Principles and Metrics to Improve Quality in SOA Applications

Joyce M. S. França[1] and Michel S. Soares[2]
[1]*Federal University of Uberlândia, Faculty of Computing, Uberlândia, Brazil*
[2]*Federal University of Sergipe, Department of Computing, São Cristóvão, Sergipe, Brazil*

Abstract: Service-Oriented Architectures (SOA) have emerged as an architectural approach for building distributed applications based on the concept of assembling services. Currently, the growing complexity of software requires greater attention about the quality of the produced applications. Many advances in SOA quality have been proposed in the literature, such as definitions of quality models, attributes and some metrics. However, many research gaps were found and thus much more needs to be done in this research area. There is a lack of metrics and tools to evaluate quality in SOA. Another aspect that needs to be addressed is the application and evaluation of research in accordance with the ISO 25000 standard. Therefore, this research project is mainly aimed at addressing relevant issues in the quality of service-oriented applications that have not been adequately explored. This paper presents the state of the art in the area of quality in SOA, the objectives that have been defined, the methodology to be followed to achieve the objectives, and the expected outcomes of the research project.

1 INTRODUCTION

Service-Oriented Computing (SOC) is the computing paradigm that utilizes services as fundamental elements for developing applications/solutions (Papazoglou, 2003). A service is a capability of the business organization that is implemented and made available on the Internet (or Intranet) so that other applications can access it. Services are autonomous, platform-independent computational entities that can be used to create large, complex applications.

An important concept related to SOC is the Service-Oriented Architecture (SOA). Many definitions on SOA were proposed in the literature. According to the Open Group SOA Working Group (SOA, 2014), SOA is an *architectural style* that supports service-orientation, and service-orientation is a way of thinking in terms of services and service-based development and the outcomes of services. SOA has also been defined as a *distributed architecture* with services distributed in an environment working together to perform some tasks (Steen et al., 2005). In summary, SOA can be defined as a paradigm or an architecture to implement loosely-coupled, platform-independent distributed software systems using communicating services. Currently, SOA is well-recognized as an established architectural paradigm for distributed systems (Goeb and Lochmann, 2011).

Services performs functions that can be simple requests for activities or complex business processes. Therefore, services can be of simple nature or composite. Composite services involve the concept of orchestration. The concept of orchestration is related to automating business processes. The orchestration allows services to be composed of other services, in such a way that the logic of the process is centralized by orchestration that enables extensibility (composition of new services).

Another important concept of the SOA paradigm is choreography (Erl, 2005), which enables collaboration between organizations of different applications. Choreography is essentially a collaborative process designed to enable organizations to interact in an environment that is not owned by any partner. The WS-CDL [1] specification (WSCDL, 2014) enables the exchange of information among multiple organizations (or even different applications within the organization).

Service-oriented computing is not the first proposed paradigm for systems integration. Several distributed solutions have been proposed, with varying degree of success and limitations. Integration of software systems became a key reference in the late 1990s and many organizations were not well-prepared for it

[1]Web Services Choreography Description Language.

(Erl, 2007). Thus, many systems were developed with little elaboration and integration point to point was created according to the needs that were emerging. This approach produced a tangle of complex connections, difficult maintenance and problems associated with lack of stability, extensibility and inadequate interoperability frameworks. The EAI [2] platforms introduced the middleware that allowed more robust and extensible architectures. Despite improvements in integration over time, middlewares were often considered to be complex and expensive (Erl, 2005).

Service-oriented computing proposes the use of open standards such as SOAP [3] (SOAP, 2014) and XML (to define the WSDL [4] (WSDL, 2014)). The SOAP protocol is used for data transmission. The WSDL describes the information about the service such as the service functionality description and data types transmitted in a structured way using a grammar described in XML. The use of these open standards facilitates the development of service-oriented applications because the exchanged information has a standardized format.

Service in SOA is platform independent (Papazoglou, 2003). This feature is of great advantage because SOA enables integration in heterogeneous environments. Thus, no matter what platform the system has been developed in, it is always possible to propose integration using SOA.

The possibility of interacting systems in heterogeneous environments makes SOA a very attractive technology because it enables reuse of legacy systems. Legacy systems hold the key and complex business processes of companies. Legacy systems were developed in the past decades using programming languages such as COBOL, RPG, and C/C++ (Bennett, 1995). These systems are known to be inflexible and difficult to maintain. It is estimated that more than 80% of business data in the world are executed on COBOL and 50% -70% of total IT costs are expent for the maintenance of these systems (Koutsoukos et al., 2006). SOA has emerged with the promise of allowing legacy systems to expose their core functionality as services (Khadka et al., 2011).

With software systems becoming more complex over time, quality assurance become increasingly important. Software quality is a challenge for software engineering, and this challenge extends to SOA applications. The emergence of service-oriented architecture introduced new challenges. One of these challenges is the need to create design principles and specific design patterns for SOA in order to design high quality applications. These quality properties need to be evaluated in practice, but few researches on specific quality attributes, tools and metrics for SOA applications were proposed in the literature. This is the core of this thesis proposal.

[2]Enterprise Application Integration middleware.

[3]Simple Object Acess Protocol.

[4]Web Services Description Language.

2 STAGE OF THE RESEARCH

The first stage of this project was to determine the real applicability of SOA design principles in practice, and further evaluating the influence of these design principles on the quality of SOA applications. This study aims to analyze if these principles are used in practice, how they are implemented, in which domains they are most used, what advantages are obtained in their use, and how the application of these principles influences the quality of the application. A systematic literature review has been performed to answer properly these research questions.

The designed strategy to perform a systematic literature review was to search empirical studies that mentioned the use of design principles. Articles presenting researches in which SOA applications were developed as case studies were recovered in the systematic review for an analysis of how the principles of SOA design are used and if any mention was made to the influence of the use of these principles in practice.

Among the results of this review, an interesting one is that few studies mentioned the use of design principles when developing SOA applications in case studies. The reason is not yet clear. Therefore, the second part of the project is to interview professionals and academic researchers with focus on trying to better understand this and other results found within the systematic review.

3 OUTLINE OF OBJECTIVES

The main objective of this study is to research on underexplored topics in SOA quality. Several research gaps were found in the area of quality in SOA. For instance, lack of tools for measuring quality of SOA applications (Goeb and Lochmann, 2011) and lack of service design principles (Papazoglou et al., 2008). In addition, few studies were published considering the ISO 25000 standard (ISO/IEC, 2005) that deals with quality and has been recently proposed to replace the ISO 9126 (Goeb and Lochmann, 2011) (ISO/IEC, 2001).

The specific objectives of this research are:

1. **To Systematically Evaluate the Applicability of SOA Design Principles.**

This study aims to contribute with design principles that improve the development and quality of SOA applications. The existing SOA principles have to be classified, for example, in relation to their real applicability in practice, measured advantages and scope of use. Thus, during this research a systematic review has been developed to investigate these issues.

2. **To Propose New Metrics.**
   Another objective of this study consists of evaluating the existent metrics to measure SOA quality and to analyze them. It is of fundamental importance to know if there exist metrics to evaluate all SOA quality attributes, including the ones described in ISO 25000. This work intends to propose new relevant metrics to assess the quality of service-oriented applications.

3. **To Propose Improvement in Tools.**
   From the literature review, it is possible to notice the lack of tools to automatically collect metrics to assess quality attributes of service-oriented applications. This study aims also to explore this research gap in SOA. The strategy to achieve the goal is to detect existing tools that measure quality in SOA applications. Thus, one can analyze the quality of these tools based on predefined criteria and verify which metrics are automatically used to measure SOA quality.

4. **To Contribute with SOA Quality Attributes Based on ISO 25000.**
   Several founded studies are based in the ISO 9126 standard that was replaced by ISO 25000. This study aims to contribute with new metrics to measure quality attributes in SOA based on ISO 25000.

5. **To Produce Case Studies to Validate the Research.**
   Software Engineering studies, in general, have received critics regarding the few amount of research with quantitative validation (Shaw, 2002) (Runeson and Höst, 2009). One objective of this research is to produce case studies with quantitative validation. The case studies will be developed to suggest design principles for SOA that improve quality application. Therefore, tools for evaluating SOA quality will be used to produce the quantitative validation. The qualitative analysis of case studies will be performed by applying questionnaires and interviews with users, students and professionals.

4 RESEARCH PROBLEM

The research problem addressed by this study is related to quality of SOA applications. Several research gaps were found in the literature in quality about SOA. Thus, there are many SOA quality topics that need to be better explored.

The research problem addressed by this study was divided into three sub-problems presented below.

1. **SOA Design Principles.**
   Initially, the purpose of this study is to focus on a research challenge presented by Papazoglou in (Papazoglou et al., 2008): Design principles for engineering service applications. This research topic covers the search for fundamental principles that are the basis for designing services. It is important to find out if this gap, proposed in 2008, has been adequately addressed by researchers in the past five years. Some research questions arise from this topic: the literature provides all the fundamental principles for design services? How these principles are used? What advantages are obtained in their use? In what areas are they being used? SOA applications are used only for legacy systems or new systems too? Is there experimental evidence on the influence of these principles in the quality of the application?

   The first hypothesis to this research is: H1 - *Current specific SOA design principles are necessary and sufficient to improve quality in SOA applications*.

2. **Lack of Metrics and Tools to Measure SOA Quality Based on ISO 25000.**
   Several quality models specific for SOA have been presented in the literature. However, these quality models have their limitations related with lack of metrics and tools to measure quality in SOA applications. The proposed quality models describe several quality attributes but do not provide tools for evaluating the degree of quality attributes in SOA applications. Most quality models were proposed in the literature in order to provide quality attributes in accordance with the old ISO 9126. However, the ISO 25000 is more complete and have a wider range of quality requirements for software. Thus, there is a lack of research studies addressing quality attributes based on ISO 25000. Another important analysis is to verify if the existing quality attributes based on ISO 9126 complies with the ISO 25000, and if ISO 25000 will bring any benefit at all when compared with ISO 9126.

   The second hypothesis is: H2 - *The ISO 25000 brings important benefits to SOA applications when compared to ISO 9126.*

5 STATE OF THE ART

Literature review involves analyzing the studies already produced in the area. Literature review for this project ia divided into two main areas: software quality for SOA and SOA design principles. The following subsections describe some studies in these two areas.

5.1 Software Quality for SOA

The evaluation of software quality is an extremely important activity in the software development process. According to Sanders and Curran (Sanders and Curran, 1994), quality is crucial for survival and success. The software market is increasingly global, and companies will have difficulties in succeeding on the market unless they produce quality products and services.

According to Sommerville (Sommerville, 2010), software quality is evaluated from internal and external quality attributes. The factors that affect quality, and can be directly measured, are called internal quality attributes (for example lines of code). Factors that can only be indirectly measured are called external quality attributes such as maintainability.

SOA quality can be evaluated by using several external attributes, as for instance (O'Brien et al., 2007): Interoperability, Performance, Security, Reliability, Availability, Modifiability, Testability, Usability, and Scalability. However, there is a lack of metrics to measure attributes such as modifiability and reusability.

Quality models are elaborated to describe, to evaluate and to predict the quality of a SOA application. These models can be provided, for example, by organizations of international standards or guidelines (Voelz and Goeb, 2010). Specific models for SOA quality have been proposed in the literature: S-Cube (Gehlert and Metzger, 2008), WSQM2.0 [5] (WSQM, 2005), Quamoco-extended (Goeb and Lochmann, 2011), and quality model for evaluating reusability (Choi and Kim, 2008). SOA Quality models are most often structured into a set of quality attributes that should be found in SOA applications. Some models, such as the one proposed by Choi and Kim (Choi and Kim, 2008) also defines metrics to assess the defined quality attributes.

S-Cube is a quality reference model for service-based applications. This quality model consists of 89 quality attributes for service oriented computing based on software engineering attributes (ISO 9126 and UML profiles). The weaknesses found in this model (Goeb and Lochmann, 2011) are the high complexity and lack of clarity in many settings, the fact that the model does not fully follow the rules of ISO 9126 and does not provide a tool to assess the proposed quality attributes (actually the authors indicate that the development of a tool will be addressed in future work).

[5]OASIS Quality Model for Web Services (WSQM2.0).

WSQM is a quality model for Web Services defined by OASIS (Organization for the Advancement of Structured Information Standards) [6]. WSQM has a similar approach to S-Cube in which it establishes a set of important attributes of quality in the field of web services. However, according to Goeb (Goeb and Lochmann, 2011), this model is not mature enough because many settings are not accurate.

The research project Quamoco [7] (Kläs et al., 2011) (Lochmann and Heinemann, 2011) formed a quality model for software that defines a generic meta-model that enables the creation of tools for many types of systems. However, the Quamoco model is restricted to unique systems, and therefore does not provide support for SOA applications. Thus, an extension to Quamoco was created to include applications in SOA meta-model (Goeb and Lochmann, 2011). The extended Quamoco listed several upcoming activities to complete the project, one of them is to create a tool to produce quality measures.

Other articles have been published presenting metrics for evaluating quality in SOA. Most case studies are focused on presenting metrics to evaluate the quality of service (QoS). One of these studies was performed by Choi and Kim (Choi et al., 2007) where QoS metrics have been defined to evaluate whether services provide the responses for consumer application (a Consumer Application is a partner organization application that provides information for service). A set of metrics has been created to evaluate each of the following six quality attributes: performance, availability, reliability, dynamic discoverability, dynamic adaptability and dynamic composability. A case study was proposed to explore the applicability of the metrics. Only one application was used as case study, a Hotel Reservation Service (HRS). Metrics related to the availability of services were used and it was concluded that the services of the HRS have high availability. Considering the services performance metrics, one service received long response time indicating that improvements should be implemented to optimize the service. The other proposed metrics have not been evaluated by limitations in the applicability. The study described in the paper was based on the ISO 9126 quality model.

[6]https://www.oasis-open.org/

[7]http://www.quamoco.de

In 2008, Choi and Kim published another study (Choi and Kim, 2008) that proposes metrics to evaluate service reusability. In the study, reusability was evaluated using quality attributes such as modularity and adaptability. One case study in the field of flights management with booking and purchasing airline tickets services was used to apply the new proposed metrics. The study did not use tools to automatically collect the results of the metrics.

It is possible to notice that much more needs to be done regarding SOA quality research. There is a lack of metrics and tools to evaluate quality in SOA. Another aspect that needs to be addressed is the production of studies in accordance with the ISO 25000 standard because it is more complete in relation to quality attributes than ISO 9126.

5.2 SOA Design Principles

According to the SWEBOK (Software Engineering Body of Knowledge) (IEEE Computer Society, 2004), software design principles are key notions considered fundamental to many different software design approaches and concepts. Several design principles for software have been proposed in the literature with the aim of improving the logic of the solution and/or deal with the problem complexity. The principles of software engineering are useful for the emergence of best practices (Bourque et al., 2002) that collaborate to create consistent designs.

Examples of design principles for software engineering found in SWEBOK (IEEE Computer Society, 2004) are abstraction, coupling and cohesion, decomposition and modularization, encapsulation/information hiding, separation of interface and implementation, sufficiency, completeness and primitiveness.

A brief description of each of the above principles are presented as follows based on various publications (Bass et al., 1998) (Bosch, 2000) (Buschmann et al., 1996) (Jalote, 1997) (Liskov and Guttag, 2000) (Pfleeger, 2001) (Pressman, 2009).

Abstraction: Abstraction is the process of representing data and programs omitting the implementation details.

Coupling and Cohesion: The coupling is defined as the degree of relationship between the modules. Cohesiveness is defined as the way that elements of a module are related.

Decomposition and Modularization: Process to decompose and modularize large software into several smaller parts, often with the goal of assigning different functions or responsibilities in different components.

Encapsulation/Information Hiding: Encapsulation/information hiding means grouping and packaging the elements and internal details of an abstraction and making those details inaccessible.

Separation of Interface and Implementation: Separation of interface and implementation involves defining a component by specifying a public interface, known by the client and separate the details of how the component is implemented.

Sufficiency, Completeness and Primitiveness: Achieving sufficiency, completeness and primitiveness means ensuring that a software component captures all the important features of an abstraction and nothing more.

A literature review was proposed with the aim of analyzing if the principles for software engineering in general are applicable to SOA. Few studies in this area were found. The study presented by Huhns and Singh (Huhns and Singh, 2005) demonstrates how the principle of abstraction is applied specifically in SOA. Another important publications is the book from Erl, T. (Erl, 2007) in which eight specific design principles for SOA are highlighted. These studies are detailed as follows.

Huhns and Singh (Huhns and Singh, 2005) show how to apply the principle of abstraction in SOA. Services provide high level of abstraction for large-scale organizational applications. The article describes a hospital system that provides a range of services to exemplify existing levels of abstraction. A large hospital has, among other activities, the challenge of making the payment, providing suitable scales and interacting with account systems. A hospital system provides service within the organization itself, which exemplifies a level of intra-enterprise abstraction, as well as with other organizations. SOA allows that legacy systems components can be encapsulated as services, preserving the component behavior and facilitating composition. That way, the services enable interoperability of systems belonging to the hospital.

Another level of abstraction highlighted in this article (Huhns and Singh, 2005) is the inter-enterprise one. For better understanding the following situation is illustrated. A traditional approach would be to send paper bills for agencies and insurance companies that receive it and type again data into their own information system. Of course, these approaches are disappearing in favour of online sales.

The service-oriented computing provides the ability for interaction between companies and allows the choreography of the parties and automatically apply local policies to suit the inter-enterprise processes.

The other level of abstraction presented was the infrastructure. The activities related to building complex applications on distributed platforms, such as network architectures, are difficult and have led to an interest in modular interfaces based services. Currently, companies can focus on their core business and outsource their computing infrastructure to specialized companies.

According to Erl (Erl, 2007), a principle of design is defined as a widespread and accepted concept by the market. A principle can be compared to good practice by the fact that both propose a means of achieving something based on experience or market acceptance. Erl presents eight specific principles for SOA design: service contract, service loose coupling, service abstraction, service reusability, service autonomy, service statelessnes, service discoverable, service discoverable and service composability. These design principles are considered fundamental principles for SOA applications and are widely referenced in the literature. A brief description of each principle is highlighted as follows.

Service Contract: services share standardized contracts. A service contract consists of a technical interface or one or more service description documents. Each service contract must comply with the same standards applied to other design services contracts within a service inventory.

Service Loose Coupling: services are loosely coupled. The existence of a service contract, ideally, is decoupled from the implementation details and technology.

Service Abstraction: services are designed to limit information in the service contract to what is really necessary for the service to be functionally useful to consumers. Information beyond that is published in a service contract and is considered private and should not be made available for creating potential consumers of service. Services are consistently abstract, and specific information about technology, logic and function in the world is outside the service boundary.

Service Reusability: services are reusable. The service encapsulates the logic that is useful for more than one service request. The encapsulated by the service logic is generic enough to allow many usage scenarios and can be used for different types of service consumers.

Service Autonomy: services are autonomous. That way, the services have a contract that expresses a well-defined functional threshold which should not involve other services.

Service Statelessnes: services minimize dependence on the state. The services are specifically designed to minimize the period during which they exist in a condition of dependence on state information.

Service Discoverable: services are designed to be effectively discovered and interpreted so that the discovery, its purpose and capabilities are clearly understood. Service contracts should contain adequate metadata that will be referenced correctly when viewing queries are issued.

Service Composability: services are designed to act as effective composition participants, regardless of the size and complexity of the composition. Services can be composed to automate business processes of a company.

Besides presenting eight principles for SOA, Erl (Erl, 2007) also provides an important insight into other principles of SOA. The author asserts that some principles and concepts of SOA were inspired by principles of the object-oriented paradigm. A comparison of design principles for service orientation was held for several fundamental principles of object orientation. A section of the book discusses how service orientation is related to each principle of object-oriented design: Encapsulation, Inheritance, Generalization and Specialization, Abstraction, Polymorphism, Open-Closed Principle (OCP), Don't Repeat Yourself (DRY), Single Responsibility Principle (SRP), Delegation, Association, Composition and Aggregation.

The work proposed by Erl defines the specific design principles for SOA. However, it is important to define how these principles are applied in practice, whether they are sufficient and to what extent they aggregate quality on applications.

6 METHODOLOGY

According to Shaw (Shaw, 2002), research in Software Engineering in general seek for better ways to develop and evaluate software. This study aims to contribute with better ways of developing and evaluating SOA applications. The following research methods were planned to achieve the proposed objectives (section 3).

Initially, the purpose of this study is to focus on a research challenge presented by Papazoglou in (Papazoglou et al., 2008): Design principles for engineering service applications. This research topic covers the search for fundamental principles that are the basis for design service. It is important to find out if this gap, proposed in 2008, has been adequately addressed by researchers in the past five years. In other words, the literature provides all the fundamental principles for design service? In addition, how these principles are used? What advantages are obtained in their use? In what areas are they being used? SOA applications are used only for legacy systems or new systems too?

The literature provides few specific design principles for SOA. Moreover, the existing principles are focused on defining the essence of a service and not to contribute to improving the quality of applications in SOA. Based on this, the first planned step in this study is developing a systematic literature review (Kitchenham, 2004). The systematic review has the purpose of analyzing the following research questions:

- What are the specific principles for SOA mostly used in service-oriented applications?
- How these principles are applied in practice?
- In which areas these principles are used?
- The application of each principle has measurable advantages in the quality of the application?
- The cited principles are sufficient to develop quality SOA applications or there is need for proposing new principles?

The proposed systematic literature review focuses on searching empirical studies that use design principles for developing their SOA applications. The found studies are analyzed with the intention of answering the research questions proposed previously.

In order to evaluate the proposed hypothesis, the initial idea is to propose case studies (Yin, 2003) (Runeson and Höst, 2009) to evaluate the quality of SOA from the application of specific principles for SOA. From experience and lessons learned in the development of case studies, principles that enable quality improvement in SOA will be proposed.

Another research gap found in the literature is a lack of metrics and tools to measure SOA quality. In the research study presented by Goeb (Goeb and Lochmann, 2011), he cited many specific quality metrics for SOA, however these models have no specific tool support. Thus, it can be seen that there is much to be done in the context of quality of services.

Case studies proposals are presented in the next section. These proposals were based on ideas for implementing service-oriented architecture in areas that do not yet use this technology, or at least they are still very incipient, and would represent an important innovation to facilitate the implementation of several activities.

6.1 Proposed Case Studies

A case study is defined as "an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident" (Yin, 2003). Rather than using samples, case study methods involve an in-depth examination of a single instance or event. When selected with care, even a single case study may be successful in terms of theory formulation and testing (Yin, 2003). This might be because it is a critical case for testing a well-formulated theory, and if the theory holds for this case, it is likely to be true for many others.

There are several applications examples in SOA that can be used as case study. The following two examples are services oriented applications that may be developed during the research.

1. **University Students.** Brief Description: Several services must be performed by a student when starting an undergraduate university course, such as: registering at the library, request the card to use University Restaurants, request various kinds of student financial support, such as student housing and transportation, access to research labs., and so on. The process to be done by the student would be simpler if there was an application for accessing all these services.
2. **Hospital System.** Brief description: A hospital system has many complex processes that can be automated. An example of a process that can be automated is a solicitation to buy hospital supplies. The service responsible for conducting the purchase can access the system by the selling company, request items, generate the full amount of purchase, request permission from the purchase to the hospital leader and then authorize to forward the invoice to the hospital financial area.

6.2 Validation

This study will be validated through qualitative and quantitative analysis. Proposing a new solution to a problem is not enough, and this solution needs to be validated for its applicability in practice. Two approaches can be used to validate a study: quantitative and qualitative. Quantitative consists in evaluating the product solution through numbers that can be analyzed by statistical methods, while qualitative involves descriptions, pictures and diagrams and can be analyzed through categorization (Runeson and Höst, 2009).

This study proposes to contribute with a set of principles, metrics and tools to collaborate with improvements in quality of SOA applications. This set of principles, metrics and tools (called product of this research) must be validated. The way to evaluate the applicability of the product will be through a qualitative evaluation. The qualitative evaluation in this case is to introduce the developed work during the research for professionals through questionnaires to assess whether the proposed product presents relevance on its use. To perform this qualitative analysis, human subjects, including master and doctoral students, university professors, in addition to professionals working in industrial SOA applications are to be used with the expectation that they can provide valuable feedback. For the development of controlled experiment, the number of people is estimated to be between 15 and 20.

Validation of case studies will be done through quantitative and qualitative evaluations. According to Seaman (Seaman, 1999), a combination of qualitative and quantitative data often provides better understanding of the studied phenomenon.

The Quantitative analysis of the case studies involves the use of metrics and tools for SOA. Thus, SOA applications developed in this research will be submitted to be evaluated with the implemented tools regarding the quality of the produced application.

The qualitative analysis will be divided into two steps. The first is based on experience gained during the development of case studies, such as lessons learned and conclusions. The second step is to produce a controlled experiment (Seaman, 1999) in which the applications are introduced to human subjects and they return feedback. The assessment will be based on questionnaires and interviews.

7 EXPECTED OUTCOME

Two paradigms characterize much of the research in Information Systems: behavioral science and design science (Hevner et al., 2004). The behavioral science paradigm seeks to develop and verify theories that explain or predict human or organizational behavior. It has its roots in natural science research methods. The main objectives are to develop and justify theories that explain or predict organizational and human phenomena surrounding the analysis, design, implementation, management, and use of information systems.

The design science paradigm, which is followed in this research, has its roots in engineering. As a matter of fact, it is fundamentally a problem solving paradigm with emphasis on products and solutions (Rossi and Sein, 2003).

There is lack of consensus as to the precise desired outputs of design research. The resulting products of this research, according to (Hevner et al., 2004), include 1) constructs or concepts, i.e., abstractions and representations that define the terms used when describing an artifact; 2) models, i.e., abstractions and representations, that are used to describe and represent the relationship among concepts; and 3) instantiations, i.e., implemented or prototyped systems, that are used to realize the artifact.

This study expects to contribute with research on SOA quality. Some research gaps were found, such as few specific design principles for SOA, few studies using ISO 25000 standards and lack of metrics and tools to evaluate quality in SOA applications. According to (Shaw, 2002), Software Engineering projects may produce different outcomes as final products. Therefore, the expected outcomes are:

1. **SOA Design Principles Report.**
   The present study aims to contribute to a catalog that defines the principles of design specific to SOA. In addition, to inform the actual applicability of these design principles. That is, how principles are applied in practice, the application domains in which SOA is more widely used, and especially what are the effects of using these principles regarding the quality of the application. Most of the answers contained in this report come from the systematic literature review conducted in this study. Another part can be produced from the experiences and lessons learned during the development of the case studies to be developed during this research.

2. **New Quality Attributes and Metrics in Accordance With ISO 25000.**
   Another expected result of this research is to propose quality attributes according to the ISO 25000. Most of studies address quality attributes in accordance to the old ISO 9126. However, the ISO 25000 provides a larger range of quality requirements for software product. In addition to proposing attributes, this study wants to produce as a result an analysis to determine whether the existing quality attributes based on the old ISO 9126 are in accordance with the ISO 25000.

3. **Set of Metrics and Tools to Evaluate SOA Quality.**
   Another important research result of this study is a package of metrics and tools that can be used to measure quality in SOA applications. The implemented tool should automatically collect metrics results to analyze quality attributes contained in SOA application such as modifiability and reusability. This tool can collaborate with SOA

developers to investigate whether the chosen solution design produced a quality application according to the quality attributes.

4. **Specific Solution - Case Studies.**
According to Shaw (Shaw, 2002), a research result in software engineering can be a specific solution which means to present a solution to an application problem that shows use of software engineering principles. Therefore, an important research result is to develop an application using design principles and thereby producing a good solution for the problem. This study will develop two case studies (proposed in subsection 6.1) to validate the metrics and tools produced. These case studies will also be useful to analyze the use of design principles and to discuss what it can be inferred in the SOA applications quality.

REFERENCES

Bass, L., Clements, P., and Kazman, R. (1998). *Software Architecture in Practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Bennett, K. (1995). Legacy Systems: Coping with Success. *IEEE Software*, 12(1):19–23.

Bosch, J. (2000). *Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA.

Bourque, P., Dupuis, R., Abran, A., Moore, J. W., Tripp, L., and Wolff, S. (2002). Fundamental Principles of Software Engineering - A Journey. *Journal of Systems and Software*, 62(1):59–70.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. (1996). *Pattern-Oriented Software Architecture: a System of Patterns*. John Wiley & Sons, Inc., New York, NY, USA.

Choi, S. W., Her, J. S., and Kim, S. D. (2007). QoS Metrics for Evaluating Services from the Perspective of Service Providers. In *Proceedings of the IEEE International Conference on e-Business Engineering*, pages 622–625. IEEE Computer Society.

Choi, S. W. and Kim, S. D. (2008). A Quality Model for Evaluating Reusability of Services in SOA. In *Proceedings of the 10th IEEE International Conference on E-Commerce Technology (CEC 2008) / 5th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (EEE 2008)*, pages 293–298.

Erl, T. (2005). *Service-Oriented Architecture Concepts, Technology, and Design*. Prentice Hall Professional Technical Reference.

Erl, T. (2007). *SOA Principles of Service Design (The Prentice Hall Service-Oriented Computing Series from Thomas Erl)*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Gehlert, A. and Metzger, A. (2008). Quality Reference Model for SBA. Technical report.

Goeb, A. and Lochmann, K. (2011). A Software Quality Model for SOA. In *Proceedings of the 8th international workshop on Software quality*, WoSQ '11, pages 18–25, New York, NY, USA. ACM.

Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly,*, 28(1):75–105.

Huhns, M. N. and Singh, M. P. (2005). Service-Oriented Computing: Key Concepts and Principles. *IEEE Internet Computing*, 9(1):75–81.

IEEE Computer Society (2004). *Software Engineering Body of Knowledge (SWEBOK)*. Angela Burgess, EUA.

ISO/IEC (2001). Software Engineering - Product Quality, ISO/IEC 9126-1. Technical report, International Organization for Standardization.

ISO/IEC (2005). ISO/IEC 25000 - Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Guide to SQuaRE. Technical report.

Jalote, P. (1997). *An Integrated Approach to Software Engineering*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2nd edition.

Khadka, R., Reijnders, G., Saeidi, A., Jansen, S., and Hage, J. (2011). A Method Engineering based Legacy to SOA Migration Method. In *Proceedings of the 2011 27th IEEE International Conference on Software Maintenance*, ICSM '11, pages 163–172, Washington, DC, USA. IEEE Computer Society.

Kitchenham, B. (2004). Procedures for Performing Systematic Reviews. Technical report, Keele University and NICTA.

Kläs, M., Lochmann, K., and Heinemann, L. (2011). Evaluating a Quality Model for Software Product Assessments – A Case Study. In *Proceedings of 4. Workshop zur Software-Qualitätsmodellierung und -bewertung (SQMB'11)*.

Koutsoukos, G., Andrade, L., Gouveia, J., and El-Ramly, M. (2006). Service Extraction. Technical Report D6.2a, Sensoria Project.

Liskov, B. and Guttag, J. (2000). *Program Development in Java: Abstraction, Specification, and Object-Oriented Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

Lochmann, K. and Heinemann, L. (2011). Integrating Quality Models and Static Analysis for Comprehensive Quality Assessment. In *Proceedings of the 2Nd International Workshop on Emerging Trends in Software Metrics*, WETSoM '11, pages 5–11, New York, NY, USA. ACM.

O'Brien, L., Merson, P., and Bass, L. (2007). Quality Attributes for Service-Oriented Architectures. In *Proceedings of the International Workshop on Systems Development in SOA Environments*, SDSOA '07, Washington, DC, USA. IEEE Computer Society.

Papazoglou, M. P. (2003). Service-Oriented Computing: Concepts, Characteristics and Directions. In *Proceedings of the Fourth International Conference on Web*

Information Systems Engineering, WISE ’03, Washington, DC, USA. IEEE Computer Society.

Papazoglou, M. P., Traverso, P., Dustdar, S., and Leymann, F. (2008). Service-Oriented Computing: a Research Roadmap. *International Journal Cooperative Information System*, 17(2):223–255.

Pfleeger, S. L. (2001). *Software Engineering: Theory and Practice*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition.

Pressman, R. S. (2009). *Software Engineering: A Practitioner’s Approach*. McGraw-Hill Higher Education, 7th edition.

Rossi, M. and Sein, M. (2003). Design Research Workshop: A Proactive Research Approach. In *Proceedings of the 26th Information Systems Research Seminar in Scandinavia (IRIS 2003)*.

Runeson, P. and Höst, M. (2009). Guidelines for Conducting and Reporting Case Study Research in Software Engineering. *Empirical Software Engineering.*, 14(2):131–164.

Sanders, J. and Curran, E. (1994). *Software Quality: a Framework for Success in Software Development and Support*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA.

Seaman, C. B. (1999). Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Softwate Engineering*, 25(4):557–572.

Shaw, M. (2002). What Makes Good Research in Software Engineering. *International Journal of Software Tools for Technology Transfer*, 4:1–7.

SOA (2014). Service Oriented Architecture: What Is SOA? online.

SOAP (2014). Simple Object Access Protocol (SOAP) 1.1. W3C Note. . online.

Sommerville, I. (2010). *Software Engineering*. Addison Wesley, Essex,UK, 9 edition.

Steen, M., Strating, O., Lankhorst, T., and ter Doest, H. (2005). *Service-Oriented Enterprise Architecture*, chapter Service-Oriented Software System Engineering: Challenges and Practice, pages 132–154. Hershey, London, UK.

Voelz, D. and Goeb, A. (2010). What is Different in Quality Management for SOA? pages 47–56. IEEE Computer Society.

WSCDL (2014). Web Services Choreography Description Language Version 1.0. online.

WSDL (2014). Web Services Description Language (WSDL) 1.1. W3C Note. . online.

WSQM (2005). Quality Model for Web Services (WSQM2.0). Working draft, OASIS.

Yin, R. K. (2003). *Case Study Research. Design and Methods*, volume 5 of *Applied Social Research Method Series*. Sage Publications, California, USA, third edition.