presents Conclusion and future work.

II. Related Work

There are few works explicitly focused on green in software architecture. For instance, in [8] an energy-layer in the software architecture has been introduced. This energy layer has the responsibility of measure energy consumption in data centers and making services migrate to hosts more energy efficient. Another approach appears in [9] where a business application has been analyzed. The aim of the author is to analyze architecture's components in order to identify drivers of energy con-

sumption. Later, it is possible to modify/extend the architecture in order to create energy savings. In [10] a model-based Energy Efficiency analysis method has been introduced. The way chosen by [10] follows the example of prediction models for other quality attributes such performance. We have found only one work only focused on Energy Efficiency and Architectural Patterns[1]. In that work [11] some architectural patterns (*distributed systems* architecture) have been assessed according to Energy Consumption. The assessment has been made by an evaluation framework that can support early architectural decisions energy-efficiency aware. The most intriguing finding of [11] is that, in some case, combination of patterns allows to result less energy trend than the single pattern usage.

Interesting is the focus on green architectural tactics [12]. Green tactics are design-decisions that allow software lower energy consumption. A similar approach has been followed by [13] where appears the quality attribute Greenability. Under this label some of the scenarios presented in [12] have been grouped, for instance Energy Monitoring and Consolidation.

III. BACKGROUND

A. From Sustainability to Greenability

As reported before we are focusing on Energy Efficiency (Greenability) as Quality Attribute. However, Green software quality dimensions might be considered more than only Energy Efficiency.

Sustainability is a broader concept and its definition depends on the context factors [14] [15]. Indeed, sustainability shows several dimensions such as social, environmental, technical and economic. In the context of Software Engineering sustainability involves software development, maintenance and usage and how those three processes use resources. Greenability refers to the degree *which a product lasts over time, optimizing the parameters, the amounts of energy and the resources used and Energy efficiency is the degree of efficiency with which a software product consumes energy when performing its functions* [16]. Energy Efficiency is also considered as quality attribute in [17]. The difference between Energy Efficiency and Greenability is that the former is a sub-characteristic of the latter (the others sub-characteristics are Resource Optimization, Capacity Optimization and Perdurability) [16]. In this work we focus only on Energy Efficiency.

B. Architecture trade-off analysis method

Since we are interested in understanding how Energy Efficiency compete/interact with others quality attributes, the Architecture Tradeoff Analysis Method (ATAM) is an appropriate framework. ATAM is aimed to understand the quality attributes tradeoffs intrinsic in the architectures of software-intensive systems [18] [19]. ATAM has been developed at the Software Engineering Institute (SEI) and has been applied on a wide range of architectures, such as air traffic control, financial management, vehicle control etc. The aim of this structured technique is to evaluate software architectures with respect to multiple conflicting quality attributes: modifiability, security, performance, availability, and so on. Quality attributes interact and it involves that, by improving one quality attribute, the other(s) becomes worse. It implies tradeoffs among competitive quality attributes. ATAM is based on a spiral model of design: candidate architectures are continuously refined as outcome of analytic and risk mitigation processes [20].

C. Architecture patterns and quality

Architectural patterns and styles are recurrent solutions to common problems [21] and they include knowledge on quality attributes [22]. In the literature, patterns have been usually described according to the functionality they deliver and the strength or liability showed with respect to several quality attributes. According to [22] strengths or liabilities assess the importance of the impact of patterns on quality attributes. For instance: a key strength or key liability determines if to use or to avoid a pattern in a specific situation. In this line of reasoning, the degree which patterns impact on quality might determine architectural choices (i.e. adopting or avoiding a pattern for a given design problem).

IV. HYPOTHESIS ON MAINTAINABILITY AND ENERGY EFFICIENCY IN ARCHITECTURAL PATTERNS

In [7] we observe an early attempt to define the relationship between Maintainability and Greenability. Maintainability has been defined according to [23]: *degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers.* Greenability has been defined as the *degree of environmental friendliness of a software system, based on its power consumption* [7]. This last definition highlights the role of power consumption. Therefore, as stated before, for the purpose of this work, we focus on Energy Efficiency. However, the terms Energy Efficiency and Greenability might be used interchangeably. In this way we can preserve the use of the term Greenability seen in [7]. Maintainability can be analyzed according its sub-characteristics: Modularity, Reusability and Modifiability, Analyzability and Testability. The relationship between Maintainability and Energy Efficiency has been reported in Table 1.

TABLE I. MAINTAINABILITY AND GREENABILITY RELATIONSHIP

Quality Attribute [a]	***Sub-Characteristics***	***Relation to Greenability***
Maintainability	Modularity	Present
	Reusability	Present
	Analyzability	Not detected
	Modifiability	Present
	Testability	Not detected

a. According to [7]

The relationship between Maintainability and Greenability cannot be considered yet as validated. Studies on both quality attributes suggest that likely we have an implication, but we do

[1] In that work the authors use the term "style". We consider the term style and pattern as equivalent. For that discussion see [24] and [25].

not know yet if it is a positive, neutral or negative and in which contexts it works. Some scenarios for supporting the existing relationship between each sub-characteristic and Greenability have been described as following:

- More modules imply more communication lines. The presence of many modules might increase the energy consumption. However, an optimized modularization design can affect positively Maintainability so a less energy consuming maintenance [7].
- In general, *better design implies less energy and time to carry out any other task of maintenance* [7].
- *Reusability has likely a good impact on Greenability.* [7].
- If a system is easy to modify it does positively influence Greenability.
- Software maintenance focuses on preserving system functionalities [7]. Preserving functionalities is accomplished through analysis, modification and improvement of the source code [7]. For instance: software refactoring, seems to have positive influence over Greenability [7].

All the previous assumptions found in [7] are hypothesis that attempt to qualify the implication between Maintainability and Greenability. They all need experimentation. To summarize we have the following hypothesis regarding Maintainability and energy efficiency:

- H1- Generic statement: better design should positively affect energy consumption.

We consider "better design" regarding maintainable solutions. This statement ceases to be generic in a concrete architecture scenario.

- H2- Modularity affects Energy Efficiency. The measure of that influence is determined by the degree of modules optimization
- H3- Reusability is a quality attribute "friendly" for energy efficiency
- H4-Modifiability is a quality attribute " friendly" for Energy Efficiency
- H5- Preserving functionalities through source code improvements supports Greenability.

For "friendly" we meant that achieving modifiability or Reusability increase energy savings. In other words, modifiability or Resusability affect positively Greenability.

At this point the question is: What do those assumptions mean in the context of architectural patterns? What do we already know on Maintainability and architectural patterns can support us for identifying energy efficient patterns? In the following section we have compared the knowledge available on architectural patterns and Maintainability to the hypothesis derivate from [7].

A. Maintainable Architectural Patterns

At first we gathered some information on the relationship between Maintainability (and its sub-characteristics) and architectural patterns. We consider which patterns are better for achieving Maintainability (or Modularity, Reusability, Testability, Analyzability; Modifiability). We exclude analyzability because in [7] no implication has been found and we have not relevant information in the context of architectural patterns. Regarding Testability we adopt a different approach: although no implications have been detected in [7] we show the information on Testability because they suggest a possible implication. Indeed, Testability often is highly supported when is possible to test single component time by time, it recalls Modularity. In a previous work (a Systematic Literature Review see [24] and [25][2]) we count Maintainability 60 times, Reusability 37, Modularity 4, Analyzability 1 and Testability 6 appearing in the primary study, with a relation (positive or not) with several architectural patterns. In the following sections we describe the interaction between the most common patterns and Maintainability. We selected nine patterns according to two of the main taxonomies available in the literature [21] [22].

B. Layered architecture (L)

Layered architecture appears as one of the best pattern for achieving Maintainability. In many studies we found a positive interaction between Maintainability and layered, for instance in [26] [27] [28] and [29]. However, those sources stated in a general way the positive interaction. Moreover, we have to cope with fragmented and incomplete information. For instance, in [30] layered is described as positive for almost all the sub-characteristics of Maintainability, except Modularity. However, the negative interaction Layered-Modularity is not explained. The following points explain the positive interaction between layered architecture and Maintainability:

a) Late change in the source code do not propagate (ripple effect) through the system [31]
b) In [32] we have a general rule that states "Maintainability is better achieved with low coupling and high cohesion". However, this statement should be tested, especially because it is not clear what means "low" and "high".
c) In [33] two alternative architectures have been described. The first one is a three-layered plug-in, the second is a separation of three plug-in each based on a single layer. The first option shows more support for Maintainability due to the easier way of detect errors; however there is a trade-off with a sub-characteristic of Maintainability, Reusability. Reusability is higher in the second solution.
d) Separation of layers supports better Maintainability [34]
e) Layered architecture allows separate modification of layers (High modifiability) [22].

[2] See also the resource online http://www.s2group.cs.vu.nl/gianantonio-me/

f) Layered architecture allows separate testing of layers (High Testability) [22].

We can hypothesize that in Layered Maintainability has a potential positive effect on energy efficiency. Indeed, we can explore the H5 with the point a); if H4 is true Layered matches that hypothesis with (e). H1 can be corroborated by specific scenarios and concrete architecture alternative like point (c). We can also advance new hypothesis regarding Testability, through point (f). Point (b) introduces more knowledge on why this pattern supports Maintainability. To conclude, layered shows a high support for Maintainability. Considering the relationship between Maintainability and Energy Efficiency we can state that in *Layered Maintainability potentially positively affects Energy Efficiency.* This rule cannot be generalized because we need to consider design optimization (for instance numbers of modules, coding design etc.) according to H1. For instance, if we explore variants of patterns we can observe a loss of Maintainability. This is the case of the Relaxed Layered System described in [21]. Indeed in this system Maintainability decreases because any layer can use services of all layers below it, not only the next layer. Performance increases, Maintainability worsens. There is no evidence available regarding the energy consumption in this case. To conclude, we can say, with caution, if we are looking for a good balance for the trade-off Maintainability-Greenability Layered Architecture Pattern is a good option.

C. Pipes & Filters Architecture (P&F)

Pipes and Filters Architecture, like Layered, shows positive support for Maintainability. In particular we have found positive relationship in [26] [30] and [32].We have found some dissonant evaluations like in [29], however without providing any example. Pipes and Filters Architecture shows high Maintainability because it addresses Reusability (filters can be maintained as individual [35]). This last pattern might be considered as supporting positive implication Maintainability-Energy Efficiency. The possibility to work on single filter prepares for Maintainability: it happens by localizing changes and minimizing their side effects on other components [22]. This increase Modularity and matches H2.

D. Blackboard Architecture (Bl)

Blackboard Architecture shows contradictions from the information we gathered. For instance it is considered not a good support for Maintainability in [26] and in both [22] and [30] shows limit in Testability. However in [21] Blackboard supports Maintainability because all modules are strictly separated but at the same time they can all communicate through the Blackboard. Modularity seems to be satisfied and the communication between modules optimized, so this pattern might be a potential energy efficient pattern. It matches H2.

E. Model View Controller (MVC)

Model View Controller shows some weaknesses in addressing Maintainability because of coupling of views and controller to model [22]. However this weakness seems to consider how the MVC has been coded than an intrinsic characteristic of the pattern. Likely if there is an implication between Maintainability and Greenability it would match H1 and H5.

F. Presentation Abstraction Control (PAC)

Presentation Abstraction Control supports Maintanability due to its separation of concerns [22]. It matches H2.

G. Microkernel (M)

Microkernel is very maintainable thanks its Flexibility and Extensibility [22]. However those two characteristics are beyond the scope of this work. In the standard ISO 25010 those sub-characteristics are not considered belonging to Maintainability.

H. Reflection (R)

Reflection represents a case where there is no explicit modification of the source code [22]. Matching H5.

I. Broker (Br)

Broker addresses Modifiability, components can be easily changed [22]. It matches H4.

J. Reflective Broker Combination (R+Br)

This combination enhance separation of concerns, incorporates different control strategies for achieving Maintainability [36]. It matches H2 and H4.

K. Final Summary

We start from the knowledge on Maintainability and Greenability (Energy Efficiency) relationship. We distillated some hypothesis (H1..H5) that can shape the trade-offs between Maintainability and Greenability. We compare those Hs to the knowledge widespread in the literature on a selection of 9 architectural patterns. We recognize that some characteristics of patterns that support Maintainability are likely good influencer for Greenability, namely:

- Source code change does not affect the overall system
- Modularity: single components like layers, communication components or filters can be analyzed singularly. This makes easy maintenance and likely allows energy savings.
- Modifiability: some patterns (Layered, Pipes&Filters and Broker) are easily modifiable.
- Testability: although not detected in [7] we advance that likely exist an implication Testability-Greenability. It follows the same behaviour of modifiability: i.e. single components are better testable.

However, there are some problems that need to be solved. First, in patterns variants Maintainability's support change (like all the others quality attributes). So we cannot say Layered is a green efficient pattern, because its degree of Greenability changes from variants to variants. Perhaps it shows a "green attitude" more than others patterns. Specifically, inside Layered architecture, Maintainability likely positively influences Greenability. The key seems to be that we cannot compare different patterns in terms of Energy Efficiency but *we have to*

compare different architectural solutions inside the same pattern chosen. This point can be endorsed by the fact that we do not find any pattern totally bad for Maintainability. In some cases, the different degree of support for quality attributes involves sub-characteristics.

Therefore, we need a more structured analysis inside the same pattern and inside the same quality attribute by exploring trade-offs between sub-characteristics. The following table recaps the matching between hypothesis and patterns.

TABLE II. MAINTAINABILITY AND GREENABILITY RELATIONSHIP

Hypothesis	***Architectural Patterns*** [a]								
	L	***P&F***	***Bl***	***MVC***	***PAC***	***M***	***R***	***Br***	***R+Br***
H1	X			X					
H2	X	X	X		X				X
H3		X							
H4	X	X						X	X
H5	X			X			X		

a. L=Layered; P&F= Pipes & Filters; Bl=Blackboard; MVC= Model View Controller; PAC= Presentation Abstraction Control; M=Microkernel; R=Reflection;Br=Broker; R+Br: Combination Reflective Broker

V. THREATS TO VALIDITY

The analysis presented in this short paper is not supported yet by evidence. We generated hypotheses, and we need further validation. Moreover our starting point shows limits. Indeed the interaction between Maintainability and Energy Efficiency found in [7] is not fully validated and has been applied with a different purpose from this work. Validity is also threatened by the contradictory and incomplete information available on architectural patterns. Finally, we used Energy Efficiency and Greenability interchangeably. This choice is a weakness. Indeed the issue about those two terms requires further and rigorous analysis.

VI. CONCLUSION

In this work, firstly we assume that the relationship between Maintainability and Greenability (or Energy Efficiency) is in reality how has been described in [7]. Secondly, we compare the knowledge that we have on Maintainability in software architectural patterns with the hypothesis on the relationship between Maintainability and Greenability. In general, seems that all the architectural patterns that support Maintainability encompass a potential positive relationship between Maintainability and Greenability. Indeed, patterns that support Maintainability shows controllable and localized changes in the source code, advantageous Modularity and Reusability. Although it is not clear if Testability can be included, it seems that it has the same behaviour of Modularity. If we follow the assumption that those characteristics are good for Greenability we should conclude that almost all the architectural patterns show a positive trade-off between Maintainability and energy consumption. Shall we conclude that they are energy efficient architectural pattern? No, because it cannot be so simple. According to the information we have regarding variants, variants of a pattern can reduce a quality attribute level. Moreover Maintainability is composed by several sub-characteristics, in some case conflicting among them.

The final consideration is that we need a structured analysis where the level of Energy Efficiency of a pattern is estimated among variants of the same pattern, and the trade-offs also examined between sub-characteristics. A candidate future work should overcome the notion of "pure pattern", designing an experiment that collects energy consumption's measure of variants of the same pattern. The goal is to identify which design decisions (changing the "pure patterns") allow us to achieve a positive balance between Maintainability and Greenability.

ACKNOWLEDGMENT

This work is part of the GINSENG (TIN2015-70259-C2-1-R) project (funded by the Spanish Ministerio de Economía y Competitividad and by FEDER-Fondo Europeo de Desarrollo Regional) and by VILMA (PEII11-0316-2878) and GLOBAL-IA (PEII-2014-038-P) projects (funded by the Junta de Comunidades de Castilla-La Mancha and by FEDER-Fondo Europeo de Desarrollo Regional).

REFERENCES

[1] C. Calero and M. Piattini, *Green in Software Engineering*, Springer, 2015.

[2] C. Calero and D.C. Torre, "How sustainable are model software artifacts in the context of Model-Driven Software", submitted to MeGSuS: 3rd International Workshop on Measurement and Metrics for Green and Sustainable Software, ESEIW, Ciudad Real, ES, September 2016.

[3] I. Gorton, *Essential software architecture*, Springer Science & Business Media, 2006.

[4] R. Kazman, G. Abowd, L. Bass and P.Clements, "Scenario-based analysis of software architecture", *IEEE software* , 13, no. 6:47-55, 1996.

[5] D.Garlan and M. Shaw, "An introduction to software architecture", *Advances in software engineering and knowledge engineering*, 1(3.4), 1993.

[6] J. Bosch, "Software architecture: The next step" in *Software architecture*, pp. 194-199, Springer Berlin Heidelberg, 2004.

[7] I.G.-R. de Guzmán, M. Piattini and R. Pérez-Castillo, "Green software maintenance", in *Green in Software Engineering*, Springer International Publishing, 2015, pp. 205-229.

[8] R. Beik, "Green cloud computing: An energy-aware layer in software architecture", in Engineering and Technology (S-CET), 2012 Spring Congress on, pp. 1-4. IEEE, 2012.

[9] E. A. Jagroep, J. M. E.M. van der Werf, R. Spauwen, L. Blom, R. van Vliet, and S. Brinkkemper, "An energy consumption perspective on software architecture", in European Conference on Software Architecture, Dubrovnik, HR, Springer International Publishing, 2015, pp. 239-247.

[10] C.Stier, A. Koziolek, H.Groenda, and Ralf Reussner, "Model-Based Energy Efficiency Analysis of Software Architectures", in European Conference on Software Architecture, Dubrovnik, HR, Springer International Publishing, 2015, pp. 221-238.

[11] C. Seo, G. Edwards, S. Malek, and N. Medvidovic, “A framework for estimating the impact of a distributed software system's architectural style on its energy consumption”, in Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA), Vancouver, CA, IEEE, 2008, pp. 277-280.

[12] G. Procaccianti, P. Lago, and G. A. Lewis, “Green architectural tactics for the cloud”, in Eleventh Working IEEE/IFIP, Conference on Software Architecture (WICSA), Sydney, AU, IEEE/IFIP, 2014, pp. 41-44.

[13] M. Salama and R. Bahsoon, “Quality-Driven Architectural Patterns for Self-Aware Cloud-Based Software”, in 8th International Conference on Cloud Computing,New York, US, IEEE, 2015, pp. 844-851.

[14] B. Penzenstadler, V. Bauer, C. Calero, and X. Franch, “Sustainability in software engineering: A systematic literature review”, in 16th International Conference on Evaluation & Assessment in Software Engineering (EASE), Ciudad Real, ES, IET, 2012, pp. 32-41.

[15] B. Penzenstadler, “Towards a Definition of Sustainability in and for Software Engineering”, in Proceedings of the 28th Annual ACM Symposium on Applied Computing, Gyeongju, KR, ACM, 2013, pp. 1183-1185.

[16] C. Calero, Mª Á. Moraga, M. F. Bertoa and L. Duboc, “Green Software and Software Quality”, in *Green in Software Engineering*, Springer International Publishing, 2015, pp. 231-260.

[17] L. Bass, P. Clements and R. Kazman, *Software architecture in practice*, 3rd ed. Addison-Wesley, 2012.

[18] R. Kazman et al., “A basis for analyzing software architecture analysis methods”, in *Software Quality Journal*, 13.4 : 329-355, 2005.

[19] R. Kazman et al, “The architecture tradeoff analysis method”, Proceedings of Fourth IEEE International Conference on Engineering of Complex Computer Systems, (ICECCS), Monterey, US, IEEE, 1998.

[20] B. Boehm. “A Spiral Model of Software Development and Enhancement”, *ACM Software Eng. Notes 11*, 4 (August 1986): pp. 22-42.

[21] F. Buschmann, R. Meunier, H. Rhonert, P. Sommerlad and M. Stal, *Pattern-Oriented software architecture: A system of patterns,* Wiley, West Sussex, England, 1996.

[22] N. Harrison and P. Avgeriou, “Leveraging architecture patterns to satisfy quality attributes”, in proceedings of First European Conference on Software Architecture (ECSA), Madrid, ES, Springer LNCS, 2007, pp.263-270.

[23] *ISO/IEC, ISO 25000*, Software product quality requirements and evaluation (SQuaRE), 2005

[24] G. Me, C. Calero and P. Lago, “Architectural Patterns and Quality Attributes Interaction”, in Workshop on Qualitative Reasoning about Software Architecture (QRASA), co-located with Working IEEE/IFIP Conference on Software Architecture (WICSA), Venezia, IT, 2016.

[25] G.Me, C. Calero and P. Lago, “A long way to a quality-driven pattern-based architecting”, in European Conference on Software Architecture (ECSA), Copenaghen, DK, *article accepted*, 2016.

[26] M. Svahnberg and C. Wohlin, “An investigation of a method for identifying a software architecture candidate with respect to quality attributes”, in *Empirical Software Engineering* 10, no. 2: pp. 149-181, 2005.

[27] E. Niemela, J. Kalaoja and P. Lago, “Toward an architectural knowledge base for wireless service engineering”, *IEEE Transactions on software engineering* 31, no. 5: 361-379, 2005.

[28] J. Berrocal, J. García-Alonso and J. M. Murillo, “Facilitating the selection of architectural patterns by means of a marked requirements model”, in European Conference on Software Architecture (ECSA), Copenhagen, DK, Springer Berlin Heidelberg, 2010, pp. 384-391.

[29] K. Babu, P. Govinda Rajulu, A. Ramamohana Reddy, and A. N. Kumari, “Selection of architecture styles using analytic network process for the optimization of software architecture", in *arXiv* preprint arXiv:1005.4271, 2010.

[30] H. Yang, S. Zheng, W. Cheng-Chung Chu, and Ching-Tsorng Tsai. “Linking functions and quality attributes for software evolution”, in 19th Asia-Pacific Software Engineering Conference, Hong Kong, HK, IEEE, 2012, vol. 1, pp. 250-259.

[31] I. Araujo and M. Weiss, “Linking patterns and non-functional requirements”, in Proceedings of the Ninth Conference On Pattern Language Of Programs (Plop 2002), Monticello, US, September 8–12, 2002.

[32] G. Grau and X. Franch, “A goal-oriented approach for the generation and evaluation of alternative architectures." in European Conference on Software Architecture (ECSA), Madrid, ES, Springer LNCS, 2007, pp. 139-155.

[33] D. Ameller, O. Collell, and X. Franch, “Reconciling the 3-layer Architectural Style with the Eclipse Plug-in-based Architecture”, in Proceedings of the 1st Workshop on Developing Tools as Plug-ins,Hawai, US, ACM, 2011, pp. 20-23.

[34] J. Berrocal, J. García-Alonso and J. M. Murillo, “Modeling business and requirements relationships for architectural pattern selection”, in *Software engineering research, management and applications*, pp. 167-181. Springer International Publishing, 2014.

[35] U. Banodha and K. Saxena, “Impact of Pipe and Filter Style on Medical Process Re-engineering”, in *International Journal of Engineering Sciences* 4, pp. 398-409, 2011.

[36] O. Silva, A. Garcia and C. Lucena, “The reflective blackboard pattern: Architecting large multi-agent systems”, in International Workshop on Software Engineering for Large-Scale Multi-agent Systems, In Proceedings of the 24th International Conference on Software Engineering (ICSE), Orlando, US Springer Berlin Heidelberg, 2002, pp. 73-93.