
Software Architecture based on a Quality Model to Develop Service Oriented Applications

Joyce Meire da Silva França



UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Uberlândia
2017

Joyce Meire da Silva França

**Software Architecture based on a Quality Model
to Develop Service Oriented Applications**

Tese de doutorado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Michel dos Santos Soares

Uberlândia

2017

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

F814s
2017 França, Joyce Meire da Silva, 1989-
Software architecture based on a quality model to develop service
oriented applications / Joyce Meire da Silva França. - 2017.
240 f. : il.

Orientador: Michel dos Santos Soares.
Tese (doutorado) - Universidade Federal de Uberlândia, Programa
de Pós-Graduação em Ciência da Computação.
Inclui bibliografia.

1. Computação - Teses. 2. Controle de qualidade - Teses. 3.
Software - Desenvolvimento - Teses. 4. Software de sistemas - Teses. I.
Soares, Michel dos Santos. II. Universidade Federal de Uberlândia.
Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 681.3

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

The undersigned hereby certify they have read and recommend to the PPGCO for acceptance the thesis entitled “**Software Architecture based on a Quality Model to Develop Service Oriented Applications**” submitted by “**Joyce Meire da Silva França**” as part of the requirements for obtaining the Doctorate’s degree in Computer Science.

Uberlândia, 28 de Junho de 2017

Orientador: _____

Prof. Dr. Michel dos Santos Soares
Universidade Federal de Aracaju

Banca Examinadora:

Prof. Dr. Flávio de Oliveira Silva
Universidade Federal de Uberlândia

Prof. Dr. Marcelo de Almeida Maia
Universidade Federal de Uberlândia

Prof. Dr^a. Andreia Malucelli
Pontifícia Universidade Católica do Paraná

Prof. Dr. Francisco Milton Mendes Neto
Universidade Federal Rural do Semi-Árido

Este trabalho é dedicado aos meus pais Jorge e Eliene França.

Agradecimentos

Seria impossível chegar até aqui sozinha e por isso quero expressar minha gratidão por todos que me ajudaram nessa caminhada.

Toda honra a Deus que tem provido “o essencial que é invisível aos olhos”, incluindo vida, saúde, paz e sabedoria.

Aos meus pais Jorge e Eliene França por dedicarem a vida pela nossa educação. Ao meu pai (*in memoriam*), expresso toda minha admiração e agradecimento pelas doces memórias e inspiração. À minha mãe, meu exemplo de vida, minha gratidão por sempre me aconselhar, incentivar e amar incondicionalmente.

Aos meus irmãos Márcio e Paula França, meu agradecimento à vocês que me ajudam a entender o quanto é bom ter uma família. Muito obrigada pelo apoio, incentivo e pelo companherismo nos momentos difíceis.

Ao meu esposo João Marcus, agradeço pelo apoio de todos os dias nessa caminhada da vida. Suas palavras me acalmam, dão paz e combatem minha ansiedade.

Ao meu orientador, professor Dr. Michel dos Santos Soares. Meu muito obrigada pelos ensinamentos, orientação e colaboração. Agradeço pela paciência dispensada a mim e por todos os conhecimentos transmitidos.

Obrigada ao grupo de pesquisa do EHR compostos por Fernanda, Josimar e professora Adicinéia. Agradeço pela colaboração e expresso meus sinceros votos de sucesso a vocês.

Obrigada a todas as pessoas que colaboraram com a pesquisa através dos questionários de avaliação. Agradeço pela gentileza de dedicarem tempo e esforço na minha pesquisa.

Aos meus amigos, colegas de doutorado e colegas de trabalho no IFNMG. Meu muito obrigada pelos momentos de alegria, suporte e incentivo.

Obrigada a todos da minha grande família conduzida pela exemplar Vó Horacina. Agradeço pelo incentivo e torcida de todos.

E por fim, gostaria de agradecer a agência de pesquisa nacional CNPq (grant 445500/2014-0) pelo apoio financeiro.

“Se podemos sonhar, também podemos tornar nossos sonhos realidade.”
(Walt Disney)

Resumo

A Arquitetura orientada a Serviços (*Service-oriented Architecture - SOA*) é um paradigma que utiliza serviços como elemento fundamental para construir aplicações. SOA tem sido amplamente adotada como um paradigma de arquitetura para sistemas distribuídos tanto na indústria quanto nas pesquisas acadêmicas. Com sistemas de software cada vez mais complexos ao longo do tempo, a garantia de qualidade das aplicações produzidas torna-se cada vez mais importante. Dessa forma, o foco desse trabalho é abordar questões relacionadas com qualidade de aplicações orientadas a serviços nas fases iniciais do processo de desenvolvimento do software. Esse trabalho descreve a partir da literatura um conjunto de princípios de design para melhorar a qualidade em SOA. Uma revisão sistemática da literatura foi conduzida para analisar os princípios de design aplicados ao design de sistemas SOA e sua influência na qualidade da aplicação. Considerando os princípios de design SOA e a carência de modelos de qualidade específicos para SOA baseados em normas ISO/IEC, foi proposto neste trabalho um modelo de qualidade específico para SOA. O modelo de qualidade proposto foi denominado SOAQM e foi desenvolvido com base no modelo de qualidade proposto pela ISO/IEC 25010. O modelo de qualidade SOAQM fornece um conjunto de atributos de qualidade para avaliar qualidade em aplicações SOA. O SOAQM foi usado neste trabalho para guiar o desenvolvimento da aplicação SOA de forma a alcançar os atributos de qualidade essenciais de um sistema orientado a serviços. A aplicação do SOAQM na prática consiste em guiar o desenvolvimento do software em suas fases iniciais por meio da definição de decisões arquiteturais focando na qualidade da aplicação. As decisões arquiteturais foram definidas usando uma abordagem criada para alcançar todos os atributos de qualidade essenciais em um sistema de saúde. O estudo de caso apresentado nesse trabalho trata de um sistema de Registro Eletrônico de Saúde (RES) desenvolvido como um projeto para a tese mas também para atender as necessidades de um hospital público. O processo de desenvolvimento proposto para o RES inclui a definição de arquitetura de software por meio de decisões e visões arquiteturais, e a definição de design por meio de princípios de design propostos na literatura e da utilização de uma linguagem de modelagem específica para SOA chamada

SoaML. A avaliação do RES foi conduzida para analisar dois aspectos importantes. A primeira avaliação aborda o processo de desenvolvimento do RES e foi realizada por um conjunto de especialistas em tecnologia da informação. A segunda avaliação foi realizada para avaliar a ferramenta por usuários do sistema que são profissionais da saúde e utilizando o modelo de aceitação de tecnologia conhecido como Technology Acceptance Model (TAM).

Palavras-chave: Arquitetura orientada a Serviços. Princípios de design. Qualidade de software. Modelo de qualidade. ISO 25010. Atributos de qualidade. Decisões arquiteturais. Visões Arquiteturais. Registro Eletrônico de Saúde.

Abstract

Service-oriented Architecture (SOA) is an architectural paradigm that uses services as fundamental element for development of software applications. SOA has been widely adopted for distributed applications development either in industry and in academic research. Currently, the growing complexity of software requires greater attention about the quality of the produced applications. Software quality assurance becomes increasingly important. Therefore, the focus of this work is to address issues related to quality of service-oriented applications in the early stages of the software development process. This thesis describes from literature a set of design principles to improve quality in SOA. A Systematic Literature Review (SLR) was conducted to analyze design principles applied in SOA design and its influence in application quality. Considering the SOA design principles and the lack of specific quality models for SOA based on ISO standards, a quality model specific for SOA was proposed in this study. The proposed quality model is named Service Oriented Architecture Quality Model (SOAQM) and it was developed with basis on ISO 25010. SOAQM provides a set of quality attributes to evaluate quality in SOA applications. SOAQM was used in this work to guide the development of SOA applications in order to achieve essential quality attributes of a service-oriented system. The use of SOAQM in practice consists of guiding software development in early stages and this approach proposes the definition of Architectural Design Decisions (ADD) with focus on quality. ADD are defined using an approach to achieve essential quality attributes in a health application. The case study presented in this work is an Electronic Health Record (EHR), which has been developed to attend the needs of a public hospital. Development process proposed to EHR includes definition of software architecture through architectural decisions and views and definition of design through principles proposed in literature and a specific modelling language for SOA named SoaML. EHR evaluation was conducted to analyze two important aspects. First evaluation addressed EHR development process that was performed by a set of information technology specialists. Second evaluation was performed to evaluate the tool by users that are health professionals using Technology Acceptance Model (TAM).

Keywords: Service-oriented Architecture. Design Principles. Software Quality. Quality Model. ISO 25010. Quality Attributes. Architectural Design Decisions. Architectural Views. Electronic Health Record Systems.

List of Figures

Figure 1 – Technology Acceptance Model (TAM)	43
Figure 2 – A conceptual view representing SOA concepts and its relations (ERL, 2007)	48
Figure 3 – Logical view of an ESB in the integration layer (BIEBERSTEIN et al., 2008)	50
Figure 4 – Logical view of SOA Reference Architecture (BIEBERSTEIN et al., 2008)	51
Figure 5 – Quality model for external and internal quality by ISO 25010	59
Figure 6 – The “4+1” view model	64
Figure 7 – UML Structural Diagrams	65
Figure 8 – UML Behavioral Diagrams	65
Figure 9 – Class diagram - class representation (OMG, 2015)	66
Figure 10 – Class diagram - Composite aggregation (OMG, 2015)	66
Figure 11 – Class diagram - association (OMG, 2015)	66
Figure 12 – Activity diagram (OMG, 2015)	67
Figure 13 – Use Case diagram (OMG, 2015)	67
Figure 14 – Place Order Service - Service Contract Diagram (OMG, 2012)	69
Figure 15 – Place Order Service - Service Interface Diagram (OMG, 2012)	70
Figure 16 – Place Order Service - Participant Diagram (OMG, 2012)	70
Figure 17 – Place Order Service - Service architecture Diagram (OMG, 2012)	71
Figure 18 – Place Order Service - Capability Diagram (OMG, 2012)	72
Figure 19 – Paper Selection Process	76
Figure 20 – Quality attributes for SOA defined by SOAQM	91
Figure 21 – UML Use Case diagram	119
Figure 22 – UML Activity diagram - Medical appointment process	121
Figure 23 – EHR Implementation View	122
Figure 24 – Service Architecture diagram representing legacy systems	124
Figure 25 – ESB Package	125

Figure 26 – Bus In	125
Figure 27 – Bus Out	125
Figure 28 – UML Class Diagram	126
Figure 29 – Participant Diagram	131
Figure 30 – Participant Diagram with EHR ports	132
Figure 31 – Participant Diagram with legacy systems ports	132
Figure 32 – Participant Diagram with ports, provided and required interfaces . . .	133
Figure 33 – Service Contract Diagrams - Get patient registration service	134
Figure 34 – Service Contract Diagrams - Validate health card service	134
Figure 35 – Service Contract Diagrams - Get users service	135
Figure 36 – Service Contract Diagrams - Get patient service	135
Figure 37 – Service Contract Diagrams - Get patient personal data service	135
Figure 38 – Service Contract Diagrams - Get list of exams	136
Figure 39 – Service Contract Diagrams - Get list of appointments	136
Figure 40 – Service Contract Diagrams - Get list of hospitalization	137
Figure 41 – Service Architecture Diagram for EHR application	137
Figure 42 – Application components	142
Figure 43 – MVC model for application implementation	143
Figure 44 – EHR screenshot - Login	144
Figure 45 – EHR screenshot - Patient location map	145
Figure 46 – EHR screenshot - Patient care map	145
Figure 47 – EHR screenshot - Patient check in	146
Figure 48 – EHR screenshot - Patient list	146
Figure 49 – EHR screenshot - Medical records	147
Figure 50 – EHR screenshot - Anamnesis	147
Figure 51 – EHR screenshot - Medicine prescription	148
Figure 52 – EHR screenshot - Patient discharge	148
Figure 53 – EHR development process	152
Figure 54 – Volunteers Experience	154
Figure 55 – Volunteers Knowledge	155

List of Tables

Table 1 – Research Approaches (CRESWELL, 2014)	36
Table 2 – Overview of Search Results and selected papers	78
Table 3 – Relevant papers	78
Table 4 – Use of SOA in each paper	79
Table 5 – Design Principles used in each paper	81
Table 6 – Validation in each paper	83
Table 7 – Modeling Diagrams used in each study	84
Table 8 – Domain Application of each article	85
Table 9 – Volunteers Opinion about importance of ISO 25010 for SOA - Likert Scale.	92
Table 10 – ISO 25010 characteristics mapped to SOA quality characteristics	93
Table 11 – Architectural Decision D01 for EHR application using SOA	107
Table 12 – Architectural Decision D02 for EHR application using SOA	109
Table 13 – Architectural Decision D03 for EHR application using SOA	111
Table 14 – Architectural Decision D04 for EHR application using SOA	112
Table 15 – Service Catalogue	123
Table 16 – EHR tools and technology	143
Table 17 – Participants Profile	153
Table 18 – Participants Knowledge Level	154
Table 19 – Specialists Opinion about SOAQM	156
Table 20 – Specialists Opinion about EHR Architectural Decisions	157
Table 21 – Specialists’ Opinion about EHR Architectural Views	158
Table 22 – Specialists’ Opinion about EHR Design	159
Table 23 – Experts’ Interview	160
Table 24 – Health professionals Profile	162
Table 25 – Paper acceptance Level	163
Table 26 – Users opinion about EHR utility	164
Table 27 – Users opinion about EHR ease of use	165
Table 28 – Users’ opinion about EHR usability	166

Table 29 – Users’ Interview	167
Table 30 – Published articles - conference proceedings.	174
Table 31 – Published article - journal	175
Table 32 – Unpublished articles	175

Acronyms list

ADD Architectural Design Decisions

API Application Programming Interfaces

EAI Enterprise Application Integration middleware

EHR Electronic Health Record

ESB Enterprise Service Bus

EJB Enterprise JavaBeans

ISO International Organization for Standardization

JPA Java Persistence API

JSF JavaServer Faces

MVC Model View Controller

OASIS Organization for the Advancement of Structured Information Standards

OMG Object Management Group

QOS Quality of Service

RES Registro Eletrônico de Saúde

SLR Systematic Literature Review

SOA Service-oriented Architecture

SOC Service-oriented Computing

SOAP Simple Object Access Protocol

SOAQM Service Oriented Architecture Quality Model

SoaML Service oriented architecture Modeling Language

TAM Technology Acceptance Model

UML Unified Modeling Language

WS-CDL Web Services Choreography Description Language

WSDL Web Services Description Language

WSQM Quality Model for Web Services

XML Extensible Markup Language

Contents

1	INTRODUCTION	27
1.1	Motivation	29
1.2	Methodology	32
1.2.1	Research objectives	32
1.2.2	Research questions	34
1.2.3	Research Approach	36
1.2.4	Research Instruments	38
1.3	Contributions	43
1.4	Thesis outline	44
2	THEORETICAL BACKGROUND	47
2.1	Service Oriented Architecture	47
2.1.1	SOA key concepts	48
2.1.2	SOA Reference Architecture	51
2.2	Design Principles	52
2.2.1	Software Design Principles	53
2.2.2	SOA Design Principles	53
2.3	Software Quality	56
2.4	SOA Quality Models	57
2.5	ISO 25010 – System and software quality models	58
2.6	Software Architecture	62
2.6.1	Architectural Decisions	63
2.6.2	Architectural Views	63
2.7	Unified Modeling Language - UML	64
2.8	SOA Modeling Language - SoaML	68
3	REVIEW OF DESIGN PRINCIPLES FOR SOA	73
3.1	Protocol	73

3.1.1	Research Questions	73
3.1.2	Search Process	75
3.1.3	Inclusion and Exclusion Criteria	75
3.2	Characterization	79
3.2.1	Use of SOA	79
3.2.2	Most used SOA design principles	80
3.2.3	Benefits of applying SOA design principles	82
3.2.4	SOA applications evaluation	83
3.2.5	SOA applications design	83
3.2.6	Domains of SOA applications	85
3.3	Discussion	86
3.4	Research Directions	87
4	QUALITY MODEL FOR SOA	89
4.1	Problem Description	89
4.2	SOAQM - Quality Model for SOA applications	90
4.2.1	Functional Suitability	92
4.2.2	Performance efficiency	94
4.2.3	Compatibility	95
4.2.4	Usability	96
4.2.5	Reliability	97
4.2.6	Security	97
4.2.7	Maintainability	99
4.2.8	Portability	100
4.3	Comparison with other SOA quality models	100
4.4	Discussion	103
5	EHR ARCHITECTURAL DESIGN DECISIONS	105
5.1	Description of Current Situation at the Hospital	105
5.2	Architectural Design Decisions	106
5.2.1	Architectural Design Decision D01 - SOA	107
5.2.2	Architectural Design Decision D02 - ESB	109
5.2.3	Architectural Design Decision D03 - Mobile devices	110
5.2.4	Architectural Design Decision D04 - Security	111
5.3	Comparison with related works	113
5.4	Discussion	114
6	EHR ARCHITECTURAL VIEWS	117
6.1	Hospital Stakeholders'	117
6.2	Architectural Views	118

6.2.1	Scenarios View	118
6.2.2	Business Process View	120
6.2.3	Implementation View	122
6.2.4	Process View	124
6.2.5	Logical View	126
6.3	Comparison with related works	127
6.4	Discussion	127
7	EHR DESIGN	129
7.1	Detailed Design of the EHR	129
7.2	SoaML diagrams for EHR	130
7.2.1	Participant Diagram	130
7.2.2	Service Contract Diagram	133
7.2.3	Service Architecture Diagram	136
7.3	Comparison with related works	138
7.4	Discussion	139
8	EHR IMPLEMENTATION	141
8.1	EHR development experience	141
8.2	EHR implementation organization	141
8.3	EHR development Tools	143
8.4	EHR Functionalities	144
8.5	Discussion	149
9	EXPERIMENTAL RESULTS	151
9.1	Evaluation of the development process	151
9.1.1	Protocol for Experts' Evaluation	151
9.1.2	Participants	153
9.1.3	Results	155
9.1.4	Interviews	159
9.2	Evaluation of the EHR Application	161
9.2.1	Protocol for End Users' Evaluation	161
9.2.2	Participants	162
9.2.3	Results	163
9.2.4	Interviews	166
9.3	Discussion	168
10	CONCLUSION	169
10.1	Answer to research questions	170
10.2	Main Contributions	172
10.3	Future Work	173

10.4	Contributions in Bibliography Production	174
	BIBLIOGRAPHY	177
	APPENDIX	191
	ANNEX	231

I hereby certify that I have obtained all legal permissions from the owner(s) of each third-party copyrighted matter included in my thesis, and that their permissions allow availability such as being deposited in public digital libraries.

Joyce Meire da Silva França

Introduction

Service Oriented Architecture (SOA) is an architectural style in which services are the main element for developing distributed systems (PAPAZOGLOU, 2003). A service is a capability of the business organization that is implemented and made available on the Internet so that other applications can access it (BIEBERSTEIN et al., 2008). Services are autonomous, platform-independent computational entities that can be used to create large, complex applications. SOA has been widely adopted (HIRSCH; KEMP; ILKKA, 2006) (DU; SONG; MUNRO, 2010) (KHADKA et al., 2013) (AHMED; LETCHMUNAN, 2015) to develop distributed applications with the promise of integrating legacy systems and better agility to develop applications by reusing services.

Considering the important role of SOA in organizations, quality should be treated as a key issue. In addition, software systems are becoming increasingly complex over time and, thus, quality assurance is becoming increasingly important as well (BOEHM, 2006) (HUANG et al., 2012). Evaluation of software quality is an important activity in the software development process. According to some authors (SANDERS; CURRAN, 1994) (AGARWAL; RATHOD, 2006) (SAVOLAINEN; AHONEN; RICHARDSON, 2012), quality is crucial for success of a software application. Software market is increasingly global, and companies will face difficulties in succeeding unless they produce high quality products and services.

Quality topics are the most cited issues in SOA context studies since 2000 (GU; LAGO, 2009) (AHMED; LETCHMUNAN, 2015). According to (GU; LAGO, 2009), the topic of ‘quality’ has been addressed by the highest number of challenges between 2000 and 2008 and these studies address quality related issues, including security, reusability, flexibility, interpretability, and performance. According to a systematic review (AHMED; LETCHMUNAN, 2015), SOA quality and Quality of Service (Quality of Service (QOS)) were considered the highest challenge in studies published from 2009 to 2014.

Several studies (OASIS, 2005) (GEHLERT; METZGER, 2008) (CHOI; KIM, 2008) (BELL, 2010) (GOEB; LOCHMANN, 2011) (ALLANQAWI; KHADDAJ, 2013) have been addressing how to measure quality in SOA applications. As result of a systematic review

(ORIOLO; MARCO; FRANCH, 2014), literature have proposed 47 quality models which are specific to Web services. This means there is a serious concern in developing quality SOA applications. Many advances in SOA quality have been proposed in the literature, however, many research gaps were found and thus much more needs to be done in this research area. By observing the works proposed in the literature, it is possible to notice that there is a need for design principles specifically proposed for SOA (PAPAZOGLU et al., 2007), quality models for SOA applications based on ISO 25010 (ORIOLO; MARCO; FRANCH, 2014) and quality attributes and metrics to measure SOA quality (GOEB; LOCHMANN, 2011).

This research project is mainly aimed at addressing relevant issues in the quality of service-oriented applications that have not been adequately explored. First, this work presents a systematic review on the current 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 SOA principles are implemented, in which domains they are most used, what advantages are obtained in their use, and how the application of these principles influences quality of the application. The results found in this systematic review defined some research directions such as most of the studies proposed their SOA systems without considering quality during development.

This thesis proposes a SOA quality model based on ISO 25010 named Service Oriented Architecture Quality Model (SOAQM). SOAQM is proposed as a quality model which is specific for SOA applications. SOAQM is based on the most recently issued ISO 25010 standard. This is different from most other works on SOA quality models, which are based on ISO 9126 or do not have a basis on a known quality model (ORIOLO; MARCO; FRANCH, 2014). SOAQM provides a set of quality attributes that are essential in a SOA application.

SOAQM has been applied in a case study in the health domain and evaluated by software professionals. An Electronic Health Record (EHR) application was developed using SOAQM to guide the development process. This approach involves using SOAQM to help in defining Architectural Design Decisions (ADD), Architectural Views and Design. The proposal is to use SOAQM to guide the development process with focus on software quality. This study presents EHR as case study, as an application in the health domain developed to attend a public hospital. The university hospital at the Federal University of Sergipe located in Aracaju, Brazil, was used as reference for understanding the hospital business processes and the architecture of the legacy systems. The Information Technology team and hospital staff were interviewed for organizational knowledge and system requirements elicitation.

Development process proposed to develop the EHR system includes definition of software architecture through architectural decisions and views. The set of ADD defined

to develop the EHR was guided by SOAQM, therefore the approach aimed to achieve essential quality attributes. The proposed architecture for EHR also includes a set of architectural views. Views can express the architecture of a software system from the perspective of specific system concerns. Definition of multiple views is considered in the literature a feasible strategy which facilitates understanding by groups of stakeholders (BELL, 2008) (ROZANSKI; WOODS, 2011).

EHR development process also included definition of system design by considering principles proposed in literature. Design principles found in literature represents that these concepts are widely adopted and accepted in software industry (ERL, 2007). Therefore, some principles could be applied in SOA context for developing the EHR.

Part of the EHR design which involved integration with legacy systems was modeled using SoaML (OMG, 2012), a specific modeling language for SOA applications. SoaML diagrams allows to model services that represent the key concept in SOA applications.

Finally, this study proposes an evaluation divided into two parts. First evaluation addressed aspects related to software development process. Experts opinion was required to analyze the development process used during development of the EHR. Specialist evaluation was performed through questionnaire and interviews. Second evaluation had focus in providing analysis about EHR system users. This evaluation was performed with health professionals using Technology Acceptance Model (TAM) (DAVIS; BAGOZZI; WARSHAW, 1989).

1.1 Motivation

The emergence of SOA paradigm introduced new challenges, as for instance, design principles for SOA (PAPAZOGLU et al., 2008), modelling languages to represent services (PAPAZOGLU et al., 2008) and specific quality models (GOEB; LOCHMANN, 2011). These research gaps represents underexplored topics in literature about SOA quality, for instance, lack of tools for measuring quality of SOA applications (GOEB; LOCHMANN, 2011) and lack of service design principles (PAPAZOGLU et al., 2008). In addition, few studies were published considering the ISO 25010 standard (ISO/IEC, International Organization for Standardization, 2011a) that deals with quality and has been recently proposed to replace the ISO 9126 (GOEB; LOCHMANN, 2011) (ISO/IEC, 1991).

In this thesis, five topics related to SOA were addressed since they were not sufficiently explored in the literature. The next paragraphs are dedicated to explaining the justification for each of these five topics.

SOA design principles

Several publications have been proposed in literature to define design principles for software engineering such as abstraction, coupling and cohesion, decomposition and

modularization and encapsulation/information hiding (see more in Section 2.2.1). However, it is important to know whether these design principles for general systems can be directly applied in SOA projects or how they can be adapted to SOA. In addition to this, it is important to investigate if the use of these design principles for SOA can improve SOA quality. Some questions related with SOA design principles are answered through a systematic literature review presented in detail in Chapter 3.

SOA Quality

The other issue of this work addresses SOA quality models. A systematic review proposed by Oriol et al. (ORIOLO; MARCO; FRANCH, 2014) presented 47 specific quality models to Web services. Most of these proposals did not take into consideration ISO standards. Only 6 out of 47 models were based on an ISO standard. According to (ORIOLO; MARCO; FRANCH, 2014) these quality models, in general, are too complex, immature and have lack precision definitions. In addition to this, these quality models are based in a ISO version proposed in 1991 and software field has evolved. Therefore, according to (GOEB; LOCHMANN, 2011) and (ORIOLO; MARCO; FRANCH, 2014) there is a need to propose a SOA quality model based on ISO 25010 that is the most recent version of this standard. This thesis, in Chapter 4, proposes a SOA quality model named SOAQM based on ISO 25010 to provide one answer to this research gap.

SOA Architectural Decisions

Some studies in the literature have been dedicated to define Architectural Design Decisions (ADD) for SOA applications. In (GU; LAGO, 2008), the authors defined two architectural decisions for a generic corporative system, addressing service discovery and service monitoring performance. In (ZIMMERMANN et al., 2007), the authors defined architectural decisions for two SOA applications, one system for order management and the other in the finance industry. These studies proposed decision meta-models and guidelines for formulating and documenting design decisions, and illustrated their methods in the context of the SOA design space. As for conclusion, the authors could observe that documenting ADD provided quality improvements on projects. However, they did not detail what were the achieved quality improvements.

A systematic mapping (TOFAN et al., 2014) presented an overview of the past and future of ADD observing the past ten years. This systematic mapping found 144 relevant papers to present an overview of the state of research on architectural decisions. Only 9 out of 144 studies proposed ADD for SOA applications. In general, these studies present approaches for decision models and templates for documenting SOA decisions. However, these studies addressing SOA ADD did not have focus

on meeting quality attributes. Only 7 out of 144 papers explicitly treat quality attributes and had focus on specific quality attributes such as security, reliability, usability, and scalability.

This thesis, in Chapter 5, proposes an approach to use ADD for EHR SOA system with focus on quality. The proposal is to use SOAQM to guide the development process with focus on important quality attributes since early phases of SOA application development.

SOA Architectural Views in Health Systems

Some studies in the literature have been dedicated to define views for SOA applications in the health domain. Two studies presented in (CHO et al., 2010) and (EL-SAPPAGH; EL-MASRI, 2014) propose SOA applications for clinical decision support. In both studies, only the implementation view of the EHR architecture is presented. One study in the health domain, presented in (TRAORE; KAMSU-FOGUEM; TANGARA, 2016), addresses telemedicine services and its related software architecture. The proposed architecture is, in fact, a model-driven approach to transform UML models into WSDL source code.

Studies (MOOR et al., 2015) and (FABIAN; ERMAKOVA; JUNGHANNS, 2015) applied SOA concepts to develop an EHR system. In these works, the EHR architecture is described as a whole big picture, in which the views are considered together. For instance, in (FABIAN; ERMAKOVA; JUNGHANNS, 2015) the architecture brings in the same picture processes, structural elements, and deployment infrastructure. In addition, application scope and which services should be implemented are not described. Work presented in (GAZZARATA; GAZZARATA; GIACOMINI, 2015) proposes an EHR SOA application with focus on secure access, and presents the architecture using only the process view.

Study (BLOBEL, 2011) uses SOA to develop an application with focus on Personal health (pHealth) environments and ontologies. The architecture specification is mainly concerned in presenting only the implementation view.

Most studies which address the implementation of an EHR using SOA concepts have focus on the application itself, but not on the architectural aspects for development. Even those studies which present the software architecture, only describes one or two views, or even a multiple number of views in only one box diagram, which may be confusing for most stakeholders. Therefore, in Chapter 6, a multiple-view architecture was proposed for the developed EHR application.

SOA Design with SoaML

SOA applications have been modeled using many general purpose modeling languages. For example, studies (PEREPLETCHIKOV; RYAN, 2011) and (MON-

SIEUR; SNOECK; LEMAHIEU, 2012) provide design models for their SOA application using Data Flows. Article (MONSIEUR; SNOECK; LEMAHIEU, 2012) presented two examples of SOA applications in the health industry. First one is a hospital application which allows nurses and doctors to demand medicines provided by a pharmacist. Second one is an application which consists of a real-life business case at a Belgian banking and insurance company. Models for both SOA applications were described using Data Flows. Studies (WU et al., 2009) and (PEREPLETCHIKOV; RYAN, 2011) used Flowcharts. The study presented in (KRAMMER et al., 2011) used a model called Functionality and Service Graph (a graph defined to identify services based on an aggregation or disaggregation of functions or functionality).

UML diagrams have also been considered for modeling SOA applications. In (KANNAN; BHAMIDIPATY; NARENDRA, 2011), SOA was used to design a solution for student course registration using UML Sequence diagrams. In (TIAN; HUANG, 2012), the authors propose a solution to find and access geospatial data over the Internet, and have also used UML Sequence diagrams to present system design. Activity diagram (GRANELL; DÍAZ; GOULD, 2010), Class diagram (PEREPLETCHIKOV; RYAN, 2011), and Components diagram (CHO et al., 2010) have also been considered for SOA design.

Other studies found in the literature (KING et al., 2010), (VESCOUKIS; DULAMIS, 2011), (KABIR; HAN; COLMAN, 2014), (GARCÍA-SÁNCHEZ et al., 2013) present their SOA projects and scope without using any modeling diagram. These studies present services and main requirements using natural language or complicated drawings or even code snippets trying to explain project scope. Due to lack of standardization, understanding the diagrams is difficult and error prone. This issue is even more challenging when developing SOA solutions in industry with many developers, contractors and companies cooperating to develop an application.

This thesis, in Chapter 7, proposes an approach to use SoaML to design the EHR application. As result of this SoaML experience, it was possible to notice that it is important to document diagrams that can represent services and SOA fundamental concepts.

1.2 Methodology

1.2.1 Research objectives

The main objective of this study is to propose an approach to consider SOA quality in early stages in software development process based on SOA and consider these quality concerns for architecting SOA applications. The specific objectives of this research are:

1. **To evaluate the state of the art of SOA design principles in practice**

This thesis aims to relate from literature design principles that improve the development and quality of SOA applications. Existing SOA principles have to be classified, for example, in relation to their real applicability in practice, measured advantages and scope of use. Thus, within this research a systematic review has been developed to investigate these issues.

2. **To contribute with SOA quality attributes based on ISO 25010**

Several studies are based on the ISO 9126 standard that was replaced by ISO 25010. This study aims to contribute with a new quality model to guide application development with focus on quality attributes based on ISO 25010.

3. **To evaluate the quality model in a case study**

Software Engineering studies, in general, have received critics regarding the few amount of research with quantitative validation (SHAW, 2002) (RUNESON; HÖST, 2009). One objective of this research is to design and implement a SOA application in the health domain to evaluate the proposed research. The case study is developed to suggest design principles for SOA that improve quality application. The qualitative analysis of case studies will be performed by applying questionnaires and interviews with users, students and professionals.

4. **To propose an approach to apply quality attributes in a SOA application**

Several quality models are proposed to define quality attributes for applications, however sometimes it is difficult to know how to make decisions to obtain these quality attributes. The proposal consists in defining a set of Architectural Design Decisions in order to achieve essential quality attributes in SOA applications.

5. **To propose a multiple-view architecture for SOA application**

Some studies in the literature (CHO et al., 2010) (BLOBEL, 2011) (EL-SAPPAGH; EL-MASRI, 2014) (TRAORE; KAMSU-FOGUEM; TANGARA, 2016) address the implementation of an EHR using SOA concepts with focus on the application itself, but not on the architectural aspects for development. Even those studies which present the software architecture, only describes one or two views, or even a multiple number of views in only one box diagram, which may be confusing for most stakeholders. Therefore, a multiple-view architecture was proposed in this study for representing the SOA application architecture.

6. **To use a modeling language for SOA**

In the literature, SOA applications have been modeled by several manners, as for instance, using Data Flows (PEREPLETCHIKOV; RYAN, 2011) (MONSIEUR; SNOECK; LEMAHIEU, 2012), Flowcharts (WU et al., 2009) (PEREPLETCHIKOV;

RYAN, 2011), UML Sequence diagrams (KANNAN; BHAMIDIPATY; NARENDRA, 2011) (TIAN; HUANG, 2012), Activity diagram (GRANELL; DÍAZ; GOULD, 2010), Class diagram (PEREPLETCHIKOV; RYAN, 2011) and Components diagram (CHO et al., 2010). One of the proposals of this thesis is to present SOA application design using a modeling language for SOA. SoaML was used for modeling SOA application and this experience is reported in this thesis.

1.2.2 Research questions

The research problem addressed by this study is divided into six sub-problems presented as follows.

1. SOA design principles

Initially, the purpose of this study is to focus on a research challenge presented by Papazoglou in (PAPAZOGLU 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 years.

The main objective of this topic is to investigate whether literature provides all the fundamental principles for design services, how these principles are used and which advantages are obtained in their use. One main research question is proposed here.

RQ1 - How are current SOA design principles being applied in practice?

This research question is answered in Chapter 3.

2. Quality model for SOA quality based on ISO 25010

Several quality models specific for SOA have been presented in the literature. However, most quality models were proposed in the literature without using reference from any standard. Only some quality models are in accordance with the old ISO 9126. These quality models for SOA are based on ISO 9126 that was proposed in 1991. This situation represents a problem because software development has had great progress since 1991. A new version of ISO was proposed in 2011 and was named ISO 25010. ISO 25010 is more complete and has a wider range of quality requirements for software. Thus, there is shortage of research studies addressing quality attributes based on ISO 25010. Another important analysis is to verify if the existing quality attributes based on ISO 9126 complies with the ISO 25010, and if ISO 25010 will bring any benefit when compared with ISO 9126.

The second research question is: *RQ2 - What ISO 25010 characteristics are most suitable when developing SOA applications?*

The proposal of RQ2 question is to investigate the applicability of ISO 25010 to SOA applications. According to ISO 25010 authors, this standard is proposed to any

kind of software system. However, SOA is a paradigm with different particularities. Therefore, the aim is to analyze all the quality attributes (characteristics and sub-characteristics) proposed by ISO 25010 and determine which of them can be directly applicable to define quality in SOA.

Research question RQ2 is addressed in Chapter 4.

3. Approach to consider quality in early stages of software development

Quality models proposed in literature and cited as theoretical foundations in Chapter 2 define quality attributes for SOA applications. However, it is difficult to know which practices have to be performed to achieve these quality attributes in the final software product. This thesis presents an approach to define architectural decisions to achieve essential quality attributes in a SOA application in the health domain. In this way, the quality model guides decisions with focus on quality from early stages of development.

The third research question is: *RQ3 - How a quality model can guide architectural design decisions in the development of a SOA Application?*

This research question is addressed in Chapter 5.

4. EHR architecture using multiple views

As mentioned before, some studies proposed in the literature present SOA applications in health domains using only one or two views to represent system architecture. However, software engineering authors suggest the use of multiple views due to the improved stakeholders understanding. In addition to this, EHR system has been proposed to hospital stakeholders that include software developers, IT director and professionals who are not IT experts such as administrators and clinicians.

The fourth research question is: *RQ4 - How to propose a SOA EHR application Architecture using multiple views?*

This research question RQ4 is addressed in Chapter 6.

5. EHR design using SoaML

Several studies provide SOA applications design using general purpose diagrams such as the ones of UML (see Section 1.1). However, general purpose diagrams are not appropriated to design SOA applications (PAPAZOGLU et al., 2008).

Recently, a modeling language specific for SOA named SoaML was proposed to design service applications. SoaML provides a set of diagrams that model the system detaching services as core elements.

The fifth research question is: *RQ5 - How to design a SOA EHR Application using SoaML?*

This research question is addressed in Chapter 7. In this chapter, the SoaML diagrams proposed for the case study in the health domain are presented.

6. Report EHR implementation and functionalities

EHR application proposed as case study of this research was developed to be used in a public hospital. Therefore, it is important that the source code be accompanied by consistent documentation so that the system can be submitted to future maintenance and evolutions.

The sixth research question is: *RQ6 - How to report and document a SOA EHR Application?*

The research question RQ6 is addressed in Chapter 8.

1.2.3 Research Approach

In order to fulfill the research objective and answer the research questions, the research approach used is described in this section.

According to (CRESWELL, 2014), four dominant research philosophies can be characterized that are Positivism, Constructivism, Critical Theory and Pragmatism. Table 1 presents the key concepts related to these philosophies.

Philosophy	Approach
Positivism	· Determination
	· Reductionism
	· Empirical observation and measurement
	· Theory verification
Constructivism	· Understanding
	· Multiple participant meanings
	· Social and historical construction
	· Theory generation
Critical Theory	· Political
	· Power and justice oriented
	· Collaborative
	· Change-oriented
Pragmatism	· Consequences of actions
	· Problem-centered
	· Pluralistic
	· Real-world practice oriented

Table 1 – Research Approaches (CRESWELL, 2014)

Positivism assumptions have represented the traditional form of research, and these assumptions hold true more for quantitative research than qualitative research. Positivists are reductionists, in that they study things by breaking them into simpler components. This corresponds to their belief that scientific knowledge is built up incrementally from

verifiable observations and inferences based on them. The problems studied by positivists reflect the need to identify and assess the causes that influence outcomes, such as the ones found in experiments.

Constructivism is typically seen as an approach to qualitative research. Constructivists perform methods that collect rich qualitative data about human activities, from which theories might emerge. Constructivists focus on the specific contexts in which people live and work in order to understand the historical and cultural settings of the participants.

Critical theorists affirms that research inquiry needs to be intertwined with politics and a political change agenda to confront social oppression at whatever levels it occurs. Critical theorists use participatory approaches in which the groups they are trying to help are engaged in the research, including helping to set its goals. In Software Engineering, it includes research that actively seeks to challenge existing perceptions about software practice, most notably the open source movement, the process improvement community and the agile community (EASTERBROOK et al., 2007).

Pragmatism arises out of actions, situations and consequences rather than antecedent conditions. There is a concern with applications -what works - and solutions to problems. Instead of focusing on methods, researchers emphasize the research problem and use all approaches available to understand the problem. In essence, pragmatism adopts an engineering approach to research. It values practical knowledge over abstract knowledge, and uses whatever methods are appropriate to obtain it.

This thesis is based on an engineering approach to research, in the sense that it makes use of a quality model to propose techniques and methods to create artifacts such as architecture and designs for service-oriented systems. Quality attributes were used to guide the development of an EHR system through definition of a set of architectural decisions. Each decision described for the EHR was defined to achieve essentials quality attributes to each aspect of system solution. In this way, the EHR system was divided into smaller concerns such as architecture of environmental and logical solution. Thus, reducing the system into smaller, more comprehensible components, is of fundamental importance. Indeed, structuring the design decisions using a common set of attributes and data has more potential do improve comprehension. Because of this, positivism, with its reductionism property, is a feasible choice for research philosophy.

Another important aspect of this research is that it opens discussions for different people, who have different opinions, about the added value of Software Engineering methods and languages. Part of this research was performed in an environment in which the users could manifest their opinions, which makes the social context important as well. In this thesis, a quality model for SOA systems was proposed and evaluated by a set of specialists that have experience in SOA development. In addition to this, EHR system proposed in this research was developed using an approach with focus on quality that also was eval-

uated by IT professionals. Another aspect evaluated in EHR system is related to users. Health professionals were interviewed to manifest their opinion about EHR usability and utility. As a result, interpretivism also has a minor role in this research

1.2.4 Research Instruments

Research instruments are used to collect and later analyze data in order to create models, methods or theories of a research strategy. Which research instrument to use depends on many factors such as the type of research questions, the research objective, and the amount of existing theories available. A set of research methods was applied during this research. Each one is briefly described as follows.

1.2.4.1 Systematic Literature Review

Systematic Literature Reviews (SLR) in Software Engineering have gained importance in recent years (KITCHENHAM et al., 2009) (KITCHENHAM; BRERETON, 2013) (ZHANG; BABAR, 2013) (WOHLIN; PRIKLADNICKI, 2013). Systematic Literature Reviews have been applied in Software Engineering with the purpose of finding empirical evidence on many domains, research areas, and activities of a software's life cycle, including Aspect-Oriented Programming (ALI et al., 2010), Requirements Engineering (PACHECO; GARCIA, 2012) and Software Testing (ENGSTROM; RUNESON; SKOGLUND, 2010).

Specifically for Service-Oriented Architecture, some publications were found. In study (PALACIOS; GARCIA-FANJUL; TUYA, 2011), the authors presented a SLR to identify the state of the art in research on testing in Service Oriented Architectures with dynamic binding. Another study (LANE; RICHARDSON, 2011), presented that SLR has as main objective to identify process models for developing SOA applications. Three main objectives of the SLR presented in paper (MAHDAVI-HEZAVEHI; GALSTER; AVGERIOU, 2013) are to assess methods for handling variability in quality attributes of service-based systems, to collect evidence about current research that suggests implications for practice, and to identify open problems and areas for improvement. Considering December of 2014 (the date the SLR was finalized), there are no specific SLR publications on design principles for SOA, their benefits when applied in practice, and their relation to application quality, as described in this thesis.

SLR proposed in Chapter 3 of this thesis was performed with basis on recommendations of Kitchenham B. (KITCHENHAM, 2004). Kitchenham B. proposed guidelines to perform the review process. A systematic review involves several activities. The stages in a systematic review can be summarised into three main phases: Planning the Review, Conducting the Review and Reporting the Review. The following paragraphs are dedicated to explain how these phases were performed.

1- Planning the Review

Planning the review includes identification of the need for a review. The review proposed in this thesis emerges from the need to identify if one research challenge presented in 2008 by Papazoglou (PAPAZOGLU et al., 2008) was addressed. The research challenge is related to creation of design principles for engineering service applications. It is important to find out if this gap, proposed in 2008, has been adequately addressed by researchers in the past years.

A SLR has been developed to investigate issues related to design principles and their real applicability in practice, measured advantages and scope of use. Therefore, some primary aspects to be investigated such as to certify if the literature provides all the fundamental principles for design services; the approach used to apply these principles; which advantages are obtained in their use; which areas are applying principles in practice; for which proposals SOA has been considered (only for legacy systems or new systems too); and verify if there is experimental evidence on the influence of these principles in the quality of the application.

Next activity consists of development of a review protocol. A review protocol specifies the methods that will be used to undertake a specific systematic review. The protocol for this SLR was described through definition of research questions, selection criteria considering inclusion and exclusion criteria and strategy for data extraction. Therefore, six research questions were defined for the SLR and they are listed and justified in Chapter 3. Selection criteria in search focus on studies that use design principles for SOA. Finally, strategy for data extraction includes defining the type of search that was defined to be automatic or manual search.

2- Conducting the Review

First activity in conducting the review consists in generating a search strategy. Preliminary searches were performed to identify the comprehensiveness of the search string. This activity was important because the first search string was exclusive for SOA. However, some studies use the term of Service-oriented Computing (SOC). Therefore, the second search string has been improved to address SOA and SOC concepts: (*“service oriented architecture” or “service oriented computing”*) and (*“design principles”*) .

Search strategies in SLR often involve automatic keyword searches in digital libraries. Venues searched included ACM Digital Library¹, ScienceDirect², Springer³, and IEEE Xplore⁴.

¹ <http://dl.acm.org/>

² <http://www.sciencedirect.com/>

³ <http://www.springer.com>

⁴ <http://ieeexplore.ieee.org>

A manual selection of articles in relevant venues was also performed. A search in IEEE TSE (IEEE Transactions on Software Engineering)⁵ and ACM TOSEM (ACM Transactions on Software Engineering and Methodology)⁶ was also performed to include articles in SLR.

3- Reporting the review

This stage consists in communicate the results of a systematic review. It is recommended to report in at least two formats: in a technical report or in a section of a PhD thesis and in a journal or conference paper. Therefore the SLR presented in this thesis and detailed in Chapter 3 was also published in a journal (SOARES; FRANÇA, 2016).

1.2.4.2 Case Study

The case study proposed in this thesis was developed in the health domain. An Electronic Health System (EHR) was developed to attend the needs of a public hospital. EHR was proposed to automate business process in the university hospital of the Federal University of Sergipe located in Aracaju, Brazil.

According to the International Organization for Standardization ISO/TR 20514:2005 (ISO, 2005), EHR means a repository of patient data in digital form, stored and exchanged securely, and accessible by multiple authorized users. EHR is a software system that contains any record of health, clinical or medical in electronic or digital format in the context of patient care.

According to a literature review in EHR (HäYRINEN; SARANTO; NYKÄNEN, 2008), EHR concept comprised a wide range of information systems, from files compiled in single departments to longitudinal collections of patient data. According to this systematic review, the EHRs are used in primary, secondary and tertiary care. Primary care is health care provided in the community by the staff of a general practice. Secondary care is medical attention provided by a specialist facility upon referral by a primary care physician, and tertiary care is provided by a team of specialists in a major hospital. EHRs are used by different health professional teams, including physicians, nurses, radiologists, pharmacists, laboratory technicians and radiographers. Some information is also registered by patients, but this is validated by physicians.

A systematic literature review (NGUYEN; BELLUCCI; NGUYEN, 2014) recently proposed highlighted what has been implemented in EHR application worldwide and they have found important aspects. According to these authors, the adoption of EHR will increase significantly worldwide due to its several benefits. This review confirms the potential of this technology to aid patient care and clinical documentation; for example, in improved documentation quality, increased administration efficiency, as well as better

⁵ <http://www.computer.org/web/tse>

⁶ <http://tosem.acm.org/>

quality, safety and coordination of care. Another important aspect detached in this review is that interoperability, portability and support for mobile work have emerged as important system quality attributes for EHRs.

1.2.4.3 Proposed Evaluation

SOAQM and EHR system proposed in this research must be evaluated. The methods chosen in this research for evaluation were surveys and interviews. Survey research is used to identify the characteristics of a broad population of individuals (EASTERBROOK et al., 2007). The defining characteristic of survey research is the selection of a representative sample from a well-defined population, and the data analysis techniques used to generalize from that sample to the population, usually to answer base-rate questions.

1.2.4.4 Experts Opinion

“Expert knowledge” is what qualified individuals know as a result of their technical practices, training, and experience (BOOKER; MCNAMARA, 2004). Expert-opinion elicitation is a formal process of obtaining information or answers to specific questions about certain issues that are needed to meet certain analytical objectives. There is an interest in these methods that can be attributed in increased interest in reliability and risk methods for dealing with technology, environment, and socioeconomic problems and challenges (COOKE, 1991).

The expert opinion process generally follows the following steps (MOSLEH et al., 1987) (LI; SMIDTS, 2003).

1. **Problem Statement** The background and problem need to be clearly articulated and defined.
2. **Selection of Experts** A fair number of experts need to be identified based on a set of criteria. The set of criteria should include the credibility, knowledgeability, and dependability of experts.
3. **Expert Training** Expert training is an important aspect of the process of familiarizing experts with the goals of the elicitation process and the details of the issues involved. The purpose of training is to ensure that there is common understanding of the issue being addressed and that the experts would be responding to the same elicitation question.
4. **Elicitation of Opinions** This step poses the right question and ensures conditions conducive to an elicitation exercise.
5. **Aggregation of Opinion** The basic idea here is to reach an aggregated opinion or a consensus based on which a decision can be made. This aggregated opinion or

consensus is generally a scalar value representing the scale of each alternative and is termed “utility”. The decision is generally the alternative with the least or highest value of utility.

6. Decision Making This step makes the decision based on the aggregated opinion.

This thesis presents, in Chapter 9, the process that was performed to evaluate the approach used in EHR development. EHR development process evaluation was performed using experts opinion (COOKE, 1991). The process of experts opinion evaluation followed some steps proposed by (MOSLEH et al., 1987) and (LI; SMIDTS, 2003).

The participants answered a questionnaire composed of 20 statements in which they made their opinions explicit about EHR development process. A 5-point Likert scale (LIKERT, 1932) was proposed to measure perceived attitudes of the employees by providing a range of responses to each statement. The scale ranged from (1) strongly disagree, (2) disagree, (3) neutral, (4) agree, and (5) strongly agree.

1.2.4.5 Technology Acceptance Model (TAM)

Technology Acceptance Model (TAM) has been developed by Davis(1989) (DAVIS; BAGOZZI; WARSHAW, 1989) and is intended to investigate how people and organizations behave in the face of new technology. TAM focus on why users accept or reject information technology and how to improve acceptance, thereby providing support to predict and explain acceptance.

In TAM model, there are two relevant factors in computer use behaviors that are perceived usefulness and perceived ease of use.

Perceived usefulness (U) is related to the prospective user’s subjective probability that using a specific application system will enhance his or her job or life performance. Usefulness factor is related to how much the user believes that using a certain technology can improve its performance.

Perceived Ease of Use (EOU) can be defined as the degree to which the prospective user expects the target system to be free of effort. EOU is related to how much the user believes that using certain technology it can minimize efforts to perform a certain task.

According to TAM, ease of use and perceived usefulness are the most important determinants of actual system use. These two factors are influenced by external variables. External variables are usually manifested in social factors, cultural factors and political factors. Social factors include language, skills and facilitating conditions. Political factors are mainly the impact of using technology in politics and political crisis. The attitude to use is concerned with the user’s evaluation of the desirability of employing a particular

information system application. Behavioral intention is the measure of the likelihood of a person employing the application.

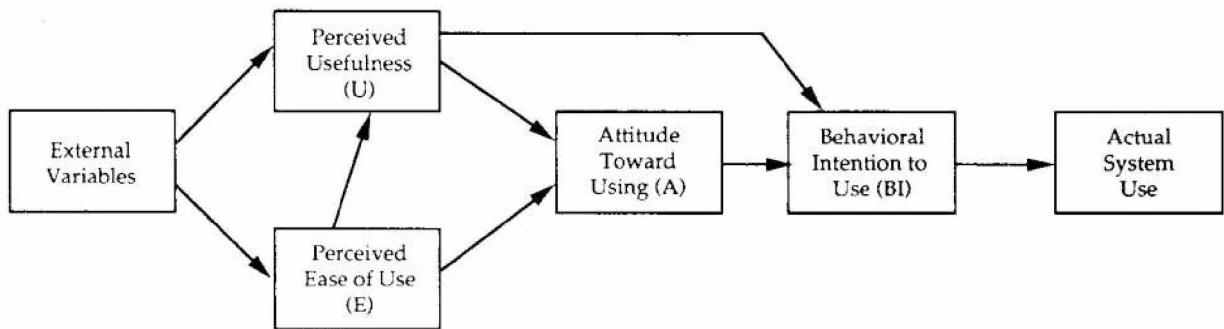


Figure 1 – Technology Acceptance Model (TAM)

TAM is one of the most popular research models to predict use and acceptance of information systems and technology by individual users (SURENDRAN, 2013). TAM has been widely studied and verified by different studies that examine the individual technology acceptance behavior in different information systems constructs.

Figure 1 presents TAM as a mediating role of two variables called perceived ease of use and perceived usefulness in a complex relationship between system characteristics (external variables) and potential system usage. Derived from the psychology-based theory of reasonable action (TRA) and theory of planned behavior (TPB), TAM has taken a leading role in explaining users' behavior toward technology.

Over the years the TAM methodology was modified and adapted by several researchers. One literature review (MARANGUNIĆ; GRANIĆ, 2015) was performed and its results indicated continuous progress in revealing new factors with significant influence on the core variables of the model.

1.3 Contributions

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 for developing and evaluating SOA applications.

Several gaps in SOA area were detached by Papazoglou et al. in (PAPAZOGLU et al., 2008). Since this paper (PAPAZOGLU et al., 2008) was published many of these gaps have not been properly addressed in the literature. Therefore, this research was proposed to address some SOA quality issues as follows.

First, this study describe a set of design principles for SOA proposed by several studies in literature. These design principles are important to improve design solution application and also increase quality application. The set of SOA design principles was collect from literature through a Systematic Literature Review. As for limitations of this work some

threats to validity can be described. The number of selected articles can be considered a threat to validity. Although most systematic literature reviews selected from 5% to 15% papers, and the one presented in this article selected approximately 6%. The final number of selected papers could be higher in future works. In addition to this, another threat to validity is related to paper selection. One risk in paper selection is to disregard an important paper from the set of retrieved papers. This threat to validity was minimized when two subjects were responsible for the retrieved paper analysis. This step certifies a double check in the paper selection.

Secondly, this work proposes a quality model specific to evaluate quality in SOA applications named SOAQM. SOAQM define a set of quality attributes to guide the development of applications. This set of attributes is based on ISO 25010 quality attributes and defines which are considered essential during SOA application development. SOAQM was proposed with the collaboration of seven SOA experts. Each expert provided their opinion about the degree of importance for SOA of ISO 25010 quality attributes. As for limitations of this proposal is related to experts amount that collaborate in SOAQM. To minimize this threat to validity, the SOAQM was evaluated by nine other experts. In this second moment of the research, the SOAQM was applied in the development of an application and nine specialists who evaluated the EHR development could provide comments on the SOAQM.

Another important contribution of this work is the approach used to develop architectural design decisions for a SOA application. Architectural decisions were proposed to consider quality in initial stages of software development.

Finally, this work contributed with development of an EHR system, which has been developed with the aim of integration and automatize business processes in a public hospital. Threat to validity related to EHR development is related to the approach used in development. SOAQM was applied as approach to guide architecture and design of EHR. This new approach proposed in research was evaluated by a set of experts through questionnaire and interviews.

1.4 Thesis outline

The thesis outline is organized as follows.

Chapter 2 provides theoretical background that is basis of this work. This chapter limits the scope in terms of which areas of Software Engineering are investigated. Therefore, Service-oriented Architecture and quality terms and concepts are addressed.

Chapter 3 presents a systematic literature review about design principles for SOA. As result of this review, a set of important design principles for SOA applications is presented.

Chapter 4 is about a quality model for SOA named SOAQM. SOAQM is based on

ISO 25010 and its proposal is to define quality attributes considered essential to SOA applications.

Chapter 5 addresses software architecture of an Electronic Health Record system. The EHR architecture is proposed by defining a set of architectural decisions.

Chapter 6 presents a multiple-view architecture for the EHR system. A set of five different views was proposed to representing architecture covering different concerns.

Chapter 7 presents the EHR design project. The software design is proposed using a modeling language specifically created for SOA solutions named SoaML. The SoaML diagrams are presented to improve the understanding of EHR scope and documenting EHR design.

Chapter 8 addresses implementation issues proposed during development of EHR system. The EHR implementation details are presented, as for instance, tools used during EHR development and implemented functionalities.

Chapter 9 provides experimental results of the research. This chapter reports the evaluation performed to assess development process used in EHR. This evaluation was performed by a set of IT experts. Another evaluation addresses tool usability. This evaluation was performed by health professionals representing EHR users.

Chapter 10 is about conclusions of this research including main contributions, future works and contributions in bibliography production.

Theoretical Background

This chapter presents theoretical background and related work of this research.

The content of this chapter is mainly based on the article entitled “Principles and Metrics to Improve Quality in SOA Applications” (FRANÇA; SOARES, 2014). This article was published on the proceedings of the International Conference on Enterprise Information Systems (ICEIS) in 2014.

2.1 Service Oriented Architecture

An important concept related to Service-oriented Computing (SOC) is the Service-Oriented Architecture (SOA). Service-oriented Architecture has been used to integrating systems, applications, processes and business. SOA reorganizes software applications and infrastructure in a set of services that interact (PAPAZOGLU, 2003). SOA has been widely used with the objective of integrating systems through services that may be reused by several systems.

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 (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 Enterprise Application Integration middleware (EAI) 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).

SOA provides some advantages in its adoption including the fact that service in SOA is platform independent (PAPAZOGLU, 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 (ERL, 2017).

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 to 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). SOA has emerged with the promise of allowing legacy systems to expose their core functionality as services (KHADKA et al., 2011).

2.1.1 SOA key concepts

Figure 2 presents a conceptual view of how the elements of service-oriented computing can inter-relate. Some important concepts related to service-oriented computing are defined in the next paragraphs.

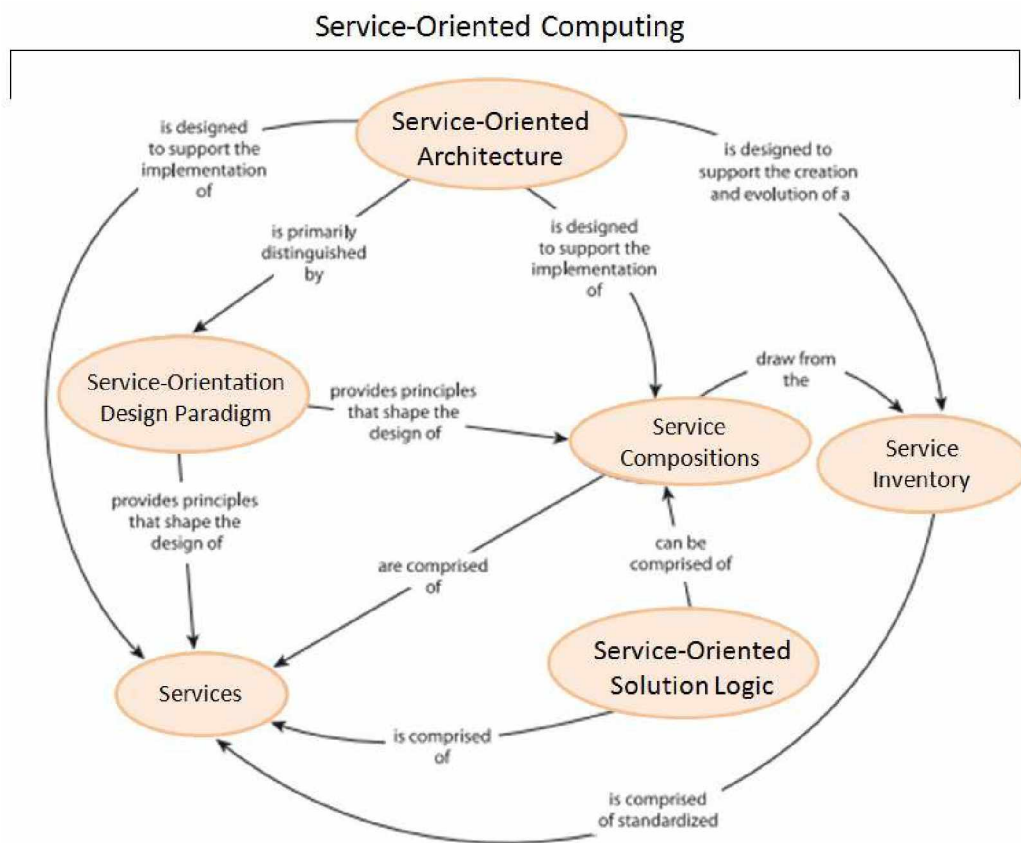


Figure 2 – A conceptual view representing SOA concepts and its relations (ERL, 2007)

Service-Oriented Computing (SOC) is the computing paradigm that utilizes services as fundamental elements for developing applications/solutions (PAPAZOGLU, 2003).

Service Oriented Architecture 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 (GROUP, 2014). SOA has also been defined as a *distributed architecture* with services distributed in an environment working together to perform some tasks (Steen, M.W.A. and Strating, O. and Lankhorst, T. and ter Doest, H.W.L., 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; LOCHMANN, 2011) (CARDOSO et al., 2015) (ERL, 2017).

Service is a capability of the business organization that is implemented and made available on the Internet so that other applications can access it (PAPAZOGLU, 2003). Services are autonomous, platform-independent computational entities that can be used to create large, complex applications. Services perform functions that can be simple requests for activities or complex business processes. Therefore, services can be of simple nature or composite.

Service composition involves the concept of orchestration. The concept of orchestration is related to automating business processes. A service composition is comprised of services that have been assembled to provide the functionality required to automate a specific business task or process (ERL, 2007).

Service Inventory is a collection of standardized services that can be independently administered within its own physical deployment environment.

Service-Oriented Solution Logic is implemented as services and service compositions (ERL, 2007).

Service Choreography 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 Web Services Choreography Description Language (WS-CDL) specification (W3C, 2016) enables the exchange of information among multiple organizations (or even different applications within the organization).

Open standards are used by SOC such as Simple Object Access Protocol (SOAP) (W3C, 2016) and Extensible Markup Language (XML). The SOAP protocol is used for data transmission. The XML is used to define the Web Services Description Language (WSDL) (W3C, 2016). WSDL describes 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.

Enterprise Service Bus (ESB) is an infrastructure software that is responsible for establishing the message flow bus between consumers and service providers. ESB will provide the management and mediation between systems that consume services and those who provide. The most popular method for exposing SOAP services across the enterprise is via a custom infrastructure known as an ESB (OGRINZ, 2009).

Figure 3 presents ESB capabilities for service consumers and providers to connect to each other, for services to be discovered using the registry, for services to be managed and for secure invocations, and provides Application Programming Interfaces (API) to facilitate the development of service connectivity.

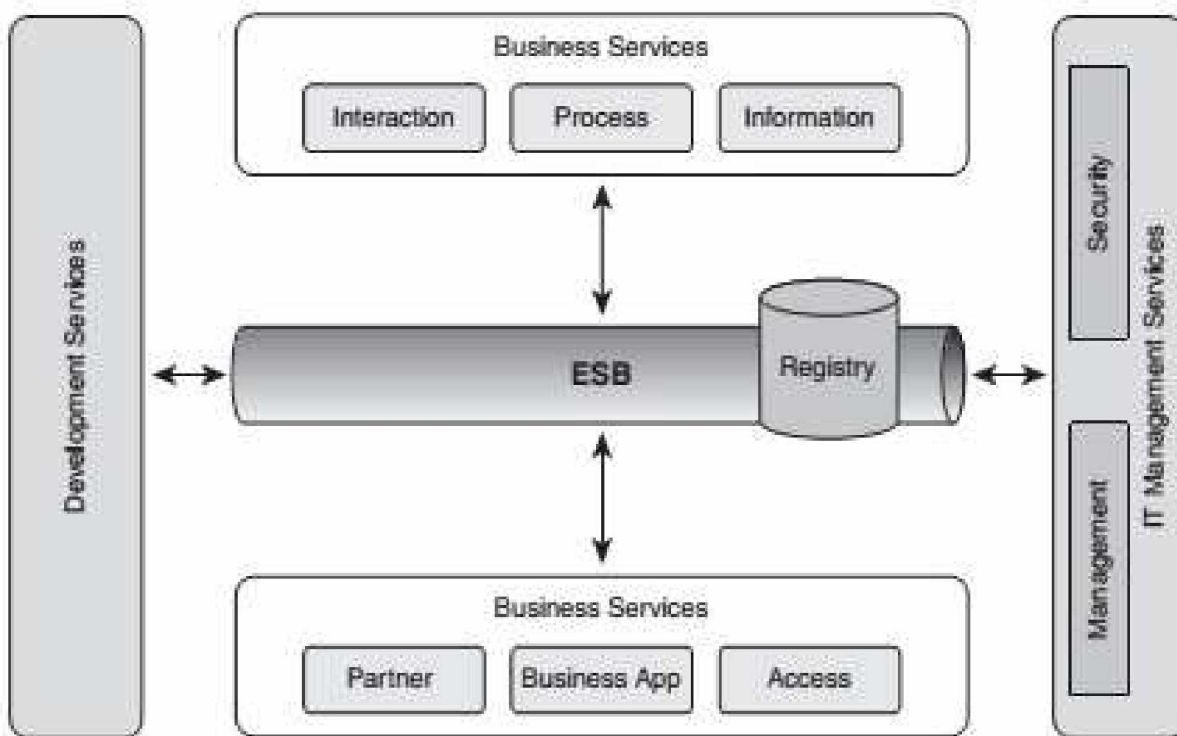


Figure 3 – Logical view of an ESB in the integration layer (BIEBERSTEIN et al., 2008)

ESB is not indispensable for SOA application development. It is possible to implement software based on SOA without using an ESB. However, adopting an ESB provides some advantages (PREVE, 2011) such as service mapping, message-processing, message transformation, process choreography, service orchestration, support for synchronous and asynchronous transport protocols, reliable delivery and transaction management.

2.1.2 SOA Reference Architecture

SOA reference architecture used in this research is the one proposed by The Open Group (GROUP, 2001), a solution often applied to various projects since 2002. The Open Group reference architecture provides an instrument to create or evaluate an architecture in terms of layers, components (building blocks), and roles to be considered in such a way to assure return of investment in technology and that the objectives are achieved.

Figure 4 depicts the architectural environment proposed by The Open Group SOA RA. The proposed architecture is composed of five horizontal layers (functional) and four vertical layers (non-functional).

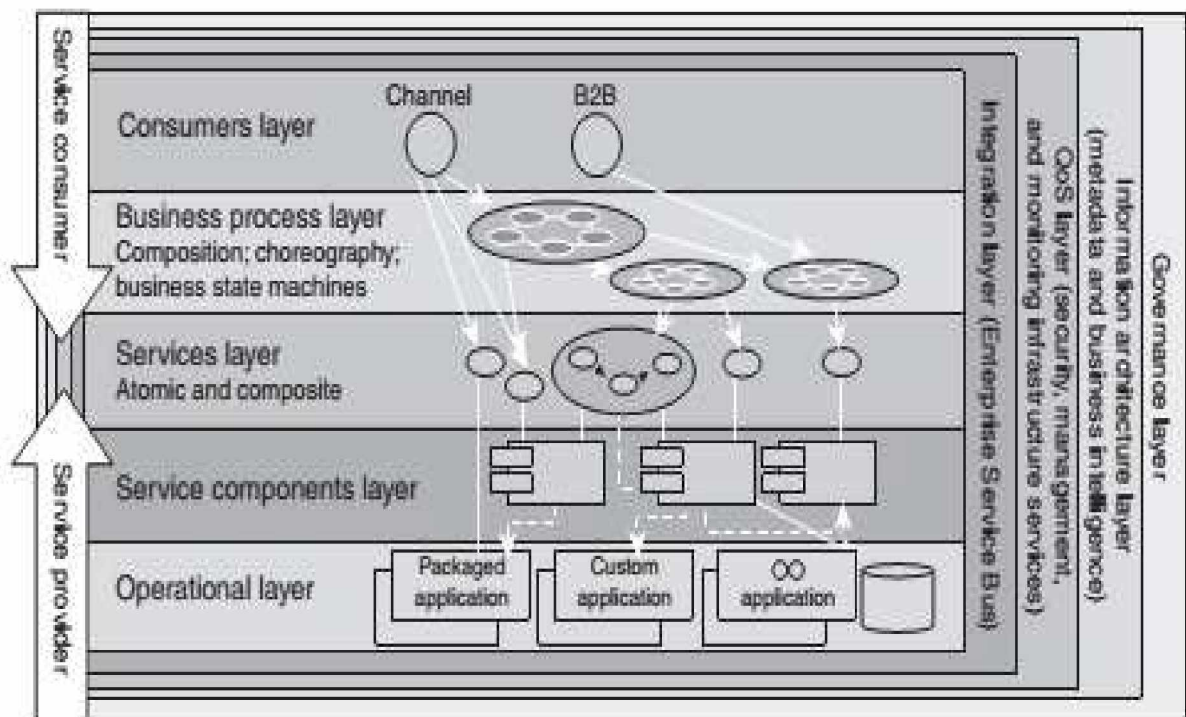


Figure 4 – Logical view of SOA Reference Architecture (BIEBERSTEIN et al., 2008)

Functional layers are briefly described as follows.

- ❑ **Operational layer** provide basic infrastructure to provide SOA functions. This layers promotes integration with legacy systems, data bases and allows services execution.
- ❑ **Service components layer** implements the services. These components connect the Service Contract with its implementation, and provide loose coupling between consumer and implementation.
- ❑ **Services layer** works as a Service container in which it is possible to find all Service Contracts.

- ❑ **Business Process layer** is responsible for composition and orchestration of services. It supports long process, and executes tasks according to policies, business rules, and constraints.
- ❑ **Consumer layer** deals with communication with users, giving support to different channels between users and applications. It also provides communication between applications and loosely coupling between consumer and implementation.

Non-Functional layers are briefly described as follows.

- ❑ **Integration layer** allows mediation, transformation, routing and transport. Services are exposed only through this layer, which centralizes business rules and provides loosely coupling between provider and consumer. This layer is generally supported by an ESB-Enterprise Service Bus.
- ❑ **Quality of Service (QoS)** is the layer that captures and monitors operational metrics, assuring reliability, availability, control, scalability and security.
- ❑ **Information architecture layer** includes data architecture, data structures (XML-Schemas) and data protocols.
- ❑ **Governance layer** defines SOA objectives and assures conformity between policies and processes. It also defines solution portfolio. Governance is applied to all layers.

A reference architecture in service-oriented solution context establishes the building blocks of SOA: services, service components, and flows that collectively support enterprise business processes and the business goals. This layering facilitates the creation of architectural blueprints in SOA and helps in reusability of solutions (BIEBERSTEIN et al., 2008).

2.2 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.

2.2.1 Software Design Principles

Examples of design principles for software engineering are found in SWEBOK (IEEE Computer Society, 2004): 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 is presented as follows based on various publications (BASS; CLEMENTS; KAZMAN, 1998) (BOSCH, 2000) (BUSCHMANN et al., 1996) (JALOTE, 1997) (LISKOV; GUTTAG, 2001) (PFLEEGER, 2001) (PRESSMAN, 2009).

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

Coupling and cohesion: 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.

2.2.2 SOA Design Principles

Service-oriented Architecture introduces specific concepts, characteristics and infrastructure. Therefore, some studies propose design principles adapted to SOA context. The study proposed in (HUHNS; SINGH, 2005) presents how the principle of abstraction is applied specifically in SOA. Another important publication 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; 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, services enable interoperability of systems belonging to the hospital.

According to Erl (ERL, 2007), a principle of design is defined as a widespread and accepted concept by software industry and academia. 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 statelessness, 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. Encapsulation performed 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, services have a contract that expresses a well-defined functional threshold which should not involve other services.

Service statelessness: services minimize dependence on the state. 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.

Article (XING; YAO, 2010) cites most of the design principles proposed by Erl, as well as additional principles, including Service Encapsulation, Service Optimization and Service Relevance. According to the authors, Service encapsulation means many services are consolidated for use under SOA. Often, such services were not planned to be under SOA. Service Optimization means high-quality services are generally preferable to low-quality ones. Finally, Service Relevance is a functionality presented at a granularity recognized by the user as a meaningful service. Nevertheless, the article does not mention who proposed these new principles.

Article (LIU et al., 2009) mentioned Service Component Modularity, Change Management and Integration as design principles. Article (XIAO-JUN, 2010) is another work presenting other types of design principles, such as Granularity. Articles (GRANELL; DÍAZ; GOULD, 2010) and (MONSIEUR; SNOECK; LEMAHIEU, 2012) also mentioned service granularity. Granularity is a design principle proposed by Leger and Vogel in 2007 (LEGNER; VOGEL, 2007).

Article (LIU, 2009) mentioned the design principle Service Oriented Usability. According to (ISO/IEC, 1991), usability is a nonfunctional software quality characteristic defined with sub-characteristics. Major sub-characteristics are: understandability, learnability, and operability. Service oriented usability is the grade given by a user of service oriented applications. Within this article, design principles related to usability of services are defined, as for instance, if diagrams are developed for service, if life cycle and lifetime of services are defined, and even if connections to access each service are defined.

Article (KANNAN; BHAMIDIPATY; NARENDRA, 2011) presented another design principle called Business Alignment, meaning the Service provides related set of functionality. For example, a Service related to student registration should not perform functionalities related to Professor.

2.3 Software Quality

The assessment of software quality is an extremely important activity in the software development process. Software industry is increasingly global with highly competitive products, and companies will have difficulties in succeeding on the market unless they produce quality products and services (SANDERS; CURRAN, 1994) (BANSIYA; DAVIS, 2002). The demand for quality software continues to intensify due to increasing dependence on software and the often devastating effect that a software error can have in terms of life, financial loss, or time delays.

According to (TIAN, 2005), software quality can have different meanings and therefore the question “what is software quality?” is bound to generate many different answers depending on stakeholders, circumstances and kind of software. According to this author, an alternative question that is probably easier is “what are the characteristics for high-quality software?”.

According to (KITCHENHAM; PFLEEGER, 1996) (PFLEEGER, 2002), quality is a complex and multifaceted concept that can be described from five different views: transcendental, user, manufacturing, product and value-based.

- ❑ In the *transcendental view*, quality is hard to define or describe in abstract terms, but can be recognized if it is present. It is generally associated with some intangible properties that delight users.
- ❑ In the *user view*, quality is fitness for purpose or meeting user’s needs.
- ❑ In the *manufacturing view*, quality means conformance to process standards.
- ❑ In the *product view*, the focus is on inherent characteristics in the product itself in the hope that controlling these internal quality indicators (or the so-called product-internal metrics) will result in improved external product behavior (quality in use).
- ❑ In the *value-based view*, quality is the customers’ willingness to pay for a software.

According to Sommerville (SOMMERVILLE, 2015), 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; MERSON; BASS, 2007): Interoperability, Performance, Security, Reliability, Availability, Modifiability, Testability, Usability, and Scalability.

2.4 SOA Quality Models

Software quality is a challenge issue in software engineering, and this challenge is extended to SOA applications. Quality models are elaborated to describe, to assess and to predict the quality of software (DEISSENBOECK et al., 2009). These models can be provided, for example, by organizations of international standards or guidelines (VOELZ; GOEB, 2010).

A number of quality models specific for SOA applications have been proposed in the literature: S-Cube (GEHLERT; METZGER, 2008), WSQM2.0 ¹ (OASIS, 2005), Quamoco-extended (GOEB; LOCHMANN, 2011), and quality model for evaluating reusability (CHOI; 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; 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; 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).

Quality Model for Web Services (WSQM) is a quality model for Web Services defined by Organization for the Advancement of Structured Information Standards (OASIS) (Organization for the Advancement of Structured Information Standards) ². 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; LOCHMANN, 2011), this model is not mature enough because many settings are not accurate.

The research project Quamoco ³ (KLÄS; LOCHMANN; HEINEMANN, 2011) (LOCHMANN; 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; 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

¹ OASIS Quality Model for Web Services (WSQM2.0)

² <https://www.oasis-open.org/>

³ <http://www.quamoco.de>

(QoS). One of these studies was performed by Choi and Kim (CHOI; HER; KIM, 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. 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.

In 2008, Choi and Kim published another study (CHOI; 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 metrics.

2.5 ISO 25010 – System and software quality models

Standards exist for many aspects of the software and systems development and evolution. The International Organization for Standardization International Organization for Standardization (ISO) is the main standards provider on the world scene (NANCE; ARTHUR, 2012) (ORIOLE; MARCO; FRANCH, 2014). The ISO 9000 family of standards applies to quality in a broad sense with ISO 9000:2000 providing vocabulary and foundational material and ISO 9001 defining the requirements for quality management. Most of remainder of the family are presented as guidelines. This is the case for ISO 10012 which deals with measurement.

Product quality model defined in ISO 25010 comprises eight quality characteristics: Functional Suitability, Reliability, Performance Efficiency, Usability, Security, Compatibility, Maintainability and Portability. Each one of these eight characteristics of ISO 25010 is composed by a set of co-related sub-characteristics which in total amount to 31 sub-characteristics as depicted in Figure 5.

A brief summary of the definition of each quality characteristic proposed by ISO 25010 is presented in this section as follows.

Functional Suitability represents the degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions.

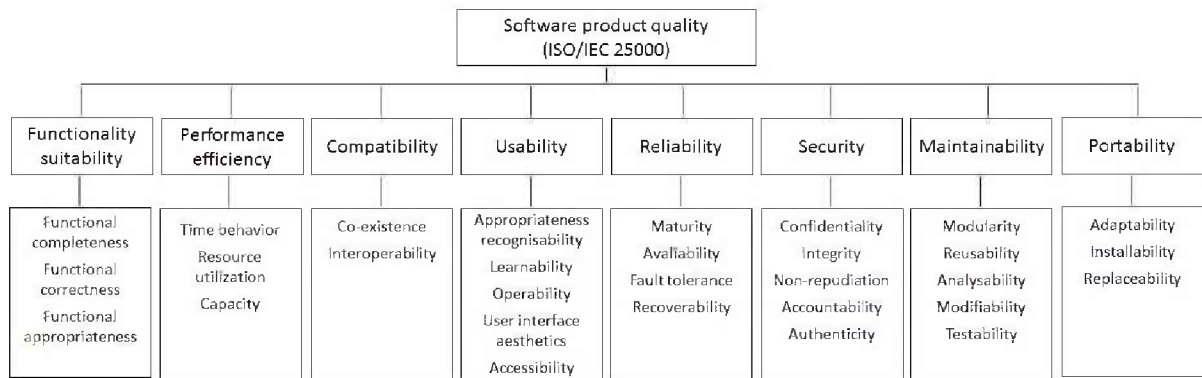


Figure 5 – Quality model for external and internal quality by ISO 25010

This characteristic is composed of sub-characteristics Functional Completeness, Functional Correctness, and Functional Appropriateness.

- ❑ Functional completeness: Degree to which the set of functions covers all the specified tasks and user objectives.
- ❑ Functional correctness: Degree to which a product or system provides correct results with the needed degree of precision.
- ❑ Functional appropriateness: Degree to which functions facilitate the accomplishment of specified tasks and objectives.

Performance efficiency represents the performance relative to the amount of resources used under stated conditions. This characteristic is composed of sub-characteristics Time Behaviour, Resource Utilization, and Capacity.

- ❑ Time behaviour: Degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements.
- ❑ Resource utilization: Degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements.
- ❑ Capacity: Degree to which the maximum limits of a product or system parameter meet requirements.

Compatibility is the degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment. This characteristic is composed of sub-characteristics Co-existence and Interoperability.

- ❑ Co-existence: Degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product.

- ❑ Interoperability: Degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.

Usability is the degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. This characteristic is composed of sub-characteristics Appropriateness Recognizability, Learnability, Operability, User Error Protection, User Interface Aesthetics, and Accessibility.

- ❑ Appropriateness recognizability: Degree to which users can recognize whether a product or system is appropriate for their needs.
- ❑ Learnability: degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use.
- ❑ Operability: Degree to which a product or system has attributes that make it easy to operate and control.
- ❑ User error protection: Degree to which a system protects users against making errors.
- ❑ User interface aesthetics: Degree to which a user interface enables pleasing and satisfying interaction for the user.
- ❑ Accessibility: Degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.

Reliability is the degree to which a system, product or component performs specified functions under specified conditions for a specified period of time. This characteristic is composed of sub-characteristics Maturity, Availability, Fault Tolerance, and Recoverability.

- ❑ Maturity: Degree to which a system, product or component meets needs for reliability under normal operation.
- ❑ Availability: Degree to which a system, product or component is operational and accessible when required for use.
- ❑ Fault tolerance: Degree to which a system, product or component operates as intended despite the presence of hardware or software faults.

- ❑ Recoverability: Degree to which, in the event of an interruption or a failure, a product or system can recover data directly affected and re-establish the desired state of the system.

Security is the degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization. This characteristic is composed of sub-characteristics Confidentiality, Integrity, Non-repudiation, Accountability, and Authenticity.

- ❑ Confidentiality: Degree to which a product or system ensures that data are accessible only to those authorized to have access.
- ❑ Integrity: Degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data.
- ❑ Non-repudiation: degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later.
- ❑ Accountability: Degree to which the actions of an entity can be traced uniquely to the entity.
- ❑ Authenticity: Degree to which the identity of a subject or resource can be proved to be the one claimed.

Maintainability represents the degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in environment and in requirements. This characteristic is composed of sub-characteristics Modularity, Reusability, Analyzability, Modifiability, and Testability.

- ❑ Modularity: Degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.
- ❑ Reusability: Degree to which an asset can be used in more than one system, or in building other assets.
- ❑ Analyzability: Degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified.
- ❑ Modifiability: Degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.

- Testability: Degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.

Portability is the degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another. This characteristic is composed of sub-characteristics Adaptability, Installability, and Replaceability.

- Adaptability: Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments.
- Installability: Degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment.
- Replaceability: Degree to which a product can replace another specified software product for the same purpose in the same environment.

2.6 Software Architecture

ISO 42010 (ISO/IEC, International Organization for Standardization, 2011b) is the International Standard that specifies the manner in which architecture descriptions of systems are organized and expressed. This International Standard specifies architecture viewpoints, architecture frameworks and architecture description languages for use in architecture descriptions.

When system complexity grows, it is commonly advised to develop applications having a well-defined software architecture as basis (GARLAN et al., 2010) (BERTOLINO; INVERARDI; MUCCINI, 2013). Architecture-based development has been applied for many years to solve issues such as interoperability between legacy systems and to help the development, integration and evolution of complex software systems (GARLAN, 2014).

Software engineering literature suggests to describe in early stages of software development process the overall software architecture (SOMMERVILLE, 2015). A well-known principle in system development is to describe in early stages of software development process the overall software architecture (BOOCH, 2007). According to ISO 42010 (ISO/IEC, International Organization for Standardization, 2011b), a software architecture is a fundamental concept of a software system in its environment embodied in its elements, relationships, and in the principles of its design and evolution. As a matter of fact, software architecture supports future activities, including software development, besides having a close relationship with requirements asked by stakeholders.

2.6.1 Architectural Decisions

According to (JANSEN; BOSCH, 2005), a software architecture is the result of a set of architectural design decisions (ADD). ADD can be defined as a description of the set of architectural additions and modifications to the software architecture, the rationale, and the design rules and constraints that realize one or more requirements on a given architecture.

Documenting ADD is important to reduce the knowledge vaporization of design decision (JANSEN; BOSCH, 2005) and therefore maintenance costs of software systems (BOSCH, 2004). According to (BABAR et al., 2009), representing architectural knowledge explicitly through ADD is crucial to gain intellectual control over a system's enormous complexity, ensure the continuity, allowing these large systems to more effectively evolve, facilitate overall maintenance and knowledge transference to other stakeholders.

Architectural Design Decisions are descriptions documented during software development that can be defined to improve software quality. According to a systematic mapping (TOFAN et al., 2014), ADD are documented for many reasons, including for achieving specific quality attributes. According to Tyree and Akerman (TYREE; AKERMAN, 2005), in order to analyze decisions' architectural significance it is necessary to identify if the decisions affect one or more system qualities.

A systematic mapping (TOFAN et al., 2014) found 144 relevant papers to present an overview of the state of research on architectural decisions. The conclusion of study (TOFAN et al., 2014) is that literature has demonstrated an increasing interest in ADD topics. However, only 9 out of 144 studies proposed ADD for SOA applications. In general, these studies had focus on templates for documenting SOA decisions. Most of these studies addressing ADD for SOA did not have focus on meeting quality attributes. Only 7 out of 144 papers explicitly treat quality attributes, although these 7 studies do not have SOA as context.

2.6.2 Architectural Views

An Architecture View (or simply, view) refers to a work product expressing the architecture of a software system from the perspective of specific system concerns. Typically, a software architecture description should include a number of architecture views (ISO/IEC, International Organization for Standardization, 2011b). An architecture view addresses one or more of the concerns held by one or a group of stakeholders.

The multiple views architecture is important to describe the different concerns of the architecture. Separating concerns in different views helps in diminishing the misunderstandings and incorrect interpretations (KRUCHTEN, 1995).

An example of model for describing software architecture is the 4+1 View Model of Architecture (KRUCHTEN, 1995). The 4+1 consists of four views (Logical, Process,

Development, and Physical) and the plus one (Scenarios) crosscutting view that integrates the other four, as depicted in Fig. 6. The Logical view is concerned with the functional requirements of the software. The process view deals with the dynamic aspects of the system, explains the system processes and how they communicate. The physical view is concerned with the topology of software components and hardware. The development view describes the static organization of the software in its development environment. The scenarios view describes sequences of interactions between objects and processes illustrated by a set of selected use cases.

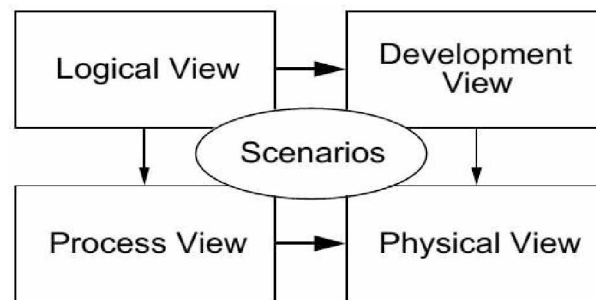


Figure 6 – The “4+1” view model

2.7 Unified Modeling Language - UML

Unified Modeling Language (UML) was introduced in 1994 from the unification of three object-oriented design methods: the Booch Method (BOOCH, 1994), the Object Modelling Technique (OMT) (RUMBAUGH et al., 1991), and the Objectory Method (JACOBSON, 1992). The UML standard was set and is managed by the Object Management Group (OMG).

Currently, the new version of UML is 2.5 and the objective of UML (OMG, 2015) is to provide system architects, software engineers, and software developers with tools for analysis, design, and implementation of software-based systems as well as for modeling business and similar processes.

UML diagrams can be classified as those that model the static aspects of the system that are structural diagrams (Fig. 7) and those that model the dynamic aspects of the system that are behavioral diagrams (Fig. 8).

According to (PETRE, 2013), class, activity and sequence diagrams are the most frequently used UML diagrams. In (PETRE, 2013), more than 50 practicing professional software developers were interviewed and these informants reported to use UML diagrams as follow: Class diagrams are used to structure, conceptual models, concept analysis of domain, architecture, interfaces; Activity diagrams are used to modeling concurrency, eliciting useful behaviors, ordering processes; Sequence diagrams are used to requirements

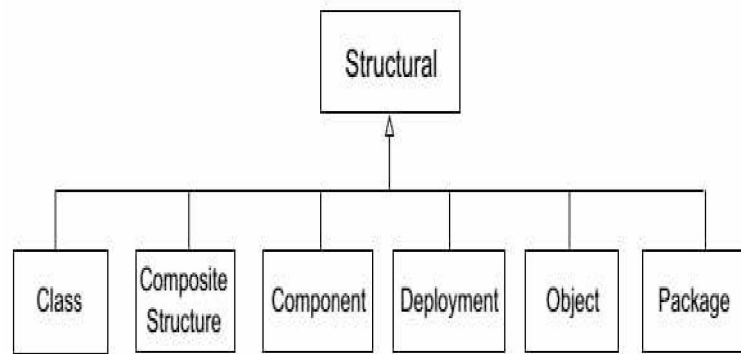


Figure 7 – UML Structural Diagrams

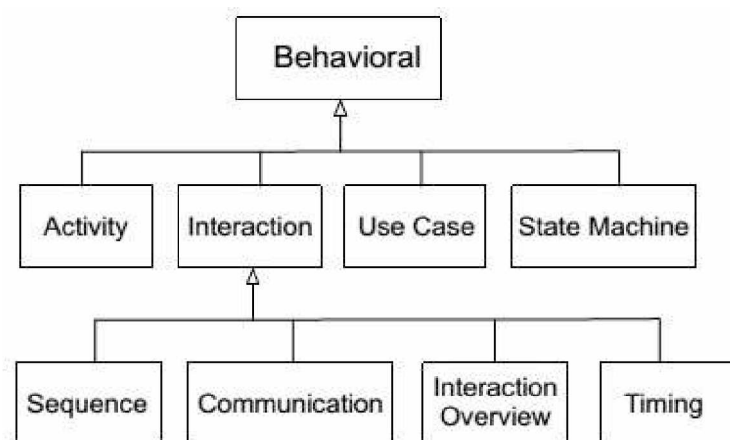


Figure 8 – UML Behavioral Diagrams

elicitation, eliciting behaviors, instantiation history; and Use case diagrams are used to represent requirements.

Definition and concepts about UML diagrams are presented as follows (OMG, 2015):

Class Diagram

A class diagram in UML is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among objects.

An example of a class in UML diagram is presented in Fig. 9 that shows three ways of displaying the Class Window, according to the options set out for Classifier notation. The top left symbol shows all compartments suppressed. The lower left symbol shows the attributes and operations compartments, each listing the features but suppressing details such as default values, parameters, and visibility markings. The right symbol shows these details, as well as the optional compartment headers.

Two examples of class diagram representing relationships between classes are presented in Fig. 10 and 11. Fig. 10 presents the black diamond notation for composite aggregation. In composition relationship, the class with a filled diamond shape con-

tains the classes that are connected on the end of line. Fig. 11 shows class diagram example of association between two classes.

Another two types of relationships between classes are possible that are generalization and aggregation. Generalization indicates that one class is considered to be a specialized form of the other. The superclass is considered a generalization of the subclass. Aggregation is an association that represents a part-whole or part-of relationship. Aggregation is graphically represented as a hollow diamond shape on the containing class with a single line that connects it to the contained class.

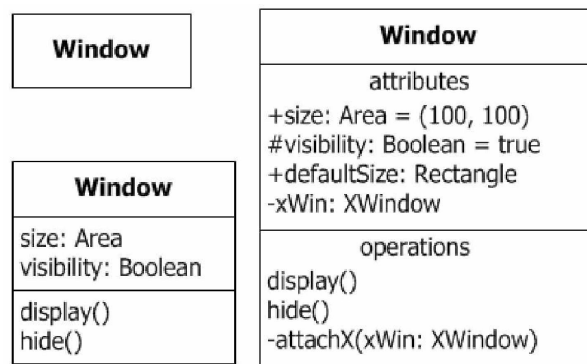


Figure 9 – Class diagram - class representation (OMG, 2015)

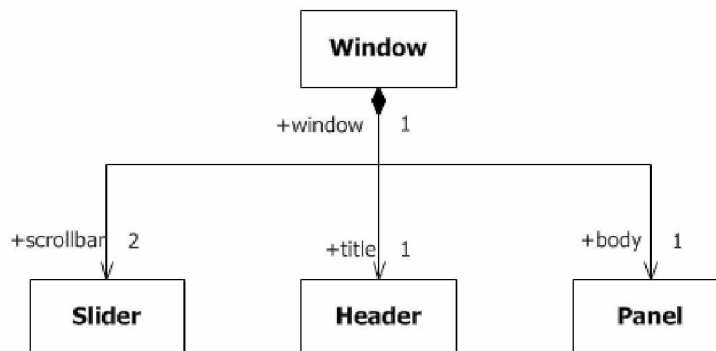


Figure 10 – Class diagram - Composite aggregation (OMG, 2015)

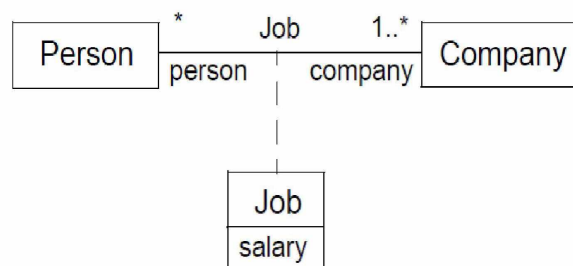


Figure 11 – Class diagram - association (OMG, 2015)

Activity Diagram

Activity diagram can be used to describe the dynamic aspects of the system. Activity diagram uses a flowchart to represent the flow from one activity to another activity. The activity can be described as node and represents an operation of the system. The flow can be sequential, branched, or concurrent. Activity diagrams deal with all types of flow control by using different elements such as fork and join. Fig. 12 illustrates an example of Activity diagram where rounded rectangles represent actions; diamonds represent decisions; bars represent the start (split) or end (join) of concurrent activities; a black circle represents the start (initial node) of the workflow; and an encircled black circle represents the end (final node).

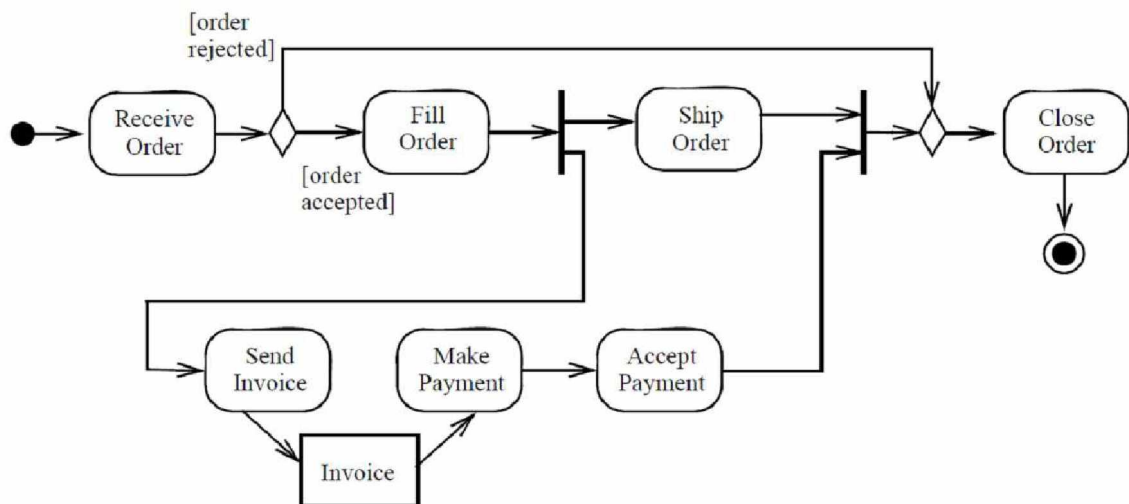


Figure 12 – Activity diagram (OMG, 2015)

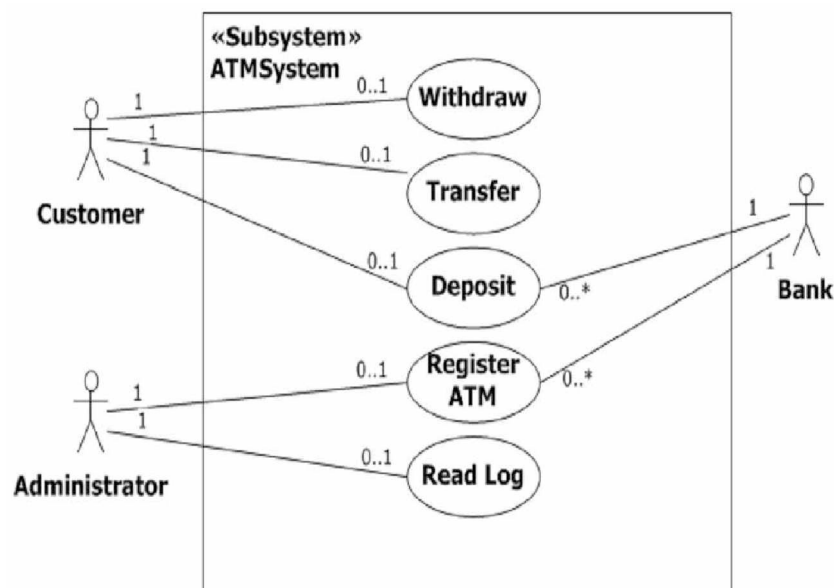


Figure 13 – Use Case diagram (OMG, 2015)

Use Case Diagram

Use case diagrams are referred to behavior diagrams and represents the requirements of a system including internal and external influences. Use case diagrams describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors). Fig. 13 illustrates an Use Case Diagram example which presents the functionalities of an “ATMsystem” subsystem and the actors authorized to interact with these functionalities.

UML was created mainly for general software development. Customizations for more specific purposes were created, such as SoaML (OMG, 2012) for SOA systems.

2.8 SOA Modeling Language - SoaML

One of many challenges in SOA research detached by Papazoglou et. al in (PAPAZOGLU et al., 2008) is the lack of specific design models developed to describe SOA concepts and elements. Authors affirm that general purpose diagrams are not appropriated to design SOA applications.

Service oriented architecture Modeling Language (SoaML) (OMG, 2012) is a UML profile for modeling and designing services within a Service-Oriented Architecture. In the SOA development process, it is important to specify services through multiple views such as service definition, and also to describe relationship between services providers and consumers.

The SoaML specification introduces the concept of services architecture to model how a group of participants that interact through services provided and used to accomplish a result. Some key concepts introduced by SoaML are briefly presented as follows.

Participants: participants are either specific entities or kinds of entities that provide or use services. Participants can represent people, organizations, or information system components.

Ports: participants provide or consume services via ports. A port is the part or feature of a participant that is the interaction point for a service - where it is provided or consumed. A port where a service is offered may be designated as a “Service” port and the port where a service is consumed may be designated as a “Request” port.

Service description: the description of how the participant interacts to provide or use a service is encapsulated in a specification for the service. The service descriptions result in interfaces and behaviors that define how the participant will provide or use a service through ports. The service descriptions define how the provider provides the service or how (or why) the consumer consumes it. A service specification

specifies how consumers and providers are expected to interact through their ports to enact a service, but not how they do it.

Capabilities: participants that provide a service must have a capability to provide it.

Capabilities can be seen from two perspectives, capabilities that a participant has that can be exploited to provide services, and capabilities that an enterprise needs that can be used to identify candidate services.

SoaML proposed UML extensions by creating stereotypes and new diagrams: participant diagram, service diagram, service interface diagram, capability diagram and service architecture diagram. SoaML diagrams and some examples of their use are introduced as follows.

❑ **Service Contract Diagram:** Documents service specification, the terms, conditions and roles that each participant (of service) must agree to in order to enact a service. UML Collaboration stereotype **«ServiceContract»** is used in this diagram.

One example of the use of Service Contract Diagram is presented in Fig. 14. *Place Order Service* is the service contract itself that define the terms and conditions under which the service can be enacted and the results of the service. The service contract can also be used in services architectures to show how multiple services and participants work together for a business purpose. The *provider : Order Taker* defines the role of the provider in the place order service. The provider is the participant that provides something of value to the consumer. The *consumer: Order Placer* is the role of the consumer in the place order service. The consumer is the participant that has some need and requests a service of a provider.

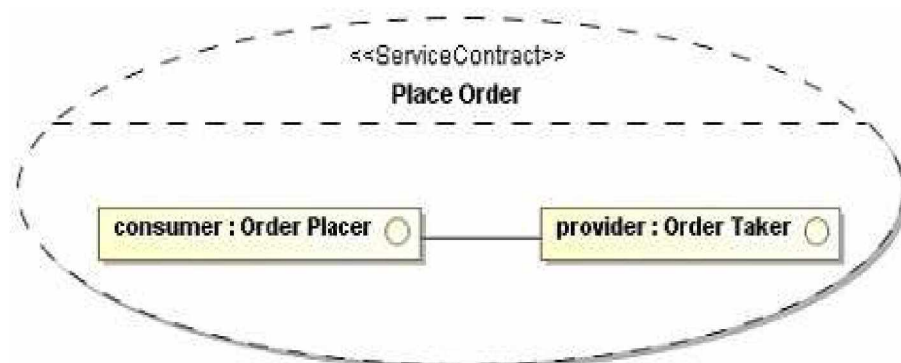


Figure 14 – Place Order Service - Service Contract Diagram (OMG, 2012)

❑ **Service Interface Diagram:** Defines the graphical representation of provided and required interfaces used by a service. This diagram uses **«ServiceInterface»** as Class and Interface stereotype.

Figure 15 presents one example of Service Interface diagram following the same scenario of *Place Order Service*. The $\ll Provider \gg$ Order Taker is the type of a place order service provider indicating all the operations and signals a providing participant may receive when enacting this service. The $\ll Consumer \gg$ Order Placer is the type of a place order service consumer. This indicates all of the operations and signals a consuming participant will receive when enacting the service with the provider.

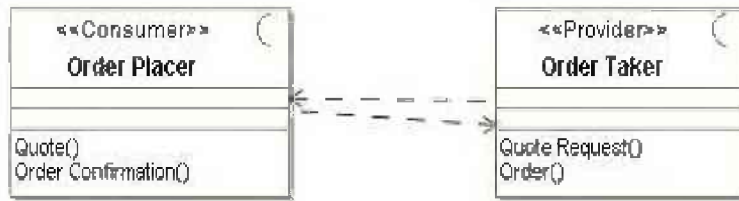


Figure 15 – Place Order Service - Service Interface Diagram (OMG, 2012)

- ❑ **Participant Diagram:** Models the person, organization, system or anyone who takes part in a service architecture. This diagram represents these participants as well as interfaces required or provided in accomplishing services. The UML Class stereotype used is $\ll Participant \gg$.

In *Place Order Service* scenario, Dealer and Manufacturer are examples of participant as presented in Fig. 16. Both the dealer and manufacturer have a “Place order service” port, each typed by one of the interfaces in the place order service. Dealer has a “Place order service” port, but this time typed by the interface representing their role in the service (“Order Placer”). The dealer’s port then provides the order placer interface and requires the order taker. These ports have provided and required interfaces that are conjugated and are therefore compatible and these ports can be connected to enact the service.



Figure 16 – Place Order Service - Participant Diagram (OMG, 2012)

- ❑ **Service Architecture Diagram:** High level description of how participants work together for a purpose by providing and using services expressed as service contracts.

The UML Collaboration stereotype **<< ServicesArchitecture >>** is used in this diagram.

Figure 17 presents Service Architecture diagram in *Place Order Service* scenario. Services Architecture diagram shows how a set of participants work together, providing and using service. In Fig. 17, services architecture shows how the dealer, manufacturer, and shipper work together using the place order service, the shipping request service, and the ship status service. The participants defined are compatible with this architecture since they have ports that provide and use the corresponding interfaces.

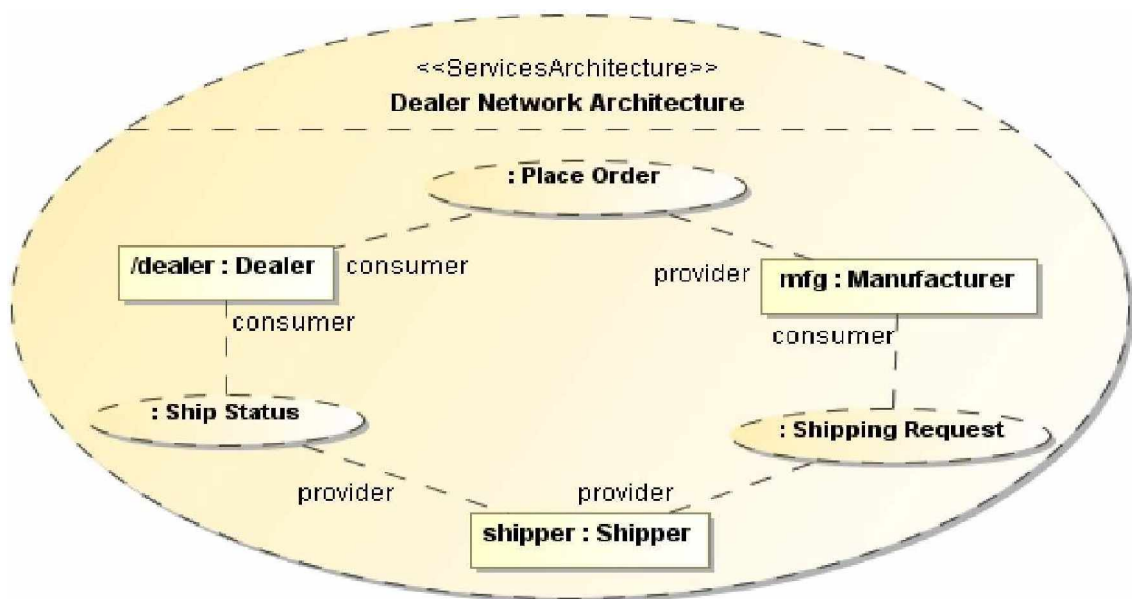


Figure 17 – Place Order Service - Service architecture Diagram (OMG, 2012)

- ❑ **Capabilities diagram:** allows the definition of capabilities required that will be realized by the defined services and participants. This diagram uses **<< Capability >>** as Class stereotype.

Figure 18 shows a network of Capabilities that might be required to process an order for goods from a manufacturer. Such networks of capabilities are used to identify needed services, and to organize them into catalogs in order to communicate the needs and capabilities of a service area, whether that is business or technology focused, prior to allocating those services to particular Participants.

One important advantage of using SoaML as modeling language to SOA application is that it is a UML extension. Several modeling tools support UML design and it is very common that these tools provide extensions. It is easy to find a tool that can provide extensions to SoaML too. There are some tools for system modeling that supports

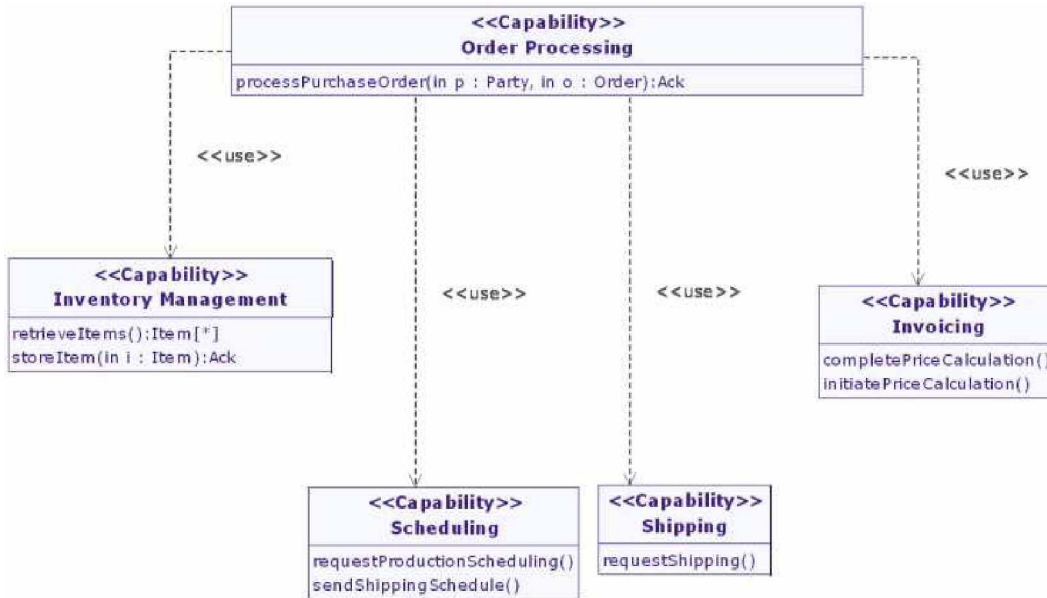


Figure 18 – Place Order Service - Capability Diagram (OMG, 2012)

SoaML, such as Visual Paradigm⁴ and MagicDraw⁵. However, these modeling tools are commercial ones. One free option is Modelio⁶, used in this project. Modelio is an open source modeling tool which can be extended through modules to add functionalities and services. In addition to providing a wide range of models and diagrams, including UML, Modelio validates the created models and supports code generation. The Modelio SoaML designer implements a dedicated GUI including six new diagrams: Capabilities, Service Contract, Service Architecture, Message, Service Interfaces, and Participant diagrams.

⁴ <http://www.visual-paradigm.com/features/soaml-modeling/>

⁵ <http://www.nomagic.com/products/magicdraw.html>

⁶ <https://www.modelio.org/>

Review of Design Principles for SOA

The content of this chapter is mainly based on the article entitled “Characterization of the Application of Service-Oriented Design Principles in Practice: A Systematic Literature Review” (SOARES; FRANÇA, 2016). This article was accepted on Journal of Software (JSW) in 2015. The article was published on JSW in April 2016, Vol. 11, No. 4.

This chapter presents a Systematic Literature Review (SLR). This chapter presents design principles specifically proposed for SOA applications. One of the objectives of this SLR is to analyze to which domains SOA has been applied in studies. In addition, this SRL aims to find out how design principles are proposed to cover particularity of SOA systems. Given these directions, it is important to identify if new design principles were proposed or if general purpose principles were adapted for SOA. General purpose principles such as abstraction and loose coupling were proposed in the past decades to be used in software engineering and it is interesting to know how they can be adapted to be applied in SOA systems. The systematic review presented in this chapter aims to describe how design principles proposed by software engineering can be applied for SOA and how SOA design principles are used in practice. Another analyzed aspect in SLR is related to what is the influence of these design principles in SOA application quality.

3.1 Protocol

Steps for the systematic review include definition of research questions, definition of venues, definition of the search string and testing the search string.

3.1.1 Research Questions

Research questions addressed in this SLR are described as follows. With the objective of differentiating the research questions from the thesis (RQ) with the research questions defined for the SLR, the research questions of the SLR will be classified as SLR-RQ. The

reason for each research question (SLR-RQ) and what is expected from each one is briefly explained after each question.

SLR-RQ1 *What are the purposes of using SOA?*

SOA has been frequently associated with legacy systems (KHADKA et al., 2011) (ERL, 2017). The objective of this question is to know most common activities related to using SOA in practice. One can expect that services are used to integrate legacy systems for developing new complex distributed applications. However, it is important to know if there are further purposes for using SOA besides integration of systems.

SLR-RQ2 *What are the most used design principles for SOA applications?*

From the list of design principles for SOA, it is interesting to catalog which ones are most used for developing SOA applications, and also why these are most used. On the other hand, if a design principle is not used at all, then it is of utmost importance to know why.

SLR-RQ3 *What are the benefits of applying SOA design principles? Are these benefits measured?*

This question aims to present advantages of SOA design principles, but also understand the expected final quality of SOA applications. This is most useful if one can find metrics to support results, as has been common with other technologies, methods and processes in Software Engineering (SMITE et al., 2010) (HARMAN; MANSOURI; ZHANG, 2012).

SLR-RQ4 *How are SOA applications evaluated in practice?*

Historically, evaluation is an issue in Software Engineering research (SHAW, 2002) (GLASS; RAMESH; VESSEY, 2004) (RUNESON et al., 2012). SRL-RQ4 aims to identify if this is true for SOA applications as well, i.e., if applications developed with SOA paradigm are evaluated in practice, and what are the employed methods for evaluation.

SLR-RQ5 *How are SOA applications modeled?*

This question was proposed to know how modeling of SOA applications is performed, and which modeling languages and specific diagrams are used for modeling SOA applications.

SLR-RQ6 *For which domains are SOA applications developed?*

The purpose with this question is to map which domains are most commonly using SOA, and if SOA has been applied only to specific domains. Therefore, it is inter-

esting to know if SOA applications are restricted to specific domains, or are widely applied in industry and academia.

3.1.2 Search Process

Search was performed in journals and conferences from 2008 to 2014. This choice of dates interval was decided based on two reasons. First reason is because Design principles for engineering service applications is a research challenge presented in 2008 (PAPAZOGLU et al., 2008). Finding out if this gap has been adequately addressed by researchers in the last seven years is an open question. Another reason is because publications to be searched must be up to date, and more importantly, design principles for SOA were systematized by Erl in his 2007's book (ERL, 2007) (although it was also looked at other design principles, published recently in other articles).

Search strategies in SLR often involve automatic keyword searches in digital libraries. Venues searched included ACM, ScienceDirect, Springer, and IEEE Xplore Digital Library. The defined search string was ((“service oriented architecture” or “service oriented computing”)) and (“design principles”). The total number of retrieved articles was 507.

According to (BRERETON et al., 2007), software engineering digital libraries do not provide good support for identification of relevant research and selection of primary studies. Therefore, a manual selection of articles in relevant venues was also performed. A search was performed in IEEE TSE (IEEE Transactions on Software Engineering) and ACM TOSEM (ACM Transactions on Software Engineering and Methodology).

3.1.3 Inclusion and Exclusion Criteria

The selection process used in this SLR is summarized in Figure 19. First activity when conducting a SLR is searching papers in journal and conference proceedings. This search formulates a list of retrieved papers that need to be filtered to select only those relevant to the SLR.

A criterion to effectively consider an article for evaluation in this systematic review was divided into two steps. First step consists in analyzing all retrieved papers and select important articles for review. This step is extremely important and it was performed by two subjects that are authors of this SLR (SOARES; FRANÇA, 2016). Second step finalizes selection process and articles was fully read to compose the final groups of articles selected for review. These two steps are detailed as follows .

3.1.3.1 Paper Selection - First step

Automatic and manual searches retrieved a set of 507 articles. First step consisted of analyzing all 507 articles in order to find those that would be the most important ones for this systematic review. In this first step, for each article, title, keywords, and

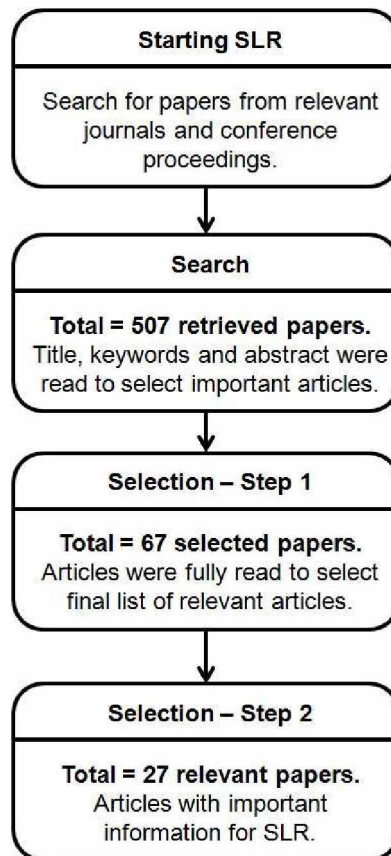


Figure 19 – Paper Selection Process

abstract were read and catalogued. In this step, duplicated papers were excluded. All 507 articles were analyzed and selected according to their relevance. At the end of this step, all articles considered relevant to the review were chosen.

This article selection consist in evaluating if each article meets inclusion criteria. Inclusion criteria is related to the focus of review driven by research questions defined for this SLR. Therefore, to be selected the article needs to addresses topics based in SOA context.

This first step is a crucial part of the process which was taken carefully. Thus, the choice was to perform this part independently by two subjects. Each subject manually classified each paper, in terms of selection for further steps, as “Definitely”, “Possibly”, or “Not Selected”. All 507 articles were classified according to their relevance to review as follow detailed.

- “Definitely”

If one article was classified as “Definitely”, it means that after reading the title, keywords and abstract the article was considered very relevant to the review and should proceed to the next analysis.

- “Possibly”

After reading the title, keywords and abstract of the article but not being absolutely

sure of the contribution of the same to the review then the article is classified as “Possibly”.

- “Not Selected”

Articles classified as “Not Selected” were considered not relevant because do not address topics in SOA context.

Each subject provided one list of articles classified as “Definitely”, “Possibly” or “Not Selected”. These two lists were merged into a final list of articles. For the final list of selected articles, the following rules were applied:

- ❑ **R1:** Article is selected if it has been classified with at least one “Definitely”.
- ❑ **R2:** Article is selected if it has been classified with two “Possibly”.
- ❑ **R3:** Articles classified with two “Not Selected” were obviously not chosen.
- ❑ **R4:** If an article was classified as one “Possibly” and one “Not Selected”, then title, abstract and keywords were read again, as well as the conclusion. Then, the article was classified again, and if classification remained the same, it was discarded. Otherwise, one of the first three rules was applied.

These rules were applied by considering the selection process has some qualitative analysis, and it is hard to make it completely deterministic.

After this first step of classification, 67 articles were selected. This number of selected articles can be considered low (about 13%). However, according to article (KITCHENHAM et al., 2009), which presents a Systematic Literature Review of Systematic Literature Reviews in Software Engineering, the percentage of selected articles normally varies from a minimum of 1,30% to a maximum of about 67%. Most works selected from 5% to 15% papers. Therefore, the final number of selected papers makes the research relevant.

Another reason why the number of selected articles was low is because the word “service” has many meanings, and is a common word even outside the service-oriented domain. However, even when the domain is the right one, most articles were not meaningful for this systematic review because they consider other aspects such as business or infrastructure.

3.1.3.2 Selection - Second step

For the second step, all 67 articles were fully read, with the purposes of comprehending main subject and trying to find answers to defined research questions. Articles were considered relevant if they answered at least one of the research questions. In summary, relevant articles considered at least one of the following aspects:

- ❑ Article proposes a real application in practice, an experiment, a case study, or industrial cases.

- ❑ Article used some kind of modeling for SOA.
- ❑ Article mentioned design principles for SOA.
- ❑ Article mentioned keywords such as modularity, architecture, coupling, cohesion or granularity.
- ❑ Article evaluated influence of a design principle in the final quality of the product.

The overview of the selection process and exclusion is depicted in Fig. 19. Total number of relevant articles, as described in Table 3, is 27 (about 5.5% of initial total number of articles).

Table 2 – Overview of Search Results and selected papers

Venues	Retrieved Papers	Selected Papers	Relevant Papers
ACM	118	13	7
Science direct	178	24	7
IEEE	63	17	9
Springer	103	6	3
IEEE-TSE	39	4	1
ACM-TOSEM	6	3	0
Total	507	67	27

Table 3 – Relevant papers

Venue	Publication	Venue/Year
ACM	(LANMAN et al., 2013)	SPLC/2012
	(CELLARY; STRYKOWSKI, 2009)	ICEGOV/2010
	(GARCÍA-SÁNCHEZ et al., 2013)	GECCO/2013
	(PAUTASSO; WILDE, 2009)	WWW/2009
	(DIRGAHAYU; QUARTEL; SINDEREN, 2008)	SAC/2008
	(KHOSHKBARFOROUSHHA; JAMSHIDI; SHAMS, 2010)	WETSoM/2010
	(ARDITO et al., 2014)	JVLC/2014
Science direct	(MONSIEUR; SNOECK; LEMAHIEU, 2012)	JSS/2012
	(GRANELL; DÍAZ; GOULD, 2010)	Env. Model. and Soft./2010
	(CHO et al., 2010)	Int. Jour. of Medical Inf./2010
	(TIAN; HUANG, 2012)	Expert Sys. App./2012
	(YAMANY; CAPRETZ; ALLISON, 2010)	IST/2010
	(WU et al., 2009)	Future Gener. Comp. Sys./2009
	(KABIR; HAN; COLMAN, 2014)	PMC/2014
IEEE	(XING; YAO, 2010)	ICSPS/2010
	(XIAO-JUN, 2010)	ICIECS/2009
	(KING et al., 2010)	ISPA/2010
	(VESCOUKIS; DULAMIS, 2011)	VS-GAMES/2011
	(LIU, 2009)	SOCA/2009
	(KANNAN; BHAMIDIPATY; NARENDRA, 2011)	SRII Global Conference/2011
	(KANNAN; SIVAKUMAR; NARENDRA, 2011)	IEE SCC/2011
	(LIU et al., 2009)	IEE SCC/2009
	(BIANCHI et al., 2014)	J-BHI/2014
	(BAGHDADI; AL-BULUSHI, 2013)	SOCA/2013
Springer	(STUBBINGS; POLOVINA, 2013)	Computing/2013
	(KRAMMER et al., 2011)	Business and Inf. Sys. Eng./2011
IEEE-TSE	(PEREPLETCHIKOV; RYAN, 2011)	IEEE TSE/2011

3.2 Characterization

Items for characterization of the research on application of Service-Oriented principles, and how SOA is applied in practice, are domain of application, situations in which SOA is applied in practice, which principles are used, how published researches are being validated, found benefits of SOA design principles, and how SOA design has being modeled.

3.2.1 Use of SOA

In this section, an analysis of how SOA has been used in practice is presented. Table 4 shows that SOA has been considered in activities related to refactoring legacy systems, integration of systems with the purpose of producing a new system, and to create new methodologies to develop systems. The aims of this subsection is to answer the following research question: *SLR-RQ1 - What are the purposes of using SOA?*

Table 4 – Use of SOA in each paper

Article	Refactoring	Integration	New System	New Method
(LANMAN et al., 2013)	●	●		
(CELLARY; STRYKOWSKI, 2009)		●	●	
(GARCÍA-SÁNCHEZ et al., 2013)			●	
(PAUTASSO; WILDE, 2009)				●
(DIRGAHAYU; QUARTEL; SINDEREN, 2008)		●		●
(KHOSHKBARFOROUSHHA; JAMSHIDI; SHAMS, 2010)		●		●
(MONSIEUR; SNOECK; LEMAHIEU, 2012)	●			
(ARDITO et al., 2014)			●	
(GRANELL; DÍAZ; GOULD, 2010)		●	●	
(CHO et al., 2010)		●	●	
(TIAN; HUANG, 2012)		●		
(YAMANY; CAPRETZ; ALLISON, 2010)		●		
(WU et al., 2009)		●		
(KABIR; HAN; COLMAN, 2014)		●		
(XING; YAO, 2010)	●	●		
(XIAO-JUN, 2010)		●		
(KING et al., 2010)		●	●	
(VESCOUKIS; DULAMIS, 2011)		●	●	
(LIU, 2009)				●
(KANNAN; BHAMIDIPATY; NARENDRA, 2011)		●	●	
(KANNAN; SIVAKUMAR; NARENDRA, 2011)		●		
(LIU et al., 2009)	●			
(BIANCHI et al., 2014)	●			
(BAGHDADI; AL-BULUSHI, 2013)	●			
(STUBBINGS; POLOVINA, 2013)				●
(KRAMMER et al., 2011)	●			
(PEREPLETCHIKOV; RYAN, 2011)	●			

Currently, SOA is well-recognized as an established architectural paradigm for distributed systems (GOEB; LOCHMANN, 2011). Several systems can be connected to exchange information and implement business requirements. Services in SOA are platform independent (PAPAZOGLU, 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 (ALONSO et al., 2010).

Some articles in this review present real applications of using SOA to integrating systems. For example, in (KANNAN; SIVAKUMAR; NARENDRA, 2011), a real world application that uses SOA to create a service called AddressManagement is developed. This service currently retrieves addresses given an address identifier. Other articles present real word applications of SOA through creation of new systems. In these cases, integration of systems originated new systems. For instance, the proposal in (GRANELL; DÍAZ; GOULD, 2010) is a new software called "The AWARE" project, which overall goal is to provide hydrologists with distributed and reusable tools to monitor and to predict water availability.

Another important contribution of SOA is to enable reuse of legacy systems. SOA has emerged promising to allow legacy systems to expose their core functionality as services (KHADKA et al., 2011).

Some studies in this review presented applications in industry that used SOA to refactoring legacy systems. In one case (LIU et al., 2009), services are created to update and insert business rules. According to the authors, SOA represents an opportunity to renovate core banking systems in a progressive manner. SOA enables banks to create an open, flexible technology framework, making integration of both internal and external applications relatively simple and allowing for far easier and faster changes to business.

In other studies, SOA is presented with new methodologies. Those articles do not present real applications of SOA, but they propose new methodologies based on SOA. Article (STUBBINGS; POLOVINA, 2013) proposed a new methodology to teach SOA using concepts of object-oriented programming. Article (LIU, 2009) proposes an approach to measure service oriented usability.

3.2.2 Most used SOA design principles

The second research question is related to SOA design principles: *SLR-RQ2 - What are the most used design principles for SOA applications?* Some specific design principles for SOA have been used in the studies. One popular work in specific design principles for SOA is the one proposed in (ERL, 2007). Some other studies have cited other authors in this area, including (LEGNER; VOGEL, 2007), (LIU, 2009), (XING; YAO, 2010), (KANNAN; BHAMIDIPATY; NARENDRA, 2011).

Table 5 presents design principles cited by each study. In this table, the following words were abbreviated as follows. C means Service Contract, LC means Service Loose Coupling, A means Service Abstraction, R means Service Reusability, At means Service Autonomy, S means Service Statelessness, D means Service discoverability, Cm means Service Composability, G means Service Granularity, E means Service Encapsulation,

Table 5 – Design Principles used in each paper

Study	C	LC	A	R	At	S	D	Cm	G	E	O	Rl	U	BA	SM	CM	I	TU
(LANMAN et al., 2013)																		
(CELLARY; STRYKOWSKI, 2009)	●	●	●	●	●	●	●	●										
(GARCÍA-SÁNCHEZ et al., 2013)		●				●	●											
(PAUTASSO; WILDE, 2009)	●					●												
(DIRGAHAYU; QUARTEL; SINDEREN, 2008)																		
(KHOSHKBARFORUSHHA; JAMSHIDI; SHAMS, 2010)				●				●										
(ARDITO et al., 2014)								●										●
(MONSIEUR; SNOECK; LEMAHIEU, 2012)		●	●	●				●	●									
(GRANELL; DÍAZ; GOULD, 2010)		●		●					●									
(CHO et al., 2010)		●		●														
(TIAN; HUANG, 2012)							●											
(YAMANY; CAPRETZ; ALLISON, 2010)	●	●	●		●		●											
(WU et al., 2009)								●										
(KABIR; HAN; COLMAN, 2014)																		
(XING; YAO, 2010)	●	●	●	●	●		●	●		●	●	●						
(XIAO-JUN, 2010)		●							●									
(KING et al., 2010)																		
(VESCOUKIS; DULAMIS, 2011)																		
(LIU, 2009)													●					
(KANNAN; BHAMIDIPATY; NARENDRA, 2011)		●		●	●	●								●				
(KANNAN; SIVAKUMAR; NARENDRA, 2011)	●		●	●														
(LIU et al., 2009)				●											●	●	●	
(BIANCHI et al., 2014)									●									
(BAGHDADI; AL-BULUSHI, 2013)																		
(STUBBINGS; POLOVINA, 2013)	●	●	●	●	●	●		●										
(KRAMMER et al., 2011)									●									
(PEREPLETCHIKOV; RYAN, 2011)		●	●	●	●			●										

O means Service Optimization, Rl means Service Relevance, U means Service Oriented Usability, BA means Business Alignment, SM means Service Component Modularity, CM means Change Management, I means Integration and TU means Transformative User Experience.

All design principles mentioned in Table 5 were briefly explained in Section 2.2.2.

Six out of twenty-seven studies did not consider any design principle. Articles (LANMAN et al., 2013), (DIRGAHAYU; QUARTEL; SINDEREN, 2008), (KING et al., 2010), (VESCOUKIS; DULAMIS, 2011), (KABIR; HAN; COLMAN, 2014) and (BAGHDADI; AL-BULUSHI, 2013) have not cited using design principles for development of their SOA applications. The fact that these studies did not cite the use of any principles does not mean that they were not considered in the SOA system, but it was not mentioned in the article its use.

Twenty-one out of twenty-seven studies cited they used at least one SOA design prin-

ciple. Service Loose Coupling and Service Reusability were the most cited principles in studies (both were cited 11 times). In general, design principles described by Erl in (ERL, 2007) - Service Contract, Service Loose Coupling, Service Abstraction, Service Reusability, Service Autonomy, Service Statelessness, Service discoverability and Service Composability - are the most widely applied. Therefore, despite the work of Erl can be considered for business professionals in industry, it is also relevant to academic studies.

Granularity is a design principle proposed in (LEGNER; VOGEL, 2007) and was considered by 3 studies ((XIAO-JUN, 2010), (GRANELL; DÍAZ; GOULD, 2010) and (MONSIEUR; SNOECK; LEMAHIEU, 2012)). Service Encapsulation, Service Optimization, Service Relevance, Service Oriented Usability, Business Alignment, Service Component Modularity, Change Management, Integration and Transformative User Experience are less addressed by studies. Each one of these design principles were cited just once.

Some articles (Articles (KING et al., 2010), (KRAMMER et al., 2011), (VESCOUKIS; DULAMIS, 2011), (BAGHDADI; AL-BULUSHI, 2013)) did not explicitly mention using design principles for implementation of the case study. Despite this, it is possible to infer that design principles were used during development. For example, in (VESCOUKIS; DULAMIS, 2011) the principle of Service Contract was used because the WSDL standard which specifies service interface is mentioned.

3.2.3 Benefits of applying SOA design principles

Some studies applied design principles in their SOA applications. Therefore, it is important to analyze if design principles for SOA provide benefits for final application quality and how these benefits are measured. The research question defined for this analysis is *SLR-RQ3 - What are the benefits of applying SOA design principles? Are these benefits measured?* Few articles (only 2 of 27) mentioned influence of design principles on final software quality.

In (XIAO-JUN, 2010), authors addressed the influence of design principles in quality of SOA applications. Considered design principles were coupling and granularity. Metrics for measuring coupling and granularity of services were proposed. As for conclusion, the paper explains that, in order to improve quality of SOA applications, it is critical to obtain low values in coupling metrics and high values in granularity metrics.

Authors of article (KANNAN; BHAMIDIPATY; NARENDRA, 2011) states Services that follow design principles are robust to changes and are largely reusable in multiple scenarios but in similar domains. Based on this, the article proposes a formal, rigorous approach to check adherence of design principles in the solution. Considered design principles were Business Alignment, Coupling, Reusable, Autonomous and Stateless. This approach will help designers to validate if Services follow principles at design time.

3.2.4 SOA applications evaluation

Most studies were validated by using case studies. The answer for *SLR-RQ4 - How are SOA applications evaluated in practice?* is that only one article was evaluated with a controlled Experiment (Table 6).

Table 6 – Validation in each paper

Study	Case Study	Controlled Experiment	Without Validation
(LANMAN et al., 2013)	●		
(CELLARY; STRYKOWSKI, 2009)	●		
(GARCÍA-SÁNCHEZ et al., 2013)	●		
(PAUTASSO; WILDE, 2009)			
(DIRGAHAYU; QUARTEL; SINDEREN, 2008)	●		
(KHOSHKBARFOROUSHHA; JAMSHIDI; SHAMS, 2010)	●		
(ARDITO et al., 2014)	●		
(MONSIEUR; SNOECK; LEMAHIEU, 2012)	●		
(GRANELL; DÍAZ; GOULD, 2010)	●		
(CHO et al., 2010)	●		
(TIAN; HUANG, 2012)	●		
(YAMANY; CAPRETZ; ALLISON, 2010)	●		
(WU et al., 2009)			●
(KABIR; HAN; COLMAN, 2014)	●		
(XING; YAO, 2010)	●		
(XIAO-JUN, 2010)	●		
(KING et al., 2010)	●		
(VESCOUKIS; DULAMIS, 2011)	●		
(LIU, 2009)			●
(BIANCHI et al., 2014)	●		
(KANNAN; BHAMIDIPATY; NARENDRA, 2011)	●		
(KANNAN; SIVAKUMAR; NARENDRA, 2011)	●		
(LIU et al., 2009)	●		
(BAGHDADI; AL-BULUSHI, 2013)	●		
(STUBBINGS; POLOVINA, 2013)			●
(KRAMMER et al., 2011)	●		
(PEREPLETCHIKOV; RYAN, 2011)	●	●	

Two types of validation were performed in (PEREPLETCHIKOV; RYAN, 2011). First one is a case study: a SOA software developed specifically for this study, based on an existing service-oriented Academic Management System. Second step on this study is to conduct a controlled experiment to evaluate proposed coupling metrics.

Three studies - (WU et al., 2009), (LIU, 2009) and (STUBBINGS; POLOVINA, 2013) - were proposed without any kind of validation.

Article (PAUTASSO; WILDE, 2009) produced a quantitative evaluation by comparing the degree of coupling implied by different Web services technologies: RESTful HTTP, RPC over HTTP, and WS-*.

3.2.5 SOA applications design

Research question addressed in this section is *SLR-RQ5 - How are SOA applications modeled?* Some issues about modeling SOA are addressed in this review, such as how modeling is performed, and which modeling languages and specific diagrams are used for modeling SOA applications. Table 7 shows which modeling diagrams have been used to

design SOA applications. The reason why not all 27 studies appear in Table 7 is because only studies that explicitly applied modeling diagrams are presented in this Table.

In Table 7, words had to be abbreviated due to lack of space. FC means Flowcharts, DF means Data Flow, FSG means Functionality and Service Graph (a graph defined to identify services based on an aggregation or disaggregation of functions or functionality). With respect to UML diagrams, Activ means Activity Diagram, Class means Class Diagram, Collabo means Collaboration Diagram, Comp means Components Diagram and Seq means Sequence Diagram.

One study (DIRGAHAYU; QUARTEL; SINDEREN, 2008) represents services graphically, but the diagrams defined by authors did not follow any standard. Due to lack of standardization, understanding of diagrams is difficult.

Nine studies have used some kind of diagram to model the developed system. Most studies, 18 out of 27, did not mention any kind of modeling language.

Table 7 – Modeling Diagrams used in each study

Study	DF	FC	FSG	UML Diagrams				
				Activ	Class	Collabo	Comp	Seq
(MONSIEUR; SNOECK; LEMAHIEU, 2012)	●							
(GRANELL; DÍAZ; GOULD, 2010)				●				
(CHO et al., 2010)							●	
(YAMANY; CAPRETZ; ALLISON, 2010)								●
(WU et al., 2009)		●						
(KANNAN; BHAMIDIPATY; NARENDRA, 2011)								●
(KRAMMER et al., 2011)			●					
(PEREPLETCHIKOV; RYAN, 2011)	●	●			●	●		●
(TIAN; HUANG, 2012)								●

SOA applications have been modeled with different diagrams. Each software designer represents the system and services which must be developed in whatever way is most appropriate. However, this represents a lack of standardization in this area. In addition, the fact most studies did not mention using diagrams can mean the description of service functionalities is performed in natural language, which can lead to problems related to misunderstandings, inconsistencies, and ambiguities (KAMSTIES, 2005) (SOARES; VRANCKEN; VERBRAECK, 2011). In this context, it is possible to reflect on some questions for future research. Can SOA applications be fully modeled using general purpose diagrams, such as UML diagrams? Does Natural language have been used due to lack of adequate diagrams?

Surprisingly, SoaML (OMG, 2012), a modeling language specifically defined for designing SOA applications, was not mentioned in this review. Reasons for this are not yet clear, and need to be investigated in future research. One possibility is the novelty of the language, officially released on March 2012 (even though beta versions were published in 2009).

Table 8 – Domain Application of each article

Article	Domain	Comments
(LANMAN et al., 2013)	Military system	United States Army Program application to military live training community.
(CELLARY; STRYKOWSKI, 2009)	Public Sector	E-services (e.g., administrative, judicial, financial) for a customer perform the whole process.
(GARCÍA-SÁNCHEZ et al., 2013)	Evolutionary Computation	Design and implementation of services for Evolutionary Computation.
(PAUTASSO; WILDE, 2009)	Not-described	-
(DIRGAHAYU; QUARTEL; SINDEREN, 2008)	Purchase Order	Integration of the ordering application of a customer.
(KHOSHKBARFOROUSHHA; JAMSHIDI; SHAMS, 2010)	Healthcare application	Healthcare processes related with Visiting Preparation.
(ARDITO et al., 2014)	Personal Information Spaces application	a platform that allows end users, who are not necessarily experts of technologies, to compose Personal Information Spaces
(MONSIEUR; SNOECK; LEMAHIEU, 2012)	Hospital and Belgian banking	Real example
(GRANELL; DÍAZ; GOULD, 2010)	Geospatial Services	Discovery, access, processing and visualization of geospatial data
(CHO et al., 2010)	Clinical decision support	Healthcare information system
(TIAN; HUANG, 2012)	Geospatial data	To find and access geospatial data over Internet using a single tool
(YAMANY; CAPRETZ; ALLISON, 2010)	Purchase products	Simulation of consumers purchasing products from a large firm
(WU et al., 2009)	Not-described	-
(KABIR; HAN; COLMAN, 2014)	Automotive Application	A cooperative convoy telematics system that allows interaction and collaboration between automotive vehicles.
(XING; YAO, 2010)	Remote Collaborative Experiment Systems	Enables storing and sharing data about scientific experiments
(XIAO-JUN, 2010)	Claim Approval	Create, pay and approve claims
(KING et al., 2010)	System for Financial Analysis	Pricing and data/information service system for financial analysis of securities
(VESCOUKIS; DULAMIS, 2011)	System for evaluation of disaster management	Collect, process, visualize and interpret geospatial data to intelligent support decision making
(LIU, 2009)	Not-described	-
(KANNAN; BHAMIDIPATY; NARENDRA, 2011)	Academic System	Student Course Registration System (not implemented).
(KANNAN; SIVAKUMAR; NARENDRA, 2011)	Address Management	Perform retrieval of matching addresses given an address identifier.
(LIU et al., 2009)	Asian bank	Real application
(BIANCHI et al., 2014)	Healthcare application	system for drug prescription, administration, and registration
(BAGHDADI; AL-BULUSHI, 2013)	Student information system	Real system
(STUBBINGS; POLOVINA, 2013)	Not-described	-
(KRAMMER et al., 2011)	Financial Service Provider	Loan division of major German financial service provider.
(PEREPLETCHIKOV; RYAN, 2011)	Academic Management System	Software system developed specifically for this study

3.2.6 Domains of SOA applications

The research question addressed is *SLR-RQ6 - For which domains are SOA applications developed?* Articles found in this SLR present diversity in terms of fields of application of SOA. Case studies of each article are presented in Table 8. This listing

has been performed to show which application domains are used in the case studies. It is interesting to note that SOA has been applied in different domains, and the application of SOA is comprehensive for various domains. In Table 8, one can analyze that SOA has been applied from academic applications to military, healthcare and infrastructure systems.

3.3 Discussion

According to the presented SLR, SOA has been applied in various areas, including financial systems, manufacturing of goods, healthcare information systems, research-oriented applications in engineering, and academic software applications. Therefore, it is safe to assume that SOA has been widely applied in projects both in academia and industry.

Specific design principles for SOA were proposed in literature with the purpose of being crucial guides on how to create applications based on services. However, few studies dealing with applying design principles and how they affect quality are mentioned. Design principles guide about how services can be reused, but the high reusability of an application is to be defined by good decisions taken by developers, which depends on knowledge and experience. This is true as well regarding granularity.

Although SOA design principles are considered fundamental for SOA applications and are widely referenced in literature, authors of six out of twenty-seven studies have not explicitly mentioned whether they have used SOA design principles. Twenty-one out of twenty-seven studies cited using at least one SOA design principle. Sixteen studies cited using two or more design principles. Design Principles are presented as key concepts, but are not always shown in case studies as being explicitly used and how they are used in practice. Design principles are proposed as key concepts to help developing good design solutions. However, only two articles mentioned influence of design principles on final software quality. Quality of SOA applications has been poorly mentioned as described in this article.

Even though a specific modeling language for SOA has been proposed, the SoaML modeling language was not mentioned in this review. The reasons for this result are not clear, and can only be speculated. One possibility refers to the novelty of the language. In addition, UML is well-known in academia and software industry, which makes it a suitable choice for most authors.

As for limitations of this work, the number of selected articles can be considered a threat to validity. Although most systematic literature reviews selected from 5% to 15% papers, and the one presented in this article selected approximately 6%, the final number of selected papers, 27, could be higher in future works. Another threat to validity is that results are based on information written in published papers only, not on interviews

with authors. Therefore, even when a design principle is not mentioned in a study, it is possible the principle was used but the authors did not mention. Another example is the choice of modeling language. Even in articles which did not mention the use of modeling languages, one cannot infer the authors of respective articles did not use one.

3.4 Research Directions

Results of SLR presented in this chapter were the basis to define research directions in this thesis. Some gaps in SOA research were found during the analyses of studies. Research gaps presented in the SLR have become opportunities to present a more comprehensive study with regard to the development of SOA applications focusing on principles of design and quality. These gaps are shown as follows.

Firstly, it was interesting to note that SOA has been applied in a variety of domains. SOA was used in several contexts such as academia, financial, geospatial, commerce and healthcare. This scenario demonstrates the SOA potential to develop systems in a wide range of domains. However, most studies (25 of 27) do not present how to apply design principles in SOA applications. Design Principles were presented as key concepts, but they were not shown in case studies as being explicitly used and how they were used in practice. These studies focus on presenting the application itself. In most cases, system functionalities were presented but it was not shared how the application was developed and how the principles were applied. Therefore, the study proposed in this thesis is concerned both with the application and how to develop the application. It is extremely important to provide completeness of systems functionalities as agreed with stakeholders. However, it is also extremely important to follow what software engineering suggests in relation to search for better ways of developing software. Therefore, it is important to develop a system using principles proposed for SOA and also present how to apply them such as it was done in this study.

Another gap presented during SLR studies analysis is related to the use of modeling language in SOA design. All studies found used general purpose languages such as UML. The problem is that these models can not represent service that is the main concept in SOA systems. UML can be used to design services through definition of extensions and stereotypes which is the SoaML proposal. In addition to this, as mentioned earlier, some studies have presented the design of the system through informal drawings or even using natural language. This represents a lack of standardization in this area. No study used modeling language specific for SOA systems although there are some proposals such as SoaML. The experience of using SoaML are report in Chapter 7. The final conclusion of the use of SoaML is that it is useful and improves SOA system understanding.

As mentioned before, most of studies focus on presenting the application itself but not on how it was implemented. This approach does not make clear which quality criteria

were used. Only two of twenty-seven articles mentioned influence of design principles on final software quality. Therefore, it is important to define which quality attributes should be considered during software development. Case study proposed in this thesis consists of development of EHR considered a set of quality attributes. Quality attributes were proposed by ISO 25010 but they were adapted to SOA as proposed by SOAQM. During the EHR development a set of architectural decisions were proposed to achieve essential quality attributes. The approach proposed focus on considering quality in early stages of development.

Quality Model for SOA

The content of this chapter is mainly based on the paper entitled “SOAQM: Quality Model for SOA Applications based on ISO 25010” (FRANÇA; SOARES, 2015). This paper was published on the proceedings of the International Conference on Enterprise Information Systems (ICEIS) in 2015.

4.1 Problem Description

Software systems are becoming increasingly complex over time and, thus, quality assurance is becoming increasingly important as well (BOEHM, 2006) (HUANG et al., 2012). To ensure adequate software quality, relevant quality characteristics must be specified, taking into account the intended use of a software product. In order to make a proper evaluation of software, relevant quality characteristics of a software product have been proposed in many quality models, including ISO standards.

ISO 9126 (ISO/IEC, 1991) has inspired several quality models. In 2011, ISO 9126 was replaced by ISO 25010 (ISO/IEC, International Organization for Standardization, 2011a). There are many specific quality models for SOA already proposed in the literature. A systematic review proposed by Oriol et al. (ORIOLO; MARCO; FRANCH, 2014) presented 47 quality models specific to Web services. Only 6 out of 47 models were based on an ISO standard. As result of this SLR, the most quality model for SOA proposed in literature did not take into consideration ISO standards. According to these authors, as most SOA quality models do not consider ISO standards is a problematic situation mainly because of lack of standardization of concepts and definitions. In addition, none of these 47 models is based on ISO 25010 that is the newest version that addresses System and Software Quality Models.

According to Oriol et. al (ORIOLO; MARCO; FRANCH, 2014), there are several proposals for quality models for general purpose software systems. These quality models are different in terminology and in the structure of quality attributes classification. Oriol et. al, affirm that the most widely adopted quality model is that proposed by ISO / IEC,

especially ISO 9126 and its successor ISO 25010. A preference for ISO quality models is due to the more concrete proposal as it classifies the attributes of software quality in a structure of characteristics and sub-characteristics. Since ISO is the most widely used software quality model, it was expected that SOA-specific quality models would be based on the ISO standard. But instead, a systematic review of Oriol showed that 41 of the 47 quality models for SOA did not consider an ISO. This represents 87% of the quality models for SOA, that is, a large majority do not follow the standard established by ISO. As a final conclusion of this SLR, the author suggests that new classifications or definitions to be avoided because the models repeat the concepts and denominate differently and if they used the same norm as base, the nomenclature pattern would be used.

One of the proposals of this thesis is to investigate the applicability of ISO 25010 to SOA applications. The aim is to analyze all the quality attributes (characteristics and sub-characteristics) proposed by ISO 25010 and determine which of them are directly applicable to define quality in SOA. In this direction, the research question RQ3 proposed in this thesis is defined as follows:

RQ2 - What ISO 25010 characteristics are most suitable when developing SOA applications?

The answer to this question in this thesis is the definition of a quality model specific for SOA based on quality attributes defined by ISO 25010.

4.2 SOAQM - Quality Model for SOA applications

ISO 25010 proposes a set of 31 characteristics to define quality in software. These quality characteristics need to be studied and adapted in order to be applied to SOA applications.

SOAQM was proposed to define the real applicability of ISO 25010 quality characteristics to SOA applications. For each characteristic was established the degree of applicability to SOA. An analysis was produced to define how these characteristics can be adapted to the SOA context. This analysis established a degree of importance for each ISO quality attribute because is important to know which characteristics are more relevant during the SOA application development process. Fig. 20 presents the result of this analysis. SOAQM was classified into three categories: essentials, importants and non-relevants. According to SOAQM, twenty quality attributes proposed by ISO 25010 are totally applicable in SOA context and they are essential in SOA development. Nine quality attributes defined by ISO 25010 was considered important to SOA and two of them are non-important.

The definition of the SOAQM quality model was produced with the collaboration of seven SOA experts. All volunteers have experience in SOA development in industry and two of them are also researchers in the SOA domain. Experts work in industry for more

SOAQM quality attributes			
Essential		Important	Non-relevant
1 Functional completeness	11 Fault tolerance	1 Capacity	1 User interface aesthetics
2 Functional correctness	12 Recoverability	2 Appropriateness	2 Accessibility
3 Functional appropriateness	13 Confidentiality	recognizability	
4 Time behaviour	14 Integrity	3 Learnability	
5 Resource utilization	15 Accountability	4 User error protection	
6 Co-existence	16 Authenticity	5 Non-repudiation	
7 Interoperability	17 Modularity	6 Analyzability	
8 Operability	18 Reusability	7 Adaptability	
9 Maturity	19 Modifiability	8 Installability	
10 Availability	20 Testability	9 Replaceability	

Figure 20 – Quality attributes for SOA defined by SOAQM

than five years, as developers, software architects or project managers. Experts have been working in different regions in Brazil, including Uberlândia-MG, São Paulo-SP and Aracaju-SE.

Volunteers answered a questionnaire to define the degree of importance of each quality characteristic in SOA applications. Each question represents a quality sub-characteristic definition and each volunteer was responsible to answer how important this sub-characteristic is for SOA. The answer varies from 1 to 5 following the Likert Scale (1-Not Important, 2-Less Important, 3-Neutral, 4-Important, 5-Very Important). This questionnaire is shown in Appendix A and the results to this survey are described in Table 9.

Table 9 presents volunteers responses about the importance of each sub-characteristic to SOA context. The last two columns represents average and standard deviation of volunteer answers. The score average was important to define three degrees of importance: essential (green), important (yellow) and non-relevant (red). High scores represents that quality attribute is very important to SOA applications and it should be considered essential. Yellow scores identify important quality attributes and the red ones non-relevants.

Table 10 presents the results of the analysis of ISO 25010 characteristics and their applicability to SOA projects. First two columns of Table 10 show all characteristics and sub-characteristics proposed by ISO 25010.

Third column of Table 10 presents the degree of importance of each sub-characteristic with relation to the SOA context. Highlighted sub-characteristics in **green** are important or very important quality attributes for SOA applications. These quality attributes were considered important because volunteers opinion indicates high values which represents scores between 4 and 5 in Likert Scale. Sub-characteristics highlighted in **yellow** are not so important, but they are relevant for SOA and can not be disregarded. Quality

Table 9 – Volunteers Opinion about importance of ISO 25010 for SOA - Likert Scale.

Characteristic	Sub-characteristics	Volunteers Opinion							Statistics	
		#1	#2	#3	#4	#5	#6	#7	Average	Stand. Dev.
Functional Suitability	Functional completeness	5	4	5	5	3	5	2	4.1	1.1
	Functional correctness	4	5	5	4	4	5	5	4.6	0.5
	Functional appropriateness	5	5	5	4	4	5	2	4.3	1.0
Performance efficiency	Time behaviour	5	5	4	5	4	5	2	4.3	1.0
	Resource utilization	3	5	4	4	4	5	4	4.1	0.6
	Capacity	1	5	3	4	3	5	1	3.1	1.6
Compatibility	Co-existence	3	5	3	5	3	5	4	4.0	0.9
	Interoperability	5	5	3	5	4	5	5	4.6	0.7
Usability	Appropriateness recognizability	5	2	4	4	3	4	1	3.3	1.3
	Learnability	5	2	4	4	4	4	3	3.7	0.9
	Operability	5	5	4	2	4	5	3	4.0	1.1
	User error protection	5	5	4	4	3	4	1	3.7	1.3
	User interface aesthetics	4	3	4	2	3	1	1	2.6	1.2
	Accessibility	4	2	4	4	3	1	1	2.7	1.3
Reliability	Maturity	4	5	4	4	4	5	4	4.3	0.5
	Availability	5	5	4	5	5	5	5	4.9	0.3
	Fault tolerance	5	5	4	5	4	5	4	4.6	0.5
	Recoverability	5	5	4	5	4	5	5	4.7	0.5
Security	Confidentiality	5	5	4	4	4	5	5	4.6	0.5
	Integrity	5	5	4	4	4	5	5	4.6	0.5
	Non-repudiation	4	3	4	4	4	4	4	3.9	0.3
	Accountability	5	5	4	4	4	4	4	4.3	0.5
	Authenticity	5	5	4	4	4	5	5	4.6	0.5
Maintainability	Modularity	5	5	4	4	4	5	5	4.6	0.5
	Reusability	5	5	4	4	4	5	5	4.6	0.5
	Analyzability	5	4	4	2	5	5	2	3.9	1.2
	Modifiability	5	5	4	4	4	5	5	4.6	0.5
	Testability	5	5	4	4	4	5	5	4.6	0.5
Portability	Adaptability	5	2	3	2	4	5	5	3.7	1.3
	Installability	4	1	3	2	4	5	5	3.4	1.4
	Replaceability	4	2	3	4	3	5	5	3.7	1.0

attributes highlighted as yellow received scores between 3 and 3.9. On the other hand, there are quality sub-characteristics considered less important or even irrelevant to SOA applications and therefore were highlighted as **red**. These quality attributes received low degree of importance to SOA with scores below 3.

The last column in Table 10 presents the likely reasons that justifies the results.

The remainder of this section is divided into eight subsections. Each of the eight subsections addresses one ISO quality characteristic and also discusses how one can suit them in the context of SOA. These subsections are also important to present the reasoning behind each definition described in Table 10.

4.2.1 Functional Suitability

Sub-characteristic Functional Completeness means the degree to which the set of functions covers all the specified tasks and user objectives. Observing from the SOA perspective, services must cover all the specified tasks and user objectives for which they were designed. The first activity during the development of SOA applications include defining requirements to develop services. This phase is commonly known as service-oriented analysis. Service-oriented analysis (ERL, 2005) is a process of determining requirements, scope of service and which services will be developed. These important definitions are

Table 10 – ISO 25010 characteristics mapped to SOA quality characteristics

Characteristic	Sub-characteristics	Importance for SOA	SOA Perspective
Functional Suitability	Functional completeness	4.1	Services must cover all the specified tasks and user objectives which were designed
	Functional correctness	4.6	Services should provide the correct results with the needed degree of precision
	Functional appropriateness	4.3	Services are designed to facilitate accomplishment of specified tasks, more precisely, the execution of a business process.
Performance efficiency	Time behaviour	4.3	Time spent by a service to process a request and return a response.
	Resource utilization	4.1	Services use resources such as servers to access information of other applications.
	Capacity	3.1	Service capacity can be defined as the ability to remain working even with large number of accesses at the same time.
Compatibility	Co-existence	4.0	Different composite services can share the use of same service operations.
	Interoperability	4.6	Services are interoperable. Services allow interaction between systems through the use of interfaces (WSDL) and communication protocols (SOAP).
Usability	Appropriateness recognizability	3.3	Users can recognize whether this service is appropriate for their needs through service description that relate information such as service functionality and data types transmitted.
	Learnability	3.7	Degree to which a service can facilitate the understanding of its operation.
	Operability	4.0	A service has a WSDL document that allows exchange messages between services.
	User error protection	3.7	The WSDL structure of a service should not allow making errors from wrong inputs.
	User interface aesthetics	2.6	Not the focus of SOA.
	Accessibility	2.7	Not the focus of SOA.
Reliability	Maturity	4.3	Whenever a service consumer requests some information, it is expected that a response is returned.
	Availability	4.9	Services must be available when they are requested.
	Fault tolerance	4.6	Services can create strategies that may be performed when a failure happens on some hardware or software.
	Recoverability	4.7	Service ability to recover data when occurs some interruption or failure.
Security	Confidentiality	4.6	Information shared by a service provider can be accessed only to an authorized service client.
	Integrity	4.6	Services must be developed to prevent unauthorized access to, or modification of private data.
	Non-repudiation	3.9	Service provider constructs strategies to prove that an information have been delivered to a service consumer
	Accountability	4.3	Service are autonomous.
	Authenticity	4.6	The identity of the external service provider should be authenticated.
Maintainability	Modularity	4.6	Service provider hosts a network accessible software module.
	Reusability	4.6	Services are reusable.
	Analyzability	3.9	Analyze change impact when services need to be modified
	Modifiability	4.6	Services are loosely coupled. This characteristic reduces the dependency between services, increasing modifiability.
	Testability	4.6	services can be tested, for instance, through automated tools for functional testing
Portability	Adaptability	3.7	Although web services run remotely on a server, it can happen a change of platform.
	Installability	3.4	Although web services run remotely on a server, it can happen a change of platform.
	Replaceability	3.7	Although web services run remotely on a server, it can happen a change of platform.

incorporated into a document known as functional document, which is used to software validation.

Sub-characteristic Functional Correctness means the degree to which a product or system provides correct results with the needed degree of precision. This sub-characteristic can be applied in SOA by the fact that services should provide the correct response with the needed degree of precision. A service requests some information and a service provider is responsible to send the correct response. Service-oriented analysis (ERL, 2005), as mentioned in previous paragraph, is also important to Functional correctness due to the relationship between what was requested by the customer and what the service offers as response.

Sub-characteristic Functional Appropriateness is the degree to which the functions facilitate the accomplishment of specified tasks and objectives. In the SOA context, services are designed to facilitate accomplishment of specified tasks, more precisely, the execution of a business process. Services perform functions that can be simple requests for activities or complex business processes. Therefore, services can be of simple nature or composite (PAPAZOGLU, 2003). Composite services are construct of the orchestration process, which allows services to be composed of other services to automating business processes.

4.2.2 Performance efficiency

Performance efficiency characteristic can be applied in SOA applications to evaluate performance of a service.

Sub-characteristic Time-behaviour analyzes the response and processing times and throughput rates of a system, when performing its functions. Contextualizing for SOA, it is possible to say that response time is related to time spent by services to process a request and return a response. Generally, a service requests an information for a provider application and then this service provides responses for a consumer application. Thus, it is possible to measure response and processing times and throughput rates of this transaction.

Sub-characteristic Resource Utilization addresses the amounts and types of resources used by a product or system, when performing its functions. Generally, software should make a best use of resources such as processor capacity, memory usage, disk capacity and network bandwidth. With regard to SOA, service based applications interact with other systems exchanging messages over the network. Therefore, it is possible to consider that services use resources such as server to access information of other applications.

Sub-characteristic Capacity means the degree to which the maximum limits of a product or system parameter meet requirements. This quality sub-characteristic was vaguely defined by ISO. It is difficult to set the real meaning of this concept. It can be supposed that capacity can be the application ability to support several accesses at the same time

without performance variation. In this sense, capacity is related with availability of a service even with many access at the same time.

Performance efficiency is an important quality characteristic which must be constantly observed in SOA applications. According to O'Brien (O'BRIEN; MERSON; BASS, 2007), performance is negatively affected in SOA. Therefore, the architecture should be carefully designed and evaluated prior to implementation to avoid performance pitfalls.

Service application performance depends on the combined performance of cooperating components and their interactions. Beyond consumers and providers applications and the need to communicate over the network, services can be composed. Composite services allows services to be composed of other services, in such a way the logic of the process is centralized by orchestration that enables extensibility (composition of new services). Thus, time behaviour depends on several factors. To deal with so many factors that can affect system performance, companies need constantly to monitor the health of their SOA applications (PAPAZOGLU et al., 2008).

4.2.3 Compatibility

Sub-characteristic Co-existence is concerned with the degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product. From the SOA perspective, co-existence concept can be redefined/extended as follows. A composite service matches a set of services that perform operations and together form a specific task. As a service can be reused by several composite services, some efforts are needed to ensure what some authors have been called service conformance (PAPAZOGLU et al., 2008). Service conformance ensures the integrity of a composite service matching its operations with those of its constituent component services, imposes semantic constraints on the component services and ensures that constraints on data exchanged by component services are satisfied. This means the co-existence concept can be extended to SOA due to different composite services that share the use of same service operations.

Sub-characteristic Interoperability refers to the degree to which two or more systems can exchange information and use the information that has been exchanged. Services are interoperable (PAPAZOGLU et al., 2008). SOA utilizes services to construct an application that allows exchanging messages through networking protocol. This means services are interoperable and they allow interaction between systems through the use of interface (WSDL) and communication protocols (SOAP). To allow increased interoperability is the most prominent benefit of SOA (O'BRIEN; MERSON; BASS, 2007).

4.2.4 Usability

Within the SOA context, usability is a measure of the quality of a user's experience in interacting with information or services. Usability as a concept has been rarely addressed in the context of SOA. Few studies have been published addressing this issue (LIU, 2009). One research named this issue as service oriented usability (LIU, 2009), suggesting that some procedures may increase usability of services, such as development of diagrams for service and definition of life cycle and lifetime of services.

Sub-characteristic Appropriateness recognizability refers to the degree to which users can recognize whether a system is appropriate for their needs. In this sense, it is important to have access to some description that relate the service operations. Relationship between services is based on an understanding that for the services to interact, they must be aware of each other and this awareness is achieved through the use of service descriptions (ERL, 2005). A service description establishes the name and location of the service, and data exchange requirements. SOA uses a standard format for service description known as WSDL (Web Services Description Language) (W3C, 2016). Therefore, sub-characteristic Appropriateness recognizability can be applied in SOA because service description relates service functionality and data types transmitted, and so users can recognize whether this service is appropriate for their needs.

Sub-characteristic Learnability is related to the degree to which a system can be used by specified users to achieve specified goals of learning to use the system in an easy way. From the SOA perspective, this sub-characteristic refers to the degree to which a service can facilitate the understanding of its operation.

Sub-characteristic Operability refers to the degree to which a product or system has attributes that make it easy to operate and control. A service has a WSDL document that allows exchanging messages between services. These messages are transmitted through a standard protocol called SOAP (Simple Object Access Protocol) (W3C, 2016). The standards defined for SOA make it easy to develop, operate and control services.

Sub-characteristic User error protection is related to the degree to which a system protects users against making errors. Within the SOA context, this quality sub-characteristics may be appropriate to SOA by the fact that a service has a description document (WSDL) and its structure should not allow different inputs from the specified ones.

Sub-characteristic User interface aesthetics refers to the degree to which a user interface enables pleasing and satisfying interaction for the user. User interface aesthetics or graphical interface is not the focus of SOA. Researches in SOA are hardly concerned with this sub-characteristic. Probably, the reason is attributed to the fact that there it is research in other areas such as human computer interaction that can be leveraged to SOA applications.

Sub-characteristic Accessibility is related to the degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve

a specified goal in a specified context of use. Accessibility is not very important to services because this issue is not the focus of SOA knowledge and research.

4.2.5 Reliability

Sub-characteristic Maturity is related to the degree to which a system meets needs for reliability under normal operation. Clearly, maturity can be applied in SOA because whenever a service consumer requests some information it is expected that a response is returned. Therefore, Maturity is a very important sub-characteristic that can be addressed in SOA applications to maintain quality.

Availability is a very important sub-characteristic in SOA applications. Availability consists in defining the degree to which a system, product or component is operational and accessible when required for use. This concept can be associated with SOA by the fact that a service provider must be available when a service consumer requests some information. Downtime could affect the provider's finances and reputation. According to O'Brien (O'BRIEN; MERSON; BASS, 2007), external service providers usually agree to provide services under a service level agreement (SLA), which defines the contract for the provision of the service with details such as who provides the service, the guaranteed availability of the service, the escalation process, and the penalties to the provider if a service level is not met.

Sub-characteristic Fault Tolerance refers to the degree to which a system operates as intended despite the presence of hardware or software faults. From the SOA perspective, fault tolerance can be applied in SOA applications as meaning that strategies must be performed when a failure happens on some hardware or software. This ability is not SOA native and should be designed and implemented by SOA developers.

Sub-characteristic Recoverability is related to the degree to which, in the event of an interruption or a failure, a product or system can recover data directly affected and re-establish the desired state of the system. This quality sub-characteristic can be adjusted to SOA as regards the service ability to recover data when any interruption or failure occurs. Auto recovery tools have been developed to provide the functionality of recover the faulted instances and Invoke and Callback messages during business process execution.

4.2.6 Security

Services are designed to limit information in the service contract to what is really necessary for the service to be functionally useful to consumers (ERL, 2007). Information beyond that is published in a service contract, is considered private, and should not be made available for creating potential consumers of service.

Confidentiality is the degree to which a system ensures data are accessible only to those authorized to have access. This quality sub-characteristic must be found in SOA

applications because information shared by a service provider can be accessed only to an authorized service client. Business process interactions are always controlled from the (private) perspective of one of the business parties involved in the process (PAPAZOGLOU et al., 2008).

Another security quality sub-characteristic is Integrity. Integrity refers to the degree to which a system, product or component prevents unauthorized access to, or modification of, software or data. Services must be developed to prevent unauthorized access to, or modification of private data. Integrity is related with one of the design principles of software engineering proposed in SWEBOK (Software Engineering Body of Knowledge) (IEEE Computer Society, 2004): Encapsulation/information hiding. Encapsulation/information hiding means grouping and packaging elements and internal details of an abstraction and making those details inaccessible. In this way, services must publish in a registry, or repository such as UDDI. Ensuring implementation details are inaccessible provides the guarantee that this information is not modified or corrupted.

Sub-characteristic Non-repudiation is related to the degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later. In the SOA context, this concept can be the ability of a service provider to prove that information has been delivered to a service consumer.

Sub-characteristic Accountability refers to the degree to which actions of an entity can be traced uniquely to the entity. According to (PAPAZOGLOU et al., 2008), services are autonomous. That way, services have a contract that expresses a well-defined functional threshold which should not involve other services. In addition, Thomas Erl (ERL, 2007) defined Service Autonomy as a design principle. Service Autonomy refers to the ability to govern itself. When something is autonomous, it has freedom and control to make their own decisions without the need for external validation or approval. Accountability in the SOA context refers to the service state of being accountable, in other words, a service has the obligation of render an account for failure to perform as expected.

Another security quality sub-characteristic is authenticity. Authenticity refers to claim and identification of a subject or resource requests access to a certain information. A system built using a SOA approach may encompass services provided by third-party organizations. The identity of the external service provider should be authenticated.

According to Papazoglou (PAPAZOGLOU et al., 2008), security is a grand challenge within SOA applications. Services must be developed to be self-protecting, which can anticipate, detect, identify and protect against threats. Self-protecting components can detect hostile behaviors, e.g., unauthorized access and use, virus infection and proliferation, and denial-of-service attacks, as they occur and take corrective actions to make themselves less vulnerable. Self-protecting capabilities allow businesses to consistently enforce security and privacy policies. One of the strategies used to increase security is encryption. Encryption should be used to preserve confidentiality and preserve integrity.

However, as mentioned in (O'BRIEN; MERSON; BASS, 2007), encryption has the effect of increasing messages size.

4.2.7 Maintainability

Maintainability quality characteristic is very important in SOA due to its primary proposal of easy maintaining distributed applications. Before emergence of SOA, several distributed solutions were proposed, with varying degree of success and limitations. 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, lack of stability, difficult maintenance, and problems associated with extensibility and interoperability frameworks (ERL, 2005). SOA has emerged with the purpose of development of rapid, low-cost and easy composition of distributed applications.

Modularity is a quality sub-characteristic that defines the capacity of a system to be composed of components such that a change to one component has minimal impact on other components. This definition is similar to a service concept. In a typical service-based scenario employing the service foundations plane a service provider hosts a network accessible software module (an implementation of a given service) (PAPAZOGLU et al., 2008).

Reusability is an important quality sub-characteristic which indicates that an asset can be used in more than one system, or in building other assets. This definition can be extended to SOA. Service reusability is a SOA design principle that means services are reusable (ERL, 2007). Services encapsulate the logic that is useful for more than one service request. Logic encapsulated by services is generic enough to allow many usage scenarios and can be used for different types of service consumers.

Sub-characteristic Analyzability indicates the degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified. Within SOA context, Analyzability can be an important quality attribute for services. In situations that a service need to be modified, analysis of which composite services use this service are necessary.

Modifiability is a quality sub-characteristic related to the capacity of software to be modified without introducing defects or degrading existing product quality. Modifiability increases as the independence between modules of the system increases because a change in one module can affect all modules that are dependent. By definition, services are loosely coupled (PAPAZOGLU et al., 2007). This means there are few well-known dependencies between services. Thus, the cost of modifying the implementation of services is reduced and the overall system modifiability increases (O'BRIEN; MERSON; BASS, 2007).

Testability is a quality sub-characteristic found in a system or component in which test criteria can be established and tests can be performed to determine whether those

criteria have been met. Services can be tested through automated tools for functional testing. According to O'Brien (O'BRIEN; MERSON; BASS, 2007), testing a system based on SOA is more complex than an isolated software. For instance, if a runtime problem occurs, it may be difficult to find the cause. It can be within the service user, the service provider, the communication infrastructure, the discovery agent (UDDI), or it can be due to the load on the platform where the service executes. Trying to replicate problems in a test environment may be difficult or even not possible within a time frame.

4.2.8 Portability

Portability is a quality characteristic concerned with the effectiveness and efficiency with which a system can be transferred from one hardware or usage environment to another.

Sub-characteristic Adaptability refers to the degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environment.

Sub-characteristic Installability is related to the degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment.

Sub-characteristic Replaceability indicates the degree to which a product can replace another specified software product for the same purpose in the same environment.

In general, some authors consider that this quality characteristic is not so important in the SOA context because web services run remotely on a server. For instance, according to (ORIOLE; MARCO; FRANCH, 2014), Installability is usually not applicable to web services because they are executed remotely at the server side. On the other hand, a change of a server platform of an application provider can be performed and then services should normally maintain their situation.

4.3 Comparison with other SOA quality models

There are several general purpose quality models proposed for software systems. They differ on the terminology, the set of attributes that define quality, and the structure of the quality model. Generic quality models, as the ones proposed in the literature, need to be adapted to be applied to SOA. Generic standards can not completely conform to the web services domain (ORIOLE; MARCO; FRANCH, 2014). For this reason, there are many specific quality models for SOA already proposed in the literature.

A systematic review proposed by Oriol et al. (ORIOLE; MARCO; FRANCH, 2014) presented 47 quality models specific to Web services. As a result of this study, contrary to what could be expected, most of the proposals did not take into consideration ISO

standards. Only 5 out of 47 models were based on ISO 9126: WSQM (OASIS, 2005), S-Cube Quality Reference Model (GEHLERT; METZGER, 2008), GESSI (AMELLER; FRANCH, 2008), Yin et al. (YIN et al., 2010), and Nadanam and Rajmohan (Nadanam, P. and Rajmohan, R., 2012). Therefore, none of the 47 quality models were based on the new ISO 25010.

In the next paragraphs, a comparison is presented between those 5 quality models for SOA based on ISO 9126 and the SOAQM model proposed in this work, which is based on ISO 25010. The main objective is to present what has been proposed in the literature and detach the novelty that this new quality model for SOA proposes.

WSQM is a Quality Model for Web Services proposed by OASIS in 2005 (OASIS, 2005). WSQM comprises a set of quality attributes that are important for Web Services domain. WSQM model is structured into six categories named Web Services Quality Factors: Business Value, Service Level Measurement, Interoperability, Business Processing, Manageability and Security. Sub-factors are defined to represent quality attributes. Besides a hierarchical structure with categories and sub-factors, there is no concern to develop the same structure and use of all the attributes set by ISO 9126. In addition, WSQM does not present a mature state because many of the contained definitions lack precision (GOEB; LOCHMANN, 2011).

S-Cube Quality Reference Model (GEHLERT; METZGER, 2008) consists of a set of quality attributes for service-based applications. S-Cube was structured into 10 categories which are Performance, Dependability, Security, Data-related Quality, Configuration-related Quality, Network- and Infrastructure-related Quality, Usability, Quality of Use Context, Cost and Other. Then, 78 quality attributes were defined and distributed in these categories. S-Cube is based on ISO 9126 but some definitions proposed by ISO 9126 were not mentioned. For instance, Functionality, which is an important characteristic for ISO 9126, is not considered as a category. In addition, the classification of attributes is considered by the authors as follows in ISO 9126, which is not always true. S-Cube is considered a model too complex to be used in an intuitive way (GOEB; LOCHMANN, 2011).

GESSI is a quality model based on ISO 9126 proposed to focus on the quality characteristics that are observable (AMELLER; FRANCH, 2008). GESSI proposes a tool called SALMon to monitor services to detect Service Level Agreement violations. However, just a small set of quality attributes proposed by ISO 9126 are observable. Therefore, just three quality attributes were addressed by GESSI, which are related to sub-characteristics Availability, Time Behaviour, and Accuracy.

In (YIN et al., 2010), an ontology is proposed to support quality of services considering the following non-functional requirements: expressiveness, robustness, flexibility, scalability, performance, completeness, friendliness and interoperability. A mapping of these quality requirements and concrete requirements is proposed, with focus on only

part of ISO 9126.

Nadanam and Rajmohan proposed in 2012 (Nadanam, P. and Rajmohan, R., 2012) a quality model for evaluating QoS (Quality of service) for web services. This quality model also considered cloud computing issues. As a result of this study, the quality model proposed some attributes and metrics considered relevant for cloud services. Two quality characteristics defined by ISO 9126 were considered in this model: Efficiency and Reliability. These quality characteristics are extended to cover cloud computing peculiar features. Based on the derived key features, Reusability, Adaptability, Availability, Composability, Understandability and Scalability are newly defined.

Contrary to the result of the systematic review proposed in (ORIOLO; MARCO; FRANCH, 2014), it was found a quality model for SOA that is actually based on ISO 25010. A Software Quality Model for SOA proposed by Goeb et. al (GOEB; LOCHMANN, 2011) was inspired in ISO 25010. However, this model analyzes only 5 quality characteristics: Consumption, Understanding, Service Reuse, Support, and Extension. Furthermore, this model was not comprehensive enough to consider the 8 quality characteristic and 31 sub-characteristics proposed by ISO 25010. In addition, two out of five considered characteristics (Support and Extension) were not defined by any version of ISO. Another issue with this quality model is that it did not follow the standardization of names established by ISO 25010 because two characteristics were renamed: Appropriateness Recognizability was renamed to Understanding, and Functional Appropriateness was called Consumption.

Furthermore, existing models are not comprehensive and do not address all quality attributes proposed by ISO 25010, as they are based on ISO 9126. Even in this case, they do not always follow the ISO 9126 standard. Although the models do follow ISO 9126, such as the S-Cube, the classification of attributes does not follow strictly ISO 9126, and important characteristics such as Functionality are not considered. In another case, WSQM, there is a hierarchical structure with categories and sub-factors, but not all attributes of the ISO 9126 structure are used. This happens in GESSI as well, in which only 3 quality attributes are addressed, and in the model proposed in (YIN et al., 2010).

By observing the works proposed in the literature, it is possible to notice that there is a need for development of a specific quality model for SOA because generic standards can not completely conform to the web services domain (ORIOLO; MARCO; FRANCH, 2014). Although some quality models have already been proposed as described in this chapter, these models represent problems. According to (ORIOLO; MARCO; FRANCH, 2014) these quality models, in general, are too complex, immature and have lack precision definitions. In addition to this, these quality models are based in a ISO version proposed in 1991. More than twenty-five years have passed since the ISO 9126 was proposed and during that time the software field has evolved. Therefore, according to (GOEB; LOCHMANN, 2011) and (ORIOLO; MARCO; FRANCH, 2014) there is a need for development of a specific quality model for SOA based on the latest ISO 25010.

4.4 Discussion

This work addresses one gap in researches about SOA. There is a lack of quality models specific for SOA based on most recent ISO 25010. In this sense, in this work is proposed a quality model for SOA applications based on quality attributes proposed in ISO 25010. ISO 25010 is a standard for software quality that presents a quality model that may be applied for any kind of software products. ISO 25010 is currently the most complete version of quality models of ISO standards due to the great amount of proposed quality attributes.

All quality attributes proposed by ISO 25010 were analyzed in this research from the SOA perspective using both experts opinion and literature review. As a result, most of quality attributes proposed by ISO 25010 may be applicable for SOA. However, these quality attributes should be adapted to SOA contexts, as discussed in Section 4.2. On the other hand, ISO 25010 has a generic nature and therefore some quality attributes proposed do not apply to SOA domain, as described in this work.

One threat to validity is the small number of subjects that responded the survey. However, as they are experts in the SOA context, their opinion can be highly considered. There is great difficulty in finding participants to collaborate with evaluations. Due to the large number of assignments, many people are willing to help but can not devote about an hour to perform an activity that has voluntary characteristics. In addition, some people in the network are not always experts in SOA context.

Another threat to validity in this study is that SOAQM was produced using average scores as presented in Table 9. It is possible to note that some averages have high standard deviation which can represent different category if some volunteer could give another score. However, SOAQM was developed with a fundamental rule based on the consensus of the participants. Quality attributes with high scores of importance attributed by all participants are considered essential. Quality attributes that obtained variation of scores represent lack of consensus, so they were classified as important or less important.

EHR Architectural Design Decisions

The content of this chapter is mainly based on paper entitled “Architectural Design Decisions for SOA Applications with focus on Quality Attributes”. This paper will be submitted to an International Journal.

This chapter presents the ADDs which are part of the application architecture of a case study in the health domain. The EHR application was proposed to attend the needs of a public hospital of Federal University of Sergipe, Aracaju - Brazil. The EHR architecture is described through architectural decisions in this chapter, and with further views and models in the following chapters.

Architectural decisions were proposed using an approach with focus on quality. Each decision aims to achieve a set of twenty essentials quality attributes proposed in SOAQM that was presented in Chapter 4. Four architectural decisions were proposed to achieve quality attributes proposed for the EHR system.

5.1 Description of Current Situation at the Hospital

The EHR application proposed in a public hospital has been developed with focus on integrating data from a number of legacy systems that store important information about patients, exams, appointments and other data. Currently, this hospital has five different legacy systems storing patient information and hospital issues. These systems are not integrated and some data have to be registered more than once in the software systems.

The main purpose of the EHR application is to create one system to centralize patient’s data, as currently in the hospital important patient’s data are scattered in five different systems. Even business processes require the same data to be recorded in two different systems. Thus, users should launch an information into a system and then do the same for another system, performing double work.

Another critical factor is the fact that patient medical record is maintained in physical papers. Frequently, clinicians need to look for old records to monitor patient evolution. Within this hospital context, the EHR application has been developed based on SOA to

solve the current presented problems. Existing legacy systems that have important data about patients will be adapted to expose their functionality through services, and the EHR application has been developed to consume these services.

Three main constraints are considered in this EHR project.

Low budget

Information and Communication Technology is a sector that has low budget in a public hospital. It is hard to apply resources for modernization of software applications. Therefore, there is no possibility to start from scratch and discard all deployed legacy systems.

Lack of integration

The lack of integration between current legacy systems makes duplication of data a common side effect. Most Hospital's legacy systems are not integrated and thus they do not meet many routine needs of hospital staff. There are many software systems for different purposes, including software to automate exams, to provide primary care, and also to register patient data. However, these modules do not interact with each other in an affordable way. Thus, data needs to be duplicated in two or more different systems.

Automation of business rules

Important business rules have not yet been implemented in existing software systems, as discovered after mapping all current business processes in the hospital. For instance, patients' data are registered in and maintained as hard copies. Daily, hundreds of patients are treated at the hospital and all data and evolution are noted in physical medical records. In addition to this, these records should be kept for long periods (most often 20 years), to allow access to information stored on it at any time by the patient or his/her legal representative. Therefore, this scenario causes a big impact regarding storage space of this large amount of physical records. The large amount of paper occupies several rooms and it promotes difficulty in retrieving patient's records when clinicians need to consult past files to analyze patient evolution.

5.2 Architectural Design Decisions

This section presents a set of ADDs for an EHR application based on a set of quality attributes. The aim is to propose a set of architectural decisions that focus on quality of an EHR software application in the early stages of software development process. Each ADD is associated to one or more quality attributes.

Quality attributes proposed in SOAQM are the base to support architectural decisions for a SOA application. The definition of attributes that affect software quality assists the

identification of relevant aspects during software development. All these quality attributes were considered during development of the architecture. An analysis was performed to ensure that all quality attributes were met through architectural decisions.

Literature proposed different ADD models. These models have their similarities and dissimilarities as presented in (SHAHIN; LIANG; KHAYYAMBASHI, 2009). According to (SHAHIN; LIANG; KHAYYAMBASHI, 2009), all the ADD models have consensus on describing four major elements: Decision, Constraint, Solution and Rationale. Decision field explains in detail the importance of addressing the ADD, Constraint field states the limitations which influence architectural design, Solution expresses alternatives for solving the design issue and Rationale express arguments for justifying the decisions made. In addition, this work append quality attributes affected in each defined architectural decision.

In this thesis, four architectural decisions are presented, respectively in Tables 11, 12, 13 and 14. These tables present important concepts about ADD considering Decision, Constraint, Solution and Rationale and Quality Attributes.

5.2.1 Architectural Design Decision D01 - SOA

Table 11 presents Decision D01 that represents the first architectural decision defined for the EHR application. This decision is related to the need of communication between legacy systems currently on execution in the hospital.

Decision D01: Develop an EHR application gathering data from systems	
Issue	Currently, important patients' data are spread in a number of legacy systems, and there is not an EHR application where these data are centralized.
Decision	Develop a new EHR application based on SOA to consume functionalities through services from involved systems.
Constraints	Only open source tools should be used for development.
Solution	Define open source tools to development. Transform legacy systems to expose services. Develop a new EHR application to consume from these services.
Rationale	SOA proposes many advantages compared to other approaches for distributed applications. In addition, several tools for SOA development can be found.
Quality attributes	Interoperability, Modularity, Reusability, Modifiability, Testability, Maturity, Availability, Time Behavior, Resource Utilization.

Table 11 – Architectural Decision D01 for EHR application using SOA

Decision D01 is defined to choose SOA as the approach to develop the EHR application. SOA applications provide reuse of legacy systems through services that represent the core functionalities of legacy systems. Service oriented computing is not the first

proposed paradigm for systems integration. Several distributed solutions have been proposed in past decades, such as integration point to point and middleware. However, these approaches were often considered to be complex and expensive (ERL, 2005). In addition, SOA has been widely adopted in academia and industry. This set of factors contributed for definition of D01: to use SOA for developing the EHR application. One constraint presented in this EHR project is the use of open-source development tools due to low budget for information technology in a public hospital.

Modularity is a quality attribute that defines the capacity of a system to be composed of components such that a change to one component has minimal impact on other components. This definition is similar to a service concept. In a typical service-based scenario, a service can expose an important system functionality that can be used by another system. An implementation of a service provider hosts a software module accessible in a network (PAPAZOGLU et al., 2008). Thus, it is possible to notice that using SOA to develop the EHR application favors the increase of system modularity.

Reusability is a quality attribute which indicates that an asset can be used in more than one system, or in building other assets. Service reusability is a SOA design principle that means services are reusable (ERL, 2007). Logic encapsulated by services is generic enough to allow many usage scenarios and can be used for different types of service consumers. A system functionality can be exposed by a service and, thus, other systems can use this service to consume some relevant information. Reusability is an important concept to SOA because to promote service reuse is one of the main benefits of using SOA.

Modifiability is a quality attribute related to the capacity of software to be modified without introducing defects or degrading existing product quality. Modifiability increases as the independence between modules of the system increases because a change in one module can affect all modules that are dependent. By definition, services are loosely coupled (PAPAZOGLU et al., 2007). This means there are few well-known dependencies between services. Thus, the cost of modifying the implementation of services is reduced and the overall system modifiability increases (O'BRIEN; MERSON; BASS, 2007).

Testability is a quality attribute found in a system or component in which test criteria can be established and tests can be performed to determine whether those criteria have been met. According to O'Brien (O'BRIEN; MERSON; BASS, 2007), testing a system based on SOA is more complex than an isolated software. For instance, if a runtime problem occurs, it may be difficult to find the cause, as it can be spread in the service user, the service provider, the communication infrastructure, the discovery agent (UDDI), or it can be due to the load on the platform where the service executes. Trying to replicate problems in a test environment may be difficult as well or even not possible within a short time frame.

Maturity and Availability are two important quality attributes related to Reliabil-

ity. Maturity is the degree to which a system meets needs for reliability under normal operation. According to SOA concepts, whenever a service consumer requests some information, it is expected that a response is returned. Availability is the degree to which a system, product or component is accessible when required for use. In SOA context, services must be available when they are requested. Maturity and Availability attributes may be affected by some external factors. For instance, when a server is unavailable, then services become inaccessible affecting system maturity and availability.

Regarding SOA performance efficiency, two quality attributes should be considered, Time Behavior and Resource Utilization. Time behaviour analyzes the response and processing times and throughput rates of a system, when performing its functions. In SOA context, this is the time spent by a service to process a request and return a response. Resource utilization refers to amounts and types of resources used by a product or system when performing its functions. Services use resources such as servers to access information of other applications. Some authors warn of SOA adoption about some issues related with performance. In (O'BRIEN; BREBNER; GRAY, 2008), the authors propose alternatives to address these in early stages in the development lifecycle. They proposed a SOA performance modeling method and tool support to transform the SOA model into a performance model. These alternatives to improve SOA application performance are considered in the EHR project.

5.2.2 Architectural Design Decision D02 - ESB

The second Architectural Decision (D02), in Table 12, addresses a decision about using Enterprise Service Bus (ESB) in the SOA application project. ESB is an infrastructure software that is responsible for establishing the message flow bus between consumers and service providers. ESB will provide the management and mediation between systems that consume services and those who provide. In addition, ESB contributes to integration aspects such as orchestration, mapping, routing and composition.

Decision D02: Develop an EHR application that uses an ESB	
Issue	Enterprise Server Bus (ESB) simplifies the integration and flexible use of business components.
Decision	Develop a new EHR application based on SOA to consume functionalities through services from involved systems.
Constraints	Define an ESB with free license to use.
Solution	Although most used ESBs in industry are commercial versions, there are free ESB solutions.
Rationale	ESB is not mandatory in SOA development but its use is recommend due to its benefits.
Quality attributes	Functional Appropriateness, Co-existence, Operability, Reusability

Table 12 – Architectural Decision D02 for EHR application using SOA

Enterprise Service Bus is not indispensable for SOA application development. However, adopting an ESB in EHR was defined because of well-known advantages such as service mapping, adapters, protocol transformation, message-processing, process choreography, service orchestration, reliable transaction management. A constraint to decision D02 is choosing an ESB with free license. Some quality attributes are influenced by decision D02.

Functional Appropriateness is concerned with the degree to which functions facilitate the accomplishment of specific tasks and objectives. This concept can be found in ESB benefits. ESB can facilitate some operations during SOA development such as message transformation and transport protocol conversion.

Co-existence is the degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product. ESB provides messages routing and this environment is shared with all published services. Different composite services can share the use of same service operations. In addition, servers are resources where legacy systems are executing and service consumers request information.

Operability is the degree to which a product or system has attributes that make it easy to operate and control. Benefits offered by ESB facilitate several operations during SOA development. Within ESB, it is easier to operate and to control tasks such as routing, provisioning, integrity and security of messages as well as service management.

Reusability indicates that an asset can be used in more than one system, or in developing other assets. An ESB service is an instance of a service type (CHAPPEL, 2004). Reusability can be accomplished by instantiating a particular type of service and applying variable data and conditions through parametrization and configuration.

5.2.3 Architectural Design Decision D03 - Mobile devices

Architectural Decision D03 (Table 13) address an important requirement about mobility and flexibility. Decision D03 presents the need of developing an EHR application that can be accessed by computers and mobile devices. Several patients in the hospital are unable to move, thus it is relevant to provide an application that uses current technology to provide mobility.

EHR flexibility can facilitate the execution of several tasks. For example, clinicians can introduce patient evolution when visiting his/her hospital room. Currently, this process of noting patient evolution is performed in paper forms. Another example of using mobile devices is the possibility of a patient to see his/her X-rays on a tablet device while the clinician is explaining the results.

Functional Appropriateness is the degree to which functions facilitate the accomplishment of specified tasks and objectives. The EHR application has been developed to provide access from a complete set of devices. Clinicians and patients can access through

Decision D03: Develop an EHR application that can be accessed by mobile devices	
Issue	Currently, patients information are available only in paper forms. The EHR application will maintain the records and evolution of patients
Decision	Develop a multi-channel system. EHR application can be accessed by computers and mobile devices such as tablets and smartphones to promote system flexibility and mobility by users.
Constraints	None.
Solution	Develop a new EHR application to be executed in different devices to consume services.
Rationale	EHR application supports distribution of content to multiple devices and channels so that the front-end layer will be different for each device but all devices will consume the same services.
Quality attributes	Functional appropriateness, Operability, Reusability.

Table 13 – Architectural Decision D03 for EHR application using SOA

desktops, laptops, tablet and smartphones. This mobility facilitates accomplishment of specified tasks such as providing patient information for application when a clinician visits a patient.

Operability refers to the degree to which a product or system have attributes that make it easy to operate and control. From the point of view of system implementation, operability can be understood as the degree of ease of using a service. SOA allows service operability through WSDL documents that allow exchanging messages between services. Moreover, it is possible to observe the operability of the system from the viewpoint of a user who uses and interacts with the system. Thus, operability quality attribute can be analyzed as the easiness of using the system. The objective of creating an EHR application for different types of devices is to facilitate system operability.

Reusability is a quality attribute positively impacted by D03. EHR application will be supported by some devices such as computers, laptops, tablets and smartphones. This decision was taken to facilitate system access by clinicians and patients. The front-layer is different for each device, but the service is the same. SOA allows reuse of services from different channels. EHR application makes requests to a service and the functionalities exposed by services will be consumed.

5.2.4 Architectural Design Decision D04 - Security

Architectural Decision D04 (Table 14) address an important requirement about security. The EHR application development should be guided based on SOA security requirements.

Authenticity refers to a claim and identification of a subject or resource that requests access to a certain information. This is an important concern in EHR systems as data stored in hospital systems should be protected. Data can only be accessed and modified by authorized people. Therefore, user authentication will be required when someone tries to use the EHR application.

Decision D04: Develop an EHR application that provides security	
Issue	Develop a new EHR application based on SOA security requirements.
Decision	Security requirements involving authentication, information confidentiality, and cryptographic need to be implemented in EHR.
Constraints	None.
Solution	Security requirements must be addressed in software development to protect information from unauthorized people and avoid losing information.
Rationale	Information security will be addressed following two approaches. First one is responsible for defining strategies for protecting the access of unauthorized persons information. The other one is responsible for avoiding loss of information such as in situations that require transaction management.
Quality attributes	Authenticity, Confidentiality, Integrity, Fault tolerance, Recoverability, Accountability.

Table 14 – Architectural Decision D04 for EHR application using SOA

Confidentiality is the degree to which a system ensures data are accessible only to those authorized to have access. This is an important topic that should be addressed in EHR systems. Most patient records are confidential and can not be disclosed to other people including other clinicians. Some patient information should only be accessed by clinicians in the same specialty or other authorized. Therefore, EHR should control the information available to each user profile.

Integrity refers to the degree to which a system, product or component prevents unauthorized access to, or modification of, software or data. This concern can be achieved by controlling access through login and user profile identification. In addition, it can be established by continuous monitoring of access and modification logs. Another security strategy that can be adopted is cryptography. Patient information can be stored using cryptography techniques.

Accountability refers to the degree to which actions of an entity can be traced uniquely to the entity. Services are autonomous, thus services have a contract that expresses a well-defined functional threshold which should not involve other services. In addition, Thomas Erl (ERL, 2007) defined Service Autonomy as a design principle. Service Autonomy refers to the ability to govern itself. When something is autonomous, it has freedom and control to make their own decisions without the need for external validation or approval. Accountability in the SOA context refers to the service state of being accountable, in other words, a service has the obligation of render an account for failure to perform as expected.

Fault tolerance is the degree to which a system operates as intended despite the presence of hardware or software faults. Services can create strategies that may be performed when a failure happens on some hardware or software. Some studies in the literature

proposes alternatives to address fault tolerance in SOA applications. Study (GADGIL et al., 2007) suggests strategies to implement fault-tolerance in a SOA system that are replication and checkpointing.

Recoverability is the degree to which, in the event of an interruption or a failure, a product or system can recover data directly affected and re-establish the desired state of the system. In SOA context, this means service ability to recover data when occurs some interruption or failure. Some ESBs supports transaction management. When a transaction fails, the ESB rolls back the operations within the transaction so that no part results in partial completion.

5.3 Comparison with related works

Recently, literature provided a systematic mapping (TOFAN et al., 2014) addressing past and future of ADD observing the past ten years. This systematic mapping found 144 relevant studies to present an overview of the state of research on architectural decisions. Only 7 out of 144 papers explicitly treat quality attributes and had focus on specific quality attributes such as security, reliability, usability, and scalability. As a result, it is clear that architectural decisions should be considered in initial stages of software development. However, a small number of studies (only 5% of papers) in the literature are concerned with ADD and quality attributes. In addition, this problem extends to SOA context since none of the studies addressed ADD and quality attributes for SOA applications. Therefore, the purpose in this chapter is to analyse and create ADD to achieve quality attributes in SOA applications.

In (GU; LAGO, 2008), the authors defined two architectural decisions addressing service discovery and service monitoring performance. This study was proposed for a generic corporative system. The focus of this paper is to point out a gap in the existing architectural knowledge models for modeling SOA process decisions. Differently, this thesis uses a case study to describe architectural decisions to satisfy a set of quality attributes considered essentials to SOA applications.

In (ZIMMERMANN et al., 2007), the authors defined architectural decisions for two SOA applications. These studies proposed decision meta-models and guidelines for formulating and documenting design decisions, and illustrated their methods in the context of the SOA design space. As for conclusion, the authors could observe that the use of ADD provided quality improvements on projects. However, this study did not detail what were the achieved quality improvements. The contribution of the work presented in (ZIMMERMANN et al., 2007) with respect to quality of SOA application was not detailed. This proposed study addressed another aspect considering architectural decisions and their influences in quality attributes. Each ADD of the case study could affect positively or negatively and therefore should detail the possible impacts and solutions for

the project.

The studies proposed in the literature are not enough to describe how architectural decisions can be used to developing SOA applications according to important quality attributes. Within this thesis, a case in the health domain is proposed in order to address this gap. In Section 4.2 has presented a set of ADD and their influence in quality attributes. All attributes that are considered fundamental for SOA applications according to the SOAQM model are considered.

Development of SOA applications should be performed with great care and planning otherwise the benefits promised by SOA will not be successful. Quality issues should be considered as one of main concerns. This work described an analysis of the influence of quality attributes in architectural design decisions for SOA applications. Thus, architectural decisions are used to guarantee quality in early stages of SOA application development. A SOA EHR application is presented as case study applied to automate business process in a public hospital. A set of architectural design decisions to achieve SOA quality attributes is proposed. The aim of meeting all quality attributes guided the definition of architectural decisions.

SOA quality has been addressed in many studies in the literature. There are several quality models specific for SOA applications. A wide range of quality attributes, metrics suite and tools have been developed to measure quality in SOA applications. This large amount of SOA quality studies indicates a concern to provide systems that satisfy customer requirements and also has high levels of quality.

5.4 Discussion

18 out of 20 important SOA quality characteristics are considered in the cited architectural decisions, according to SOAQM quality model. Two quality attributes are not directly cited because they are related with the whole project, that is, Functional Completeness and Functional Correctness.

All ADD taken in this software project affect an important quality attribute that is Functional completeness. Functional Completeness means the degree to which the set of functions covers all the specified tasks and user objectives. Observing from the SOA perspective, services must cover all the specified tasks and user objectives for which they were designed.

Another important quality attribute affected by all ADD is Functional Correctness. Functional Correctness means the degree to which a product or system provides correct results with the needed degree of precision. This characteristic can be applied in SOA because services should provide the correct response with the needed degree of precision. A service requests some information and a service provider is responsible to send the correct response.

Some quality attributes do not have total warranty that they will be completely satisfied because of possible influence of other factors. For instance, reusability of a service can be proportioned by SOA, but it depends on the service granularity. Therefore, a service with a huge scope is at risk of not being reused. Another example is interoperability that is another quality granted by adoption of SOA. This is because the architecture is based on services and services are interoperable. Therefore, it is possible to exchange data between different systems and interoperability is achieved. However, this ideal scenario may be affected by serious interoperability deficiencies (KASUNIC, 2004).

In order to analyze these problems that are not based on architectural decisions, it is possible to set some strategies. During implementation, several testing cycles have been developed. Literature provides studies about how to measure quality aspects. Study (ALDRIS et al., 2013) proposed a method for measuring the degree of service orientation of SOA systems. This study was proposed to measure how interoperable, adaptable and reusable a service is. The authors propose a framework called DoSO Framework that defines a metric suite using an automated static code analysis.

The set of four Architectural Design Decision have defined four fundamental concepts for the EHR including approach for integration, bus, mobile and security. These concepts represent four fundamental needs in the hospital. For each need, it was analysed which solution could provide resources that focus on quality attributes. For example, first ADD proposed for EHR addresses the need of system integration and there are a set of options in this case such as point to point integration, EAI platform and SOA. As result of an analysis focus on quality, D01 was defined to apply SOA paradigm in EHR due to its advantages in interoperability and reusability. The others ADDs proposed for the EHR application followed this same approach: for each fundamental need in hospital, one solution was defined based on SOAQM essentials quality attributes.

EHR Architectural Views

The content of this chapter is mainly based on two papers. First paper entitled “Layered Implementation View of a SOA Based Electronic Health Record” (LIMA et al., 2016). This paper was published on the proceedings of the International Conference on Software Engineering and Knowledge Engineering (SEKE) in 2016. Second paper was entitled “Development of an Electronic Health Record Application using a Multiple View Service Oriented Architecture” (FRANÇA; LIMA; SOARES, 2017). This paper was published on the proceedings of the International Conference on Enterprise Information Systems (ICEIS) in 2017.

This chapter presents the architecture views proposed to develop the EHR application of a public hospital at Federal University of Sergipe, Aracaju - Brazil.

6.1 Hospital Stakeholders’

The EHR system has been developed due to a demand from a public hospital as mentioned in Chapter 5. Three main constraints are considered in this EHR project. Firstly, low budget for Information and Communication Technology in a public hospital. It is not possible to start from scratch and discard all deployed legacy systems. Second, lack of integration between current legacy systems. Finally, important business rules must be implemented in the EHR, asked by many stakeholders.

Due to the constraints presented before, the proposed solution is to use Service Oriented Architecture principles and techniques to gather all important information from legacy systems on services to be consumed by the EHR application.

Requirements proposed to the EHR system were analyzed to propose a solution using SOA. Solution proposed for EHR was presented to hospital stakeholders through architectural views.

Most studies in literature address the implementation of an EHR using SOA concepts with focus on the application itself, but not on the architectural aspects for development. Even those studies which present the software architecture, only describes one or

two views, or even a multiple number of views in only one box diagram, which may be confusing for most stakeholders.

Stakeholders in hospital context includes software developers, directors and health professionals such as doctors and nurses. Stakeholders should understand the EHR solution to provide their opinion and approval. However, in some cases, stakeholders are not IT professionals and because of this, the solution should present information with ease understanding. Therefore, it is extremely important to propose architectural views to cover different concerns and perspective of EHR system.

6.2 Architectural Views

Architecture Views proposed as follows express the architecture of EHR system from the perspective of specific system concerns. The multiple views architecture is important to describe the different aspects of the architecture. This set of views is useful to describe the entire system for different stakeholders, including doctors, nurses, hospital managers and software developers, from different perspectives. Parts of the most representative views are presented as follows.

6.2.1 Scenarios View

One part of the EHR functionalities is presented in Fig. 21. The EHR developed in this research is able to schedule medical appointments, register appointments, store patient records, visualize entire patient records history, present results for patients' exams, and checking recommendations prescribed by doctors or health professionals.

Several requirements and Use Cases were documented in this EHR project but the main requirements is listed as follows. This set of Use Cases can provide a glimpse of how the EHR improves automation of hospital business processes.

UC1 Authentication

UC1.1 User login.

UC1.2 User logout.

UC1.3 Password recovery.

UC2 Insert electronic medical records about patient.

UC3 View electronic medical records.

UC4 View patient exam results.

UC5 View patient disease historic.

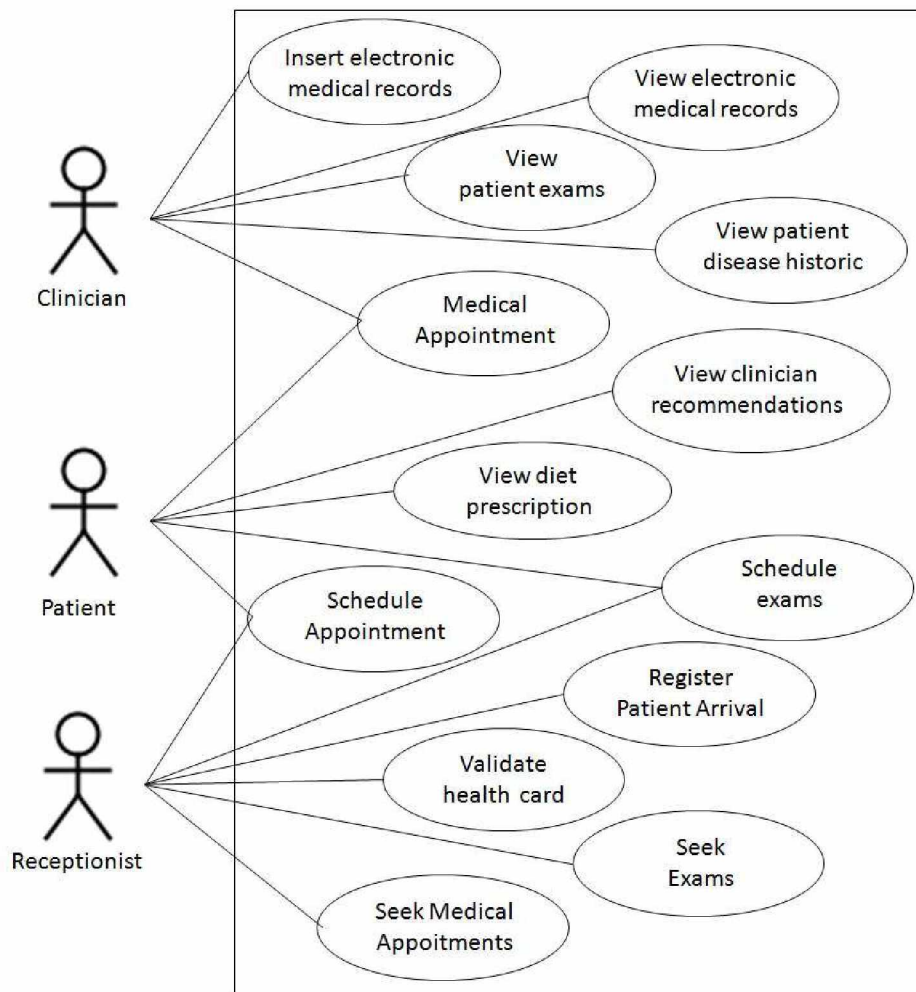


Figure 21 – UML Use Case diagram

UC6 Schedule medical appointments.

UC7 Seek medical appointments scheduled for specific dates.

UC8 Schedule exams.

UC9 Seek exams scheduled for specific dates.

UC10 Validate health card.

UC11 View patient diet prescription.

UC12 View clinician recommendations.

UC13 Register patient arrival.

UC14 Perform medical appointment.

All use cases presented in Fig. 21 can be performed after UC1 - Authentication. However, UC1 included in whole diagram produces a tangle of arrow that can make it difficult to understand.

6.2.2 Business Process View

Some business processes are identified in the hospital as follows.

1. Register Patient and Schedule Appointment in the Clinic;
2. Perform Appointment and Manage Evolving Documentation;
3. Perform Schedule Interappointments and Exams;
4. Perform Multiprofessional Team Accompaniment;
5. Perform Non-Medical Treatment;
6. Schedule Surgery;
7. Patient Hospitalization;
8. Perform Surgery;
9. Reassess and Patient Discharge.

The main focus of the EHR application is to automate business process related to medical record and appointment management. Therefore, business process “2- Perform Appointment and Manage Evolving Documentation” is presented in Annex A and it is detailed in the next paragraphs.

Fig. 22 depicts an Activity diagram which presents the process for a medical appointment in the hospital. This process is detailed in the next paragraphs.

In most cases, appointments are scheduled by patients through the Basic Health Unit (BHU) located near their residence sector. Each patient has one BHU near to his/her house to request assistance when medical care is needed. Some basic health services are solved by clinicians in the BHU. Other more complicated health situations are forwarded to the public hospital.

On the appointment’s day, patient is attended in the hospital by an attendant at reception. The attendant is a hospital employee who has login to access the system and register the patient’s arrival. Hospital attendant requests the number of the patient’s Health Card, and then search in the EHR patient’s registration that contains date of appointment and medical specialty.

Patients are consulted by the physician and then the appointment is performed. Physician reports patient records including clinical status and medicines prescription. Finally, the physician describes exams and new appointments.

This process is responsible for scheduling and managing appointments. This is a small part of the EHR scope and, currently, it is required to access five different legacy systems to perform this task. In addition to this, patient record is manually written which makes

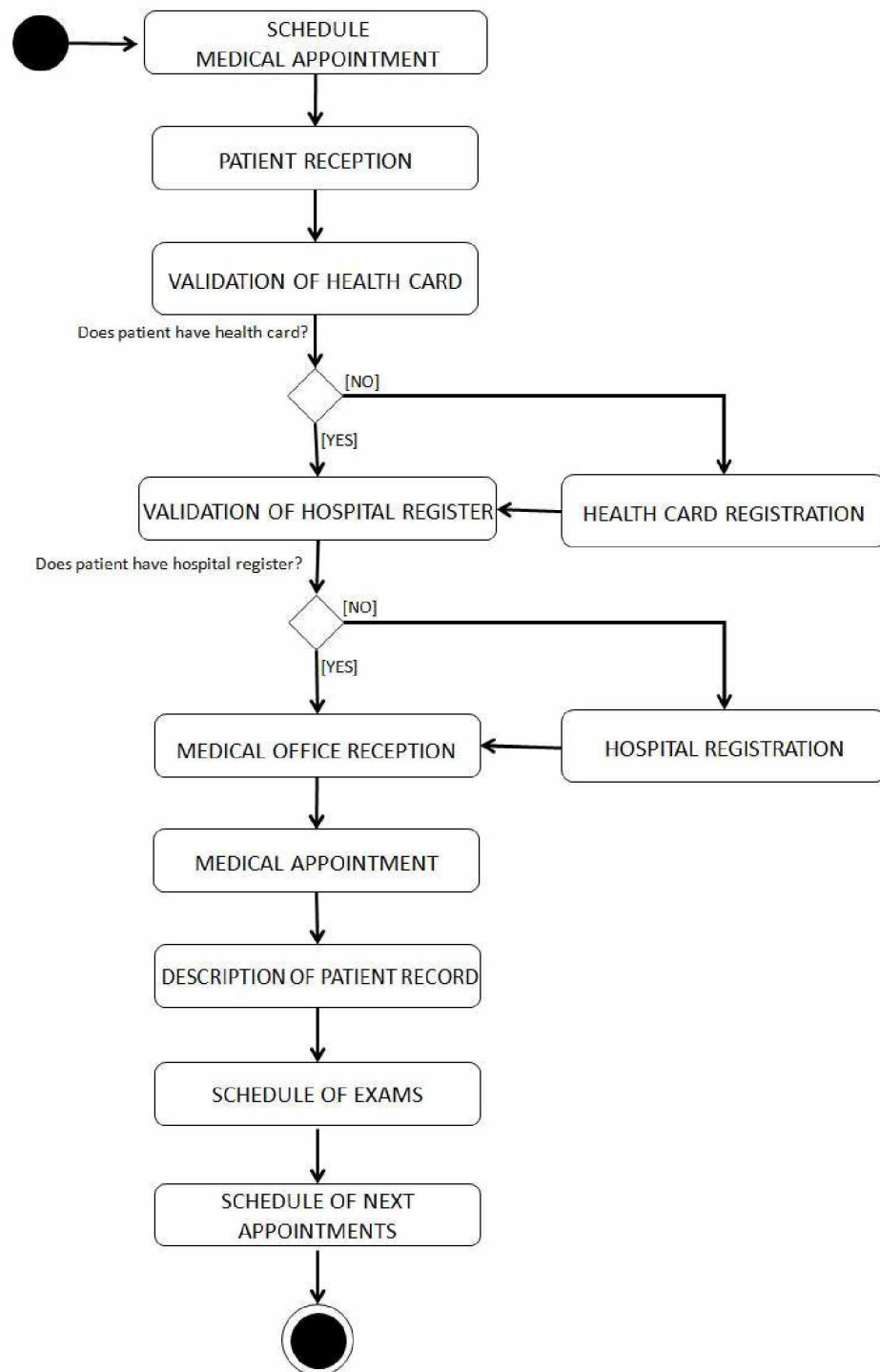


Figure 22 – UML Activity diagram - Medical appointment process

it difficult further access by doctors. The developed EHR centralize this process, and also makes it possible to use only one system because the other systems are integrated by web services. Another advantage provided by the EHR is the fact that all information about patient health is stored in the system, which facilitates medical diagnostic.

6.2.3 Implementation View

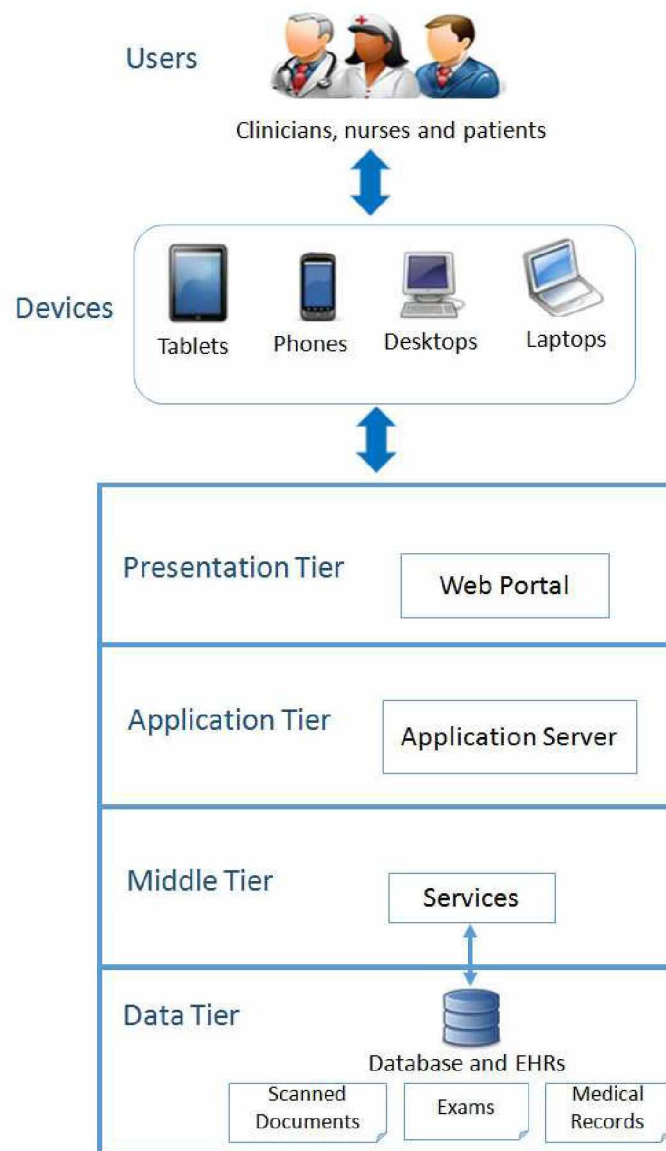


Figure 23 – EHR Implementation View

The EHR implementation view of the architecture proposed in Fig. 23 is designed as a multi-layered style. First layer, named Presentation, is about presentation components that provide or receive information to users. A web portal is developed to represent this first layer. According to previous decisions, as the EHR application can be executed on different devices, this layer must be independent from the lower layers.

Application layer is composed by an application server where the user information inputs will be collected and processed. This middle layer is composed by the services layer that resolves technical issues for the integration of different applications, solving problems of different data structures, connectivity to different protocols and specific needs of the involved systems. Typically, components transformation, and Routing Adapter are used.

The middle layer is composed by web services. Within Service-Oriented Applications,

it is extremely important to clearly define some settings such as which services should be implemented, which systems are providers and/or consumers, definition of how to establish the interaction between systems and which data types will be supplied in requests and responses.

The data layer abstracts the contact point of the SOA bus with legacy systems. Services providers will supply the EHR application with patient data requested by users.

Services catalogue presents all identified services. These services (Table 15) can be used by the whole organization in their process behavior with the purpose of achieving a service oriented integration.

Table 15 – Service Catalogue

ID	DESCRIPTION	PROVIDER
1	Data for Login	AGHU
2	Create medical appointment	ACONE
3	Validate health card	CADWEB
4	Get patient's data	AGHU
5	List of appointments	MEDLYNX
6	List of exams	MEDLYNX
7	List of medical hospitalization	IMHOTEP
8	List of procedures	AGHU

Legacy systems involved in this project as providers of services are ACONE, CADWEB, AGHU, Medlynx and Imhotep. Each legacy system has important information about patients that will be consumed by the EHR.

Patient registration is performed by ACONE, which maintains also appointments schedule. CADWEB, a national health system contains registration of country citizens. Each registered citizen in CADWEB has a unique health card number, a national identifier that allows free access to public hospitals, exams, appointments, medicines and surgeries. AGHU is the hospital's main legacy system that contains important information about patients. One module of AGHU is responsible for maintaining patient personal data such as name, address, identity documents, and health card number. The other AGHU module controls patient exams, including storing all exams to which patients were submitted, exam results and a list of approved and pending exams. There are yet other modules that are responsible for maintaining history of attendances, hospitalization, surgery, medicines and medical procedures. Medlynx is a legacy system used to manage medical appointments and exams. Imhotep is a legacy system responsible to manage financial data and all information about products at the hospital, including medicines and other health related materials.

Figure 41 presents SoaML Service Architecture diagram that provides a global view of system services. SoaML Service Architecture diagram presents all services that should be implemented, which systems are involved and which consumers and providers interact

with the application. Figure 41 presents eight services, which are represented by an ellipse. Each service represents one functionality of a legacy system that is exposed in an Enterprise Service Bus (ESB) to be consumed by a system consumer, the EHR in this case.

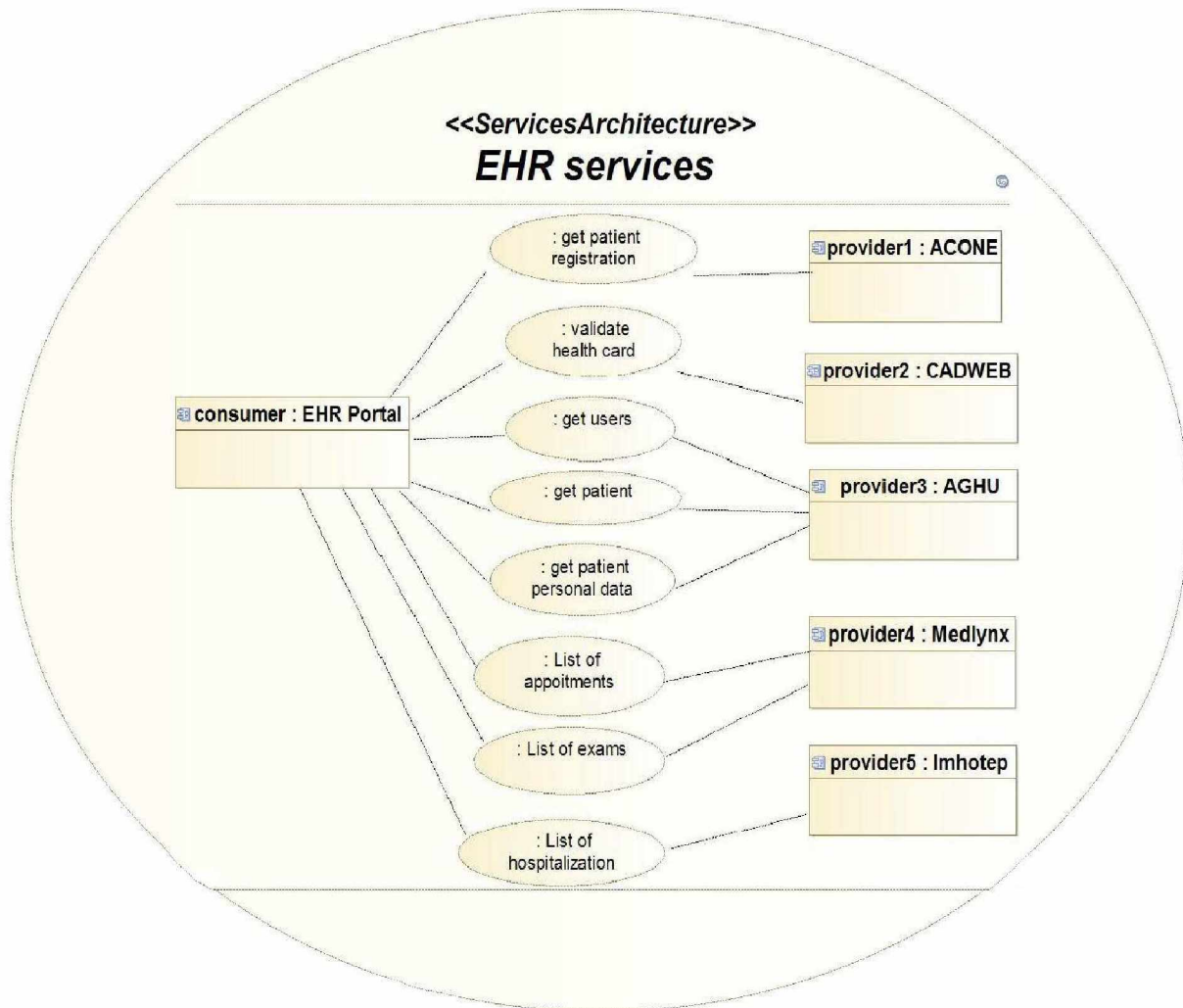


Figure 24 – Service Architecture diagram representing legacy systems

6.2.4 Process View

The Process View of the architecture is composed by flow of services in Enterprise Service Bus. Fig. 25 depicts a UML Package diagram that presents how packages are organized in ESB. Files related to flow of services generated by ESB as well as web services source code are organized in package App.

The flow of services performed by the bus management is presented in Figs. 26 and 27.

Input bus is the single entry point of a message from the services bus. In this initial stage, important tasks, such as security, input log, auditing, transformation, formatting

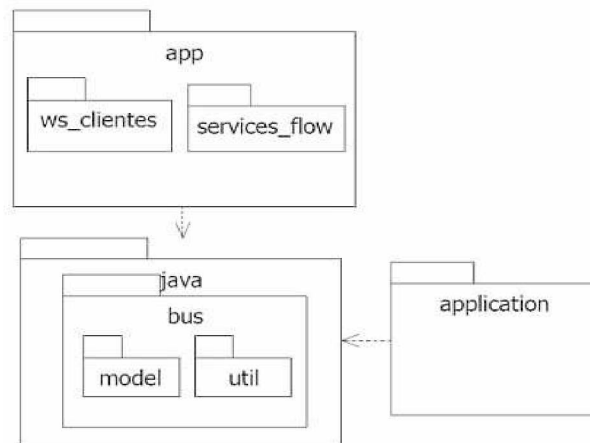


Figure 25 – ESB Package

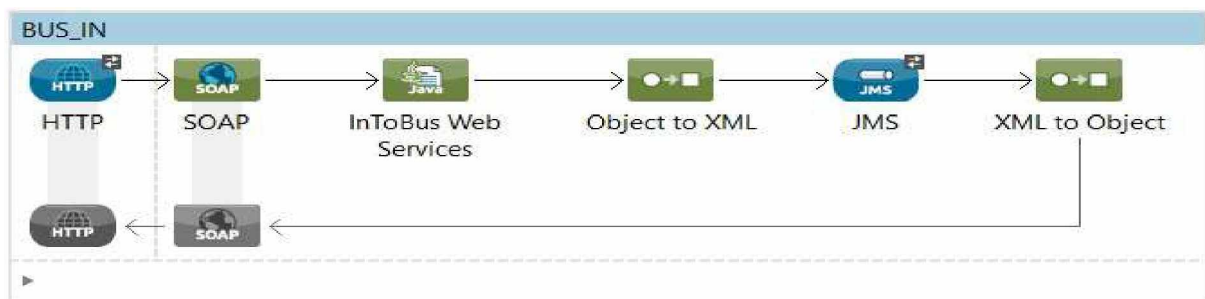


Figure 26 – Bus In

and validation are executed. After these functions, the message flow is routed for the specific service asked by the consumer. As depicted in Fig. 26, Mule receives objects through HTTP and executes all necessary transformations.

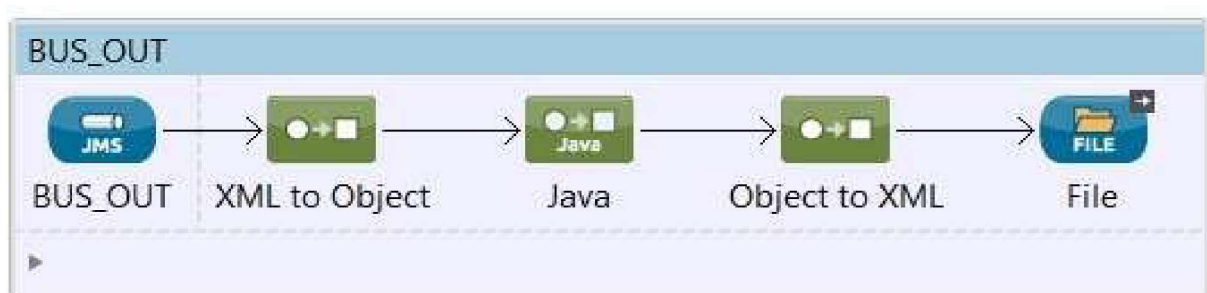


Figure 27 – Bus Out

Output bus is the single output point in the services bus, being responsible for routing responses from the flow of services to the consumers. Some necessary tasks to finalize the message flow in the service are executed here, including log registry, data formatting, data transformation, validations, among others. Output bus implemented in this application is presented in Fig. 27, which makes it clear that the output bus does the inverse operation of the input bus.

Package Java has bus as subpackage, which has bus util and model as subpackages.

Subpackage bus has source code to manipulate services flow as depicted in Figs. 26 and 27. Package model has source code related to entities that will be retrieved by client web services. Package util has source code used to manage services flow. Package applications is the package to store source code related to all applications developed based on this implementation architecture.

6.2.5 Logical View

Logical view is described by means of two different and complementary diagrams, UML Class diagram and SoaML Participant diagram. SoaML diagrams are detailed EHR design in Chapter 7.

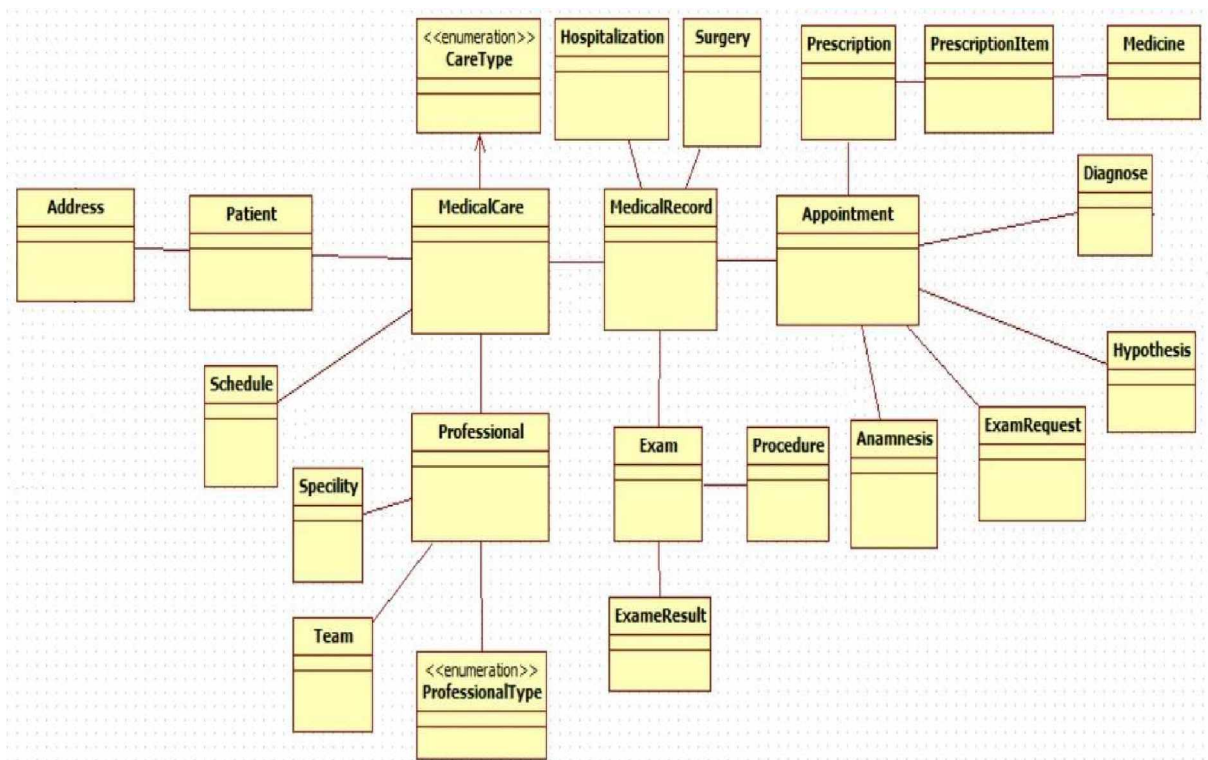


Figure 28 – UML Class Diagram

The UML Class diagram, as depicted in Fig. 28, is applied as a basis for developing the EHR system. This class diagram was used to develop Java classes of the EHR application. An appointment is composed by Diagnostic, DiagnosticHypothesis, Anamnesis and Prescription, which may have many Medicines. An medical care is associated to a patient, a care type and a team which is associated to a set of physicians. An medical record is storage in the EHR application and it has a aggregation relationship with appointment, medical care, surgery, hospitalization and exams. The UML Class diagram including attributes and multiplicity is presented in Annex B.

6.3 Comparison with related works

Literature, in most cases, proposes SOA application in EHR domain without addressing architectural concepts. Most studies address the implementation of an EHR using SOA concepts with focus on the application itself, but not on the architectural aspects for development. Even those studies which present the software architecture, only describes one or two views, or even a multiple number of views in only one box diagram, which may be confusing for most stakeholders.

As presented in Section 2.6, some studies propose the EHR architecture using natural language and some using informal draws. This scenario can represent that there is focus on the application itself. However, in software development process, it is necessary to address architectural elements as well. Some other studies concerned on EHR architecture presents the software architecture by describing only one or two architectural views((BLOBEL, 2011), (CHO et al., 2010), (GAZZARATA; GAZZARATA; GIACOMINI, 2015) and (EL-SAPPAGH; EL-MASRI, 2014)), or even two views described on the same diagram ((MOOR et al., 2015) and (FABIAN; ERMAKOVA; JUNGHANN, 2015)), making it difficult for most stakeholders to understand the architectural models.

Contrary to most other works found in the literature, this study proposed the EHR architecture using multiple views. Five architectural views, Scenario, Business Process, Logical, Process and Implementation, are presented to provide an overview of the development of a SOA application in the health domain. Each view presents different elements, improving separation of concerns. Software architecture is documented in multiple views in order to help stakeholders to better understand the future software system. In addition, architecture description using multiple views allows managing complexity and risks during software development.

6.4 Discussion

This study describes part of the EHR architecture, which is applied to gather requirements by hospital stakeholders such as doctors, nurses, receptionist and health professionals in general. Architecture views presented are important because, in general, hospital stakeholders are not experts in information technology technical concepts. Thereby, scenarios and business process views are fundamental to facilitate understanding of future system functionalities through diagrams such as Use Cases and Activity diagrams.

The proposed EHR architecture has been crucial for software developers. They clearly could understand through multiple scenarios, business processes, layers, and architectural decisions what should be implemented. The development process was evaluated by a set of IT experts and this evaluation is presented in Chapter 9. As result of this evaluation, volunteers' opinion indicate that multiple views are important to development process.

Documenting software architecture facilitates the development but also further maintenance. As a conclusion of this case study, after the experience of development of the EHR application using SOA, it is clear that investing time and effort documenting the software architecture is beneficial to the system development process.

The software architecture supports future activities, including software development and integration of legacy systems, besides having a close relationship with requirements asked by stakeholders.

EHR Design

The content of this chapter is mainly based on two papers. One is entitled “An Experience of using SoaML for Modeling a Service-Oriented Architecture for Health Information Systems” (SILVA et al., 2015). This paper was published on the Proceedings of the International Conference on Enterprise Information Systems (ICEIS) in 2015. The other paper entitled “A Case Study on SoaML to Design an Electronic Health Record Application Considering Integration of Legacy Systems” (FRANÇA; LIMA; SOARES, 2016). This paper was published on the Proceedings of the Computer Software and Applications Conference (COMPSAC) in 2016.

The SOAQM model was also applied to define the EHR design. Two quality attributes considered essential by SOAQM are important for software design: Functional Completeness and Functional Correctness. According to SOAQM, Functional Completeness definition address completeness and covers all the specified tasks and user objectives. Therefore, services must cover all the specified tasks and user objectives for which they were designed. Functional Correctness is related to the system correct results with the needed degree of precision. Therefore, service requests some information and a service provider is responsible to send the correct response. These two quality attributes guided the EHR design to establish agreement between design and user objectives. EHR design is presented in this chapter using SoaML as modeling language.

7.1 Detailed Design of the EHR

One part of the EHR scope that is being developed will be responsible for centralizing and controlling appointments. Actually, appointments management requires that a hospital employee has to access several systems. Currently, in the hospital, there are five software systems for different purposes which are AGHU, ACONE, CADWEB, Medlynx and Imhotep. These five legacy systems, including software to automate exams, to provide primary care, and also to register patient data. However, these modules do not interact with each other in an affordable way. Thus, data needs to be duplicated in two

or more different systems.

The EHR application provides an important requirement that enables appointment scheduling using only EHR. Therefore, the other systems will be accessed by EHR through web services.

Other more complicated problems are forwarded to the public hospital. Currently, patients' information is registered in the patient medical record and maintained as hard copies. The large amount of paper occupies several rooms and it promotes difficulty in accessing patient's records when clinicians need to consult past files to analyze patient evolution. The EHR application storages patient information therefore the access to past files of patient is faster.

7.2 SoaML diagrams for EHR

Within Service-Oriented applications, it is important to clearly define some settings such as which services should be implemented, which systems are providers and/or consumers, definition of how to establish the interaction between systems and which data types will be supplied in requests and responses. Therefore, the EHR was designed using SoaML that is a modeling language specific for SOA systems as mentioned before.

There are some other modeling languages for SOA design such as SOA Ontology (Open Group, 2010), SOMF (BELL, 2010), SOA-RFA (OASIS, 2012), SOA-RM (Brown, P. F. and Hamilton, R. M. B. A., 2006), PIM4SOA (BENGURIA et al., 2007) and SOMA (AR-SANJANI et al., 2008). However, SoaML was chosen following the recommendations of a Literature Review proposed in (MOHAMMADI; MUKHTAR, 2013) that compared these seven model approaches and concluded that SoaML allows detailed modeling of services. Another reason for choosing SoaML is that the language is the only approach which is a UML extension. This may represent an interesting advantage over other modeling languages because UML has been widely adopted in software projects in many domains in both industry and academia. In addition, UML is widely supported by tools.

7.2.1 Participant Diagram

Participant Diagram is a SoaML diagram that presents project participants. A participant represents logical or real people or organizational units that participate in services architectures and/or business processes. Participants are marked with a UML Class stereotype **« Participant »**.

Legacy systems involved in this project are ACONE, CADWED, AGHU, Medlynx and Imhotep. Each legacy system has important information about patients that will be consumed by the EHR. Participant Diagram that shows all involved systems is depicted in Fig. 29.



Figure 29 – Participant Diagram

CADWEB is a national health system. Country citizens are registered by the government to control the population health. Each citizen registered in CADWEB has a unique health card number. Health card is a national identifier that allows free access to public hospitals, appointments, exams, surgeries and medicines.

ACONE is a system used in Basic Health Unit (BHU) to patient registration. Some patients have health problems that cannot be treated by doctors in the BHU. They are forwarded for appointments at hospitals. ACONe maintain patient registration and allows appointment schedule in hospitals.

AGHU is a hospital legacy system that contains important information about patients. AGHU is divided into modules. One module is responsible for maintaining patient personal data such as name, identity documents, address and health card number. Another AGHU module controls patients exams, including storing all exams to which patients were submitted, exams results and a list of approved and pending exams. Other modules are responsible for maintaining history of attendances, hospitalization, surgery, medicines and medical procedures.

Imhotep is a legacy system responsible to manage financial data and all information about products at the hospital, including medicines and other health related materials.

Medlynx is a legacy system used for health management and it is responsible to manage medical appointments, exams, electronic point and maintenance service order.

The full specification of a participant includes ports for every service contract in which the participant participates within the services architecture. Participant diagram for EHR system presents systems participant showing the use of ports. This diagram was design as big size picture because contains six systems involved with its ports. Therefore, participant diagram was divided into two figures (Fig. 30 and 31) to improve legibility.

Fig. 30 presents EHR as system participant in the SOA project with its ports. Ports classified as **<<RequestPoint>>** are used to represent a system consumer that requests a service. Each port in EHR represents a service that is consumed for get some information from systems providers.

Fig. 31 presents five legacy systems participants with its ports. These legacy sys-

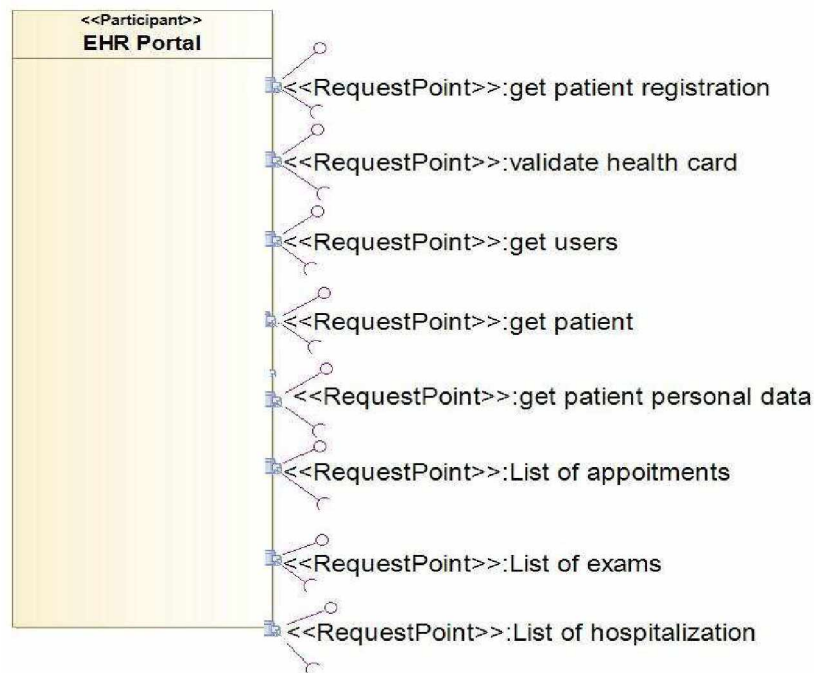


Figure 30 – Participant Diagram with EHR ports

systems are responsible to provide patient information for the EHR application. System providers offer functionalities exposing services and thus this participant type uses **«ServicePoint»** to represent services in participant diagram.

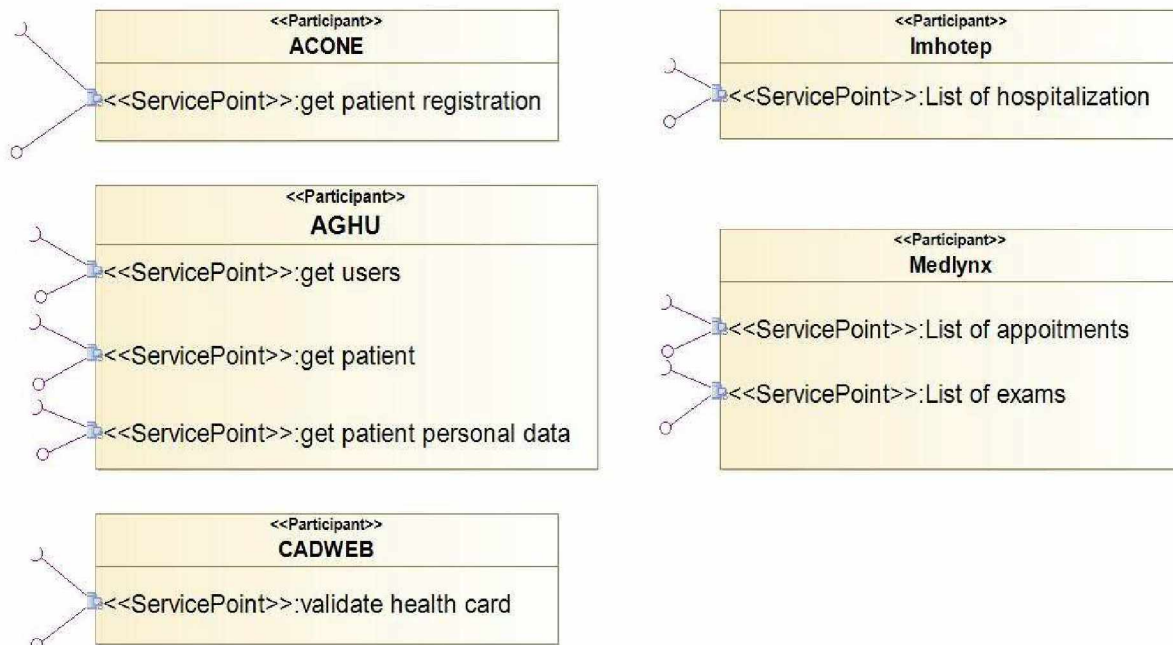


Figure 31 – Participant Diagram with legacy systems ports

Participant diagram also can be represented through interfaces exchange between providers and consumers participants such as depicted in Figure 32. A participant port

presented in Fig. 32 have two types of links: circle and semi-circle. When a port represents a service point, a semi-circle link is like an open hand that receives data in request and the circle link returns data in response. Inversely, when port is a request point a circle link indicates that data is given in request and data is taken in response as shown in the circle link. This alternating structure allows a view of the interaction between providers and consumers participants, as well as data transmitted in requests and responses.

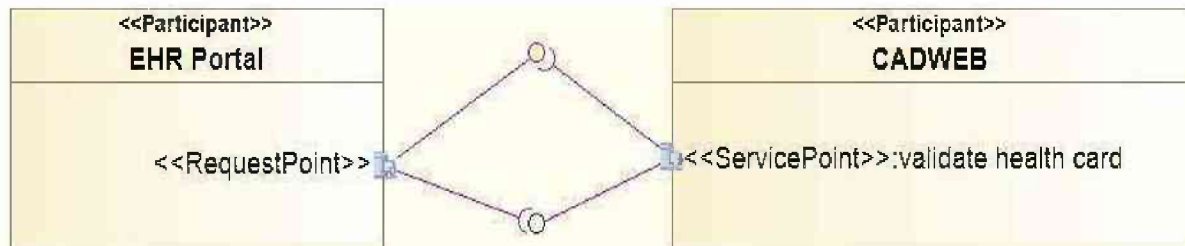


Figure 32 – Participant Diagram with ports, provided and required interfaces

Fig. 32 depicts two systems that can communicate with each other through a service. The EHR is a consumer system that request for a valid health card service exposed by CADWEB. EHR gives a request containing a health card number and CADWEB receives this provided number as request. Then, the valid health card service is executed and provides a confirmation as response that is captured by the EHR.

7.2.2 Service Contract Diagram

Service Contract Diagram for EHR makes explicit what services have to be implemented. This diagram defines which system is the provider and which one is the consumer for each defined service. SoaML Service Contract Diagram for EHR was divided into five figures: Fig. 33, Fig. 34, Fig. 35, Fig. 36 and Fig. 37.

In the Service Contract Diagram, each ellipse represents a service. Figures 33, 34, 35, 36 and 37 present five services, which are represented by stereotype **<< ServiceContract >>**. Each service is described in detail as follows.

Service diagram to **get patient registration** is depicted in Fig. 33. This service will be accessed when a patient, who has just arrived at a Basic Health Unit, requires an appointment at the hospital. ACONE legacy system stores the patients' registration in the BHU. The EHR system, as signaled in diagram 33, will be the consumer system that will make a request to the provider system (ACONE) asking for the patient registration at BHU. The request provides data that uniquely identifies the patient such as the ID card number. Response provided by ACONE returns the patient registration containing the health card number, the appointment date and medical specialty needed by the patient.

Service **validate health card** is depicted in Fig. 34. All patients treated at the hospital must have a health card number registered in CADWEB. To perform this service,

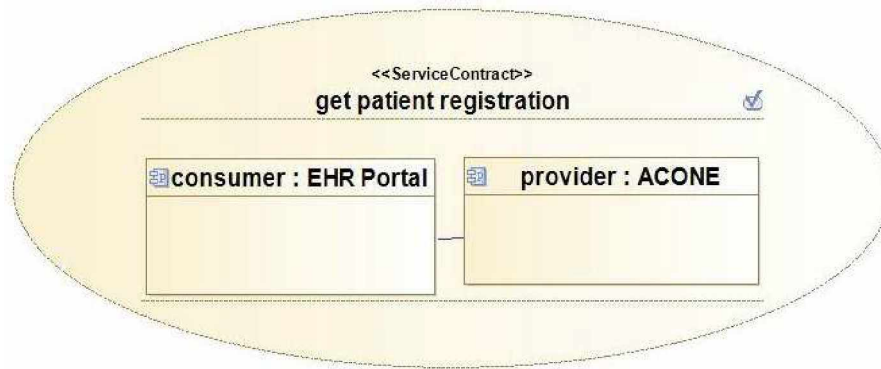


Figure 33 – Service Contract Diagrams - Get patient registration service

the EHR will request a validation to confirm that the patient has health card number in CADWEB. The service request will send the card number reported by the patient and the CADWEB returns a response indicating whether the card is valid.

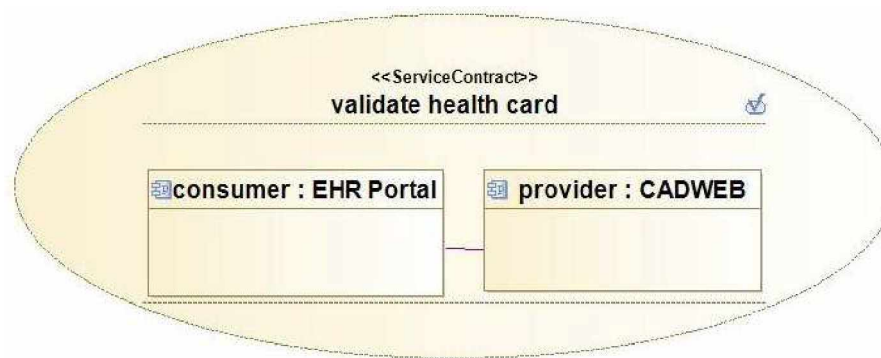


Figure 34 – Service Contract Diagrams - Validate health card service

The **get users** Service diagram can be visualized in Fig. 35. Only authorized users can access the EHR. Users can be doctors, nurses, psychologists, receptionists and other staff working in the hospital. Data about hospital staff is stored in AGHU. When a user attempts to sign in the EHR, the authentication function requests access to the get user service exposed by AGHU provider system. Besides enabling the authentication of users in the EHR, get user service can be used to treat permissions to access the system. As is known, some patient records are confidential and can not be disclosed to other people including other clinicians. Some patient information should only be accessed by clinicians in the same specialty or other authorized. Therefore, EHR should control the information available to each user profile.

Service **get patient** is depicted in Fig. 36. Every patient treated at the hospital must have a registration on AGHU. The get patient service is responsible for making a search on AGHU to verify that the patient is registered in the system. ID card is provided in the request and the registered patient is returned in the response. If the patient is not found then he/she must be registered.

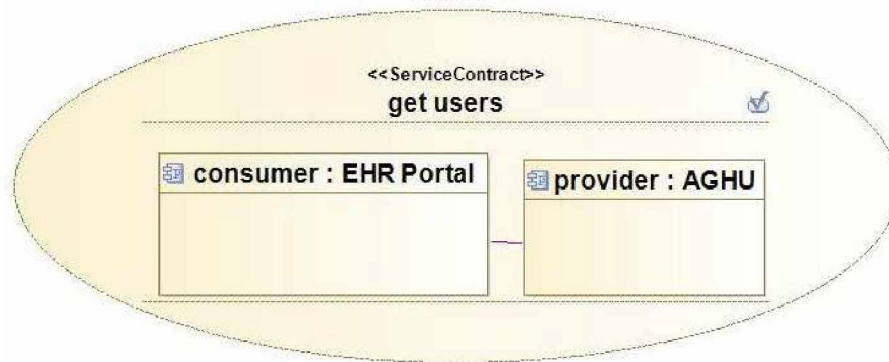


Figure 35 – Service Contract Diagrams - Get users service

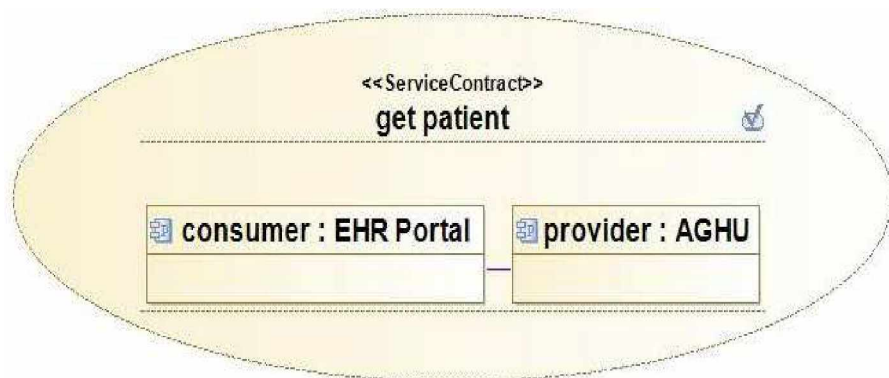


Figure 36 – Service Contract Diagrams - Get patient service

The service diagram for representing **get patient personal data** is depicted in Fig. 37. This service is responsible for providing personal data to compose the patient record. The EHR will be available in the system at the time of the appointment so that the health team can describe the current situation of the patient, symptoms, analyze evolution and prescribe treatments and medicines.

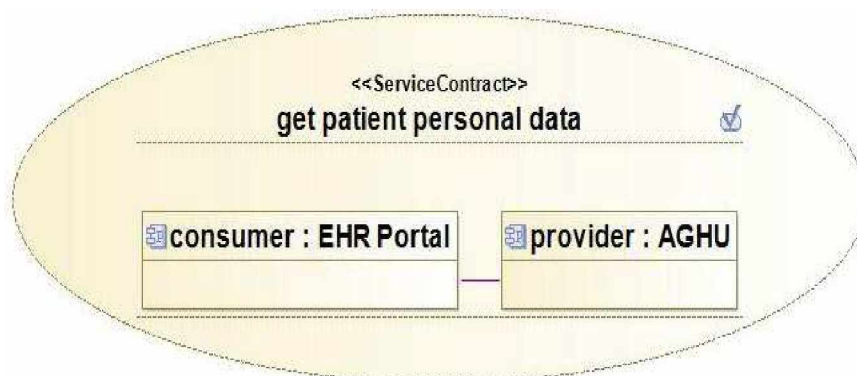


Figure 37 – Service Contract Diagrams - Get patient personal data service

Service **get list of exams** is depicted in Fig. 38. Every patient exam performed at the hospital must have a registration on Medlynx. The get list of exams is responsible for

making a search on Medlynx and return patient exams prescription and exams results.

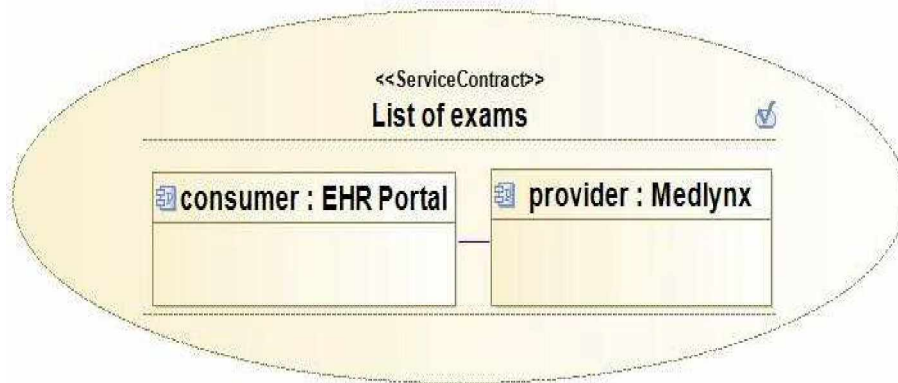


Figure 38 – Service Contract Diagrams - Get list of exams

Service **get list of appointments** is depicted in Fig. 38. Every scheduled appointments at the hospital must have a registration on Medlynx. The get list of appointment is responsible for making a search on Medlynx and return patient appointments scheduled for a specific date.



Figure 39 – Service Contract Diagrams - Get list of appointments

Service **get list of hospitalization** is depicted in Fig. 38. Patient hospitalization at the hospital must have a registration on Imhotep. Details of hospitalization are important information about patient that should be stored as medical record in EHR. The get list of hospitalization is responsible for making a search on Imhotep and return patient hospitalization registered during his life.

7.2.3 Service Architecture Diagram

Service Architecture Diagram for EHR is composed for EHR application, legacy systems and services. Fig. 41 presents the Service Architecture Diagram for EHR. Diagram depicted in Fig. 41 provides a global view of system services. In addition to this, service architecture diagram clearly presents all services that should be implemented, which systems are involved and what are the consumers and providers.

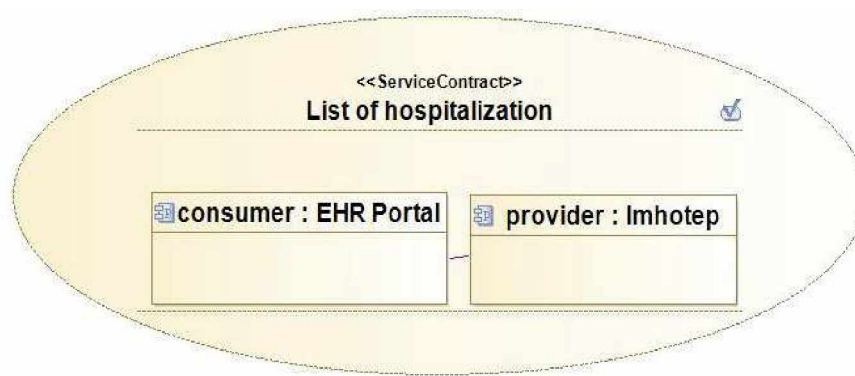


Figure 40 – Service Contract Diagrams - Get list of hospitalization

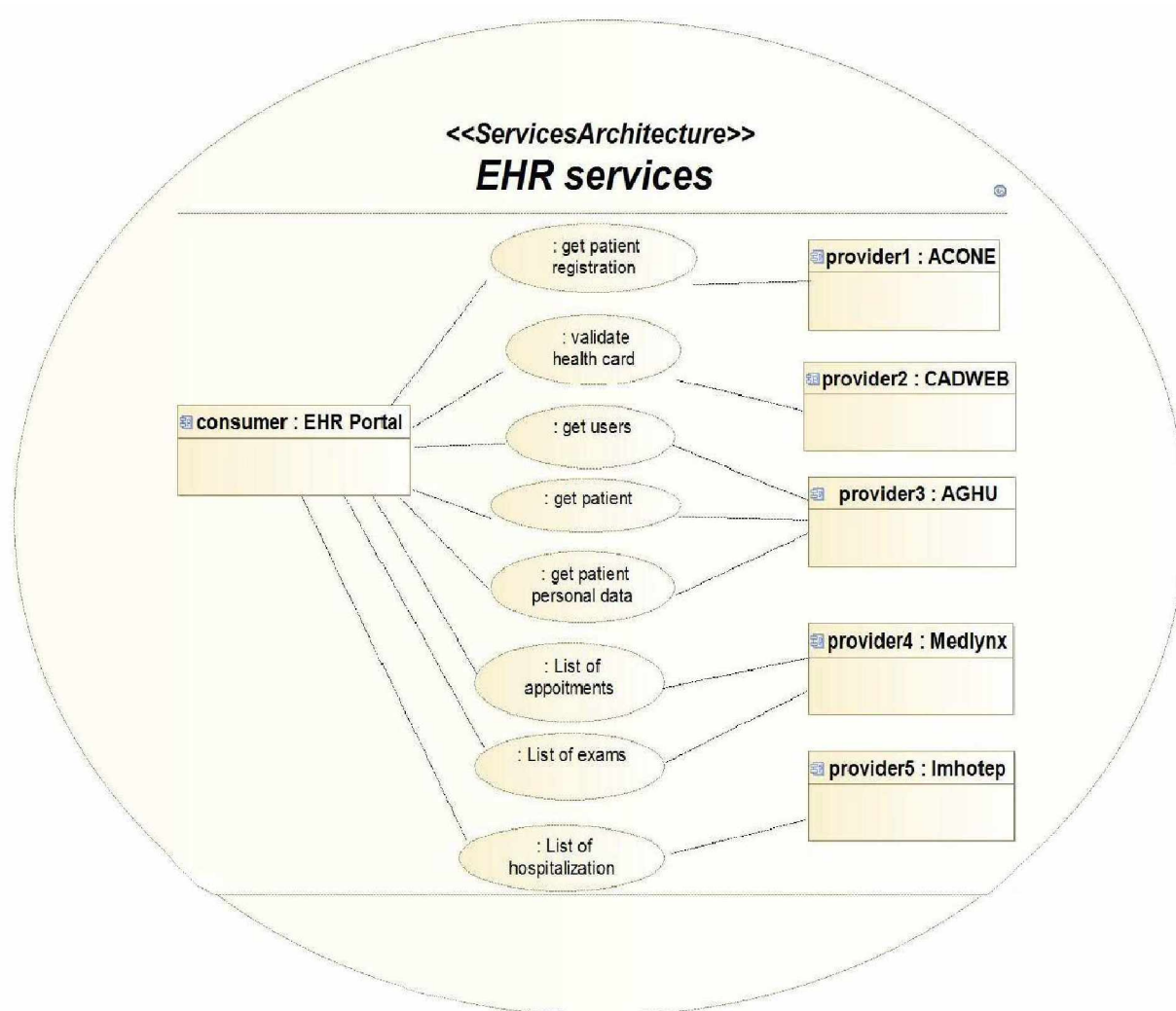


Figure 41 – Service Architecture Diagram for EHR application

EHR system is being developed for consuming several capabilities scattered across multiple systems. Within the service architecture diagrams, it is possible to note that the EHR is a main consumer of data. This important data about patients is exposed by systems providers, including ACONE, CADWEB, AGHU, Medlynx and Imhotep.

7.3 Comparison with related works

This chapter provides a proposal of an EHR design using SoaML. SoaML was chosen as modeling language because the EHR is a SOA based software application, and it is interesting to use a language that can represent services as main concept. Other six modeling approaches for SOA applications were proposed in literature such as SOA Ontology (Open Group, 2010), SOMF (BELL, 2010), SOA-RFA (OASIS, 2012), SOA-RM (Brown, P. F. and Hamilton, R. M. B. A., 2006), PIM4SOA (BENGURIA et al., 2007) and SOMA (ARSANJANI et al., 2008). Although there are some different options of modeling languages, SoaML was chosen based on the following reasons.

First, a recent published Literature Review (MOHAMMADI; MUKHTAR, 2013) compared these seven modelling languages, and as result, they recommend the use of SoaML because it allows detailed modeling of services. Secondly, among the seven options, SoaML is the only approach which is a UML extension. This may represent an interesting advantage over other modeling languages because UML has been widely adopted in software projects in many domains in both industry and academia. In addition, in this case, most designs provided by the legacy systems are UML based. Finally, SoaML has been proposed recently by some studies in different domains. For instance, learning platform domain (GONZÁLEZ et al., 2014), game domain (CARVALHO et al., 2015) and health domain (DELGADO et al., 2011).

As it was mentioned in this thesis, several studies in the literature have used general purpose diagrams such as Data Flows, Flowcharts, UML models including Activity, Class, Component and Sequence diagrams, or even natural language for service analysis. Evidently, literature in software engineering does not recommend the use of natural language to describe system design due to its ambiguity, inconsistency and lack of precision (KAMSTIES, 2005). Regarding the other approaches used in the studies, a brief comparison between our approach is as follows.

Studies that presented Data Flow models provides a view of which data are transmitted between stakeholders. This view is important to understand business process, not services. At the design level, it is important to know which are the relevant data types for a service and how interfaces of request and response are set to transmit data. SoaML diagrams can represent this view using Participant and Service Interface Diagram.

Other studies represent their SOA design using Flowcharts on which it is possible to represent the flow of operations. However, some important details are not represented, such as participant systems and data types transmitted during service execution. This Flowchart limitation is not found in SoaML diagrams.

Activity diagrams present the sequence of activities to achieve some operation. Nevertheless, many other details are required to service development such as definition of message flow and identification of consumer and provider systems. Class diagram represents system classes, their relationships and source code dependencies. In the SOA

context, it is not always possible to know the internal structure of a system. Generally, what is known of a provider system is the functional capacity that it can provide and how to get information in a request. Component diagram divides the system into components according to their responsibility. In the case of a SOA application, some important information is not represented, including services that may not be defined, involved systems are not displayed and transmitted data are not shown. Sequence diagrams present operations flow and systems involved, but have a limitation in defining clearly service scope and data transmitted.

By comparing these approaches with the proposed solution presented in this paper, it is possible to note that SoaML is adequate for modeling SOA systems. Those approaches are not appropriate due to the limitation of not representing services and their interactions explicitly. UML diagrams provide important views to document design decisions and improve understandability of systems functionalities. However, UML is not enough to represent several important details in SOA applications.

7.4 Discussion

This chapter provides SoaML models to design an EHR application that has been developed for a public hospital. The development of this EHR application has been anxiously awaited by hospital employees due to current situation of communication, difficulty of finding health records and the issue of registering data in several legacy systems.

The proposed SoaML diagrams provide graphical representation of services that should be implemented to fulfill all system requirements. SoaML allows new capabilities that the other approaches can not provide such as identifying services as well as the requirements they are intended to fulfill, defining functional capabilities of a service, setting services consumers and providers, presenting the service information exchanged between consumer and providers and establishing policies for using and providing services.

As result of this study, it is possible to note that SoaML is an interesting modeling language for SOA applications. General purpose modeling languages such as UML can not represent services, which are a fundamental part of SOA application design. Although there is no empirical data to support this result, the experience in this study is that those who are familiar with UML do not consider learning SoaML hard.

Chapter 3 in this thesis presented a set of design principles proposed for SOA solution. Eighteen design principles were applied in SOA applications and presented in the studies. Nine of eighteen design principles was not adopted due to lack of precise definition in literature. The others nine design principles were widely adopted in studies that are Service Contract, Service Loose Coupling, Service Abstraction, Service Reusability, Service Autonomy, Statelessness, Service discoverability, Service Composability, and Service Granularity. These design principles were considered in the EHR design as presented in

SoaML diagram in this Chapter. Service contract was applied when services share standardized contracts. Service loose coupling was used when services are decoupled from the implementation details and technology. Service abstraction was important to design the limit information in the service contract to what is really necessary for the service to be functionally useful to consumers. Service reusability was concerned in encapsulates the logic that is useful for more than one service request. Service autonomy is related to express a well-defined functional threshold which should not involve other services. Service statelessness was important to minimize dependence on the service state. Service discoverable was related to service discovery and interpret. Service composability refers of service composition to automate business processes. Granularity specifies the scope of business functionality and the structure of the message payload in a service operation that is provided.

EHR Implementation

The content of this chapter is mainly based on paper entitled “Layered Implementation View of a SOA Based Electronic Health Record” (LIMA et al., 2016). This paper was published on the Proceedings of the International Conference on Software Engineering and Knowledge Engineering (SEKE) in 2016.

This chapter presents the EHR implementation details documented during development.

8.1 EHR development experience

The development of the EHR was performed by an IT professionals team composed by six participants. Two university professors, one PhD student and three master students composed this team. During this research project, each participant collaborate with important aspect that resulted in the EHR final software product. The initial proposal of development of a hospital system was agreed through a partnership between the research group and the IT direction of the public hospital at Federal University of Sergipe (UFS).

The development of the EHR application took approximately 2 years. Initial phases included initial phases included face-to-face meetings at hospital to understanding daily routine of health professionals, real needs and which functionalities should be implemented, as well as understanding the business processes at the hospital. EHR team held meetings with the hospital IT staff, doctors, nurses, receptionist and some others professionals responsible for exams, laboratory and pharmacy. EHR system is currently hosted by a machine at UFS.

8.2 EHR implementation organization

The organization of the implementation applied for development of the EHR application is depicted in Fig. 42. UML package diagram represents an overview of how source code was organized in EHR. EHR system main package includes source code responsible

for implementation of services that implements SOA code for access information from external systems.

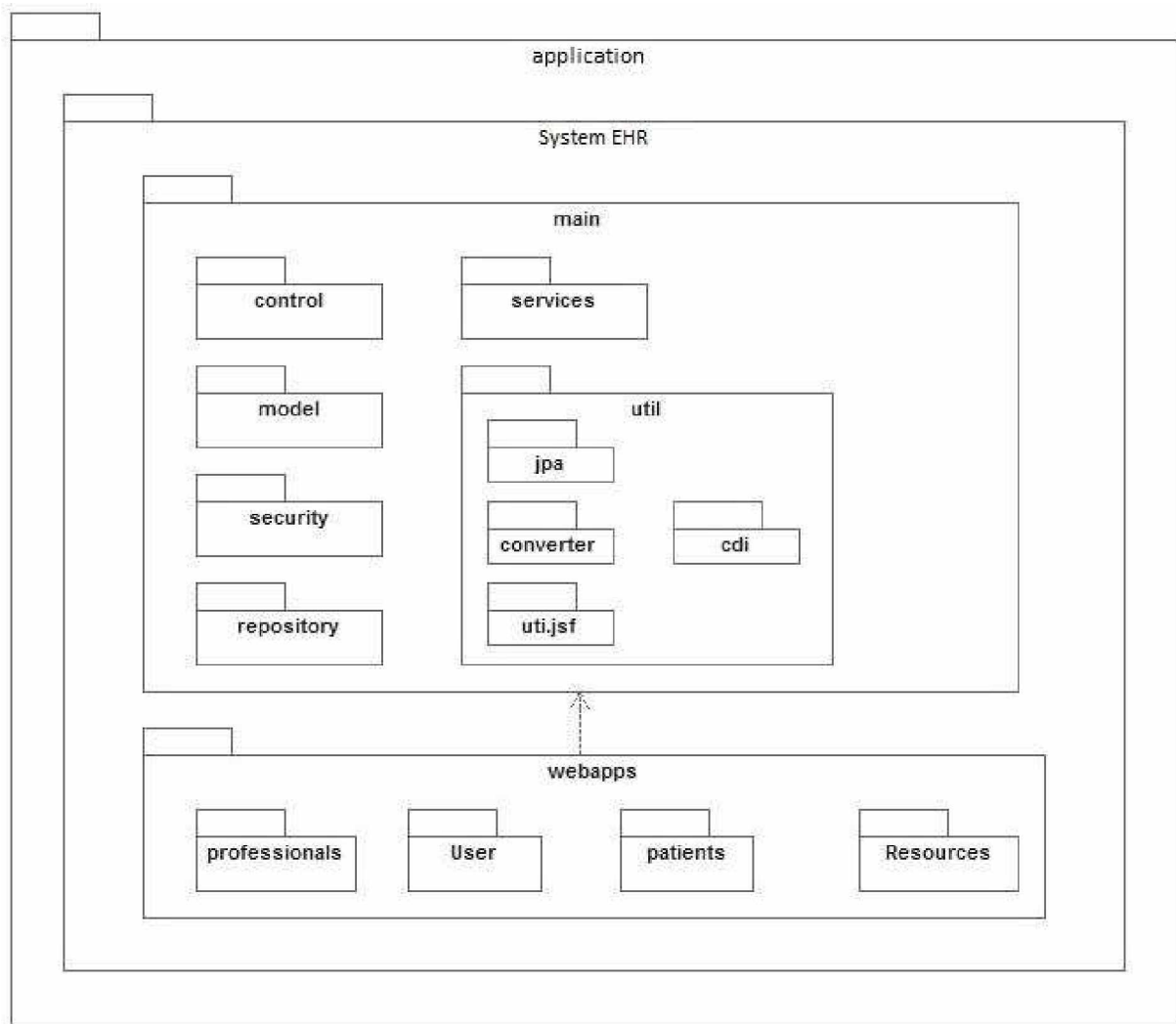


Figure 42 – Application components

Development of the EHR application is based on the Model View Controller (MVC) pattern, as depicted in Fig. 43. EHR source code was organized into three modules that are model, view and controller. Model considers components that directly manages the data, logic and rules of the application using Enterprise JavaBeans (EJB) and Java Persistence API (JPA). View generates output representation of EHR system using two frameworks that are primeFaces and JavaServer Faces (JSF). Control addresses input manipulation and converts it to commands for the model or view. Control represents components in EHR responsible for flow control and business rules.

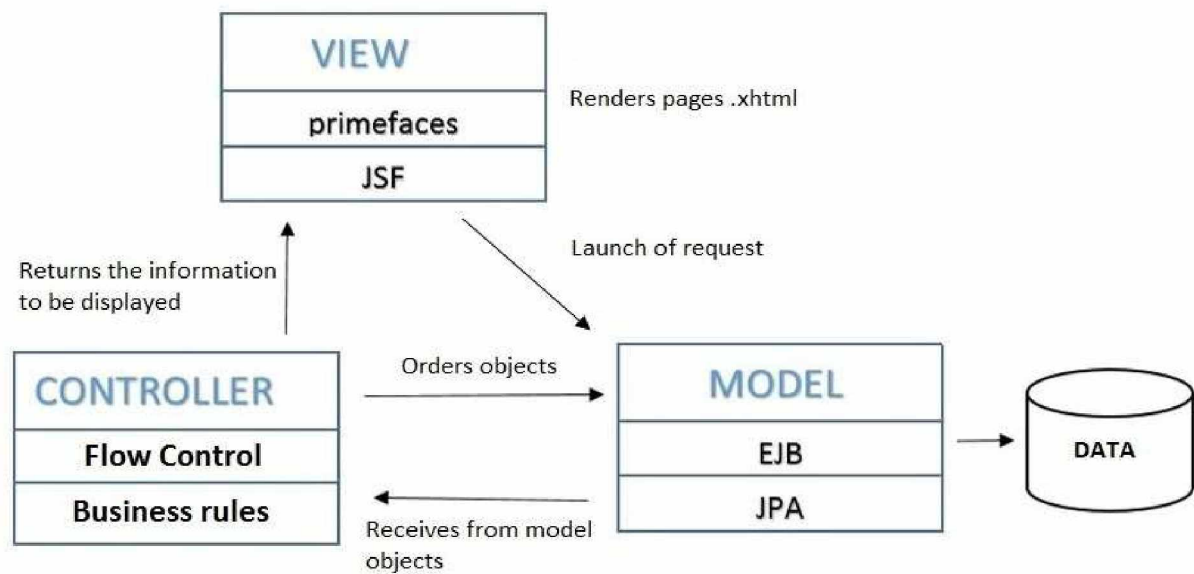


Figure 43 – MVC model for application implementation

8.3 EHR development Tools

Solution proposed to EHR included the use of free tools and technology due to low budget in public hospital. This section describes the tools and technology that composed the EHR development as presented in Table 16.

Java is the programming language used for implementation, together with frameworks Java Server Faces and Primefaces. Postgre is the relational database, together with Hibernate as JPA framework. Apache Maven was used for dependency injection, JBoss as application server, and Mule ESB as Enterprise Service Bus.

Technology	Layer	Role
PostgreSQL	Data	Relational Database
Web Services	Systems Integration	XML Interface
Mule ESB	Enterprise Service Bus	Web Services Integration
Hibernate	Distribution	Object-Relational Mapping
JSF, AJAX, PrimeFaces	Front-End	User interface communication
Java	Exception	Exception Handling
Apache Maven	Deploy	IDE configuration for Deploy
Log4J	Log	Logging resources implementation
JBoss	Distribution	application server

Table 16 – EHR tools and technology

8.4 EHR Functionalities

EHR was developed as a web application and it can be accessed through a browser from any device connected to Internet. Hospital professionals suggested that EHR should be access by mobile devices such as tablets and smartphones. Frequently, computers available at the hospital are not enough to meet all the professionals who need to use it. Therefore, portability is one important aspect of software and it was considered in EHR.

Graphical user interface of the EHR application was developed according to the hospital business process. Each patient is firstly attend by a hospital receptionist that search for patient health card. Next step consists in searching patient care map for the current date and identify if patient is scheduled for appointments. Receptionist check-in patient when he/she arrives and then his/her status is changed to ‘waiting for appointment’. Next paragraphs are dedicated to explain the main screens of the EHR application. Screens of the EHR are in Portuguese because it is the native language of users.

The first screen of the EHR is login, depicted in Fig. 44. To access the EHR, hospital professionals should enter with her username and password. EHR was developed with access control that includes three types of profiles that are receptionist, clinicians and administrator. Each profile only has access to authorized information, for example, a receptionist is not allowed to access medical records of patients. Receptionist profile only has access to patient personal data such as ID, health card number, address and scheduled appointments. Clinician profile can access medical record of patient and exams results. Administrator profile can modify settings about information display.

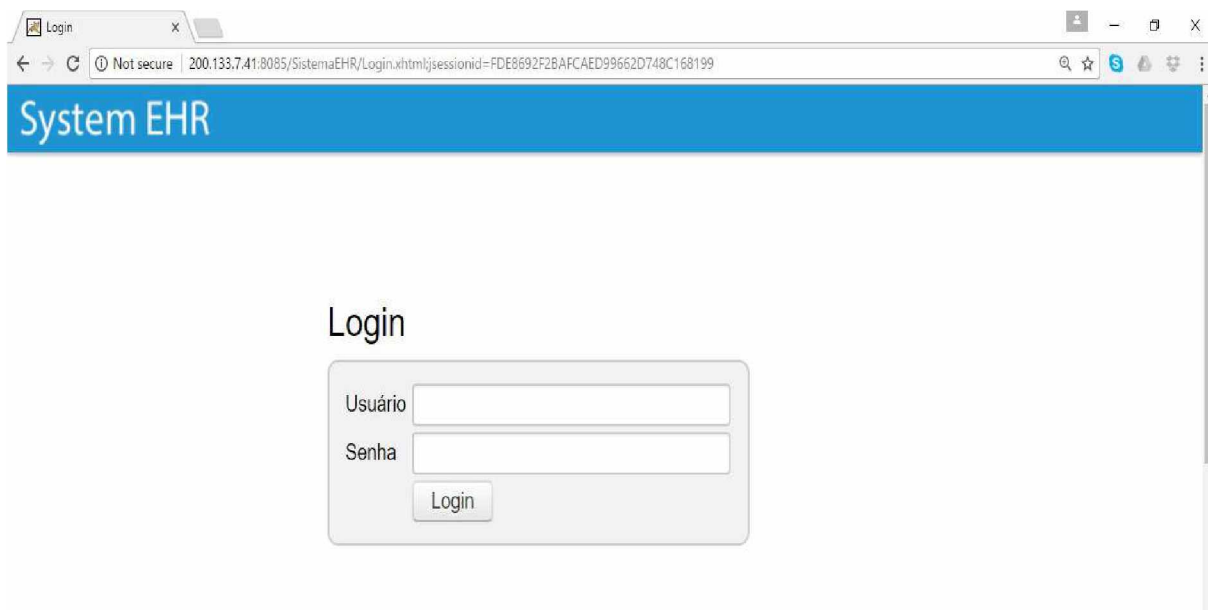


Figure 44 – EHR screenshot - Login

After logging into the system, a patient location map is presented (Fig. 45). It is possible to observe the location of patients at the hospital. Therefore, this information

is important to analyze locations where there are large concentrations of patients. To access patient care map, receptionist has to click the menu ‘care’ and then click the button ‘patient care map’.

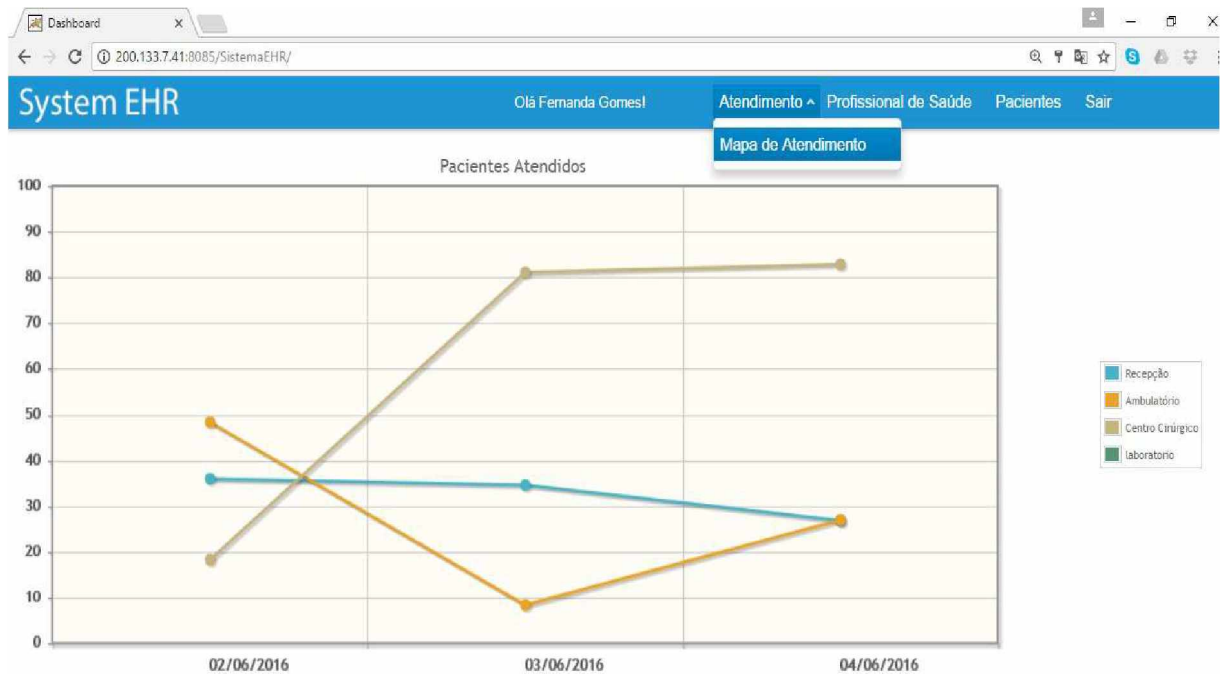


Figure 45 – EHR screenshot - Patient location map

Receptionists have access to patient care map as depicted in Fig. 46. Patient care map is to provide patients scheduled for appointments in certain date.

Mapa de Atendimentos

Check-in Realizado com Sucesso!

Check-in Cancelar Paciente Atualizar Imprimir

Data de Atendimento: 02/05/2017

Tipo de Atendimento: Todos

Status do Atendimento: Aguardando Atendimento

Chekin	Ordem	Tipo de Atendimento	Especialidade	N° Cartão SUS	Nome	Data de Nascimento	Status	Local de Atendimento	Início	Fim
ok	1	Consulta	Dermatologia	222222222222222222	Mario de Andrade	02/11/1987	Aguardando Atendimento	Ambulatório		
ok	3	Consulta	Dermatologia	11111111	Ana Franca	25/01/2014	Aguardando Atendimento	Ambulatório		

Figure 46 – EHR screenshot - Patient care map

Patient care map presents patient information such as name, birth date and health card number. Receptionist role is to request patient ID and health card to check with the information on the care map and then insert check-in when the scheduled patient arrives. Patient care map is provided to EHR application via service from AGHU legacy system.

When the receptionist selects the patient and click the check-in button, the patient is inserted in the list of patients waiting for the doctor's care. Fig. 47 presents a confirmation screen to double-check patient arrival.

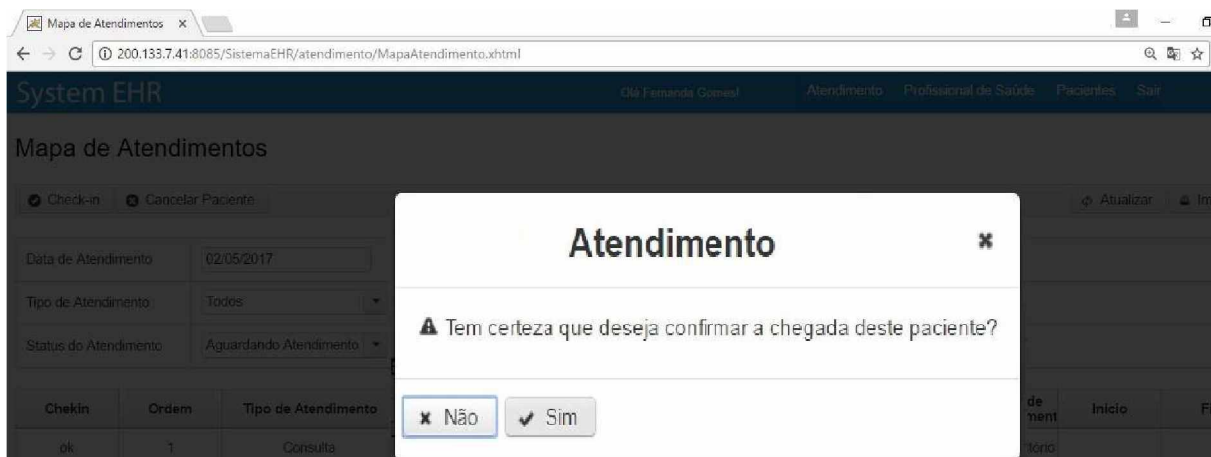


Figure 47 – EHR screenshot - Patient check in

Health professionals such as doctor, nurses, psychologist or physician have clinician profile. Therefore, they have access to patient medical record. When a clinician logs into the EHR, a screen similar to the presented in Fig. 48 appears.

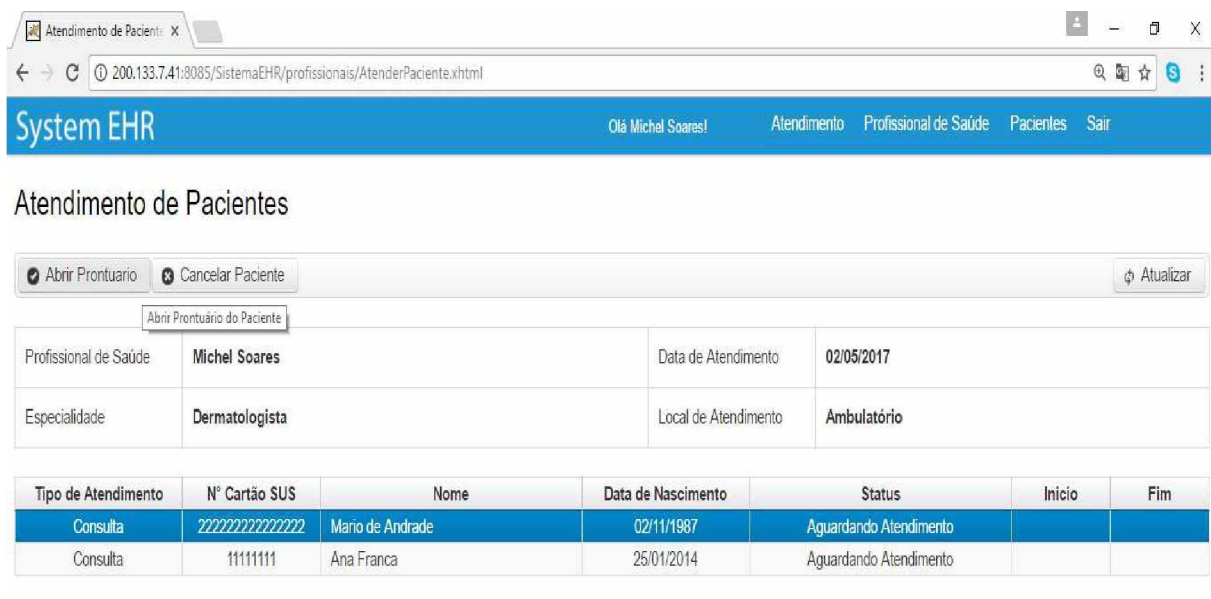


Figure 48 – EHR screenshot - Patient list

Clinicians are authorized to visualize patient care information including treatment evolution, exams and medicine prescription. Screen depicted in Fig. 48 presents a list of

patients that are ‘waiting for an appointment’.

Clinician selects a patient and clicks on button to open medical record. Information about patient is presented similar to screen in Fig. 49. EHR stores medical record of all appointments of patient at hospital.

System EHR
 Dr. Michel Soares
 Especialidade: Dermatologista

Alta do Paciente Encaminhar Enviar por e-mail Imprimir

Número do Prontuário	1	Data de Nascimento	02/11/1967
Nome	Maria de Andrade	Sexo	Masculino
Estado Civil	Solteira	Nº Cartão SUS	22222222222222

Anamnese Exame Físico Hipótese Diagnóstica Receita Solicitação de Exames Evolução

Inserir Anamnese

Data	Queixa Principal	Histórico da Doença	Histórico Familiar	Histórico Social	Alergia	Histórico Sexual
07/02/2017	dor					
07/02/2017					coceira	
23/02/2017	febre					
23/02/2017			cancer			
07/03/2017	dor de cabeça					
15/04/2017					Pele irritada	
20/04/2017		hemia de disco				

Figure 49 – EHR screenshot - Medical records

In each appointment, clinician can insert new medical records about patient including claims, history disease, familiar history disease and allergies. Fig. 50 presents the screen to insert patient anamnesis.

System EHR
 Dr. Michel Soares
 Especialidade: Dermatologista

Alta do Paciente Encaminhar Enviar por e-mail Imprimir

Informe os dados da Anamnese

Queixa Principal

Histórico de Doença

Histórico Familiar

Histórico Social

Alergia

Histórico Sexual

Adicionar

Figure 50 – EHR screenshot - Anamnesis

Clinician can search patient history and insert new information such as physical examination, diagnostic hypothesis, prescription medicines, exam request and patient's clinical evolution. Fig. 51 depicts screen to store medicines prescription. One important functionality to improve time spend on medical record fulfillment is to auto-complete medicines name.



Figure 51 – EHR screenshot - Medicine prescription

At the end of the appointment and after completion of the medical record, the clinician can click on the patient's discharge button. Fig. 52 shows the screen to confirm patient discharge on EHR system.



Figure 52 – EHR screenshot - Patient discharge

8.5 Discussion

Developing new software systems from scratch is frequently not possible, as there are many legacy systems with stored data that need to be maintained. With this in mind, the SOA paradigm is useful to integrate legacy systems by means of web services. For this reason, considering that having a well-defined software architecture is crucial for success of complex systems, this chapter describes the implementation details to develop EHR systems based on the SOA paradigm to integrate legacy systems.

The complexity of developing health information systems is well-known in the literature for many reasons, including heterogeneity of data and issues when defining the system architecture. In addition, within health information systems, the need for interoperability between legacy systems has been described in the literature, as these are normally heterogeneous systems that have a long lifespan.

The choice for interoperability is to identify services from legacy systems, providing a services catalogue, and then integrating them into an EHR application. This choice was taken as the SOA paradigm is useful to integrate legacy systems by means of web services.

The restrictions of use only free tools and technology due to low budget in public hospital it was not a problematic situation. The experience of use this set of tools to develop the EHR application was satisfactory because it was easy to use and Internet provides big amount of reference materials and tutorials.

Experimental Results

The content of this chapter is mainly based on the article entitled “Electronic health record system evaluation using users and experts opinion”. This article was submitted to the Journal of Systems and Software (JSS).

The SOAQM and the EHR application proposed in this research are evaluated in this chapter. This chapter proposes an evaluation divided into two parts. First evaluation addressed aspects related to software development process. Experts’ opinions (COOKE, 1991) were required to analyze the development process used during development of the EHR. Specialist evaluation was performed through questionnaire and interviews. The second evaluation had focus in providing analysis about EHR system users. This evaluation was performed with health professionals using Technology Acceptance Model (TAM) (DAVIS; BAGOZZI; WARSHAW, 1989).

9.1 Evaluation of the development process

In this section, the evaluation performed to assess development process used in the EHR is presented. The process of experts’ opinion evaluation followed some steps that are detailed as follows.

9.1.1 Protocol for Experts’ Evaluation

EHR application is a software system developed during this study using Service Oriented Architecture. SOA was considered to develop the EHR due to one important requirement related to legacy systems integration. EHR development process was guided by a particular format using quality model specially proposed for SOA applications named SOAQM. The main proposal of using this quality model is to consider quality attributes at early stages of process development.

Specialists in software development were invited to analyze the process of development of the EHR applied in this study. EHR was developed using SOAQM quality model to

guide the development of the architecture, design and implementation with focus on fulfilling essential quality attributes. Figure 53 presents the cycle of development process used in the EHR system.

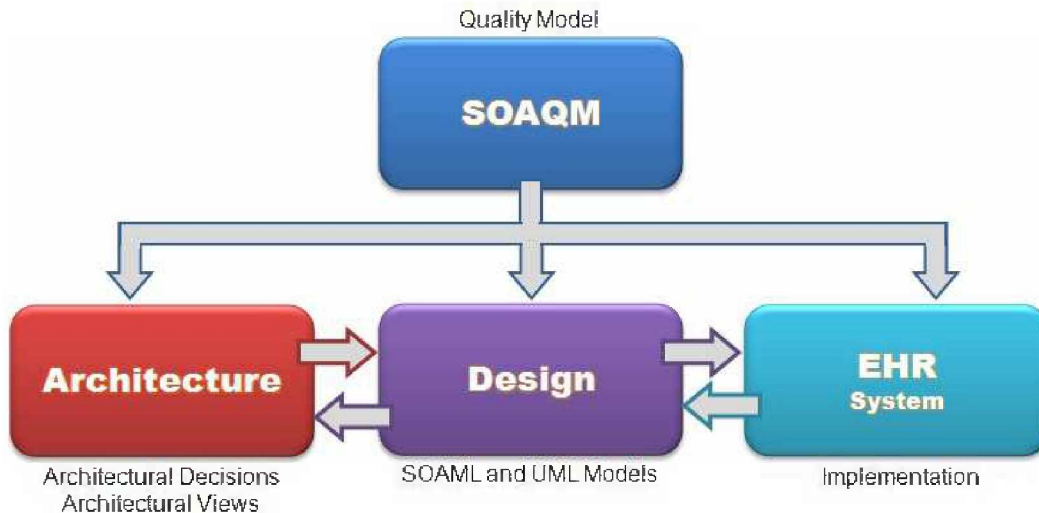


Figure 53 – EHR development process

Each volunteer was invited to participate on this evaluation by collaborating with three activities in the following manner:

1. Analyze the technical report

The technical report is a document (Appendix B) of four pages containing a short summary of the development process used to develop the EHR. This document includes description of EHR functionalities, SOAQM quality model, architecture described through architectural design decisions and architectural views, and design through SOAML diagrams.

2. Answer the questionnaire

The questionnaire is composed of 19 questions about development process. The objective of the questionnaire is to identify specialists' opinion about the method used to develop the EHR. Questions were formulated as affirmations related to possible conception of the EHR. Questionnaire responses have followed the Likert Scale. Participants have answered values between 1 and 5 representing: 1 - Strongly disagree, 2 - Disagree, 3 - Neither agree nor disagree, 4 - Agree and 5 - Strongly agree. Questionnaire ¹ was made available online through Google form platform and the responses of each volunteer are shared in a spreadsheet. This questionnaire is presented in Appendix C and it was written in Portuguese because it is the native language of participants.

¹ docs.google.com/forms/d/e/1FAIpQLSeXbhhttS18bodQM2ajUZXAifYaCrXz4GfH9Lgl4lxuPIR4BA/viewform

3. Interview

Participants were interviewed to provide opinion about evaluation and give final comments about the EHR. In addition to this, they could provide feedbacks about the development and SOAQM. Interviews were realized personally, by phone or via video conference.

Technical report document was sent via email to participants. Therefore, each participant had choice to define a better moment and how much time was needed to read and analyze the document. Face-to-face meeting would be very difficult because participants live in different cities (São Paulo/SP, Uberlândia/MG, Montes Claros/MG and Catalão/GO) and each one has several time restrictions. Questionnaire was made available online through Google form platform and the responses of each volunteer are shared in a spreadsheet.

9.1.2 Participants

Evaluation of EHR development process was performed by a set of eight volunteers. In total, fifteen participants were invited to participate of this evaluation but only eight volunteers could invest time to collaborate in further in the research. Technical participants have a bachelor degree in Computer Science (or similar graduation) and they are specialists in software development. All participants had no involvement in this study or during the stages of EHR development before the technical evaluation.

Volunteers have experience in Information Technology in several positions such as developer, project manager, software architect and academic as university docent. Two volunteers are from industry and six are from academia, of which three of them also have experience in industry.

Volunteer	Degree	Area	Job	Experience
TP01	PhD student	Industry	Software Developer	> 8 years
TP02	Bachelor	Industry	Software Architect	> 8 years
TP03	PhD student	Academia	University docent	6 years
TP04	PhD student	Academia	University docent	2 years
TP05	Master	Academia	University docent	6 years
TP06	Master	Academia	University docent	5 years
		Industry	Software Developer	5 years
TP07	Master	Academia	University docent	5 years
		Industry	Software Developer	2 years
TP08	PhD student	Academia	University docent	2 years
		Industry	Software Developer	2 years

Table 17 – Participants Profile

Table 17 presents the participants' profile. Volunteers have relevant experience in academia or industry or both. The set of eight technical participants includes seven

Masters in Computer Science and three of them are PhD students. Two volunteers have large experience in industry with more than 10 years working in software architecture and development. Another six volunteers have experience in academia and three of them have also worked in industry. All participants of this technical evaluation have previous knowledge in SOA concepts and issues.

According to Figure 54, six volunteers have worked (or still work) as university docents of which two of them have experience between 5 and 8 years. Four volunteers have worked (or still work) as software developer in industry and one of them works in software development for more than 8 years. Four volunteers worked in industry with project management during up to two years. Three volunteers have worked (or still work) as software architect in industry of which one of them works in this activity for more than 8 years.

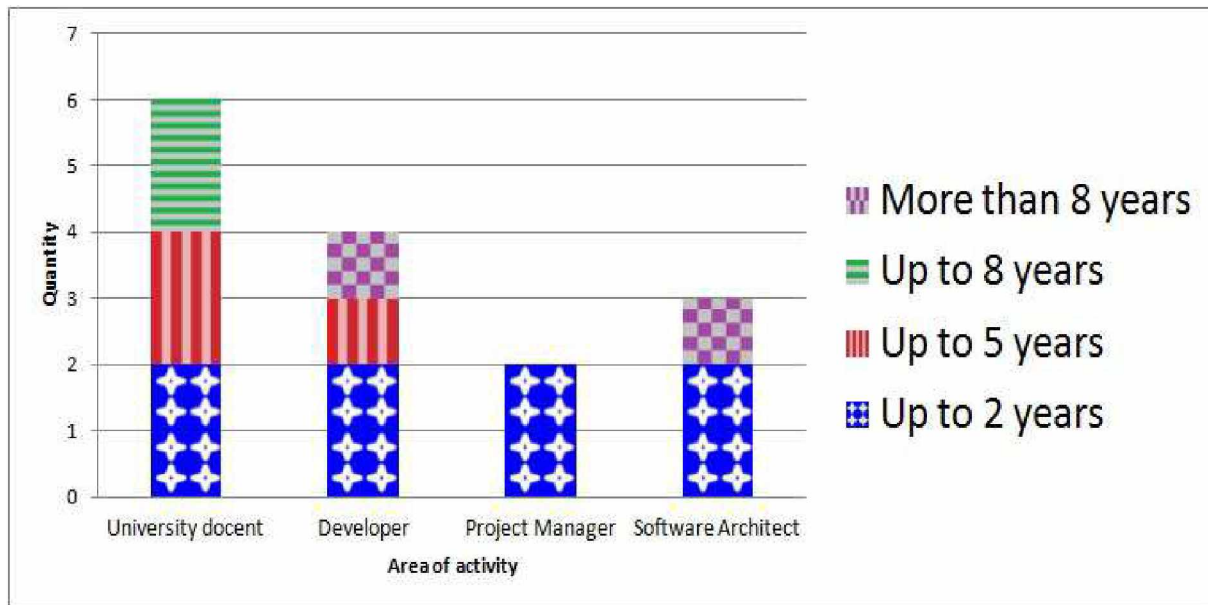


Figure 54 – Volunteers Experience

Each volunteer answered about his knowledge level about SOA and model quality standards. Table 18 presents the issues and knowledge according to their own opinion.

Table 18 – Participants Knowledge Level

Issue	TP01	TP02	TP03	TP04	TP05	TP06	TP07	TP08
I have knowledge of Service Oriented Architecture (SOA).	4	5	4	4	5	4	4	5
My work involves development of SOA based applications.	5	4	2	4	5	4	1	4
I have knowledge about ISO 25010.	1	2	4	2	5	2	4	4
I am interested in quality models for SOA applications.	2	4	3	4	5	3	4	5

From the results shown in Figure 55, some observations can be made. First, all participants have knowledge about SOA and therefore they are perfectly able to participate

of this technical evaluation. Secondly, the majority of participants frequently works with development of SOA applications. Another relevant observation is related to the fact that half of the participants do not know ISO 25010. This result may represent that participants do not apply ISO standards in practice because they do not know how to use them.

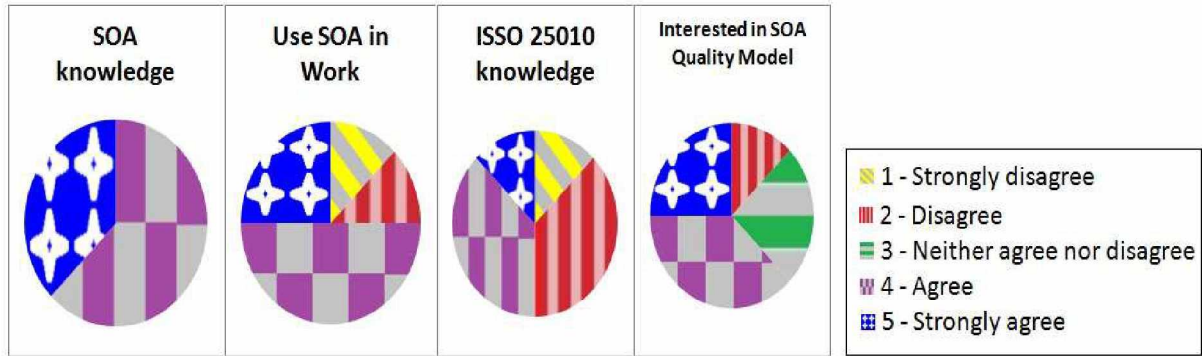


Figure 55 – Volunteers Knowledge

9.1.3 Results

Evaluation results are obtained through specialist volunteers that participate answering the questionnaire about EHR development. The questionnaire was formulated with nineteen sentences that should be answered with scores between 1 to 5. The questionnaire was divided into four sections. First section was about SOAQM topics with six sentences dedicated to analyze SOAQM issues in EHR development process. Second and third sections addressed EHR Architectural issues and eight sentences in the questionnaire are about architectural decisions and views. Fourth section is related to five sentences about EHR Design.

9.1.3.1 Sentences about SOAQM

SOAQM was created based on ISO 25010 for SOA systems. Quality attributes proposed in SOAQM can be used to guide development and this approach was applied to develop the EHR. First sentences of evaluation analyze the importance to use a quality model. In addition to this, some sentences aim to analyze specialists opinion about using quality model in early stages of development.

Table 19 presents in the first column six sentences related to SOAQM in development process. The next eight columns present specialists opinion with values between 1 and 5. The last two columns present the average and the standard deviation of participant responses.

Sentence 1 in questionnaire is related to the importance of using ISO 25010 standards. ISO 25010 was proposed with the aim to define important quality characteristic when

Sentence	TP01	TP02	TP03	TP04	TP05	TP06	TP07	TP08
1. I consider that it is important to follow the software development standards proposed by ISO 25010.	4	5	4	3	5	4	5	4
2. I consider that using a quality model in the early stages in the software development process is important to ensure higher product quality.	4	5	5	5	5	5	5	4
3. I consider that it would take short time to learn more about SOAQM.	3	4	3	3	3	4	5	3
4. I consider that using SOAQM would improve the quality of SOA-based applications.	3	5	4	4	4	5	4	4
5. Overall, I consider that the development process used can be applied in the development of other SOA systems.	4	5	4	4	4	5	4	4
6. Assuming I have a more in-depth knowledge of the SOAQM quality model, I could use it in developing other SOA systems.	4	4	4	4	4	5	5	4

Table 19 – Specialists Opinion about SOAQM

evaluating software product. Therefore, SOAQM is based in an international standard. According to specialists answering, in general they consider it is important to follow ISO 25010 standards.

Second sentence aims to analyze specialists opinion about the use of quality model in early stages. In the technical report, participants were introduced on how SOAQM can be applied in the early stages of EHR development. In general, specialists agreed that using a quality model in the early stages in the software development process is important to ensure higher product quality.

Third sentence aims to analyze what is their expectation for the learning curve on SOAQM. According to their answers, some of them would have no difficulties and consider that it would take short time to learn more about SOAQM.

Sentence 4 analyzes the relation between SOAQM and quality of SOA applications. The great majority of the interviewees can affirm that using SOAQM would improve quality of SOA-based applications.

Sentence 5 analyzes the applicability of SOAQM in other SOA systems projects. All participants have agreed that the development process used with EHR can be applied to development of other SOA systems.

Sixth sentence assesses the degree of interest in replicating the approach of development process used in EHR to other SOA systems. According to specialists, SOAQM quality model could be adopted in their SOA systems development.

9.1.3.2 Sentences about Architectural Decisions

Architectural Design Decisions were defined during development of the EHR. The ADDs were proposed with regard to meeting quality attributes. Technical report pre-

sented to experts before questionnaire contained the format to define ADD and its purpose.

Table 20 presents questionnaire sentences related to EHR Architectural Decisions. Questions 7 to 11 analyze the importance of defining ADD in software development, especially for EHR.

Sentence	TP01	TP02	TP03	TP04	TP05	TP06	TP07	TP08
7. I consider that it is important to define the format of architectural decisions in the software development process.	4	5	4	4	4	4	5	4
8. I consider that using architectural decisions based on quality model is important to ensure higher product quality.	4	5	4	4	5	5	4	4
9. I consider that it would take short time to learn more about how to define architectural decisions from a quality model.	3	2	3	3	3	3	3	3
10. Overall, I consider that the architectural decisions used can be applied to the development of other SOA systems.	3	4	4	4	4	3	4	4
11. Assuming I have more in-depth knowledge on the architecture used, I could use it in developing other SOA systems.	4	4	4	4	4	4	5	4

Table 20 – Specialists Opinion about EHR Architectural Decisions

Sentence 7 checks participants' opinion about the importance of defining the format of ADD. In general, participants consider that it is important to define architectural decisions in the software development process.

Sentence 8 is related to the use of quality models to motivate the definition of ADD. Architectural decisions proposed for EHR were defined using SOAQM quality attributes. This approach aims to focus on attending essential quality attributes on SOA systems. Specialists' opinion indicate that using architectural decisions based on quality model is important to ensure higher product quality.

Ninth sentence aims to evaluate time spend to learn the approach to define ADD from SOAQM. Participants used a neutral answer to this question, which may indicate that they are not sure what level of difficulty in learning this approach may be required.

Tenth sentence analyzes applicability of using ADD approach in other SOA projects. Most volunteers think that the architectural decisions used in EHR can be applied to development of other SOA systems.

Sentence 11 is about possible use of ADD approach in other SOA systems. Participants' answers indicates that whether they could have more in-depth knowledge on the architecture used, it is possible to consider using ADD in other SOA systems.

9.1.3.3 Sentences about Architectural Views

Architectural Views were defined during EHR development process. Views can help stakeholders to understand systems's functionalities. Table 21 presents sentences 12 to 14

that analyzes specialists' opinion about the importance of defining architectural views.

Sentence	TP01	TP02	TP03	TP04	TP05	TP06	TP07	TP08
12. I consider that it is important to define architectural views in the software development process.	4	5	5	4	5	4	4	4
13. I consider that defining architectural views is important to improve understanding of the system's functionalities.	4	5	5	4	5	4	5	4
14. Overall, I consider that architectural views should be applied in the development of other SOA systems.	3	5	5	4	5	3	4	4

Table 21 – Specialists' Opinion about EHR Architectural Views

Sentence 12 analyzes whether it is relevant to propose architectural views. Technical report presented views defined to EHR and participants evaluate that it is important to define views in software development process.

Sentence 13 aims to evaluate which benefits are aggregated as result of considering multiple views. Specialists' opinion indicate that defining architectural views is important to improve understanding of the system's functionalities.

Fourteenth sentence is related to using multiple views in other systems. In general, participants consider that architectural views should be applied to development of other SOA systems.

9.1.3.4 Sentences about SOA Design

EHR design was developed using a modelling language specific for SOA applications. SoaML was used to provide diagrams that represent systems legacy integration. SoaML is based on UML and this aspect represents one great advantage because UML has been widely used in other systems design. UML extension of SoaML can provide service representation that is fundamental concept of SOA systems. SoaML diagrams developed for EHR application were documented in the technical report.

Volunteers analyzes SoaML design presented in technical report and provide their opinion about the importance of using SOAML. Table 22 presents five questions related with the use of SoaML diagrams in EHR design.

Sentence 15 evaluates the importance of using a modelling language for SOA design. During the EHR development process, SoaML was used to produce diagrams that represent services. As result of this question, it is possible to note that specialists were not unanimous. Five out of eight participants consider that it is important to use a modeling language specific to SOA. In contrast, the other three participants consider that it is not so important to use a modelling language for SOA.

Sentence 16 analyses whether the use of SoaML provides an improvement in design understanding. As result, all participants' answers that using SoaML can improve understanding of SOA application design.

Sentence	TP01	TP02	TP03	TP04	TP05	TP06	TP07	TP08
15. I consider that it is important to use a modeling language specific to SOA.	3	4	5	2	4	5	5	3
16. I consider that using SoaML would improve understanding of SOA application design.	4	4	4	4	4	4	4	4
17. I consider that it would take short time to learn more about SoaML.	2	4	3	4	3	4	4	3
18. Overall, I consider that SoaML diagrams can be applied to development of other SOA systems.	3	4	4	4	4	5	5	4
19. Assuming I have more in-depth knowledge on SoaML, I could use it for developing other SOA systems.	4	4	4	4	4	5	5	4

Table 22 – Specialists’ Opinion about EHR Design

Sentence 17 aims to evaluate how much time would it take to learn how to use SoaML. Half of the participants considered that with short time it is possible to learn more about SoaML. The other half is not sure about how much time it would take.

Sentence 18 addresses the applicability of SoaML in other SOA projects. In general, participants consider that SoaML diagrams can be applied to the development of other SOA systems.

Sentence 19 deals with the adoption of SoaML. Assuming that the participants learned more about SoaML, they are prone to develop other SOA applications using SoaML for design.

9.1.4 Interviews

Specialists also provide their opinion through interviews. Volunteers presented their analyzes about evaluation and gave final comments about development of the EHR application. This part of evaluation is important to capture specialist feedbacks about approach used to develop the EHR. In addition to this, general comments help to understand some responses to the questionnaire.

Table 23 presents their opinion in relevant topics that they wanted to highlight.

According to interviews, experts’ opinion indicate some positive and negative aspects about SOAQM. Some advantages of using SOAQM are listed such as ISO 25010 adapted version for SOA systems. Therefore, learning curve is reduced because the focus is on what is relevant to SOA applications. Negative aspects about SOAQM were identified such as difficulty to understand how to use SOAQM in practice and some quality attributes also be considered as essentials.

Two participants provide opinion about Architectural Design Decisions proposed to the EHR. Experts’ opinion indicate that the format of ADD proposed in the EHR development is beneficial because it can evaluate design quality and improve understanding of architecture.

SOAQM	
TP01	Efficiency and capacity should be add as essential quality attributes.
TP02	Interesting because it is based on a software quality model and consequently reduces the learning curve and facilitates communication by sharing terms and concepts.
TP04	SOAQM presents more specific characteristics for SOA facilitating verification of quality for this type of system.
TP05	SOAQM provides details of software quality characteristics and the strong applicability in SOA systems.
TP06	SOAQM should be more applicable.
TP07	Ease of identifying essential attributes and differentiating them from those not relevant to a SOA system.
TP08	Adapted version of ISO 25010 allows better adoption for SOA systems.
Decisions	
TP02	Evaluates design quality and SOA development.
TP06	Improve understanding of architecture.
Views	
TP01	Views are very important for different stakeholders.
TP06	Improve understanding of requirements.
TP08	Ease of understanding for non technical professionals.
SoaML	
TP01	SoaML participant diagram clearly illustrate important information in SOA project.
TP06	Improves understanding of EHR system.
General Comments	
TP01	Need to reduce time and resources in software development and then SoaML does not seem to be essential.
TP02	SOA for EHR systems is necessary because it provides reuse and system legacy integration.
TP03	Coherent framework was followed for the description, elaboration and representation of different related services and legacy systems.
TP05	Approach can help developers in their SOA system development processes.
TP07	The development process adopted for development of the EHR system was excellent mainly for adopting a quality model to be followed from the early stages of development.

Table 23 – Experts’ Interview

Architecture of the EHR application was proposed based on multiple views. Experts’ opinion indicates some advantages in the EHR views including improve understanding of system requirements and easy understanding for different stakeholders.

The EHR design was proposed using SoaML modeling language. Most participants never used a modeling language specific to SOA. However, they considered that SoaML improves understanding of the EHR with regard to concepts related to SOA. In addition to this, SoaML diagrams clearly illustrate important information in SOA projects.

Although SoaML is considered beneficial in SOA projects, one participant warns that industry often needs to reduce time and effort during software development. Therefore, this participant asserts that SoaML does not seem to be essential.

In general comments, participants provide comments about the approach proposed to the EHR. Experts’ opinion indicate that the development process adopted in the EHR was a good approach due to the use of a quality model in early stages of development. In addition to this, they considered that this approach can help developers in other projects.

9.2 Evaluation of the EHR Application

This section presents evaluation that was performed to evaluate EHR usability. Users' evaluation focus in analyzing how well users can learn and use the EHR system. In addition to this, two quality attributes in SOAQM are related to completeness and correctness of system functionalities that are Functional Completeness and Functional Correctness. The user evaluation is important to verify if the EHR application covers all user objectives. This evaluation was performed by a set of nine health professionals and the process was based in TAM (DAVIS; BAGOZZI; WARSHAW, 1989) protocol.

9.2.1 Protocol for End Users' Evaluation

As mentioned before, EHR system was develop to be used in health context integrating hospital legacy systems and automate business process such as appointment management and medical record storage. Users of the EHR include health professionals, as for instance, doctors, nurses, psychotherapists and receptionists.

Users' evaluation was fulfilled by health professionals that attend patient in hospital or clinics. Doctors, nurses, psychologists and physiotherapists used EHR in order to understand system features. Therefore, to evaluate EHR usability, health professionals were invited to use the system and provide their opinion about this experience.

Due to agenda restrictions, it was not possible to gather all the volunteers in a single meeting. Each participant was interviewed individually and each meeting took about 30 minutes to finalize the entire evaluation process. Each volunteer was invited to participate of this evaluation collaborating with three activities in the following manner:

1. Execution of a use case scenario

Volunteers were introduced to EHR to understand its main functionalities. Each health professional was invited to use the EHR through execution of a case scenario. Participants were asked to search for medical records of a particular patient and to insert information about anamnesis and medicines prescription. This activity aims to investigate whether the EHR user interface is intuitive and easy to use. As result, all volunteers were able to easily perform the proposed activities.

2. Answer the questionnaire

The questionnaire is composed of nineteen questions about EHR functionalities and user interface. The objective of the questionnaire is to identify health professionals' opinion about using the EHR in their work routine. The questions were formulated as affirmations related to acceptance of the EHR as technology to be adopted in their work environment. Questionnaire responses have followed the Likert Scale. Participants have answered values between 1 and 5 representing: 1 - Strongly disagree, 2 - Disagree, 3 - Neither agree nor disagree, 4 - Agree and 5 - Strongly agree.

Questionnaire ² was made available online through Google form platform and the responses of each volunteer are shared in a spreadsheet. This questionnaire is presented in Appendix D and it was written in Portuguese because it is the native language of participants.

3. Interview

Participants were interviewed to provide opinion about evaluation and give final comments about the EHR. In addition to this, they could give feedbacks and suggestions about the EHR. Questionnaire was made available online through Google form platform and the responses of each volunteer are shared in a spreadsheet.

9.2.2 Participants

Evaluation of EHR was performed by nine volunteers. In total sixteen participants were invited to participate of this evaluation but only nine volunteers could invest time to collaborate in further to the research. The set of health professionals includes two doctors, three nurses, two psychologists and two psychotherapists. All participants work in their training field and attend patient in hospital or clinics.

Table 24 depicts professions of the nine participants that includes nurses, psychologists, psychotherapists and doctors. All participants of user evaluation are responsible to fulfill patient medical records during their work routine. Filling out medical records is performed through paper, software or text editors. Clinician software is used by some professionals. Two volunteers could describe which clinician software is used, as for instance ‘Fast Medic’ and ‘Alert’.

Volunteer	Profession	Medical record fulfillment	Medical record format	Work
U01	Nurse	Yes	Fast Medic	Hospital
U02	Nurse	Yes	Paper	Hospital
U03	Nurse	Yes	Paper	Hospital
U04	Psychologist	Yes	Text editor (Word)	Clinic
U05	Psychologist	Yes	Paper	Clinic
U06	Psychotherapist	Yes	Paper	Clinic
U07	Psychotherapist	Yes	clinician software	Clinic
U08	Doctor	Yes	clinician software	Clinic
U09	Doctor	Yes	Paper and Alert	Hospital

Table 24 – Health professionals Profile

Five out of nine participants uses medical records in paper format. According to participants, paper is not a good solution but in some cases it is the only option because a software application is not available. Paper is not a good solution mainly for

² docs.google.com/forms/d/e/1FAIpQLSe8l3bHH8lha5xJmASz2MutLNiWqdsp2SS5fF47StV6dhTfDA/viewform

three reasons: illegibility, storage and difficult access. In general, participants considered that medical record in paper format frequently presents illegibility problems because it is handwritten. Another problem is related to medical records storage. The large amount of paper accumulated over the years takes up a lot of space within the hospital. Several rooms that are occupied by medical records files could be destined for other purposes. In addition to this, another problem related to large paper accumulation is that the demand for records of previous appointments become a complicated activity. Table 25 presents participants opinion about using medical record in paper format.

Issue	U01	U02	U03	U04	U05	U06	U07	U08	U09
The patient records handwritten often presents illegibility problems.	5	5	5	3	4	5	5	4	5
Search patient records is difficult due to the accumulation of papers in the medical record file.	5	5	3	5	5	5	5	5	5

Table 25 – Paper acceptance Level

9.2.3 Results

Following TAM model, EHR user evaluation considers three important aspects related to user interface that are utility, ease of use and usability. Questionnaire contains nineteen sentences related to these three aspects, detailed in the next sections.

9.2.3.1 Sentences about EHR utility

One section of the questionnaire presents sentences to identify users' opinion about EHR utility. In this questionnaire, eight sentences were proposed to investigate participants' opinions related to how useful is the EHR to their routine at work. These eight sentences were dedicated to identify which participants consider that EHR can improve performance, productivity, organization, effectiveness and quality of patient care.

Participants responses about utility of the EHR are presented in Table 26. Volunteer users in this experiment used EHR for few minutes but it was enough to identify that the EHR is useful to perform their activities at work. This is a result of sentence 1 which all users agreed that the EHR is useful to perform their activities at work.

Sentence 2 is dedicated to analyze if using the EHR can improve users performance at work. Comparing the user current procedure and the use of the EHR, all participants agreed that using EHR would improve performance because it would be faster to access the patient's disease history.

Third sentence addresses users' productivity using the EHR. According to users, using the EHR, productivity increases because filling in medical records was faster with the EHR than current procedures.

Sentence	U01	U02	U03	U04	U05	U06	U07	U08	U09
1. I consider that the EHR is useful to perform my activities at work.	5	5	5	5	5	5	4	5	5
2. I consider that using the EHR would improve my performance because it would be faster to access the patient's disease history.	5	5	5	5	4	5	5	5	5
3. I consider that using the EHR to fill medical records would improve my productivity because I could save time in my work.	5	5	5	5	5	5	5	5	5
4. I consider that using the EHR would improve the organization of my work because the space used to store the records of the medical records file could be used for other purposes.	5	5	5	5	4	5	5	5	5
5. I consider that using the EHR would improve the effectiveness of my work because the hospital system would be integrated with other important external systems.	5	5	5	5	4	5	5	5	5
6. I consider that using the EHR would improve the quality of the medical record produced.	5	5	5	5	4	5	5	5	5
7. I believe that using the EHR would improve patient care, as regards the speed of filling the medical record.	4	5	3	4	4	5	5	5	4
8. I consider that using the EHR improve the management of patients within the hospital regarding the referral of patients for offices, laboratories and surgical rooms.	4	5	5	5	4	5	5	5	5

Table 26 – Users opinion about EHR utility

Sentence 4 is related to space organization. Frequently, hospital waste space to organize medical records in paper format. According to users, using the EHR would improve the organization because the space used to store records of the medical records file could be used for other purposes.

Fifth sentence aims to identify the real necessity of system integration. Health professionals have observed that hospital systems should be integrated with other systems. Users consider that using the EHR would improve the effectiveness of their work because the hospital system would be integrated with other important external systems.

Sixth sentence is related to the quality of medical record produced. All participants considered that using the EHR would improve the quality of the medical record produced.

Sentence seven aims to identify users' opinion about quality of patient care using the EHR. Most participants considered that using the EHR would improve patient care, as regards the speed of filling the medical record. One participant provide a neutral response and his opinion is based on the fact that it is not possible to generalize because some users present difficulty to use the technology.

Sentence eight addresses medical referral of patient at hospital. EHR application allows to visualize patient location at hospital and users considered that this functionality is beneficial. Therefore, users believe that using the EHR can improve the management of patients within the hospital regarding the referral of patients for offices, laboratories

and surgical rooms.

9.2.3.2 Sentences about EHR ease of use

Another section of questionnaire of tool evaluation consists in identifying EHR ease of use. Each participant of this evaluation was challenged to use the EHR through execution of a case scenario. Participants were asked to search for medical records of a particular patient and to insert information about anamnesis and medicines prescription. This activity aims to investigate whether the EHR user interface is intuitive and easy to use.

The experience of using the EHR by volunteers allows them to provide opinion about EHR ease of use. In this questionnaire, six sentences were proposed to investigate participants' opinions related to how easy is to manipulate the EHR. Participants' responses about this aspect of the EHR are presented in Table 27.

Sentence	U01	U02	U03	U04	U05	U06	U07	U08	U09
9. I considered that the EHR screens are organized with simplicity.	5	5	5	5	4	5	5	5	5
10. Overall, I consider that the EHR easy is to use.	5	5	5	5	4	5	5	5	5
11. I find it easy for me to learn to use the EHR.	5	5	4	5	4	5	5	5	5
12. I can easily fill in the medical record in the EHR.	5	5	5	5	4	5	5	5	5
13. I can easily perform patient data searches in the EHR.	5	5	5	5	4	5	5	5	5
14. I can easily identify the number of appointments scheduled on the day in the EHR.	5	5	5	5	4	5	5	5	5

Table 27 – Users opinion about EHR ease of use

Sentence nine is related to the EHR screen organization. All users considered that the EHR screens are organized in a simple way.

Tenth sentence aims to identify users' opinion whether the EHR is easy or difficult to use. Overall, all users considered that the EHR is easy to use.

Eleventh sentence addresses difficulties in learning how to use the EHR. Overall, all users considered that it was easy for them to learn to use the EHR.

Sentence twelve is related to user experience in filling out a medical record in the EHR. All users could easily fill medical record in the EHR.

Sentence thirteen aims to identify user ability to perform searches in the EHR. All users affirmed that it is easy to perform patient data searches in the EHR.

Fourteenth sentence addresses the user ability to perform a search for appointments at a hospital. Overall, users considered that it is ease to identify the number of appointments scheduled on the day in the EHR.

9.2.3.3 Sentences about EHR usability

Another important aspect to consider in a tool evaluation is usability. Five sentences in the questionnaire are dedicated to evaluate users' opinion about EHR usability. Participants' responses about this aspect are presented in Table 28.

Sentence	U01	U02	U03	U04	U05	U06	U07	U08	U09
15. Assuming I have access to the EHR system, I intend to use it.	4	5	4	5	4	5	4	5	5
16. Assuming I have access to the EHR system, I would recommend its use for other co-workers.	4	5	5	5	5	5	5	5	5
17. I consider that it would take little time to learn how to use the EHR.	4	5	5	5	4	5	5	5	5
18. I considered that the EHR would have high acceptance in a hospital.	4	5	3	4	4	5	5	5	5
19. I consider that the portability of the EHR is important, with respect to the possibility of access through computers, tablets or mobile phones.	5	5	2	5	4	5	5	5	5

Table 28 – Users' opinion about EHR usability

Sentence fifteen aims to identify intention of using the EHR. As response to this sentence, users considered that if they would have access to the EHR system, they intend to use it.

Sentence sixteen addresses user' opinion about the use of the EHR by other health professionals. Overall, users would recommend using the EHR for other co-workers.

Seventeenth sentence is related to users' opinion about how much time they considered that they would spend to learn EHR functionalities. Users responses indicates that little time would be spent to learn more about how to use the EHR.

Eighteenth sentence aims to identify users' opinion about acceptance of the EHR by hospital professionals. Generally, users considered that the EHR would have ease acceptance in the hospital.

Sentence nineteen is related to portability aspect of the EHR. Most users considered that portability of the EHR is important and they agreed that the possibility of access through computers, tablets or mobile phones is beneficial. However, one user did not consider that this functionality would improve current procedures in his work.

9.2.4 Interviews

Users participants of the EHR evaluation also provides their opinion through interviews. Volunteers presented their analyzes about evaluation and gave final comments about the EHR. This part of evaluation is important to capture feedbacks about developed tool and its acceptance. In addition to this, general comments help to understand some responses of questionnaire.

Table 29 presents users' opinion in relevant topics that they have wanted to highlight.

Comparison between EHR and current procedures of patient care	
U01	Safer and easier access.
U02	Facilitates the appointment process and confidentiality of the patient because the information is accessible only to the relevant professionals.
U03	Agility, ease, avoids paper handling that is perishable.
U04	Facilitates access to the patient's history in all aspects (complaints, family history, date of appointments), which is usually more difficult to access with conventional care procedures.
U05	Facilitates the process of care, contains all the necessary information of the patient and help from the reception to the appointment.
U06	Save time.
U07	Ease of identifying essential attributes and differentiating them from those not relevant to a SOA system.
U08	Faster and reduces the risk of information loss.
U09	Simpler format facilitates its use by physicians who do not dominate much the knowledge in technology, also allows access to the history of the patient and optimizes the prescription of medicines reducing the time spent in care.
EHR advantages	
U01	Agility, ease, practicality and execution of the system is fast.
U02	Easy access and saves space for paper storage.
U03	Faster, easier, avoid paper.
U04	Fast access to information and ease of access to all patient history data.
U05	Fast access to information, ease of use and agility in the appointment process.
U06	Rapidity.
U07	Stores information on one system.
U08	Agility and precision.
U09	Time optimization, data security, ease of prescription, easy access to patient's clinical history.
EHR disadvantages	
U01	Older professionals could have difficulty with technology.
U02	None.
U03	Dependent on energy and Internet.
U04	None.
U05	None.
U06	Dependent on server availability.
U07	Dependent on energy and Internet and people adaptable to new technologies.
U08	None.
U09	Dependent on Internet.
Reasons to use or not to use the EHR	
U01	Easy to operate and run on mobile device.
U02	Easy to use, confidential, reduced risk of data loss.
U03	The system provides many benefits for patients and work.
U04	The system provides advantages, mainly for the practicality, speed and ease of filling data.
U05	Ease of describing patient information, as well as promoting sustainability.
U06	Could access anywhere.
U07	Avoid frequent access to the file.
U08	Ease to use.
U09	Ease to use and improve appointments.

Table 29 – Users' Interview

According to interviews in Table 29, users' opinion indicates some positives about the EHR. Users have compared their current procedures at hospital between the EHR and some advantages about the EHR include agility, ease of medical record access, improved confidentiality, save time and avoid paper handling.

Users have indicated some disadvantages in the EHR adoption such as rejection by non-experts in technology and system is dependent on energy, Internet and server availability.

However, these problems are inherent in the adoption of any web system.

Users also described reasons to use or not to use the EHR application. All users have defined only reasons to use due to the good acceptance of the EHR to volunteers. Participant named User01 detached that the EHR is better than current software used because EHR runs on mobile devices. Other participants affirm that could use the EHR due to its many benefits for patients and work.

Another aspect addressed in users' interviews was difficulties found in using the EHR. This aspect is not in Table 29 because users were unanimous in reporting that they had no difficulty in using the EHR.

9.3 Discussion

This chapter presented evaluation of research proposals. Two different evaluations were performed to observe EHR development process and EHR as tool. Both questionnaires were previously validated by volunteers. Questionnaires were firstly answered by volunteers to test questions against errors, ambiguities and misinterpretations. This step was important because errors in questionnaire were identified and were corrected before being widely answered.

Threat to validity of these evaluations is related to the sample of participants. Few people can invest a considerable part of their time to collaborating with evaluation surveys. However, the users and experts involved are experienced in their expertise areas and can provide accurate opinions on the issues. Some evaluations use the approach of soliciting opinion in forums and communities however, it is not possible to guarantee the seriousness of the answers of unknown people.

Conclusion

This thesis was proposed to address underexplored issues in quality of Service-Oriented Architecture systems. SOA has been widely adopted to develop increasingly complex software systems and thus quality becomes an important concern. Several research gaps in SOA quality were found in the literature, and many studies have been proposed to achieve improvements in SOA quality applications. This situation shows that SOA quality is an essential concern that has to be more often considered by research projects.

This thesis proposes approaches to consider SOA quality in early stages of software development. First, specific design principles for SOA systems are presented. A systematic literature review was performed to identify SOA design principles and also how they can be applied in practice and which are the benefits in SOA quality provided from their use. As result of this SLR, it is possible to note that in most studies design principles are presented as key concepts, but are not always shown in case studies as being explicitly used and how they are used in practice. Another interesting result of the review is that few studies mention how design principles affect quality.

Another important issue addressed in this thesis is related to quality models for SOA. Some quality models have been proposed in the literature but they are based on standard ISO 9126 or even are not based on a known quality model. Recently, a more comprehensive ISO standard for quality model named ISO 25010 has been proposed. This standard includes quality attributes that should be considered in a software system. However, these quality attributes have to be adapted to SOA because they have focus on any type of software system. In this way, this thesis proposes a quality model specific for SOA applications named SOAQM. SOAQM comprises a set of quality attributes considered essential for SOA system development. In addition, SOAQM was applied in an approach to guide the definition of software architecture and design of an EHR application. The architecture was defined through architecture design decisions to achieve essential quality attributes.

Finally, this thesis proposes an implementation of a case study in the health domain. An Electronic Health Record has been developed to attend the needs of a public hospital.

The EHR scope includes legacy systems integration and a SOA solution was proposed. SOAQM has been used to develop the EHR system from its beginning. The approach used to develop the EHR was evaluated by a set of IT experts. In general, experts provided favorable opinion about SOAQM and the approach used to develop the EHR. In addition to this, the EHR was evaluated according to the degree of acceptance in a hospital. Users' opinion clearly indicate acceptance related to the EHR utility, usability and ease of use.

10.1 Answer to research questions

A set of six research questions was proposed in this thesis as defined in Subsection 1.2.2. Answers to these research questions are detailed as follows.

1. SOA design principles

Initially, one research question related to SOA design principles was proposed. *RQ1 - How are current SOA design principles being applied in practice?*

This research question is answered in Chapter 3. The research instrument to provide the response to RQ1 was a Systematic Literature Review. According to SLR results, Service Loose Coupling and Service Reusability were the most cited principles in studies. In general, design principles described by Erl in (ERL, 2007) - Service Contract, Service Loose Coupling, Service Abstraction, Service Reusability, Service Autonomy, Service Statelessness, Service discoverability and Service Composability - are the most widely applied. Therefore, this set of design principles was important to define design and implementation of the EHR. In addition to this, main results of the performed SLR were important to guide next steps in research.

2. Quality model for SOA quality based on ISO 25010

The second research question is: *RQ2 - What ISO 25010 characteristics are most suitable when developing SOA applications?*

In order to answer this question, in this thesis a quality model named SOAQM was developed. According to SOAQM, a set of twenty quality attributes are essential in SOA development that are Functional completeness, Functional correctness, Functional appropriateness, Time behaviour, Resource utilization, Co-existence, Interoperability, Operability, Maturity, Availability, Fault tolerance, Recoverability, Confidentiality, Integrity, Accountability, Authenticity, Modularity, Reusability, Modifiability and Testability.

ISO is the main standards provider for many aspects of software and systems development and evolution (NANCE; ARTHUR, 2012) (ORIOLE; MARCO; FRANCH, 2014). ISO 25010 proposes a set of quality attributes for software products. However, SOA context requires adaptation of these quality attributes.

A number of specific quality models for SOA applications has been proposed in the literature: S-Cube (GEHLERT; METZGER, 2008), WSQM2.0 (OASIS, 2005), Quamoco-extended (GOEB; LOCHMANN, 2011), and quality model (CHOI; KIM, 2008). However, a lack of SOA quality models based on ISO 25010 was found in literature.

Research question RQ2 is addressed in Chapter 4.

3. Approach to consider quality in early stages of software development

The third research question is: *RQ3 - How a quality model can guide architectural design decisions in the development of a SOA Application?*

This research question is addressed in Chapter 5. In this chapter, a set of architectural design decisions is proposed with the aim of considering quality attributes in the early stages of development process. SOAQM quality attributes were used to guide the definition of these ADDs. The set of ADDs was defined to be used in the EHR application but, according to experts' evaluation, these ADDs can be applied in other SOA systems.

4. EHR architecture using multiple views

The fourth research question is: *RQ4 - How to propose a SOA EHR application Architecture using multiple views?*

This research question RQ4 is addressed in Chapter 6. It is recommended by software engineering literature that a software architecture description include a number of architecture views (ISO/IEC, International Organization for Standardization, 2011b). Multiple views are important due to separating concerns that help understanding by different stakeholders.

In this thesis, a set of views is proposed to compose SOA architecture of the EHR application. Five views were considered important during the EHR due to different kind of hospital stakeholders. Proposed views are Scenarios View, Business Process View, Implementation View, Process View and Logical View. As result of this experience, it is possible to affirm that multiple views are beneficial during SOA development process.

5. EHR design using SoaML

The fifth research question is: *RQ5 - How to design a SOA EHR Application using SoaML?*

This research question is addressed in Chapter 7. In order to answer this question, in this thesis SoaML diagrams were developed to represent SOA aspects in the EHR application.

Some studies (WU et al., 2009) (GRANELL; DÍAZ; GOULD, 2010) (CHO et al., 2010) (KANNAN; BHAMIDIPATY; NARENDRA, 2011) (PEREPLETCHIKOV;

RYAN, 2011) (MONSIEUR; SNOECK; LEMAHIEU, 2012) (TIAN; HUANG, 2012) in the literature use general purpose modeling languages to create SOA design. However, comparing these studies with the EHR application developed in this thesis, it is possible to note that SoaML can clearly represent SOA aspects such as service, legacy systems involved and interactions.

6. Report EHR implementation and functionalities

The sixth research question is: *RQ6 - How to report and document a SOA EHR Application?*

The research question RQ6 is addressed in Chapter 8. The EHR application developed in this research was proposed to a public hospital. Therefore, the hospital IT staff will be responsible for performing EHR maintenance. The EHR documentation must be clearly detailed to help future maintenance and evolution in the EHR. Every tools and technologies used are contained in the EHR technical report. The technical report also includes EHR requirements and models such as UML diagrams to represent project scope and SoaML diagrams to represent SOA elements.

10.2 Main Contributions

Some contributions are presented as result of this research as follows:

1. Applicability of SOA design principles

This thesis presents a set of design principles and how they can be applied in practice. The existing SOA principles were classified in relation to their real applicability in practice, measured advantages in development and quality of SOA applications and scope of use. Thus, during this research a systematic literature review has been developed to investigate these issues.

2. SOA quality model based on ISO 25010

This study contributes with a new quality model that defines essential quality attributes for SOA applications. The quality model proposed in this research project is named SOAQM. SOAQM is used to guide application development with focus on quality attributes in SOA based on ISO 25010.

3. Approach to apply quality attributes in application

This thesis presents an approach to describe architecture decisions to achieve essential quality attributes in SOA applications.

4. Case study in health domain

This research project was used to implement a SOA application to attend the public hospital needs. EHR system was designed and implemented using SOAQM model to guide development. The case study was used to validate the research.

10.3 Future Work

In this section, future research is proposed. Future work in this research includes some activities related to SOA design and quality. It is possible to describe at least three interesting topics to be more explored in SOA research as follows.

Evaluation of SoaML

One of many challenges in SOA research detached by Papazoglou et. al in (PAPAZOGLOU et al., 2008) is the lack of specific design models developed to describe SOA concepts and elements. Several studies (WU et al., 2009) (GRANELL; DÍAZ; GOULD, 2010) (CHO et al., 2010) (KANNAN; BHAMIDIPATY; NARENDRA, 2011) (PEREPLETCHIKOV; RYAN, 2011) (MONSIEUR; SNOECK; LEMAHIEU, 2012) (TIAN; HUANG, 2012) in literature have not used modeling languages that are specific for SOA to represent SOA design.

SoaML is a UML profile that is gaining attention in the literature (MOHAMMADI; MUKHTAR, 2013) (TOUNSI et al., 2013) (IONITA; MOCANU; CIOLOFAN, 2013) (KOSEK; GEHRKE, 2014) (GUERMAZI; BEN-ABDALLAH; AYED, 2015) (RHAZALI; HADI; MOULOUDI, 2016). Therefore, there is a challenge to evaluate SoaML to make SoaML acceptable and widely adopted in the software development community. For instance, it is expected that new courses, books and processes are going to be proposed by researchers and also in industry. In addition, how SoaML can be used together with other modeling languages, mainly UML, is an open research theme.

SOA quality metrics

Literature has proposed some metrics to measure quality in SOA applications. One of these studies was performed by Choi and Kim (CHOI; HER; KIM, 2007) where QoS metrics have been defined to evaluate whether services provide the responses for consumer application. 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. Another study (CHOI; KIM, 2008) proposes metrics to evaluate service reusability. In the study, reusability was evaluated using quality attributes such as modularity and adaptability. Another study (Nadanam, P. and Rajmohan, R., 2012) describes a quality model for evaluating QoS (Quality of service) for web services. This quality model proposed some attributes and metrics considered relevant for cloud services. Two quality characteristics were considered in this model: Efficiency and Reliability.

A small set of metrics is proposed in literature to measure SOA quality (GOEB; LOCHMANN, 2011). It is possible to notice that there is a need for metrics to measure quality attributes proposed by ISO 25010 such as Modularity, Modifiability,

Availability and Reusability. Therefore, future work could use SOAQM to propose new metrics for SOA applications.

Tools for measuring quality in SOA applications

Another research gap found in the literature is the lack of tools for measuring quality of SOA applications (GOEB; LOCHMANN, 2011). The research project Quamoco (KLÄS; LOCHMANN; HEINEMANN, 2011) (LOCHMANN; HEINEMANN, 2011) is a quality model for software that defines a generic meta-model. However, Quamoco listed several upcoming activities to complete the project; one of them is to create a tool to produce quality measures. Another quality model, S-Cube (GEHLERT; METZGER, 2008), proposes 89 quality attributes for service oriented computing based on software engineering attributes. However, the authors also indicate that the development of a tool will be addressed in future work.

By observing the works proposed in the literature, it is possible to notice that there is a need to develop tools to automatically collect the results of metrics. After that, large portions of data need to be acquired from SOA projects in order to establish minimal parameters of quality for each metric.

10.4 Contributions in Bibliography Production

This work has resulted in the production of nine articles. Until this moment, this research has seven published articles. Six papers were accepted in international conferences and published in conference proceedings. These six papers were published in conferences that are currently classified by CAPES with score A2 or B1. The papers are listed in Table 30.

	Paper	Venue	Year	Reference
1	“Principles and Metrics to Improve Quality in SOA Applications”	DC ICEIS	2014	(FRANÇA; SOARES, 2014)
2	“SOAQM: Quality Model for SOA Applications based on ISO 25010”	ICEIS	2015	(FRANÇA; SOARES, 2015)
3	“An Experience of using SoaML for Modeling a Service-Oriented Architecture for Health Information Systems”	ICEIS	2015	(SILVA et al., 2015)
4	“A Case Study on SoaML to Design an Electronic Health Record Application Considering Integration of Legacy Systems”	COMPSAC	2016	(FRANÇA; LIMA; SOARES, 2016)
5	“Layered Implementation View of a SOA Based Electronic Health Record”	SEKE	2016	(LIMA et al., 2016)
6	“Development of an Electronic Health Record Application using a Multiple View Service Oriented Architecture”	ICEIS	2017	(FRANÇA; LIMA; SOARES, 2017)

Table 30 – Published articles - conference proceedings.

Another article was accepted on an International Journal (Table 31). This article was accepted on Journal of Software (JSW) in 2015. The paper was published in JSW in April 2016, Vol. 11, No. 4. Currently, JSW is a journal classified by CAPES with qualis B1.

	Article	Venue	Year	Reference
7	“Characterization of the Application of Service-Oriented Design Principles in Practice: A Systematic Literature Review”	JSW	2016	(SOARES; FRANÇA, 2016)

Table 31 – Published article - journal

In addition, two articles have not been published up to this moment. One paper was submitted to an International conference and the other article was submitted to Journal of Systems and Software (JSS) but the result notification has not been sent yet (Table 32).

	Article
8	“Architectural Design Decisions for SOA Applications with focus on Quality Attributes”
9	“Electronic health record system evaluation using users and experts opinion”

Table 32 – Unpublished articles

Bibliography

AGARWAL, N.; RATHOD, U. Defining Success for Software Projects: An Exploratory Revelation. **International Journal of Project Management**, v. 24, n. 4, p. 358–370, 2006.

AHMED, M. M.; LETCHMUNAN, S. A Systematic Literature Review on Challenges in Service Oriented Software Engineering. **International Journal of Software Engineering and Its Applications (IJSEIA)**, v. 9, n. 6, p. 173–186, 2015.

ALDRIS, A.; NUGROHO, A.; LAGO, P.; VISSER, J. Measuring the Degree of Service Orientation in Proprietary SOA Systems. In: **Seventh IEEE International Symposium on Service-Oriented System Engineering, SOSE 2013, San Francisco, CA, USA, March 25-28, 2013**. [S.l.: s.n.], 2013. p. 233–244.

ALI, M. S.; BABAR, M. A.; CHEN, L.; STOL, K.-J. A Systematic Review of Comparative Evidence of Aspect-Oriented Programming. **Information and Software Technology**, v. 52, n. 9, p. 871–887, 2010. ISSN 0950-5849.

ALLANQAWI, K. L. S. K.; KHADDAJ, S. A conceptual approach for assessing soa design defects' impact on quality attributes. In: **2013 12th International Symposium on Distributed Computing and Applications to Business, Engineering Science**. [S.l.: s.n.], 2013. p. 66–70.

ALONSO, G.; CASATI, F.; KUNO, H.; MACHIRAJU, V. **Web Services: Concepts, Architectures and Applications**. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2010.

AMELLER, D.; FRANCH, X. Service Level Agreement Monitor (SALMon). In: **Proceedings of the Seventh International Conference on Composition-Based Software Systems (ICCBSS)**. [S.l.: s.n.], 2008. p. 224–227.

ARDITO, C. et al. User-driven visual composition of service-based interactive spaces. **Journal of Visual Languages and Computing**, v. 25, n. 4, p. 278–296, 2014.

ARSANJANI, A. et al. SOMA: A Method for Developing Service-oriented Solutions. **IBM Systems Journal**, v. 47, n. 3, p. 377–396, jul. 2008.

BABAR, M. A.; DINGSYR, T.; LAGO, P.; VLIET, H. van. **Software Architecture Knowledge Management: Theory and Practice**. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2009.

BAGHDADI, Y.; AL-BULUSHI, W. A guidance process to modernize legacy applications for SOA. n. 1, July 2013.

BANSIYA, J.; DAVIS, C. G. A hierarchical model for object-oriented design quality assessment. **IEEE Transactions on Software Engineering**, v. 28, n. 1, p. 4–17, Jan 2002. ISSN 0098-5589.

BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software architecture in practice**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1998. ISBN 0-201-19930-0.

BELL, M. **Service-Oriented Modeling (SOA): Service Analysis, Design, and Architecture**. [S.l.]: Wiley, 2008. (NetLibrary, Inc). ISBN 9780470255704.

BELL, M. **SOA Modeling Patterns for Service Oriented Discovery and Analysis**. [S.l.]: Wiley Publishing, 2010. 223–305 p.

BENGURIA, G. et al. **Enterprise Interoperability**. [S.l.]: Springer London, 2007.

BENNETT, K. Legacy Systems: Coping with Success. **IEEE Softw.**, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 12, n. 1, p. 19–23, jan. 1995.

BERTOLINO, A.; INVERARDI, P.; MUCCINI, H. Software Architecture-based Analysis and Testing: A Look into Achievements and Future Challenges. **Computing**, v. 95, n. 8, p. 633–648, 2013.

BIANCHI, L. et al. Design of a restful web information system for drug prescription and administration. **IEEE J. Biomedical and Health Informatics**, v. 18, n. 3, p. 885–895, 2014.

BIEBERSTEIN, N.; LAIRD, R. G.; JONES, K.; MITRA, T. **Executing SOA: A Practical Guide for the Service-Oriented Architect**. 1st. ed. [S.l.]: IBM Press, 2008. ISBN 0132353741, 9780132353748.

BLOBEL, B. Ontology Driven Health Information Systems Architectures Enable pHealth for Empowered Patients. **Int. J. of Medical Informatics**, v. 80, n. 2, p. 17–25, 2011.

BOEHM, B. A View of 20th and 21st Century Software Engineering. In: **Proceedings of the 28th International Conference on Software Engineering**. [S.l.: s.n.], 2006. (ICSE '06), p. 12–29.

BOOCH, G. **Object-oriented Analysis and Design with Applications (2Nd Ed.)**. Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc., 1994.

BOOCH, G. The Economics of Architecture-First. **IEEE Software**, v. 24, n. 5, p. 18–20, 2007.

BOOKER, J. M.; MCNAMARA, L. A. Solving black box computation problems using expert knowledge theory and methods. **Rel. Eng. & Sys. Safety**, v. 85, n. 1-3, p. 331–340, 2004.

BOSCH, J. **Design and use of software architectures: adopting and evolving a product-line approach**. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000.

- BOSCH, J. Software Architecture: The Next Step. In: **EWSA**. [S.l.]: Springer, 2004. (Lecture Notes in Computer Science, v. 3047), p. 194–199.
- BOURQUE, P. et al. Fundamental principles of software engineering - a journey. **Journal of Systems and Software**, Elsevier Science Inc., New York, NY, USA, v. 62, n. 1, p. 59–70, maio 2002. ISSN 0164-1212.
- BRERETON, P.; KITCHENHAM, B. A.; BUDGEN, D.; TURNER, M.; KHALIL, M. Lessons from Applying the Systematic Literature Review Process Within the Software Engineering Domain. **Journal of Systems and Software**, v. 80, n. 4, p. 571–583, abr. 2007. ISSN 0164-1212.
- Brown, P. F. and Hamilton, R. M. B. A. **Reference Model for Service oriented Architecture**. [S.l.], 2006. Disponível em: <<http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf>>.
- BUSCHMANN, F.; MEUNIER, R.; ROHNERT, H.; SOMMERLAD, P.; STAL, M. **Pattern-oriented software architecture: a system of patterns**. New York, NY, USA: John Wiley & Sons, Inc., 1996.
- CARDOSO, J. et al. **Fundamentals of Service Systems**. [S.l.]: Springer International Publishing, 2015. ISBN 9783319231952.
- CARVALHO, M. B. et al. A Case Study on Service-Oriented Architecture for Serious Games. **Entertainment Computing**, v. 6, p. 1–10, 2015.
- CELLARY, W.; STRYKOWSKI, S. E-Government Based on Cloud Computing and Service-Oriented Architecture . In: DAVIES, J.; JANOWSKI, T. (Ed.). **ICEGOV**. [S.l.]: ACM, 2009. (ACM International Conference Proceeding Series, v. 322), p. 5–10.
- CHAPPEL, D. A. **Enterprise Service Bus**. United States of America: O'Reilly Media Inc, 2004.
- CHO, I.; KIM, J.; KIM, J.; KIM, H. Y.; KIM, Y. Design and implementation of a standards-based interoperable clinical decision support architecture in the context of the korean ehr. **International Journal of Medical Informatics**, v. 79, n. 9, p. 611–622, 2010.
- CHOI, S. W.; HER, J. S.; KIM, S. D. Qos metrics for evaluating services from the perspective of service providers. In: **ICEBE**. [S.l.]: IEEE Computer Society, 2007. p. 622–625.
- CHOI, S. W.; KIM, S. D. A Quality Model for Evaluating Reusability of Services in SOA. In: **Proc. of the 10th IEEE Int. Conf. on E-Commerce Technology (CEC 2008)**. [S.l.: s.n.], 2008. p. 293–298.
- COOKE, R. M. **Experts in uncertainty: opinion and subjective probability in science**. [S.l.]: Oxford University Press on Demand, 1991.
- CRESWELL, J. W. **Research Design: Qualitative, Quantitative, and Mixed Methods Approaches**. 4. ed. [S.l.]: Sage Publications Ltd., 2014.

- DAVIS, F. D.; BAGOZZI, R. P.; WARSHAW, P. R. User acceptance of computer technology: A comparison of two theoretical models. **Manage. Sci.**, INFORMS, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, v. 35, n. 8, p. 982–1003, ago. 1989.
- DEISSENBOECK, F.; JUERGENS, E.; LOCHMANN, K.; WAGNER, S. Software Quality Models: Purposes, Usage Scenarios and Requirements. In: **Proceedings of the WoSQ-09**. [S.l.]: IEEE CS Press, 2009.
- DELGADO, A.; RUIZ, F.; GUZMÁN, I. G. R. de; PIATTINI, M. Business Process Service Oriented Methodology (BPSOM) with Service Generation in SoaML. In: **Proceedings of the Advanced Information Systems Engineering - 23rd International Conference, CAiSE 2011**. [S.l.: s.n.], 2011. p. 672–680.
- DIRGAHAYU, T.; QUARTEL, D.; SINDEREN, M. van. Designing interaction behaviour in service-oriented enterprise application integration. In: **Proceedings of the 2008 ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2008. (SAC '08), p. 1048–1054. ISBN 978-1-59593-753-7.
- DU, X.; SONG, W.; MUNRO, M. A method for transforming existing web service descriptions into an enhanced semantic web service framework. In: _____. **Information Systems Development: Towards a Service Provision Society**. Boston, MA: Springer US, 2010. p. 217–226.
- EASTERBROOK, S.; SINGER, J.; STOREY, M.; DAMIAN, D. **Selecting Empirical Methods for Software Engineering Research**. [S.l.]: Springer, 2007.
- EL-SAPPAGH, S. H.; EL-MASRI, S. A Distributed Clinical Decision Support System Architecture. **Journal of King Saud University - Computer and Information Sciences**, v. 26, n. 1, p. 69–78, 2014.
- ENGSTROM, E.; RUNESON, P.; SKOGLUND, M. A Systematic Review on Regression Test Selection Techniques. **Information and Software Technology**, v. 52, n. 1, p. 14–30, 2010. ISSN 0950-5849.
- ERL, T. **Service-oriented architecture concepts, technology, and design**. [S.l.]: Prentice Hall Professional Technical Reference, 2005.
- ERL, T. **SOA Principles of Service Design**. [S.l.: s.n.], 2007. (The Prentice Hall Service-Oriented Computing Series from Thomas Erl).
- ERL, T. **Service-Oriented Architecture: Analysis and Design for Services and Microservices**. 2. ed. Boston: Prentice Hall, 2017. (Prentice Hall Service Technology Series). ISBN 978-0-13-385858-7.
- FABIAN, B.; ERMAKOVA, T.; JUNGHANNS, P. Collaborative and Secure Sharing of Healthcare Data in Multi-Clouds. **Information Systems**, v. 48, p. 132–150, 2015.
- FRANÇA, J. M. S.; LIMA, J. de S.; SOARES, M. S. A case study on soaml to design an electronic health record application considering integration of legacy systems. In: **40th IEEE Annual Computer Software and Applications Conference, COMPSAC 2016, Atlanta, GA, USA, June 10-14, 2016**. [S.l.]: IEEE Computer Society, 2016. p. 353–358.

FRANÇA, J. M. S.; LIMA, J. de S.; SOARES, M. S. Development of an electronic health record application using a multiple view service oriented architecture. In: **ICEIS 2017 - Proceedings of the 17th International Conference on Enterprise Information Systems, Porto, Portuga, April, 2017**. [S.l.: s.n.], 2017.

FRANÇA, J. M. S.; SOARES, M. S. Principles and metrics to improve quality in SOA applications. In: **ICEIS 2014 - Proceedings of the 17th International Conference on Enterprise Information Systems, Volume 2, Lisboa, Portugal, April, 2014**. [S.l.: s.n.], 2014.

FRANÇA, J. M. S.; SOARES, M. S. SOAQM: quality model for SOA applications based on ISO 25010. In: **ICEIS 2015 - Proceedings of the 17th International Conference on Enterprise Information Systems, Volume 2, Barcelona, Spain, 27-30 April, 2015**. [S.l.: s.n.], 2015. p. 60–70.

GADGIL, H.; FOX, G.; PALLICKARA, S.; PIERCE, M. Scalable, Fault-tolerant Management in a Service Oriented Architecture. In: **Proceedings of the 16th International Symposium on High Performance Distributed Computing**. [S.l.: s.n.], 2007. (HPDC '07), p. 235–236.

GARCÍA-SÁNCHEZ, P. et al. Developing Services in a Service Oriented Architecture for Evolutionary Algorithms. In: **Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation**. New York, NY, USA: ACM, 2013. (GECCO '13 Companion), p. 1341–1348.

GARLAN, D. Software Architecture: A Travelogue. In: **Proc. of the on Future of Software Engineering**. [S.l.: s.n.], 2014. (FOSE 2014), p. 29–39.

GARLAN, D. et al. **Documenting Software Architectures: Views and Beyond**. 2nd. ed. [S.l.]: Addison-Wesley Professional, 2010. ISBN 0321552687, 9780321552686.

GAZZARATA, G.; GAZZARATA, R.; GIACOMINI, M. A Standardized SOA Based Solution to Guarantee the Secure Access to EHR. **Procedia Computer Science**, v. 64, p. 1124–1129, 2015.

GEHLERT, A.; METZGER, A. **Quality Reference Model for SBA**. [S.l.], 2008.

GLASS, R. L.; RAMESH, V.; VESSEY, I. An Analysis of Research in Computing Disciplines. **Communications of the ACM**, v. 47, n. 6, p. 89–94, 2004.

GOEB, A.; LOCHMANN, K. A software quality model for SOA. In: **Proceedings of the 8th international workshop on Software quality**. New York, NY, USA: ACM, 2011. (WoSQ '11), p. 18–25.

GONZÁLEZ, M. Á. C.; GARCÍA-PENALVO, F. J.; FORMENT, M. A.; MAYOL, E.; LLAMAS, C. F. Implementation and Design of a Service-Based Framework to Integrate Personal and Institutional Learning Environments. **Science of Computer Programming**, v. 88, p. 41–53, 2014.

GRANELL, C.; DÍAZ, L.; GOULD, M. Service-oriented applications for environmental models: Reusable geospatial services. **Environmental Modelling and Software**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 25, n. 2, p. 182–198, fev. 2010. ISSN 1364-8152.

- GROUP, O. **SOA Reference Architecture**. [S.l.], 2001. Disponível em: <http://www.opengroup.org/soa/source-book/soa_refarch/p1.htm>.
- GROUP, O. **Service Oriented Architecture: What Is SOA?** [S.l.], 2014. Disponível em: <<http://www.opengroup.org/soa/source-book/soa/soa.htm>>.
- GU, Q.; LAGO, P. SOA Process Decisions: New Challenges in Architectural Knowledge Modeling. In: **Proceedings of the 3rd International Workshop on Sharing and Reusing Architectural Knowledge**. [S.l.]: ACM, 2008. (SHARK '08), p. 3–10.
- GU, Q.; LAGO, P. Exploring service-oriented system engineering challenges: a systematic literature review. **Service Oriented Computing and Applications**, v. 3, n. 3, p. 171–188, 2009.
- GUERMAZI, E.; BEN-ABDALLAH, H.; AYED, M. B. Modeling a secure cloud data warehouse with soaml. In: **2015 11th International Conference on Information Assurance and Security (IAS)**. [S.l.: s.n.], 2015. p. 55–60.
- HARMAN, M.; MANSOURI, S. A.; ZHANG, Y. Search-based Software Engineering: Trends, Techniques and Applications. **ACM Comput. Surv.**, v. 45, n. 1, p. 11:1–11:61, 2012.
- HIRSCH, F.; KEMP, J.; ILKKA, J. **Mobile Web Services: Architecture and Implementation**. [S.l.]: Wiley, 2006. ISBN 9780470015964.
- HUANG, C.-Y.; LEUNG, H.; LEUNG, W.-H. F.; MIZUNO, O. Software quality assurance methodologies and techniques. **Advances in Software Engineering**, 2012.
- HUHNS, M. N.; SINGH, M. P. Service-Oriented Computing: Key Concepts and Principles. **IEEE Internet Computing**, v. 9, n. 1, p. 75–81, 2005.
- HäYRINEN, K.; SARANTO, K.; NYKÄNEN, P. Definition, structure, content, use and impacts of electronic health records: A review of the research literature. **International Journal of Medical Informatics**, v. 77, n. 5, p. 291 – 304, 2008.
- IEEE Computer Society. **Software Engineering Body of Knowledge (SWEBOK)**. [S.l.], 2004. Disponível em: <<http://www.swebok.org/>>.
- IONITA, A. D.; MOCANU, M.; CIOLOFAN, S. N. Modeling with soaml applied for warning and water management services. In: **2013 19th International Conference on Control Systems and Computer Science**. [S.l.: s.n.], 2013. p. 624–627.
- ISO, I. O. f. S. **ISO 20514:2005 Health informatics, Electronic health record - Definition, scope and context**. [S.l.], 2005.
- ISO/IEC. **Software Engineering - Product Quality, ISO/IEC 9126**. [S.l.], 1991.
- ISO/IEC, International Organization for Standardization. **ISO/IEC 25010 - Systems and Software Engineering - Systems and Software Quality Requirements and Evaluation (SQuaRE) - System and Software Quality Models**. [S.l.], 2011.
- ISO/IEC, International Organization for Standardization. **ISO/IEC 42010 - Systems and Software Engineering - Architecture Description**. [S.l.], 2011.

JACOBSON, I. **Object Oriented Software Engineering: A Use Case Driven Approach**. [S.l.]: Addison-Wesley, 1992.

JALOTE, P. **An Integrated Approach to Software Engineering**. 2nd. ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1997. ISBN 0387948996.

JANSEN, A.; BOSCH, J. Software Architecture As a Set of Architectural Design Decisions. In: **Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture**. [S.l.: s.n.], 2005. (WICSA '05), p. 109–120.

KABIR, M. A.; HAN, J.; COLMAN, A. W. Sociotelematics: Harnessing social interaction-relationships in developing automotive applications. **Pervasive and Mobile Computing**, v. 14, p. 129–146, 2014.

KAMSTIES, E. Understanding Ambiguity in Requirements Engineering. In: AURUM, A.; WOHLIN, C. (Ed.). **Engineering and Managing Software Requirements**. Berlin, Germany: Springer-Verlag, 2005. p. 245–266.

KANNAN, K.; BHAMIDIPATY, A.; NARENDRA, N. C. Design time validation of service orientation principles using design diagrams. **Annual (Services Research and Innovation Institute) SRII Global Conference**, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 795–804, 2011.

JACOBSEN, H.-A.; WANG, Y.; HUNG, P. (Ed.). **SATE-Service Boundary and Abstraction Threshold Estimation for Efficient Services Design**. [S.l.]: IEEE, 2011.

KASUNIC, W. A. M. Technical note, **Measuring Systems Interoperability: Challenges and Opportunities**. 2004.

KHADKA, R.; REIJNDERS, G.; SAEIDI, A.; JANSEN, S.; HAGE, J. A method engineering based legacy to soa migration method. In: **Proceedings of the 2011 27th IEEE International Conference on Software Maintenance**. Washington, DC, USA: IEEE Computer Society, 2011. (ICSM '11), p. 163–172.

KHADKA, R.; SAEIDI, A.; IDU, A.; HAGE, J.; JANSEN, S. Legacy to soa evolution: A systematic literature review. In: **In A. D. Ionita, M. Litoiu, G. Lewis (Eds.) Migrating Legacy Applications: Challenges in Service Oriented Architecture and Cloud Computing Environments**. [S.l.: s.n.], 2013.

KHOSHKBARFOROUSHHA, A.; JAMSHIDI, P.; SHAMS, F. A metric for composite service reusability analysis. In: **Proceedings of the 2010 ICSE Workshop on Emerging Trends in Software Metrics**. New York, NY, USA: ACM, 2010. (WETSoM '10), p. 67–74.

KING, G.-H. et al. A high-performance multi-user service system for financial analytics based on web service and gpu computation. **International Symposium on Parallel and Distributed Processing with Applications (ISPA)**, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 327–333, 2010.

KITCHENHAM, B. **Procedures for Undertaking Systematic Reviews**. [S.l.], 2004.

- KITCHENHAM, B. et al. Systematic Literature Reviews in Software Engineering - A Systematic Literature Review. **Information and Software Technology**, v. 51, n. 1, p. 7–15, 2009. Special Section - Most Cited Articles in 2002 and Regular Research Papers.
- KITCHENHAM, B.; BRERETON, P. A Systematic Review of Systematic Review Process Research in Software Engineering. **Information & Software Technology**, v. 55, n. 12, p. 2049–2075, 2013.
- KITCHENHAM, B.; PFLEEGER, S. L. Software quality: The elusive target. **IEEE Softw.**, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 13, n. 1, p. 12–21, 1996.
- KLÄS, M.; LOCHMANN, K.; HEINEMANN, L. Evaluating a quality model for software product assessments – a case study. In: **Proceedings of 4. Workshop zur Software-Qualitätsmodellierung und -bewertung (SQMB'11)**. [S.l.: s.n.], 2011.
- KOSEK, A. M.; GEHRKE, O. Model-driven development of smart grid services using soaml. In: **IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society**. [S.l.: s.n.], 2014. p. 3563–3569.
- KRAMMER, A.; HEINRICH, B.; HENNEBERGER, M.; LAUTENBACHER, F. Granularity of services - an economic analysis. **Business and Information Systems Engineering**, v. 3, n. 6, p. 345–358, 2011.
- KRUCHTEN, P. The 4+1 view model of architecture. **IEEE Software**, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 12, n. 6, p. 42–50, nov. 1995.
- LANE, S.; RICHARDSON, I. Process Models for Service-Based Applications: A Systematic Literature Review. **Information and Software Technology**, v. 53, n. 5, p. 424 – 439, 2011. Special Section on Best Papers from {XP2010}.
- LANMAN, J. T.; DARBIN, R.; RIVERA, J.; CLEMENTS, P. C.; KRUEGER, C. W. The challenges of applying service orientation to the u.s. army's live training software product line. In: KISHI, T.; JARZABEK, S.; GNESI, S. (Ed.). **SPLC**. [S.l.]: ACM, 2013. p. 244–253.
- LEGNER, C.; VOGEL, T. Design principles for b2b services - an evaluation of two alternative service designs. In: **IEEE International Conference on Services Computing**. [S.l.]: IEEE Computer Society, 2007. p. 372–379.
- LI, M.; SMIDTS, C. A ranking of software engineering measures based on expert opinion. **IEEE Trans. Softw. Eng.**, IEEE Press, Piscataway, NJ, USA, v. 29, n. 9, p. 811–824, set. 2003. ISSN 0098-5589.
- LIKERT, R. A technique for the measurement of attitudes. **Archives of Psychology**, v. 22, n. 140, p. 1–55, 1932.
- LIMA, J. de S.; FRANÇA, J. M. S.; MENEZES, J. S. S. de; OLIVEIRA, A.; SOARES, M. S. Layered implementation view of a SOA based electronic health record. In: **The 28th International Conference on Software Engineering and Knowledge Engineering, SEKE 2016, Redwood City, San Francisco Bay, USA, July 1-3, 2016**. [S.l.]: KSI Research Inc. and Knowledge Systems Institute Graduate School, 2016. p. 323–328.

LISKOV, B.; GUTTAG, J. **Program Development in Java: Abstraction, Specification, and Object-Oriented Design**. [S.l.]: ADDISON WESLEY Publishing Company Incorporated, 2001. (Pearson educación).

LIU, L.-L. Design principles and measurable service oriented usability. In: **IEEE International Conference on Service-Oriented Computing and Applications, (SOCA)** . [S.l.]: IEEE, 2009. p. 1–4.

LIU, R.; WU, F.; PATNAIK, Y.; KUMARAN, S. Business entities: An soa approach to progressive core banking renovation. **IEEE International Conference on Services Computing (SCC)**, IEEE Computer Society, Los Alamitos, CA, USA, p. 466–473, 2009.

LOCHMANN, K.; HEINEMANN, L. Integrating quality models and static analysis for comprehensive quality assessment. In: **Proceedings of the 2Nd International Workshop on Emerging Trends in Software Metrics**. New York, NY, USA: ACM, 2011. (WETSOM '11), p. 5–11.

MAHDAVI-HEZAVEHI, S.; GALSTER, M.; AVGERIOU, P. Variability in Quality Attributes of Service-Based Software Systems: A Systematic Literature Review. **Information and Software Technology**, v. 55, n. 2, p. 320 – 343, 2013.

MARANGUNIĆ, N.; GRANIĆ, A. Technology acceptance model: A literature review from 1986 to 2013. **Univers. Access Inf. Soc.**, Springer-Verlag, Berlin, Heidelberg, v. 14, n. 1, p. 81–95, mar. 2015. ISSN 1615-5289.

MOHAMMADI, M.; MUKHTAR, M. A Review of {SOA} Modeling Approaches for Enterprise Information Systems. **Procedia Technology**, v. 11, p. 794–800, 2013.

MONSIEUR, G.; SNOECK, M.; LEMAHIEU, W. Managing data dependencies in service compositions. **Journal of Systems and Software**, v. 85, n. 11, p. 2604–2628, 2012.

MOOR, G. D. et al. Using Electronic Health Records for Clinical Research: The Case of the EHR4CR Project. **Journal of Biomedical Informatics**, v. 53, p. 162–173, 2015.

MOSLEH, A. et al. **Methods for the Elicitation and Use of Expert Opinion in Risk Assessment: Phase I, a Critical Evaluation and Directions for Future Research**. [S.l.]: Division of Reactor Accident Analysis, Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission, 1987.

Nadanam, P. and Rajmohan, R. QoS Evaluation for Web Services in Cloud Computing. In: **Proceedings of the Third International Conference on Computing Communication and Networking Technologies (ICCCNT)**. [S.l.: s.n.], 2012. p. 1–8.

NANCE, R.; ARTHUR, J. **Managing Software Quality: A Measurement Framework for Assessment and Prediction**. [S.l.]: Springer London, 2012. ISBN 9781852333935.

NGUYEN, L.; BELLUCCI, E.; NGUYEN, L. T. Electronic health records implementation: An evaluation of information system impact and contingency factors. **I. J. Medical Informatics**, v. 83, n. 11, p. 779–796, 2014.

OASIS. **Quality Model for Web Services (WSQM2.0)**. [S.l.], 2005. Disponível em: <<http://www.clip.dia.fi.upm.es/Projects/S-CUBE/papers/oasis05:WSQM-2.0.pdf>>.

OASIS. **Reference Architecture for Service Oriented Architecture Foundation**. [S.l.], 2012. Disponível em: <<http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra.pdf>>.

O'BRIEN, L.; BREBNER, P.; GRAY, J. Business Transformation to SOA: Aspects of the Migration and Performance and QoS Issues. In: **Proceedings of the 2nd International Workshop on Systems Development in SOA Environments**. [S.l.: s.n.], 2008. (SDSOA '08), p. 35–40.

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

OGRINZ, M. **Mashup Patterns: Designs and Examples for the Modern Enterprise**. 1. ed. [S.l.]: Addison-Wesley Professional, 2009.

OMG. **Service Oriented Architecture Modeling Language (SoaML) Specification**. [S.l.], 2012. Disponível em: <<http://www.omg.org/spec/SoaML/1.0/PDF>>.

OMG. **Unified Modeling Language (UML) Specification**. [S.l.], 2015. Disponível em: <<http://www.omg.org/spec/UML/2.5/PDF>>.

Open Group. **The Open Group SOA Ontology**. [S.l.], 2010. Disponível em: <<http://www.opengroup.org/standards/soa>>.

ORIOLE, M.; MARCO, J.; FRANCH, X. Quality Models for Web Services: A Systematic Mapping. **Information and Software Technology**, v. 56, n. 10, p. 1167–1182, 2014.

PACHECO, C.; GARCIA, I. A Systematic Literature Review of Stakeholder Identification Methods in Requirements Elicitation. **Journal of Systems and Software**, v. 85, n. 9, p. 2171–2181, 2012. ISSN 0164-1212. Selected papers from the 2011 Joint Working IEEE/IFIP Conference on Software Architecture (WICSA 2011).

PALACIOS, M.; GARCIA-FANJUL, J.; TUYA, J. Testing in Service Oriented Architectures with Dynamic Binding: A Mapping Study. **Information and Software Technology**, v. 53, n. 3, p. 171–189, 2011.

PAPAZOGLU, M. P. Service -oriented computing: Concepts, characteristics and directions. In: **Proceedings of the Fourth International Conference on Web Information Systems Engineering**. Washington, DC, USA: IEEE Computer Society, 2003. (WISE '03).

PAPAZOGLU, M. P.; TRAVERSO, P.; DUSTDAR, S.; LEYMAN, F. Service-Oriented Computing: State of the Art and Research Challenges. **Computer**, v. 40, n. 11, p. 38–45, 2007.

PAPAZOGLU, M. P.; TRAVERSO, P.; DUSTDAR, S.; LEYMAN, F. Service-Oriented Computing: a Research Roadmap. **International Journal Cooperative Information System**, v. 17, n. 2, p. 223–255, 2008.

- PAUTASSO, C.; WILDE, E. Why is the web loosely coupled?: A multi-faceted metric for service design. In: **Proceedings of the 18th International Conference on World Wide Web**. New York, NY, USA: ACM, 2009. (WWW '09), p. 911–920.
- PEREPLETCHIKOV, M.; RYAN, C. A controlled experiment for evaluating the impact of coupling on the maintainability of service-oriented software. **IEEE Transactions on Software Engineering**, v. 37, n. 4, p. 449–465, 2011.
- PETRE, M. Uml in practice. In: **Proceedings of the 2013 International Conference on Software Engineering**. Piscataway, NJ, USA: IEEE Press, 2013. (ICSE '13), p. 722–731. ISBN 978-1-4673-3076-3. Disponível em: <<http://dl.acm.org/citation.cfm?id=2486788.2486883>>.
- PFLEEGER, S. L. **Software Engineering: Theory and Practice**. 2nd. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.
- PFLEEGER, S. L. Solid software: Is it rocket science? In: KONTIO, J.; CONRADI, R. (Ed.). **Software Quality — ECSQ 2002: Quality Connection — 7th European Conference on Software Quality Helsinki, Finland, June 9–13, 2002 Proceedings**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. p. 7–12. ISBN 78-3-540-47984-0.
- PRESSMAN, R. S. **Software Engineering: A Practitioner's Approach**. 7th. ed. [S.l.]: McGraw-Hill Higher Education, 2009.
- PREVE, N. **Grid Computing: Towards a Global Interconnected Infrastructure**. [S.l.]: Springer, London, 2011. (Computer Communications and Networks).
- RHAZALI, Y.; HADI, Y.; MOULOUDI, A. A model transformation in mda from cim to pim represented by web models through soaml and ifml. In: **2016 4th IEEE International Colloquium on Information Science and Technology (CiSt)**. [S.l.: s.n.], 2016. p. 116–121.
- ROZANSKI, N.; WOODS, E. **Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives**. [S.l.]: Pearson Education, 2011. ISBN 9780132906128.
- RUMBAUGH, J.; BLAHA, M.; PREMERLANI, W.; EDDY, F.; LORENSEN, W. **Object-oriented Modeling and Design**. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1991. ISBN 0-13-629841-9.
- RUNESON, P.; HÖST, M. Guidelines for Conducting and Reporting Case Study Research in Software Engineering. **Empirical Software Engineering**, Kluwer Academic Publishers, v. 14, n. 2, p. 131–164, abr. 2009. ISSN 1382-3256.
- RUNESON, P.; HOST, M.; RAINER, A.; REGNELL, B. **Case Study Research in Software Engineering: Guidelines and Examples**. 1. ed. [S.l.]: Wiley Publishing, 2012.
- SANDERS, J.; CURRAN, E. **Software quality: a framework for success in software development and support**. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1994.

SAVOLAINEN, P.; AHONEN, J. J.; RICHARDSON, I. Software Development Project Success and Failure from the Supplier's Perspective: A Systematic Literature Review. **International Journal of Project Management**, v. 30, n. 4, p. 458–469, 2012.

SHAHIN, M.; LIANG, P.; KHAYYAMBASHI, M.-R. Architectural Design Decision: Existing Models and Tools. In: **WICSA/ECSA**. [S.l.]: IEEE, 2009. p. 293–296.

SHAW, M. What Makes Good Research in Software Engineering? **STTT**, v. 4, n. 1, p. 1–7, 2002.

SILVA, F. G.; MENEZES, J. S. S. de; LIMA, J. de S.; FRANÇA, J. M. S.; SOARES, M. S. An experience of using soaml for modeling a service-oriented architecture for health information systems. In: **ICEIS 2015 - Proceedings of the 17th International Conference on Enterprise Information Systems, Volume 3, Barcelona, Spain, 27-30 April, 2015**. [S.l.: s.n.], 2015. p. 322–327.

SMITE, D.; WOHLIN, C.; GORSCHKE, T.; FELDT, R. Empirical Evidence in Global Software Engineering: A Systematic Review. **Empirical Software Engineering**, Springer US, v. 15, n. 1, p. 91–118, 2010.

SOARES, M. S.; FRANÇA, J. M. S. Characterization of the application of service-oriented design principles in practice: A systematic literature review. **JSW**, v. 11, n. 4, p. 403–417, 2016.

SOARES, M. S.; VRANCKEN, J.; VERBRAECK, A. User Requirements Modeling and Analysis of Software-intensive Systems. **Journal of Systems and Software**, v. 84, n. 2, p. 328–339, fev. 2011. ISSN 0164-1212.

SOMMERVILLE, I. **Software Engineering**. 9. ed. Essex, UK: Addison Wesley, 2015.

Steen, M.W.A. and Strating, O. and Lankhorst, T. and ter Doest, H.W.L. Service-Oriented Enterprise Architecture. In: _____. London, UK: Hershey, 2005. cap. Service-Oriented Software System Engineering: Challenges and Practice, p. 132–154.

STUBBINGS, G.; POLOVINA, S. Levering object-oriented knowledge for service-oriented proficiency - enhancing service capability in developers. **Computing**, v. 95, p. 817–835, 2013.

SURENDRAN, P. Technology acceptance model: A survey of literature. **International Journal of Business and Social Research**, v. 2, n. 4, p. 175–178, 2013. ISSN 2164-2559.

TIAN, J. **Software Quality Engineering: Testing, Quality Assurance, and Quantifiable Improvement**. [S.l.]: Wiley, 2005. (Wiley - IEEE). ISBN 9780471722335.

TIAN, Y.; HUANG, M. Enhance discovery and retrieval of geospatial data using soa and semantic web technologies. **Expert Systems with Applications**, v. 39, n. 16, p. 12522–12535, 2012.

TOFAN, D.; GALSTER, M.; AVGERIOU, P.; SCHUITEMA, W. Past and Future of Software Architectural Decisions - A Systematic Mapping Study. **Information & Software Technology**, v. 56, n. 8, p. 850–872, 2014.

- TOUNSI, I.; KACEM, M. H.; KACEM, A. H.; DRIRA, K.; MEZGHANI, E. Towards an approach for modeling and formalizing soa design patterns with event-b. In: **Proceedings of the 28th Annual ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2013. p. 1937–1938.
- TRAORE, B. B.; KAMSU-FOGUEM, B.; TANGARA, F. Integrating MDA and SOA for Improving Telemedicine Services. **Telematics and Informatics**, v. 33, n. 3, p. 733–741, 2016.
- TYREE, J.; AKERMAN, A. Architecture Decisions: Demystifying Architecture. **IEEE Software**, v. 22, p. 19–27, 2005.
- VESCOUKIS, V.; DULAMIS, N. D. Disaster management evaluation and recommendation. **Conference in Games and Virtual Worlds for Serious Applications (VS-GAMES)**, IEEE Computer Society, Los Alamitos, CA, USA, p. 244–249, 2011.
- VOELZ, D.; GOEB, A. What is different in quality management for soa? In: . [S.l.]: IEEE Computer Society, 2010. p. 47–56.
- W3C. **Simple Object Access Protocol (SOAP) 1.1. W3C Note**. [S.l.], 2016. Disponível em: <<http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>>.
- W3C. **Web Services Choreography Description Language (WSCDL) Version 1.0**. [S.l.], 2016. Disponível em: <<http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/>>.
- W3C. **Web Services Description Language (WSDL) 1.1. W3C Note**. [S.l.], 2016. Disponível em: <<http://www.w3.org/TR/wsdl>>.
- WOHLIN, C.; PRIKLADNICKI, R. Systematic Literature Reviews in Software Engineering. **Information and Software Technology**, v. 55, n. 6, p. 919–920, 2013.
- WU, B.; CHI, C.-H.; CHEN, Z.; GU, M.; SUN, J. Workflow-based resource allocation to optimize overall performance of composite services. **Future Generation Computer Systems**, v. 25, n. 3, p. 199 – 212, 2009.
- XIAO-JUN, W. Metrics for evaluating coupling and service granularity in service oriented architecture. In: **Proc. of the International Conference on Information Engineering and Computer Science (ICIECS)**. [S.l.: s.n.], 2010. p. 1–4.
- XING, Y.; YAO, E. Remote collaborative experiments based on service-oriented architecture(soa). In: **Proc. of the International Conference on Signal Processing Systems (ICSPPS)**. [S.l.: s.n.], 2010. v. 2, p. 605–608.
- YAMANY, H. F. E.; CAPRETZ, M. A. M.; ALLISON, D. S. Intelligent security and access control framework for service-oriented architecture. **Information and Software Technology**, Butterworth-Heinemann, Newton, MA, USA, v. 52, n. 2, p. 220–236, fev. 2010. ISSN 0950-5849.
- YIN, B.; YANG, H.; FU, P.; CHEN, X. A Semantic Web Services Discovery Algorithm Based on QoS Ontology. In: . [S.l.]: Springer, 2010. (Lecture Notes in Computer Science, v. 6335), p. 166–173.

ZHANG, H.; BABAR, M. A. Systematic Reviews in Software Engineering - An Empirical Investigation. **Information and Software Technology**, v. 55, n. 7, p. 1341–1354, 2013.

ZIMMERMANN, O.; GSCHWIND, T.; KÜSTER, J.; LEYMAN, F.; SCHUSTER, N. Reusable Architectural Decision Models for Enterprise Application Development. In: **Proceedings of the Quality of Software Architectures 3rd International Conference on Software Architectures, Components, and Applications**. [S.l.: s.n.], 2007. p. 15–32.

Appendix

APPENDIX A

Questionnaire SOAQM

Service Oriented Architecture Quality Model based on ISO 25010

Questionário - SOAQM

Este questionário tem como objetivo avaliar quais atributos de qualidade propostos pela ISO 25010 são relevantes/aplicáveis para SOA (Service Oriented Architecture).

A ISO 25010 estabelece atributos de qualidade (8 características e 31 subcaracterísticas) como mostra a figura 1.

De acordo com a ISO, os atributos de qualidade podem ser aplicados para qualquer tipo de software. Contudo, sabemos que o contexto de SOA pode não abranger todos os atributos propostos. A proposta dessa pesquisa é indicar quais atributos de qualidade são relevantes para SOA e quais não são.

Cada item desse formulário informará um atributo de qualidade proposto pela ISO 25010. Além disso, disponibilizamos uma breve descrição de cada atributo para facilitar o desenvolvimento das respostas. Cada atributo deve ser analisado com relação à sua relevância para SOA. A resposta corresponderá a um nível de importância (1 a 5) de cada atributo para SOA.

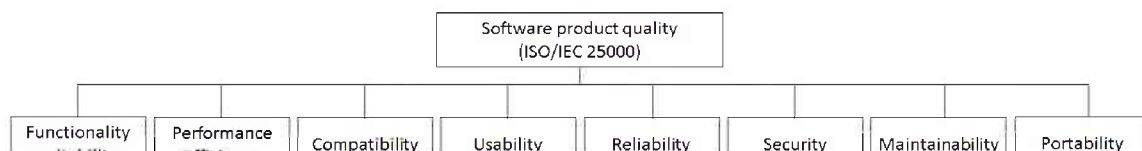
O significado dos números são:

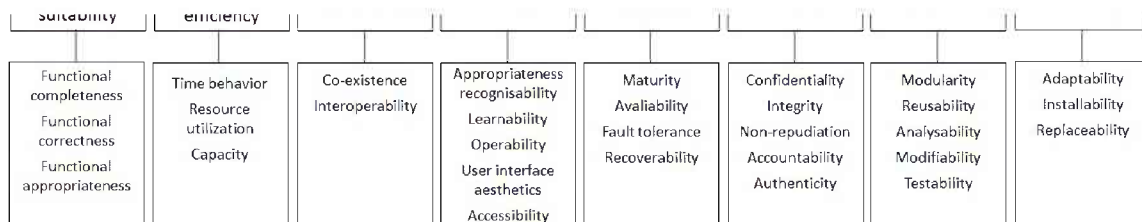
- 1 - Não Importante (Irrelevante);
- 2 - Pouco Importante;
- 3 - Neutro;
- 4 - Importante;
- 5 - Muito Importante.

Observação: Cada característica possui um campo de texto que pode ser usado para comentar caso queira explicar o motivo de cada resposta.

* Required

Figura 1 - Modelo de qualidade para qualidade externa e interna por ISO 25010





Nome

Escreva seu nome para identificação nas respostas.

Your answer

Profissão

Your answer

Funcionalidade

Capacidade do produto de software de prover funções que atendam às necessidades explícitas e implícitas, quando o software estiver sendo utilizado sob condições especificadas.

Your answer

Funcionalidade: Completude *

Capacidade do produto de software de prover um conjunto de funções que abrange todas as tarefas e objetivos específicos de usuários.

1

2

3

4

5

☐
☐
☐
☐
☐

Funcionalidade: Acurácia *

Capacidade do produto de software de prover, com o grau de precisão necessário, resultados ou efeitos corretos ou conforme acordados.

1

2

3

4

5

☐
☐
☐
☐
☐


Funcionalidade: Adequação *

Capacidade do produto de software de prover um conjunto apropriado de funções para tarefas e objetivos do usuário especificados.

1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Eficiência

Capacidade do produto de software de apresentar desempenho apropriado, relativo à quantidade de recursos usados, sob condições especificadas.

Your answer

Eficiência: Comportamento em relação ao tempo *

Capacidade do produto de software de fornecer tempos de resposta e de processamento, além de taxas de transferência, apropriados, quando o software executa suas funções, sob condições estabelecidas.

1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Eficiência: Utilização de recursos *

Capacidade do produto de software de usar tipos e quantidades apropriados de recursos, quando o software executa suas funções sob condições estabelecidas.

1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Eficiência: Capacidade *

Grau em que os limites máximos de um produto ou sistema requisitos parâmetro se encontram.

1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>



Compatibilidade

Capacidade de um produto, componente ou sistema pode trocar informações com outros produtos, sistemas ou componentes, e / ou realizar suas funções necessárias, ao compartilhar a mesma hardware ou ambiente de software.

Your answer

Compatibilidade: Co-existência *

Grau em que um produto pode desempenhar as suas funções de forma eficiente necessários ao compartilhar um ambiente e dos recursos comum com outros produtos, sem impacto negativo sobre qualquer outro produto.

1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Compatibilidade: Interoperabilidade *

Capacidade do produto de software de interagir com um ou mais sistemas especificados.

1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Usabilidade

Capacidade do produto de software de ser compreendido, aprendido, operado e atraente ao usuário, quando usado sob condições especificadas.

Your answer

Usabilidade: Inteligibilidade *

Capacidade do produto de software de possibilitar ao usuário compreender se o software é apropriado e como ele pode ser usado para tarefas e condições de uso específicas.

1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>



Usabilidade: Compreensão *

Capacidade do produto de software de possibilitar ao usuário aprender sua aplicação.

1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Usabilidade: Operacionalidade *

Capacidade do produto de software de possibilitar ao usuário operá-lo e controlá-lo.

1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Usabilidade: Proteção de erro do usuário *

Proteção de erro do usuário está relacionado com o grau em que um sistema protege os usuários contra a cometer erros.

1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Usabilidade: Estética da interface do usuário *

Estética da interface do usuário refere-se ao grau em que uma interface de usuário permite a interação agradável e satisfatória para o usuário.

1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Usabilidade: Acessibilidade *

A acessibilidade está relacionado com o grau a que um produto ou sistema pode ser usado por pessoas com a gama mais ampla de características e capacidades para conseguir um objectivo especificado num determinado contexto de uso.

1	2	3	4	5
---	---	---	---	---



☐ ☐ ☐ ☐ ☐

Confiabilidade

Capacidade do produto de software de manter um nível de desempenho especificado, quando usado em condições especificadas.

Your answer

Confiabilidade: Maturidade *

Grau em que um sistema, produto ou componente satisfaz necessidades de confiabilidade em operação normal.

1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Confiabilidade: Disponibilidade *

Disponibilidade é a capacidade de um produto de software de estar pronto para executar uma função requisitada num dado momento, sob condições especificadas de uso.

1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Confiabilidade: Tolerância a falhas *

Capacidade do produto de software de manter um nível de desempenho especificado em casos de defeitos no software ou de violação de sua interface especificada.

1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Confiabilidade: Recuperabilidade *

Capacidade do produto de software de restabelecer seu nível de desempenho especificado e recuperar os dados diretamente afetados no caso de uma falha.

1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>



Segurança

A segurança é o grau em que um produto ou sistema protege informações e dados, de modo que as pessoas ou outros produtos ou sistemas têm o grau de acesso de dados apropriado para os seus tipos e níveis de autorização.

Your answer

Segurança: Confidencialidade *

Capacidade do produto de software de proteger informações e dados, de forma que pessoas ou sistemas não autorizados não possam lê-los nem modificá-los e que não seja negado o acesso às pessoas ou sistemas autorizados.

1

☐

2

☐

3

☐

4

☐

5

☐

Segurança: Integridade *

Integridade refere-se ao grau em que um sistema, o produto ou componente impede o acesso não autorizado ou modificação de, software ou dados.

1

☐

2

☐

3

☐

4

☐

5

☐

Segurança: Não-repúdio *

Não-repúdio está relacionado com o grau em que as ações ou eventos pode ser comprovado ter ocorrido, para que os eventos ou ações não pode ser repudiada mais tarde.

1

☐

2

☐

3

☐

4

☐

5

☐

Segurança: Responsabilidade *

Responsabilidade refere-se ao grau em que as ações de uma entidade pode ser rastreada exclusivamente para a entidade.

1

☐

2

☐

3

☐

4

☐

5

☐

Segurança: Autenticidade *

Autenticidade refere-se a reivindicar e identificação de um acesso assunto ou recursos pedidos de uma determinada informação.

1

☐

2

☐

3

☐

4

☐

5

☐

Manutenibilidade

Capacidade do produto de software de ser modificado. As modificações podem incluir correções, melhorias ou adaptações do software devido a mudanças no ambiente e nos seus requisitos ou especificações funcionais.

Your answer

Manutenibilidade: Modularidade

Capacidade de um sistema a ser composta de componentes de tal modo que uma mudança para um componente tem um impacto mínimo sobre os outros componentes.

1

☐

2

☐

3

☐

4

☐

5

☐

Manutenibilidade: Reusabilidade

Indica que um ativo pode ser usado em mais de um sistema, ou na construção de outros ativos.

1

☐

2

☐

3

☐

4

☐

5

☐

Manutenibilidade: Analisabilidade

Capacidade do produto de software de permitir o diagnóstico de deficiências ou causas de falhas no software, ou a identificação de partes a serem modificadas.

1

☐

2

☐

3

☐

4

☐

5

☐

Manutenibilidade: Modificabilidade

Capacidade do produto de software de permitir que uma modificação especificada seja implementada sem introduzir defeitos ou diminuir a qualidade do produto.

1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Manutenibilidade: Testabilidade

Capacidade do produto de software de permitir que o software, quando modificado, seja validado.

1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Portabilidade

Capacidade do produto de software de ser transferido de um ambiente para outro.

Your answer

Portabilidade: Adaptabilidade

Capacidade do produto de software de ser adaptado para diferentes ambientes especificados, sem necessidade de aplicação de outras ações ou meios além daqueles fornecidos para essa finalidade pelo software considerado.

1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Portabilidade: Instalabilidade

Capacidade do produto de software para ser instalado e/ou desinstalado em um ambiente especificado.

1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>



Portabilidade: Capacidade para substituir

Capacidade do produto de software de ser usado em substituição a outro produto de software especificado, com o mesmo propósito e no mesmo ambiente.

1

☐

2

☐

3

☐

4

☐

5

☐

Comentários

Faça comentários sobre a norma ISO 25010 e sua relação com SOA.

Your answer

SUBMIT

Never submit passwords through Google Forms.

This content is neither created nor endorsed by Google. [Report Abuse](#) - [Terms of Service](#) - [Additional Terms](#)

Google Forms



APPENDIX B

Technical Report
Development Process of the Electronic Health System

INSTRUÇÕES INICIAIS

- Prezado participante, muito obrigada por colaborar com essa avaliação, sua opinião é importante para minha pesquisa!
- Esse relatório técnico é um breve resumo da pesquisa de doutorado realizada durante os últimos anos.
- Por gentileza, leia atentamente esse relatório e em seguida será necessário apenas 10 minutos para preencher um questionário disponível em: <https://docs.google.com/forms/d/e/1FAIpQLSeXbhhttS18bodQM2aiUZXAifYaCrXz4GfH9Lgl4IxuPIR4BA/viewform?c=0&w=1>
- Qualquer dúvida entrar em contato através do email: joycefranca@gmail.com

Relatório técnico

Processo de Desenvolvimento do sistema de Registro Eletrônico de Saúde (RES)

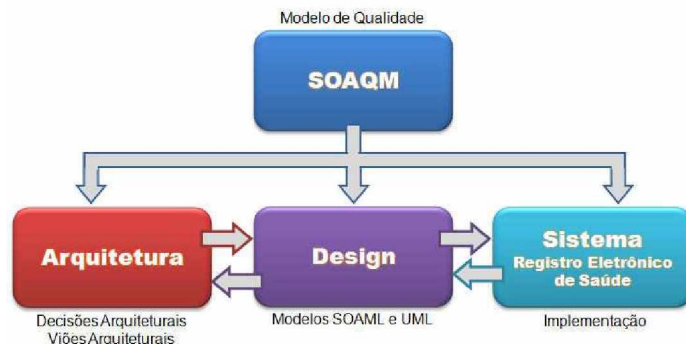
O SISTEMA

O Registro Eletrônico de Saúde (RES) é um protótipo de um sistema desenvolvido para auxiliar as atividades de rotina de um hospital e foi originalmente criado para atender as demandas do Hospital Universitário da UFS.

O RES simula o acesso de pacientes cadastrado no SUS, registra consultas, acompanha o atendimento e armazena informações do prontuário do paciente. As informações do sistema podem ser inseridas e consultadas pelos funcionários do hospital como médicos, enfermeiros e atendentes na recepção.

O sistema de prontuário eletrônico foi desenvolvido utilizando Arquitetura orientada a Serviços (SOA) para integração com sistemas legados. Além disso, o RES foi desenvolvido com base em um modelo de qualidade, específico para aplicações SOA denominado SOAQM, e foi projetado desde suas fases iniciais do processo para atender requisitos de qualidade. A Figura 1 apresenta um esquema ilustrando o processo de desenvolvimento do sistema RES.

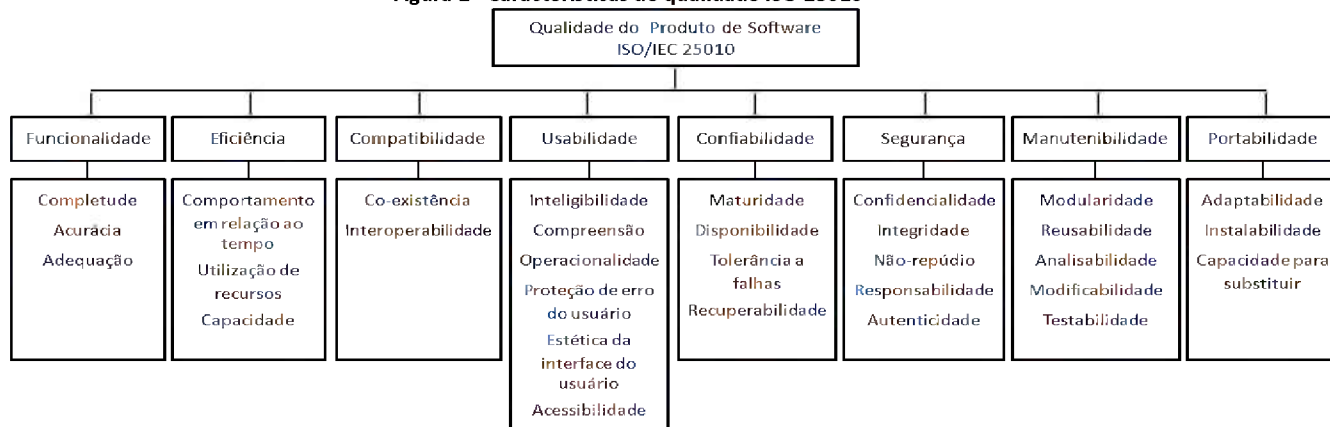
Figura 1 - Processo de desenvolvimento



O SOAQM

Os modelos de qualidade são modelos elaborados para descrever, avaliar e prever a qualidade. Esses modelos podem ser fornecidos, por exemplo, por organizações de padrões internacionais ou orientações gerais. O modelo de qualidade de produto de software definido pela ISO 25010 é composto por oito características de qualidade, são elas: Funcionalidade, Eficiência, Compatibilidade, Usabilidade, Confiabilidade, Segurança, Manutenibilidade e Portabilidade. Cada uma das oito características da ISO 25010 é composta por um conjunto de sub-características relacionadas que no total somam 31 sub-características, como mostrado na Figura 2.

Figura 2 - Características de qualidade ISO 25010



Os 31 atributos de qualidade de software propostos na ISO 25010 são para software de propósito geral e precisam ser adaptados para atender as peculiaridades de um sistema SOA. Sendo assim, O SOAQM (Service oriented Architecture Quality Model) foi

criado com base na ISO 25010 e propõe a definição de quais das características de qualidade, apresentadas anteriormente, podem ser aplicáveis e adaptáveis especificamente para sistemas SOA.

O SOAQM é um modelo de qualidade para aplicações SOA que propõe 31 características (ou atributos) de qualidade, como mostra na Tabela 1. Cada um dos atributos foram definidos na perspectiva de poderem ser aplicados em sistemas SOA, atendendo as peculiaridades inerentes a sistemas distribuídos. Além disso, o SOAQM classifica as características de qualidade com três níveis de importância, são eles: essencial, relevante e pouco importante. Os atributos essenciais (sinalizados com a cor verde) são características de qualidade muito importante e que são indispensáveis para definir qualidade em qualquer tipo de sistema SOA. As características sinalizadas com amarelo não são tão importantes, mas são relevantes para SOA e não podem ser descartadas. Os atributos em vermelho são menos importantes.

Tabela 1 - Características de qualidade SOAQM

Característica	Sub Característica	Perspectiva para SOA
Funcionalidade	Compleitude	Os serviços devem atender todas as tarefas especificadas e os objetivos do usuário conforme projetado.
	Acurácia	Os serviços devem prover os resultados corretos com grau determinado de precisão.
	Adequação	Os serviços são projetados para facilitar a realização de tarefas específicas, mais precisamente, a automação de processos de negócios.
Eficiência	Comportamento em relação ao tempo	Tempo utilizado pelo serviço para processar uma requisição e retornar uma resposta.
	Utilização de recursos	Os serviços utilizam recursos com os servidores para acessar informação de outras aplicações.
	Capacidade	A capacidade do serviço pode ser definida como a habilidade de permanecer funcionando mesmo com um grande número de acessos ao mesmo tempo.
Compatibilidade	Co-existência	Diferentes serviços compostos podem compartilhar o uso das operações de um mesmo serviço.
	Interoperabilidade	Os serviços são interoperáveis. Os serviços permitem interações entre diferentes sistemas através do uso de interfaces (WSDL) e protocolos de comunicação (SOAP).
Usabilidade	Inteligibilidade	Os usuários podem identificar se um determinado serviço é apropriado para suas necessidades através da descrição do serviço que relaciona informações tais como funcionalidade de serviço e tipos de dados transmitidos.
	Compreensão	Grau pelo qual um serviço pode facilitar o entendimento de sua operação.
	Operacionalidade	Um serviço possui o document WSDL que permite troca de mensagens entre os serviços.
	Proteção de erro do usuário	A estrutura do WSDL de um serviço não deve permitir erros de entradas de dados erradas.
	Estética da interface do usuário	Não representa o foco de SOA.
	Accessibilidade	Não representa o foco de SOA.
Confiabilidade	Maturidade	Sempre que um consumidor de serviços solicita alguma informação, espera-se que uma resposta seja retornada.
	Disponibilidade	Os serviços devem estar disponíveis quando solicitados.
	Tolerância a falhas	Os serviços podem criar estratégias que podem ser executadas quando ocorre uma falha em algum hardware ou software.
	Recuperabilidade	Capacidade do serviço em recuperação de dados quando ocorre alguma interrupção ou falha.
Segurança	Confidencialidade	As informações compartilhadas por um provedor de serviços só podem ser acessadas a um cliente de serviço autorizado.
	Integridade	Os serviços devem ser desenvolvidos para impedir o acesso não autorizado ou a modificação de dados privados.
	Não-repúdio	O provedor de serviços constrói estratégias para comprovar que uma informação foi entregue a um consumidor de serviços
	Responsabilidade	Os serviços são autônomos.
	Autenticidade	A identidade do prestador de serviços externo deve ser autenticada.
Manutenibilidade	Modularidade	O provedor de serviços hospeda um módulo de software acessível pela rede.
	Reusabilidade	Os serviços são reusáveis.
	Analisabilidade	Analisar o impacto da mudança quando os serviços precisam ser modificados
	Modificabilidade	Os serviços possuem baixo acoplamento. Esta característica reduz a dependência entre serviços, aumentando a modificabilidade.
	Testabilidade	Os serviços podem ser testados separadamente, por exemplo, através de ferramentas automatizadas para testes funcionais.
Portabilidade	Adaptabilidade	Embora os serviços da Web sejam executados remotamente em um servidor, pode ocorrer uma mudança de plataforma.
	Installabilidade	Embora os serviços da Web sejam executados remotamente em um servidor, pode ocorrer uma mudança de plataforma.
	Capacidade para substituir	Embora os serviços da Web sejam executados remotamente em um servidor, pode ocorrer uma mudança de plataforma.

A ARQUITETURA

Os atributos de qualidade propostos pelo modelo de qualidade SOAQM podem ser utilizados para fundamentar as Decisões Arquiteturais de aplicações SOA. O objetivo dessa fase foi elaborar decisões arquiteturais focadas em alcançar aspectos de qualidade

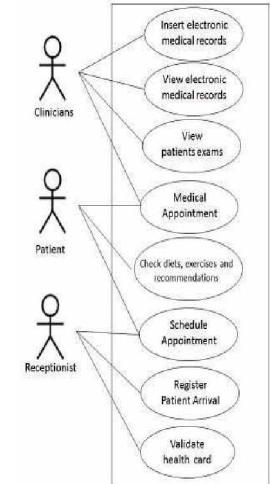
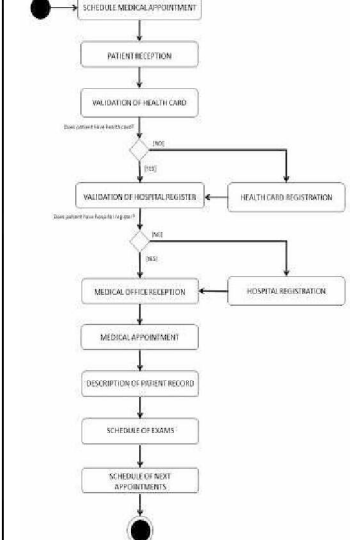
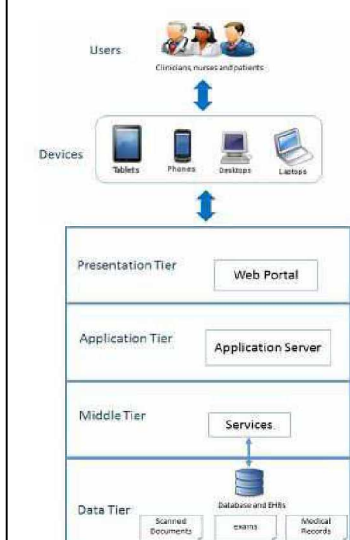
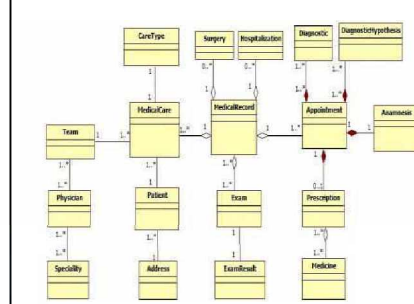
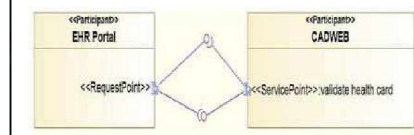
desde o início do processo de desenvolvimento e assim poder garantir que os atributos de qualidade sejam alcançados. No desenvolvimento do sistema RES foram definidas decisões arquiteturais descritas conforme o modelo mostrado a seguir. As outras decisões não serão apresentadas nesse documento, mas seguem o mesmo padrão de definição.

Tabela 2 - Decisão Arquitetural D01 - Consumir informações de sistemas externos

Decisão D01: Desenvolver uma aplicação de registro de saúde que coleta dados de sistemas externos	
Assunto	Atualmente, dados importantes dos pacientes estão espalhados em diversos sistemas legados e não há um sistema onde os dados são centralizados.
Decisão	Desenvolver uma nova aplicação RES baseada em SOA para consumir a funcionalidades dos sistemas envolvidos através de serviços.
Restrições	Somente ferramentas <i>open source</i> devem ser usadas para o desenvolvimento.
Solução	Definir ferramentas open source para o desenvolvimento. Transformar sistemas legados para expor serviços. Desenvolver uma aplicação RES para consumir esses serviços.
Justificativa	A arquitetura SOA propõe muitas vantagens comparada a outra abordagens para sistemas distribuídos. Além disso, diversas ferramentas gratuitas para desenvolvimento SOA podem ser encontradas.
Atributos de qualidade	Interoperabilidade, Modularidade, Reusabilidade, Modificabilidade, Testabilidade, Maturidade, Disponibilidade, Comportamento com relação ao tempo, Utilização de Recursos.

A descrição da arquitetura do software RES foi definida através de múltiplas visões arquiteturais, conforme definido pela ISO 42010. As múltiplas visões são usadas para descrever diferentes aspectos da arquitetura para diferentes grupos de *stakeholders*, incluindo médicos, enfermeiras, administradores do hospital e desenvolvedores de software. As visões mais representativas são apresentadas a seguir na Tabela 3.

Tabela 3 - Visões Arquiteturais do sistema RES

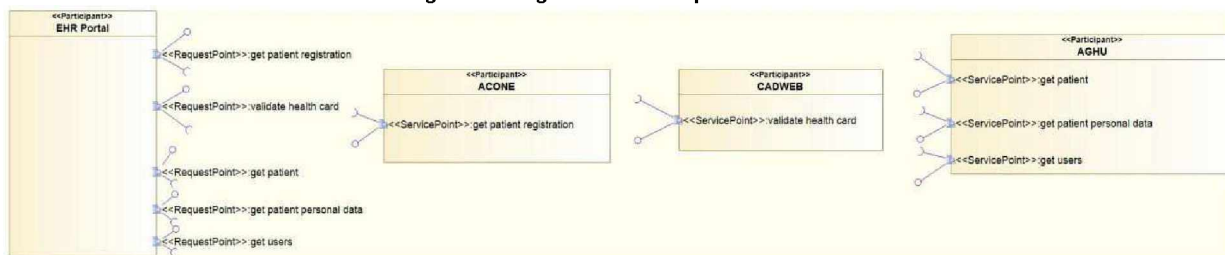
Visão de cenário	Visão de Processo de Negócio	Visão de Implementação	Visão Lógica
Nessa visão as funcionalidades do RES são descritas através de um conjunto de casos de uso documentados, bem como os digramas de caso de uso para explicitar os participantes e suas atribuições.	Nessa visão o principal foco é detalhar os processos de negócios para explicitar o procedimento envolvido de cada atividade, suas peculiaridades e suas restrições. A Figura 4 apresenta o processo para gerenciamento de consultas no hospital.	A arquitetura do RES foi projetada no estilo de multicamadas: Apresentação, Aplicação, Intermediária e Dados. A primeira camada de apresentação pode ser acessada através de computadores e dispositivos móveis.	A visão lógica é descrita através de diagramas que apresentem como as informações serão armazenadas bem como os dados relevantes e suas interações. Dois diferentes e complementares diagramas são apresentados, diagrama de classes da UML e diagrama de participantes da SoaML.
 <p>Figura 3 - Diagrama de casos de uso UML</p>	 <p>Figura 4 - Diagrama de atividades UML</p>	 <p>Figura 5 - Camadas da arquitetura</p>	 <p>Figura 6 - Diagrama de Classes UML</p>  <p>Figura 7 - Diagrama de participantes SoaML</p>

O DESIGN

A modelagem do sistema RES foi desenvolvida usando UML e SoaML. A SoaML é uma linguagem de modelagem específica para aplicações SOA e sua compreensão se torna mais simplificada porque é baseada na UML e seus estereótipos. Nesse relatório será representado apenas alguns diagramas da SoaML por questões de espaço.

Uma parte do escopo do sistema RES que foi desenvolvido é responsável por centralizar e controlar consultas médicas. Atualmente, o gerenciamento de consultas exige o acesso a diversos sistemas legados como ACONTE, CADWEB e AGHU. O sistema RES foi desenvolvido para centralizar essas informações através de serviços. O diagrama da SoaML apresentado na Figura 8 mostra os sistemas envolvidos e os serviços que cada sistema legado vai fornecer através de um barramento de serviços para ser consumido pelo portal RES.

Figura 8 - Diagrama de Participantes do RES



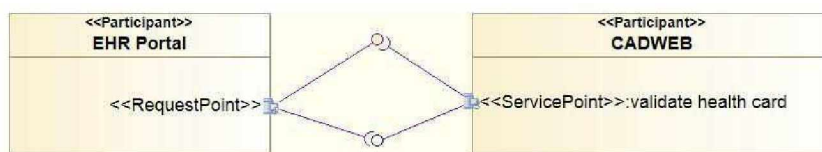
O sistema legado CADWEB é um sistema nacional de saúde onde está registrado o número do cartão do SUS de todos os brasileiros. O serviço que será fornecido pelo CADWEB será responsável por validar o cartão do SUS do paciente.

O sistema ACONTE é um sistema utilizado nas unidades básicas de saúde onde estão cadastrados todos os registros pessoais e onde é realizado o agendamento de consultas médicas. O serviço fornecido desse sistema é responsável por retornar os dados do paciente e suas consultas.

O AGHU é sistema legado do hospital que dentre as diversas funcionalidades armazena as informações de todos os funcionários e pacientes atendidos no hospital. Os serviços fornecidos são responsáveis por retornar o paciente através do CPF ou cartão do SUS, retornar dados pessoais do paciente e usuários do sistema.

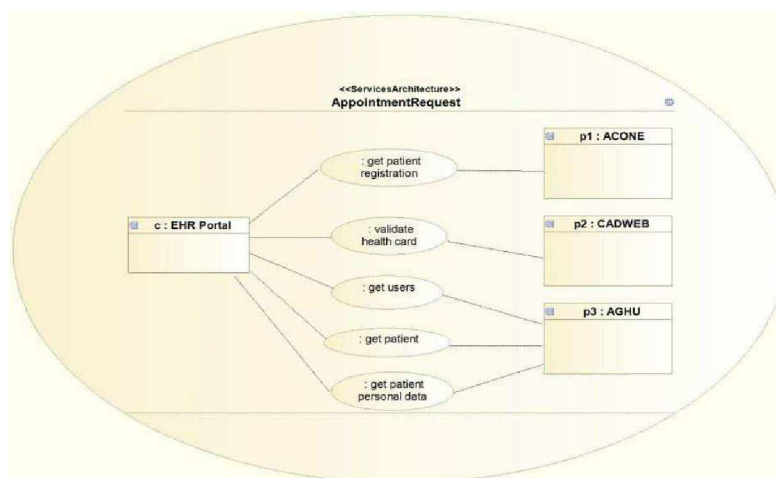
A especificação completa de um participante no diagrama inclui as portas que sinalizam o sistema provedor do serviço e o consumidor. Na Figura 9, o sistema CADWEB possui uma porta classificada como <<ServicePoint>> que representa o serviço exposto para ser utilizado. O RES por sua vez, possui uma porta <<RequestPoint>> que sinaliza o serviço a ser consumido em uma requisição.

Figura 9 - Diagrama de participantes com portas



O diagrama de Arquitetura de Serviço (Figura 10) fornece uma visão global que claramente apresenta todos os serviços que deverão ser implementados, quais sistemas estão envolvidos e quais são os sistemas consumidores e provedores.

Figura 10 - Diagrama de Arquitetura de Serviço



APPENDIX C

Questionnaire for Experts
Evaluation of Development Process of the Electronic Health System

Avaliação sistema RES - especialistas

* Required

Perfil do Participante



Nome *

Your answer

Área de atuação

	Até 2 anos	Até 5 anos	Até 8 anos	Mais de 8 anos
Professor	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Desenvolvedor	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Gerente de Projetos	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Arquiteto de Software	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Possuo conhecimento de Arquitetura Orientada a Serviços (SOA). *

☐ 1. Discordo totalmente

☐ 2. Discordo

☐ 3. Indiferente



- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

Meu trabalho envolve desenvolvimento de aplicações baseadas em SOA. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

Possuo conhecimento sobre a norma ISO 25010. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

Possuo interesse em modelos de qualidade para aplicações SOA. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente



- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

Questionário

Baseado na breve apresentação, avalie o processo de desenvolvimento do sistema de Registro Eletrônico de Saúde usando o modelo de qualidade SOAQM

SOAQM

1. Eu considero importante seguir os padrões de desenvolvimento de software propostos pela ISO 25010. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

2. Eu considero que utilizar um modelo de qualidade nas fases iniciais no processo de desenvolvimento de software é importante para garantir maior qualidade do produto final. *

- ☐ 1. Discordo totalmente

- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

3. Eu considero que seria necessário pouco tempo para aprender mais sobre o SOAQM. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

4. Usar o SOAQM melhoraria a qualidade das aplicações baseadas em SOA. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

5. No geral, eu considero que o processo de desenvolvimento utilizado pode ser aplicado no desenvolvimento de outros sistemas SOA. *



- ☐ 1. Discordo totalmente

- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

6. Supondo que eu tenha um conhecimento mais aprofundado no modelo de qualidade SOAQM, eu poderia utilizá-lo no desenvolvimento de outros sistemas SOA. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

Decisões arquiteturais



7. Eu considero importante definir o formato da decisões arquiteturais no processo de desenvolvimento de software. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo



- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

8. Eu considero que utilizar decisões arquiteturais baseadas no modelo de qualidade é importante para garantir maior qualidade do produto final. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

9. Eu considero que seria necessário pouco tempo para aprender mais sobre como definir decisões arquiteturais a partir de um modelo de qualidade. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

10. No geral, eu considero que as decisões arquiteturais utilizadas podem ser aplicadas no desenvolvimento de outros sistemas SOA. *

- ☐ 1. Discordo totalmente



- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

11. Supondo que eu tenha um conhecimento mais aprofundado na arquitetura utilizada, eu poderia utilizá-lo no desenvolvimento de outros sistemas SOA. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

Visões arquiteturais



12. Eu considero importante definir visões arquiteturais no processo de desenvolvimento de software. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo



- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

13. Eu considero que definir visões arquiteturais é importante para melhorar a compreensão das funcionalidades do sistema. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

14. No geral, eu considero que as visões arquiteturais devem ser aplicadas no desenvolvimento de outros sistemas SOA. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente



15. Eu considero importante utilizar uma linguagem de modelagem específica para SOA. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo



- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

16. Eu considero que utilizar SOAML melhoraria a compreensão do design de aplicações SOA. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

17. Eu considero que seria necessário pouco tempo para aprender mais sobre o SOAML. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

18. No geral, eu considero que os diagramas SOAML podem ser aplicados no desenvolvimento de outros sistemas SOA. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo



- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

19. Supondo que eu tenha um conhecimento mais aprofundado em SOAML, eu poderia utilizá-lo no desenvolvimento de outros sistemas SOA. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

20. Comentários gerais sobre o processo de desenvolvimento utilizado para construção do sistema RES. *

Your answer

21. Descreva os motivos que te levariam a usar ou não o SOAQM para desenvolvimento de sistemas SOA. *

Your answer

SUBMIT

Never submit passwords through Google Forms.

This content is neither created nor endorsed by Google. Report Abuse - Terms of Service - Additional Terms



Google Forms

APPENDIX D

Questionnaire for Users
EHR Evaluation using Technology Acceptance Model (TAM)

Avaliação Usuários RES

O Registro Eletrônico de Saúde (RES) é um protótipo de um sistema desenvolvido para auxiliar as atividades de rotina de um hospital e foi originalmente criado para atender as demandas do Hospital Universitário da UFS.

O RES simula o acesso de pacientes cadastrado no SUS, registra consultas, acompanha o atendimento e armazena informações do prontuário do paciente. O sistema RES está disponível em um endereço eletrônico e pode ser acessado por qualquer equipamento com acesso à Internet. As informações do sistema podem ser inseridas e consultadas pelos funcionários do hospital como médicos, enfermeiros e atendentes na recepção.

O RES tem por objetivo facilitar o preenchimento do prontuário do paciente que frequentemente é feito em papel. Além disso, o preenchimento eletrônico do prontuário agiliza o acesso ao histórico de doenças que geralmente é dificultado devido à grande quantidade de papéis acumulado durante anos.

* Required

Perfil do Participante



Nome *

Your answer

Profissão *

☐ Médico (a)

☐ Enfermeiro (a)

☐ Atendente



☐ Other:

Preenchimento do Prontuário *

☐ Papel

☐ Software específico

☐ Other:

Meu trabalho no hospital exige o preenchimento de prontuário do paciente *

☐ 1. Discordo totalmente

☐ 2. Discordo

☐ 3. Indiferente

☐ 4. Concordo

☐ 5. Concordo totalmente

O prontuário escrito à mão frequentemente apresenta problemas de ilegibilidade. *

☐ 1. Discordo totalmente

☐ 2. Discordo



- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

Procurar histórico do paciente é difícil devido ao acúmulo de papéis no arquivo de prontuário. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

Utilidade do RES



1. Eu considero o RES útil para desempenhar minhas atividades no trabalho. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

2. Usar o RES melhoraria meu desempenho porque seria mais rápido acessar o histórico de doenças do paciente utilizando o RES. *



- ☐ 1. Discordo totalmente

- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

3. Usar o RES para preenchimento de prontuário melhoraria minha produtividade porque eu poderia economizar tempo no meu trabalho. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

4. Usar o RES melhoraria a organização do meu trabalho porque o espaço utilizado para armazenar os papéis do arquivo de prontuário poderia ser aproveitado para outros fins. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

5. Usar o RES melhoraria a eficácia do meu trabalho porque o sistema do hospital seria integrado com outros sistemas externos importantes. *

- ☐ 1. Discordo totalmente



- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

6. Usar o RES melhoraria a qualidade do prontuário produzido. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

7. Usar o RES melhoraria o atendimento do paciente, no que diz respeito a rapidez de preenchimento do prontuário. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

8. Usar o RES melhoraria o controle de pacientes dentro do hospital no que diz respeito a encaminhamentos para consultórios, laboratórios e salas cirúrgicas. *

- ☐ 1. Discordo totalmente



- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

Facilidade de Uso do RES



9. Eu considero que as telas do RES está organizado de forma simples. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

10. No geral, eu considero o RES fácil de usar. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

11. Eu considero fácil para mim aprender a usar o RES. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente



- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

12. Eu consigo facilmente preencher o prontuário. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

13. Eu consigo facilmente realizar pesquisas de dados de paciente.

*

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

14. Eu consigo facilmente identificar a quantidade de atendimentos agendados no dia. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo



- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

Usabilidade do RES



15. Supondo que eu tenha acesso ao sistema, tenho a intenção de usá-lo. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

16. Supondo que eu tenha acesso ao sistema, eu recomendaria o seu uso para outros colegas de trabalho. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

17. Eu considero que seria necessário pouco tempo para aprender a usar o RES. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo



- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

18. Eu considerado que o RES teria facilidade de aceitação no hospital. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

19. Eu considerado que é importante a portabilidade do RES, no que diz respeito a possibilidade de acesso através de computadores, tablet ou celulares. *

- ☐ 1. Discordo totalmente
- ☐ 2. Discordo
- ☐ 3. Indiferente
- ☐ 4. Concordo
- ☐ 5. Concordo totalmente

Comentário gerais 



20. Descreva sua opinião sobre o sistema RES em comparação com os procedimentos atuais de atendimento ao paciente. *

Your answer

21. Descreva as vantagens sobre a utilização do sistema RES. *

Your answer

22. Descreva as desvantagens sobre a utilização do sistema RES. *

Your answer

23. Descreva as dificuldades encontradas na utilização do RES. *

Your answer

24. Descreva os motivos que te levariam a usar ou não o sistema RES. *

Your answer

SUBMIT

Never submit passwords through Google Forms.

This content is neither created nor endorsed by Google. Report Abuse - Terms of Service - Additional Terms

Google Forms

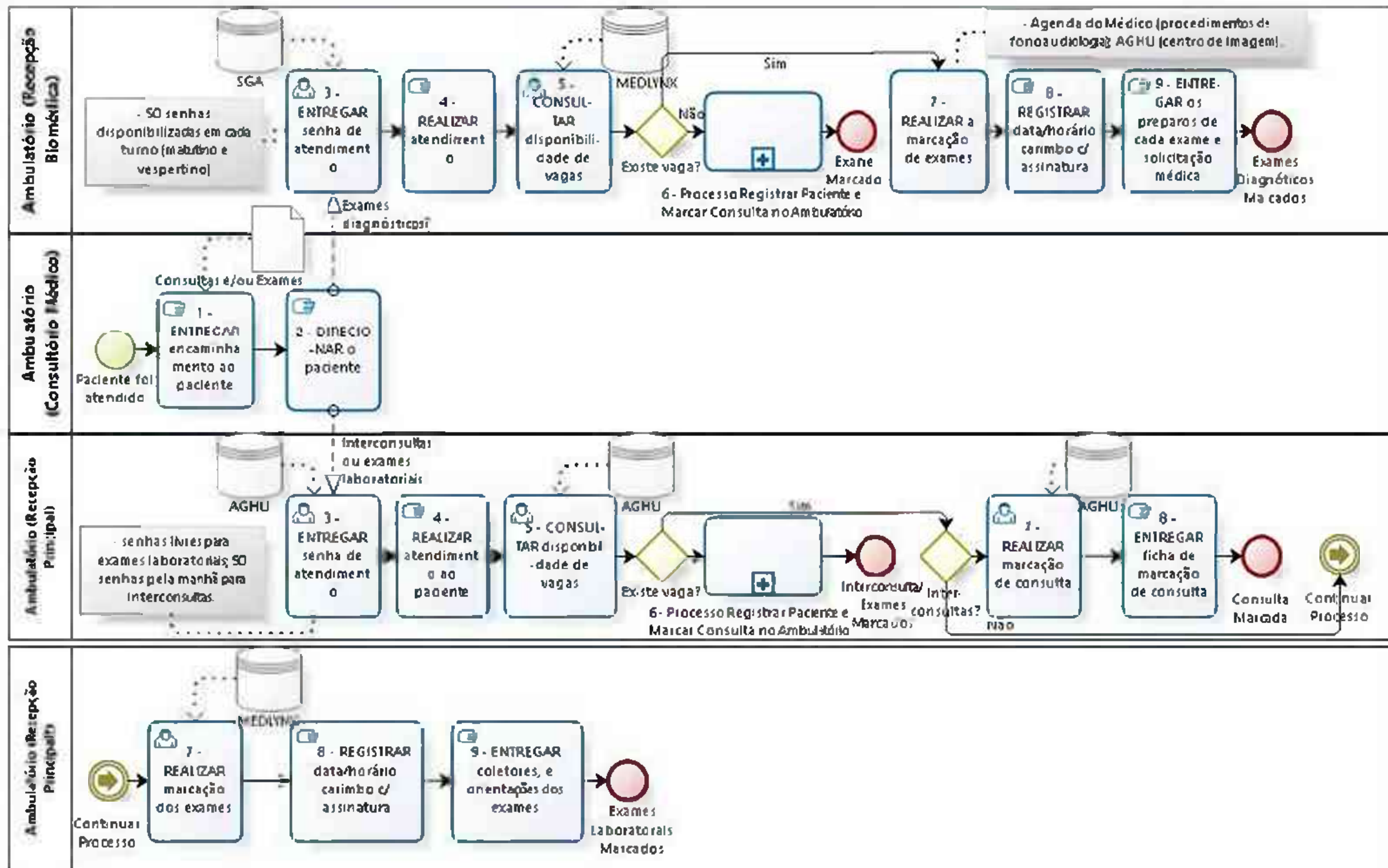


Annex

ANNEX A

Business Process

Perform Appointment and Manage Evolving Documentation



ANNEX B

UML Class Diagram with attributes and multiplicity

