Int J Syst Assur Eng Manag (November 2017) 8(Suppl. 2):S886–S909
DOI 10.1007/s13198-016-0546-8

ORIGINAL ARTICLE

A guideline for software architecture selection based on ISO 25010 quality related characteristics

Mariem Haoues[1] · Asma Sellami[1] · Hanêne Ben-Abdallah[1,2] · Laila Cheikhi[3]

Received: 22 December 2015 / Revised: 16 September 2016 / Published online: 8 November 2016

Abstract As the complexity of software increases, the choice of the appropriate software architecture becomes a critical task. This paper provides a guideline for selecting the appropriate software architecture based on pertinent ISO 25010 quality characteristics. The guideline was established through an analytical survey of 113 papers published from 2010 to 2014. Through this survey, we first identified a set of commonly used software architectures in the software engineering literature. Secondly, we applied the Formal Concept Analysis technique to classify each one of these architectures according to ISO 25010 quality characteristics. Finally, we identified the relationships among ISO 25010 quality characteristics, which in turn helped us to develop a guideline on how to select the appropriate software architecture with respect to ISO 25010 quality characteristics. In order to make sure about the validation of the proposed guideline, a survey with industrial experts is in progress. Data were collected from two companies working in the software development field (ST2i and Telnet).

Keywords Software architecture selection · Quality characteristics relationships · Formal Concept Analysis · ISO 25010 quality characteristics

✉ Mariem Haoues
mariem_haoues@yahoo.fr

[1] Mir@cl Laboratory, University of Sfax, Sfax, Tunisia

[2] King Abdulaziz University, Jeddah, Saudi Arabia

[3] ENSIAS, Mohammed V-Souissi University, Rabat, Morocco

1 Introduction

Due to the increasing complexity and challenging quality requirements of software systems, Software Architecture (SA) selection has become a crucial part of the software development. In fact, many researchers argue that the SA design phase is considered one of the most critical phases in the Software Development Life Cycle (SDLC) (Bass et al. 2012): Because the SA is decided relatively early in the SDLC, it can have a considerable impact on the quality of the final product (Bass et al. 2012).

During recent years, many definitions of SA have been proposed. For example, (Lüders 2003) defines SA as "the structural decomposition of software". On the other hand, (Garlan 2003) sees SA as a means to provide "high-level abstractions in representing the structure, behaviour, and key properties of complex software". In contrast, the IEEE defines SA as "the fundamental organization of a system embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution" (ISO/IEC/IEEE 42010 2011); this IEEE definition is the most adopted definition. Based on these definitions, it can be observed that the SA is created through a set of decisions that:

- represent the structure, the behaviour and the global attributes of a system at a high-level of abstraction (design phase);
- define all the functional requirements; and
- define the non-functional requirements (quality attributes).

The choice of the appropriate SA should overcome the challenges introduced by the three types of software requirements: According to (COSMIC 2015), software

requirements can be classified into three categories: Functional User Requirements (FUR), Non-Functional Requirements (NFR), and Project Requirements and Constraints (PRC). FUR express "what the software shall do in terms of tasks and services" (COSMIC 2015). NFR include "any requirement for a hardware/software system or for a software product, including how it should be developed and maintained, and how it should perform in operation, except any functional user requirement for the software" (COSMIC 2015). PRC describe "how a software system project should be managed and resourced or constraints that affect its performance" (COSMIC 2015).

The above sources of complications on the SA definition and on the various quality models make software architects' role vital in the success of the software project. Indeed, the choice of the appropriate SA can be beneficial in reducing the software development cost (Microsoft 2009), while an incorrect choice may increase it; enormous problems can be induced by an incorrect choice of the SA like the instability of the software, the inability to support existing or future business requirements, etc. (Northrop 2003). Due to the important number of software architectures, software architects face a challenge in choosing the SA appropriate to their needs. In this paper, we propose a guideline for selecting the appropriate SA that addresses the ISO 25010 quality characteristics.

Researchers agree that software quality should be taken into consideration throughout the SDLC phases. In particular, many studies argue that the SA plays an important role in achieving the quality of the final product (*cf.* (Losavio et al. 2003; Aleti et al. 2013; Wang and Yang 2012)). As such, the SA should be selected with respect to the desired quality characteristics. Nonetheless, existing works either examine quality characteristics of the SA itself (*e.g.,* Reliability (Delač 2012), Maintainability (Espinha et al. 2012), etc.), or the quality characteristics involved in the architecture design process (Losavio et al. 2003). In this study, we provide an evaluation of the most used software architectures in the literature according to a set of quality characteristics (as identified by (ISO/IEC 25010 2011). This evaluation was the basis for the elaboration of a guideline on SA selection with respect to the desired quality characteristics.

More specifically, we surveyed 113 papers published from 2010 to 2014 to identify a set of commonly used architectures in the software engineering literature. We applied the Formal Concept Analysis technique to classify these architectures according to ISO 25010 quality characteristics. Finally, we identified the relationships among ISO 25010 quality characteristics, which in turn helped us to develop the guideline on how to select the appropriate software architecture with respect to ISO 25010.

The remainder of this paper is organized as follows: Sect. 2 presents an overview of the most popular software quality models and their usage in the agile environment and introduces the related works including those that tackled the software architectures research area. Section 3 briefly presents the Formal Concept Analysis (FCA) technique and its application in the software engineering literature; it also describes how we applied FCA in our study. Section 4 presents and discusses the survey results, and provides a guideline for SA selection. In Sect. 5, we also discuss several threats to internal, external and construct validity. Section 6 summarizes the presented work and outlines some further works.

2 Background

2.1 Software product quality models

In the software engineering literature, different quality models have been proposed, e.g., McCall's (McCall et al. 1977), Boehm's (Boehm 1978), Dromey's (Dromey 1996), FURPS (Grady 1992), ISO 9126 (ISO/IEC 9126-1 2001), and recently ISO 25010 (ISO/IEC 25010 2011). Each model defines a set of quality attributes. As compared in (Couto et al. 2011), several attributes are common among some of these models, e.g., "Reliability"; other attributes are specific to some models, e.g., "Maintainability" is present in ISO 9126-1 but it is absent in the FURPS model which alone introduces the "Supportability". In addition, we note that some attributes are considered as a quality sub-characteristic in a model and a characteristic in another model. As an example, "Security" is a sub-characteristic in ISO 9126-1 while it is a characteristic in ISO 25010. We noticed also the use of different terminologies for the same concept (e.g., changeability in Boehm's and modifiability in ISO 25010), or by contrast (e.g., stability in ISO 9126-1 and instability in ISO 25010).

In a quest to cover most of the quality characteristics, we selected, in our analysis, the ISO 25010 model since it is the recent quality model and it is built based on an international consensus. Figure 1 presents a product quality model as described in (ISO/IEC 25010 2011). This quality model categorizes software product quality attributes into eight characteristics: functional suitability, performance efficiency, usability, compatibility, reliability, security, maintainability and portability. Each characteristic is composed of a set of related sub-characteristics. For example, "Portability" includes three sub-characteristics: adaptability, instability, and replaceability.

In the literature, many studies have examined the factors that affect software quality (Reliability, Maintainability, Usability, etc.). As an example, the study in (Zhang and

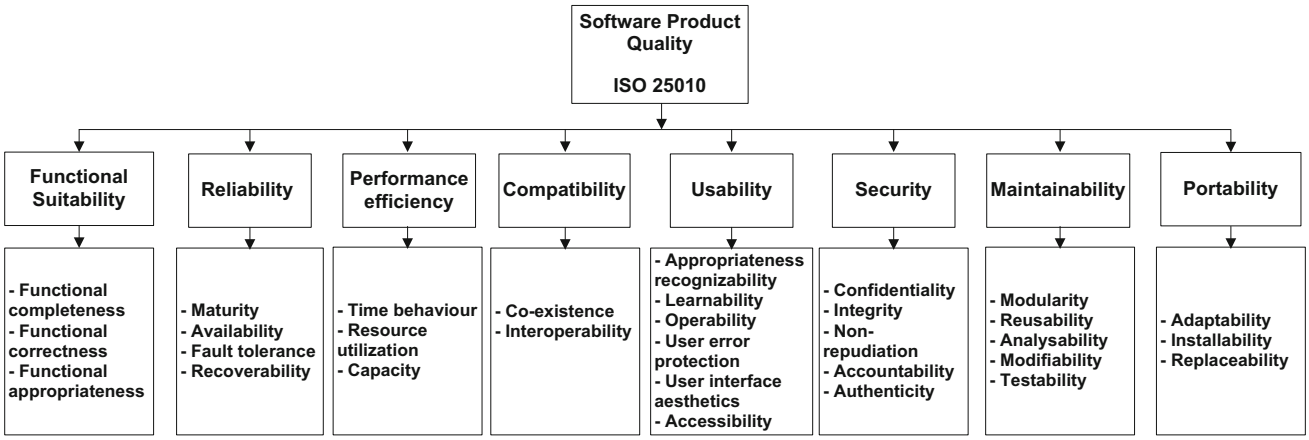

Fig. 1 Quality model-ISO/IEC 25010 (2011)

Pham 2000) focused on the factors affecting the software reliability. They did an experimentation with 13 companies to identify factors that have significant impact on the software reliability. They identified 32 factors involved in every phase of the software development phases (design, coding, testing, etc.) such as: Program complexity, Programmer skills, Testing coverage, etc. They analyzed the identified factors in order to determine these having a significant impact on the software reliability. They used two techniques (the relative weight method and the analysis of variance technique) to rank the identified factors. According to this study, the program complexity is the most important factor influencing on the software reliability. On the other hand, the study proposed by (Ghosh et al. 2011) focused on the factors affecting the software maintainability. They consider that the software complexity is the primary factor that affects the two sub-characteristics of maintainability (Testability and Modifiability). Other factors such as programming language, the trace of design, etc. may affect the software maintainability. Moreover, (Dehaghani and Hajrahimi 2013) studied the factors affecting the cost of software maintenance. They identified 32 factors such as Complexity Product, DataBase size, Programming language experience, etc. (Mahrin et al. 2008) focused on the factors affecting the usability of a software. Their results were based on an investigation from three sources of information: the literature, data collected from a survey of practitioners at the 5th Australia SEPG conference, and an analysis of core elements of software process meta-models. From the literature, they identified 19 factors such as: Consistency of presentation style, Use diagrams, tables and narrative effectively, etc. From the exploratory survey, they identified 25 factors such as: The use of understandable terminology, Easy to implement, etc. From the core elements of software process meta-models they identified four factors such as: Guidance, Work product, etc.

2.2 Software product quality models in Agile environment

Experts consider that an iterative agile approach may improves the quality of the software projects and reduce the time and effort development. However, in the literature it is not comprehensive which characteristics of is improved by the use of agile processes in developing software. Thus, (Mnkandla and Dwolatzky 2006) proposed to evaluate agile methodologies in order to determine which factors of software quality they improve. They used the ISO 9126 quality model. Each quality characteristic/sub-characteristic of the ISO 9126 quality model (Compatibility, Correctness, Ease of use, etc.) is evaluated in relation to agile techniques that implement the factors. For example, for the quality sub-characteristic (Ease of use), agile development ensure that the customer is part of the team, he gives feedback and asks always for improvement. Thus, he will likely recommend a system 'easy to use'. Then, agile development can guarantee the 'easy to use' sub-characteristic. (Mnkandla and Dwolatzky 2006) proposed a tool to evaluate quality in agile process. This tool has been evaluated using the Extreme Programming (XP) and the Lean Development (LD). They mentioned that XP ensure the software correctness through the User stories, the Unit tests, and the Customer feedback. While, LD ensure the software correctness through the meet of user requirements. Booth XP and LD ensure the software maintainability through the iterative development. In summary, many quality characteristics can be guarantee by the agile development techniques. However, this study didn't evaluate all the ISO 9126 quality model.

2.3 Related works

In the software engineering literature, various software architectures are used like: Model Driven Architecture-

MDA (*cf.* (Cheng 2010)) Component Based Architecture-CBA (*cf.* (Lee et al. 2010)), Service Oriented Architecture-SOA (*cf.* (Shanmugasundaram et al. 2012)), Object Oriented Architecture–OOA (*cf.* (Acheson 2010)), Event Driven Architecture-EDA (*cf.* (Tragatschnig and Zdun 2013)), Aspect Oriented Architecture–AOA (*cf.* (de Oliveira et al. 2013)), etc.

Several literature reviews on SA have been published. For example, (Couto et al. 2011) used FCA to present a literature survey on software architectures. They used a collection of 38 scientific papers published between 1992 and 2010, which encompassed different types of publications (surveys, evaluations, new proposals, etc.). Through this survey, the authors tried to answer the following four questions:

- What are the most supported definitions of software architecture?
- What are the most popular research topics in software architecture?
- What are the most relevant quality attributes of the software architecture? and,
- What are the topics that researchers point out as being more interesting to explore in the future?

This survey showed that there was no clear definition of the term "software architecture". In addition, it showed that SA quality was frequently examined in the literature (*cf.* (O'Brien et al. 2007), and (Immonen and Niemelä 2008)). It also showed that a number of quality attributes are widely used in the decisions taken during the SA design (e.g., maintainability, reusability, usability, performance, and reliability), whereas other quality attributes like scalability, interoperability and testability are used only in a few studies (*cf.* (O'Brien et al. 2007)). This empirical study managed to identify and classify the quality attributes regarding the software architectures. However, it did not use the quality attributes as a comparative criteria to capture the SA most appropriate for a particular set of quality characteristics; such a comparison can help a software architect understand how their architecture addresses the quality attributes in comparison with other architectures. The appropriate SA selection is one of the most important decisions guiding the quality of system/software product.

Trying to fulfil the need for comparative results on software architectures, the Systematic Literature Review (SLR) recently presented in (Aleti et al. 2013) provided an analysis and a classification of SA optimization methods. This review examined 188 researches published from 1992 to 2011 in order to find answers to the owing questions:

- How the current research on software architecture optimization can be classified?
- What is the current state of software architecture optimization research with respect to this classification? and,
- What can be learned from the current research results that will lead to topics for further investigation?

This SLR reported that most of the SA optimization methods try to improve SA quality. These methods can be classified into three categories: (i) improve one quality attribute (e.g., performance, cost, etc.); (ii) improve more than one quality attribute (*e.g.,* performance and reliability); or (iii) improve independently of any of the quality attributes. This study presented an interesting SLR of the SA optimization methods to advance the architecture optimization research area.

In an attempt to help in selecting the proper SA style for a given software, (Wang and Yang 2012) conducted a literature review in order to identify the SA styles that are commonly used, and how to select the proper SA style. This literature review proposed an evaluation of the most used SA styles according to some quality attributes. However, this evaluation is not based on any quality model. The selection of the appropriate SA style is provided using a systematic selection process powered by the Analytic Hierarchy Process (AHP). AHP is originally a mathematical decision-making technique proposed by Saaty in 1980. This technique provides how measuring an intangible criteria and how to interpret correctly measurements of tangibles (Saaty 2008). AHP has been widely used in different fields in order to help decision-makers understanding the problem and taking the best decision. (Wang and Yang 2012) illustrate their selection process to show that the Service-Oriented style is the most appropriate for B2B applications since, as argued in their AHP application, it better guarantees usability, maintainability, cost, and scalability. However, to be generalized across all types of software architectures and application domains, this proposed selection process requires a rich knowledge base to power the AHP. Such a knowledge base is typically built manually by experts, which is not a trivial task.

3 Formal concept analysis for software architecture selection

3.1 FCA fundamentals

In information science, FCA is a mathematical theory used for data analysis information retrieval, and knowledge discovery (Couto et al. 2011). FCA uses two concepts: data tables, and lattices. A data table represents binary relations between objects and keywords. A lattice is used to represent a structured view of the concepts under study (Couto

et al. 2011). The view/lattice can be parsed to understand which keyword is related to which object. (Couto et al. 2011) describe how to use FCA in preparing a literature review: First, for every research question/objective, a set of keywords are defined. Secondly, for each object (research study in this case), we identify the keywords used in order to determine the relationships among keywords and objects. Afterwards, a lattice can be drawn using the "Lattice Miner"[1] tool. An analysis of the lattice provides an answer of the related question.

3.2 Formal concept analysis in software engineering

Researchers proved that FCA has been widely used in different research areas, including the software engineering field for many purpose and during different phases of the SDLC. (Poelmans et al. 2010) showed that FCA has been applied in different phases of the SDLC such as software requirements analysis, implementation, maintenance, etc. For example, (Hesse and Tilley 2005) used FCA in requirements analysis phase in order to identify class candidates in use case descriptions. FCA also has been used in literature surveys since it can be helpful in classifying and structuring the bibliography. For example, (Couto et al. 2011) used FCA in order to provide a survey in the SA research area; (Hesse and Tilley 2005) surveyed papers that include the term "Formal Concept Analysis" in their abstract. This study showed that FCA can be applied in many disciplines such as: software engineering, knowledge discovery, information retrieved, etc.

3.3 FCA application

To achieve the objective of our study (i.e., providing a guideline for SA selection according to ISO 25010), three steps can be clearly distinguished: (1) Identify software architectures the most used in the literature, (2) Classify the identified architectures with regard to the ISO 25010 quality characteristics, and (3) Analyze the relationships between ISO 25010 quality characteristics. For the first and the second steps, we use FCA for papers classification. First we identify the set of pertinent keywords, which are classified into primary and secondary keywords. Primary keywords are those directly related to the corresponding step. For example, for step 1, we search for software architectures the most used in the literature, then primary keyword is "software architecture". Secondary keywords can be derived from the definition of SA as provided in the introduction, such as: design, structure, behaviour, etc. Once keywords are identified, we select from the collected papers those that include the identified keywords in their title, abstract, keywords, introduction or conclusion. Afterwards, we construct the data table where the rows are the selected papers and the columns are the identified keywords. Finally, by using the "Lattice Miner" tool, we can get a lattice that classifies these papers in terms of their keywords coverage. The next section discusses the FCA results of our survey.

[1] http://sourceforge.net/projects/lattice-miner/.

4 FCA results and discussions

4.1 Step 1: Software architectures most used in the literature

A set of papers are collected to reach the objective of this study. These papers are from four databases: IEEE Xplore, ACM Digital Library, Science Direct and CiteSeer. Papers should include the hereafter keywords that are deducted from the definition of "software architecture" as presented in the introduction.

- Software architecture: definition of the software architecture,
- Design: software architecture is identified at the design phase,
- Behaviour: software architecture presents the behaviour of systems,
- Structure: software architecture presents the structure of systems, and
- Quality: software architecture relying on non-functional requirements.

We classify the above keywords as (i) Primary keyword: software architecture; and (ii) Secondary keywords: design, behaviour, structure, and quality. Then, we classify each paper according to the keywords it includes: papers that do not include the primary keyword are excluded even if they contain secondary keywords. However, papers dealing with SA types (e.g., MDA, SOA, etc.) are included in our review. We added the SA type as a keyword to take into account the practical fact that the type might be imposed in some projects.

Papers collected in our review are from different types of publications. For instance:

- Surveys: (Chen et al. 2010) present an analysis and an evaluation of software architectures that are used during the last ten years. In addition, (Majidi et al. 2010) provide a comparative analysis between three kinds of architectures classifications in the literature; this study proposes some criteria to help in choosing the appropriate software architecture style in different applications. Furthermore, (Aleti et al. 2013) proposes a SLR to analyze and classify software architectures

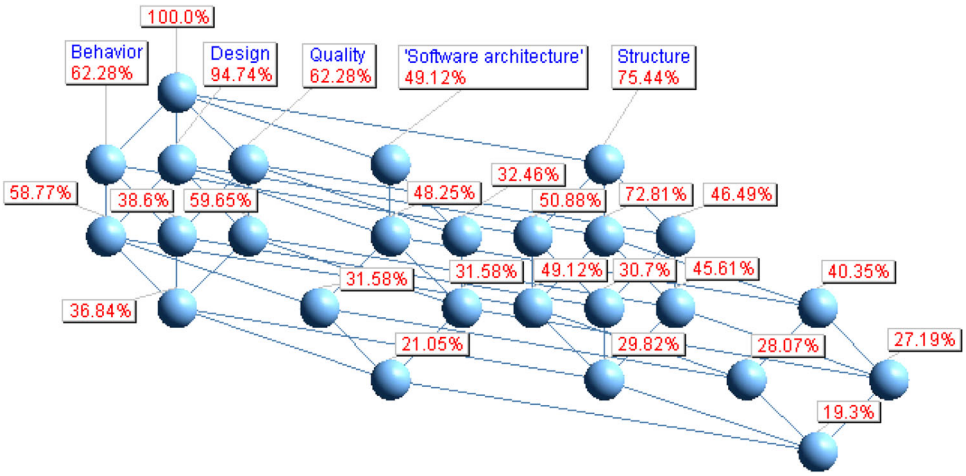

Fig. 2 Concept lattice for keywords repartition in step 1

optimization methods, in order to help researchers to classify their research approach and to identify new directions for further works.

- Evaluations: Some studies present an evaluation and a comparison between software architecture types or styles (*cf.* (Majidi et al. 2010)). Other studies proposed a comparison between SA types. For example, (Li and Huang 2013) provided an evaluation of SA reliability based on hypergraph grammar. They proposed a reliable hypergraph grammar by extending the hyperedges with reliable elements from reliability model. Software architecture is then described by using the hypergraph grammar. Since, the hypergraph is based on the reliability model, then the reliability of the corresponding software architecture can be predicted.
- Combination: Some studies propose the combination or the grouping of two architectures to improve the software development quality, such as the research conducted by (Yue et al. 2010). This study proposed the grouping between CBA and SOA to improve software reusability.
- Style: Some studies are related to a specific SA type. These studies examine various related subjects such as: architecture refinement (*cf.* (Zhang et al. 2010; Gao et al. 2010)), quality improvement or evaluation (*cf.* (Fan et al. 2012; Delač 2012)), etc.
- New proposal: Some studies propose a new SA that can be more adapted to their specific needs (*cf.* (Weini et al. 2012)). They proposed a new software architecture designed for ultra-large-scale rendering cloud. It includes five levels: infrastructure layer, rendering service layer, rendering application layer, service management and access interface. (Weini et al. 2012) proposed also the development of a Golden Farm Cluster Rendering Platform where some experimental results can be carried out. The proposed platform is considered as an effective solution for ultra-large-scale rendering, since it provides a high performance-computing cluster, and it has a high availability, reliability and scalability.

As showed in Fig. 2, 49.12% of the selected papers used the primary keyword "Software Architecture". In addition, we note that:

- 48.25% of the selected papers used the primary keyword "Software Architecture" with the secondary keyword "design", because the software architecture is defined in the design phase;
- 32.46% of the selected papers focused on the quality of software architecture, they used the primary keyword "Software Architecture" with the secondary keyword "quality"; and
- 59.65% used the two secondary keywords "design" and "quality". Then, we note that the quality of design phase is widely discussed in the literature. In fact, the design quality will have an effect on the quality of the final product.

Furthermore, the behavioural and structural aspects of software architecture have been also studied:

- 40.35% of the selected papers focused on the structural aspect of software architecture in the design phase; they used the primary keyword "Software Architecture" with the secondary keywords "design" and "structure".
- 31.58% of the selected papers focused on the behavioural aspect of the software architecture in the design phase; they used the primary keyword "Software Architecture" with "design" and "behaviour" the secondary keywords.

After an initial scan of a set of papers selected according to the primary keyword "Software Architecture", we restricted ourselves to 113 papers. We found that:

- four papers provide a literature survey (Aleti et al. 2013; Chen et al. 2010; Couto et al. 2011; and Majidi et al. 2010);
- 13 papers studied the combination between two software architectures, such as: EDA with SOA (Tragatschnig and Zdun 2013; and Yuan and Watton 2012), MDA with AOA (Fang et al. 2010; and Linehan and Clarke 2012), MDA with SOA (Bispo et al. 2010; Borek et al. 2012; and Yang et al. 2012), SOA with CBA (Cheaito et al. 2010; Quintero et al. 2010; and Yue et al. 2010), CBA with AOA (Tizzei et al. 2011), and CBA with EDA (Klatt et al. 2011). As an example, (Yue et al. 2010) investigated the improvement of software development quality with software reuse. Their approach is based on a component-based SOA to guide the reusable software development;
- one paper provides a new proposal (Weini et al. 2012). They proposed a five-level software architecture designed for ultra-large-scale rendering cloud;
- three papers studied the comparison between software architecture types (Abdelmoez et al. 2012; Li and Huang 2013; and Majidi et al. 2010). For example, (Majidi et al. 2010) focused on the comparison between OOA and AOA regarding to "Maintainability";
- two papers studied the refinement of an architecture (Li and Huang 2013; and Zhang et al. 2010); and
- 90 papers are related to specific software architecture with quality attributes. Where only one study used the Object Oriented Architecture (Tizzei et al. 2011).

Based on the above analysis, we had to exclude the following papers to meet the objective of our research:

- papers that do not specify a SA type, or papers that do not include the primary keyword "software architecture" are excluded;
- papers used a software architecture type cited few times are also excluded;
- papers that propose new software architecture or a comparison of two architectures are excluded; and, finally
- papers presenting surveys are also excluded. Hence, we kept only those papers focusing on the refinement, the improvement or the evaluation of a specific software architecture, and papers proposed the combination of two architectures.

The retained 105 papers for a specific software architecture types are classified into five groups: SOA, CBA, MDA, EDA, and AOA. Figure 3 shows that most of the collected papers focused on: SOA (39%), CBA (27%), and MDA (17%).

Our close analysis of these 105 papers showed that different subjects have been studied. Since quality is widely considered when working with software architecture, many researches focused on the use of quality characteristics with their selected architectures. Other studies proposed the combination of two architectures to improve a quality characteristic (*cf.* (Yue et al. 2010)). MDA and SOA have been used together the most; this is justified by the fact both concepts are in fact close.

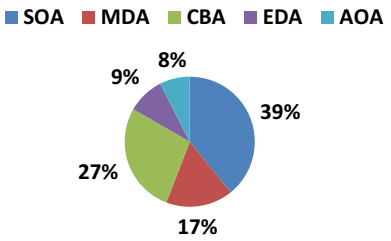

Fig. 3 The used software architectures-Summary of the quantitative results

4.2 Step 2: Classification of Popular SA Relying on the ISO 25010 Quality Characteristics

In this step, papers that do not include the use of a specific software architecture type with a quality attribute were excluded, such as papers studying the refinement of software architectures. We limited ourselves on the 103 remaining papers. An initial scan of these papers showed that some studies focus on the ISO 25010 quality characteristics (*e.g.* maintainability, reliability, etc.). Other studies use quality attributes that appear to be not supported by ISO 25010 quality model because of the ambiguity in using terminologies. In fact, some researchers used quality attributes that are represented by one of the ISO 25010 quality characteristics or sub-characteristics or through more than one ISO 25010 quality characteristic or sub-characteristic. For example, the term "Changeability" is frequently used to reflect the concept of the ISO 25010 sub-characteristic "Modifiability". Also, the term "Scalability" refers to the "Adaptability" sub-characteristic related to the ISO 25010 quality characteristic "Portability". In our study, we provide a classification of the 103 papers with regard to the ISO 25010 characteristics in order to identify which characteristic is mostly used with which software architecture. The detailed classification of the used 103 papers regarding the software architecture type and the ISO 25010 quality characteristics is provided in Table 5 (see Appendix 1). Moreover, from Tables 6, 7, 8, 9, 10, 11, 12, 13 we provide the classification of the used 103 papers on the basis of the software architecture type and ISO 25010 quality sub-characteristics (see Appendix 2). Finally, Table 14 presents the interest of the selected papers on other quality attributes un-supported directly by ISO 25010 quality model such as scalability, flexibility, etc. (see Appendix 2).

For this purpose, we identified the following keywords:

- ISO 25010 quality characteristics; and
- ISO 25010 quality sub-characteristics.

The above keywords are then classified into (i) Primary keywords: ISO 25010 quality characteristics; and (ii) Secondary keywords: ISO 25010 quality sub-characteristics. A scan of the selected papers showed that some studies limited on one quality characteristic, As an example, (Ahmed and Wu 2013) focused on the "Reliability" for SOA. However, many researchers covered more than one characteristic, such as "Maintainability" and "Compatibility" for MDA as provided by (Espinha et al. 2012). And, other researchers focused on sub-characteristics, such as "Accessibility" as provided by (Hebiri et al. 2010).

The objective of step 2 is to classify the selected papers according to ISO 25010 quality characteristics for each software architecture identified in step 1 (see Appendix 2). These classifications showed that most of the selected papers focused on "Performance Efficiency" and "Maintainability", while other characteristics are rarely studied such as "Usability" and "Functional Suitability". As an example, we present a classification of the selected papers using ISO 25010 quality characteristics with SOA. The lattice in Fig. 4 presents the relationships between ISO 25010 quality characteristics and SOA.

From Fig. 4, we note that:

- "Performance Efficiency" of SOA (43.21%);
- "Security" of SOA (37.04%);
- "Portability" of SOA (7.41%); and
- "Performance Efficiency" and "Maintainability" are used together (4.94%).

Looking across these percentages, we find that most research using SOA focused mostly on the following quality characteristics: "Performance Efficiency", and "Security". Indeed, a quality characteristic, which is widely used with a specific architecture, does not mean that it is an advantage. In fact, "Security" is used with SOA by 37.04%. Although "Security" is a major concern for SOA, it represents one of the most problems in SOA environment. It makes a negative impact on the other ISO 25010 quality characteristics and sub-characteristics (such as, performance efficiency, interoperability and modifiability). It is worth noting that, during this step of our study, different terminologies are used to represent the same concept (e.g., changeability and modifiability), or by contrast (e.g., stability and instability). At the end of this step, we excluded the papers that did not include one of the primary keywords (i.e. one of the ISO 25010 quality characteristics), which left us with 75 papers.

Table 1 presents a comparison between the identified software architectures (step 1) in terms of their coverage of the ISO 25010 sub-characteristics. In Table 1, "+" in the box indicates the presence of the related sub-characteristic in the corresponding software architecture; "−" indicates the absence of the related sub-characteristic in the corresponding software architecture; or the presence of the corresponding sub-characteristic may lead to a problem, e.g., a bad impact on another sub-characteristic or an effort is required to ensure it. And, "0" indicates that the sub-

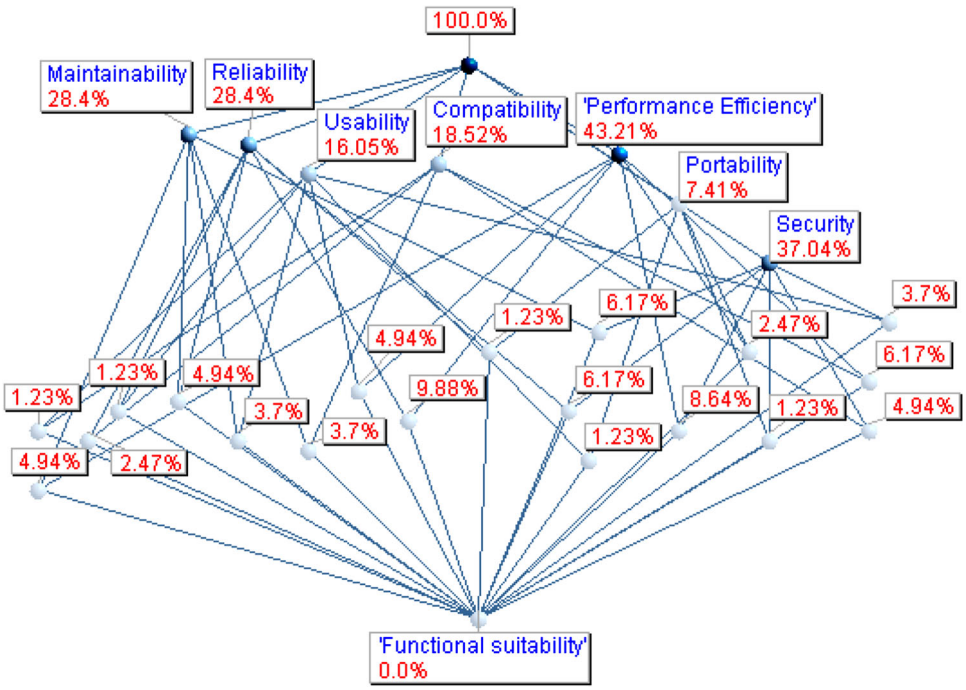

Fig. 4 Concept lattice for SOA with ISO 25010 quality characteristics

Table 1 Comparison between software architectures in terms of ISO 25010 sub-characteristics

ISO 25010 characteristics	ISO 25010 sub-characteristics	SOA	MDA	CBA	EDA	AOA
Functional Suitability	Functional Completeness	**0**	**0**	+	**0**	**0**
	Functional Correctness	+	**0**	+	**0**	+
	Functional appropriateness	**0**	**0**	**0**	**0**	**0**
Reliability	Maturity	−	**0**	**0**	–	**0**
	Availability	**0**	+	+	−	+
	Fault Tolerance	−	+	+	+	+
	Recoverability	**0**	**0**	+	**0**	+
Usability	Appropriateness Recognizability	+	**0**	**0**	**0**	**0**
	User interface aesthetics	−	+	**0**	**0**	**0**
	Operability	**0**	+	**0**	**0**	**0**
	User error protection	−	**0**	**0**	**0**	**0**
	Learnability	−	**0**	**0**	**0**	**0**
	Accessibility	+	+	**0**	+	**0**
Compatibility	Coexistence	+	+	**0**	**0**	**0**
	Interoperability	+	+	+	+	+
Performance efficiency	Time Behaviour	−	**0**	+	+	**0**
	Resource Utilization	+	+	**0**	+	+
	Capacity	−	−	−	+	+
Portability	Adaptability	+	+	+	+	+
	Instability	−	−	−	**0**	+
	Replaceability	+	+	+	**0**	**0**
Security	Confidentiality	−	−	**0**	−	**0**
	Integrity	−	+	+	−	**0**
	Non-repudiation	**0**	−	**0**	**0**	**0**
	Accountability	**0**	**0**	**0**	**0**	**0**
	Authenticity	−	−	**0**	**0**	**0**
Maintainability	Modularity	+	+	+	+	+
	Reusability	+	+	+	+	−
	Analyzability	+	**0**	+	**0**	**0**
	Modifiability	+	+	+	−	+
	Testability	−	**0**	−	**0**	**0**

characteristic is not taken into consideration. In the following sub-sections, we present an analysis of the results provided in Table 1.

Quality sub-characteristics addressed by SOA: The SOA quality has been widely investigated in the literature. For example: a "+" for "Interoperability" and "Reusability" means that these two sub-characteristics are positively impacted with SOA. (O'Brien et al. 2005) mentioned that SOA "provides important advantages like interoperability, reusability and flexibility". A "−" for "Testability" because testing in SOA is "a challenging task due to the high dynamism, the low coupling of services" (Hebiri et al. 2010). (Espinha et al. 2012) mentioned that "testability can be negatively impacted when using a SOA due to the complexity of the testing services that are distributed across a network". In fact, several studies focused on the improvement of "Testability" in SOA environment, such as the study proposed by (Hebiri et al. 2010); they proposed to improve Web service testability by developing Web services with built-in structural testing capabilities.

Quality sub-characteristics addressed by MDA: The use of MDA can improve the "Portability" since it offers a separation between the business model and implementation technique. A "+" is attributed for "Adaptability", because many researchers considered that the use of MDA improve the "Adaptability" and reduce the development costs and

Table 2 ISO 25010 quality characteristics relationships

	Functional suitability	Performance efficiency	Usability	Compatibility	Reliability	Security	Maintainability	Portability
Functional suitability	+	−	+	0	+	−	+	0
Performance efficiency	0	+	−	−	0	−	−	−
Usability	+	±	+	0	+	0	±	0
Compatibility	0	0	0	+	0	−	±	+
Reliability	+	0	+	0	+	0	+	0
Security	0	−	−	−	+	+	0	0
Maintainability	+	−	0	+	±	±	+	+
Portability	0	−	0	+	0	0	+	+

effort (*c.f.* (Eler et al. 2010; and Linehan and Clarke 2012)). On the other hand, most of the studies focus on "Maintainability" use the sub-characteristic "Reusability". A "+" is attributed to "Reusability". Indeed, MDA is "favorable for improving software model reusability and system maintainability" (O'Brien et al. 2005).

Quality sub-characteristics addressed by CBA: Table 1 illustrates that researchers argue that the use of CBA supports "Reusability", "Modifiability" and "Adaptability". As an example, a "+" is attributed to "Analyzability" and "Functional correctness" because many researchers mentioned that structuring code into components has a good benefit on "Analyzability" and "Functional correctness" (*c.f.* (Magableh and AlBeiruti 2012)). On the other hand, a "0" is attributed to most of the sub-characteristics of "Security" indicates that CBA does not take into consideration these sub-characteristics.

Quality sub-characteristics addressed by EDA: As provided in Table 1, a "+" for "Modularity" and "Adaptability" indicates that using EDA aims at "improving the modularity and flexibility of automation software with satisfactory control performance" (Zhang et al. 2010). On the other hand, a "−" is attributed to "Availability", "Confidentiality" and "Integrity" because researchers considered that working with EDA may produce problems that affect the "Availability" and many security aspects, such as "Confidentiality" and data "Integrity" (Zhang et al. 2010).

Quality sub-characteristics addressed by AOA: Several researchers used AOA, since it can "improve the modularity of the project" (Wang et al. 2010). For that reason, a "+" is attributed to "Modularity". Therefore, a "+" for "Functional Correctness", because using AOA is "an effective way of improving the reliability and correctness of complex system" (de Oliveira and Soares 2012). In addition, a "+" is attributed for "Capacity" because many researches mentioned that using "aspect-oriented software development improve the system evolution capacity", since "the system evolution can be realized at run-time" (Fan et al. 2012).

4.3 Step 3: Relationships among quality characteristics

Identifying the relationships among quality characteristics is an important subject that has been widely discussed in the literature. In fact, understanding the relationships between quality characteristics is important to sustain a sufficient level of system/software quality and its development process. In this section, we identify the relationships among ISO 25010 quality characteristics. For this purpose, we examined a set of studies that investigated these relationships. Many studies provide the relationships by using the experience-based approach (Al-Daajeh et al. 2012). While, other studies identified the relationships between quality characteristics by combining different views, such as academic, industry, and literature. In this paper, our analysis is based on surveying a set of research papers. We are not limited on the collected papers from Step 1, but we used other papers focused on analyzing the relationships between quality attributes such as the study conducted by (Delač 2012).

(Delač 2012) focused on the ISO 9126 quality attributes relationships for Web-based application. Among six quality characteristics of ISO 9126, they identified 15 relationships. These relationships can be either "Positive", "Negative", or "Independent". In our case, we focused on eight quality characteristics derived from ISO 25010. Evidently some quality characteristics relationships are common with (Delač 2012), since ISO 9126 and ISO 25010 has six common quality characteristics. The main difference between ISO 9126 and ISO 25010 is given by

the addition of "Security" and "Compatibility" characteristics. Actually, "Security" in ISO 25010 is a characteristic and a sub-characteristic in ISO 9126. Thus, we identified 28 relationships. As provided in Table 2, there are four types of relationships:

- A positive relationship "+", i.e. a good value of one quality characteristic result in a good value of the other. As an example, a "+" between "Security" and "Reliability" means that "Security" has a positive impact on the "Reliability".
- A negative relationship "−", *i.e.* a good value of one quality characteristic result in a bad value of the other. As an example a "-" between "Security" and "Performance Efficiency", because "Security mechanisms often have a negative impact on performance" (O'Brien et al. 2005).
- "±", i.e. a good value on a quality sub-characteristic and a bad value on another quality sub-characteristic for the same quality characteristic. As an example: a "±" between "Security" and "Maintainability" because "Security mechanisms often have a negative impact on modifiability" (O'Brien et al. 2005).
- Independent relationship "0", *i.e.* a number of quality characteristics that does not affect each other. As an example, a "0" between "Usability" and "Security" indicates that there is no relationship between these two quality characteristics.

In Table 1, we noted that SOA does not support "Performance Efficiency" quality characteristic. On the other hand, this architecture offers the "Maintainability". For instance, Table 2 illustrates that "Performance Efficiency" and "Maintainability" have a negative relationship, since the addition or the modification of a Functional User Requirement may have a negative effect on the "Performance Efficiency". For this reason, only 4.94% studying SOA used "Maintainability" and "Performance Efficiency" together. Moreover, "Maintainability" and "Reliability" have a positive relationship. Thus, (Loganathan and Gandhi 2015) proposed to minimize the cost of "Maintainability" based on a set of "Reliability" constraint.

4.4 Guideline for Selecting the Suitable SA Based on ISO 25010 Quality Characteristics

Based on the analysis of Table 1 and Table 2, we propose a guideline as provided in Table 3 to select the appropriate software architecture on the basis of the ISO 25010 quality characteristics. The general phases of the guideline are provided as follow:

1. Identify the required quality characteristics according to the Non-Functional Requirements.
2. Verify the relationships between the required quality characteristics as we provided in Table 2.
3. Select the appropriate software architecture as provided in Table 3 according to the desired quality characteristics.

Identifying the required quality characteristics: The customer/user specifies the quality characteristics about the delivered software product. One or more quality characteristics can be identified. The selection of the SA related to one quality characteristic is the simplest. In fact, software architect can select the first choice from Table 3. As an example if the software architect is looking for "Maintainability" quality characteristic, he may select firstly MDA architecture. Secondly he may choose between SOA and CBA architectures. Finally, software architect can choose between EDA and AOA architectures.

Quality characteristics relationships: In the case where customer/user is looking for more than one quality characteristics, software architect should take into consideration the relationships between the required characteristics. According to the customer/user needs, software architect must decide which software architecture to prioritize and which one to forsake. The selection of the quality characteristics should ensure that the selected software architecture is appropriate and it contributes to the customer/user needs. As an example, the relationship between "Maintainability" and "Performance efficiency" quality characteristics is a *negative* relationship. Then, the software architect should choose which characteristic to prioritize. If he selects "Maintainability" then he can use MDA as the

Table 3 SA selection guideline

	Functional suitability	Performance efficiency	Usability	Compatibility
1st choice	CBA	EDA	MDA	SOA|MDA
2nd choice	SOA|AOA	AOA	EDA	CBA|EDA|AOA
3rd choice	MDA|EDA	MDA|CBA	CBA|AOA	
	Reliability	Security	Maintainability	Portability
1st choice	CBA|AOA	CBA	MDA	AOA
2nd choice	MDA	AOA	SOA|CBA	EDA
3rd choice	EDA	EDA	EDA|AOA	SOA|MDA|CBA

first choice, then SOA or CBA, and finally EDA or AOA. Whereas, if the software architect selects to prioritize "Performance efficiency" then he can use EDA as the first choice, then AOA as the second choice, then MDA or CBA, and finally SOA.

SA selection: In this step, software architect can choose the appropriate software architecture according to their needs in terms of quality characteristics and by referring to Table 3. Here, we provide three choice-levels of granularity for software architecture (first, second, and third choice).

5 Threats to validity

In this section, we discuss threats to validity of the herein presented study. These include: internal, external, and construct validity threats.

Internal validity: is related to the limited number of papers, the selected list of keywords and the limitations of the employed databases. To decrease the risk of the incompleteness of this research, we were not limited only on the identified keywords, but we included some papers that do not include explicitly primary keywords. As an example, for the first step, the primary keyword was "Software Architecture". Whereas, some papers use one specified software architecture type were included in this research even if they do not include the primary keyword "Software Architecture". In addition, in order to omit the limitations implied by employing a particular database, we used the most relevant databases (IEEE Xplore, ACM Digital Library, Science Direct and CiteSeer). Moreover, collected papers are from different conferences, workshops and journal and include different domains such as quality characteristics, software architecture evaluation, etc. Another important issue is the use of quality characteristics as provided by ISO 25010 quality model. In fact, we noted also the use of different terminologies to represent the same ISO 25010 quality characteristic or sub-characteristic. Indeed, this can lead to high level of ambiguity in the domain of software architecture quality. Thus, we propose to substitute the appropriate ISO 25010 quality model terms. As an example, some researchers used the term "Changeability" to reflect the meaning of the ISO 25010 sub-characteristic "Modifiability". Moreover, researches tend to define their quality attributes individually rather than rely on a standard. To overcome this problem, we try to provide a classification of the selected papers based on ISO 25010 quality characteristics, sub-characteristics, and also the quality attributes most used which seem not supported by ISO 25010 quality model. However, the results of this study remain the same even if the researchers used different terminologies. Consequently, where applicable, this guideline will greatly help to avoid many of the ambiguities found so often in software architecture fields.

Table 4 Examples of the survey's results

	Project 1	Project 2	Project 3	Project 4
The used SA				
SOA	X	X	X	X
CBA				
MDA				
AOA				
EDA				
ISO 25010 quality characteristics				
Functional suitability	B	B	B	B
Performance efficiency	C	C	C	C
Usability	C	E	C	E
Compatibility	D	E	E	E
Reliability	D	D	C	D
Security	C	C	C	B
Maintainability	B	B	B	B
Portability	E	E	E	E

A very good *B* rather good *C* average *D* poor *E* very low

External validity: is concerned with the possibility to generalize the results of the study. Due to the limited number of papers for this study, results might be difficult to generalize. However, the provided guideline was established through an analytical survey of 113 scientific papers. Selected papers are not only from academic view but also from industrial experiences. In addition, and in order to make sure about the validation of the proposed guideline, a survey with industrial experts is in progress. We collected data from two companies working in the software development field (ST2i and Telnet). Each company provides, for a set of software projects, the required ISO 25010 quality characteristics and the used software architecture in each project (see Table 4). After data collection, we determine the extent to which the proposed guideline is appropriate; we compare the used architecture (in the company) and the determined architecture (from the guideline). Until now, we noted that software architecture the most used in the industry is SOA (as it is identified in the literature). And the most required quality characteristics are "Security" and "Reliability" while the less used quality characteristic is "Portability". This investigation delivers notably encouraging results. However, providing

more detailed data from other companies will be needed regarding the guideline validity.

Construct validity: is related to the relation between theory and observation. In fact, the selected papers for this study were collected by our self. Moreover, we do not judge the papers' pertinence. However, we used multiple source and different domains. Thus, we thought that the quality of the selected papers can improve the results of this study. Another issue is related to the problem of the quality of search engines which may influence the completeness of the pertinent papers selected for this study. In fact, since our selection is based on a set of primary keyword, we possibly missed some other studies where their authors used other terms to represent a quality characteristic.

6 Conclusion

Software quality characteristics should be taken into consideration during all the SDLC phases to guarantee the quality of the final product. Selecting the suitable software architecture that satisfy the Non-Functional Requirements as specified by the customers/users is a challenge task, because of the various software architectures proposed in the literature. In this paper, we proposed a guideline for software architecture selection based on ISO 25010 quality characteristics. For this purpose, we followed three steps. In the first step, we identified architectures the most used in the literature. We presented the results of a literature review using a Formal Concept Analysis of 113 papers collected from four databases (IEEE Xplore, ACM Digital Library, Science Direct and CiteSeer). The investigation of these papers allows us to determine which software architecture used the most in the software engineering literature. In the second step, we proposed a classification of the identified software architecture according to the ISO 25010 quality characteristics. In the third step, we analyzed the relationships between the different ISO 25010 quality characteristics.

As a result, we identified in this research study five architectures: SOA, CBA, MDA, EDA, and AOA. Among these popular architectures, we found that SOA was the most popular. Then, we classified these architectures according to the ISO 25010 quality characteristics. We highlighted which quality characteristic is used the most by each software architecture. Furthermore, we focused on the relationships between the quality characteristics supported by ISO 25010. Finally, we proposed a guideline to help software architects in choosing the appropriate architecture according to their needs in terms of the ISO 25010 quality characteristics following the three identified steps.

Based on this study, we suggest that further research should be undertaken to select the appropriate architecture that supports requirements changes in terms of ISO 25010 quality characteristics. In an environment where the requirements are likely to change, we suggest to analyze how some of the ISO 25010 quality characteristics support the extensibility of user requirements. The limit of this study is that it takes into account only two quality characteristics. A possible extension can be proposed to include more than two quality characteristics.

Acknowledgements We would like to record our thanks to the companies: Telnet in Sfax-Tunisia and ST2i in Tunis-Tunisia, for their participation in the investigation and for sharing their perspectives and experiences.

Appendix 1

Table 5 presents the classification of the selected papers for this study in respect to the software architecture type and the ISO 25010 quality characteristics.

Table 5 ISO 25010 Quality characteristics for SA type-Summary of quantitative results

SA	Qte							
	Functional suitability	Performance efficiency	Usability	Compatibility	Reliability	Security	Maintainability	Portability
MDA	(Linehan and Clarke 2012)	(Matković and Fertalj 2012), (Bocciarelli and D'Ambrogio 2014), (Hoisl et al. 2014), (Ren 2011), (Magableh and AlBeiruti 2012)			(Linehan and Clarke 2012), (Hoisl et al. 2014), (Bocciarelli and D'Ambrogio 2014),(Ren 2011), (Gui and Luo 2012)	(Fang et al. 2010), (Borek et al. 2012), (Hoisl et al. 2014), (Ren 2011), (Magableh and AlBeiruti 2012), (Kang and Liang 2013), (Matković and Fertalj 2012)	(Cheng 2010), (Fang et al. 2010), (Yuan and Watton 2012), (Linehan and Clarke 2012), (Magableh and AlBeiruti 2012), (HaiTao and Wei 2010)	(Linehan and Clarke 2012), (Bispo et al. 2010), (Tielin et al. 2010)
SOA		(Shanmugasundaram et al. 2012), (Yue et al. 2010), (Delač 2012), (Lee and Kim 2010), (Jehan et al. 2013), (Potena 2013), (Raafat and Cecelja 2011), (Eler et al. 2010), (Zheng and Lyu 2010), (Dwornikowski et al. 2011), (Waris et al. 2013), (Mezghani and Ben Halima 2012), (Babamir and Arabfard 2012), (Massoud and Dumke 2012), (EL Yamany et al. 2010), (Trilles et al. 2013), (Matković and Fertalj 2012), (Chen and Tang 2013), (Ozkaya et al. 2010), (Yue and Tao 2012), (Hoisl et al. 2014), (Tragatschnig and Zdun 2013)	(Shanmugasundaram et al. 2012), (Waris et al. 2013), (Raafat and Cecelja 2011), (Massoud and Dumke 2012), (White et al. 2012), (Cheaito et al. 2010)	(Massoud and Dumke 2012), (EL Yamany et al. 2010), (Espinha et al. 2012), (Chen and Tang 2013), (Jehan et al. 2013), (Ozkaya et al. 2010)	(Delač 2012), (Ahmed and Wu 2013), (Dwornikowski et al. 2011), (Zheng and Lyu 2010), (Potena 2013), (Waris et al. 2013), (Al Helal and Gamble 2014), (Zhou et al. 2010), (Chen and Tang 2013), (Hu et al. 2011), (Yang et al. 2013), (Rafea and Mahdianb 2011), (Hoisl et al. 2014)	(Yue and Tao 2012), (Hu et al. 2011), (Georgieva and Goranova 2011), (EL Yamany et al. 2010), (Waris et al. 2013), (Massoud and Dumke 2012), (Zhou et al. 2010), (Matković and Fertalj 2012), (Felhi and Akaichi 2013), (Mircea 2011), (Rafea and Mahdianb 2011), (Ozkaya et al. 2010), (Hoisl et al. 2014), (Borek et al. 2012), (Yuan and Watton 2012), (Trinugroho et al. 2013), (Cheaito et al. 2010)	(Fan et al. 2012), (White et al. 2012), (Espinha et al. 2012), (Waris et al. 2013), (Zhou et al. 2010), (Shanmugasundaram et al. 2012), (Sneed et al. 2013), (Hu et al. 2011), (Ozkaya et al. 2010), (Yuan and Watton 2012), (Yue et al. 2010), (Li et al. 2013), (Tragatschnig and Zdun 2013)	(Zheng and Lyu 2010), (Shanmugasundaram et al. 2012), (Mircea 2011), (Bispo et al. 2010)

Table 5 continued

SA	Qte							
	Functional suitability	Performance efficiency	Usability	Compatibility	Reliability	Security	Maintainability	Portability
EDA		(Pang et al. 2014), (Hammami et al. 2012), (Tran et al. 2011), (Zappia et al. 2012), (Nguyen and Thoai 2012), (Klatt et al. 2011), (Tragatschnig and Zdun 2013), (Li et al. 2013)		(Pang et al. 2014)	(Hammami et al. 2012; Klatt et al. 2011), (Li et al. 2013)	(Hammami et al. 2012), (Yuan and Watton 2012), (Trinugroho et al. 2013), (Klatt et al. 2011), (Zappia et al. 2012)	(Zappia et al. 2012), (Klatt et al. 2011), (Tragatschnig and Zdun 2013), (Li et al. 2013)	(Zappia et al. 2012)
AOA	(Linehan and Clarke 2012)	(Wang et al. 2010), (de Oliveira and Soares 2012), (Wen et al. 2013), (Magableh and AlBeiruti 2012)	(de Oliveira and Soares 2012)		(Klatt et al. 2011; Linehan and Clarke 2012), (de Oliveira and Soares 2012), (Wen et al. 2013)	(de Oliveira and Soares 2012; Fang et al. 2010), (Magableh and AlBeiruti 2012), (Wang et al. 2010), (Molesini et al. 2010)	(de Oliveira et al. 2013), (Fang et al. 2010), (Linehan and Clarke 2012), (Wang et al. 2010), (de Oliveira and Soares 2012; Tizzei et al. 2011), (Magableh and AlBeiruti 2012), (Abdelmoez et al. 2012)	(Linehan and Clarke 2012)
CBA	(Fuentes-Fernández et al. 2012)	(Yue et al. 2010), (Klatt et al. 2011), (Kounev et al. 2013), (Magableh and AlBeiruti 2012), (Bures et al. 2011), (Smith and Llado 2011)	(Cheaito et al. 2010)	(Brada 2011), (Snajberk et al. 2013)	(Yue et al. 2010), (Klatt et al. 2011), (Tyagia and Sharmab 2014), (Brosch et al. 2010), (Dormoy et al. 2012), (Pham and Defago 2013)	(Klatt et al. 2011), (Magableh and AlBeiruti 2012), (Cheaito et al. 2010), (Du et al. 2013)	(Yue et al. 2010), (Klatt et al. 2011), (Tizzei et al. 2011), (Magableh and AlBeiruti 2012)	

Appendix 2

Table 6 identifies the usability of the sub-characteristics of "Functional suitability" which is being the most rarely used quality characteristic.

Table 7 presents the classification of the sub-characteristics of the quality characteristic "performance efficiency". "Capacity" is the widely used sub-characteristic, whereas "time behaviour" is the rarely used sub-characteristic.

Table 8 provides a classification of sub-characteristics of the quality characteristic "Usability". Most of the sub-characteristics of "Usability" are rarely used in the. "Accessibility" is the most used sub-characteristic.

Table 9 identifies the usefulness of the sub-characteristics of the "Compatibility" quality characteristic. "Interoperability" is one of the important advantages of SOA. This is why most researches focused on "Interoperability" used by SOA.

As presented in Table 10, the sub-characteristics of "Reliability" have been widely used in the literature especially for SOA, and "Availability" is the most used sub-characteristic. In addition, many researchers mentioned that "SOA introduces more challenges to achieve: data

Table 6 Functional Suitability sub-characteristics for SA type-Summary of the quantitative results

Functional Suitability	SOA	MDA	CBA	EDA	AOA
Functional completeness					
Functional correctness	Matković and Fertalj (2012), Delač (2012), Jehan et al. (2013)	Matković and Fertalj (2012)	Bures et al. (2011), Oster (2013)		Wen et al. (2013)
Functional appropriateness					

Table 7 Performance Efficiency sub-characteristics for SA type-Summary of the quantitative results

Performance efficiency	SOA	MDA	CBA	EDA	AOA
Time behaviour			Sirer et al. (2011)		
Resource utilization		Kang et al. (2011)		Nguyen and Thoai (2012)	
Capacity	(Mezghani and Ben Halima 2012), Quintero et al. (2010), Lee and Kim 2010), Hu et al. (2011), Zhou et al. (2010), Yang et al. (2013)	Quintero et al. (2010), Linehan and Clarke (2012), Klatt et al. (2011), Qazi et al. (2013)	(Mezghani and Ben Halima 2012)	Klatt et al. (2011), (Hammami et al. 2012)	Linehan and Clarke (2012), (Wang et al. 2010)

Table 8 Usability sub-characteristics for SA type-Summary of the quantitative results

Usability	SOA	MDA	CBA	EDA	AOA
Appropriateness Recognizability					
User interface aesthetics		Akiki et al. (2015)			
Operability	Chen and Tang (2013), Matković and Fertalj (2012)	Matković and Fertalj (2012)			
User error protection					
Learnability					
Accessibility	Babamir and Arabfard (2012), Trilles et al. 2013), Mircea (2011), Rafea and Mahdianb (2011), Salva and Rabhi (2010), Alferez and Pelechano (2011), Delač (2012)	(Hebiri et al. 2010), Laabidi and Jemni (2010)	Pordel et al. (2011)	Hammami et al. (2012)	

Table 9 Compatibility sub-characteristics for SA type-Summary of the quantitative results

Compatibility	SOA	MDA	CBA	EDA	AOA
Interoperability	(Trilles et al. 2013), (Yuan and Watton 2012), (Felhi and Akaichi 2013), (Yue and Tao 2012), (White et al. 2012), (Espinha et al. 2012), (Boukhedouma et al. 2013), (Matković and Fertalj 2012), (Jehan et al. 2013), (Chen and Tang 2013), (Mircea 2011), (Yang et al. 2013), (Raafat and Cecelja 2011), (EL Yamany et al. 2010), (Ozkaya et al. 2010), (Bispo et al. 2010), (Georgieva and Goranova 2011), (Massoud and Dumke 2012), (Waris et al. 2013)	(Matković and Fertalj 2012), (Yang et al. 2013), (Bispo et al. 2010)	(Smith and Llado 2011), (Labejof et al. 2012), (Jurado et al. 2012)	(Hammami et al. 2012), (Pang et al. 2014), (Zappia et al. 2012), (Yuan and Watton 2012)	(de Oliveira and Soares 2012)
Coexistence					

Table 10 Reliability sub-characteristics for SA type-Summary of the quantitative results

Reliability	SOA	MDA	CBA	EDA	AOA
Maturity	(Trinugroho et al. 2013), (Mircea 2011), (Massoud and Dumke 2012), (Welke et al. 2011), (Espinha et al. 2012), (Waris et al. 2013), (Shanmugasundaram et al. 2012)			(Trinugroho et al. 2013)	
Availability	(Rafea and Mahdianb 2011), (Mezghani and Ben Halima 2012), (Delač 2012), (Mircea 2011), (Matković and Fertalj 2012), (Lee and Kim 2010), (Hu et al. 2011), (Zhou et al. 2010), (Yang et al. 2013), (Chen and Tang 2013), (Babamir and Arabfard 2012), (Raafat and Cecelja 2011), (Espinha et al. 2012), (Waris et al. 2013), (Potena 2013), (Shanmugasundaram et al. 2012), (Eler et al. 2010), (Zheng and Lyu 2010), (Dwornikowski et al. 2011), (Al Helal and Gamble 2014), (EL Yamany et al. 2010), (Pordel et al. 2011), (Sun et al. 2011)	(Magableh and AlBeiruti 2012), (Matković and Fertalj 2012), (Laabidi and Jemni 2010)	(Magableh and AlBeiruti 2012), (Mezghani and Ben Halima 2012), (Dormoy et al. 2012), (Leger et al. 2010)	(Hammami et al. 2012), (Pang et al. 2014), (Zappia et al. 2012), (Li et al. 2013)	(Wang et al. 2010), (Magableh and AlBeiruti 2012)
Fault tolerance	(Delač 2012), (Rafea and Mahdianb 2011), (Babamir and Arabfard 2012), (Hu et al. 2011), (Dwornikowski et al. 2011), (Al Helal and Gamble 2014)	(Gui and Luo 2012)	(Pham and Defago 2013), (Wu et al. 2012), (Wolf et al. 2011), (Staroswiecki 2010)	(Li et al. 2013)	
Recoverability			(Staroswiecki 2010)		

Table 11 Portability sub-characteristics for SA type-Summary of the quantitative results

Portability	SOA	MDA	CBA	EDA	AOA
Instability	(Jehan et al. 2013), (Massoud and Dumke 2012), (Ozkaya et al. 2010)		(Tizzei et al. 2011)		(Molesini et al. 2010), (Tizzei et al. 2011)
Adaptability	(Yuan and Watton 2012), (Cheaito et al. 2010), (Mircea 2011), (Quintero et al. 2010), (Yang et al. 2013), (Felhi and Akaichi 2013), (Shanmugasundaram et al. 2012), (Felhi and Akaichi 2013)	(Magableh and AlBeiruti 2012), (Quintero et al. 2010), (Yang et al. 2013)	(Magableh and AlBeiruti 2012), (Cheaito et al. 2010)	(Yuan and Watton 2012)	(de Oliveira and Soares 2012), (Wang et al. 2010), (Magableh and AlBeiruti 2012)
Replaceability	(Al Helal and Gamble 2014)				

Table 12 Security sub-characteristics for SA type-Summary of the quantitative results

Security	SOA	MDA	CBA	EDA	AOA
Confidentiality	(Borek et al. 2012), (Hu et al. 2011), (Georgieva and Goranova 2011), (Matković and Fertalj 2012), (Mircea 2011), (Yang et al. 2013), (Yue and Tao 2012), (Shanmugasundaram et al. 2012), (Hoisl et al. 2014), (Boukhedouma et al. 2013)	(Borek et al. 2012), (Matković and Fertalj 2012), (Hoisl et al. 2014)		(Hammami et al. 2012)	
Integrity	(Mircea 2011), (Ozkaya et al. 2010), (Georgieva and Goranova 2011), (Waris et al. 2013), (Borek et al. 2012), (Hu et al. 2011), (Massoud and Dumke 2012), (Yue and Tao 2012), (Matković and Fertalj 2012), (Yang et al. 2013), (Babamir and Arabfard 2012), (Hoisl et al. 2014)	(Borek et al. 2012), (Matković and Fertalj 2012), (Hoisl et al. 2014), (Wang et al. 2013)		(Hammami et al. 2012)	
Non-repudiation	(Borek et al. 2012), (Hu et al. 2011)	(Borek et al. 2012)			
Accountability					
Authenticity	(Borek et al. 2012), (Georgieva and Goranova 2011), (Hu et al. 2011), (Matković and Fertalj 2012), (Felhi and Akaichi 2013), (Yue and Tao 2012), (EL Yamany et al. 2010), (Felhi and Akaichi 2013), (Georgieva and Goranova 2011), (Yue and Tao 2012)	(Borek et al. 2012), (Matković and Fertalj 2012)			

availability and reliability". This is especially the case for the availability of complex services in environment where concurrent requests should be satisfied.

Table 11 presents the interest of the selected papers in the "Portability" sub-characteristics. Most of the selected papers focused on the "Adaptability" sub-characteristic used by SOA.

Table 12 presents the classification of the selected papers based on the "Security" sub-characteristic which, as expected, has been widely used with SOA because "Security" is a critical problem in SOA. Researchers proposed the use of "application-independent protocols to develop a secure SOA" (e.g. Transport Layer Security "TLS" or Web service security protocols). These protocols can guarantee two sub-characteristics: "Confidentiality" and "Integrity".

As shown in Table 13, the sub-characteristics for "Maintainability" have been widely discussed in the literature in comparison with other sub-characteristics. In addition, most of the SOA-based papers focused on Reusability and Testability. In fact, Reusability is a main advantage of SOA due to the independence between the various services, which makes the modification or the replacement of a service by another an easy task. However, testing SOA applications is a challenging task due to the high dynamism, the low coupling and the low testability of services in isolation. This in fact prompted some approaches to investigate how to improve SOA testability.

Table 14 presents the interest of the selected papers on other quality attributes un-supported by ISO 25010 such as scalability, flexibility, etc.

Table 13 Maintainability sub-characteristics for SA type-Summary of the quantitative results

Maintainability	SOA	MDA	CBA	EDA	AOA
Modularity	(Yang et al. 2013), (Yuan and Watton 2012), (Yue et al. 2010), (Zhou et al. 2010), (Mircea 2011), (Shanmugasundaram et al. 2012), (Dwornikowski et al. 2011)	(Yang et al. 2013), (Linehan and Clarke 2012), (Magableh and AlBeiruti 2012)	(Tizzei et al. 2011), (Magableh and AlBeiruti 2012), (Lee et al. 2010), (Yue et al. 2010)	(Pang et al. 2014), (Zappia et al. 2012), (Yuan and Watton 2012)	(de Oliveira and Soares 2012), (Tizzei et al. 2011), (Linehan and Clarke 2012), (Magableh and AlBeiruti 2012), (Molesini et al. 2010)
Reusability	(Mircea 2011), (Shanmugasundaram et al. 2012), (Waris et al. 2013), (Yue et al. 2010), (Trilles et al. 2013), (Tragatschnig and Zdun 2013), (Quintero et al. 2010), (Jehan et al. 2013), (Zhou et al. 2010), (Chen and Tang 2013), (Raafat and Cecelja 2011), (Ozkaya et al. 2010), (Bispo et al. 2010), (EL Yamany et al. 2010)	(Linehan and Clarke 2012), (Quintero et al. 2010), (Bispo et al. 2010)	(Tizzei et al. 2011), (Lee et al. 2010), (Yue et al. 2010), (Snajberk et al. 2013)	(Pang et al. 2014), (Tragatschnig and Zdun 2013), (Minguez et al. 2011)	(de Oliveira and Soares 2012), (Tizzei et al. 2011), (Linehan and Clarke 2012), (Molesini et al. 2010), (Wang et al. 2010)
Analyzability			(Bures et al. 2011)		
Modifiability	(Babamir and Arabfard 2012), (Massoud and Dumke 2012), (Ozkaya et al. 2010), (Shanmugasundaram et al. 2012)	(Magableh and AlBeiruti 2012)	(Magableh and AlBeiruti 2012)		(de Oliveira and Soares 2012), (Magableh and AlBeiruti 2012), (Molesini et al. 2010)
Testability	(Delač 2012), (Pordel et al. 2011), (Sun et al. 2011), (Eler et al. 2010) (Shanmugasundaram et al. 2012), (Waris et al. 2013)				

Table 14 Quality attributes (uncovered by ISO/IEC 25010) for SA type-Summary of the quantitative results

	SOA	MDA	CBA	EDA	AOA
Scalability	(Yue et al. 2010), (Matković and Fertalj 2012), (Jehan et al. 2013), (Lee and Kim 2010), (Hu et al. 2011), (Zhou et al. 2010), (Chen and Tang 2013), (Yang et al. 2013), (Felhi and Akaichi 2013), (Ozkaya et al. 2010), (Yue and Tao 2012), (Waris et al. 2013), (Dwornikowski et al. 2011), (Felhi and Akaichi 2013), (Yue and Tao 2012), (Fan et al. 2012), (Ozkaya et al. 2010), (Wang et al. 2013), (Klatt et al. 2011; Kounev et al. 2013)	(Linehan and Clarke 2012), (Magableh and AlBeiruti 2012), (Matković and Fertalj 2012), (Yang et al. 2013)	(Magableh and AlBeiruti 2012), (Klatt et al. 2011), (Yue et al. 2010), (Abdellatif 2012), (Wallis et al. 2010)	(Nguyen and Thoai 2012), (Klatt et al. 2011), (Hammami et al. 2012), (Pang et al. 2014), (Zappia et al. 2012), (Li et al. 2013), (Tran et al. 2011)	(Linehan and Clarke 2012), (Magableh and AlBeiruti 2012), (Wang et al. 2010)

Table 14 continued

	SOA	MDA	CBA	EDA	AOA
Flexibility	(Rafea and Mahdianb 2011), (Waris et al. 2013), (Massoud and Dumke 2012), (Dwornikowski et al. 2011), (Tragatschnig and Zdun 2013), (Mircea 2011), (Yue et al. 2010), (Mezghani and Ben Halima 2012), (Cheaito et al. 2010), (Quintero et al. 2010), (Delač 2012), (Jehan et al. 2013), (Zhou et al. 2010), (Yang et al. 2013), (White et al. 2012), (Ozkaya et al. 2010), (Fan et al. 2012), (Ozkaya et al. 2010), (Borek et al. 2012), (Raafat and Cecelja 2011), (Felhi and Akaichi 2013), (Bispo et al. 2010), (Felhi and Akaichi 2013), (Boukhedouma et al. 2013)	(Borek et al. 2012), (Quintero et al. 2010), (Bispo et al. 2010), (Ren 2011), (Quintero et al. 2010)	(Klatt et al. 2011), (Yue et al. 2010), (Mezghani and Ben Halima 2012), (Cheaito et al. 2010), (Biggs et al. 2010), (He et al. 2010)	(Klatt et al. 2011), (Pang et al. 2014), (Tragatschnig and Zdun 2013)	(de Oliveira and Soares 2012), (Molesini et al. 2010)
Changeability	(Tragatschnig and Zdun 2013)			(Tragatschnig and Zdun 2013)	
Robustness	(Jehan et al. 2013)			(Li et al. 2013)	
Manageability	(Zhou et al. 2010)				

References

Abdellatif T (2012) Building reliable security systems: the case of an e-voting system. In: The international conference on information technology and e-Services, IEEE, Sousse Tunisia

Abdelmoez W, Khater H, El-shoafy N (2012) Comparing maintainability evolution of object-oriented and aspect-oriented software product lines. In: The 8th international conference on informatics and systems, IEEE, Cairo Egypt

Acheson P (2010) Methodology for object-oriented system architecture development. In: The 4th annual IEEE systems conference, IEEE, San Diego CA

Ahmed W, Wu YW (2013) Reliability prediction model for SOA using Hidden Markov Model. In: The 8th China grid annual conference, IEEE, Changchun

Akiki PA, Bandara AK, Yu Y (2015) Adaptive model-driven user interface development systems. ACM Comput Surv J 47(1):9

Al Helal H, Gamble R (2014) Introducing replaceability into web service composition, IEEE Trans Serv Comput

Al-Daajeh SH, Al-Qutaish RE, Al-Qirem F (2012) A tactic-based framework to evaluate the relationships between the software product quality attributes, Int J Softw Eng 5(1)5–26

Aleti A, Buhnova B, Grunske L, Koziolek A, Meedeniya I (2013) Software architecture optimization methods: a systematic literature review. IEEE Trans Softw Eng 39:658–683

Alferez GH, Pelechano V (2011) Systematic reuse of web services through software product line engineering. In: The 9th European conference on web services, IEEE, Lugano

Babamir SM, Arabfard M (2012) Improving service accessibility in service-oriented HIS. J Med Syst 36:4021–4030

Bass L, Clements P, Kazman R (2012) Software architecture in practice. Addison-Wesley Professional, Salt Lake

Biggs G, Ando N, Kotoku T (2010) Native robot software framework inter-operation. In: The 2nd international conference on simulation, modeling, and programming for autonomous robots, Darmstadt Germany, Springer, Berlin Heidelberg

Bispo CP, Maciel RSP, David JMN, Ribeiro Í, Conceição R (2010) Applying a model-driven process for a collaborative service-oriented architecture. In: The 14th international conference on computer supported cooperative work in design, IEEE, Shanghai China

Bocciarelli P, D'Ambrogio A (2014) A Model-driven method for enacting the design-time QoS analysis of business processes. Softw Syst Model J 13:573–598

Boehm BW (1978) Characteristics of software quality

Borek M, Moebius N, Stenzel K, Reif W (2012) Model-driven development of secure service applications. In: The 35th software engineering workshop, IEEE, Greece

Boukhedouma S, Oussalah M, Alimazighi Z, Tamzalit D (2013) Adaptation patterns for service based inter-organizational

workflows. In: The 7th international conference on research challenges in information science, IEEE, Paris
Brada P (2011) Enhanced type-based component compatibility using deployment context information. Electron Notes Theor Comput Sci J 279:17–31
Brosch F, Koziolek H, Buhnova B, Reussner R (2010) Parameterized reliability prediction for component-based software architectures. In: The 6th international conference on the quality of software architectures, Prague Czech Republic, Springer, Berlin Heidelberg
Bures T, Jezek, Pavel, Malohlava M, Poch T, Sery O (2011) Strengthening component architectures by modeling fine-grained entities. In: The 37th EUROMICRO conference on software engineering and advanced applications, IEEE, Oulu
Cheaito M, Laborde R, Barrère F, Benzekri A (2010) A deployment framework for self-contained policies. In: The international conference on network and service management, IEEE, Niagara Falls ON
Chen J, Tang T (2013) Research on distributed simulation framework of train control system based on SOA. In: The international conference on intelligent rail transportation, IEEE, Beijing
Chen Y, Li X, Yi L, Liu D (2010) A ten-year survey of software architecture. In: The international conference on software engineering and service sciences, IEEE, Beijing
Cheng F (2010) MDA Implementation based on patterns and action semantics. In: The 3rd international conference on information and computing, IEEE, Wuxi Jiang Su
COSMIC (2015) Guideline on Non-Functional and project requirements: How to consider non-functional and project requirements in software project performance measurement, benchmarking and estimating
Couto L, Oliveira JeN, Ferreira M, Bouwers E (2011) Preparing for a literature survey of software architecture using formal concept analysis. In: The 5th international workshop on software quality and maintainability, Oldenburg Germany
de Oliveira KS, Soares MS (2012) A systematic review on aspects in software architecture design. In: The 31st international conference of the Chilean computer science society, IEEE, Valparaiso
de Oliveira KS, Franca JMS, Soares MS (2013) Extensions of SysML for modeling an aspect oriented software architecture with multiple views. In: The 3rd international conference on information technology: new Generations, IEEE, Las Vegas NV
Dehaghani SMH, Hajrahimi N (2013) Which factors affect software projects maintenance cost more? Acta Informatica Medica 21(1):63–66
Delač G (2012) Reliability modeling for SOA systems. In: The 35th international convention MIPRO, IEEEE, Opatija, pp 847–852
Dormoy J, Kouchnarenko O, Lanoix A (2012) Using temporal logic for dynamic reconfigurations of components. In: The 7th international conference on formal aspects of component software, Guimarães Portugal. Springer, Berlin Heidelberg
Dromey RG (1996) Cornering the Chimera [software quality]. IEEE Softw 13:33–43
Du C, Li X, Shi H, Hu J, Feng R, Feng Z (2013) Architecture security evaluation method based on security of the components. In: The 20th Asia-Pacific software engineering conference, IEEE, Bangkok
Dwornikowski D, Kobusińska A, Kobusiński J (2011) Failure detection in a RESTful way. In: The 9th international conference on parallel processing and applied mathematics, Torun Poland. Springer, Berlin Heidelberg
Eler MM, Delamaro ME, Maldonado JC, Masiero PC (2010) Built-In structural testing of web services. In: The Brazilian symposium on software engineering, IEEE, Salvador Bahia
Espinha T, Chen C, Zaidman A, Gross H-G (2012) Maintenance research in SOA-towards a standard case study. In: The 16th European conference on software maintenance and reengineering, IEEE, Szeged. pp 391–396
Fan Y-H, Wu J-O, Wang S-F (2012) Software synthesis of middleware for heterogeneous embedded systems. In: The 2nd international conference on consumer electronics, communications and networks, IEEE, Yichang
Fang Yq, Wang Gd, Ge Jw, Jun X (2010) A model weaver for dynamic evolution base on MDA aspect-oriented software architecture. In: The 3rd international conference on advanced computer theory and engineering, IEEE, Chengdu
Felhi F, Akaichi J (2013) Pervasive e-healthcare system based on selfadaptability of SOA to the context. In: The 3rd international conference on information technology and e-Services, IEEE, Sousse Tunisia
Fuentes-Fernández R, Pavón J, Garijo F (2012) A model-driven process for the modernization of component-based systems. Sci Comput Program J 77:247–269
Gao B, Ban X, Lv Q, Li X (2010) A component-based method for software architecture refinement. In: The international conference on intelligent control and information processing, IEEE, Dalian
Garlan D (2003) Formal modeling and analysis of software architecture: Components, connectors, and events. In: The 3rd international school on formal methods for the design of computer, communication and software systems: software architectures, Bertinoro Italy. Springer, Berlin Heidelberg, pp 1–24
Georgieva J, Goranova M (2011) Security as a service model in SOA. In: The 11th international conference on applied informatics and communications, Florence Italy
Ghosh S, Kumar S, Rana A (2011) Comparative study of the factors that affect maintainability. Int J Comput Sci Eng 3(12):37–63
Grady RB (1992) Practical software metrics for project management and process improvement. Prentice-Hall, Englewood Cliffs
Gui S, Luo L (2012) Reliability analysis of task model in real-time fault-tolerant systems. In: The 12th international conference on computer and information technology, IEEE, Chengdu
HaiTao W, Wei C (2010) Research on modeling and model converting approaches based on MDA. In: The 2nd world congress on software engineering, IEEE, Wuhan
Hammami A, Simoni N, Salman R (2012) Ubiquity and QoS for cloud security. In: The 41st international conference on parallel processing workshops, IEEE, Pittsburgh PA
He R, Lacoste M, Leneutre J (2010) Virtual security kernel: a component-based OS architecture for self-protection. In: The international conference on computer and information technology, IEEE, Bradford United Kingdom
Hebiri H, Laabidi M, Jemni M (2010) User centered model to provide accessible e-Learning systems. In: The 10th international conference on advanced learning technologies, Sousse Tunisia
Hesse W, Tilley T (2005) Formal concept analysis used for software analysis and modelling. In: The formal concept analysis, foundations and applications. Springer, Berlin Heidelberg
Hoisl B, Sobernig S, Strembeck M (2014) Modeling and enforcing secure object flows in process-driven SOAs: an integrated model-driven approach. Softw Syst Model 13:513–548
Hu J, Khalil I, Han S, Mahmood A (2011) Seamless integration of dependability and security concepts in SOA: a feedback control system based framework and taxonomy. J Netw Comput Appl 34:1150–1159
Immonen A, Niemelä E (2008) Survey of reliability and availability prediction methods from the viewpoint of software architecture. Softw Syst Model 7:49–65
ISO/IEC 25010 (2011) Systems and software engineering: Systems and software Quality Requirements and Evaluation (SQuaRE)—System and software quality models. International Organization for Standardization, Geneva

ISO/IEC 9126-1 (2001) Software product evaluation-quality characteristics and guidelines for their use. International Organization for Standardization, Geneva

ISO/IEC/IEEE 42010 (2011) Systems and software engineering: Architecture description

Jehan S, Pill I, Wotawa F (2013) SOA grey box testing: A constraint-based approach. In: The 6th international conference on software testing, verification and validation workshops, IEEE

Jurado F, Redondo MA, Ortega M (2012) Blackboard architecture to integrate components and agents in heterogeneous distributed eLearning systems: an application for learning to program. J Syst Softw 85:1621–1636

Kang W, Liang Y (2013) A Security ontology with MDA for software development. In: The international conference on cyber-enabled distributed computing and knowledge discovery, IEEE, Beijing

Kang W, Kapitanova K, Son SH (2011) Semantics-aware communication in sensor network applications. In: The international conference on service-oriented computing and applications, IEEE, Irvine CA

Klatt B, Rathfelder C, Kounev S (2011) Integration of event-based communication in the palladio software quality prediction framework. In: The Joint ACM SIGSOFT conference—QoSA and ACM SIGSOFT Symposium—ISARCS on Quality of Software Architectures—QoSA and Architecting Critical Systems—ISARCS, ACM, Boulder, Colorado, USA

Kounev S, Rathfelder C, Klatt B (2013) Modeling of event-based communication in component-based architectures: state-of-the-art and future directions. Electron Notes Theor Comput Sci J 295:3–9

Laabidi M, Jemni M (2010) Personalizing accessibility to e-Learning environments. In: The 10th international conference on advanced learning technologies, IEEE, Sousse, Tunisia

Labejof J, Leger A, Merle P, Seinturier L, Vincent H R-MOM (2012) A component-based framework for interoperable and adaptive asynchronous middleware systems. In: The 16th international enterprise distributed object computing conference workshops, IEEE, Beijing

Lee JY, Kim SD (2010) Software approaches to assuring high scalability in cloud computing. In: The 7th international conference on e-business engineering, IEEE, Shanghai

Lee T-Y, Seo H-R, Lee B-H, Shin D-R (2010) A software component model and middleware architecture for intelligent mobile robot. In: The 2nd international conference on computer and automation engineering, Singapore

Leger M, Ledoux T, Coupaye T (2010) Reliable dynamic reconfigurations in a reflective component model. In: The 13th international conference on component-based software engineering, Prague Czech Republic. Springer-Verlag

Li X, Huang L (2013) Evaluation of software architectures reliability based on hypergraph grammar. In: The 37th annual computer software and applications conference, IEEE, Kyoto

Li C, Zhao C, Yan H, Zhang J (2013) Event-driven fault tolerance for building nonstop active message programs. In: The 10th international conference on high performance computing and communications and the international conference on embedded and ubiquitous computing, IEEE, Zhangjiajie

Linehan E, Clarke S (2012) An aspect-oriented, model-driven approach to functional hardware verification, J Syst Architect

Loganathan MK, Gandhi OP (2015) Maintenance cost minimization of manufacturing systems using PSO under reliability constraint. Int J Syst Assur Eng Manag 7:47–61

Losavio F, Chirinos L, Lévy N, Ramdane-Cherif A (2003) Quality characteristics for software architecture. J Object Technol 2:133–150

Lüders F (2003) Use of component-based software architectures in industrial control systems. Mälardalen University, Västerås

Magableh B, AlBeiruti N (2012) Detecting the onset of dementia using context-oriented architecture. In: The international conference on next generation mobile applications, services and technologies, Paris France

Mahrin M, Carrington D, Strooper P (2008) Investigating factors affecting the usability of software process descriptions, international conference on software process, ICSP, Leipzig, Germany

Majidi E, Alemi M, Rashidi H (2010) Software architecture: a survey and classification. In: The 2nd international conference on communication software and networks, IEEE, Singapore

Massoud A, Dumke R (2012) Efficient reference architecture for integrated legacy applications based-SOA. In: The joint conference of the 22nd international workshop on software measurement and the 7th international conference on software process and product measurement, IEEE, Assisi

Matković J, Fertalj K (2012) Models for the development of web service orchestrations. In: The 35th international convention, IEEE, MIPRO Opatija

McCall JA, Richards PK, Walters GF (1977) Factors in software quality. US Rome Air Development Center Reports, US Department of Commerce

Mezghani E, Ben Halima R (2012) DRF4SOA: A dynamic reconfigurable framework for designing autonomic application based on SOA. In: The 21st international workshop on enabling technologies: infrastructure for collaborative enterprises, IEEE, Toulouse

Microsoft (2009) Microsoft application architecture guide (patterns and practices). Microsoft Press

Minguez J, Zor S, Reimann P (2011) Event-driven business process management in engineer-to-order supply chains. In: The 15th international conference on computer supported cooperative work in design, IEEE, Lausanne

Mircea M (2011) SOA adoption in higher education: a practical guide to service- oriented virtual learning environment procedia: social and behavioral sciences. World Confer Learn Teach Admin 31:218–223

Mnkandla E, Dwolatzky B (2006) Defining agile software quality assurance, international conference on software engineering advances, pp. 36–36

Molesini A, Garcia A, Chavez CVFG, Batista TV (2010) Stability assessment of aspect-oriented software architectures: a quantitative study. J Syst Softw 83:711–722

Nguyen D, Thoai N (2012) EBC: Application-level migration on multi-site cloud. In: The international conference on systems and informatics, IEEE, Yantai

Northrop L (2003) The importance of software architecture. Software Engineering Institute Carnegie Mellon University, Pittsburgh

O'Brien L, Bass L, Merson P (2005) Quality attributes and service-oriented architectures. Carnegie Mellon University, Pittsburgh

O'Brien L, Merson P, Bass L (2007) Quality attributes for service-oriented architectures. In: The international workshop on systems development in SOA environments, IEEE, Minneapolis, MN

Oster ZJ (2013) Reasoning with qualitative preferences to develop optimal component-based systems. In: The international conference on software engineering, IEEE, San Francisco CA USA

Ozkaya I, Diaz-Pace A, Gurfinkel A, Chaki S (2010) Using architecturally significant requirements for guiding system evolution. In: The 14th European conference on software maintenance and reengineering, IEEE, Madrid

Pang C, Yan J, Vyatkin V (2014) Time-complemented event-driven architecture for distributed automation systems transactions on systems, man, and cybernetics: Systems 45: 1165–1177

Pham T-T, Defago X (2013) Reliability prediction for component-based software systems with architectural-level fault tolerance mechanisms. In: The 8th international conference on availability, reliability and security, IEEE, Regensburg

Poelmans J, Dedene G, Snoeck M, Viaene S (2010) Using formal concept analysis for verification of process-data matrices in conceptual domain models, Software engineering

Pordel M, Khalilzad NM, Yekeh F, Asplund L (2011) A component based architecture to improve testability,targeted FPGA-based vision systems. In: The 3rd international conference on communication software and networks, IEEE, Xi'an

Potena P (2013) Optimization of adaptation plans for a service-oriented architecture with cost, reliability, availability and performance tradeoff. J Syst Softw 86:624–648

Qazi A, Shamim A, Bano H (2013) Model driven architecture with encapsulated quality check and enhancement feature. In: The 3rd international conference on innovative computing technology, IEEE, London

Quintero R, Zepeda L, Vega L (2010) Model driven software development of applications based on web services, Int J Web Grid Serv

Raafat T, Cecelja F (2011) An ontological approach to inter-operation of mobile services. In: The 4th IFIP international conference on new technologies, mobility and security, IEEE, Paris

Rafea V, Mahdianb F (2011) Style-based modeling and verification of fault tolerance service oriented architectures Procedia Computer Science. World Conf Inf Technol 3:972–976

Ren Y (2011) A dynamic comprehensive web service for content management. In: The 3rd international conference on communication software and networks, IEEE, Xi'an

Saaty TL (2008) Decision making with the analytic hierarchy process. Int J Serv Sci 1:83–98

Salva S, Rabhi I (2010) A preliminary study on BPEL process testability. In: The 3rd international conference on software testing, verification, and validation workshops, IEEE, Paris

Shanmugasundaram G, Prasanna Venkatesan V, Punitha Devi C (2012) Research opportunities in service reusability of service oriented architecture. In: The international conference on emerging trends in science, engineering and technology, IEEE, Tiruchirappalli, Tamilnadu India

Sirer EG, de Bruijn W, Reynolds P, Shieh A, Walsh K, Williams D, Schneider FB (2011)Logical attestation: an authorization architecture for trustworthy computing. In: The 23rd ACM symposium on operating systems principles, ACM, Cascais Portugal

Smith CU, Llado CM (2011) Model interoperability for performance engineering: survey of milestones and evolution. In: performance evaluation of computer and communication systems. milestones and future challenges, Vienna Austria. Springer, Berlin Heidelberg

Snajberk J, Holy L, Brada P (2013) Visualization of Component-based applications structure using AIVA. In: The 17th European conference on software maintenance and reengineering, IEEE, Genova

Sneed HM, Verhoef C, Sneed SH (2013) Reusing existing object-oriented code as web services in a SOA. In: The 7th international symposium on the maintenance and evolution of service-oriented and cloud-based systems, IEEE, Eindhoven

Staroswiecki M (2010) On reconfiguration-based fault tolerance. In: The 18th Mediterranean conference on control and automation Marrakech, IEEE

Sun C-A, Wang G, Mu B, Liu H, Wang Z, Chen TY (2011) Metamorphic testing for web services: framework and a case study. In: The international conference on web services, IEEE, Washington DC

Tielin Q, Yafen L, Pu W (2010) A model transformation platform design based on model driven architecture. In: The International Conference on Intelligent Computation Technology and Automation, IEEE, Changsha

Tizzei LP, Dias M, Rubira CMF, Garcia A, Lee J (2011) Components meet aspects: assessing design stability of a software product line. Inf Softw Technol J 53:121–136

Tragatschnig S, Zdun U (2013) Enterprise integration using event actor based event transformations. In: The 28th annual ACM symposium on applied computing, ACM Coimbra Portugal

Tran N-L, Skhiri S, Zimanyi E (2011) EQS: an elastic and scalable message queue for the cloud. In: The 3rd international conference on cloud computing technology and science, IEEE, Athens

Trilles S, Juan P, Díaz L, Aragó P, Huerta J (2013) Integration of environmental models in spatial data infrastructures: a use case in wildfire risk prediction. IEEE J Select Top Appl Earth Observ Remote Sens 6:128–138

Trinugroho YBD, Gerdes M, Amjad MMM, Reichert F, Fensli R (2013) A REST-based publish/subscribe platform to support things-to-services communications. In: The 19th Asia-Pacific conference on communications, IEEE, Denpasar

Tyagia K, Sharmab A (2014) An adaptive neuro fuzzy model for estimating the reliability of component-based software systems. Appl Comput Inf 10:38–51

Wallis M, Henskens F, Hannaford M (2010) Expanding the Cloud: A component-based architecture to application deployment on the Internet. In: The 10th IEEE/ACM international conference on cluster, cloud and grid computing, IEEE, Melbourne, Australia

Wang Q, Yang Z (2012) A method of selecting appropriate software architecture styles: quality Attributes and analytic hierarchy process. University of Gothenburg, Chalmers University of Technology, Göteborg

Wang B, Wei Z, Jinfang S (2010) Holographic view language hvl4dcam for aspectual middleware platform. In: The 2nd international conference on computer engineering and technology, Chengdu, China 16–18 April 2010. IEEE

Wang S, Wainer G, Goldstein R, Khan A (2013) Solutions for scalability in building information modeling and simulation-based design. In: the symposium on simulation for architecture and urban design, San Diego California

Waris M, Khan SA, Fakhar MZ (2013) Factors Effecting service oriented architecture implementation. In: The science and information conference, IEEE, London

Weini Z, Yongquan L, Pengdong G, Chu Q, Quan Q (2012) A new software architecture for ultra-large-scale rendering cloud. In: The 11th international symposium on distributed computing and applications to business, engineering and science, IEEE, Guilin

Welke R, Hirschheim R, Schwarz A (2011) Service-oriented architecture maturity computer 44:61–67

Wen X, Yu H, Zheng H (2013) Aspect-oriented design method for embedded systems based on timed statecharts China Communications

White L, Wilde N, Reichherzer T, Baskin A, Hartmann B, Manea M (2012) Understanding interoperable systems: challenges for the maintenance of SOA applications. In: The 45th Hawaii international conference on system science, IEEE, Maui HI

Wolf F, Balasubramanian J, Tambe S, Gokhale A, Schmidt DC (2011) Supporting component-based failover units in middleware for distributed real-time and embedded systems. J Syst Arch: EUROMICRO J 57:597–613

Wu Y, Huang G, Song H, Zhang Y (2012) Model driven configuration of fault tolerance solutions for component-based software system. In: The 15th international conference on model driven engineering languages and systems, Innsbruck Austria. Springer, Berlin Heidelberg

Yamany HFEL, Capretz MAM, Allison DS (2010) Intelligent security and access control framework for service-oriented architecture. Inf Softw Technol J 52:220–236

Yang D, Liu M, Wang S (2012) Object-oriented methodology meets MDA software paradigm. In: The 3rd international conference on software engineering and service science, IEEE, Beijing

Yang Z-X, Ning H-Y, Sun J-Q, Yang J-B (2013) Service portfolio optimization algorithm based on value model and graph theory in SOA. In: The 4th international conference on software engineering and service science, IEEE, Beijing

Yuan ED, Watton M (2012) An event-driven service oriented architecture for space command and control decision making. In: The Aerospace Conference, IEEE, Big Sky MT

Yue H, Tao X (2012) Web services security problem in service-oriented architecture. In: International conference on applied physics and industrial engineering

Yue D, Li CC, Liu WJ, Feng WX (2010) Based on SOA architecture and component software reuse architecture research. In: The 2nd international conference on information management and engineering, IEEE, Chengdu

Zappia I, Paganelli F, Parlanti D (2012) A lightweight and extensible complex event processing system for sense and respond applications, Expert Systems with Applications

Zhang X, Pham H, (2000) An analysis of factors affecting software reliability. In: Journal of Systems and Software, 43–56

Zhang J, Ban X, Lv Q, Chen J, Wu D (2010) A component-based method for software architecture refinement. In: The 29th Chinese control conference, IEEE, Beijing

Zheng Z, Lyu MR (2010) Collaborative Reliability prediction of service-oriented systems. In: The 32nd international conference on software engineering, IEEE, Cape Town

Zhou N, Zhang L-J, Chee Y-M, Chen L (2010) Legacy asset analysis and integration in model-driven SOA solution. In: International Conference on Services Computing, IEEE, Miami FL