Software Quality Assurance by means of Methodologies, Measurements and Knowledge Management Issues

Káthia Marçal de Oliveira

University of Valenciennes and Hainaut-Cambrésis
LAMIH CNRS UMR 8201

Habilitation à Diriger des Recherches - HDR

Káthia Marçal de Oliveira

Software Quality Assurance by means of Methodologies, Measurements and Knowledge Management Issues

HDR publicly defended in 30th June 2014

Committee

Reviewer (*Rapporteurs* in French):

- Pr. Gaelle Calvary, University of Grenoble, France
- Pr. Oscar Pastor Lopez, Universitat Politècnica de València, Spain
- Pr. Isabelle Wattiau, Conservatoire National de Arts et Métiers, Paris

Examiners (*Examinateurs* in French):

- Pr. Ana Regina Cavalcanti da Rocha, Federal University of Rio de Janeiro, Brazil
- Pr. Camille Rosenthal-Sabroux, University of Paris-Dauphine, France

Sponsors (*Co-promoteurs* in French):

- Pr. Mourad Abed, University of Valenciennes and Hainaut-Cambrésis, France
- Pr. Christophe Kolski, University of Valenciennes and Hainaut-Cambrésis, France

"Accept the fall
But do not feel down
Stand up, push the dust away
And start all over again"
From the original "Reconhece a queda e não desanima
Levanta, sacode a poeira
E dá a volta por cima"
(Paulo Vanzolini)

"They did not know it was impossible, so they did it!"
(Mark Twain)

"No one I ever met was against quality, and
most of the management people I meet think, they are, in fact, managing it.
If that were true we would be living in a world bathed in prevention,
rather than on continually seeking cures and corrective actions."
("Quality is free – if you understand it", Philip Crosby)

Acknowledgements

So many people contributed to my career in so many ways that I could write pages and pages of acknowledgments. Students, teachers, colleagues, family and friends without whom it would be impossible to get here. My sincere *thanks* to all of them. However, I beg them all to thank some people in particular. I am sure they will understand it.

First of all, I would like to thank Pr. Ana Regina Cavalcanti da Rocha. With her I learned everything that I know: to perform research, to work in-group, to supervise students, to work as a consultant in industry, and, mainly, I learned to love software quality assurance. If I am here, it is because of her.

I would like also to thank Pr. Guilherme Horta Travassos that co-supervised my PhD thesis and showed me that we need to take risks to go further.

My thanks to Kival Chaves Weber who always believed in me and invited me for the definition and institutionalization of the Brazilian Stantard of Software Process Improvement (MPS.BR). With this work I could really see software quality assurance in practice.

Thanks to Dr. Álvaro Rabelo Jr. that has showed me since my first research project that we can go wherever we want if we work hard.

I would like to thank Pr. Claudio Chauke Nehme, the head of Computer Science Department of Catholic University of Brasília. He supported me in all my initiatives during the eight years I worked there. Thanks to this, we built the software quality area in the university with a group of associate professors that worked not only in the academy but also in industry.

Arriving in France I had the chance to met people who also motivated me. People like Pr. Thierry Marie Guerra that, without knowing my research domain, has always encouraged me to move forward. His confidence made me believe it would be possible to continue my career in France.

Thanks to Pr. Jean-Mark Jezequel who offered me the chance to know a new research domain that was essential to get an associate professor position in France.

Special thanks to Pr. Christophe Kolski, that even before starting to work at the University of Valenciennes, gave me the possibility to explain my ideas and motivated me to come to LAMIH. Thanks for his carefully reading and correction of this document, for his ethics and professional competence that serves as a guide to all his colleagues.

Thanks also to Pr. Mourad Abed for giving me the opportunity for a first doctorate co-supervision, which resulted in a great evolution in my career.

Many thanks to Pr. Camille Sabroux-Rosenthal for believing and supporting all my ideas to strengthen the area of quality evaluation of information systems in France. Moreover, she was the first one to encourage me to do my HDR and to assure that it would be possible.

My sincere thanks to Pr. Rossana Andrade for giving me the opportunity to explore software quality assurance for a new application domain. With this opportunity, she gave me oxygen and motivation in the exact moment that I needed.

Undoubtedly, thank you very much to the professors Gaelle Calvary, Oscar Pastor and Isabelle Wattiau for their precious time spent in reading this document and writing the evaluation report.

Finally, I would like to thank some close friends who helped me put my pieces together when I was feeling down. Who gave me their hands to stand up and helped me move on. Brazilian and French, no difference, they taught me the true value of friendship. Big thanks to Carla Valle-Klann, Claudia Gama, Elisângela Aguiar, Emmanuelle Grislin, Mirella Quadros, Sophie Lepreux, Sylvia Estivie and, my “elder sister”, Veronique Delcroix.

Software Quality Assurance by means of Methodologies, Measurements and Knowledge Management Issues

Káthia Marçal de Oliveira

Abstract:
Software quality assurance has emerged in the past years for the market and academia as a better solution to achieve customer satisfaction. Organizations recognize that it is only by offering better quality products that they can have an edge over their competitors and thus they can ensure customer loyalty. In this context, researchers on software quality have proposed methodologies, techniques and different approaches to support this activity and its effective use in industry. This report presents a piece of research conducted in this direction. Methodologies and measurements were defined and applied to support the assessment of software processes and products for different types of systems (e.g. web applications, information systems, ubiquitous systems, legacy systems). Knowledge management issues were also explored for the organization of knowledge with ontologies; capture and dissemination of lessons learned in projects to support software quality improvement. The experience and results obtained with this research conducted to the definition of new research perspectives. The first research perspective aims at defining new approaches for quality evaluation based on measurements for interactive systems (in particular for the predictive evaluation of usability and the quality evaluation of ubiquitous systems). The second one aims at exploring knowledge management concepts for the benefit of the quality evaluation activity itself, considering that software quality assurance is a knowledge-rich activity including explicit documented knowledge in standards, quality measurements and techniques, as well as tacit knowledge of each individual that participates in the quality evaluation process (evaluators, users and other stakeholders). In this way, all this research contributes to the introduction of practical software quality assurance procedures throughout the software processes to enable their real adoption by organizations.

Content

Part I. Extended Curriculum Vitae

Overview **2**
Teaching **5**
Administrative Activities **6**
Graduate Supervision **7**
Research Projects **10**
Scientific Expertise **11**
Publications **13**

Part II. Software Quality Assurance by means of Methodologies, Measurements and Knowledge Management Issues

Chapter 1 Introduction **23**
1.1 Context and Historical Research **23**
1.2 Objective and Research Challenges **27**
1.3 Report Organization **29**
Chapter 2 Software Quality Assurance **31**
2.1 Introduction **31**
2.2 Basic Definitions **31**
2.2.1 Software Quality 31
2.2.2 Software Quality Assurance 32
2.2.3 Measurement 33
2.3 Approaches for Measurement Definition and Collection **36**
2.4 Standards and Models for Software Quality **38**
2.5 The Role of Knowledge Management **40**
2.5.1 Knowledge and Knowledge Management: Concepts and Application 41
2.5.2 Knowledge Capture and Dissemination 45
2.5.3 Knowledge Organization with Ontologies 48
2.6 Conclusion: Research Motivation **50**
Chapter 3 Methodologies and Measurements for Software Quality Assurance **51**
3.1 Introduction **51**
3.2 Software Process Quality Assurance **51**
3.2.1 Software Process Evaluation 52
3.2.2 Software Process Improvement 54
3.2.3 Return on Investment in Software Process Improvement 60

3.3 Software Product Quality Assurance 65
3.3.1 Quality Evaluation of Web Applications 65
3.3.2 Quality Evaluation of Legacy Systems 71
3.3.3 Quality Evaluation of Human-Computer Interaction in Ubiquitous Systems 79
3.3.4 Evaluation of Information Density of Graphical User Interfaces 85
3.3.5 Quality Evaluation of Human-Computer Interaction 92
3.4 Conclusion 95

Chapter 4 Applying Knowledge Management Issues for Software Quality Assurance 97
4.1 Introduction 97
4.2 Knowledge Management Issues to Support Software Maintenance Process 98
4.2.1 Identifying and Formalizing the Knowledge Needed for Software Maintenance 98
4.2.2 Capturing Knowledge for Maintenance Using Ontologies 103
4.3 Capturing and Disseminating Knowledge with Experience Factory 109
4.4 Pro-active Dissemination of Knowledge 110
4.5 Using Ontologies to Improve Usability of Personalized User Interfaces 114
4.6 Conclusion 124

Chapter 5 Synthesis and Research Perspectives 125
5.1 Introduction 125
5.2 Synthesis of Contributions 125
5.3 Research Perspectives 128
5.3.1 Predictive Usability Evaluation 129
5.3.2 Quality Evaluation of HCI in Ubiquitous Systems 136
5.3.3 Towards Knowledge Management *of* and *for* Software Quality Assurance 140
5.4 Conclusion 143

References 145

Appendix A. Defect Density Indicator 161

List of Figures

Figure 0.1. Chronological view of my career 2
Figure 1.1. Research workflow 27
Figure 1.2 Research Goal and Challenges 28
Figure 2.1. Quality definitions 32
Figure 2.2. Definitions related to software quality assurance 33
Figure 2.3. Measurement definitions 33
Figure 2.4. Key relationships in the measurement information model (ISO/IEC 15939, 2007) 35
Figure 2.5. Example of information model concepts for productivity **(**ISO/IEC 15939, 2007) 36
Figure 2.6. Approaches for measurement definition 36
Figure 2.7. Hierarchical structure of the GQM approach 37
Figure 2.8. Standards and models for software quality 38
Figure 2.9. Examples of measures for effectiveness 38
Figure 2.10. Example of guidelines for World Wide Web user interfaces 39
Figure 2.11. Stakeholder requirements Process (Activities and Tasks) 40
Figure 2.12. CMMI Maturity levels x Process Areas 41
Figure 2.13. Data, Information and Knowledge (adapted from Probst *et al.* (2000)) 42
Figure 2.14. Knowledge management definitions 43
Figure 2.15. Knowledge sharing according to Nonaka and Takeuchi (1995) 43
Figure 2.16. Creation and transfer of common knowledge (Dixon, 2002) 44
Figure 2.17. Quality Improvement Paradigm 45
Figure 2.18. Post-mortem analysis phases 46
Figure 2.19. Learning history phases 47
Figure 2.20. Ontology definitions 48
Figure 2.21. Illustrative example of an ontology 49
Figure 3.1. Example of measures for software process evaluation defined with GQM 53
Figure 3.2. Examples of measures report 55
Figure 3.3. Steps for definition of a catalog of indicators (**Monteiro and Oliveira, 2011**) 57
Figure 3.4. Example of statements of the questionnaire to evaluate service quality 60
Figure 3.5. Research protocol (**Ramos *et al.,* 2013**) 62
Figure 3.6. Elements for analysis of ROI in SPI 63
Figure 3.7. Strategy for the analysis of benefits in SPI (**Ramos *et al.*, 2013**) 64
Figure 3.8. Methodology for definition of the techniques (**Valentin *et al.*, 2012**) 70
Figure 3.9. Efficiency and effectiveness boxplots by techniques 72
Figure 3.10. Distribution graphs for S1 77
Figure 3.11. Ubiquitous computing 80
Figure 3.12. Number of quality characteristics by study 81
Figure 3.13. UI 1 to get direction, UI 2 to choose preferences and UI3 to select related services 87
Figure 3.14. Composition 1 {orderIndependance, meet}: in the same frame. 88
Figure 3.15. Composition 2 {interleaving, equals}: a) UI'1, b) UI'2 and c) UI'3 in tabs 89
Figure 3.16. Composition 3{enabling, covers}: a) UI'1, b) UI'2 and c) UI'3 in sequence 90

Figure 3.17. The proposed mapping model 93
Figure 3.18. Extract of indicator specification 94
Figure 4.1. Steps for identification of knowledge for software maintenance 98
Figure 4.2. Knowledge identification 99
Figure 4.3. Sub-ontologies of the maintenance ontology (**Anquetil *et al.,* 2006**) 100
Figure 4.4. Software system sub-ontology (**Anquetil *et al.,* 2006**) 101
Figure 4.5. Example of formalization of an axiom 102
Figure 4.6. ISO14764 maintenance process with the intermediary (1 and 2) and final (3) PMAs (**Anquetil *et al.*, 2006**) 105
Figure 4.7. Extract of the post-analysis questionnaire 106
Figure 4.8. Captured knowledge about the software process execution 108
Figure 4.9. Mapping of the learning history process to Dixon's knowledge cycle 112
Figure 4.10. Extract of a learning history (translated from Portuguese) 113
Figure 4.11. Examples of user interface personalization in a transportation system (**Oliveira *et al.*, 2013**) 115
Figure 4.12 The context model: user profile (**Bacha *et al.,* 2011a**) 117
Figure 4.13. The transportation ontology global view (**Oliveira *et al.,* 2013**) 118
Figure 4.14. Meta-model for mapping context and ontology elements (**Bacha *et al.* 2011a**) 119
Figure 4.15. Mapping examples (**Oliveira *et al.,* 2013**) 120
Figure 4.16. Example of a UI specification 121
Figure 4.17. Annotation of task model elements (**Bacha *et al.,* 2011d**) 122
Figure 5.1. Predictive usability evaluation 130
Figure 5.2. Task model for the radio player (Molina *et al.*, 2005) 134
Figure 5.3. Evaluation of UI design model x Final UI for input validity data 136
Figure 5.4. Overview of quality evaluation of HCI in ubiquitous systems 139
Figure 5.5. Tacit Knowledge transfer (Arduin, 2013) 142

List of Tables

Table 3.1. Measures proposed for software process evaluation 53
Table 3.2. Indicators X Measures 58
Table 3.3. Accessibility evaluation measures 67
Table 3.4. Evaluation data of 10 users with visual disability 69
Table 3.5. Example of Verification Items 70
Table 3.6. Measures for assessing the documentation of the system 73
Table 3.7. Measures for assessing system source code 74
Table 3.8. Measure results 76
Table 3.9. Decision of impact in maintainers' contracts 78
Table 3.10. Classification of the impact in the contract of the evaluated systems 78
Table 3.11. Characteristics for quality evaluation of HCI in ubiquitous systems 82
Table 3.12. Measures for context awareness 84
Table 3.13. the performed scenarios in our evaluation 85
Table 3.14. Results for the evaluation of context-awareness 85
Table 3.15. Measures for information density 86
Table 3.16. Proposed measures applied to the three composed UI of the case study 91
Table 3.17. Examples of the mapping between qualitative and quantitative data 94
Table 4.1. Number of concepts used in two studies (**Anquetil *et al.*, 2003**) 103
Table 4.2. The three maintenance PMAs and the types of knowledge they focus on 106
Table 4.3. Captured knowledge about the system: instantiated concepts from the ontology 107
Table 4.4. Examples of lessons learned for software process estimation 110
Table 4.5. Cost of a knowledge cycle using learning history 113
Table 4.6. Using ontology properties in path finding 123
Table 5.1. Example of analysis of measures 132
Table 5.2. Some proposed measures 133
Table 5.3. Resuls of measures for the "Radio Player" 135
Table 5.4. Examples of measures and methods used for quality evaluation 138

Acronyms

AUI	Abstract User Interface
CMMI	Capability Maturity Model Integration
CMMI-DEV	Capability Maturity Model Integration for Developers
CSUQ	Computer System Usability Questionnaire
CTT	Concur Task Tree
CUI	Concrete User Interface
GQM	Goal-Question Metric
HCI	Human-Computer Interaction
MIT	Model Inspection Technique for usability evaluation
MPS.BR	Brazilian Software Process Improvement Standard (from Portuguese, *Melhoria de Processo do Software Brasileiro*)
MR-MPS-SW	Brazilian Software Process Improvement Reference Model (from Portuguese – *Modelo de Referência para Melhoria do Processo de Software*)
PMA	Post-Morten Analysis
QIP	Quality Improvement Paradigm
QUIM	Quality in Use Integrated Measurement
RFID	Radio-Frequency Identification
ROI	Return on Investment
SDE	Software Development Environment
SERVQUAL	Service Quality model
SLA	Service Level Agreement
SPI	Software Process Improvement
SPICE	Software Process Improvement and Capability dEtermination
SQA	Software Quality Assurance
SQuaRE	Software product Quality Requirements and Evaluation
UI	User Interface
UML	Unified Modeling Language
WCAG	Web Content Accessibility Guidelines
WIMP	Windows, Icons, Menus and Pointing device

Part I. Extended Curriculum Vitae

Overview

I have a somewhat unusual career. I did my Masters in Computer Science from 1993 to 1995 in Brazil. As part of my studies for the Master degree, I have taken part in a research project at the Cardiology Bahian Foundation in collaboration with the Federal University of Rio de Janeiro (*Universidade Federal do Rio de Janeiro - UFRJ*). Parallel to the Master studies, I have worked as a substitute teacher in some universities.

Following that, I did my Doctorate studies at the Federal University of Rio de Janeiro (1995-1999). During this period, I spent a year at the University of Ottawa (Canada) under the direction of Professor Stan Matwin to do research on the use of knowledge representation methods in software engineering. After I finished my Doctorate degree in 1999, I continued to work at the same university doing post-doctoral research for one year. In this time, I also co-supervised two Masters students who worked on themes related to my Doctorate research.

After my post-doctoral at UFRJ I worked for eight years (2001-July 2009) for the Master Programme of Knowledge Management and Information Technology of the Catholic University of Brasilia, Brazil. This university does not have a Doctorate program, therefore all my research, during this period, was carried out with master students. As part of my research I supervised/co-supervised 21 master students, sometimes with researchers from different domains, such as Psychologists. During this period, I have coordinated two research projects funded by the National Brazilian Research Agency (CNPq - National Counsel of Technological and Scientific Development)".

For four years (2005-2008) I have participated in the scientific committee of the Brazilian software quality standard (MPS.BR - *Melhoria de Processo do Software Brasileiro*), an important experience where I could apply the research in practice in the industry.

My career in France started in December 2008 with a post-doctoral opportunity at the University of Rennes - IRISA (12/2008 to 05/2009) in a project in model-driven engineering domain, coordinated by Professor Jean-Marc Jezequel. I have been working at the University of Valenciennes, Laboratory of Industrial and Human Automation Control, Mechanical Engineering and Computer Science (LAMIH) since September 2009 in the DIM (Decision, Interaction and Mobility) team where I have started to work in the Human-Computer Interaction research domain.

Figure 0.1 summarizes my career with a chronological view (details will be presented in next sections).

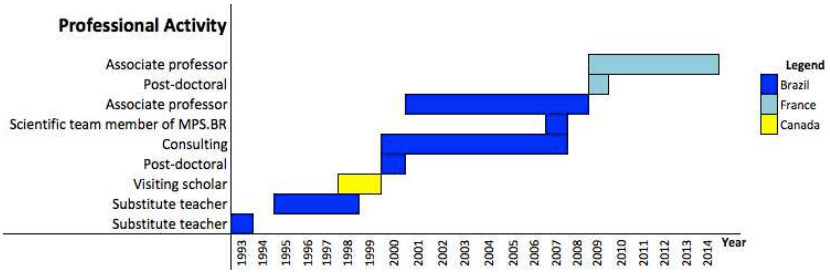


Figure 0.1. Chronological view of my career

Education

1995-1999 **D.Sc. in System Engineering and Computer Science**
University: Federal University of Rio de Janeiro (Brazil)
Title: *Model for the construction of Domain-Oriented Software Development Environments*
Advisors: Ana Regina Cavalcanti da Rocha
Gulherme Horta Travassos
Summary:
During software development, one of the most critical activities for software engineers is the correct description and identification of product requirements. This activity involves understanding the problem, which is essential to allow one to define a solution for it. To do this, it is important not only to understand the tasks routinely performed which are part of the problem, but more importantly to understand the domain in which the system will take place. Believing that the use of domain knowledge during software development can be very useful to support software development activities, I defined the concept of "Domain-Oriented Software Development Environment" (DOSDE). This kind of environment readies knowledge about a specific domain in a symbolic representation (a domain ontology). It also considers a library of potential tasks from the domain to support problem understanding. Domain-specific tools support the use of this knowledge during a well-defined software process.

1993-1995 **Master in System Engineering and Computer Science**
University: Federal University of Rio de Janeiro (Brazil)
Title: *Quality assurance of Expert Systems*
Advisor: Ana Regina Cavalcanti da Rocha

1989-1993 **B.Sc. in Computer Science**
Federal University of Bahia, Salvador, BA, Brazil

Professional Experience

09/2009 - Current **Associate professor**
University of Valenciennes and Hainaut-Cambrésis, France

12/2008 - 05/2009 **Post-doctoral INRIA**
University of Rennes (IRISA), France
Subject: *Transformation model with Kermeta (Faros project)*
Responsible: Professor Jean-Marc Jezequel

02/2001 - 07/2009 **Associate professor**
Catholic University of Brasilia, Brazil

03/2005 - 12/2007 **Scientific team member of the Brazilian Software Quality Standard**
SOFTEX, Brazil

01/2000 – 12/2007 **Consulting in Software Quality**
Brazilian enterprises

01/2000 – 12/2000 **Post–doctoral**
Federal University of Rio de Janeiro, Brazil
Subject: *Domain-oriented software development environment*
Responsible: Ana Regina Cavalcanti da Rocha

09/1997-07/1998 **Visiting scholar**
University of Ottawa, Canada
Subject: *Knowledge representation with ontologies*
Responsible: Professor Stan Matwin

08/1995 - 02/ 1998 **Substitute teacher**
Catholic University of Bahia, Bahia, Brazil

05/1993 – 12/1993 **Substitute teacher**
Federal University of Bahia, Brazil

Teaching

Institution	Level	Course	Year	CM	TD	TP	EqTD
UFBA, Bahia, Brazil	BSc. in Accounting	Introduction to informatics*	1993	60			60
	BSc. in Computer Science	Software engineering*	1995 to 1997	240			240
UCB, Brasília, Brazil *	BSc. in Computer Science	Coding*	2001	272			272
		Software Engineering Fundamentals *	2001 to 2007	748			748
		Software Quality*	2001 to 2007	748			748
	Master in Knowledge Management and Information Technology	Special topics in Software Engineering*	2000	60			60
		Software Quality	2001 to 2006	340			340
		Methods and techniques on software engineering*	2003 to 2007	272			272
EPITECH, Nantes, France	Education in Informatics	Project supervision	2008	3	52		56.5
		Project management		2	20		23
IUT UVHC, Maubeuge, France	DUT on Computer Science - *Initial Formation*	Software Quality	2009 to 2013	27	36,5	39	116
		Introduction do UML (Unified Modeling Language)		30	48	90	183
		Advanced UML		30	64	57	166
		Monitoring and training project			9		9
	DUT on Computer Science – *Learning Formation*[1]	Software Quality	2009 to 2013	32	8	24	80
		Introduction do UML (Unified Modeling Language)		40	20	51	131
		Advanced UML		-		18	18
Total of teaching hours (approximate)							3.466

* In Brazil, courses are not divided into CM (*Cours Magistral*), TD (*Travail Dirigé*), TP (*Travail Pratique*).

[1] In French, *Formation par Apprentissage*

Administrative Activities

2006 - 2007 **Coordination of the group of associate professors who implanted the Brazilian software quality standard (MPS.BR)**
Institution: Catholic University of Brasília, Brazil
Activities: Responsible for making contact with companies, definition of contracts, payment verification.

2001 - 2004 **Coordination of the software engineering courses in the B.Sc.**
Institution: Catholic University of Brasília, Brazil
Activities: Responsible for the distribution of teachers in the disciplines, monitoring student progress and definition of concepts to acquire for each discipline.

2003 **Assistant director of the Master of Knowledge Management and Information Technology**
Institution: Catholic University of Brasília, Brazil
Activities: time schedule courses planning, scientific reporting for the master evaluation by educational government agencies

2001-2002 **Coordinator of the professional license in Software Engineering (L3)**
Institution: Catholic University of Brasília, Brazil
Activities: Definition of course selection, contract management of teachers, student progress monitoring, payment control, schedule course planning

1995 - 2000 **Research assistant**
Institution: Federal University of Rio de Janeiro, Brazil
Activities: Selection and recruitment of trainees in several research projects

Graduate Supervision

Doctoral Co-Supervision (3)

- Firas Bacha[2], Integration of content personalization in the design and generation of interactive applications based on MDA, (*Intégration de la personnalisation du contenu dans la conception et la génération des applications interactives basée sur une approche MDA*), co-supervised with Pr. Mourad Abed (50%), University of Valenciennes, France. (2009-2013) (Publications: 2 papers in international journals, 5 in international conference and 1 in national conference)
- Cristiane Soares Ramos, A strategy for analysing benefits of Software Process Improvement programs *(Uma estratégia para análise de benefícios em programas de melhoria de processo de software*), co-supervised with Pr. Ana Regina Cavalcanti da Rocha (50%), Federal University of Rio de Janeiro, Brazil. (2011- on going). (Publications: 2 papers in international conferences and 1 in national conference)
- Ahlem Assila, Conception of a tool for the Integration of Qualitative and Quantitative Evaluation for HCI (*Intégration de l'Évaluation Qualitative and Quanttative pour Interaction Homme-Machine*), co-supervised with Pr. Houcine Ezzedine (50%), University of Valenciennes. (2013 - on going). (Publications: 3 papers in international conferences)

Masters (Co) Supervision (28)

- Rainara Maia Santos. Quality Evaluation of Ubiquitous System by Measurements. Federal University of Ceará, 2014. (co-supervided with Rossana Andrade) (Publications: 1 paper in international conference)
- Natasha Malveira Costa Valentim, A Set of Reading Techniques for the Usability Inspection in Design Models, Federal University of Amazonas (Brazil), 2013. (co-supervided with Tayana Uchôa Conte) (Publications: 1 paper in national conference)
- Wagner Lindberg Baccarin Arnaut. Definition of an ontology for use in a service-oriented architecture development time and independent of the implementation technology, Catholic University of Brasilia (Brazil), 2008 (Publications: 1 paper in international conference)
- Luis Felipe Salin Monteiro. Definition of a catalog of measures for performance analysis of software processes, Catholic University of Brasilia (Brazil), 2008 (Publications: 1 paper in international journal)
- Rodrigo Duran Lima, Guidelines for the use of methods of investment assessment in the context of legacy systems maintenance. Catholic University of Brasilia (Brazil), 2008. (co-supervised with Nicolas Anquetil)
- André Luiz Pimentel Queiroz, Proactive assessment of the deterioration of information systems through management measures, Catholic University of Brasilia (Brazil), 2008. (Publications: 1 papers in national journal)

[2] Currently, Firas Bacha works as Eclipse modeling consultant at Thales Research and Technology, Paris, France.

- Alexandre Henrique de Souza Cruz, Capture and Dissemination of Knowledge in Software Projects, Catholic University of Brasilia (Brazil), 2007. (co-supervised with Nicolas Anquetil) (Publications: 1 book, 1 paper in international conference)
- Rodrigo Pinheiro dos Santos, Construction and Validation of an Instrument for Analysis of Quality of Service Companies Assessed at CMM/CMMI, Catholic University of Brasilia (Brazil), 2007. (Publications: 1 paper in international journal)
- Sinésio Teles de Lima, Evaluation of the Accessibility of Web Sites by Software Metrics, Catholic University of Brasilia (Brazil), 2007. (co-supervised with Fernanda Lima) (Publications: 2 papers in international conferences)
- Anderson Itaborahy, Project Management Software based on Business Value. Catholic University of Brasilia (Brazil), 2007. (co-supervised with Rildo Silva) (Publications: 1 paper in international conference)
- Mirian Cristiane Alves Brito, Integrating Instructional Material and Experience in a Collaborative Environment Support Teaching in Scope Issue Areas in a Higher Education Institution, Catholic University of Brasilia (Brazil), 2006. (co-supervised with Germana Nóbrega) (Publications: 1 paper in international conference and 1 in national conference)
- Regina Teixeira Almeida, Lessons Learned and Perceived Climate for Learning in a Software Factory, Catholic University of Brasilia (Brazil), 2006. (co-supervised with Julia Pantoja)
- Ricardo Ajax Dias Kosloski, Continuous Improvement Effort Estimation for Software Development: An Approach to Productivity, Catholic University of Brasilia (Brazil), 2005. (Publications: 1 paper in international conference and 1 in national conference)
- Kênia Pereira Batista Webster, Risks of Software Maintenance: Taxonomy and Prioritization. Catholic University of Brasilia (Brazil), 2005. (co-supervised with Nicolas Anquetil) (Publications: 1 paper in international conference and 1 in national conference)
- Sérgio Cozzetti Bertoldi de Souza, Essential Documentation: A Focus on Required Documentation for Software Maintenance. Catholic University of Brasilia (Brazil), 2005. (co-supervised with Nicolas Anquetil) (Publications: 1 paper in international conference and 1 in national conference)
- Cristiane Soares Ramos, Quality Evaluation of Legacy Systems, Catholic University of Brasilia (Brazil), 2004. (Publications: 1 paper in international conference and 1 in national conference)
- Kleiber Damian de Souza, Capturing Knowledge in Software Maintenance, Catholic University of Brasilia (Brazil), 2004. (co-supervised with Nicolas Anquetil) (Publications: 1 paper in international journal, 1 paper in international conference, 1 in national conference)
- Edmeia Leonor Pereira de Andrade, Use Case Points and Function Points in managing size estimation of software projects object-oriented, Catholic University of Brasilia (Brazil), 2004. (Publications: 1 paper in national journal, 1 in national conference)
- Paulo Roberto Corrêa Leão, Organization of Knowledge Management Skills for Information Technology Professionals, Catholic University of Brasilia (Brazil), 2004. (co-supervised with Eduardo Moresi) (Publications: 1 paper in national conference)
- Fabiano Mariath D'Oliveira, Planning Approval in Software Testing: A Customer Oriented Approach Catholic University of Brasilia (Brazil), 2003. (Publications: 1 paper in national conference)
- Angelica Toffano Seidel Calazans, Size Measurement Systems for Data Mart, Catholic University of Brasilia (Brazil), 2003. (Publications: 1 paper in international conference, 1 in national conference)

- Marcio Greyck Batista Dias, Organization of Knowledge Used in Software Maintenance, Catholic University of Brasilia (Brazil), 2003. (co-supervised with Nicolas Anquetil) (Publications: 2 papers in international journal, 2 papers in international conference, 1 book chapter, 1 in national conference)
- Carla Pena, An Approach to the Quality Inspection Object-Oriented Projects, Catholic University of Brasilia (Brazil), 2003. (co-supervised with Walcélio Melo)
- Fabio Zlot. Modeling Task Knowledge for Domain-Oriented Development Software Environments. Federal University of Rio de Janeiro (Brazil), 2002. (co-supervised with Ana Regina Cavalcanti da Rocha) (Publications: 1 paper in international journal, 1 paper in international conference)
- Ana Mirtes Fouro. Supporting the Construction of Research Database in Domain-Oriented Development Software Environments. Federal University of Rio de Janeiro (Brazil), 2002. (co-supervised with Ana Regina Cavalcanti da Rocha)
- Augusto Gomes. Software Processes Evaluation based on Measurements. Federal University of Rio de Janeiro (Brazil), 2001. (co-supervised with Ana Regina Cavalcanti da Rocha) (Publications: 1 paper in international conference, 1 in national conference)
- Catia Gallota. Neptune: A Domain-Oriented Development Software Environment for Submarine Acoustic. Federal University of Rio de Janeiro (Brazil), 2000. (co-supervised with Ana Regina Cavalcanti da Rocha) (Publications: 1 paper in international journal, 1 paper in international conference)
- Luis Filipe Machado. Model for Definition, Specialization and Instantiation of Process in TABA Workstation. Federal University of Rio de Janeiro (Brazil), 2000. (co-supervised with Ana Regina Cavalcanti da Rocha). (Publications: 1 paper in international conference, 3 in national conferences)

Research Projects

Research Projects Coordination (3)

- A Measurement-based Approach for the Quality Evaluation of Human-Computer Interaction in Ubiquitous Systems, CNRS-INRIA-FAP's 2012 no 155673 – Brazil, Septembre/2012- July/2014.
  Participants: University of Valenciennes and Hainaut-Cambrésis, France
  Federal University of Ceará, Brazil
- A Model for Evaluating Accessibility of Web Sites, National Counsel of Technological and Scientific Development (Conselho Nacional de Desenvolvimento Científico e Tecnológico), MCT/CNPQ 15/2007 (Brazil), 2008-2009.
  Participants: Catholic University of Brasília, Brazil
  University of Brasília, Brazil
- Knowledge Management in Software Engineering (Conselho Nacional de Desenvolvimento Científico e Tecnológico), PDPG-TI CT-INFO CNPQ (Brazil), 2002 - 2005.
  Participants: Catholic University of Brasília, Brazil
  Federal University of Espírito Santo, Brazil

Research Project Participation (3)

- Context-Awareness Testing for Ubiquitous Systems, Brazil National Counsel of Technological and Scientific Development (Conselho Nacional de Desenvolvimento Científico e Tecnológico), MCT/CNPQ 14/2013 (Brazil), 2014-2016.
  Participants: Federal University of Rio de Janeiro, Brazil
  Federal University of Ceará, Brazil
  University of Valenciennes and Hainaut-Cambrésis, France
- Development of an Expert System for Cardiology, IBM Brazil-FINEP (Brazilian Agence for Inovation), 1993-1997.
  Participants: Federal University of Rio de Janeiro, Brazil
  Foundation of Cardiology from Bahia, Brazil
- CardioEducar – an educational meta-environment for cardiology teaching, National Counsel of Technological and Scientific Development (Conselho Nacional de Desenvolvimento Científico e Tecnológico), 1999-2001.
  Participants: Federal University of Rio de Janeiro, Brazil
  Foundation of Cardiology from Bahia, Brazil

Scientific Expertise

Reviewer of Journals (6)

- Expert System with Applications (Elsevier), 2014.
- Journal of Systems and Software (Elsevier), 2010, 2013.
- Journal of Software: Evolution and Process (Wiley), 2012, 2013.
- Engineering Applications of Artificial Intelligence (Elsevier), 2013, 2014.
- Ingénierie des Systèmes d'Information (Hermes), 2013.
- Journal of System Enginering Research and Development (Springer), 2013, 2014.

Invited Speaker for Tutorial in Conferences (3)

- Evaluating the Quality of Human-Computer Interaction: Facing new challenges, in Software Quality Brazilian Symposium, June 2012 (in Portuguese)
- Software Measurement for Quality and Productive Improvement, in Software Quality Brazilian Symposium, June 2005 (in Portuguese)
- Knowledge modeling in organizations with ontologies, in Software Engineering Brazilian Symposium, June 2001 (in Portuguese)

Member of Conference Program Committees (11)

- INFORSID, INFormatique des ORganisations et Systèmes d'Information et de Décision, 2010, 2011, 2012, 2013, 2014
- ACM CHI Conference on Human Factors in Computing Systems, Works-in-Progress, 2013, 2014
- International Conference on the Quality of Information and Communications Tecnhology, Track on Evidence Based Software Quality Engineering, 2014
- International Conference on Internet and Web Applications and Services, 2014
- Conférence Francophone sur l'Interaction Homme-Machine, 2013, 2014
- International Conference on Advanced Logistics and Transport, 2013
- Human-Computer Interaction International, 2011, 2012.
- Simpósio Brasileiro de Qualidade de Software, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014
- International Conference Interfaces and Human-Computer Interaction, 2012, 2013, 2014
- XVIII Simpósio Brasileiro de Engenharia de Software 2002, 2003, 2004, 2008, 2009
- IADIS International Conference WWW/Internet. 2002, 2003, 2004
- International Software Metrics Symposium. 2003
- Jornada Ibero-Americana de Engenharia de Software e Engenharia de Conhecimento, 2002
- Ibero American Conference on Web Engineering. 2002
- Workshop on Software Quality, 2006, 2007, 2008, 2009.

Member of Doctoral Committees (4)

- Mlle. Sarah AYAD, supervised by Pr. Isabelle Comyn-Wattiau and Mme. Samira Si-Said Cherfi, (Title: Business Process Models Quality: Evaluation and Improvement), Le Conservatoire National des Arts et Métiers, Paris, 2013.
- M. Pierre-Emmanuel Arduin, supervised by Pr. Camille Rosenthal-Sabroux, (Title: Towards a commensurability metric of interpretation schemas), Paris-Dauphine University, 2013.
- M. Quang-Minh Doan, supervised by Pr. Camille Rosenthal-Sabroux (Title: Preservation of knowledge in small and medium-sized Vietnamese companies: model and process), Paris-Dauphine University, 2012.
- M. Mariano Montoni, supervised by Pr. Ana Regina Cavalcanti da Rocha (Title: A Research on the Critical Success Factors for Software Process Improvement Initiatives), Federal University of Rio de Janeiro, 2010.

Member of selection committee for Associate Professor (2)

- Profile: Computer Science in Knowledge Management, Information Systems and Social Network - Paris-Dauphine University, Paris, France, 2014.
- Profile: The position is concerned with the consideration of semantic in the process of modeling and evolving computer software, MCF 1301 - University of Littoral, Côte D'Opale, Calais, France, 2013.

Coordination and Participation in Network Research Groups (2)

- Coordinator of the research group from GDR-I3 – Information System Evaluation, 2011-2013 (for more details see §1.1, page 26).
- Member of the research group GDR-GPL – Expert-User Modeling, 2012-2013.

Conference Organization (general chair and program committee chair) (1)

- Brazilian Symposium on Software Quality, Brasília, Brazil, 2004 (350 participants, 1 international keynote, two national keynotes, three collocated workshops, 5 tutorials).

International Workshop Organization (2)

- Ontology in Action Workshop, co-located event with the International Knowledge Engineering and Software Engineering Conference, Banff, Canada, 2004.
- Workshop on Software Quality, co-located event in the International Conference on Software Engineering, Orlando, USA, 2002.

Research and Consulting in Software Quality

- Co-editor of the Software Process Improvement Brazilian Standard (MR-MPS-SW), 2007.
- Implantation and Certification of the Software Process Improvement Brazilian Standard in enterprises, 2007-2008.

Publications

Summary	
International Journal Papers	8
National Journal Papers (Brazilian)	3
Editor of Journal Special Issue	1
Book	1
Book chapters	11
International conference papers	52
National conference papers	69
Doctoral dissertation	1

International Journal Papers (8)

Oliveira K.M., Bacha F., Mnasser H., Abed M. Transportation Ontology Definition and Application for the Content Personalization of User Interfaces. Expert Systems with Applications, 40 (8), 2013, pp. 3145-3159. [IF=2.203]

Monteiro L., Oliveira K.M. Defining a catalog of indicators to support process performance analysis. Journal of Software Maintenance and Evolution: Research and Practice, vol 23, Issue 6, 2011, pp. 395–422. [IF=0.606]

Bacha F., Oliveira K., Abed M. Using Context Modeling and Domain Ontology in the Design of Personalized User Interface. International Journal on Computer Science and Information Systems (IJCSIS), 6, ISSN 1646-369, 2011, pp. 69-94.

Santos, R. P. Dos, Oliveira K. M. De, Silva, W. P. da., Evaluating the Service Quality of Software Providers appraised in CMM/CMMI, Software Quality Journal, Springer Verlag, v. 17 (3), September 2009, pp. 283-301. [IF=0.977]

Anquetil, N., Oliveira, K. M., Souza, K. D. , Dias, M. Software Maintenance Seen as a Knowledge Management Issue. Information and Software Technology, v. 49 (5), 2007, pp. 515 - 529. [IF=1.692]

Oliveira, K. M., Zlot, F., Rocha, A. R., Travassos, G. H., Gallota, C., Menezes, C. Domain-oriented software development environment. Journal of Systems and Software, v. 172, 2004, pp.145-161. [IF=1.322]

Dias, M., Anquetil, N., Oliveira, K. M. Organizing the Knowledge Used in Software Maintenance. Journal of Universal Computer Science, v. 9, 2003, pp. 641-658. [IF=0.76]

Rabelo Jr, A., Rocha, A.R., Oliveira, K. M. Ximenes, A., Souza, A., Andrade, C., Onnis, D., Lobo, N., Ferreira, N., Werneck, V. An Expert System for the Diagnosis of Acute Myocardial Infarction with EKG Analysis. Artificial Intelligence in Medicine, v. 1, 1997, pp. 75-92. [IF=1.767]

National Journal Papers (3)

Queiroz, A., Anquetil, N. Oliveira, K.M. Avaliação pró-ativa da deterioração de sistemas de informação por meio de medidas de gestão, Revista de Informática Téorica e Aplicada, v. 16, n. 1, 2009, pp. 45-68.

Souza, S. C. B., Anquetil, N., Oliveira, K. M. Which documentation for software maintenance? Journal of the Brazilian Computer Society, v. 3, 2007, pp. 31-44.

Andrade, E. L. P. De, Oliveira, K. M. Aplicação de Pontos de Função e Pontos de Casos de Uso de Forma Combinada no Processo de Gestão de Estimativa de Tamanho de Projetos de Software Orientados a Objetos. IP. Informática Pública, v. 7, n. 1, 2005, pp.13-30.

Editor of Journal Special Issue (1)

Oliveira K., Rosenthal-Sabroux C Numéro spécial Évaluation des Systèmes d'information, Ingénierie des systèmes d'information (ISI), 18 (3), Hermes, Paris, ISBN 1633-1311, 2013.

Book (1)

Torres, A.H., Oliveira, K.M., Anquetil, N., Lucena, G. Historias de aprendizagem em projetos de software: da teoria à prática. Editora Universa, Brasília, DF, Brasil, 2009.

Book Chapters (11)

Kolski, C., Labour, M., Lepreux, S., Oliveira, K. M. A Pedagogic Patterns Model approach to designing and documenting educational good practice in HCI. Seffah, A., The Patterns of HCI Design and the HCI Design of Patterns. Human Computer Interaction Series, Springer Verlag, 2014. (accepted for publication)

Bacha F., Oliveira K., Abed M. Context-Aware MDA Approach for Content Personalization in User Interface Development. Diaz V.G., Lovelle J.M.C., Garcia-Bistelo B.C.P, Martinez O.S., Progressions and Innovations in Model-Driven Software Engineering, IGI Global, 2013, pp. 88-105.

Oliveira, K. M., Villela, K., Rocha, A., Travassos, G.H. Use of Ontologies in Software Development Environments. Ontologies for Software Engineering and Software Technology, ed. Heidelberg: Springer, v. 1, 2006, pp. 275-309.

Anquetil, N., Oliveira, K. M., Dias, M. Software Maintenance Ontology. Ontologies for Software Engineering and Software Technology ed.Heildelberg: Springer, 2006, v. 1, pp. 153-173.

Rocha, A.R., Rabelo Jr, A., Monat, A., Oliveira, K. M., et. al. A construção de um Meta-ambiente Educacional para Cardiologia. Internet e Educação a Distância, ed. Salvador: EDUFBA, 2002, pp. 337-371.

Rocha, A.R., Machado, L. F., Oliveira, K. M., Falbo, R. Automatização da Definição de Processos de Software. Qualidade de Software: Teoria e Prática ed.: Prentice Hall, 2001.

Belchior, A., Oliveira, K. M., Cerqueira, A., Rocha, A. R. Definição de Requisitos de Qualidade de Software. Qualidade de Software: Teoria e Prática ed.: Prentice Hall, 2001.

Rocha, A.R., Maidantchick, C., Oliveira, K. M., Travassos, G.H. Experiência em Definição, Uso e Melhoria de Processos de Software. Qualidade de Software: Teoria e Prática ed.: Prentice Hall, 2001.

Oliveira, K. M., Rocha, A.R., Rabelo JR, Á. Qualidade de Software Médico. Qualidade de Software: Teoria e Prática: Prentice Hall, 2001.

Oliveira, K. M., Lima, R., Rocha, A.R. Qualidade de Software Web. Qualidade de Software: Teoria e Prática ed.Prentice Hall: Prentice Hall, 2001.

Oliveira, K. M., Werneck, V., Ximenes, A., Rocha, A.R. Gerência da Qualidade do Processo e do Produto em Sistemas Especialistas: uma experiência bem sucedida. Qualidade de Software - Seleção de Textos, 1996, v. 1, pp. 23-43.

International Conference Papers (52)

Assila, A., Oliveira, K.M.; H.Ezzedine, K.M. Towards qualitative and quantitative data integration approach for enhancing HCI quality evaluation, 16th International Conference on Human-Computer Interaction, Springer, 22-27, June 2014, Creta, Greece, June 2014. (accepted for publication)

Santos, R., Oliveira, K., Andrade, R., Santos, I., Lima, E. A Quality Model for Human-Computer Interaction Evaluation in Ubiquitous Systems. 6th Latin American Conference on Human Computer Interaction, Lecture Notes in Cumputer Science, n. 8278, Guanacaste, Costa Rica, December 2013, pp. 63-70.

Gabillon Y., Lepreux S., Oliveira K. Towards ergonomic User Interface composition: a study about information density criterion. M. Kurosu, 15th International Conference on Human-Computer Interaction, HCI International, Lecture Notes in Computer Science, Las Vegas, USA, July 2013, pp. 211-220.

Ramos, C., Oliveira, K. M., Rocha, A. Towards a strategy for analysing benefits of Software Process Improvement programs. The 25th International Conference on Software Engineering and Knowledge Engineering, Boston, USA, June 2013.

Assila, A., Ezzedine, H., Oliveira, K., Bouhlel, M. Towards improving the subjective quality evaluation of Human Computer Interfaces using a questionnaire tool. International Conference on Advanced Logistics and Transport, Sousse, Tunisia, May 2013, pp. 275-283.

Bacha, F., Oliveira, K., Abed, M. Supporting models for the generation of personalized user interfaces with UIML. Software Support for User Interface Description Language, UIDL'2011, Interact'2011 workshop, Lisbon, Portugal, Septembre 2011.

Kolski, C., Uster, G., Robert, J., Oliveira, K., David, B. Interaction in mobility: the evaluation of interactive systems used by travellers in transportation contexts. International Conference, HCI International, Lecture Notes in Computer Science, n. 6763, Orlando, USA, July 2011, pp. 301-310.

Bacha, F., Abed, M., Oliveira, K. Providing Personalized Information in Transport Systems: A Model Driven Architecture Approach. First IEEE International Conference on Mobility, Security and Logistics in Transport, Hammamet, Tunisia, June 2011, pp. 452-459.

Bacha, F., Oliveira, K., Abed, M. A model driven architecture approach for user interface generation focused on content personalization. Fifth IEEE International Conference on Research Challenges in Information Science, Guadeloupe - French West Indies, France, May 2011, pp. 1-6.

Oliveira, K. New Research Challenges for User Interface Quality Evaluation. Conference Internationale Francophone sur l'Interaction Homme-Machine, Luxembourg, Septembre 2010, pp. 145-148.

Bacha, F., Oliveira, K., Abed, M. Foundations of a Model Driven Engineering Approach for Human-Computer Interface Focused on Content Personalization. The 11th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems, Valenciennes, France, Septembre 2010.

Mnasser, H., Oliveira K., Khemaja M., Abed M. Towards an Ontology-based Transportation System for User Travel Planning. P. Borne, F.G. Filip, 12th LSS Symposium, Large Scale Systems: Theory and Applications, Villeneuve D'Ascq, France, July 2010.

Mnasser, H., Khemaja, M., Oliveira, K., Abed, M. A public transportation ontology to support user travel planning, 4th International Conference on Research Challenges in Information Systems, Nice, France, May 2010, pp. 127-186.

Arnaut, W., Oliveira, K. Lima, F. OWL-Soa: A Service Oriented Architecture Ontology Useful During Development Time And Independent From Implementation Technology, 4th IEEE International Conference on Research Challenges in Information Systems, Nice, France, May 2010, pp. 527-536.

Houda, M., Oliveira, K., Khemaja, M., Abed, M. Towards An Ontology-Based Transportation System for User Travel Planning, IFAC LSS, July 2010.

Bacha, F, Oliveira, K., Abed, M. Foundations of a Model Driven Engineering Approach for Human-Computer Interface Focused on Content Personalization, 11th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems, Valenciennes, France, 2010.

Lima, S., Lima, F., Oliveira, K. M. Evaluating the Accessibility of Websites to Define Indicators in Service Level Agreements, 11th International Conference on Enterprise Information Systems, Milan, Italy, May 2009, pp. 858-869.

Itaborahy, A., Oliveira, K. M., Santos, R. Value-Based Software Project Management: a business perspective on software projects, 10th International Conference on Enterprise Information Systems, Bracelona, Spain, June 2008, pp. 218-225.

Lima, S., Lima, F., Oliveira, K.M., Towards Metrics for Web Accessibility Evaluation. International Workshop on Design & Evaluation of e-Government Applications and Services, Rio de Janeiro, Brazil, 2007.

Torres, A. H. S., Anquetil, N., Oliveira, K. M. Pro-active dissemination of knowledge with learning histories. 8th International Workshop on Learning Software Organization, Rio de Janeiro, Brazil, September 2006, pp. 19-27.

Rocha, A. R., Montoni, M., Santos, G., Oliveira, K. M., Natali, A., Mian, P., Conte, T., Mafra, S., Barreto, A., Barreto, A., Bianchi, F. Success Factors and Difficulties in Software Process Deployment Experiences based on CMMI and MR-MPS.BR, International Workshop on Learning Software Organizations, Rio de Janeiro, 2006, pp. 77-87.

Brito, M. C. A., Nóbrega, G.M., Oliveira, K. M. Integrating instructional material and teaching experience into a teachers' collaborative learning environment. European Conference on Technology Enhanced Learning. Lecture Notes in Cumputer Science, n. 4227, 2006, pp. 458-463.

Kosloski, R.A., Oliveira, K. M. An Experience Factory to Improve Software Development Effort Estimates. 6th International Conference Product Focused Software Process Improvement, Lecture Notes in Computer Science, n. 3597, Oulu, Finland, June 2005, pp. 560-573.

Webster, K.P., Oliveira, K. M., Anquetil, N. A Risk Taxonomy Proposal for Software Maintenance. IEEE International Conference on Software Maintenance, Budapest, Hungary, September 2005, pp. 453-461.

Souza, S.C.B., Anquetil, N., Oliveira, K. M.A Study of the Documentation Essential to Software Maintenance. International Conference on Design of Communication (SIGDOC), 2005, Conventry, UK, September 2005, pp. 68- 75.

Anquetil, N., Oliveira, K.M., Santos, A. Silva Jr., Pc., Araujo Jr., L., Vieira, S. Software Re-Documentation Process and Tool. 17th Conference on Advanced Information Systems Engineering, CAISE Forum, Porto, Portugal, June 2005, pp. 95-100.

Calazans, A.T.S., Oliveira, K. M., Santos, R. R. Adapting Function Point Analysis to Estimate Data Mart Size. 10th IEEE International Symposium on Software Metrics, Chicago, USA, September 2004, pp. 300-311.

Souza, K.D., Anquetil, N., Oliveira, K. M. Learning Software Maintenance Organizations. International Workshop on Learning Software Organizations, Lecture Notes in Cumputer Science, n. 3096, Banff, Canada, June 2004, pp. 67-77.

Ramos, C. S., Oliveira, K. M., Anquetil, N. Legacy Software Evaluation Model for Outsourced Maintainer. 8th IEEE European Conference on Software Maintenance and Reengineering, Tampere, Finland, March 2004, pp. 48-57.

Villela, K., Santos, G., Travassos, G.H.; Rocha, A.; Oliveira, K. CORDIS-FBC: An Enterprise Oriented Software Development Environment. Learning Software Organization - WM 2003 – Professionelles Wissensmanagement, Lecture Notes in Informatics, Lucerne, Switzerland, April 2003, pp. 91-96.

Anquetil, N., Oliveira, K. M., Dias, M., Ramal, M., Menezes, R. Knowledge for Software Maintenance. Software Engineering and Knowledge Engineering, San Francisco, USA, July 2003, pp. 6-68.

Dias, M., Anquetil, N., Oliveira, K. M. Organizing the Knowledge used in Software Maintenance. Learning Software Organization- Professionelles Wissensmanagement, Lecture Notes in Informatics, Lucerne, Switzerland, April 2003, pp. 65-72.

Valle, C, Costa, V., Garcia, A. C., Montoni, M., Rocha, Ar, Oliveira, K. M., Rabelo, L., Rabelo Jr, A. CardioMeeting: A Learning Environment to Support the Discussion of Scientific Papers in Cardiology. World Conference on Educational Multimedia, Hypermedia and Telecomnucation, Denver, USA, June 2002, pp. 573-577.

Zlot, F., Oliveira, K. M., Rocha, A. Modeling Task Knowledge to Support Software Development. Software Engineering and Knowledge Engineering, Ischia, Italy, July 2002, pp. 35-42.

Villela, K., Oliveira, K. M., Travassos, G.H., Rocha, A. R. The Definition and Automated Support of Software Processes, taking Domain Knowledge and Organizational Culture into Consideration. Workshop on Software Quality - ICSE 2002, Orlando, USA, May 2002.

Zlot, F., Oliveira, K. M., Rocha, A. Using the Description of Tasks to Support Software Development. 15th International Conference Software & Systems Engineering and their Applications, Paris, France, November 2002, pp. 1-8.

Villela, K., Zlot, F., Santos, G., Bonfim, C., Salvador, B., Oliveira, K. M., Travassos, G. H., Rocha, A. R. Knowledge Management in Software development Environments. 14th International Conference Software & Systems Engineering and their Applications, Paris, France, November 2001, pp. 1-8.

Oliveira, K. M., Ximenes, A., Matwin, S., Rocha, A.R. A Generic Architecture for Knowledge Acquisition Tools in Cardiology. 5th Intelligent Data Analysis in Medicine and Pharmacology - Workshop at the 14th European Conference on Artificial Intelligence, Berling, Germany, Augsut 2000, pp. 43 – 45.

Maidantchick, C., Oliveira, K. M., Vidal, H., Masiero, M. L. Applying Management Model and Ontology on a Telecommunication Company. 13th International Conference Software & Systems Engineering and their Applications, Paris, France, November 2000.

Rocha, C., Oliveira, K. M., Rocha, A.R., Montoni, M., Timbo, A., Sampaio, L. C., Rabelo Jr, A. CardioSurgery: An Environment to Support Surgical Planning and Follow-up in Cardiology. WebNet - World Conference on the WWW and Internet, San Antonio, USA, 2000, pp. 459-463.

Lima, R., Sampaio, F., Oliveira, K. M., Rocha, A.R. Evaluating Web Sites For an Educational Environment Target For Cardiology. The 3rd European Software Measurement Conference - Federation of European Software Metrics Associations, 2000.

Gomes, A., Machado, L. F., Oliveira, K. M., Rocha, A.R., Software Process Improvement Through Measurement And Experts Judgement. The 3rd European Software Measurement Conference - (Federation of European Software Metrics Associations), 2000.

Machado, L. F., Oliveira, K. M., Rocha, A.R. Using Standards And Maturity Models For The Software Process Definition. 4th International Software Quality Week Europe, Brussels, Belgium, November 2000.

Oliveira, K. M., Gallota, C., Menezes, C., Travassos, G.H, Rocha, A.R. Defining and Building Domain-Oriented Software Development Environments. 12th International Conference Software & Systems Engineering and their Applications, Paris, France, November 1999.

Oliveira, K. M., Menezes, C., Travassos, G.H, Rocha, A.R. Using Domain-Knowledge in Software Development Environments. Software Engineering and Knowledge Engineering, Kaiserslautern, Germany, June 1999, pp. 180-187.

Oliveira, K. M., Rocha, A.R., Travassos, G.H., Matwin, S. Towards a Domain Oriented Software Development Environment for Cardiology. 5th Doctorium Consortium CAISE, Pisa, Italy, June 1998.

Werneck, V., Oliveira, K. M., Rabelo Jr, A., Rocha, A.R. A Software Development Process for Expert Systems. 10th International Symposium on Methodologies for Intelligent Systems, Charlotte, USA, 1997, pp. 1-10.

Rabelo Jr, A., Rocha, A.R., Souza, A., Ximenes, A., Onnis, D., Oliveira, K. M., Olivaes, I., Lobo, N., Ferreira, N., Werneck, V. The Validation of an Expert System for Diagnosis of Acute Myocardial Infarction. Artificial Intelligence in Medicine, Lecture Notes in Artificial Intelligence, 1997, pp. 243-246.

Werneck, V., Oliveira, K. M., Rocha, A.R., Rabelo Jr, A., Souza, A., Lobo, N., Ximenes, A. A Software Process Based on ISO 9000-3 used for developing an Expert System in Cardiology. 2nd Medical Engineering Week of the World, Taipei, China, May 1996.

Oliveira, K. M., Rabelo Jr, A., Rocha, A.R., Souza, A., Ximenes, A., Andrade, C., Asanome, C., Onnis, D., Olivaes, I., Lobo, N., Ferreira, N., Werneck, V. An Experience of Software Quality Assurance for an Expert System in Cardiology. 2nd Medical Engineering Week of the World, Taipei, China, May 1996.

Rabelo Jr, A., Rocha, A.R., Oliveira, K. M., Werneck, V., Souza, A., Ximenes, A., Werther Filho, J., Lobo, N. An Expert System for Diagnosis of Acute Myocardial Infarction. ACM Symposium on Applied Computing (SAC), Nashville, Tennessee, February 1995, pp. 96-100.

Rabelo Jr, A., Rocha, A.R., Oliveira, K. M., Werneck, V., Souza, A., Ximenes, A., Lobo, N., Olivaes, I. An Expert System for Diagnosis of Acute Myocardial Infarction: Software Quality Assurance Procedures. European Symposium on Validation and Verification of Knowledge Based Systems, Chambéry, June 1995, pp. 117-127.

National Conference Papers (69)

Dupuy-Chessa, S., Oliveira, K.M, Si-Said Cherfi, S. Qualité des modèles : retour d'expériences. INFORSID'2014, INFormatique des ORganisations et Systèmes d'Information et de Décision, Lyon, France, May 2014. (accepted for publication).

Ramos, C., Oliveira, K. M., Rocha, A. Uma Abordagem para Análise de Retorno sobre Investimentos em Programas de Melhoria de Processos de Software. WAMPS, Workshop Anual do MPS.BR, Campinas, Brazil, October 2012, pp. 43-54.

Valentim, N.M.C., Oliveira K., Conte, T. Definindo uma abordagem para inspeção de usabilidade em modelos de projeto por meio de experimentação. IHC '12, Proceedings of the 11th Brazilian Symposium on Human Factors in Computing Systems, Cuiaba, Brazil, Novembre 2012, pp. 165-174.

Oliveira K., Thion V., Dupuy-Chessa S., Gervais M., Si-Said Cherfi S., Kolski C. Limites de l'évaluation d'un système d'information : une analyse fondée sur l'expérience pratique. INFORSID, INFormatique des ORganisations et Systèmes d'Information et de Décision, Montpellier, France, May 2012, pp. 411-427.

Kolski, C., Uster, G., Robert, J., Oliveira, K., David, B. Interaction in mobility: the evaluation of interactive systems used by travellers in transportation contexts. International Conference, HCI International, Lecture Notes in Computer Science, n. 6763, Orlando, USA, July 2011, pp. 301-310.

Bacha F., Oliveira K., Abed M. Transformation des modèles de BPMN vers UIML. Actes des 7èmes Journées sur l'Ingénierie Dirigée par les Modèles, Lille, June 2011, pp. 77-83.

Anquetil, N., Vizcaino, A., Oliveira, K. M., Piattini, M. Asynchronous Merging of Software Ontologies: an Experience. Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento, Lima, Peru, February 2007, pp. 99-106.

Lima, S. T., Lima, F., Oliveira, K. M. Avaliação da Acessibilidade de Sitios Web por meio de Métricas de Software. Simpósio Brasileiro de Qualidade de Software, Porto de Galinhas, Brazil, 2007, pp. 95-109.

Santos, R. P., Oliveira, K. M., Silva, W. P. Percepção dos Clientes sobre a Qualidade do Serviço de Provedores Formalmente Avaliados nos Modelos CMM/CMMI. Simpósio Brasileiro de Qualidade de Software, Fortaleza, Brazil, Junho 2007, pp. 203-217.

Crispim, E., Calixto, I., Brito, M., Nóbrega, G.M., Oliveira, K. M. Um ambiente Web para a captura de experiência docente baseado em Objetos de Aprendizagem e na colaboração: da concepção à prototipagem VIII Simpósio Brasileiro de Informática na Educação, Sào Paulo, Brazil, 2007, pp. 318-327.

Porto, D., Souza, L., Martinez, M., Anquetil, N., Oliveira, K. M. Avaliação da Confiabilidade de um Software utilizando Aspectos. V Simpósio Brasileiro de Qualidade de Software, Vila Velha, Brazil, June 2006, pp. 334-342.

Macedo, C. C., Lima, S., Rocha, A.R., Natali, A.C., Oliveira, K. M., Mian, P., Barreto, A., Barreto, A., Santos, G., Conte, T. Implantação de Melhoria de Processos de Software no Tribunal Superior Eleitoral. V Simpósio Brasileiro de Qualidade de Software, Vila Velha, Brazil, June 2006, pp. 351-358.

Nery, A., Bandeira, L., Lima, F., Oliveira, K. M. Qualidade de Software aplicada à navegabilidade na Web. Simpósio Brasileiro de Sistemas Multimídia e Web, Natal, Brazil, November 2006.

Kosloski, R.A.D., Oliveira, K. M. Melhoria Contínua de Estimativa de Esforço para o Desenvolvimento de Software: Uma Abordagem sobre Produtividade. V Simpósio Brasileiro de Qualidade de Software, Vila Velha, Brazil, June 2006, pp. 409-423.

Itaborahy, A., Radis, E., Longhi, F., Oliveira, K. M., Figueiredo, R. Aplicação do método SCAMPI para avaliação do processode gerenciamento de projetos de software numa instituiçãofinanceira. VII Simposio Internacional de Melhoria de Processo de Software, São Paulo, Brazil, November 2005, pp.1-26.

Ramos, C.S., Oliveira, K. M., Anquetil, N. Avaliação de Sistemas Legados. IV Simpósio Brasileiro de Qualidade de Software, Porto Alegre, Brazil, June 2005, pp. 139-144.

Brito, M., Nóbrega, G.M., Oliveira, K. M. Capturando experiência docente para guiar o design instrucional colaborativo e contínuo. XVI Simpósio Brasileiro de Informática na Educação, Juiz de Fora, Brazil, 2005, pp. 147-157.

Duran, R., Marinho, M., Figueiredo, R., Oliveira, K. M. Institucionalização da Gerência de Configuração no Desenvolvimento de Software de uma Organização. VII Simposio Internacional de Melhoria de Processo de Software, São Paulo, Brazil, Novembro 2005, pp. 60-85.

Sousa, M., Sândi, V.T., Oliveira, K. M., Figueiredo, R. Processo de Aquisição de Produtos e Serviços de Softwarepara uma Instituição Bancária. VII Simposio Internacional de Melhoria de Processo de Software, São Paulo, Brazil, November 2005, pp. 37-53.

Webster, K., Anquetil, N., Oliveira, K. M. Taxonomia de Riscos para Manutenção de Software. IV Simpósio Brasileiro de Qualidade de Software, Porto Alegre, Brazil, Junho 2005, pp. 119-134.

Leão, P.R., Oliveira, K. M., Moresi, E. Ontologia para gestão de competências dos profissionais em Tecnologia da Informação. IV Jornadas Iberoamericanas en Engenieria del Softfware e Ingenieria del Conocimento, Madrid, Spain, 2004, v. 2, pp. 651-655.

Webster, K., Oliveira, K. M., Anquetil, N., Figueiredo, R. Priorização de riscos para manutenção de software. IV Jornadas Iberoamericanas en Engenieria del Softfware e Ingenieria del Conocimento, Madrid, Spain, 2004, pp. 647-660.

Nascimento, A. A., Knoth, C. R., Fagundes, J. S., Figueiredo, R., Oliveira, K. M. Aplicações de Práticas Selecionadas do Nível 2 do Modelo eSCM-SP v2 em um Órgão da Administração Pública Federal Brasileira. 5th Conference for Quality in Information and Communications Tecnhology, Porto, Portugal, October 2004, pp.153-159.

Santos, R. P., Souza, S. M., Lima, S. T., Oliveira, K. M., Figueiredo, R., Knoth, C. R. Definição de SLA para Processos de Testes de Software - Um estudo de caso em unidade de teste de integração de sistemas. 5th Conference for Quality in Information and Communications Tecnhology, Porto, Portugal, October 2004, pp. 127-134.

Souza, S. C. B., Neves, W. C. G., Oliveira, K. M., Anquetil, N., Figueiredo, R. Investigação da documentação de maior importância para manutenção de software. IV Jornadas Iberoamericanas en Engenieria del Softfware e Ingenieria del Conocimento, Madrid, Spain, 2004, pp. 217-230

Weber, K.C., Rocha, A. R., Rouiller, A. C., Alves, A., Crespo, A., Goncalves, A., Paret, B., Vargas, C., Salviano, C., Oliveira, K. M. Uma Estratégia para Melhoria de Processo de Software nas Empresas Brasileiras. 5th Conference for Quality in Information and Communications, Porto, Portugal, October 2004, pp. 73-78.

Galotta, C., Oliveira, K. M., Rocha, A.R. Apoio à Interação entre Processos de Negócio e de Software através de Gerência de Conhecimento. III Simpósio Brasileiro de Qualidade de Software, Brasília, Brazil, Junho 2004, pp. 117-130.

Silva Neto, N., Ramos, F., Oliveira, K. M., Figueiredo, A. Avaliação da Melhoria da Qualidade de Vida no Trabalho com a Implantação do Nível 2 do Modelo SW-CMM. III Simpósio Brasileiro de Qualidade de Software, Brasília, Brazil, Junho 2004, pp. 94-101.

Souza, K.D., Anquetil, N., Oliveira, K. M. Captura de conhecimento durante a manutenção de software. II Workshop de Tecnologia da Informação e Gerência do Conhecimento, Brasília, Brazil, Junho 2004, pp. 1-10.

Ramos, C. S., Oliveira, K. M., Anquetil, N. Conhecendo Sistemas Legados através de Métricas de Software. III Simpósio Brasileiro de Qualidade de Software, Brasília, Brazil, Junho 2004, pp. 328-342.

Corrêa, G., Figueiredo, R., Oliveira, K. M., Araújo, J. Diretrizes para a Melhoria da Gerência e Desenvolvimento de Requisitos em uma Empresa de Software. Simpósio Internacional de Melhoriade Processos de Software, São Paulo, Brasil, Novembro 2004, pp. 95-106.

Calazans, A.T.S., Oliveira, K. M., Santos, R. Medição de Tamanho para Sistemas de Data Mart. III Simpósio Brasileiro de Qualidade de Software, Brasília, Brazil, June 2004, pp. 384-398.

Oliveira, K. M., Leão, P.R., Moresi, E. Ontologia de Competências Profissionais em Tecnologia da Informação. II Workshop de Tecnologia da Informação e Gerência do Conhecimento, Brasília, Brazil, Junho 2004, pp.1-10.

Weber, K.C., Rocha, A. R., Rouiller, A. C., Alves, A., Crespo, A., Goncalves, A., Paret, B., Vargas, C., Salviano, C., Oliveira, K. M. Uma Estratégia para Melhoria de Processo de Software nas Empresas Brasileiras. 5th Conference for Quality in Information and Communications, 2004, pp.73-78.

Andrade, E.L.P., Oliveira, K. M. Uso Combinado de Análise de Pontos de Função e POntos de Caso de Uso na Gestão de Estimativa de Tamanho de Projetos de Software Orientado a Objetos. III Simpósio Brasileiro de Qualidade de Software, Brasilia, Brazil, June 2004, pp. 234-248.

Souza, K.D., Oliveira, K. M., Anquetil, N. Aplicação do GQM para Avaliação de Processo de Manutenção de Software. III Jornadas Iberoamericana de Ingenieria del Software e Ingenieria del Conocimento, 2003, pp. 65-74.

Andrade, E.L.P., Calazans, A.T.S., Oliveira, K. M., Figueiredo, A. Avaliação do processo de atendimento de demandas de produtos de software da Embrapa. Simpósio Brasileiro de Qualidade de Software, 2003, pp. 92-99.

Calazans, A.T.S, Oliveira, K. M., Ribeiro, R. Dimensionando Data Marts: Uma Adequação de uma Métrica Funcional. Simpósio Brasileiro de Qualidade de Software, Fortaleza, Brazil, Junho 2003, pp. 189-202.

D'Oliveira, F.M., Oliveira, K. M., Figueiredo, Adelaide. Priorização de Testes de Software: Uma abordagem orientada ao cliente. Simpósio Brasileiro de Qualidade de Software, Fortaleza, Brazil, 2003, pp. 218-232.

Dias, M., Oliveira, K. M., Anquetil, N. Organização do Conhecimento Utilizado na Manutenção de Software. XXIX Conferencia Latinoamericana de Informática, La Paz, Bolívia, September 2003.

Anquetil, N., Oliveira, K. M. Processo de Redocumentação: Uma necessidade. Simpósio Brasileiro de Qualidade de Software, Gramado, Brazil, June 2002, pp. 30-43.

Carazza, A., Mello, W., Oliveira, K. M. Avaliação da Qualidade de Software: um estudo de caso. Workshop de Qualidade, Rio de Janeiro, Brazil, June 2001, pp.183–188.

Oliveira, K. M., Garcia, A. C., Valle, C., Montoni, M., Costa, V., Moraes, T., Rabelo Jr, A., Rocha, A. CardioMeeting: Um Ambiente para Discussão de Artigos em Cardiologia. 30a Jornadas Argentinas de Informática e Investigación Operativa - Simpósio Argentino de Informática y Salud - SADIO, Buenos Aires, Argentina, 2001, pp. 71-75.

Gomes, A., Oliveira, K. M., Rocha, A.R. Métricas para Medição e Melhoria de Processos de Software. 4o Encontro de Qualidade nas Tecnologias de Informação e Comunicações, 2001, Lisboa, Portugal, March 2001, pp. 71- 78.

Rocha, A.R., Oliveira, K. M., Rabelo Jr, A. Qualidade de Software Médico. 30o Jornadas Argentinas de Informática e Investigación Operativa - Simpósio Argentino de Informática y Salud, Buenos Aires, Argentina, 2001, pp.76-86.

Gomes, A., Oliveira, K. M., Rocha, A.R., Avaliação de Processos de Software baseada em Medições. XV Simposio Brasileiro de Engenharia de Software, Rio de Janeiro, RJ, October 2001, pp. 84-99.

Gomes, A., Mafra, S., Oliveira, K. M., Rocha, A.R., Avaliação de Processos de Software na Estação TABA. XV Simposio Brasileiro de Engenharia de Software, Rio de Janeiro, Brazil, October 2001, pp. 344-350.

Villela, K., Oliveira, K. M., Santos, G., Mafra, S., Travassos, G. H., Rocha, A.R., CORDIS-FBC: um Ambiente de Desenvolvimento de Software para Cardiologia. Workshop de Informática Médica, Rio de Janeiro, Brazil, October 2001, pp.1-5.

Villela, K., Santos, G., Gallota, C., Miranda, R., Negrão, R., Farias, L., Zlot, Fabio, Barcelos, M., Salvador, B., Oliveira, K. M., Travassos, G.H., Rocha, A.R. Estendendo a Estação TABA para criação de Ambientes de Desenvolvimento de Software Orientados a Organização. XV Simposio Brasileiro de Engenharia de Software, Rio de Janeiro, Brazil, October 2001, pp. 332-337.

Oliveira, K. M., Santos, G., Zlot, F., Guedes, G., Gallota, Catia, Cerqueira, A., Machado, L. F., Lima, K., Rapchan, F., Falbo, Ricardo, Travassos, G. H., Rocha, A.R.. Construção de Ambientes de Desenvolvimento de Software Orientados a Domínio na Estação TABA. Terceiro Workshop Ibero-americano de Engenharia de Requisitos e Ambientes de Software, 2000.

Machado, L. F., Oliveira, K. M., Rocha, A.R. Modelo para Definição de Processos de Software baseado na ISO/IEC12207, em Modelos de Maturidade e Características do Projeto. Terceiro Workshop Ibero-americano de Engenharia de Requisitos e Ambientes de Software, Cancun, Mexico, April 2000, pp.73-84.

Oliveira, K. M., Travassos, G.H., Rocha, A.R. A Estação TABA e Ambientes de Desenvolvimento de Software Orientados a Domínio. XIV Simposio Brasileiro de Engenharia de Software, João Pessoa, Brazil, October 2000, pp. 343-346.

Oliveira, K. M., Rocha, A.R., Rocha, A. R. Ambiente de Desenvolvimento de Software Orientado a Domínio. Simposio Brasileiro de Engenharia de Software, João Pessoa, Brazil, October 2000, pp. 275-290.

Oliveira, K. M., Travassos, G.H., Rocha, A.R. CORDIS: um Ambiente de Desenvolvimento de Software Orientado ao Domínio de Cardiologia. Congresso Brasileiro de Informática em Saúde, São Paulo, Brazil, October 2000.

Machado, L. F., Santos, G., Oliveira, K. M., Rocha, A.R. Def-Pro: Apoio automatizado para Definição de Processos de Software. Caderno de Ferramentas - XIV Simposio Brasileiro de Engenharia de Software, João Pessoa, Brazil, October 2000, pp. 359-362.

Machado, L. F., Oliveira, K. M., Rocha, A.R. Def-Pro: uma Ferramenta para Apoiar a Definição de Processo Padrão. XI Conferencia Internacional de Tecnologia de Software, 2000, pp. 165-179.

Gomes, A., Oliveira, K. M., Machado, L. F., Rocha, A.R. Medição e Melhoria de Processos de Software. Workshop de Qualidade de Software, João Pessoa, Brazil, October 2000, pp. 306-316.

Oliveira, K. M., Menezes, C., Travassos, G., Rocha, A.R. CORDIS: Assistência Automatizada no Desenvolvimento de Software em Cardiologia. 28 Jornadas Argentinas de Informática e Investigación Operativa, Buenos Aires, Brazil, 1999, pp. 38-48.

Cerqueira, A., Oliveira, K. M., Rocha, A.R., Apoio Automatizado para a Definição de Requisitos de Qualidade de Software. Workshop Qualidade de Software - XIII Simpósio Brasileiro de Engenharia de Software, Florianópolis, Brazil, October1999.

Oliveira, K. M., Rocha, A.R., Travassos, G.H., O uso da Teoria do Domínio no Processo de Desenvolvimento de Software. X Conferencia Internacional de Tecnologia de Software, Curitiba, Brazil, 1999, pp. 223-235.

Rabelo Jr, A., Rocha, A.R., Oliveira, K. M., Onnis, D., Ferreira, N., Lobo, N., Souza, A., Ximenes, A. Validação de Sistemas Especialistas com Casos Reais. Workshop de Qualidade de Software, São Carlos, São Paulo, October 1996, pp. 21-25.

Oliveira, K. M., Rocha, A.R., Rabelo Jr, A. Verificação e Validação de Sistemas Especialistas. XXI Conferencia Latinoamericana de Informática, 1995, pp.351-362.

Oliveira, K. M., Asanome, C., Rabelo Jr, A., Rocha, A.R., Avaliação da Confiabilidade de um Sistema Especialista para Diagnóstico de Infarto Agudo do Miocardio. Workshop de Qualidade de Software, Recifie, Brazil, October 1995, pp. 127-131.

Werneck, V., Oliveira, K. M., Rabelo Jr, A., Rocha, A.R. Definition, evaluation and improvement of a software process based on ISO 9000/3: The experience with an expert system for diagnosis of acute myocardial infarction. II International Congress on Information Engineering, 1995, pp. 324-333.

Rabelo Jr, A., Rocha, A.R., Souza, A., Oliveira, K. M. Testing an Expert System for Diagnosis of Acute Myocardial Infarction. II International Congress on Information Engineering, 1995, pp. 146-154.

Oliveira, K. M., Werneck, V., Rocha, A.R. Avaliação da Qualidade de Sistemas Especialistas. Workshop de Qualidade de Software, , Curitiba, Brazil, October 1994, pp. 7-9.

Oliveira, K. M., Werneck, V., Rocha, A.R.. Definição dos Requisitos de Qualidade de um Sistema Especialista em Cardiologia. Congresso Brasileiro de Informática em Saúde, Porto Alegre, Brazil, 1994, pp. 141-145.

Rabelo Jr, A., Rocha, A.R., Oliveira, K. M., Werneck, V., Souza, Agnaldo, Ximenes, Antonio, Lobo, Nelson, Olivaes, Ivan Experiência na Definição, Uso e Avaliação do Processo de Desenvolvimento de Sistemas Especialistas em Cardiologia. Workshop de Qualidade de Software, Curitiba, Brazil, October 1994, pp. 37-39.

Werneck, V., Oliveira, K. M., Rocha, A.R. Uma Experiência na Definição do Processo de Desenvolvimento de Sistemas Especialistas em Cardiologia. Congresso Brasileiro de Informática em Saúde, Porto Alegre, Brazil, 1994, pp. 146-150.

Doctoral Dissertation (1)

Oliveira, K. M., Model for the construction of Domain-Oriented Software Development Environments, Doctoral Thesis, Federal University of Rio de Janeiro, October 1999.

Part II. Software Quality Assurance by means of Methodologies, Measurements and Knowledge Management Issues

Chapter 1 Introduction

1.1 Context and Historical Research

Since the so-called software crisis of 1960s, 1970s and 1980s, the need of software quality assurance (SQA) has emerged and has expanded over the years with several methodologies, techniques and standards. SQA is defined by IEEE standard (1990) as "a planned and systematic pattern of all actions necessary to provide adequate confidence that an item or product conforms to established technical requirements" and "a set of activities designed to evaluate the process by which the products are developed or manufactured". From this first definition, research in the software quality domain expanded encompassing preoccupation with how to ensure quality of processes and software products generated in these processes. Standards and models for software process improvement have been defined (Software Process Improvement and Capability dEtermination- SPICE (EMAM *et al.*, 1998), ISO/IEC 12207 (2008), ISO/IEC 15504-2 (2003), Capability Maturity Model Integration - CMMI (SEI, 2010), and Brazilian Software Process Improvement (SOFTEX, 2012; Montoni *et al.* 2009) and have been increasingly applied in practice in the industry[3]. Quality of software products had to evolve with the introduction of new design methodologies (from structured to object-oriented design paradigms) and the emergence of new technologies (from personal computers to advanced technologies - for example, mobile and tablets). Software systems began to be everywhere supporting everyday activities and quality assurance of these systems has become crucial to ensure their real adoption by the users. This context implies the need of particular specifications not only for the different technologies but also for each application domain (e.g. health services, finance, insurance, education and transportation).

My personnel experience with SQA started in 1993 when I was designated to perform quality assurance of an expert system for the cardiology domain. This was a rich experience during my master degree that allowed me to not only learn how to define SQA procedures but also to really execute them. This experience has guided my career, in the Master and Doctorate research, and working as a professional (see Part I, section Overview).

In this section, I will present the context of software quality area intertwining with the research I have carried out throughout my career.

Masters in System Engineering and Computer Science

During my masters, I have worked on the definition and quality evaluation of software products using quality criteria and measures in a large project that aimed to develop an expert system for the cardiology domain. At that time, although evaluation techniques (e.g. walkthrough, inspections and code review) had already been proposed, the practice of SQA was focused on testing activities in the final code. In this project, we defined a set of quality

[3] Since 2006, more than 7,800 CMMI have been reported to CMMI institute from over 40 countries (see http://cmmiinstitute.com/resource/process-maturity-profiles/). In Brazil, there are more than 500 officially appraisals on the national standard (MPS.BR) created in 2005 (see http://www.softex.br/mpsbr/).

characteristics specific for expert systems **(Rabelo *et al.*, 1997; Oliveira *et al.*, 1996)**[4] and we used phased inspections (Knight and Myers, 1993) to assure the quality of the system throughout the software development process. Phased inspection defines that the same software product should be evaluated by different evaluators (in our case, technical team, managers and cardiologists) in different phases of the software development life cycle, considering different quality criteria. The inspection procedures were integrated in a software process specifically defined for the development of knowledge-based systems (**Werneck *et al.*, 1997**). Thanks to the SQA procedures, the first version of the system was concluded with a high-level of reliability (**Rabelo *et al.*, 1997**).

With this experience I realized that quality evaluation by measurements was a large research field. Quality evaluation of software products based on specific quality characteristics and measures have been a recurrent challenge. Several papers have been published for different technologies – e.g. object-oriented software systems (Orenyi *et al.*, 2012; Briand *et al.*, 1999; Harrison *et al.*, 1998), web systems (Kallepalli and Tian, 2001; Olsina and Rossi, 2002), and embedded systems (Mayr *et al.*, 2012) applied to several application domains. For some years, many papers proposed measures to evaluate software quality. The application of these measures in real projects became a need in order to validate them and to confirm their utility. So far, several difficulties emerged and they prevent software developers/evaluators to perform adequate validation of these measures. Some of these difficults are finding real projects to apply the evaluation procedures and qualified people to analyze and interpret the results to assure the quality of the software products. Measures were also defined for software process evaluation and improvement (SPI) (e.g. **Monteiro and Oliveira, 2011**; Wang and Li, 2005; **Gomes *et al.*, 2000**; Khoshgoftaar *et al.*, 1998) and to support project planning by the effort and cost estimation (e.g. De Marcos *et al.*, 2012; **Kosloski and Oliveira, 2005**; **Calazans *et al.*, 2004a**). In parallel to this, the dissemination of the empirical software engineering domain (Kitchenham *et al.*, 2002) brought a need of formalization of experiments. On the one hand it makes the experiments more credible, on the other hand it requires formal exigencies to follow (e.g. the formalization of the hypothesis, correct definition of subjects, treats of validity, etc.).

Doctorate in System Engineering and Computer Science

The expert system project revealed to me another important issue: the importance of knowledge domain for the software development. After a year working full time with cardiologists, I had got some knowledge about the cardiology domain used in information system development. As a consequence, I was often called to participate in the requirements elicitation and quality evaluation sessions of new projects in the cardiology hospital. Based on this finding and the experience with knowledge representation in the expert system development, I started to investigate how to organize the domain knowledge in a way that could support the software engineers in the software development and SQA activities. This investigation allowed me to extend the classical Software Development Environments (or Software Engineering Environments) to Domain-Oriented Software Development Environments (**Oliveira *et al.*, 1999a, 2004**).

Software Development Environment (SDE) is a computational system that provides support for the construction, management and maintenance of a software product (Brown *et al.*, 1992). An SDE consists of a repository that stores all the information related to the software project throughout its life cycle, and a set of tools that supports the technical and

[4] In this document, references in bold means that I am one of the authors.

managerial activities involved. DOSDE was defined to be a new class of SDE with two additional features: representing domain-knowledge and using this knowledge during software development. To represent the knowledge, we used domain ontologies (Gruber, 1995). To use it during the software development, specific domain activities were introduced in the traditional software processes to support the requirements elicitation, software design and quality evaluation (**Oliveira *et al.*, 2004**). Domain-oriented SDE was later on extended to the conception of Enterprise-oriented SDE (**Oliveira *et al.*, 2006**) that includes also the organizational knowledge to support knowledge management for software engineering.

In this context, the software quality community has largely explored knowledge management concepts. For instance, ontologies have been defined to organize the knowledge of software quality itself (e.g. Henderson-Sellers, 2014; Barcelos *et al.*, 2010; Kayed *et al.*, 2009); to improve the quality of business process models (Cherfi *et al.,* 2013); to facilitate standards-based Business-to-Business interoperability (Heravi *et al.*, 2014). Moreover, techniques for lessons learned capture and organization have been defined, applied and integrated in the software process (Cristal and Reis, 2006; Viravan, 1997); models and architectures for knowledge management have been proposed (Ji, 2011; Kim *et al.*, 2005). In particular, I explored the use of experience factories (Basili *et al.,* 1994b), a knowledge management infrastructure largely used in software engineering, for the educational (**Brito *et al.*, 2005, 2006**) and software engineering domains (**Kosloski and Oliveira, 2005**), and the capture and organization of lessons learned in software projects. Those experiences showed that much knowledge about the software quality activity itself must be captured, organized and shared to assist software designers in quality assurance activities.

Working as an associate professor

I spent a year (2000) in a post-doctoral position at the Federal University of Rio de Janeiro working on software quality procedures definition in the development of an educational meta-environment for cardiology (**Rocha *et al.*, 2000, Lima *et al.*, 2000**). I also co-supervised four master students in the domain of knowledge representation with ontologies (**Zlot *et al.*, 2002**) and quality evaluation of software process (**Gomes *et al.*, 2001a, 2001b, 2000; Machado *et al.*, 2000a, 2000b, Oliveira *et al.*, 1999b**).

From 2001 and 2009, I joined the Master Program of the Catholic University of Brasilia in Brazil. I supervised several master students investigating:

- The use of measurements and quality assurance procedures (**Monteiro and Oliveira, 2011; Santos *et al.*, 2004, 2007, 2009; Queiroz *et al.*, 2009; Andrade *et al.*, 2004, 2005; Lima *et al.*, 2007, 2009; Porto *et al.*, 2006; Macedo *et al.*, 2006; Nery *et al.*, 2006; Kosloski and Oliveira, 2005, 2006; Calazans *et al.*, 2004a, 2004b; Ramos *et al.*, 2004, 2005; Itaborahy *et al.*, 2005, 2008; Souza *et al.*, 2003, 2005, 2007; Duran *et al.*, 2005Nascimento *et al.*, 2004; Weber *et al.*, 2004; Corrêa *et al.*, 2004; Sousa *et al.*, 2005; D'Oliveira *et al.*, 2003, Dias *et al.*, 2003, Carazza *et al.*, 2001**);
- Knowledge management issues in search of software quality assurance (**Anquetil *et al.*, 2006, 2007; Torres *et al.*, 2006, 2009; Arnaut *et al.*, 2010; Brito *et al.*, 2005, 2006; Crispim *et al.*, 2007; Kosloski and Oliveira, 2005, 2006; Souza *et al.*, 2004, Oliveira *et al.*, 2004; Leão, 2004**).

During this period I also worked as a consultant in software quality assurance in Brazilian enterprises. I worked in the definition (***SOFTEX, 2006***), training and institutionalization of the Brazilian quality standard "MPS.BR". Nowadays, this standard is largely used in the Brazilian industry with more than five hundreds enterprises officially appraised.

In December 2008, I started a second post-doctoral research, which lasted for six months at the University of Rennes. This was an opportunity to return to the basics of my Doctorate research on modeling (and meta-modeling) in different application domains. Since September 2009, as an associate professor at the University of Valenciennes, LAMIH, I continue working with software quality assurance, but focused on the evaluation of interactive systems, and in particular in the Human-Computer Interaction (HCI) domain, one of the main themes of the LAMIH laboratory. I co-supervised a master student for the definition of a domain ontology for the transportation domain (**Mnasser *et al.,* 2010**); followed by a Doctoral student about the use of this domain ontology for the personalization of Human-Computer interfaces (**Bacha *et al.,* 2011a, 2011b, 2011c, 2011d, 2010, 2013; Oliveira *et al.,* 2013).** I also started to explore quality evaluation issues of interactive systems **(Gabillon *et al.,* 2013; Santos *et al.,* 2013; Valentin *et al.,* 2012; Kolski *et al.,* 2011; Oliveira *et al.,* 2012; Kolski *et al.*, 2012; Oliveira, 2010**). Currently, I am co-supervising two theses on the domain of measurements: one about HCI assessment (**Assila *et al.,* 2013, 2014**) and the other about return on investment for software process improvement[5] (**Ramos *et al.,* 2012, 2013**).

During two years, from October 2011 and December 2013, I have coordinated a research group in France about *Information System Evaluation*. Eighteen professionals (full professors, associate professors, and doctoral students) from nine laboratories from France discussed different challenges and the research being performed in the subject. Several papers (**Dupuy-Chessa *et al,* 2014**; Arduin *et al*, 2012**,** 2013**; Oliveira *et al.,* 2012**; **Kolski *et al.*, 2012**) and a special issue in a national journal (**Oliveira and Rosenthal-Sabroux, 2013**) were produced by the group.

Nowadays, the research on software quality assurance continues to explore the use of measurements. On the one hand, some researchers look, for instance, for the application of measures in small companies (e.g. Pino *et al.*, 2010, Sulayman *et al.*, 2012), the integration to the business process (Delgado *et al.*, 2014) and the investigation on quality characteristics and success factors of software process improvement (Kroeger *et al.*, 2014, Ianzen *et al.*, 2013). On the other hand, quality assurance for software systems has to be continually adapted to the modern technologies. The improvement in computational device miniaturization and in wireless communication has enabled relevant advances in new technologies, such as smart mobiles (e.g. Pranata *et al.*, 2013; Marinho *et al.*, 2013) and ubiquitous (e.g. **Santos *et al.*, 2013**; Traue and Kobayashi, 2011) systems development. Efforts in these new technologies expanded the place where the software system is used, and the classical mode of interaction beyond the desktop computers. Now one can use systems in everyday spaces changing the focus of interest from computer technology to the users and their needs. In this context, HCI issues became an area even more important than previously, where the main goal of software developers is to make computer systems really useful in the different situations encountered in the real world (Tang *et al.*, 2011).

Figure 1.1 summarizes my research since the first work on quality evaluation of expert systems (in 1993) to my main actual interests; that is, quality evaluation of interactive systems, with particular interest on HCI issues, and knowledge management for SQA. In this figure it is highlighted the research on quality of software products, quality of software process, knowledge management, and my current interests.

[5] This thesis is a co-supervision in Federal University of Rio de Janeiro, Brazil.

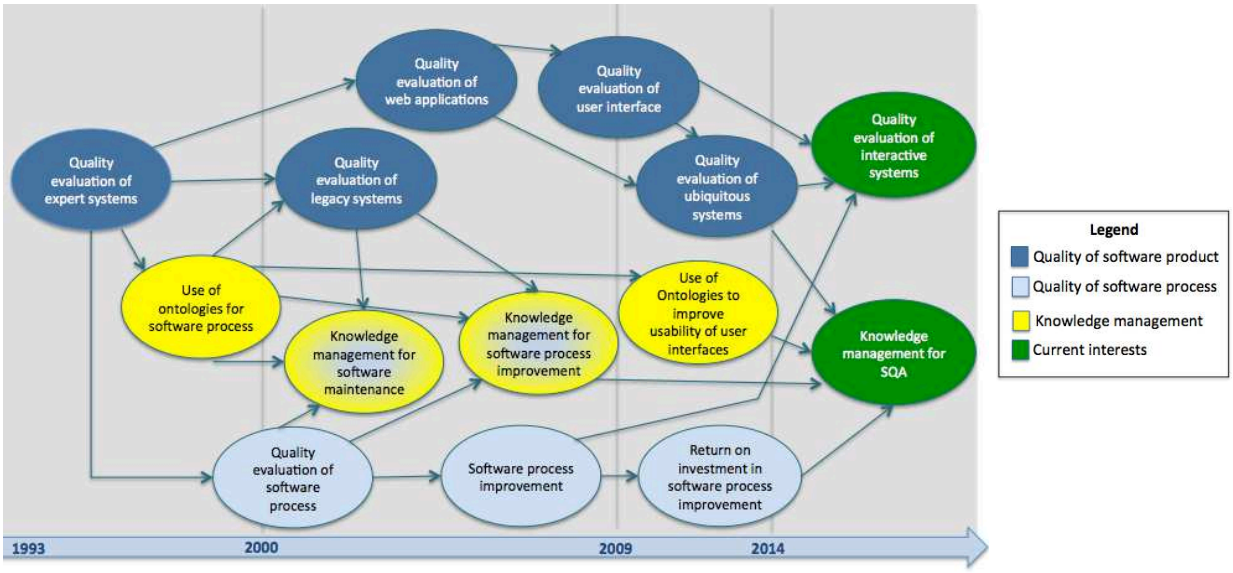


Figure 1.1. Research workflow

1.2 Objective and Research Challenges

Considering the context and historical research presented in the previous section, my research goal is:

> **To define, use and incorporate methodologies, measurements and knowledge management issues for the quality assurance throughout the software processes to enable their real adoption by organizations.**

To address this goal some research challenges are identified. These challenges were collectively identified in a research group about quality evaluation of information system, which I coordinated in France during the last almost three years. Eighteen researchers from nine French laboratories participated in this group and the final result of these challenges was published in a collective paper from **INFORSID (2012)**. I summarize these challenges in four primary concerns (Figure 1.2):

i) Definition of quality measures and their interdependencies

To produce software, different processes (business, development, organizational, etc.) are used and generate different types of deliverables (models, architecture, user interface, code, etc.) with different languages. SQA involves assessing all of these objects (deliverables and process) considering their own characteristics and, therefore, requires specific measures (for functionality, usability, security, etc.) according to the interests of stakeholders (analysts, end users, managers, etc.). In this context, the definition of thresholds is particularly difficult because they vary according to the needs that are often difficult to identify. Furthermore, these measures are implicitly linked, since they evaluate the same object from different angles. The identification of this interdependence has a direct impact on the analysis and interpretation of measured values in an evaluation.

Figure 1.2 Research Goal and Challenges

The definition of measures is not sufficient for evaluation. Great difficulty lies also on the analysis and interpretation of the measured results. Appropriate methods for collecting and analyzing the measured results must be defined. Several evaluation methods exist, but they need to be applied and, possibly, adapted for different types of software products and domain applications.

This challenge is even more pronounced for the Human-Computer Interaction (HCI) field. With the diversity of computer platforms (PC, PDA, smartphones, tablets, etc.), devices (e.g. mouse, laser point, 3D pointer), and modes of interaction (e.g. text, video, image, speech, touch screen), the quality assessment procedures must consider not only technical issues of computer science, but also human aspects, such as ergonomic and psychological issues about the interaction between users and interactive systems.

ii) Empirical experimentation for SQA

In order for results of case studies to be valid, they must be generalizable to other projects contexts, which are different from those to which the evaluation was carried out. In this case, empirical experiments must be implemented using quantitative and/or qualitative approaches. These experimentations generate a lot of data, images, statistics, text, graphics or videos. In some cases, the analysis and interpretation of these results require a collaborative effort of different professionals. For instance, for the HCI field, psychologists, ergonomics professionals and computer scientists should work together in the definition of the experimental protocols, analysis and interpretation of the results.

iii) Implementation of continuous software process improvement

To remain competitive, organizations must constantly improve the quality of their services and systems, reducing costs and production time, to get a better customer satisfaction and users. For this reason a program of quality continuous improvement should be established. This involve not only the definition of software processes, but also the evaluation and continual improvement of the software process and products.

With the wide use of interactive systems, concerns with continuous evaluation of HCI issues interaction from the initial stages of the software process becomes even more evident and require the actual integration of specific procedures in software process improvement models.

iv) Definition and use of knowledge management issues for SQA

Considering that SQA is a knowledge-intense activity, it is essential to investigate how to use knowledge management principles not only to support the SQA activities but also to manage the knowledge of the SQA activity itself; for example, exploring which knowledge of the evaluator can be used to support the SQA activity. This is particular important for HCI field, where different research areas are complementary. Investigating the knowledge used by the people of these areas can be a real advantage for the definition and implementation of procedures to ensure the quality of user interfaces.

1.3 Report Organization

This document is organized in four chapters besides this introduction.

Chapter 2 presents briefly the basic concepts of software quality assurance, exploring definition of measures, methodologies and the role of knowledge management in the context of software quality. The goal of this chapter is not to provide a state of the art about the subject, but to introduce the main concepts that will be used in the following chapters.

Chapters 3 and 4 present the research I have been working on in the area of software quality assurance since the conclusion of my Doctor degree. Although the research of my Masters and of my Doctoral degrees have had great importance in my professional work, they are not described in this report. Here, I focused on the research performed as an associate professor at the different universities[6].

Chapter 3 presents my research regarding software quality assurance practices considering the use of measurements and methodologies. This research addresses the three first challenges presented in the previous section.

Chapter 4 presents the use of knowledge and some knowledge management practices in the context of software quality assurance. This research seeks to answer the fourth challenge presented in the previous section.

Chapter 5 presents my conclusions summarizing the main contributions in the domain and research perspectives.

[6] University of Valenciennes and Hainaut-Cambrésis (09/2009 - current), Catholic University of Brasilia (2001-08/2009), Federal University of Rio de Janeiro (as a researcher, 10/1999-2000).

Chapter 2 Software Quality Assurance

2.1 Introduction

Software quality assurance is a very broad domain. A lot of methodologies, techniques, standards exist to support all the activities necessary to ensure the quality of software processes and products. The goal of this chapter is to introduce the main concepts of the domain and briefly describe some important approaches that were used in the research that will be presented in the next chapters.

This chapter starts with basic definitions about software quality, software quality assurance and measurements (section 2.2). Then, some approaches for measurement definition (section 2.3), standards and models for software quality are presented (section 2.4). Section 2.5 describes some concepts of knowledge management, models, and techniques for capturing, disseminating and organizing the knowledge. Finally, section 2.6 concludes the chapter by stating general questions that guided my research on this domain.

2.2 Basic Definitions

2.2.1 Software Quality

Several definitions can be found about quality in the context of software. Figure 2.1 presents definitions found in quality standards.

Some important aspects can be observed in these definitions. The first one is related to the object for which we qualify: a system, component or process (definition 1), or a set of programs, procedures, documentation and data (definition 4). To summarize, we can say that the object under consideration can be a *product* or a *process*. A *product* is a "result of a process" (ISO9000: 2000, 2000). It "includes intermediate products, and products intended for users such as developers and maintainers" (ISO/IEC 12207, 2008). Therefore, it can be the system, a component, a set of programs, documentation, data, or anything produced by the process or activities in the process with the goal of software production or maintenance. A *process*, in turn, is defined as "a set of interrelated or interacting activities that transforms inputs into outputs" (ISO 9000:2000, 2000). Thus, in this document, when we say software quality, it means quality of software process **and** quality of software product.

A second important aspect is about the conformance to what, in the definitions. It is considered the conformance to requirements (definitions 1 and 3); and to needs or expectations (definitions 2 and 4). These definitions reinforce what was defended by Crosby (1979) when he defined that quality "means conformance to requirements"; and, also by Juran (1990) when he defined that quality "means those features of products which meet customer needs and thereby provide product satisfaction". ISO/IEC 25000 (2005) defines requirement as "the expression of a perceived need that something be accomplished or realized", that may

be specified as a part of a contract, or by the development organization when a product is developed for unspecified users. In summary, the quality should be conforming to requirements that represent the users needs and expectations; thus, quality implies user/customer satisfaction.

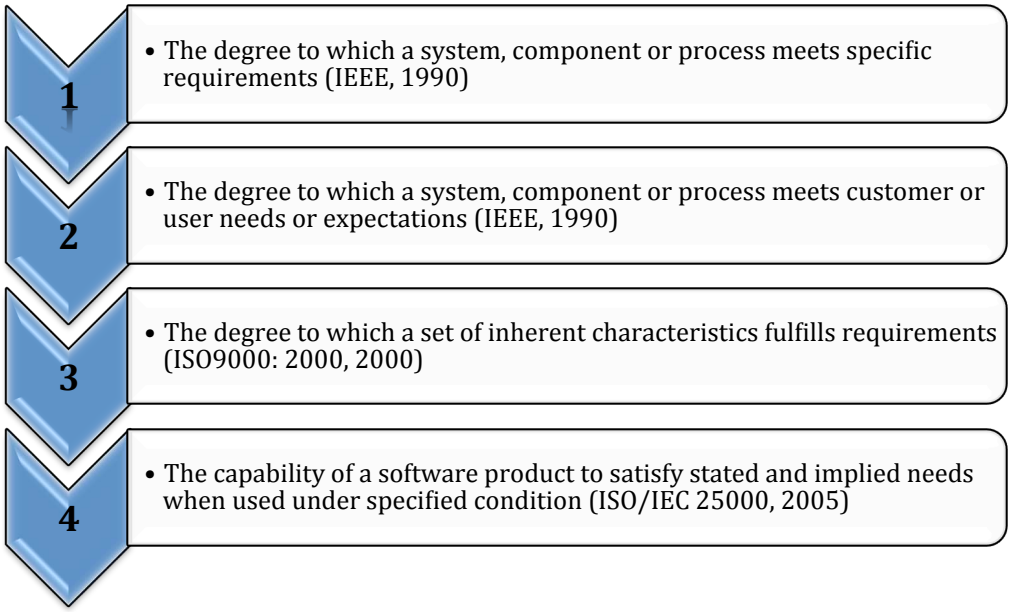


Figure 2.1. Quality definitions

The last important aspect is related to the inherent characteristics (definition 3). According to ISO/IEC 9000:2000 (2000), inherent characteristics are characteristics that are a natural part of something and cannot be separated from it. It opposes the term inherent to assigned and defines inherent as existing in something, especially as a permanent characteristic. There are many types of characteristics, such as: sensory (e.g. smell, touch, visual appearance), behavioral (e.g. courtesy, truthfulness), related to time (e.g. promptness, reliability, availability) and ergonomic characteristics (e.g. characteristics related to psychology of persons). A quality characteristic is an inherit characteristic of a product, a process or a system related to a requirement. Some standards proposed set of characteristics as a framework for specifying quality requirements and evaluating criteria (see § 2.4).

2.2.2 Software Quality Assurance

Three distinct concepts are important when defining software quality assurance (Figure 2.2): quality assurance (ISO 9000:2000), software quality assurance (SQA) (IEEE, 1990) and quality control (IEEE, 1990).

Galin (2003) extends IEEE definitions defending that SQA should not be limited to the development process, but should be extended to cover the long years subsequent to product delivery, that is, its maintenance. He considers also that SQA actions should not be limited to the technical aspects of the functional requirements, but should include also activities that deal with scheduling and the budgeting since undesirable results can be expected with project that are in budgetary and time constraints.

Quality assurance should be contrasted with quality control. According to Galin (2003) quality control activities take place when the product is completed and quality assurance aims to prevent the causes of errors, and detect and correct them early in the development process.

In other words, quality assurance is a set of activities for ensuring quality in the processes by which products are developed. Quality control is a set of activities for ensuring quality of products. The activities focus on identifying defects in the all generated products. Quality control activities are only a part of the total range of quality assurance activities (Galin, 2003).

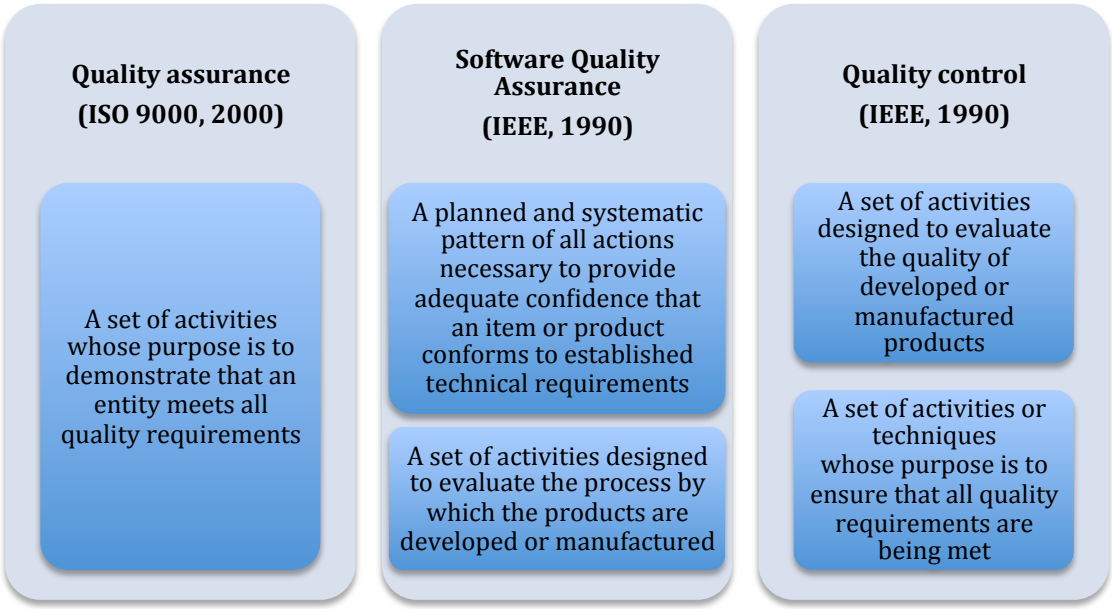


Figure 2.2. Definitions related to software quality assurance

2.2.3 Measurement

Figure 2.3 presents two classical definitions of measurement. In the definition 1, Fenton and Pfleeger (1997) sets that an entity is an object (such as a person, a model or a room) or an event (such as a journey or the testing phase of a software project). ISO/IEC 15939 (2007) summarizes that an entity is an object (a process, product, project or resource) that is characterized by measuring its attributes. An attribute is a property of an entity (such as, the color of a room, the cost of a journey or the elapsed time of the test phase). The attributes are often defined using numbers and symbols (such as a number of euros for the cost, or the different labels for colors). Thus, we measure attributes of things by using specific measurement methods.

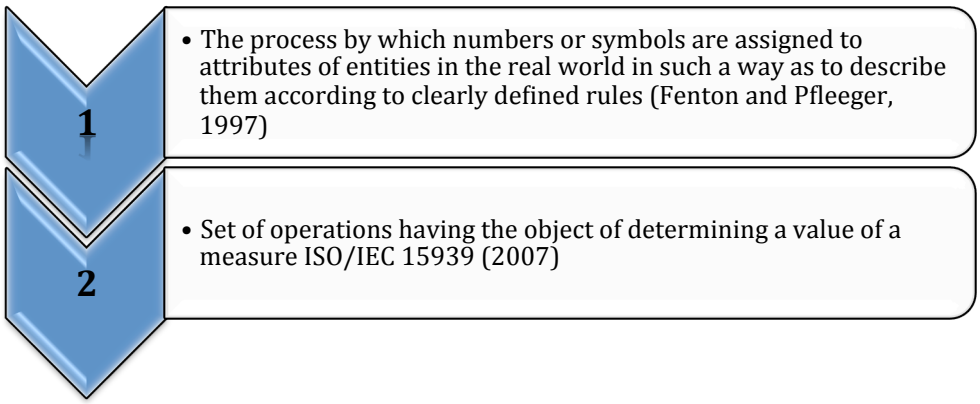


Figure 2.3. Measurement definitions

ISO/IEC 15939 (2007) organizes all these concepts associated to the measurement process in an information model (Figure 2.4).

In Figure 2.4, a measurement process is driven by information needs, which are "insights necessary to manage objectives, goals risks and problems", for instance: evaluate effectiveness of the software development, determine if the personnel resources are adequate to meet project commitments, or evaluate the reliability of the product. To address the information needs, it is defined *indicators*, a measure defined using derived and base measures and that is the basis for analysis and decision-making. A *model* supports the analysis of the evaluation combining one or more base and/or derived measures with associated decision criteria (thresholds or targets used to determine the need for action or further investigation, or to describe the level of confidence in a given result). The analysis model is based on an understanding of, or assumptions about, the expected relationship between the component measures and/or their behavior over time. *Derived measures* are measures defined as *function* of two or more values of base measures. *Base measures* are independent measures defined in terms of an attribute (of an entity) and the *method* for quantifying it. These methods in turn can be of two types: subjective, when the quantification of an attribute involves human judgment; or, objective, when the quantification is based on numerical rules such as counting, performed manually or with automated tools. Finally, one of the following *scales* is associated to the measurement (ISO/IEC 9126, 2001):

- Nominal - the measurement values are categorical. For example, the classification of defects by their type does not imply order among the categories;
- Ordinal - the measurement values are rankings. For example, the assignment of defects to a severity level is a ranking;
- Interval - the measurement values have equal distances corresponding to equal quantities of the attribute. For example, cyclomatic complexity has the minimum value of one, but each increment represents an additional path. The value of zero is not possible;
- Ratio - the measurement values have equal distances corresponding to equal quantities of the attribute where the value of zero corresponds to none of the attribute. For example, the size in terms of the number of requirements is a ratio scale because the value of zero corresponds to no requirements and each additional requirement defined represents an equal incremental quantity.

Measures using nominal or ordinal scales produce qualitative data, and measures using interval and ratio scales produce quantitative data (ISO/IEC 25000, 2005).

Figure 2.5 shows an example of the measurement information model. In this example, the information need is to estimate productivity of future project. To that end, an indicator with the average productivity for each project is proposed. The productivity is computed based on the lines of code per expended hour of effort. The analysis model is done based on the deviation of productivity values.

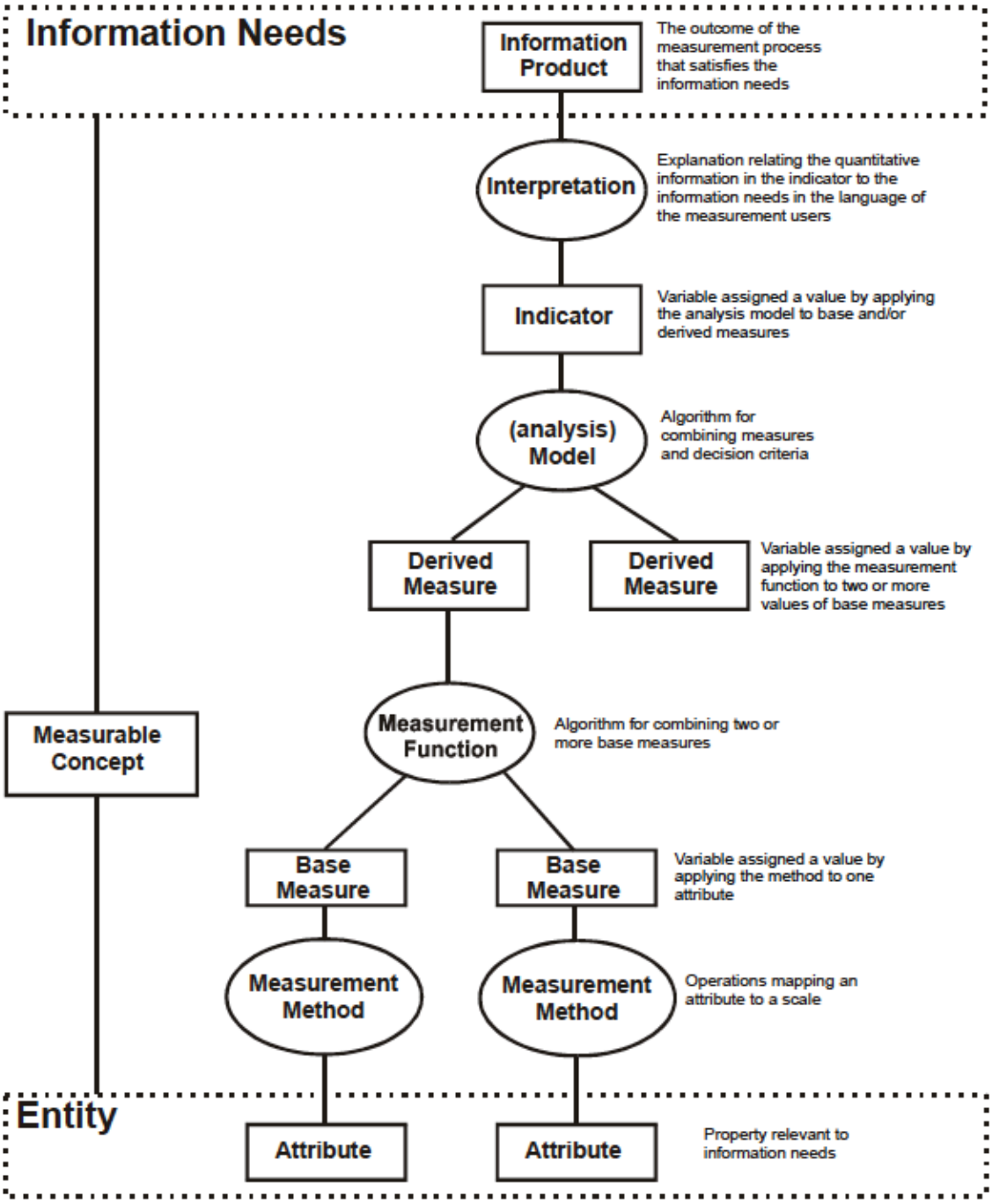


Figure 2.4. Key relationships in the measurement information model (ISO/IEC 15939, 2007)

Information Need	Estimate productivity of future project
Measurable concept	Project productivity
Relevant Entities	1. Code produced by past projects 2. Effort expended by past projects
Attributes	1. C++ language statements (in code) 2. Timecard entries (recording effort)
Base Measures	1. Project X Lines of Code 2. Project X Hours of Effort
Measurement Method	1. Count semicolons in Project X code 2. Add timecard entries together for Project X
Type of Measurement Method	1. Objective 2. Objective
Scale	1. Integers from zero to infinity 2. Real numbers from zero to infinity
Type of Scale	1. Ratio 2. Ratio
Unit of Measurement	1. Line 2. Hour
Derived Measure	Project X Productivity
Measurement Function	Divide Project X Lines of Code by Project X Hours of Effort
Indicator	Average productivity
(Analysis) Model	Compute mean and standard deviation of all project productivity values
Decision Criteria	Computed confidence limits based on the standard deviation indicate the likelihood that an actual result close to the average productivity will be achieved. Very wide confidence limits suggest a potentially large departure and the need for contingency planning to deal with this outcome.

Figure 2.5. Example of information model concepts for productivity (ISO/IEC 15939, 2007)

2.3 Approaches for Measurement Definition and Collection

During the last years, several approaches have been proposed for the definition of measures (Figure 2.6), for instance: Balanced Scorecard (Kaplan and Norton, 1992), Goal-Question-Metric (GQM) (Basili *et al.*, 1994a); the Goal-Question-Indicator-Metric (Park *et al.*, 1996), Software Quality Measure (DOT, 1991), Practical Software Measurement (McGarry *et al.*, 2002) and ISO/IEC 15939 (2007). GQM is one of the most known and largely used in literature (see, for example, several case studies described in (Solingen *et al.*, 1999) and the one I have worked since 2000.

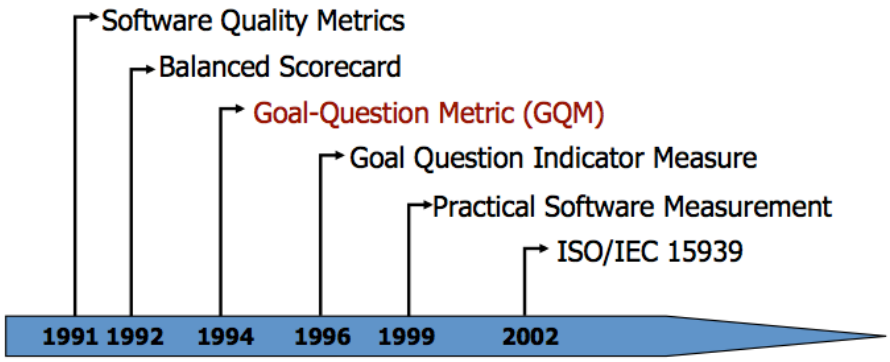


Figure 2.6. Approaches for measurement definition

The GQM helps identifying the metrics[7] (measures) required and the reasons why the data are being collected. The main idea of GQM is that measurement should be goal-oriented. It is a top-down methodology (Solingen *et al.*, 1999) (Figure 2.7) starting with the definition of an explicit measurement goal that is refined in several questions that break down the goal into its major components. Then, each question is refined into metrics that, when measured, will provide information to answer the questions. By answering the questions, we will be able to analyze if the goal has being attained. A metric can be used to answer several questions.

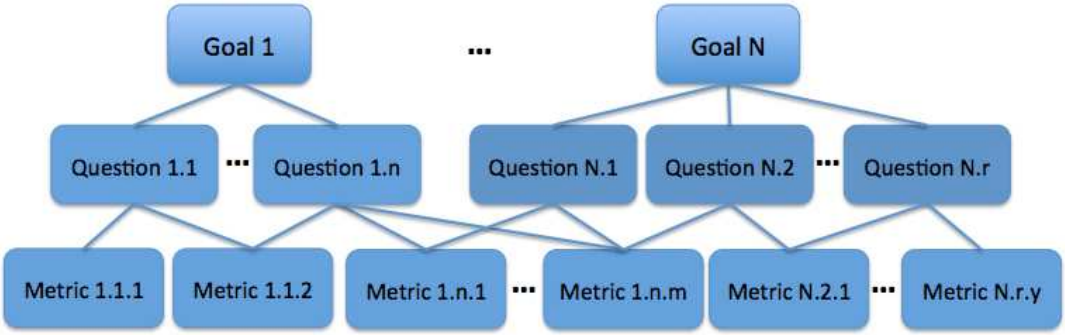


Figure 2.7. Hierarchical structure of the GQM approach

To collect the measures defined in a project, automated tools can be developed or subjective evaluation methods can be applied. Evaluation methods, defined to find defects in documents or errors in software programs, are usually successfully used to collect the data for the measures. Some of the evaluation methods that are commonly used are:

- Software inspection (Fagan, 1986) - it defines a process as a certain activity with a pre-specified entry and exit criteria. Verification checklists are usually used to support finding defects. Inspections can be used to validate if the output of the process complies with the exit criteria specified for the process;
- Phased-inspection (Knight and Myers, 1993) – it examines the product (requirements specifications, design, etc.) in a series of small inspections named phases. Each phase is designed to ascertain whether the product possesses some desirable properties (or criteria). It allows individual and group inspections in the same product considering different criteria, being performed in a rigorous and systematic way;
- Reading-techniques (Travassos *et al.*, 1999) – it provides a set of guidelines that are used to examine (by "reading") a given software artifact and identify defects. One can write guidelines looking for the specific information necessary for the defined measures;
- Software testing (Myers, 2004) - it is the process of executing a program or software application with the intent of finding errors. Software testing methods are traditionally divided into white-box testing (tests internal structures of a program) and black box testing (examines functionality without any knowledge of internal implementation).

[7] The term "metric" is used in GQM with the same meaning of the definition of measurement by ISO/IEC 15939 (2007). In fact, actually, this term is not used anymore, being replaced by the terms measurement and measure. However this term is still vey popular and well known.

2.4 Standards and Models for Software Quality

Many models and standards for quality assurance can be found in the literature. Some of them are largely used in industry; others work as essential references for the practice of software quality. In general, those standards and models focus either on the quality of the software products or software processes (Figure 2.8).

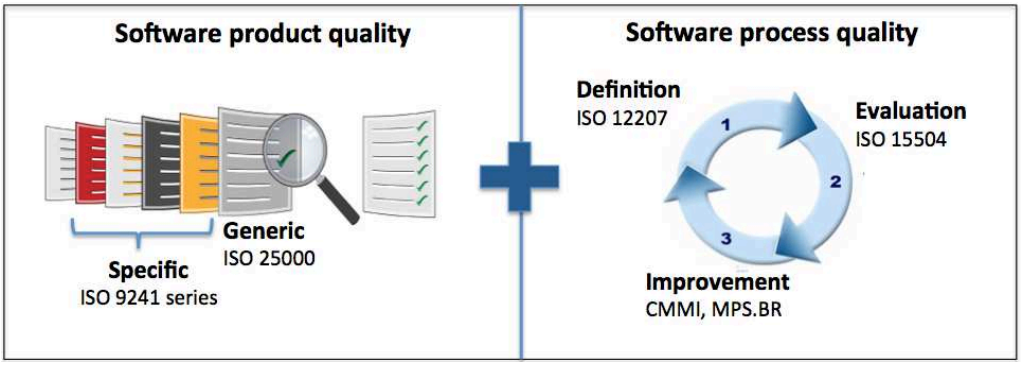


Figure 2.8. Standards and models for software quality

For software products, the proposal specifies guidelines, criteria and measures to evaluate a generic or a specific software product. For the first case, the actual standard is the SQuaRE (Software product Quality Requirements and Evaluation) (ISO/IEC 25000, 2005)[8]. It defines a set of quality characteristics (which are further subdivided into sub-characteristics) that are measured internally, externally or when a product is used in a specific context. Figure 2.9 shows some examples of measures defined for the effectiveness characteristic (ISO/IEC 25022, 2012).

Effectiveness		The accuracy and completeness with which users achieved specified goals Note: It does not take account of how the goals were achieved, only the extent with they were achieved
Measures	Task completion	X=A/B A=number of tasks completed B=total number of tasks attempted
	Task effectiveness	$\{X=1-\Sigma A_i \mid X > 0\}$ A_i=proportional value of each missing or incorrect component in the task output (maximum value=1)
	Error frequency	X=A/B A=number of errors made by the user B=number of tasks

Figure 2.9. Examples of measures for effectiveness

An important example of specific software product quality standards is the 9241 series, a multi-part standard covering ergonomics of human-computer interaction. They provide guidelines for different types of interaction (e.g. textual, tactile, organic, haptic) and computer systems, for instance: software accessibility (ISO 9241-171, 2008), World Wide Web user interfaces (ISO/IEC 9241-151, 2008), and tactile and haptic interaction (ISO/IEC 9241-920,

[8] ISO/IEC 25000 is a new standard that integrates the predecessors ISO/IEC 9126 (2001) and ISO/IEC 14598 (1999).

2009). Figure 2.10 shows some examples of guidance on navigation for World Wide Web user interfaces (ISO/IEC 9241-151, 2008).

General guidance on navigation	
Making navigation self-descriptive	Navigation should be designed to help users understand where they are, where they have been and where they can go next.
Showing users where they are	Each presentation segment (page or window) should provide the user with a clear and sufficient indication of where he or she is in the navigation structure and of the current segment position with respect to the overall structure.
Supporting different navigation behaviors	Users can exhibit different navigation behaviors depending upon their goals. They might know what they are searching for or might simply follow links that appear to lead to useful or interesting information. When designing navigation structures, the different user goals and navigation strategies should be considered by analyzing the different behaviors that users are likely to exhibit and by allocating priorities to them.
Offering alternative access paths	Alternative access paths for navigating to a specific unit of content should be offered to support different navigation strategies.
Minimizing navigation effort	The number of navigation steps needed to reach a certain piece of content should be minimized as long as different mental models, navigation strategies and tasks of the user are taken into account.

Figure 2.10. Example of guidelines for World Wide Web user interfaces

For software process, the proposals establish guidelines to support software process definition (ISO/IEC 12207,2008), evaluation (ISO/IEC 15504-1, 2004) and improvement (e.g. CMMI (Capability Maturity Model Integration) (SEI, 2010) and Brazilian Software Process Improvement Reference Model (SOFTEX, 2012). These three tasks should be continually executed to ensure the effectiveness of software process in the enterprises.

ISO/IEC 12207 (2008) provides a defined set of processes to facilitate communication among acquirers, suppliers and other stakeholders in the life cycle of a software product. It contains processes (its purpose and outcomes), activities, and tasks to be applied during the acquisition of a software product or service and during the supply, development, operation, maintenance and disposal of software products. Figure 2.11 presents, as an example, the activities and tasks described for the stakeholder requirements process in ISO/IEC 12207.

ISO/IEC 15504 (ISO/IEC 15504-1, 2004) provides a framework to be applicable across all organizations, and for conducting assessments using a variety of methods, techniques and tools. This framework:

- facilitates self-assessment;
- provides a basis for use in process improvement and capability determination[9];
- takes into account the context in which the assessed process is implemented;
- produces a process rating;
- addresses the ability of the process to achieve its purpose;
- is appropriate across all application domains and sizes of organization;
- may provide an objective benchmark between organizations.

[9] ISO/IEC 15504 is used, for example, for the Brazilian software process improvement standard (SOFTEX, 2012)

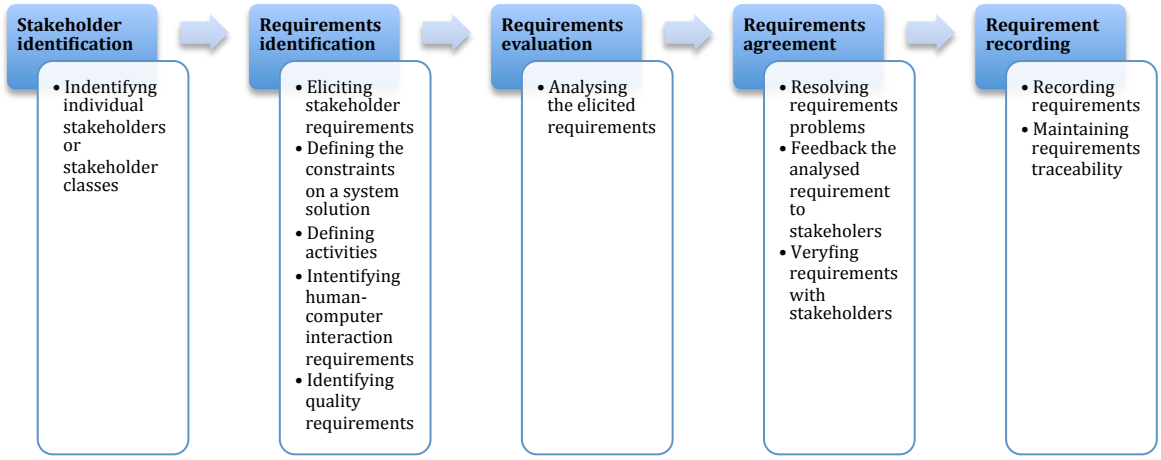


Figure 2.11. Stakeholder requirements Process (Activities and Tasks)

Capability Maturity Model Integration (CMMI) (SEI, 2010) is a software process improvement (SPI) approach that provides organizations with the essential elements of effective processes. The core element of CMMI is the process areas. A process area is a cluster of related practices in an area that, when implemented collectively, satisfies a set of goals considered as important for making significant improvement in that area. CMMI for Development (CMMI-DEV) version 1.3 consists of 22 process areas. The CMMI-DEV comes with two different representations - staged and continuous. The *staged representation* (Figure 2.12) offers a systematic, structured way to software process improvement one step at time. It groups process areas into five maturity levels used for an appraisal. Achieving each stage (i.e., maturity level) ensures that an adequate software process improvement has been laid as a foundation for the next maturity level and allows for lasting, incremental improvement. The *continuous representation* offers a flexible approach to process improvement by allowing the organization to improve the performance of a single process area (e.g. Requirements Management). It can work on several areas that are closely aligned to the organization business objectives.

2.5 The Role of Knowledge Management

Knowledge management came out as a reaction to the recognition that employees in an organization gather, as part of their daily activities, knowledge that is valuable to the organization. The typical image, cited by Basili *et al.* (2001), is that knowledge has legs and walks home every day. Knowledge management concept emerged in mid-1980s and rapidly increased since 1990s. In software engineering, it works toward software process improvement (Rus and Lindivall, 2002) by explicitly and systematically addressing the management of organizational knowledge tying together daily production activities, improvement activities, and business goals, thereby supporting the establishment of a learning organization. Rus and Lindvall (2002) affirm that other software process improvement approaches, such as CMMI (see previous section), might suggest that knowledge should be managed, but do not explicitly state what knowledge needs to be managed and how, when, where or by and for whom.

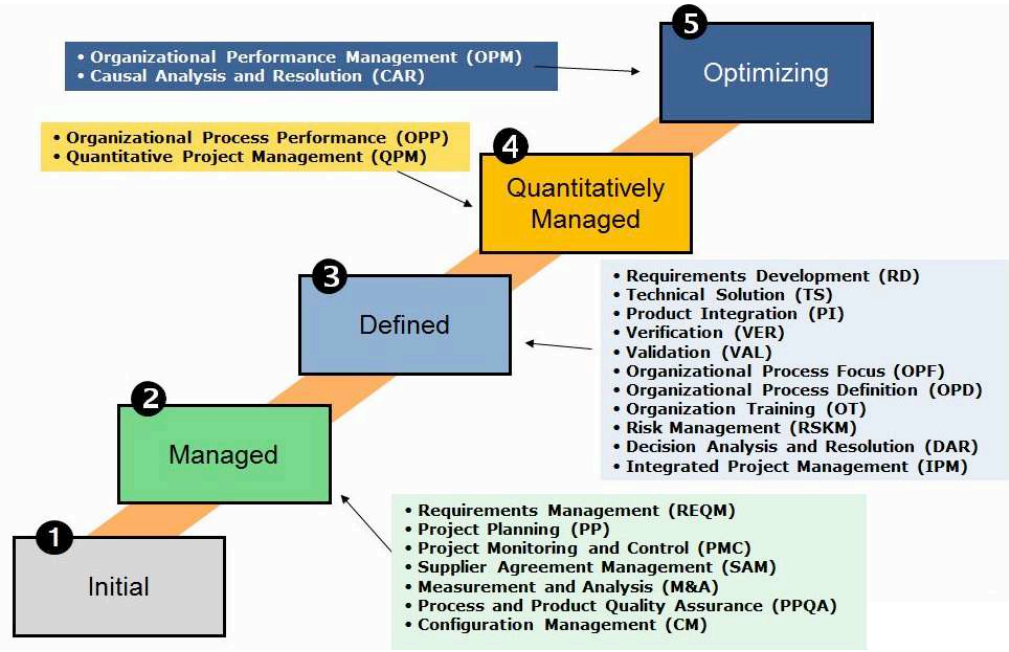


Figure 2.12. CMMI Maturity levels x Process Areas

Several purposes, approaches and empirical studies have been published in the last years (Tiwana, 2000; Dingsøyr and Conradi, 2002; Aurum *et al.*, 2003; Bjørnson and Dingsøyr, 2008). In general, these publications tried to particularize the traditional methodologies, techniques and approaches of knowledge management for software engineering domain with the goal of obtaining better quality and productivity. Consequently, practices of knowledge management for software engineering start to be integrated in enterprises. In a large systematic literature review on the topic, for example, seven hundred and sixty-two articles were identified, of which 68 were studies in an industry context (Bjørnson and Dingsøyr 2008).

In the following sections we start with a general view of knowledge and knowledge management, briefly quoting what exists in terms of knowledge management for software engineering towards software quality. Thus, we present experience factory, a knowledge management approach specific for this domain; and some techniques for organization, capture and dissemination of knowledge.

2.5.1 Knowledge and Knowledge Management: Concepts and Application

The definition of knowledge is normally built bottom-up from data, to information and then knowledge (Figure 2.13) (see, for example, (Rus and Lindival, 2002, Probst *et al.*, 2000)). Data are raw facts, for example: 1.36. Information is data in context; it is organized to make the data useful for end users who perform tasks and make decisions; for instance, saying that 1.36 is the exchange rate between the US dollar and the euro currency. Knowledge requires understanding of the information. It is a net of information based on one's particular experience, for example one's knowledge of the currency exchange mechanisms. "Knowledge is a fluid mix of framed experience, values, contextual information, expert insight and grounded intuition […]. It originates and is applied in the minds of knowers" (Davenport and Prusak, 1998).

Two types of knowledge are commonly defined: explicit and tacit. Explicit knowledge is knowledge that can be captured and organized in a form that allows its distribution (for example, reports, a book). Tacit knowledge means knowledge that a human is not able to

express explicitly, but is guiding the behavior of the human (Lindval at al., 2001). Tacit knowledge is particular to each individual and difficult to share, as one is usually not even aware of all one knows (**Anquetil *et al.*, 2006**). Moreover, Grundstein (2012) adds two aspects:

- Knowledge is not an object - it is in the interaction between an interpretative framework (in the head in an individual or integrated in an artifact) and data. Indeed, the knowledge arising from the interpretation by an individual of information, meaning that takes information may differ from one individual to another (Arduin *et al.*, 2012);
- Knowledge is linked to action - knowledge is created by the action of individuals, taking into account the context and the situation (which includes other people and artifacts).

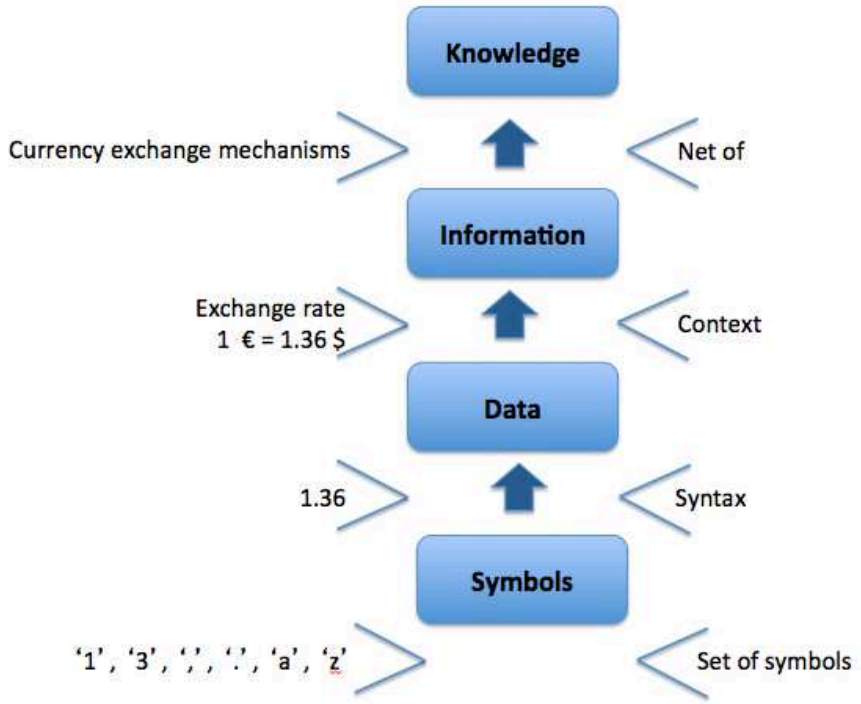


Figure 2.13. Data, Information and Knowledge (adapted from Probst *et al.* (2000))

Knowledge management is usually defined in terms of the activities that comprise it, as shown in Figure 2.14. Different models describe knowledge management activities. Some examples are: the proposals of Nonaka and Takeuchi (1995), Probst *et al.* (2000), Dixon (2002) and Grundstein (2012).

The model proposed by Nonaka and Takeuchi (1995) is one of the most referenced for knowledge management. It demonstrates the ability of an organization to create new knowledge continuously by successive conversions between tacit and explicit knowledge. It is called SECI (socialization, externalization, combination and internalization) model that is defined as follows (Figure 2.15):

- Socialization - is the process of sharing knowledge doing things, knowledge is not made explicit, but rather a knower shows to one who does not know, how to do things. It is the conversion of tacit knowledge into another tacit knowledge;

- Externalization - is the process of explicating what one knows. Through externalization, a knower may express (e.g. writing a manual) what he knows and this knowledge may then be circulated among a large group or across time;
- Combination - is the process of combining various sources of explicit knowledge to create a new one, as one would do in a literature survey;
- Internalization - is the process by which one makes some explicit knowledge one's own, by integrating it to one's own net of information.

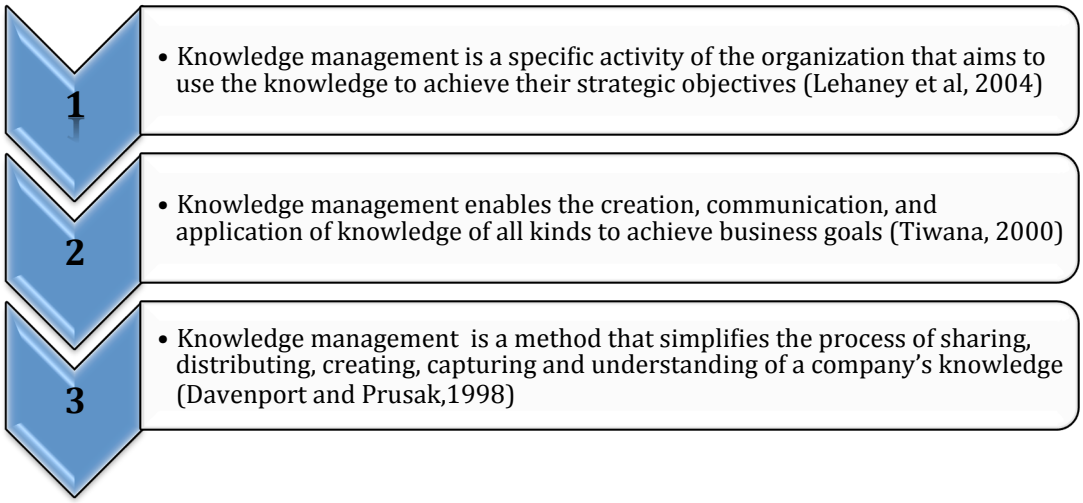


Figure 2.14. Knowledge management definitions

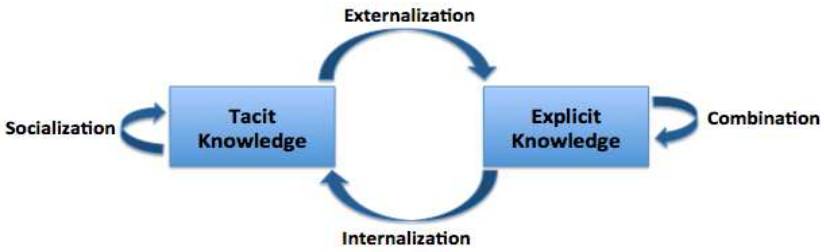


Figure 2.15. Knowledge sharing according to Nonaka and Takeuchi (1995)

The model proposed by Probst *et al.* (2000, p. 2-3) consists of core and additional processes. Core processes consider (i) the identification of knowledge, (ii) external knowledge acquisition, (iii) the development of knowledge, (iv) the sharing and dissemination of knowledge, (v) the use of knowledge, and (vi) the retention of knowledge that builds organizational memory. Additional processes use these six processes in the strategic goals of the organization, by assessing the knowledge to measure and monitor its implementation and guide the knowledge management goals.

Dixon (2002) proposes a process where the knowledge is captured and transferred in continual cycle (Figure 2.16). First, a team performs a task, reaching a given result (satisfactory or not). From this, the team must pause and reflect on what happened and its actions contribute to the result obtained. After this activity, the team has captured and expressed new knowledge (how the action "created" the result) and is in a better position to perform a new, similar, task. That means the team translate its experience into knowledge, by

gathering existing knowledge in the minds of stakeholders in a common understanding, what Dixon named common knowledge. But the knowledge also needs to be transferred to other members of the organization. For this, a knowledge transfer system must be selected, the knowledge must be expressed in a form that will allow its transfer (e.g. in a list of "lessons learned") and, finally, it must be transferred to another team that will integrate it and will be able to use it to perform its tasks. A knowledge transfer[10] system suitable for the type of knowledge is, thus, selected. Finally, knowledge is translated into a usable form and the team that receives the knowledge adapts it to its problem.

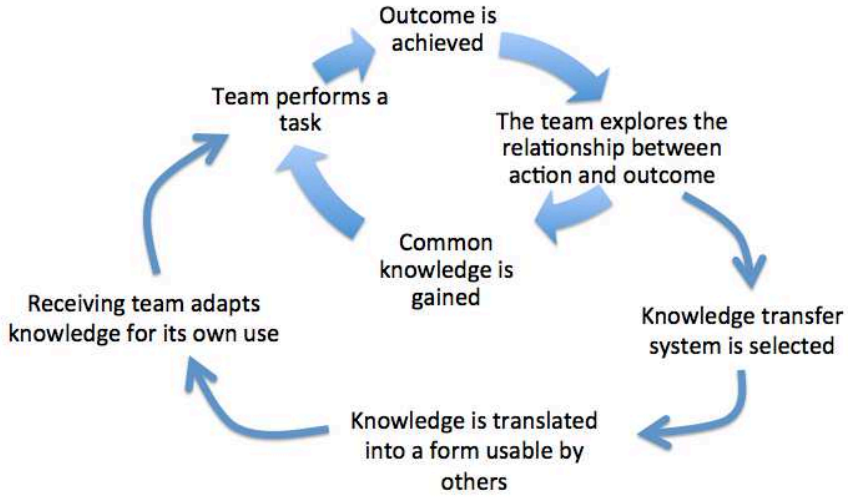


Figure 2.16. Creation and transfer of common knowledge (Dixon, 2002)

Finally, Grundstein (2012) defines a set of what he calls generic knowledge management processes; they are: (i) locating knowledge management process, that deals with the location of explicit and tacit knowledge; (ii) preserving process deals with the retention of knowledge and skills; (iii) enhancing process that deals with the benefit of knowledge and skills; and, (iv) actualizing process that deals with the actualization of knowledge and skills.

2.5.1.1 Experience Factory

Experience factory (Basilli *et al.*, 1994b, 2001) is a technical and social knowledge management infrastructure largely used in software engineering that aims to reuse life cycle experience, processes and products. Experience is collected from software development projects, and are packaged and stored in an experience base. This infrastructure is supported by two major concepts: a concept of evolution - the Quality Improvement Paradigm (QIP), and a concept of measurement and control - the Goal-Question-Metric (presented in 2.3).

QIP emphasizes the continuous improvement by learning from experience, both at the level of projects and at the level of organizations. It consists of the following steps:

[10] Dixon developed five categories of knowledge transfer: (i) *serial transfer* applies to a team that does a task and than the same team repeats the task in a new context; (ii) *near transfer* involves transferring knowledge from a source team to a receiving team that is doing a similar task in a similar context but in a different location; (iii) *far transfer* involves transferring knowledge from a source team to a receiving team when knowledge is about a non routine task; (iv) *strategic transfer* involves transferring very complex knowledge from one team to another in cases where the teams may be separated by both time and space; and (v) *expert transfer* involves transferring explicit knowledge about a task that may be done infrequently.

- Characterize - understand the environment based on available models, data, etc. This characterization requires that we classify the current project with respect to a variety of characteristics to isolate a class of projects with characteristics similar to the project being developed;
- Set goals - on the basis of the initial characterization, we set quantifiable goals for success and improvement (use of GQM);
- Choose Process - on the basis of the characterization and of the goals, we choose the appropriate process to improve;
- Execute – we execute the process constructing the products and providing project feedback based upon the data on goal achievement that are being collected;
- Analyze - at the end of each specific project, analyze the data and the information gathered to evaluate the current practices, identify problems, record findings, and make recommendations for future projects improvement;
- Package - consolidate the experience gained in the form of new, or updated and refined, models and other forms of structured knowledge gained from the projects, and store it in an experience base so it is available for future projects.

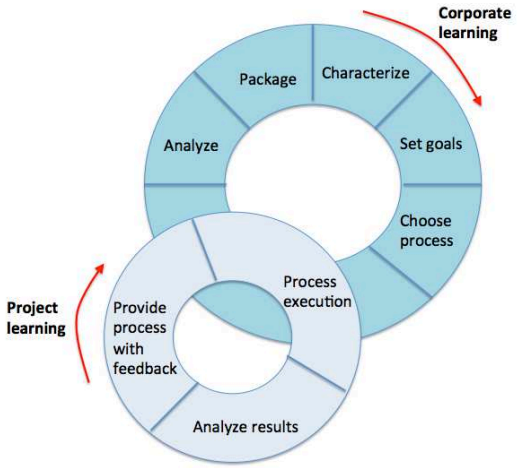


Figure 2.17. Quality Improvement Paradigm

Some examples of experience packages are (Dingsøyr and Conradi, 2002):

- Product packages - information about the life cycle of a product, information on how to reuse it and lessons learned from reuse;
- Process packages - information on how to execute a life cycle process, and how to reuse it;
- Relationship packages - used for analysis and forecasts (e.g. cost and defect models).
- Tool packages - instructions for use of a tool and experience with it.

2.5.2 Knowledge Capture and Dissemination

A well-known knowledge management technique for knowledge capture is the Postmortem Analysis (PMA). Another technique that comes from social science is Learning History, which is used not only to capture the knowledge but also to disseminate it. Both techniques are briefly presented in this section.

2.5.2.1 Post-Morten Analysis

PMA, also called project review or project retrospective, simply consists in "[gathering] all participants from a project that is ongoing or just finished and ask them to identify which aspects of the project worked well and should be repeated, which worked badly and should be avoided, and what was merely 'OK' but leaves room for improvement" (Stålhane *et al.*, 2001). In Nonaka's framework for knowledge management (see section 2.5.1), PMA is a tool to externalize knowledge.

PMA is mainly viewed as a software process improvement tool. According to Stålhane *et al.* (2003) it is an obvious way to improve a software development process. Other authors (Birk *et al.*, 2002; Dingsøyr *et al.*, 2001; Rising *et al.*, 1999; Yourdon, 2001) assume the same point of view, either explicitly or implicitly.

The term post-mortem implies that the analysis is done after the end of a project, although, as recognized by Stålhane in the preceding quote (Stålhane *et al.*, 2003), it may also be performed during a project, after a significant mark has been reached.

A PMA may be more or less structured, and focused or "catch all". One of the great advantages of the technique is its flexibility. It may be applied on a small scale with little resources (e.g. a 2 h meeting with all the members of a small project team, plus one hour from the project manager to formalize the results). Depending on the number of participants in the PMA, it may use different levels of structuring, from a relatively informal meeting where people simply gather and discuss the project, to a more formal process.

In general, PMA process comprises three phases (Birk *et al.*, 2002) (Figure 2.18):

- Preparation - seeks to identify the history of the project to better understand what has happened. The available project documents are reviewed and the goal for APM is set;
- Data collection – the stakeholders have a group discussion for collecting experiences. Both negative aspects, such as activities to be avoided in the future, and project's successful aspects, such as recommended practices should be collected;
- Analysis – where the positive and negative aspects identified previously are discussed. It is analyzed the causes of success or failure and defined what can be done to improve that wherein they are deficient, or what should be done to repeat in future projects what happened in a positive way. Data collection and analysis can be performed in the same session.

Figure 2.18. Post-mortem analysis phases

It is important to remember that post-mortem analysis is not sufficient to form a complete politic of knowledge management (such as Experience Factory). It is mainly a useful tool to elicit the knowledge, i.e. to discover the relevant pieces of experience obtained from a project. Other important steps as packaging of the knowledge, or disseminating it, are not considered by the PMA.

2.5.2.2 Learning History

Learning History is a technique that aims at capturing not only the results of a project of interest, but the mental processes, feelings or opinions of the project's members (Senge *et al.*, 1999, p. 531). It is a technique for capturing and disseminating knowledge from the various perspectives of the participants of the project. A learning history is the story of the events as told by the people who participated in it. It should present the points of view of all the participants including contradictions and conflicting views. Its goal is not to reach a final verdict or lesson from the event but to present the facts as they were perceived by the participants, allowing the reader to build his-her own understanding and learn from it (Kleiner and Roth, 1995).

Basically a learning history is a two-column document where the right column presents the story as told by the participants and the left column presents comments and analyses made by the historian (the person who writes the history). The story is organized in sections (or chapters) presenting the various notable results of the project covered.

Kleiner and Roth (1995) propose a process for the construction and use of learning history (Figure 2.19) composed of the following phases:

- Planning, where the scope of the project is defined and its "notable results" —that justify the need for knowledge transfer— identified;
- Reflective research, where the stories are collected interviewing the relevant persons;
- Distillation, where the relevant topics are selected by the historian to be, later, refined in a form suitable for reading;
- Writing, where the document is created;
- Validation, where the document is validated by the people who told the stories;
- Dissemination, where a receiving team reads the story, discusses it and tries to relate it to its experience to build its own knowledge. Finally, there is a possible publication of the story for a larger public.

Figure 2.19. Learning history phases

Some interesting characteristics of learning history are (Kleiner and Roth, 1995; Bradbury and Mainemelis, 2001):

- It includes much more than the mere facts of the project, by telling their stories, people include their opinions, sentiments, belief in the tale;
- Contradictory opinions may (and should) appear, presenting all the points of view and giving room for the interpretation of the events and results in their broader context;
- The "knowledge" embodied in the story is not explicited (e.g. as in best practices or lessons learned), but the participants in the dissemination must construct it from the story and from their own experience and discussion;
- Learning histories are mainly presented for large projects resulting in entire books.

2.5.3 Knowledge Organization with Ontologies

One important aspect of knowledge management is to preserve the knowledge externalized during the process. Even that when organized it becomes an object, it will be useful for future projects to allow learning, new interpretation and generation of new tacit knowledge. The knowledge captured can be organized on different ways: as lessons learned, diagrams, packages of experiences with their context (see sections 2.5.1.1 and 2.5.2.1), histories (see section 2.5.2.2), and also on ontologies. Ontologies have been largely used to represent knowledge thanks to its capacity of representing concepts in an unambiguous way, using formal languages that enable reasoning.

Several definitions of ontology can be found in the literature. Two of them are presented in Figure 2.20. The first definition (Gruber, 1995) sets that it should be an explicit specification, that means to be formal in a way that hamper consensus, enables reasoning and avoid misunderstanding. The second one details the components of an ontology. Basically, an ontology consists of concepts and relations, their definitions, properties and constraints expressed by axioms (Chandrasekaran *et al.*, 1999). Figure 2.21 presents an illustrative example of ontology to represent the concepts, relation, property and axioms for a family.

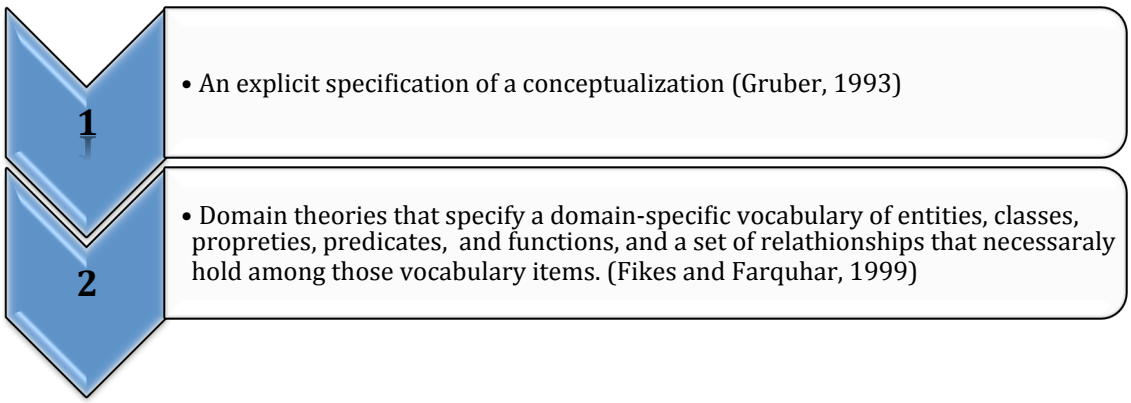


Figure 2.20. Ontology definitions

According to Guarino (1998) there are different kinds of ontologies:

- Top-level ontologies - that describe very general concepts like space, time, matter, event, etc.;
- Domain ontologies - that describe the vocabulary related to a generic domain (such as medicine, or transportation);
- Task ontologies - that describe generic tasks or activities (like diagnosis or selling);
- Application ontologies - that describe concepts depending on a particular domain and task. Application ontologies are specializations of both the domain and task ontologies.

Ontologies may serve to various purposes in the context of knowledge management (**Anquetil *et al.*, 2007**):

- Reference on a domain - explicit knowledge serves as a reference to which people, looking for detailed information on the domain modeled, may use;
- Classification framework - the concepts explicited in an ontology are a good way to categorize information on the domain modeled. Indication of synonyms in the

ontology helps avoiding duplicate classification. Other relations among the concepts of the ontology help one browsing it and finding an information one is looking for;

- Interlingua - tools and/or experts wishing to share information on the domain modeled, may use the ontology as a common base to resolve differing terminologies.

Ontologies have been largely used for knowledge management related to SQA activities, in particular, to software process improvement. For instance, ontologies were developed to organize the knowledge about software quality (e.g. Henderson-Sellers, 2014; Kayed *et al.*, 2009; Bertoa *et al.*, 2006; Barcellos *et al.*, 2010) and software process (e.g. Palomäki and Keto, 2006; Wongthongtham *et al.*, 2006); and as the main support in knowledge management systems (e.g. Al Balushi et al., 2013; Surrephong *et al.*, 2008). For my doctorate, I used domain ontologies to improve the traditional software process by including a domain specific activity **(Oliveira, 1999)**. This activity supports the software developers in the requirements elicitation, system modeling and database design with the use of domain and task ontologies (**Oliveira *et al.***, **2004**).

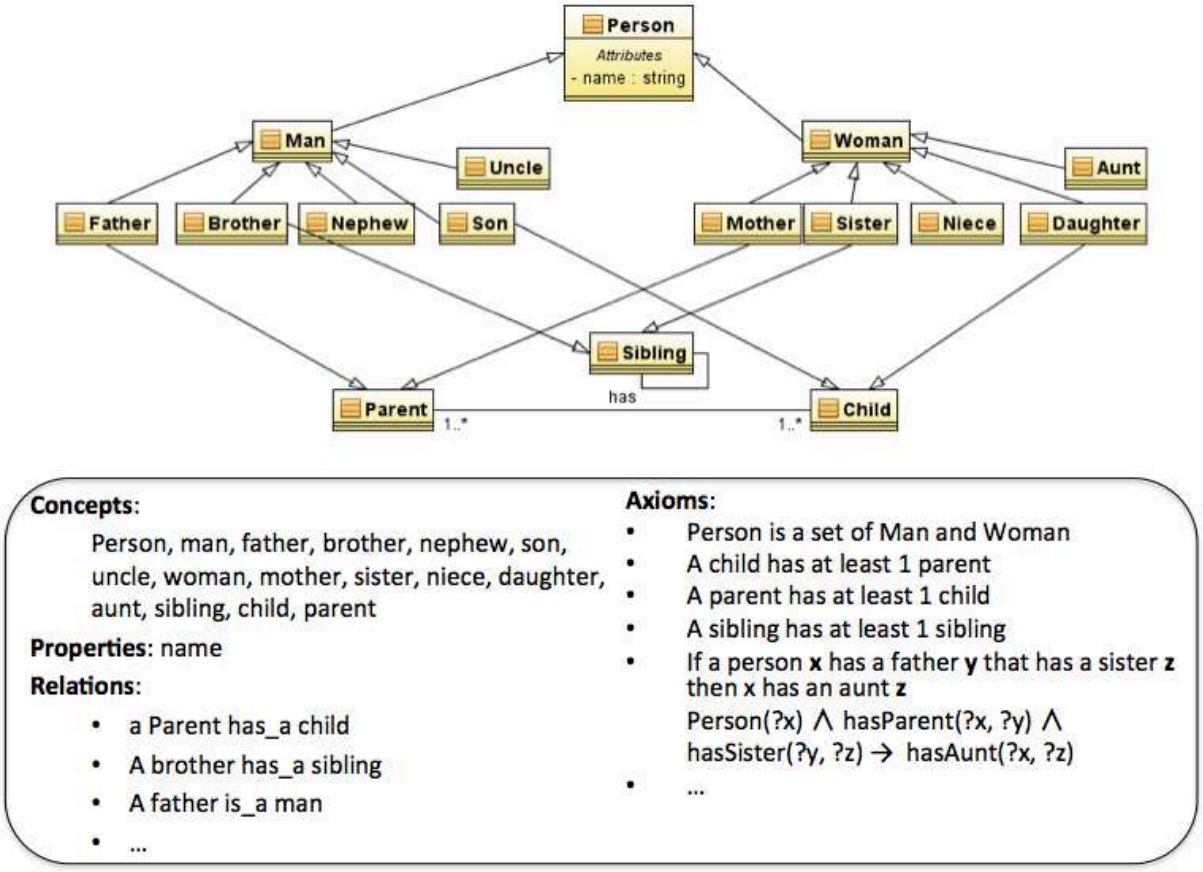


Figure 2.21. Illustrative example of an ontology[11]

[11] Extract of the ontology about family relationships available in the Protégé ontology library (http://protegewiki.stanford.edu/wiki/Protege_Ontology_Library). (Accessed April 10, 2014)

2.6 Conclusion: Research Motivation

This chapter presented an overview of the main concepts related to software quality assurance and knowledge management. In summary, the literature proposes a set of standards and methodologies (not limited to the ones presented in this chapter), as well as the need of measurements to ensure the quality of software process and products. Moreover, with the emergence of knowledge management concerns in the last years of the 20th century, the software engineering community started to apply them in order to improve quality and productivity in software production. Aware of these concepts and the challenges presented in Chapter 1 (§ 1.2), it is possible to identify two main questions that motivated my research:

- How to perform quality assurance for software processes and products (considering its variety of technologies and types) in a way that could be really applicable and useful in the practice of the industry?
- How to apply knowledge management issues in order to improve the quality of software processes and products?

Chapters 3 and 4 present, respectively, the research performed to answer each one of these questions.

Chapter 3
Methodologies and Measurements for Software Quality Assurance

3.1 Introduction

As presented in Chapter 1, we can divide software quality into two main concerns: quality of software process and quality of software products. Methods, techniques, measures and tools have been developed during the last years to support the activities necessary to address both concerns.

In general, the research about the quality of software process deals with issues related to the definition, evaluation and improvement of software process. Standards were defined for the definition (ISO/IEC 12207, 2008) and evaluation of software process (ISO/IEC 15504-2, 2003, ISO/IEC 15504-3, 2004). Maturity models (e.g. SEI, 2010) also present a structure for software process implantation and improvement in organizations. These maturity models establish the software engineering practices to be institutionalized in an organization. Section 3.2 presents the main investigations we performed on this topic that comprises the evaluation and improvement of software process.

The research of quality of software product is usually related to the evaluation of the different software products produced during a software process (e.g. requirements specification, design model, code, user interface, etc.). Guidelines, standards (see Chapter 1, § 2.4) and measures defined using appropriated methods (see Chapter 1, § 2.3) are usually used in these evaluations. Section 3.2.3 presents some of the main studies on quality evaluation of different software products by using software measures.

Section 3.3.5 concludes this chapter with a summarization of our contributions and a discussion on limitations of software quality evaluation identified by performing the research presented in this chapter.

3.2 Software Process Quality Assurance

As presented in Chapter 1 (§ 2.2.1), the research on software process quality includes its definition, evaluation and improvement.

The definition of software process was the main motivation for the research at the end of the last century (and beginning of my professional activity). With the proliferation of web application systems and the use of new technologies (e.g. mobiles), the state of practice in industry has revealed the need of definition of specific software process adequate for the development of each kind (e.g. object-oriented systems, web systems, critical systems) and size of software system to be developed in an organization. Moreover, the software process defined for each project should follow a standard software process to organize and discipline

the software development, determine the fundamental activities that shall be present in any defined process, and assure common understanding of all stakeholders.

Eman *et al.* (1998) defined a standard process as the basic process, guiding the establishment of a common process in the organization. A standard process defines a single structure to be followed by all the teams involved in a software project independently from the software characteristics to be developed. However, the effectiveness of defined processes depends on their adequacy to the characteristics of the organization, of the software system to be developed, and of the project. With this on mind, our first research proposed an approach for the definition a software process for a project based on standard software and maturity models (**Machado *et al*., 2000a**). This approach included: (i) definition of a standard process, (ii) specialization of a standard process considering the type of software and to the development paradigm to be adopted, and (iii) instantiation for projects considering their specific characteristics (size, team, life cycle model, etc.).

This approach was later generalized to be used for the definition and institutionalization of different software processes in an organization, for instance: for configuration management (**Duran *et al*., 2005**), requirement development (**Corrêa *et al*., 2004**) and software acquisition (**Sousa *et al*., 2005**).

With the dissemination and use of software processes in industry, the importance of their evaluation and, as consequence, their improvement has emerged. In this section, we present a piece of research done to tackle these aspects.

3.2.1 Software Process Evaluation

Once the software process is defined and institutionalized in an organization it is important to evaluate it to allow its improvement. To do so we defined a set of measures, integrated them into the software process and applied them to real cases in industry (**Gomes *et al*., 2000, 2001a, 2001b; Souza *et al*., 2003**).

For the definition of measures, we used the Goal-Question Metric (GQM) approach (see Chapter 1, § 2.3). To define the measurement goals, we conducted interviews with software project managers in industry to find out the main problems with the software projects. Several problems were cited, but three stood out as being the most relevant: the lack of accuracy of project estimates, the low quality observed in the products released for use, and the high cost involved in software development. Three objectives were then defined for the measurement program:

- to improve the accuracy of project estimates;
- to increase the quality of products released for use; and,
- to reduce the final cost of projects.

Once the objectives are defined, they must be used to guide the definition of questions to be answered in order to examine whether the targets were achieved properly. Figure 3.1 shows, as an example, the definition of measures using GQM for the first objective. Similarly, measures were defined for the other two objectives. At the end, we obtained a set of 21 measures (Table 3.1).

Moreover, we argue that the experience of the project team can also explain possible problems encountered during the project. Productivity, number of errors, time and spent effort are some of the information that may be influenced by the team's capacity. Therefore, we had included in the original set of measures the following information related to the experience of the team: experience in programming language, in the application domain, tools, method and development process; type of training in software engineering; total experience time.

Goal 1: To analyze project estimates With the purpose of improving With respect to accuracy From the viewpoint of project managers
Question 1: How accurate is the schedule estimates of project? Measure 1.1) Accuracy of schedule estimate of entire project = $\frac{\text{real time of the entire project}}{\text{estimated time for the project}}$ Measure 1.2) Accuracy of schedule estimate by software process activity = $\frac{\text{real time of an activity}}{\text{estimated time for the activity}}$
Question 2: How accurate is the effort estimates of project? Measure 2.1) Accuracy of effort estimate in the entire project = $\frac{\text{real effort of the entire project}}{\text{estimated effort for the project}}$ Measure 2.2) Accuracy effort estimate by software process activity = $\frac{\text{real effort of an activity}}{\text{estimated effort for the activity}}$

Figure 3.1. Example of measures for software process evaluation defined with GQM

Table 3.1. Measures proposed for software process evaluation

Category	Measures
Schedule	Total project time Time in the analysis, design, coding, unit tests performed by analysts, system and acceptance tests Time spent in review meetings Time in rework.
Accuracy of schedule estimates	Accuracy of schedule estimate of entire project Accuracy of schedule estimate to the analysis, design, coding, unit tests performed by analysts, system and acceptance tests.
Effort	Total effort in the entire project Effort in the analysis, design, coding, unit tests performed by analysts, system and acceptance tests. Effort in review meetings Effort in re-reviews (new review meetings carried out due to the non approval of the product at former meetings) Effort in reworks.
Accuracy of effort estimates	Accuracy of effort estimate in the entire project Accuracy of effort estimate to the analysis, design, coding, unit tests performed by analysts, system and acceptance tests.
Size	Number of lines of code.
Number of errors	Number of errors in the requirements specification and in the design of the system found in review meetings; errors in coding found in the unit tests performed by analysts.
Number of modifications	Number of modifications of the requirements specification, design and coding after their approval.
Defect density	Number of errors added to the modification number in relation of the size of the system.
Personnel turnover	Percentage of people, who have joined, left or changed functions during the project.
Productivity	Number of lines of code produced by effort unit.
Software deterioration	Relationship between the efforts spent to correct problems found after the release of the system for the user compared to the effort spent before release for the user.

For a better interpretation of the results, operational definitions of the measures should be elaborated by minimizing ambiguities and homogenizing the understanding of the usage of each measure. For example, all measures of time in Figure 3.1 involved the number of days from the date of commencement of work (of an activity or the entire project) to the date of its closure. For effort we set that it would be measured in man-hours; one man-hour being equal to one-hour time employed by a member of the project team. Thus, effort is the sum of the total number of hours used by all team members in the execution of an activity.

The process of measurement collection should be incorporated to the defined process itself, using documents already generated by the team. Some examples in this context are important milestones in the project follow-up documentation, the definition of the number of errors on inspection meetings and control of time spent on development activities. In defining collection methods as part of the development process, we have obtained the different measures results during the practice of software process activities without generating new workload for the team.

Those measures were applied in several enterprises in Brazil. One case of application was performed for a large project about cost planning and budget. The measures were collected and analyzed in a final meeting with the project managers. Figure 3.2 shows the report of the base measures (real time, estimated time, real effort and estimate effort for each activity of the software process, see Figure 3.1) collected for one of the system's module. For this analysis, it is important to be aware that the measures are inter-correlated; that means, one measure could impact the result of the other measure and, therefore, should be analyzed in group. From the analysis of all measures, it was concluded that the main problems of this project were: the establishment of project estimates without using any formal method or historical basis since it was made an *ad hoc* estimation based on the manager experience; a high staff turnover resulting in a decrease of team performance; and the lack of experience of the project team. It was also possible to identify the process activities with more problems to make corrective actions. **Gomes *et al.* (2000, 2001a)** present in more details this study.

We also defined and applied measures in a large enterprise with the goal of verify the adherence of the defined software maintenance process (i.e., the team was really following the planned activities) and the general quality of the artifacts (**Souza *et al*., 2003**). The measures were: density of task execution, percentage of time accuracy, accuracy of effort, frequency of errors detected by the customer, average of requirements errors, average of project design errors, average of source code errors, percentage of delivered requests, and customer satisfaction.

3.2.2 Software Process Improvement

Evaluate software process execution allows to identify problems (e.g. high estimates errors, low personnel experience, high turnover) and bottlenecks (e.g. activities with low accuracy of estimates, activities with high rework) in the software process making possible its correction for a next project. However, to achieve high-level maturity in software process improvement (SPI), organizations must be able to manage their projects quantitatively by (i) establishing and maintaining a quantitative understanding of the performance of their standard processes; and (ii) providing process performance data, baselines, and models to quantitatively manage the organization's projects (SEI, 2010). Defining all of these aspects is neither customary nor automatic for most software managers.

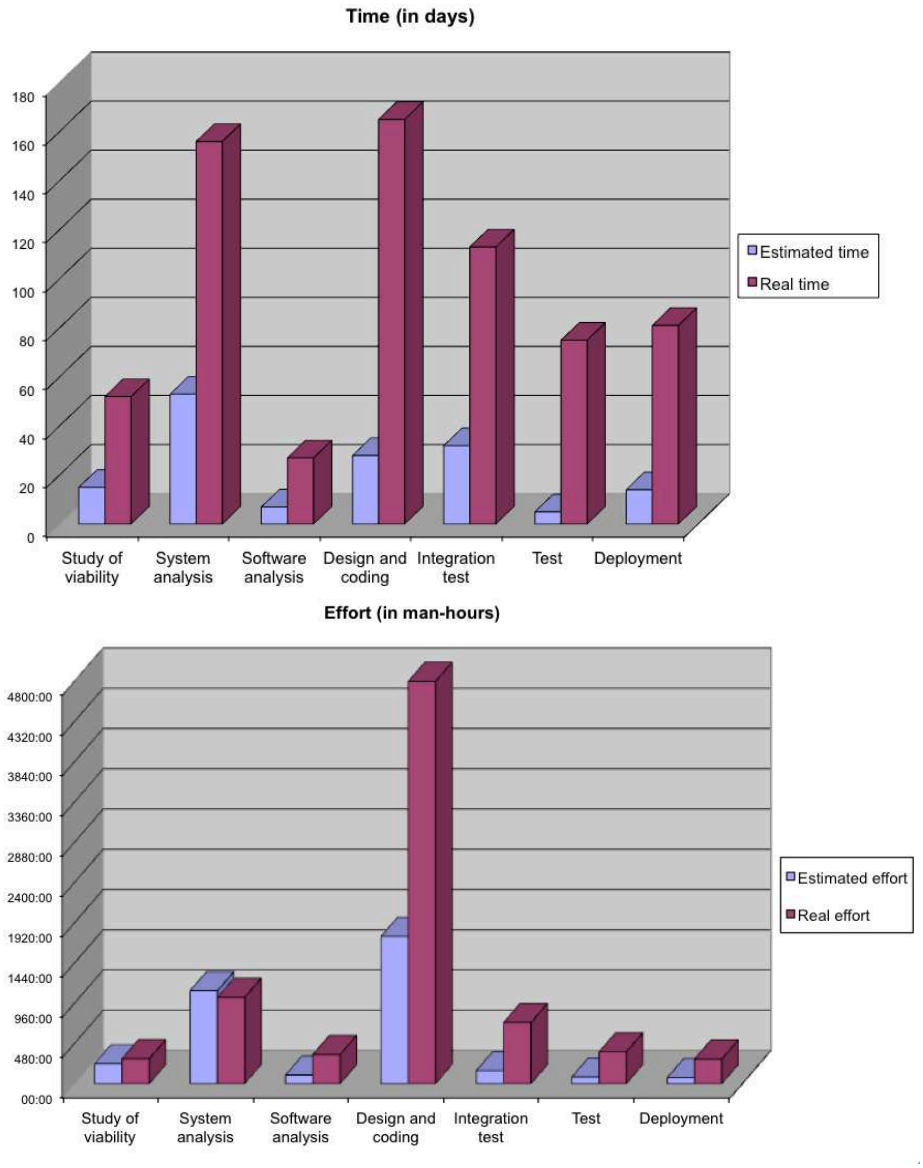


Figure 3.2. Examples of measures report

Quantitative understanding of process performance analysis varies for each organization, depending on the performance measures collected during the execution of software projects (Chrissis *et al.*, 2011). Establishing this historical repository is a current challenge for all organizations involved with SPI. Suitable measures should be clearly defined at the beginning of SPI programs. Yet organizations rarely have the experience to define these measures and cannot always predict which processes will be required to perform process performance analysis. Although SPI models call for early implementation of measures in SPI programs, managers often realize the significance and complexity of those measures only when reaching the high levels of the maturity models. Moreover, some authors (Hall and Fenton, 1997; Agresti, 2006; Berander and Johnson, 2006) argue that several measurement programs in organizations fail because they define too many measures that are not actually implemented and analyzed in decision-making.

With the aim of helping organizations define the measures that can be effectively used in organizational process performance analysis, we conducted a broad search of the literature in order to define a catalog of measures organized by indicators. The idea was that this catalog

could be useful for several analyses of software process performance (such as requirements development, verification, and validation).

To define the catalog (**Monteiro and Oliveira, 2011**) we decided to use measures that have been defined and implemented in organizations as proposed by different authors. As presented in Figure 3.3, we first conducted a broad literature review to identify existing measures for process performance analyses; then, we analyzed those measures and selected the most relevant ones to compose the indicators to be specified.

Out of more than 500 papers found in the literature review on the subject of software process measurement (*step 1* in Figure 3.3), 32 were considered relevant. These cited 869 performance measures used in organizations. Nevertheless, there were redundancies among them: for example, percentage of rework time, rework ratio, and rework were listed as different measures. An analysis of all of the measures was performed in the next four steps.

To start the analysis (*step 2*), we decided to classify each measure according to its main goal in different categories: time, effort, cost, scope, productivity and quality. Next, measures in the same category were analyzed according to name, description and formula (***step 3***). Analyzing all of the similarities among the 869 measures, we ended up with 584 distinct measures.

These 584 measures were mapped with the CMMI-DEV process areas (***step 4***) to allow the choice of measures considering the specific software process to be improved. In this analysis, some measures were mapped with all of the process areas, others only with specific ones. For example, the measures of *cost performance index*, *effort*, and *productivity* are applicable to all of the CMMI-DEV process areas; therefore they were mapped with all of the process areas. On the other hand, we can cite, for example, *requirement volatility* and *delivery defect density* measures that were mapped with specific process areas.

In the end, 345 measures were related to the engineering process areas[12]. With the goal of producing a practical catalog as a generic framework of measures, we decided to reduce the scope of measures by selecting those that were cited more frequently in the literature (step 5), settling on measures that were referenced by three or more authors in different publications. Thus 48 of the total 345 measures were ultimately selected. These were further evaluated for goal crossover, that is, to identify whether different measures shared the same application goal. With this analysis, we eliminated additional measures. For example, we eliminated "requirements stability" (with four references) because it had the same goal as "requirements volatility" (with five references). These measures are based on the percentage of requirements changed ("requirements volatility"), or not changed ("requirements stability") in relation to the total requirements of the project. Despite the differences in the methods of calculation, the two measures assess the same process areas in relation to changes in project scope, thus the measure with the greatest number of literature citations was the one chosen to be maintained in the catalog. The final set of measures is presented in the two last columns of Table 3.2. The complete description of these measures can be found in (**Monteiro and Oliveira, 2011**).

During the execution of *steps 3, 4,* and *5* of the research, in which the measures were examined individually, we reviewed the list of measures, possibly changing the classifications made in previous steps.

To specify the catalog, we adapted the documentation from (McGarry *et al.*, 2002). The decision criteria and indicator interpretation in this documentation was completed with a focus on process performance analysis. In (McGarry *et al.*, 2002) measures are described

[12] The engineering process areas are: requirements management, requirements development, product integration, technical solution, verification, and validation.

according to their indicators. Thus, we looked for correlations among all of the measures (*step 6*). The idea was to group the correlated measures into unique indicators for better quantitative management of processes. Therefore, all measures were specified in the indicator specification.

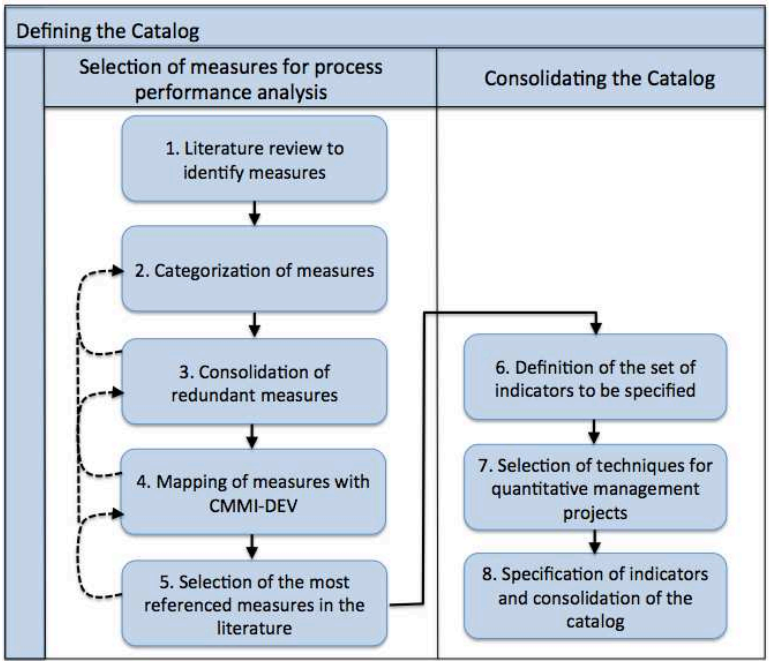


Figure 3.3. Steps for definition of a catalog of indicators (**Monteiro and Oliveira, 2011**)

Each specified indicator uses one or more derived measures, and those measures in turn use one or more base measures. Therefore, some base measures can be specified in different indicators. For each indicator we also determined which control chart to use (*step 7*).

Table 3.2 presents the indicators and their related measures. We note that some measures are specialized in several measures, for instance number of defects is specialized in number of internal test defects, number of internal peer review defects, number of external test defects, number of external peer review defects.

In *step 8***,** we specified each indicator. Appendix A presents a complete specification of the Defect Density indicator. Another indicator specification and a complete description of this study can be found in (**Monteiro and Oliveira, 2011**).

Finally, believing that SPI has a great importance for the final perception of quality by customers, we performed in 2008 a formal evaluation of customer perception of the service quality offered by the software providers (companies that develop software for other companies) appraised in CMM/CMMI (**Santos and Oliveira, 2009, 2007**). The study was formally planned considering the experimental software engineering principles defined by (Wholin *et al.*, 2000), from the clear and formal definition of our goal to the verification of the validity of the results. Based on the work of Parasuraman *et al.* (1988), where the perceived service quality is defined as the difference between the perception (P) and the expectation (E) of customers, we set the following hypotheses:

- H_1: Customers' perception of the service quality does not correspond to their expectation. In other words, customers' perception is inferior to their expectation;

- H_2: Customers and prospective customers have high expectations with regard to the service quality of software providers appraised in CMM/CMMI.

Table 3.2. Indicators X Measures

Indicator	Base Measures	Derived Measures
Earned Value	Planned value Earned value Actual cost Budget at completion	Cost performance index Schedule performance index Cost variance Percentage of cost variance (CV%) Estimate at completion Estimate to complete Variance at completion
Test Coverage	Size of the software tested: Source lines of code, or Number of function points, or Number of requirements Total size of the software product: Source lines of code, or Number of function points, or Number of requirements	Test coverage
Cost of Quality	Cost of conformance Cost of quality control Cost of quality assurance Cost of nonconformance Cost of rework Cost of warranty work Cost of waste and loss of reputation Actual cost	Cost of quality Percentage cost of quality
Defect Density	Number of defects Total number of defects Number of defects by type Source lines of code Number of function points Number of requirements Document page count	Defect density Total defect density Defect density by type
Peer Review and Test Efficiency	Number of defects Number of internal test defects Number of internal peer review defects Number of external test defects Number of external peer review defects	Test efficiency Peer review efficiency
Scope Changes	Number of changes Source lines of code Number of requirements Number of changed requirements Total number of requirements Number of function points	Change rate Requirements volatility
Cycle Time	Cycle time Source lines of code Number of function points Number of requirements	Cycle time relative to project size
Mean Time to Failure	Mean time to failure	

Table 3.2 (cont.) Indicators X Measures

Indicator	Base Measures	Derived Measures
Productivity	Effort Source lines of code Number of function points Number of requirements Schedule	Productivity Delivery rate
Reuse of Code	Source lines of code	Reuse of code
Defect Removal Rate	Number of defects Number of open defects Number of closed defects	Defect removal rate
Rework	Rework effort Total rework effort Rework effort by discipline Effort Total project effort	Rework Total rework Rework by discipline
Effort Variance	Effort Planned effort Actual effort	Effort variance Total effort variance Effort variance by discipline
Schedule Variance	Schedule Planned schedule Actual schedule	Schedule variance Total schedule variance Schedule variance by discipline
Size Variance	Source lines of code Number of function points Number of requirements	Size variance

Two service quality assessment questionnaires were created based on the SERVQUAL (Service Quality) model (Parasuraman *et al.*, 1988) with statements to evaluate the expectation and perception of the respondents (see an example of the statements in Figure 3.4). The first questionnaire was defined for respondents who had never been customers of the software providers appraised in CMM/CMMI; that is, they did not use the service of these providers but they were familiar with the model and could contract one of the providers in the future (prospective customers). The second questionnaire was defined for respondents who had already been customers of software providers appraised in the models. The aim of the noncustomers' questionnaire was merely to identify the expectations of prospective customers about the software providers appraised in CMM/CMMI. Accordingly, it included only the expectation statements of the questionnaire. The customers' questionnaire, in contrast, aimed at assessing the perceived service quality (Perception – Expectation) and included the entire questionnaire.

The questionnaires were made electronically available for the 192 respondents from public and private institutions. We obtained 78 responses to the noncustomer questionnaire and 42 to the questionnaire directed to customers of software providers appraised in the CMMI. Internal consistency is proved through Cronbach's alpha superior to 0.9 and threats of validity were verified.

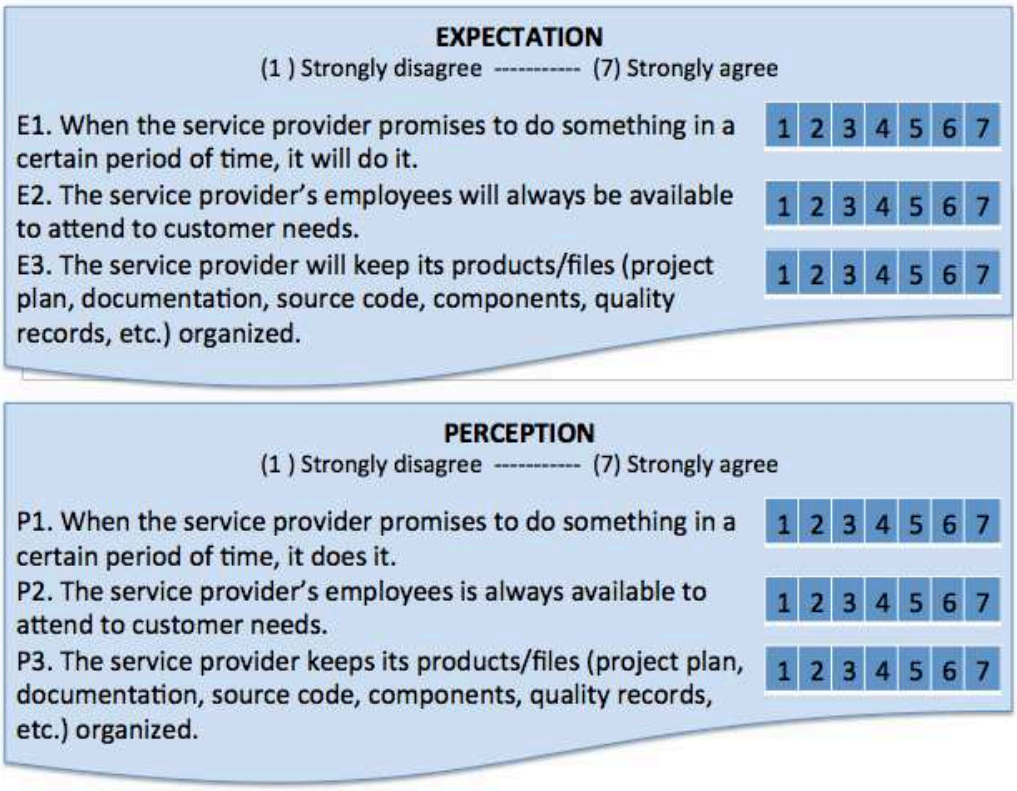


Figure 3.4. Example of statements of the questionnaire to evaluate service quality

This study confirmed that: the customers have a perception of the service quality offered by a CMM/CMMI provider significantly inferior to their expectations; and, the customers and prospective customers have very high expectations of a service offered by a CMM/CMMI providers. In summary, this study has identified some important results for the software industry:

- Customers and prospective customers have similar assessments concerning their expectations, which suggests that the level of demand does not increase when an individual becomes a customer;
- Customers and prospective customers have high expectations concerning service providers, especially in the requirement and project management, and in technical activities, such as technical solution and project integration;
- Perception of customers concerning the services rendered shows a misalignment with regard to expectations, especially on giving information about service completion times.

A detailed description of this empirical study can be found in **(Santos and Oliveira, 2009).**

3.2.3 Return on Investment in Software Process Improvement

Despite the fact that SPI is widely explored in the literature, the lack of evidence about the return on investment (ROI) in SPI is still one of the major reasons that impedes the organizations to implement it (Peixoto *et al.*, 2010; Galinac, 2009; Khurshid *et al.*, 2009). In fact, several approaches have been proposed to support the return on investment (ROI) of SPI (see, for instance (Rico, 2004; Eman, 1998)). The main feature of these approaches is the focus on economic aspects. Nevertheless, several authors (e.g. Unterkalmsteiner *et al.*, 2012;

Campo, 2012; Ferreira *et al.,* 2008, Asato *et al.*, 2009; Lazić and Mastorakis, 2010) defend that the ROI of an improvement program can also be measured by the other benefits (than economics) achieved by the organization such as, productivity and customer satisfaction). Therefore, it is necessary to investigate these benefits and to define how they can be characterized, measured and considered for the return on investment in SPI.

Based on this need, we performed a systematic mapping review (Kitchenham and Charters, 2007) (§ 3.2.3.1) to identify what is been considered in the definition and application of ROI for SPI programs; and we defined a strategy for the analysis of benefits of SPI programs (§ 3.2.3.2).

3.2.3.1 Systematic mapping study about ROI

Systematic mapping is an empirical methodology that provides a wide overview of a research area to establish if research evidence exists on a topic (Kitchenham and Charters, 2007)

Figure 3.5 presents the review protocol defined for this systematic mapping, specifying: the research question(s) being addressed, the methods that will be used to perform the review, the search strategy with the keyword strings and sources used, and the explicit inclusion and exclusion criteria to assess each potential study.

We applied the search string in all digital libraries in November 2012. The search was done in titles, abstracts and keyword. We got at the end 338 papers. By reading the abstracts of all papers and applying the three first inclusion criteria, we selected 112 papers for the full text analysis. From this group, 13 were eliminated by the criteria 5. After the full text reading, 28 papers were selected considering the criteria 2, 3 and 4. After performing the paper analysis, we extracted the data answering each one of the secondary questions to be used in the reporting the review phase.

Figure 3.6 presents the results for Q1 (methods), Q2 (benefits related to measures and intangible benefits) and Q3 (reported result elements) identified in the 28 papers.

Moreover, this study showed to us that (**Ramos *et al.,* 2013**):

- No proposition was used in practice or validated with some kind of experimentation;
- From 9 papers (32%) that present methods, only 5 (17%) presents description of some application in practice;
- Only 10 papers (35%) show some result of the ROI with some application and 2 (7%) presents some proposition in this subject. These two papers present a methodology to compute the cost saving and the ROI of a SPI program (McGibbon and Nicholls, 2002), and the results of the use of a framework (Scott et al., 2001) to the implementation of SPI in small and media enterprises.

Based on the results of this systematic mapping, we identified the need of defining a strategy lighter for the definition of return on investment in SPI that does not use only economic aspects, but that shows the benefits that an organization can reach with those programs aligned to their previous expectations.

Quality focus: Identify the elements that are used to analyze the return on investment in SPI programs.

Main research question: "What are the elements used to analyze the return on investment in software process improvement?".

Secondary research questions:
Q1. What methods, processes and tools are used to analyze the return on investment in SPI programs?

Q2. What measures and other elements are used for analysis of return on investment in SPI programs?

Q3. What return on investment in SPI programs that organizations have reported?

Population: Published papers about ROI in SPI

Sources selection: SCOPUS, ACM and IEEE

Keywords and synonyms:
Software process
Return on investment= ROI, investment analysis, money, profit, engineering economics, investment.
Improvement = SPI
CMMI = CMM, CMMI-DEV[1]
MR-MPS = MPS, MPS.Br
Maturity level = Capability level

Search string:
(“return on investment” OR roi OR “investment analysis” OR money OR profit OR "engineering economics" OR investment) AND ((("software process" AND improvement) OR SPI) OR (cmm OR "CMMI-DEV" OR CMMI OR "MR-MPS" OR MPS OR "MPS.Br" OR "maturity level" OR "capability level"))

Inclusion (IC) and exclusion (EC) criteria:
1. IC - Papers should be written in English;
2. IC – Papers that present approaches about analysis of ROI or benefits in SPI even that are not based on maturity models.
3. IC – Papers that present some kind of study about methods, tools, process, measures or any other item applied for the analysis of ROI or benefits in SPI;
4. EC – Papers that present methods, tools, process, measures that are not related with SPI.
5. EC - Papers not available on the internet.

Intervention: studies involving return on investment in SPI.
Data extraction: methods, tools, process, measures or any other item used for the analysis of ROI or benefits in SPI.

Figure 3.5. Research protocol (**Ramos *et al.*, 2013**)

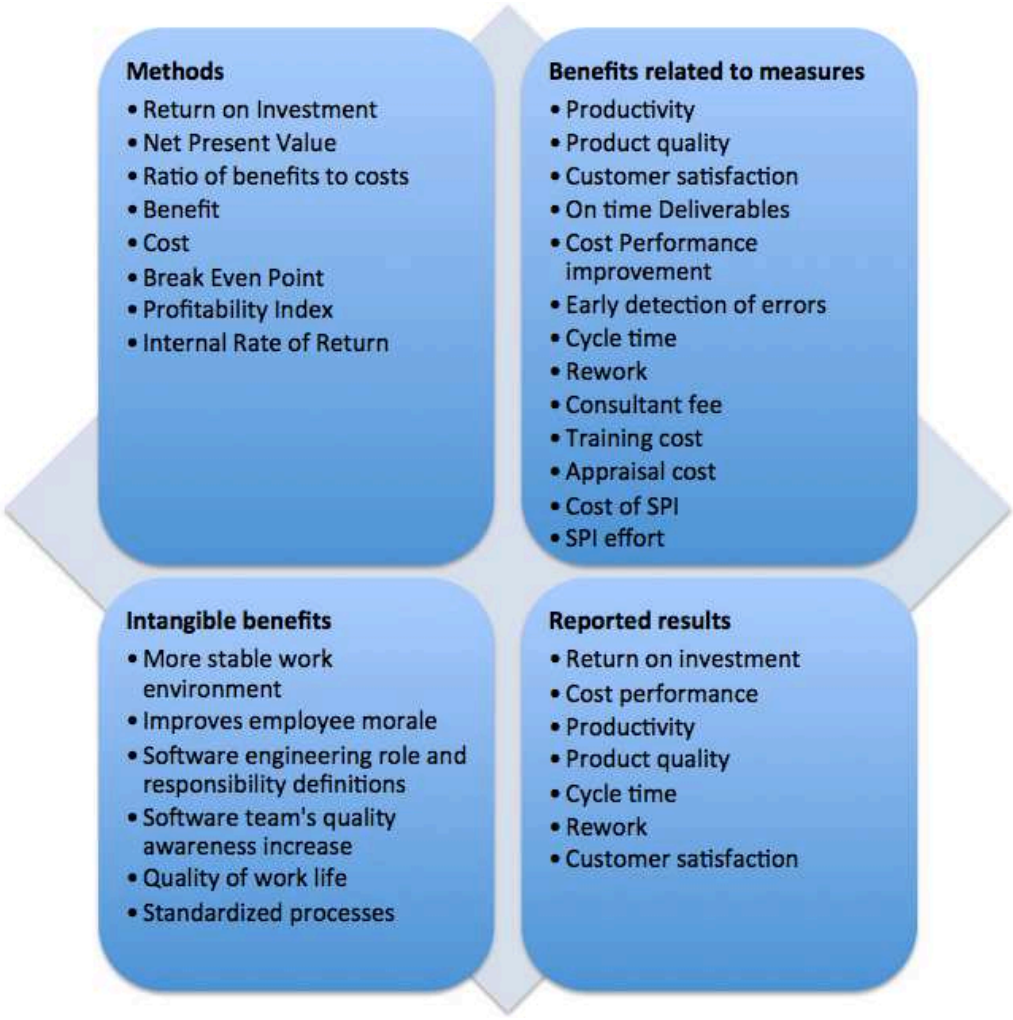


Figure 3.6. Elements for analysis of ROI in SPI

3.2.3.2 A Light Strategy for The Analysis of Benefits of SPI Programs

A SPI program should be aligned to the business goals of the organization. However, it is very common, that organizations start SPI programs without first defining what are their expectations of benefit and quantifying the expected values for these benefits. Moreover, often, organizations do not assess later whether expectations were actually met.

We proposed a strategy to support consultants and process improvement teams working in organizations. Figure 3.7 shows the general view of this strategy that is composed of four main phases explained below.

Identify expected benefits

In this phase the organization's expectation for the institutionalization of a SPI program is identified considering potential benefits of a SPI program. Different stakeholders (technical team, managers, etc.) should participate in this activity since the expectation of benefits is a key element in this strategy. Thus, an analysis should be performed to align expectations of benefits to the organization's business goals, and to verify the feasibility of achieving those benefits considering the expectation of the benefits and the software process practices that will be implemented in the organization.

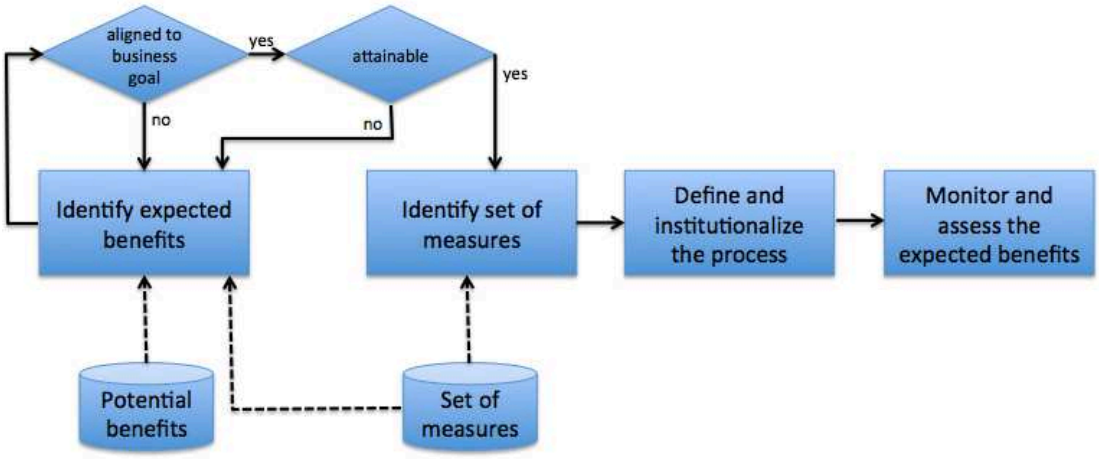


Figure 3.7. Strategy for the analysis of benefits in SPI (**Ramos *et al.*, 2013**)

To support this activity, we are defining a set of potential benefits of SPI programs based on the results of the systematic mapping previously presented. These potential benefits are associated with practices (from CMMI-DEV) or expected outcomes (from MR-MPS-SW). In this way, it is possible to identify what should be incorporated into the standard software process of the organization to enhance the achievement of expected benefits. This mapping between benefits and practices/outcomes also allows evaluating if the expectation of benefits identified is feasible considering the maturity level it will be implemented in the organization.

Identify set of measures

This phase aims at supporting the identification of measures that should be collected to monitor the benefits obtained by the SPI program. The measures must be incorporated into the organization measurement plan that is required since the first maturity levels of the SPI models. These measures must be consistent with the expectations of benefits established in the previous activity.

To support this activity, we are mapping a set of measures that can be used to support the measurement of the potential benefits and costs of the improvement program. We are using the study presented in previous section (**Monteiro and Oliveira, 2011**) where we defined a set of indicators with their software process measures.

Define and institutionalize the process

This phase corresponds to the traditional activity of defining the standard software process for the organization as required in the maturity models. The outcome of this phase is the complete institutionalization of the standard software process that means its real use in the software projects development.

It is important to highlight here that the measures and indicators defined in the previous phases are correlated with process areas (from CMMI-DEV) as defined in (**Monteiro and Oliveira, 2011**). In this way, by defining the standard software process, we know which measures we can use to evaluate the benefits.

Monitor and assess the expected benefits

In this phase, the expected benefits are continually evaluated based on the measures defined in the phase 2. Reports of indicators and measures supports the analysis of benefits being obtained with the SPI programs.

This approach is under development as a Doctoral research in co-supervision in Federal University of Rio de Janeiro. Preliminary results of this research can be found in (**Ramos *et al.*, 2012, 2013**).

3.3 Software Product Quality Assurance

The quality of software product (software requirements, design models, source code, final software system) concerns the assessment according to one or several quality characteristics. It is common that the quality of a software product is multidimensional because it depends of: the software product under evaluation, the point of view of the evaluator (for example, users are interested in some aspect different than a project manager or a software engineer), and the quality characteristic to be addressed (for example, those presented in ISO/IEC 25000 (see Chapter 1, § 2.4). In this section, we present some experiences on the quality evaluation of software products.

3.3.1 Quality Evaluation of Web Applications

With the advent of the web in the late 20th century, we were surprised by a large number of Internet applications, from simple informational sites to complete online applications. With the facility to provide these web applications, it is crucial to ensure the suitable quality for their adoption.

Our first work on this direction was an evaluation of educational web sites for cardiology (**Lima *et al.*, 2000)**. The goal was to select credible and good web sites that could support the students learning. An evaluation questionnaire was defined based on quality characteristics from the literature and standards (see Chapter 1, § 2.4). In this way, the final selection of web sites was justified by the quality assessment performed. In this section, we present two other studies: one for the evaluation of accessibility of web applications; and, the other one for the definition of a technique that ensures the usability of web application throughout the software development.

3.3.1.1 Evaluation of the Accessibility of Web Sites

Accessibility is defined by ISO 9241 (ISO 9241-171, 2008) as the usability of a product, service environment, or facility by people with the widest range of capabilities. ISO 9241 emphasizes also that accessibility aims to achieve levels of effectiveness, efficiency and satisfaction that are as high as possible considering particular attention to the full range of capabilities within the user population. The World Wide Web consortium defines web accessibility as the degree to which people with visual, auditory, physical, speech, cognitive, or neurological disabilities can perceive, understand, navigate, and interact with the Web (Henry, 2005). Accessibility gained large attention in the research community and specific standards and guidelines were developed. One of the most known, the Web Content Accessibility Guidelines (WCAG) was in its first version composed of 14 general guidelines organized into checkpoints that address specific aspects of accessibility. Each checkpoint is rated according to the criticality of implementation of the accessible content (that is, when a content developer must [priority 1] or should [priority 2] satisfy the checkpoint; or when s/he may address the checkpoint [priority 3]). In WCAG version 2.0, checkpoints and priorities are referenced as success criteria and conformance level, respectively. It also defines four basic

principles: the content must be perceivable, the interface components must be operable, the content and controls must be understandable, and the content must be sufficiently robust to interact with the user. Each principle contains general guidelines that are organized into conformance levels and success criteria (Caldwell *et al.*, 2008).

Motivated by Decree 5,296 from Brazil, which makes compliancy to the rules of Web accessibility and efficiency mandatory for Brazilian government websites, we performed a research for the definition and application of measures to evaluate accessibility (**Lima *et al.*, 2009**). We used the Goal-Question Metric (GQM) approach (see Chapter 1, § 2.3) to define the measurements considering the accessibility principles of the WCAG (Table 3.3). Objective methods (based on numerical rules) and subjective methods (based on user opinion) (ISO/IEC 15939, 2002) were used to collect the measurements. The objective measurements were done with tools that collect data concerning real and potential violations, as well as the principles of accessibility (resulting in measurements M2, M7, and M9).

The subjective measurements were registered in a protocol based on participant observation method (Melo *et al.*, 2004) consisting of evaluation sessions with users with visual disability. In this method, the observer interacts with the user to provide him/her with basic information and an accurate understanding of the objective for each task, thereby establishing a dialogue for registering difficulties and strategies adopted by the user during task execution. After completion of the sessions, the perception of users with regard to the accessibility of the site was registered (measurements M1, M3, M4, M5, M6, M8, and M11) along with the time spent to complete each task (M10).

Finally, to complete the evaluation one must:

- Choose a tool that supports the Web navigator in use, allowing evaluation of single pages in a specific Web domain (intra-domain analysis) and multilevel evaluation of each site;
- Choose a website that is considered to have the broadest array of services and that is most relevant and interesting to users participating in this study;
- Choose tasks relative to the evaluation of each site;
- Plan sessions of evaluation with respect to user choice; where to perform the evaluation; configuration of the necessary resources for the evaluation (including technologies for assistance); scheduling and definition of time necessary for each evaluation;
- Execute evaluation sessions;
- Collect and analyze the data using the protocol and the chosen tool.

The initial application of this approach was in the context of government sites for the definition of Service Levels Agreements (SLA). Service level agreements (SLAs) are formal agreements that aim to establish quality standards for services rendered by providers to their clients (Muller, 1999). In general, an SLA contains a service catalog, that establishes measurable attributes of the characteristics of the services, and the steps needed to confirm whether the expected service level was achieved in each service rendered (OGC, 2001).

We followed all the steps presented previously. A detailed description of all those steps and results can be found in **(Lima *et al.*, 2009)**. Three websites that render government services (Secretary of the Internal Revenue, Social Security, Brazilian Institute of Geography and Statistics) were evaluated. To minimize the evaluation time, only one task was chosen for each site, as follows:

- T1 - consult an individual income tax return;
- T2 - register for Social Security;
- T3 - contact the Brazilian Institute of Geography and Statistics.

Table 3.3. Accessibility evaluation measures

Goal: To analyze content on websites, with the purpose of evaluation with respect to accessibility From the viewpoint of the user with visual disabilities (in order to limit the scope).		
Question	Measure	Description
Q1. What is the degree of perception of the content of the websites evaluated?	M1. Degree of perception of content.	Measures the user's perception regarding Web content read by a software screen reader (Jaws, Virtual Vision etc.).
	M2. Number of violations of the Perception principle.	Measures the number of violations found on Web pages pertaining to a task and related to items in WCAG 1.0.
Q2. What is the degree of operability of the content of the websites evaluated?	M3. Degree of operation of content with use of the keyboard.	Measures user perception regarding the operation of Web content with the use of a keyboard or directional instrument (mouse).
	M4. Degree of operation of content in relation to the time of execution.	Measures user perception regarding the time spent during interaction with Web content.
	M5. Degree of operation of content in relation to the complexity of the navigation.	Measures user perception regarding the complexity of navigating during an interaction with Web content.
	M6. Degree of operation of content in relation to the existence of anchors.	Measures user perception regarding the ease with which users are able to find anchors during the interaction with Web content.
	M7. Number of violations of the Operation principle for the task.	Measures the number of violations found on Web pages related to the items that compose the Operation principle.
Q3. What is the degree of understanding of the content of the websites evaluated?	M8. Degree of Understanding.	Measures user perception with regard to understanding of Web content during the execution of a task.
	M9. Number of violations of the Understanding principle for the task.	Measures the number of violations found on Web pages related to items that compose the Understanding principle.
Q4. What is the degree of productivity of the users during their interaction on the Web?	M10. Time of execution of the task	Measures the time spent by the user to execute a task.
Q5. What is the degree of satisfaction of the users when executing tasks?	M11. Degree of satisfaction in the context of accessibility	Measures the degree of user satisfaction in relation to the interaction with Web content in the context of accessibility.

Ten users with different levels of visual disability and experience using the Web participated as volunteers in the evaluations. The average time for each session was two and a half hours. Table 3.4 shows evaluation data of users regarding the items for each question. This table presents the average value calculated, considering the evaluation of the ten users and the average value for the question. All the users managed to complete tasks T2 and T3, but just one completed task T1. Therefore, the measure considered only the time of execution for this user. Some of our conclusions from the measurement analysis are:

- Users, in general, obtained a high degree of perception of the content of tasks T2 and T3;
- All content operation tasks can be considered to be in the satisfactory range, with the exception of task T1, which only one user completed and obtained a degree of operation of content below the one recorded for the other tasks;
- The higher the level of content operation, the lower the number of violations found for this principle per task;
- The higher the levels of comprehension of task content, the lower the quantity of violations found for that principle;
- User satisfaction was affected by the number of principle violations; that is, the higher the rate of violations, the lower the level (degree) of user satisfaction;
- In contrast, the degree of user satisfaction has a direct relation to the measure of execution time.

These measures were then used to define the measurable attributes of the service catalog (**Lima *et al.*, 2007, 2009**).

3.3.1.2 Defining an Inspection Approach for Usability Evaluation based on Experimentation

In recent years, the researchers have increasingly defended the idea of ensuring a good level of usability in the early stages of the development process, the so-called "Early Usability" (Panach *et al.,* 2013; Molina and Toval, 2009). The idea is to evaluate the usability through inspection of models used in the design of applications, such as, UML diagrams (Unified Modeling Language) and interface mockups. However, these approaches have been developed independently for specific models. Considering the importance of the inspection of classical artifacts from software engineering and human-computer interaction (HCI) areas, we proposed a set of techniques for usability evaluation of design models called MIT (Model Inspection Technique for usability evaluation) (**Valentin *et al.*, 2012**), to increase the possibility of detecting defects.

To define these techniques, we consider three basic premises: using the positive aspects of some existing proposals for usability evaluation; using reading techniques because they guide the evaluators to better carry out inspections (Travassos *et al.*, 1999); performing formal experimentation to better define and evaluate the techniques. Following our methodology (Figure 3.8), we first conducted a systematic review of the literature (Kitchenham *et al.*, 2010). This literature review assisted in the identification and analysis of issues not covered by existing techniques, providing the basis for the definition of our techniques. After that, we defined an initial proposal of the techniques. Then, we performed an experimental study, which aims to evaluate the feasibility of using the proposed techniques, and helped us to improve the proposed set of techniques. New feasibility studies can also be targeted.

The systematic mapping looked for journal and conferences papers published between 2000 and 2010 in three digital libraries (IEEE Xplore library, ACM and Scopus), containing technical or usability inspection methods in design models. As a result nine techniques were found: four of them support usability inspection in design models; and five propose models that predict usability. From the four techniques for usability inspection, three require specific computational support for their implementation that restricts the adoption of these techniques only for organizations with access to the specific tool. Although the restrictions on their use three techniques found in systematic mapping contributed to the definition of our proposal, they are: Hornbæk *et al.*'s (2007) proposition for evaluation of usability in Use Case diagrams, Luna *et al.*'s (2010) proposition for usability test planning based on mockups and

interaction diagrams, and Atterer's (2005) proposition for activity diagram design considering usability aspects.

Table 3.4. Evaluation data of 10 users with visual disability

Task	Question	Measure	Measure Average
T1	Q1 (Perceivable)	M1	2.90
		M2	Real violations = 23 Potential violations = 151
	Q2 (Operable)	M3	5.40
		M4	2.30
		M5	5.00
		M6	5.20
		M7	Real violations = 2 Potential violations = 69
	Q3 (Understandable)	M8	3.30
		M9	Real violations = 5 Potential violations = 36
	Q4 (Productivity)	M10	7
	Q5 (Satisfactory)	M11	1.00
T2	Q1 (Perceivable)	M1	5.50
		M2	Real violations = 59 Potential violations = 445
	Q2 (Operable)	M3	5.80
		M4	5.10
		M5	5.80
		M6	6.00
		M7	Real violations = 0 Potential violations = 58
	Q3 (Understandable)	M8	5.60
		M9	Real violations = 4 Potential violations = 19
	Q4 (Productivity)	M10	10.8
	Q5 (Satisfactory)	M11	5.60
T3	Q1 (Perceivable)	M1	6.00
		M2	Real violations = 22 Potential violations = 59
	Q2 (Operable)	M3	6.20
		M4	5.80
		M5	6.00
		M6	5.20
		M7	Real violations = 0 Potential violations = 53
	Q3 (Understandable)	M8	3.30
		M9	Real violations = 0 Potential violations = 38
	Q4 (Productivity)	M10	8
	Q5 (Satisfactory)	M11	1.00

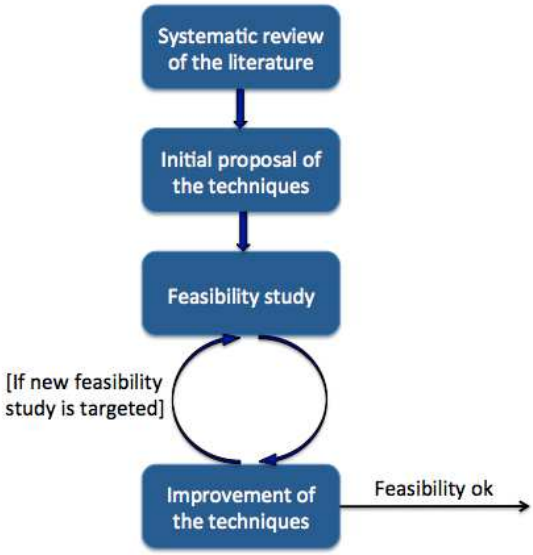


Figure 3.8. Methodology for definition of the techniques (**Valentin *et al.*, 2012**)

Based on these propositions and the Nielsen's heuristics (Nielsen, 1993), we defined three reading techniques. According to Travassos *et al.* (1999), reading techniques are a specific type of inspection technique, which contains a series of individual steps for the analysis of a software product, in order to achieve understanding required for a specific task. The goal is to increase the effectiveness of inspections providing guidelines that can be used by reviewers to analyze a given software artifact and identify defects. Thus, the proposed techniques have a set of instructions that guides the inspector during the evaluation of usability. Three reading techniques were defined to evaluate different artifacts based on a set of usability verification items: 21 verification items to evaluate Use Cases; 32 to evaluate mockups; and, 13 to evaluate activity diagrams. The verification items were organized according to Nielsen's heuristics. Table 3.5 shows examples of verification items for mockup evaluation.

The planning feasibility study was done to evaluate the set of techniques in relation to heuristic evaluation (Nielsen, 1994) since this technique was used as the basis for their definition. To perform this assessment, two quantitative indicators were defined: efficiency (ratio between the number of defects detected and the time spent on inspection) and effectiveness (ratio between the number of defects detected and the total existing defects).

Table 3.5. Example of Verification Items

MIT-2A. Visibility of System Status	
MIT-2A1	Check for any textual information in the Mockups that informs where the user is in a specific moment while using the system.
MIT-2A2	Check for any informational text or message that tells the user what was done after data persistence. For example, when there are update or deletion of some date, a text message should be displayed.

The study was performed with 16 students from a computer science course. They were divided into two groups: one that used heuristic evaluation and the other our reading techniques. They evaluated one use case, two mockups and one activity diagram from a real system. Specific training was performed for both groups. They worked in different rooms and had a moderator who conducted the evaluation. In general, the results showed that the reading

techniques helped to detect more defects than using heuristic evaluation. However, the difference between the number of defects found in the evaluation of mockups by the groups is not statistically significant.

To compare the efficiency and effectiveness of both samples, we used boxplots and nonparametric Mann-Whitney test. For statistical analysis, we used the SPSS software with $\alpha = 0.10$.

Figure 3.9 presents the boxplots that compares both techniques. Analyzing these boxplots we note that:

- The median group for the reading technique for use cases is higher than the median group for heuristic evaluation. Furthermore, when comparing the two samples using the Mann-Whitney test, significant differences were found between the two groups (p=0.0541 for efficiency, p=0.0012 for effectiveness). These results suggest that the reading technique was more efficient and more effective than the heuristic evaluation to inspect the usability of the use case;
- The median of group for heuristic evaluation for mockups is slightly higher than the median for he reading technique. By comparing the two samples using the Mann-Whitney test, there was no significant difference between the two groups (p=0.8665 for efficiency and p=0.5358 for effectiveness). These results suggest that the reading technique and heuristic evaluation provided similar efficiency and effectiveness when used to inspect mockups;
- The median of Group 3 MIT is higher than the median Group HA. This shows that the group of inspectors who used the MIT 3 achieved a performance slightly higher than the group that used the heuristic evaluation. By comparing the two samples using the Mann-Whitney test, significant differences were found between the two groups (p= 0.0022) for efficiency and p=0.0721 for effectiveness). These results suggest that MIT 3 has greater efficiency and efficacy than heuristic evaluation when used to inspect activity diagrams.

We also analyzed the evaluators' comments in the feedback forms. In general, the comments related problems with the ambiguity of some heuristics, the large number of heuristics to be verified and the difficulty for understanding and applying some heuristics. Based on the evaluation and the comments, improvements were performed and a new version of the techniques developed.

A detailed description of this study is presented in (**Valentin *et al.*, 2012**).

3.3.2 Quality Evaluation of Legacy Systems

Since the early 90s (Abran and Nguyenkim, 1991; Harrison and Cook, 1990), it is recognized that software maintenance activity is consuming most of the financial, human and technological resources in an organization, representing about 80% of the cost of development. To deal with this problem, organizations routinely subcontract the maintenance of their software assets to specialized companies, that is to say, outsourced maintainers.

A great challenge for outsourced maintainer is to rapidly evaluate the quality of the systems they will have to maintain so as to accurately estimate the amount of required work. To answer these concerns, we developed a framework of measures to evaluate the complexity of a legacy software system and helps outsourcing maintainers to define their contracts (**Ramos *et al.*, 2004, 2005**). For that purpose, we also used Goal-Question-Metric (presented in Chapter 1, § 2.3).

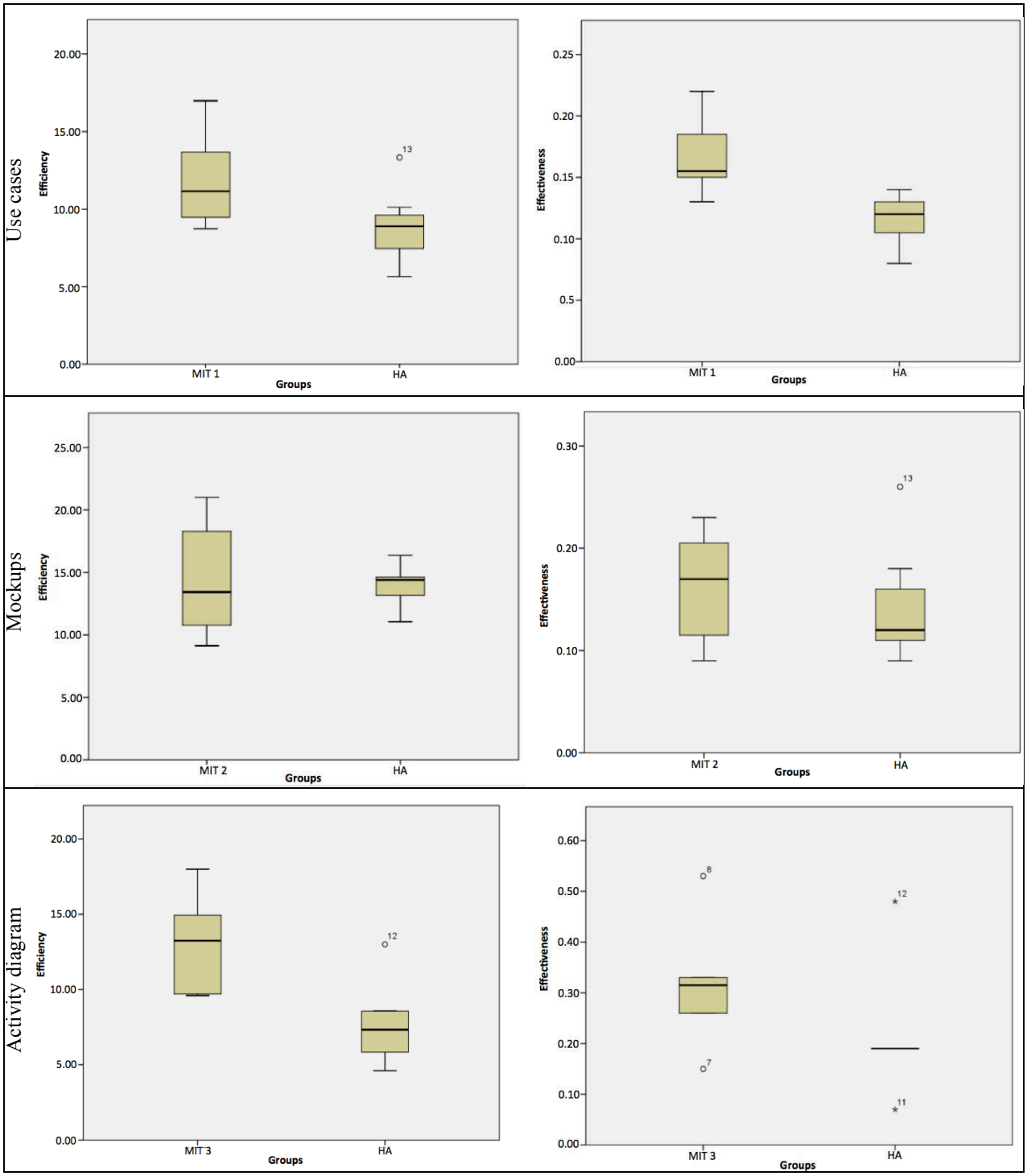


Figure 3.9. Efficiency and effectiveness boxplots by techniques

We defined two measurement goals: assessing the documentation of the system, and assessing its source code. For each goal we defined measurable questions and measures to evaluate them. An important requisite of the measurement plan was that it should not be long to collect the measures, giving preference to automated measures when possible. The reason is that the outsourced maintainer may not spend days or weeks studying a system it may not be contracted to maintain.

Our main interest on this research refers to legacy software. According to Brooke and Ramage (2001), legacy software is perceived by the business to be critical to its operations,

and yet difficult to modify involving high costs in terms of time, skill, etc. It is often described as being any of all of the following (Warren and Ransom, 2002; Brooke and Ramage, 2001; Comella-Dorda *et al.*, 2000; Bennett *et al.*, 1999): large, old, coded with old languages, without documentation (or not updated one), with degraded infrastructure, heavily modified and difficult to maintain. Independently of these characteristics, the system works and owners see no reason to discard it.

To define the measures for the first goal (Table 3.6), we focused on old systems coded and documented with "old-fashioned" languages and methodologies since they exist in a large number. For instance, 60% to 70% of all software in the world are coded in COBOL, and that is tending to grow over time: the number of lines of COBOL code increases by 5 billion new lines each year, and 15% of new systems will still be written in COBOL (Searcord *et al*., 2003). Considering that the possible method to document those systems was structured analysis method, we chose to evaluate: context diagram, data flow diagram, data model and requirements specification. Each measure was defined using a specific type of scale. Some metrics had been defined from the discussion between GQM team members and others were selected from the literature.

Table 3.6. Measures for assessing the documentation of the system

Goal 1: To analyze the system documentation for the purpose of assessing with respect to completeness, consistency and facility to understand from the viewpoints of analysts and programmers in the context of the outsourced maintainer	
Question	Measures
Q1. To what extent is the system documented?	M1.1.1. Percentage of elements (external entities and data flows) from the context diagram that are documented
	M1.1.2. Percentage of elements (external entities, data flows, processes and repositories) from the DFD diagram (preferably level 1) that are documented
	M1.1.3. Percentage of elements (tables, columns and domains) from the data model, which are documented
	M1.1.4. Detail level of requirements documentation
	M1.1.5. Percentage of functional requirements documented in the requirements specification that is described
	M1.1.6. Percentage of functional requirements that has description of business rules
Q2. How easy is the documentation to be understood by the maintainers?	M1.2.1. Ease of understanding the context diagram
	M1.2.2. Ease of understanding the DFD (preferably level 1)
	M1.2.3. Ease of understanding the data dictionary of the data model
	M1.2.4. Ease of understanding the functional requirements specification
Q3. To what extent is the documentation consistent?	M1.3.1. Percentage of integrations with other systems described in the requirements specification that are consistent with the context diagram
	M1.3.2. Percentage of integrations with other systems described in the requirements specification that are consistent with the DFD (preferably level 1)
	M1.3.3. Percentage of elements of the data model (tables, columns and domains) that are consistent with the database.
	M1.3.4. Percentage of all external entities represented in the DFD (preferably level 1) that are represented in the diagram context

To define the measures for the second goal (Table 3.7), a comprehensive review of the literature about software system complexity was performed. Accordingly, maintenance complexity is related to the difficulty to understand/maintain a system considering the number

of lines of code (total and per program/module), number of tables used by the software system itself or by other software systems), number of page-screens to be manipulated, number of technologies languages/platforms) involved, degree of coupling between the programs and the complexity of the code itself.

Table 3.7. Measures for assessing system source code

Goal 2: To analyze the system source code for the purpose of assessing with respect to the complexity to understand and modify it from the viewpoints of analysts and programmers in the context of the outsourced maintainer.	
Question	Measures
Q1. What is the size of the system?	M2.1.1. Size of the system in LoC (physical lines in the file)
	M2.1.2. Average number of LoC per program
	M2.1.3. Distribution of LoC per program
	M2.1.4. Number of tables used (read/update) from the system itself
	M2.1.5. Number of tables used (read/update) from other systems
	M2.1.6. Number of LoC per module
	M2.1.7. Distribution of LoC per module
	M2.1.8. Number of programs
	M2.1.9. Average number of programs per module
	M2.1.10. Number of screens
	M2.1.11. Number of programs for each programming language used
Q2. What is the level of internal transaction in the system?	M2.2.1. Average number of tables from the system accessed in input mode
	M2.2.2. Average number of tables from the system accessed in output mode
	M2.2.3. Average fan-out[13] per program
	M2.2.4. Distribution fan-out per program
Q3. What is the complexity of the code?	Reuse of measure M2.2.3 - Average fan-out per program
	M2.3.1. Distribution of cyclomatic complexity per program
	M2.3.2. Average cyclomatic complexity per program
	M2.3.3. Distribution of indirections by program
	M2.3.4. Average number of indirections per program.
	M2.3.5. Percent of LoC with comments
	M2.3.6. Distribution of comment lines per program
	M2.3.7. Average number of libraries used per program
	M2.3.8. Average number of nested IF's programs
	M2.3.9. Number of programming languages
	M2.3.10. Distribution of programs by programming language
	M2.3.11. Intermodular and intramodular complexity
	M2.3.12. Distribution of Halstead effort metric per program
	M2.3.13. Distribution of Maintainability Index* per program
Q4. What is the complexity of the user interface?	Reuse of measure M2.1.10
	M2.4.1. Distribution of screens per program
	M2.4.2. Distribution of screens fields per program
	M2.4.3. Average number of fields per screen
Q5. What is the complexity of the interface with other systems?	Reuse of measure M2.1.5.
	M2.5.1. Number of tables read from another system
	M2.5.2. Number of updated tables from another system
	M2.5.3. Distribution of tables accessed from other systems by program

* Maintainability index is proposed by (Pearse and Oman, 1995)

[13] Number of programs that is called by a given program

These measures were applied to evaluate legacy systems maintained by a large outsourced maintainer organization from Brazil. This organization has more than 85 systems in maintenance of various sizes and programmed using different programming languages. We chose to work initially with Cobol source code, as it is one of the languages most used in legacy software systems. For this study, we selected 10 software systems entirely or partly developed in Cobol.

Three software analysts evaluated the software documentation. They did not have any knowledge about the systems and had no contact with users or with the maintenance staff. Therefore, they focused only on the available documents. They spent about one hour on the document examination. Then, they responded to some questions elaborated to answer each one of the measures. To carry out the data collection for the second goal, we implemented a tool using Cobol and SQL parsers written in JavaCC. The tool reads the source code of programs written in Cobol and generates reports with the measures obtained. Nevertheless, the measures for the question 4 were not collected.

Table 3.8 presents the results for the measures.

We note about the quality of documentation (goal 1) that:

- For question 1 (system documentation level), when the systems have design diagrams (S1, S2, S5, S6, S7, S8, and S9) they do not have description of their elements (attributes, external entities, repositories, etc.). However, nine of the systems (except S10) have a good level of requirement specification; and, five of them (S1, S5, S7, S8 and S9) have all the functional requirements described;
- For question 2 (easiness of documentation understanding), with the exception of S8, the diagrams, when present, were possible to be understood). However, the level of understanding of the specification was considered as acceptable only for five systems. M1.2.3 was not evaluated because the diagrams do not have description of the elements, considered as essential by the evaluators;
- For question 3 (documentation consistency), only S1 had a good level of consistency between diagrams and specification; the others, when described, do not present consistency. Only the systems S1, S2 and S7 had a high level of consistency between the context diagram and DFD. M1.1.3 was not collected because we do not have access to the database scripts.

We note about the quality of code (goal 2) that:

- For questions 1 (size of the system), bigger systems have more programs (M2.1.1 and M2.1.8), and the number of accessed tables (from 2 to 85) from the system itself is larger than the number of accessed tables (from 5 to 12) from other systems; that means they have low level of interaction with other systems;
- For question 2, in general, the level of tables read and updated as well as the *fan-out* of programs are low;
- For question 3, in general, the systems are considered as complex, even that there were not a lot of indirections (*goto*), and poorly documented;
- For question 5, all systems have a little level of interaction with other systems.

Table 3.8. Measure results

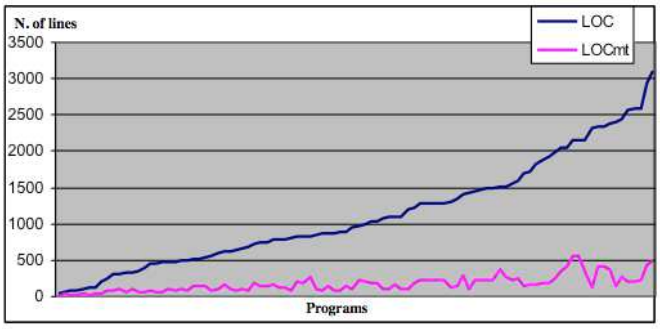
Measures			S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
Goal 1	Q1	M1.1.1	0%	0%	-	-	0%	0%	0%	0%	-	-
		M1.1.2	0%	0%	-	-	0%	0%	0%	0%	-	-
		M1.1.3	-	-	-	-	-	-	-	0%	0%	-
		M1.1.4	3	-	1	-	2	0	2	2	3	-
		M1.1.5	100%	-	50%	-	100%	100%	100%	74%	100%	-
		M1.1.6	100%	-	100%	-	100%	0%	98%	38%	100%	-
	Q2	M1.2.1	2	2	-	-	2	2	2	1	-	-
		M1.2.2	2	2	-	-	3	2	2	1	-	-
		M1.2.3	-	-	-	-	-	-	-	-	-	-
		M1.2.4	1	-	2	-	3	0	2	2	3	-
	Q3	M1.3.1	83%	-	0%	-	-	-	0%	-	0%	-
		M1.3.2	83%	-	0%	-	-	-	0%	-	0%	-
		M1.3.3	-	-	-	-	-	-	-	-	-	-
		M1.3.4	100%	100%	-	-	33%	42%	100%	-	0%	-
Goal 2	Q1	M2.1.1	110,817	135,686	49,646	77,650	62,276	100,236	14,528	99,517	153.395	2,781
		M2.1.2	1,131	590	814	388	490	964	2,075	630	1.322	348
		M2.1.4	41	67	2	85	37	32	7	74	48	7
		M2.1.5	12	1	0	1	10	5	0	1	0	0
		M2.1.6	1,131	590	814	388	490	964	2,075	630	1.322	348
		M2.1.8	98	230	61	200	127	104	7	158	116	8
		M2.1.9	98	230	61	200	127	104	7	158	116	8
	Q2	M2.2.1	3.26	1.77	0.06	2.34	1.45	1.47	3	3.65	3,68	0.62
		M2.2.2	0.83	0.75	0.03	0.56	0.41	0.26	0.14	1.05	1,01	0.87
		M2.2.3	2.07	3.2	0.93	0	0.007	2.83	7	0	4,71	0
	Q3	M2.3.2	32.74	24.26	57.95	15.74	23.78	47.52	45.28	40.80	39,06	8.25
		M2.3.4	0.33	0.10	0.95	0.0787	0.18	0.45	6.46	0.25	0,33	1,03
		M2.3.5	15.31%	14.57%	16.00%	14.41%	16.61%	21.58%	15.14%	16.26%	15.63%	25.85%
		M2.3.8	4.72	3.32	10.19	0.98	2.80	8.65	8	5,5	2,79	0,375
		M2.2.3	2.07	3.2	0.93	0	0.007	2.83	7	0	4,71	0
		M2.3.11	8.05	17.2	4.37	0	0.007	68.14	211.28	0	25,35	0
	Q4	Not collected										
	Q5	M2.5.1	12	1	0	1	6	5	0	1	0	0
		M2.5.2	6	0	0	0	5	0	0	0	0	0

Legend:
- M1.2.1, M1.2.2, M1.2.3, M1.2.4: 0=Insufficient, 1=Difficult to understand, 2= Understandable, 3=Easy to understand
- M1.1.4: 0= Insufficient, 1=Little detailed, 2=Detailed, 3=Well detailed

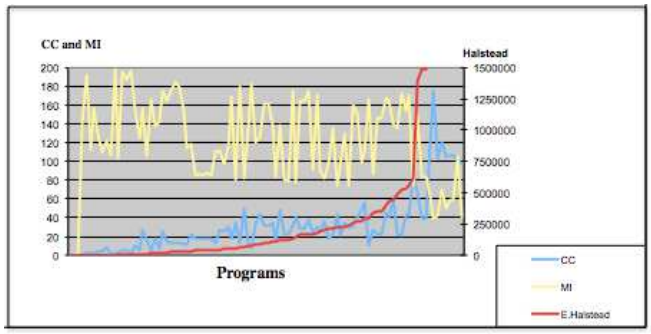
Some measures require the generation of distribution graphs. To decrease the number of necessary graphics and facilitate their interpretation, these measures were organized into three groups: (a) lines of code and comment lines in order to assess the relationship between the percentage of reviews and the size of programs, (b) Halstead, cyclomatic complexity and maintainability index, the three metrics of distribution that are more related to the complexity of the programs, and finally, the (c) fan-out and number of indirections (goto). Distribution graphs were produced for each group of measures. For instance, Figure 3.10 shows the three distribution graphs generated for system 1, for which we conclude that:

- The amount of comment lines grows slowly compared to the growth of number of lines of program code;

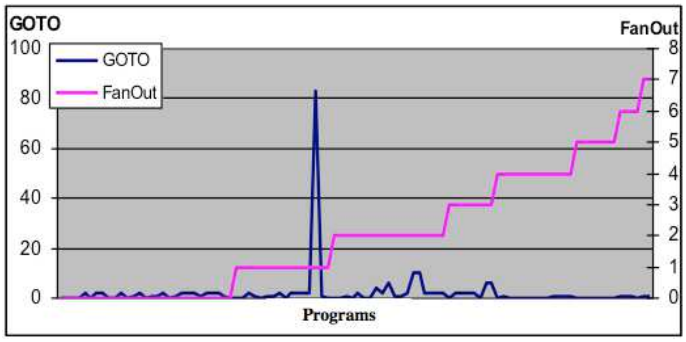
- Two of the complexity measures (cyclomatic complexity and Halstead effort) have consistency, that is, when the values for cyclomatic complexity increases the Halstead effort also increases;
- In general, it has a low level of indirections (*goto*) but some programs have high values of *fan-out* (i.e., it calls a lot of other programs) considering that the value 7 is indicated as the maximum accepted by Page-Jones (1988).



(a) LOC and commented LOC distribution



(b) Complexity distribution



(c) *Fan-out* and *goto* distribution

Figure 3.10. Distribution graphs for S1

After collecting all the measures, they were organized in a same document for each one of the system. The documentation and measures were analyzed in a feedback meeting with five experts in maintenance (2 academic experts and 3 from industry). After a discussion about the results of each system, the experts classified the documentation and the system code in a 3-point Likert scale (low, moderate, high) by analyzing all measures.

Finally, to define the impact of the quality of the system in the contracts, we proposed a combination of the classifications presented in Table 3.9 (for example, low quality of documentation and high complexity of the system code imply a high impact in the contract).

Table 3.10 presents the final classification of the systems impact in maintenance subcontracting. For instance, S1 has a moderate impact because the code has moderate complexity despite a good documentation.

Table 3.9. Decision of impact in maintainers' contracts

(e.g. Documentation= low quality and System code=high complexty ➔ high impact in the contracts)

System code / **Documentation**	High complexity	Moderate complexity	Low Complexity
High quality	Moderate	Moderate	Low
Moderate quality	High	Moderate	Moderate
Low quality	High	High	Moderate

Table 3.10. Classification of the impact in the contract of the evaluated systems

System	Documentation	System code	Impact
S1	High quality	Moderate complexity	Moderate
S2	Low quality	Moderate complexity	High
S3	Low quality	Low complexity	Low
S4	Low quality	Moderate complexity	High
S5	High quality	Moderate complexity	Moderate
S6	Moderate quality	Moderate complexity	Moderate
S7	High quality	High complexity	Moderate
S8	Moderate quality	Moderate complexity	Moderate
S9	Moderate quality	Moderate complexity	Moderate
S10	Low quality	Low complexity	Moderate

When maintenance becomes more frequent and more expensive, decisions regarding continued investments in the maintenance of a system, or its replacement, take place. Several methods for decision support on the modernization of systems have been proposed (Sneed, 1995; Ransom *et al.*, 1998; Visaggio, 2000; De Lucia *et al.*, 2001; Aversano *et al.*, 2004). The modernization strategies proposed in these methods, such as reengineering, replacement or re-implementation are typically used when you already have the perception that some action must be performed to modernize a system. Therefore, it is a reactive way of dealing with the problem (Koskinen *et al.*, 2005). It would be important to detect the problem as early as possible to take preventive actions.

The importance of proactive methods in decision support systems on modernization stands before the current levels of use and relevance of these assets in organizations. This proactivity is achieved through the early detection of deterioration of the software systems, optimizing the use of these resources and the possibility of dealing with problems in its early stages. To detect possible deviations, periodic evaluations of technical, operational and functional performance of software systems should be performed. Based on these evaluations, the manager can better plan corrective actions. Jermacovics *et al.* (2007) define software deterioration as a piece of software that ages while it is being maintained, as consequence maintenance becomes more and more difficult. According to Harris and McRory (2006), continuous maintenance can lead the software system to degradation of its technical quality and its value for the organization.

Based on this scenery, we developed an approach for continuous monitoring of the software system based on management measures for the early identification of deterioration (**Queiroz *et al.*, 2009**). This approach support managers to carry out preventive actions with

the possibility of treating problems in early stages, before the adoption of the modernization strategies proposed in the traditional methods. Similarly, for definition of the measures we used GQM.

From a large literature review about maintainability where we found 125 different measures and managers interview, we end up with 12 measures that fitted our proposal, that is, measures that focus on software deterioration and consider the managers point of view, that could be evaluated over a period of continuous operation of the system (therefore, they should be automated measures), and that are simple and feasible of determination with respect to the cost and frequency of evaluation. The 12 measures that met all criteria were: failure volume, backlog size, effort per demand, productivity, corrections cycle efficiency, maintenance cost, mean time between failures and response time of the system, average cost per demand, budget participation percentage, number of new demands, and level of use of hardware.

To allow analysis of the deterioration, it was necessary to define decision criteria for determining the need for action or further investigation. The decision criteria are represented by the historical values of the measures over the operation of the system. Specifically, for measurements computed annually, we used the values calculated in the previous year as a decision criterion for the identification of deterioration. Thus, the measures indicate deterioration of the system if the calculated values in the current year are higher or worse than the historical values calculated.

This approach was applied to assess the deterioration of three software systems from a large organization from Brazil. The results showed that the measures are feasible to use in practice and that they really supports the system deterioration monitoring. More information about this study can be found in (**Queiroz *et al.*, 2009**).

3.3.3 Quality Evaluation of Human-Computer Interaction in Ubiquitous Systems

Ubiquitous Computing is a new computing paradigm, announced by Mark Weiser in 1991 (Weiser, 1991), that proposes the adoption of computational devices in various sizes, shapes and functions to support daily activities of users. These systems will be everywhere around users, connected to each other, and providing services which have to be as natural as possible (Figure 3.11). To achieve this, the applications are embedded in everyday objects and capable of monitoring user behavior and environment (Rocha *et al.*, 2012).

In ubiquitous computing, the interaction between the user and the system is of utmost importance and the quality of this interaction has a direct impact on the use and adoption of the system. Human-Computer Interaction (HCI) issues are, therefore, on top of the researcher's attention. These issues are even more relevant if we consider the diversity of existing technologies (e.g. smartphones, tablets) using different types of interaction (e.g. audio, video, text, image) between user and computer devices. The interfaces should be adapted to different technologies, considering different contexts in transparent and intuitive ways for the end user. To properly carry out a quality evaluation of the HCI in ubiquitous systems, it is essential to know which quality characteristics and measures should be considered for this specific type of application.

With the goal of identifying those quality characteristics, we performed a large literature review using a systematic mapping study (Santos, 2014). Using this methodology, we found 868 different papers pertinent to the subject. By reading all abstracts, we selected 119 papers. With a deep reading of all papers, we selected 32 of these papers that discussed some quality aspect specific to the HCI in ubiquitous systems (Abi-Char *et al.*, 2010; Cappiello *et al.*, 2009; Chang and Lin, 2011; Damián-Reyes *et al.*, 2011; de Moor *et al.*, 2010; Evers *et al.*,

2010; Haapalainen *et al.*, 2010; Iqbal *et al.*, 2005; Jafari *et al.*, 2010; Jia *et al.*, 2009a; Kemp *et al.*, 2008; Kim *et al.*, 2008; Ko *et al.*, 2010; Kourouthanassis *et al.*, 2008; Kryvinska *et al.*, 2011; Lee and Yun, 2012; Lee *et al.*, 2008; Liampotis *et al.*, 2009; Ranganathan *et al.*, 2005; Ross and Burnett, 2001; Rubio and Bozo, 2007; Schalkwyk *et al.*, 2010; Scholtz and Consolvo, 2004; Sousa *et al.*, 2011; Sun and Denko, 2008; Thompson and Azvine, 2004; Toch, 2011; Wagner *et al.*, 2012; Waibel *et al.*, 2010; Warnock, 2011; Weihong-Guo *et al.*, 2008; Wu and Fu, 2012; Zhang *et al.*, 2006).

Figure 3.11. Ubiquitous computing[14]

From those 32 papers, we extracted quality characteristics and issues presented in the papers as important to be considered in an HCI evaluation of a ubiquitous system but not formally defined as a quality characteristic. We consider them all as characteristic in our analysis. For this analysis, we identified only characteristics that impact on user interaction with the ubiquitous system. We found 134 quality characteristics. Figure 3.12 show the distribution of quality characteristics in the studies that present most quality characteristics.

[14] source: http://www.internationalnews.fr/article-31818821.html (Accessed March 24, 2014)

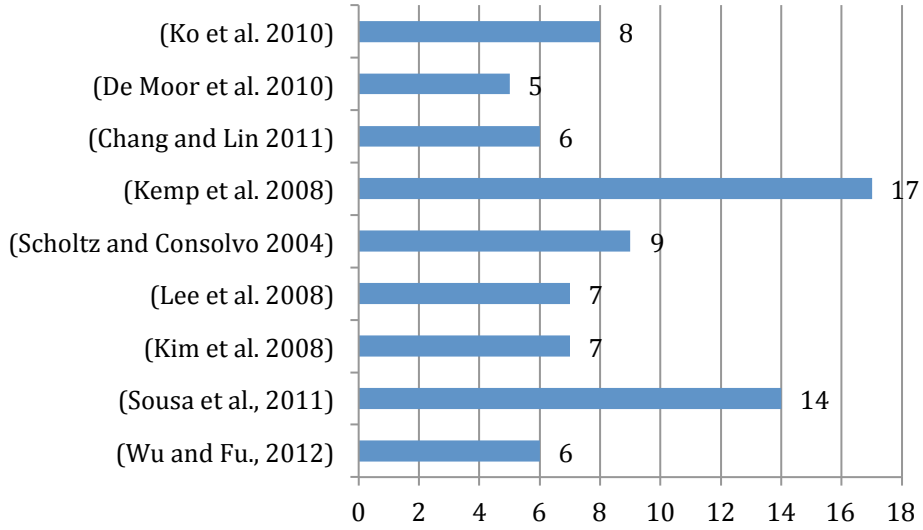


Figure 3.12. Number of quality characteristics by study

Some limitations identified in those studies are:

- Some papers propose only a list of quality characteristics, but without a clear definition of them (Kim *et al.*, 2008; Souza *et al.*, 2011; Zhang *et al.*, 2006);
- When the definition is provided, we found papers that use different names for the same characteristic; for example, Kim *et al.* (2008) define *transparency*, while Kourouthanassis *et al.* (2008) define it as *diffusion*; Lee and Yun (2012) define context-awareness as *context support*, while Kim *et al.* (2008) define it as *adaptability* and Scholtz and Consolvo (2004) as *adaptability*;
- Some papers have no new characteristic specific for HCI of ubiquitous system, but use only the traditional one such as *usability* (Ross and Burnett, 2001; Abi-Char *et al.*, 2010, Cappiello *et al.*, 2009);
- Some papers propose new characteristics specific for ubiquitous systems (e.g. context-awareness, transparency), but do not apply them for quality evaluations (Kourouthanassis *et al.*, 2008; Kim *et al.*, 2008).

Thus, it was necessary to perform an analysis of all definitions to obtain a final set of quality characteristics. This analysis was performed in three steps, as follows:

i) One researcher read all papers and identified characteristics and their synonyms, and produced a document;
ii) This document was peer reviewed (two other researched performed the review by examining the definitions and the papers in case of doubt). For the peer review, the researchers identified if they agreed or not with the synonym identification. If they disagreed, they should write down some justification and propose a new organization;
iii) A meeting was performed for the final consensus.

At the end we got 27 characteristics presented in Table 3.11.

.

Table 3.11. Characteristics for quality evaluation of HCI in ubiquitous systems

Characteristic	Definition
Acceptability	It refers to the desire to use an application and utilization frequency.
Availability	It is the system's capability to provide users with continuous access to information resources anywhere and anytime.
Calmness	It refers to the new technology that prevents humans from feeling overwhelmed by information.
Context-Awareness	The ability to perceive contextual information with respect to system users, system and environment and proactively and dynamically adapt its functionality.
Device Capability	It refers to properties of the device where the application will run: screen size, color depth, battery life.
Ease of Use	The system should be easy to use for the target user group
Effectiveness	The accuracy and completeness with which users achieve certain goals
Efficiency	The relation between the accuracy and completeness with which users achieve certain goals and the resources expended in achieving them
Familiarity	User interactions with the system should improve the quality of their work. The user should be treated with respect.
Focus	The ability to keep the user's attention to their main activity and not on the system and the technology involved.
Interconnectivity	Ability of sharing into the interconnected network
Mobility	Ability to provide users with continuous access to resources and information, regardless of their location within the limits of the system.
Network Capability	It refers to information network infrastructure: signal strength, delay, jitter.
Predictability	Ability to predict the result of running the system from past experiences
Privacy	The ability to maintain information and private data.
Reliability	The ability to maintain a particular level of performance when used under specific software conditions.
Robustness	Degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions
Safety	The level of risk of harm to people, business, software, property or the environment in a specified context of use
Scalability	Ability to provide services to few or a large number of users
Security	The protection of transport and storage of information and security controls about who can access, use and/or modify context information.
Simplicity	The user interface and the instructions are simple
Transparency	The integration of a system into the user environment. It is the extension which the system consists of hidden components in the physical space and interaction is performed through natural interfaces
Trust	It is the users' belief that the system uses their data properly and not cause any harm. It involves issues of privacy and control.
Usability	The ability of the software to be understood, learned, used and attractive to the user, when used under specified conditions
User satisfaction	The degree of user acceptance and how the system is attractive for the user
Utility	Ability to provide value to the user. The application should provide assistance to the user that previously was not available.

We note that there are characteristics that are clearly sub-characteristics from others, for instance: *ease of use*, *efficiency*, *effectiveness*, *users' satisfaction*, *simplicity* and *familiarity* characteristics are clearly components of *usability; security* and *privacy* can be encapsulated in *trust*.

It is also important to note the difference from *utility* and *usability*. Utility, according to Nielsen (1993), refers to the system being able to do what is necessary for the user, and usability with regard to how well users can perform the functionality of a system. They all impact the *acceptability* of the system by the user.

Many of these characteristics are presented in international standards like SQUARE (ISO /IEC 25000, 2005) and ISO 9241 (ISO 9241-11,1998), for example, usability, reliability, efficiency and effectiveness. They usually can be used to evaluate any type of product. Moreover, some characteristics are not defined in these standards, and yet they are likely particular for the user interaction with ubiquitous systems (e.g. context-awareness, transparency, focus and calmness).

Measures to evaluate those characteristics are not well defined in the papers. Usually there is a general description, but no definition about their measurement compute, collect and interpretation procedures. Therefore, we started to define and to evaluate measures for each one of those quality characteristics. We began by defining measures for context-awareness, transparence, mobility, focus and calmness because we considered these characteristics as being the most specific for ubiquitous system.

To define the questions and measures, we used GQM. Table 3.12 shows, for instance, the measures defined for context-awareness. To that end, we had to analyze the meaning of being context-aware. Abowd *et al.* (1999) define context-awareness when a system uses context information to provide relevant services to the user. This context can include any information used to characterize the situation of an entity, which is a person, place, or object considered relevant to the interaction, including the user and application themselves. Some aspects of context-awareness directly impact HCI quality and, therefore, they need to be evaluated. One aspect is the adaptation correctness, which means the system adapts correctly with regard to the services and information provided. Some factors that can influence this correctness are: the context correctness, as if the context is wrong, the adaptation will be likely be wrong too; and the context changing frequency, as if the changes occur quite often, the adaptation may not take place before another change occurs (Cheng *et al.*, 2009).

Another aspect is the time taken to adapt, since the information and/or services should be delivered in a reasonable time to the user. Based on this analysis, we defined the questions and measures presented in Table 3.12. We note that to calculate Adaptation Correctness and Context Correctness measures, we should identify which adaptations the ubiquitous system proposes to do (for example, adaptation for different devices) and also which context information they use for these adaptations (for example, the screen resolution context information to adapt the application behavior in different devices). The resulting measures will be the average from the individual adaptation and context correctness. The interpretation values are an initial proposition, based on our own experiences and (Cheng *et al.*, 2009). They will be refined after concluding more case studies.

Those measure were applied (**Santos *et al.*, 2013**) for the evaluation of a mobile and context-aware application called GREat Tour (Marinho *et al.*, 2013). This application runs on the visitor's mobile device and provides information about the laboratory's rooms that this user is visiting, using texts, images and videos.

To collect the measures presented in Table 3.12, we first identified the context information and adaptations considered by GREat Tour. It presents two adaptations (N=2) and two context information (M=2). The first adaptation is the laboratory map view according to user's location. To this purpose, the system identifies the room through the user's mobile device that reads the QR Code installed in all the doors of the laboratory's rooms. With this input, GREat Tour updates the user's map. The second adaptation is about showing media according to the device battery level. When the battery level is low (0-9%), only text appears, when it is medium (10-20%), texts and images are displayed and when it is high (21-100%), text, images and videos are displayed. Thus, the Adaptation Correctness measure takes into account the laboratory map view (i=1) and the media view (i=2) as adaptations. The Context

Correctness measure takes into account user's location through QR Codes (j =1) and battery level (j =2) as context information. The Context Frequency takes into account changes in location (F) and the Adaptation Time, the time required to show the new map to the user (T).

Table 3.12. Measures for context awareness

Goal: To analyze ubiquitous systems for the purpose of evaluating with respect to the context-awareness from the viewpoints of final users		
Question	Measures	
Q1. What is the adaptation correctness degree?	M1.1. Adaptation correctness	$\frac{\sum_{i=1}^{N}(Ai/Bi)*100}{N}$ N=Number of adaptations Ai=Number of correctly performed adaptations i Bi= Number of performed adaptations i
	M1.2. Context correctness	$\frac{\sum_{i=1}^{N}(Ai/Bi)*100}{N}$ M=Number of different context information Aj=Number of correct collected context information j Aj=Number of collected context information j
	M1.3. Context frequency	F=Frequency of changing
Q2. What is the adaptation average time?	M2.1. Adaptation time	T= The time taken to adapt

The data needed to compute the measures were collected both automatically and manually. Automatic data were recorded in logs that contain: the URL of map presented according to QR Code captured, the time taken to show the map after the capture of the QR Code, the exact time (hour, minute, second and millisecond) when the context information was collected to calculate the context changing, the device battery level and the media presented with this value. Qualitative data were collected in forms filled in by evaluators that followed users during the tours, observing if the application worked correctly, in other words, identifying if the system performed a correct adaptation with correct information.

Twelve users participated in the evaluation. All of them had experience with context-awareness mobile application and were from computer science domain. Their tasks were divided into three laboratory tours, with each tour consisted of three rooms to be visited.

The visit consisted of updating the user map and viewing all the available information. Each tour was done with a device in different battery charge levels making possible to evaluate the different types of adaptation. The twelve users were equally divided in three groups to execute the test with different sequences of battery level, as presented in Table 3.13.

The final result was calculated by the average of all users tours. The result is shown in Table 3.14. The measures about correctness had high results; only one of them was not equal to 100% (Adaptation Correctness). This happened because with adaptation i=1, the wrong map was displayed. We investigated this result and identified that the instability of the wireless network was a possible cause for this adaptation problem. The Context Frequency measure was low, because the result was in minutes, that is to say, it takes about one minute for a context change to happen. If changes occur frequently, the adaptation may not take place before another change occurs, influencing measures of adaptation correctness. The Adaptation

Time was short (milliseconds). It is interesting to note that this result was inferior than the Context Frequency, favoring adaptation correctness.

Table 3.13. the performed scenarios in our evaluation

Tour	Visited Rooms	Group 1	Group 2	Group 3
		Battery Level		
1	Seminars Room Library High Low Medium Administrative Room 1	High	Low	Medium
2	Prototyping Room Software R&D Lab 1 Medium High Low	Medium	High	Low
3	Kitchen Administrative Room 2 Research Lab	Low	Medium	High

Table 3.14. Results for the evaluation of context-awareness

Measures	Results		Interpretation
M1.1. Adaptation Correctness	when i=1, 96% when i=2, 100%	98%	High Correctness
M1.2. Context Correctness	when j=1, 100% when j=2, 100%	100%	High Correctness
M1.3. Context Frequency	00:01:37		Low Frequency
M2.1. Adaptation Time	539 ms		Short Time

Based on the collected results, we can see that the high degree of correctness and low adaptation time provide to the GREat Tour application a good HCI regarding context-awareness measures.

The study described in this section is a long-term project in partnership between Federal University of Ceará (Brazil) and University of Valenciennes. This first case study can be found in (**Santos *et al.*, 2013**). Two other case studies are described in (Santos, 2014).

3.3.4 Evaluation of Information Density of Graphical User Interfaces

One new way to design interactive systems is to automatically compose them from existing systems. An interactive system encompasses a functional core (FC) and a user interface (UI). In consequence, to compose an interactive system, it is necessary to compose FC and UI. Many studies of the software engineering community focus on design or runtime composition of FC through components (Szyperski *et al.*, 2002) or services (Papazoglou, 2003). However, providing UI of good quality is essential to make the composed system acceptable to the users. To address this need, the HCI community has studied how to compose UI at different levels of granularity (see for example, UI generation (Myers, 2009), adaptive UI (Tan *et al.*, 2004), Mashups (Lin *et al.*, 2009), UI composition (Gabillon *et al.*, 2011; Lepreux *et al.*, 2010; Lepreux *et al.*, 2006; Pinna-Dery *et al.*, 2003). Several options of UI composition are available in these works. The main challenge is how to choose the best composition option in order to provide UI of good quality. We propose to address this challenge by defining measures to evaluate automatically the quality of composed UI (**Gabillon *et al.*, 2013**). Note that we mean by UI, a WIMP[15] graphical UI composed from existing known components.

[15] Windows, Icons, Menus and Pointing device.

Several quality models for usability have also been proposed, such as the most cited proposed by Nielsen (1993), the standard ISO 9241-11 (1998) and the consolidated model called QUIM (Quality in Use Integrated Measurement) (Seffah *et al.*, 2006). Guidelines and ergonomic criteria for usability evaluation were also proposed. For example, Vanderdonkt (1994) proposed 3.700 rules to evaluate ergonomic aspects for user interfaces. Scapin and Bastien (1997) proposed a set of criteria (guidance, workload, explicit control, adaptability, error management, consistency, significance of codes, and compatibility) sub-criteria and recommendations for UI development that has been largely used by several researchers.

To evaluate automatically UI, we need to define objective measures and perform quantitative evaluation. In this way, we decide to perform the automatic evaluation of the composed UI by using the ergonomic criteria proposed by Scapin and Bastien (1997). Our first study was focused on the information density criterion. Information density concerns the users' workload from a perceptual and cognitive point of view with regard to the whole set of information presented to the users rather than each individual element or item. Therefore, it is an important criterion since it is directly worried about the user point of view. In addition, this criteria can be based on objective and digital information, such as screen size or the number of labels, which facilitates the automatic evaluation Finally, information density has a great influence on other criteria (for example legibility or prompting).

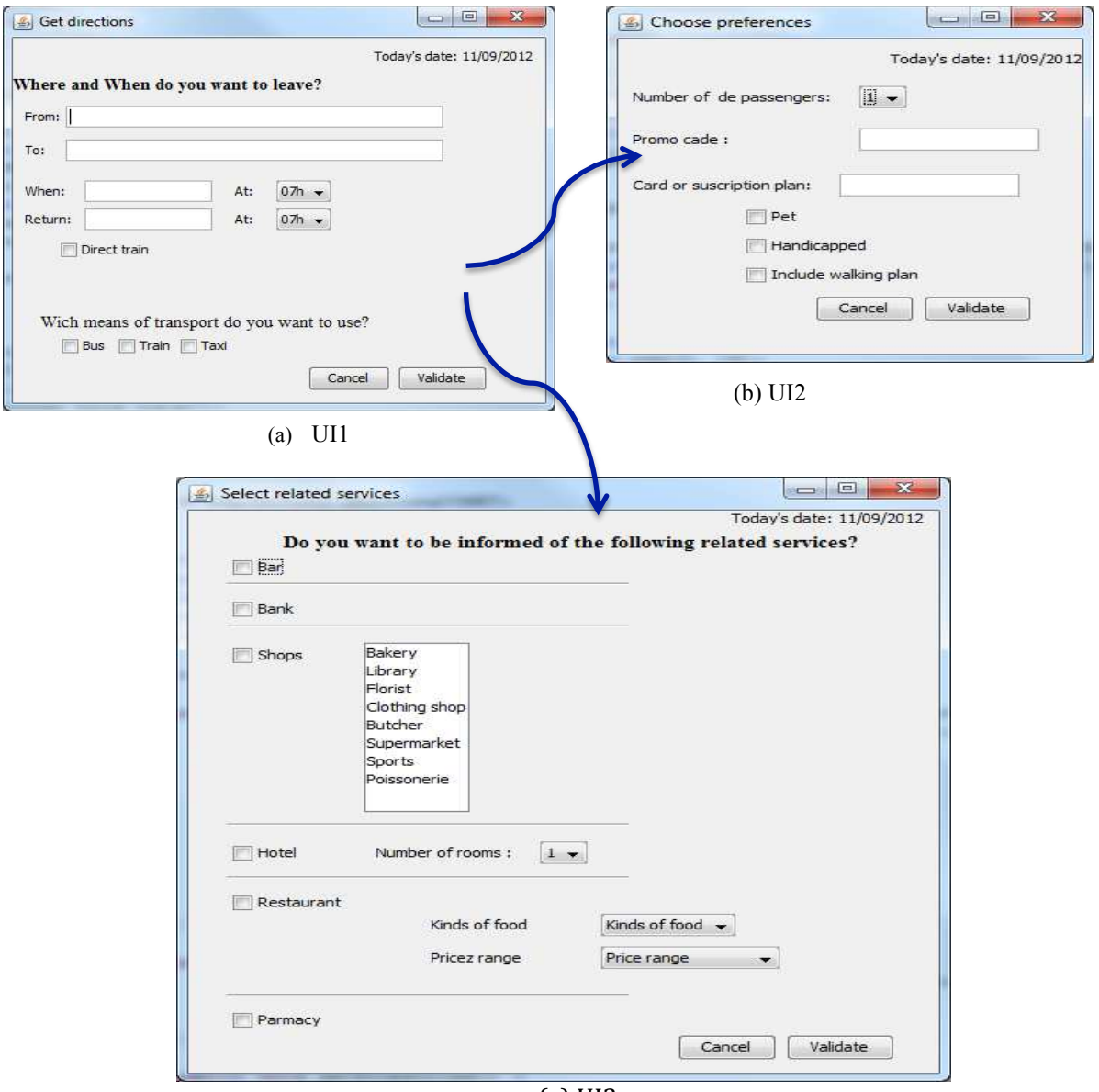
Our approach is to define objective measures, that is, measures that can be calculated automatically before UI composition and are independent of the subjective evaluation of a designer. Proposed measures are shown in Table 3.15. Measures 1 and 2 are based on criteria of Scapin and Bastien. Measures 3 to 9 are basic measures for the calculation of other measures (derived measures). Measures 6, 10, 11, and 12 are defined based on QUIM.

Table 3.15. Measures for information density

Measure	Definition/formula
M1. Memorizing rate	(Number of data that must be stored from one page-screen to the other) / Total number of data
M2. Mental calculus rate	(Number of data that require calculation of the user when it could be done automatically) / Total number of data
M3. Number of inputs	Number of fields where the user must enter a value (e.g. text field, list)
M4. Number of outputs	Number of fields where the software displays a value (e.g. resulting information as a result of user input).
M5.Number of labels	Number of entire labels, i.e., the complete sentence and not every word.
M6. Number of buttons/ icons	Number of buttons and/or icons with its label
M7. Number of pictures	Number of pictures used
M8. Number of screens	Number of screens used to perform user task
M9. Size of the screen	Size of each screen used to perform user task
M10. Density rate	(Nb inputs + Nb outputs + Nb labels + Nb buttons/icons + Nb pictures) / Nb screens
M11. Global density	Used space / total space
M12. Uniformity	It indicates how the visual elements of the user interface are placed evenly between page-screens. It corresponds to standard deviation of densities of each screen.

We performed a case study as an illustrative example for the application of those measures. Consider the following scenario: Yoann lives in Valenciennes. Its main objective is to find a travel that could be viewed on a map. Options are needed to drive research and others are selectable by Yoann for booking the travel he wishes (locomotion, number of people, etc.). The system should be able to advise if he wants to access a service (doctor, hotel, etc.).

These UI allow Yoann to get direction of his travel (UI1, Figure 3.13a), choose preferences as the number of passengers, etc. (UI2, Figure 3.13b), and select related services as doctor, bakery, etc. (UI3, Figure 3.13c). We suppose that the UI of travel planning system useful to Yoann can be composed from these three existing UI with three different composition operators. The first operator composes UIs in the same frame (Figure 3.14). The second and the third operator compose UIs respectively in tabs (Figure 3.15), and in sequence (Figure 3.16).



(a) UI1

(b) UI2

(c) UI3

Figure 3.13. UI 1 to get direction, UI 2 to choose preferences and UI3 to select related services

For the designer at design time or for an adaptive system at runtime, it is an important issue to choose the best composition to produce the UI for the travel planning system. To address user needs (find a travel), using these composition operators and the evaluation of measures, we evaluate the three potential composed UIs and choose the best one considering the information density criterion.

From the three UI to be composed, three operators of composition have been applied considering the spatial and temporal relationships. The first composition is computed by the operators *{orderIndependance, meet}* with an option for the vertical placement of components (corresponding to the union operator (Lepreux *et al.*, 2006; Pinna-Dery *et al.*, 2003)). With these operators, all information are made in a single screen, duplicate information (in this example: today's date) were placed only once (Figure 3.14). The second composition is computed by the operators *{interleaving, equals}*. It provides tabs composed of three screens corresponding to the initial UI (Figure 3.15). The third composed UI *{enabling, covers}* provides a sequence of three UI components (Figure 3.16). Note that the final composite UI is generated according to selected operators with the tool composition called COMPOSE (Gabillon *et al.*, 2011) that composes UI dynamically.

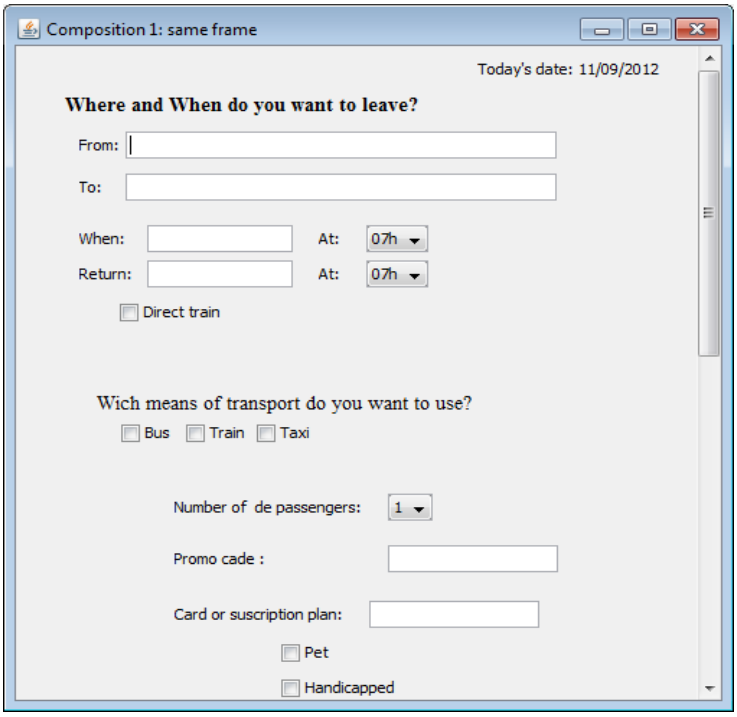


Figure 3.14. Composition 1 {orderIndependance, meet}: in the same frame.

We applied the proposed measures to these three compositions to compare their quality according to the information density criterion.

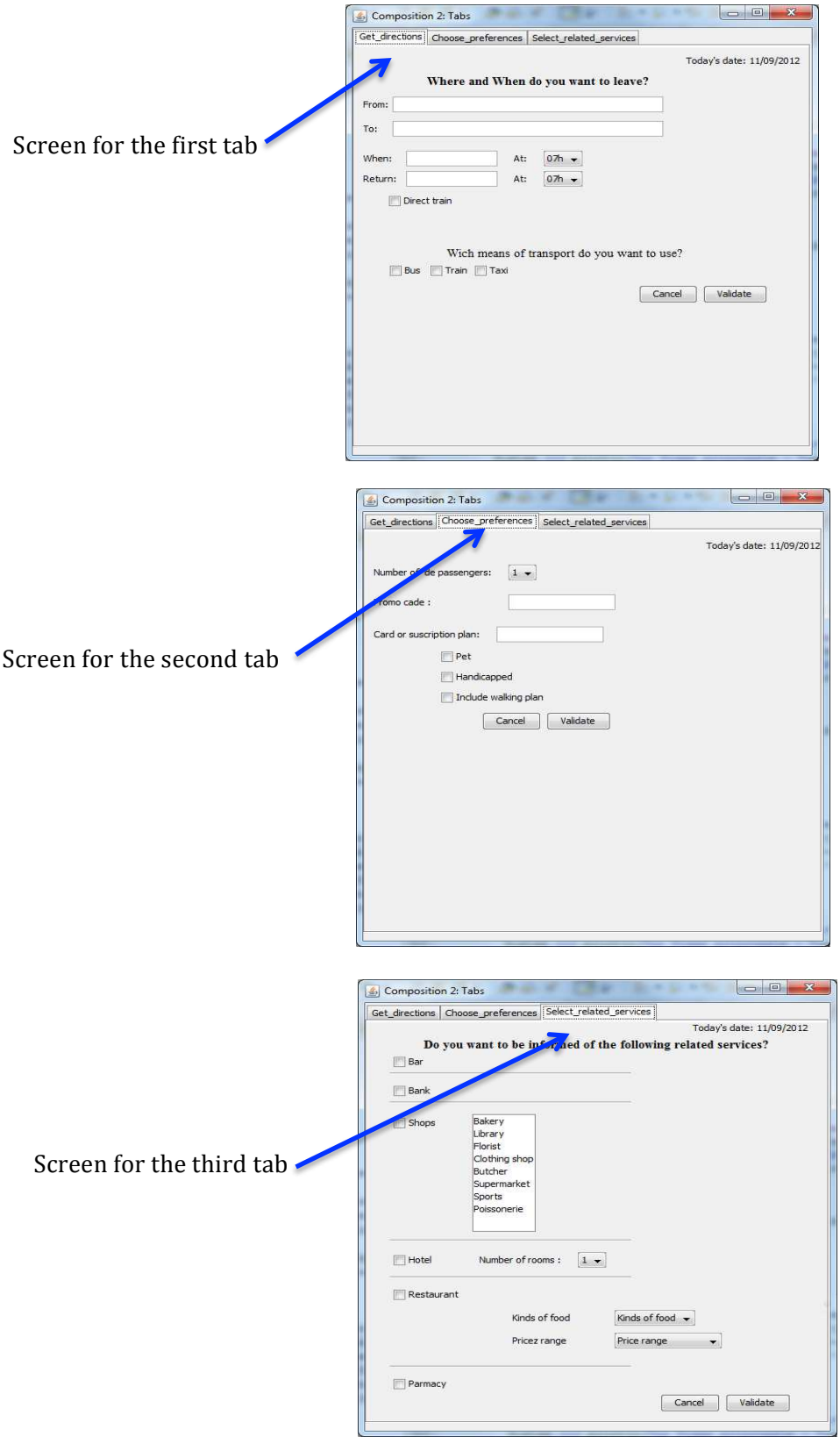


Figure 3.15. Composition 2 {interleaving, equals}: a) UI'1, b) UI'2 and c) UI'3 in tabs

Figure 3.16. Composition 3{enabling, covers}: a) UI'1, b) UI'2 and c) UI'3 in sequence

Table 3.16 presents the results of measurements for the three possible compositions. In the first interface composed (Figure 3.14), we observe that since it has a lot of information, we must consider the need for navigation on the screen (vertical scrolling). So, there are two main ways to measure this composed UI: considering (1) the number of information visible for each navigation, or (2) all information on a screen. Since it is difficult to anticipate the number of required navigation and their results, we chose the second way, that is, measures

are performed for all information available on a screen. As consequence, in Figure 3.15, we considered a single screen: local density corresponds to the global density (there are no measures of intermediate screens like the other compositions). For the other two compositions, measures of each screen are computed (thus three UI to measure for each composition) and then for the composed one (the sum for the base measures and the average for the derived ones). Thus, global measures correspond to meaningful measures for the user task (find a travel).

Table 3.16. Proposed measures applied to the three composed UI of the case study

M.	Comp.1	Composition 2				Composition 3			
		UI1'	UI2'	UI3'	Global task	UI1'	UI2'	UI3'	Global task
M1	0	0	0	0	0	0	0	0	0
M2	0	0	0	0	0	0	0	0	0
M3	26	10	6	10	26	10	6	10	26
M4	2	2	0	0	2	2	0	0	2
M5	29	13+3=16	7+3=10	11+3=14	28	13	7	11	31
M6	2	2	2	2	6	2	2+1= 3	2+1= 3	8
M7	0	0	0	0	0	0	0	0	0
M8	1	1	1	1	3	1	1	1	3
M9	521954	265468	265468	265468	796404	159960	102400	265468	527828
M10	59	28	16	24	58/3=19,3	27	15	21	67/3=22,3
M11	18,28	13,47	8,27	17,78	13,17	22,26	23,25	18,48	20,58
M12					4,76				2,52

As the result of our case study, we identified that the composition 2 is the best one because it has the lowest global density percentage (composition 2=13.17%, composition 1 = 18.28% and composition 3= 20.58). However, this finding is not completely generalizable, because it depends on the number of UI to be composed. Indeed, we note that in the case where the number of UIs is important, the number of labels will be more important and will involve an increase of the corresponding measure; then, it will not impact the other compositions (that means, the stability of the other measures). Composition 1 does not add information (label or button), it just increases the size of the screen. Composition 3 does not increase the need of information except for the navigation buttons (next and previous).

Note that the measurement of information density of the composition 1 gives a good result in placing all information on the same screen that is enlarged according to the number of information. Adapting the screen size is proportional to the amount of information preserves the measure of this criterion. However, it generates increased perceptive users' workload (criterion not studied here). We remember that information density criterion is one of 8 criteria proposed by Scapin and Bastien (1997). These criteria are interrelated; an individual one cannot allow the choice of the best composition in absolute.

Concerning the composition 1, as we said earlier, we have chosen to consider all the information on the total size of the screen. The user should navigate to access all the information and it could be interesting to consider subsets of information and a number of different screens. We justified this choice by the fact that it is difficult to know automatically which is the number of navigations and related information. However, a study on this issue would be interesting to measure the global density of the composition.

Measurement of uniformity allows knowing the standard deviation between the densities of individual screens. This can be noted in the composition 2, where the density is lowest because the three screens provided and presented in tabs have the same size. This final size is

the size of the largest initial UI to be composed. Thus, the global density measurement of UI'1 and UI'2, and composed UI are decreased. While in the composition 3, screen sizes are independent and thus the densities remain close to initial densities.

Moreover, we do not want to remove measures 1, 2 and 7 because they seem important for this criterion, even they were not completely illustrated by the case study. Other examples could be analyzed involving this information to conclude on their impact on information density. A better detail of this study can be found in (**Gabillon *et al.*, 2013**).

It is important to emphasize that we are aware that criteria are interrelated and should not be evaluated only individually. Therefore, we are now working on the definition of measures for other ergonomic criteria.

3.3.5 Quality Evaluation of Human-Computer Interaction

Several method, tools and quality models have been proposed over the years for user interface evaluation. Holzinger (2005) summarized the methods in two categories: inspection methods (without end users) and test methods (with end users). Inspection methods include for instance: heuristic evaluation (Nielsen, 1993), cognitive walkthrough (Polson *et al.*, 1992; Mahatody *et al.,* 2010), and action analysis (Card *et al.*, 1983). Some of the most common test methods are (Dumas and Fox, 2009; Holzinger, 2005): thinking aloud, field observation, and questionnaires. All these methods generate qualitative and/or quantitative data (see Chapter 2, § 2.2.3) that are usually presented individually requiring an effort of the evaluator to analyze them in an integrated way. For instance, we can have as a result of a questionnaire that the users found the evaluated system "not very simple" to use (selected from an ordinal scale); and in the effort spent to execute the tasks; both results presented separately. We argue that both data are complementary for the analysis of the HCI quality and integrating both results in a single view can facilitate the decision-making.

To that end we start a research that proposes the use of indicators concepts defined by the ISO 15939 standard (see Chapter 2, § 2.2.3) for providing a basis for decision-making in HCI evaluation. Qualitative and quantitative data are respectively extracted from highly cited HCI quality questionnaires and from existing tools.

Before specifying indicators we had to identify which quality and quantitative data could be coherently integrated. Therefore, we carried out a deep study for defining a mapping that combines qualitative with quantitative data based on the evaluated quality criteria. These criteria are related to both ergonomic and functional aspects of interactive systems. We adapted respectively Bastien and Scapin ergonomic criteria to evaluate the user interface ergonomic quality (Scapin and Bastien, 1997), and the ISO/IEC 9241-11 (1998) criteria to evaluate functional aspects of interactive systems. This mapping model allows specifying the required attributes for the construction of each indicator. It links each qualitative data with the appropriate quantitative data in a complementary manner. Figure 3.17 depicts the proposed mapping model.

As the proposed mapping integrates and synthesizes the data issued from different evaluation tools, it exploits different tools that have been developed to facilitate and to automate a variety of quality evaluation aspects (i.e. the functional and ergonomic aspects).

The tools related to the qualitative data are the classical questionnaires of users' satisfaction. Based on a broad literature review, we have selected the CSUQ (Computer System Usability Questionnaire) questionnaire (Lewis, 1995) among 23 of the most known and the most validated tools. The choice of CSUQ compared to the other questionnaires is due to the fact that it is a successful record of practical and academic applications in its original

version supporting an applicability in usability evaluation of computer systems in various areas as well as in research into the measurement of the construct of usability (Erdinc and Lewis, 2013; **Assila *et al.*, 2014**). We used as qualitative data some items of CSUQ.

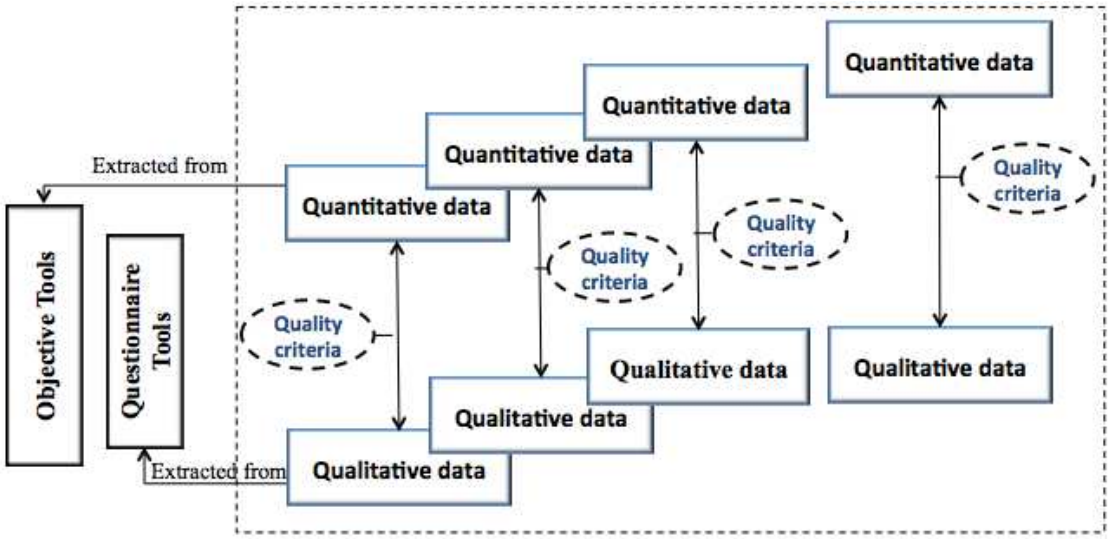


Figure 3.17. The proposed mapping model

To perform a quantitative evaluation, we chose two tools respectively related to the functional and ergonomic aspects of interactive systems interfaces. These tools are:

- EISEval electronic informer (Environment for Interactive System Evaluation) (Tran *et al*, 2013) - it is a generic and configurable electronic informer used for agent-based interactive systems evaluation. It automatically captures and analyses HCI information such as user actions and their consequences on the interactive system. It generates different attributes such as the rate of completion of correct task, time of task execution, etc.;
- The ergonomic guidelines inspector (Charfi *et al.*, 2014) - it aids for ergonomic quality evaluation of interactive systems. It evaluates the ergonomic consistency of a HCI according to ergonomic guidelines in order to detect the ergonomic inconsistencies. It is based on different graphical components attributes such as the writing size, the writing color, the informational density, etc. After parsing the interface graphical controls attributes, it provides the evaluator with a set of ergonomic recommendations and inconsistencies.

Table 3.17 illustrates some examples of the obtained mapping results of qualitative (items from CSUQ) and quantitative data (from EISEval) extracted from the specified evaluation tools.

Based on the measurement information model described in Chapter 2 (§ 2.2.3), we are defining several indicators. In this work, each indicator is composed of two measures that produce qualitative and quantitative data, previously defined in the mapping. Figure 3.18 shows an extract of the specification of an indicator with their related measures. For each indicator, a specific analysis model is being defined to support the decision-making as defined by ISO/IEC 15939 (2007) (see Chapter 2, § 2.2.3). To that end, several experimentations will be performed.

This research is under development as a doctorate at the University of Valenciennes. Preliminary results can be found in **Assila *et al.* (2013, 2014)**.

Table 3.17. Examples of the mapping between qualitative and quantitative data

Quality criteria	The mapped data	
	Qualitative data Item from CSUQ questionnaire (1) Strongly agree -- (7) Strongly disagree	Quantitative data Data from EISEval
Effectiveness: ISO 9241-11 criterion	It is **simple to use** this system	The rate of completion of each task
Efficiency: ISO 9241-11 criterion	I am able to **complete** my work **quickly** using this system	The rate of completion of each task Execution time of each task
Informational density: Bastien and Scapin criterion	It is **easy to find** the information I needed	Informational density per interface Images density Components dimensions Objects density
Guidance (readability): Bastien and Scapin criteria	The information provided with this system is **clear**	Writing font Writing size Writing color

Information Need	Judge the quality of an HCI relatively to the effectiveness quality criterion
Attributes	Tasks to be realized for a session It is simple to use this system (Strongly agree 1 2 3 4 5 6 7 Strongly disagree)
Base Measures	The value of the completion of each task for a session The tasks number to be executed The question response (Resp) The users number
Measurement Method	It is a quantitative measure extracted from the EISEval tool. It is a Boolean measure that can be equal either 0 or 1. It is a quantitative measure extracted from the EISEval tool It is a qualitative measure that quantified the user response using CSUQ questionnaire It is a quantitative measure that indicates the number of users involved in the evaluation
Derived Measures	The rate of completion of correct tasks per user (RTT) The rate of users responses per question (Rresp)
Measurement Functions	Divided the sum of the value of the completion of each task for a session per user by the total tasks number to be executed For each question, calculate the number of users who chose the same response and divided it by total users
Indicator	Effectiveness of correct completion of tasks in a HCI. This indicator combines two measures (the RTT derived measure and the Resp base measure).

Figure 3.18. Extract of indicator specification

3.4 Conclusion

This chapter has presented several works performed in search of quality of software product and process. In the context of software process, our main contribution is related to the exploitation of the whole software process quality cycle, from its definition to its improvement. Our approach for software process definition presented previously was later largely used in several companies in Brazil. We can quote, therefore, the following main contributions:

- Definition of procedures about how to define a standard software process and how to instantiate it to be used in a specific project. These procedures take into account that before instantiating to a project we can specialize the software process considering the type of software and software development paradigm (object-oriented software, web, etc.);
- The definition of measures for software process evaluation applied in practice;
- The large study of measures for software process evaluation and improvement. This study investigate more than 500 measures published in literature and synthetize them into a set of the most used and applied measures (15 base measures and 37 derived measures);
- The definition of a catalog of 15 indicators with a detailed specification for the application, evaluation and interpretation of the results to support process performance analyses.

Regarding the quality of software products, our research explored mainly the definition and application of quality software measures for different types of software (web systems, legacy systems and ubiquitous systems). To that end, the Goal-Question-Metric method was used in different contexts. GQM proved to be a suitable methodological approach, easy to use and useful for defining and interpreting outcome measures. Our main contribution was to attest that the definition and use of measurements is not only feasible but also that measurements are a real and good support for decision-making. In all studies the use of measures support some kind of decision, from the simple selection of web sites for education training, till the decision support for maintenance sub-contracting or the dynamic selection of UI composition. We can quote, therefore, some specific contributions:

- The definition and application of an approach to evaluate legacy systems to better define the contracts;
- The definition and application of an approach for monitoring the deterioration of information systems;
- The definition of an approach based on measurement to support the choice among different user interfaces compositions while composing services;
- The evaluation based on measurements for different kinds of applications (web-based systems, ubiquitous systems, information systems).

Nevertheless, those experiences also reveled some difficulties and limitations related to software quality as follows (**Oliveira *et al.*, 2012**):

- Large number of measures of process – in (**Monteiro and Oliveira, 2011**) more than 500 different measures for software processes were identified. We also find a lot of papers and measures propositions while defining measures for the evaluation of the different quality products. Choosing the right measure(s) in each situation is not an easy task;
- Difficulty and complexity of experimental protocols definition for performing experimentation - experimental protocols are generally complex and difficult to

define: how many participants are necessary? What should their profile be? What tasks should they perform for an evaluation? How to limit bias? etc.;

- Unavailability of professionals – once the protocol is defined, the measures should be collected for the quality evaluation. Thus, we need to juggle with the availability of personnel and costs for its realization. For a valid experiment, all bias should be avoided. However, the unavailability of professionals and the difficulty to involve them in experiments and collect their evaluation, lead usually to carry out the experiments in "laboratory", with students whose instructors are those who have proposed the approaches to validate, which results therefore in several limitations of the study;
- Interdependency of quality characteristics - the characteristics/measures of quality are usually interdependent. A classic example concerns the dimensions of completeness and readability. The more complete (detailed) is a process description, the more, in general, complex it is, which often leads to degrade its readability. These dependencies are problematic for the interpretation of results (you have to understand what measures are interrelated and why) and the choice of improvement actions to be taken (an improvement action to improve a measure should not lead to in degrade another one); and,
- Difficulty for collecting and interpreting measurements - collection procedures must be integrated into the software process. If the measures are computed in an automatic way or if the organization uses some kind of process management support tool, this integration is easier and more reliable than when the measures are collected by manual activities (e.g. in paper forms of inspection meetings). Moreover, historical data are needed so that the measures are valid and really useful. Finally, once they are collected, their interpretation often depends on an experts' analysis. Our experience shows that the results are often approximate and the thresholds are not enough, because the context of the project (size, team development, culture, pressure calendars, etc.) has a strong influence for the final interpretation.

Those difficulties and limitations open new research perspectives that will be presented in the last chapter of this document.

Chapter 4 Applying Knowledge Management Issues for Software Quality Assurance

4.1 Introduction

In the previous chapter several works about software quality assurance were presented. All of them were enriched by practical case studies performed, usually, in industrial contexts. These works deal with the definition and application of measurements and the use of methods. They can be summarized considering three main aspects:

- Quality evaluation of software process (with measures to evaluate software process and perform process performance analysis);
- Maintainability evaluation of legacy systems;
- Quality evaluation of software products with regard to several aspects related to usability (i.e., accessibility of web applications; context-awareness of ubiquitous systems since it impacts on the human-computer interaction, and information density of user interfaces);

The previous experience with the expert system development (from 1994 to 1996), my doctoral research and the emergence of the knowledge management discipline, motivated me to work about how to use knowledge management principles to get better quality of software product and process. Continuing in the same direction of the research presented in the previous chapter, we focused on the three aspects mentioned previously: quality of software process, maintainability and usability of software systems.

As stated in chapter 2, software process maturity models defend that it is necessary for software process improvement, to capture lessons learned in the projects to better plan new projects, and therefore, get better process. To address this need, knowledge management issues have been largely used (see for example, the literature review by Bjørnsonand Dingsøyr (2008), and the research on process asset libraries of Garcia *et al.* (2011)). Following this trend, we used two knowledge management techniques (*post-mortem* analysis and learning history introduced in Chapter 1, § 2.5) to capture lessons learned in software projects with the goal of improving the software process.

According to ISO/IEC 25023 (2011), to evaluate maintainability implies "to measure the degree of effectiveness and efficiency with which a product system can be modified by intended maintainers". Believing that the more we know about the maintenance being done, better effectiveness and efficiency will be achieved, we performed a research to identify which knowledge is needed for software maintenance. After that, we organized this knowledge in a domain ontology to support knowledge capture in future maintenance projects.

Finally, we have investigated how to use the domain knowledge to provide content personalization of user interface for a better efficiency and user satisfaction. This goal goes towards what is expected for usability evaluation, defined as the "degree to which a product or system can be used by specified user to achieve the specified goals with effectiveness, efficiency and satisfaction in a specified context of use" (ISO/IEC 25023, 2011).

Next sections present these different works where we explored practices from knowledge management domain in search of software quality. We chose to present these works chronologically, which means, in the order we developed them. In this way, section 4.2 presents the research to support software maintenance considering the organization of knowledge and the capture of knowledge in maintenance projects. Then, section 4.3 presents an approach for capture and dissemination of knowledge in software projects. Finally, in section 4.4, we present our later research of using knowledge to support the design of personalized user interface in order to obtain better usability. We conclude this chapter (4.5) with a discussion about our contributions and new research challenges.

4.2 Knowledge Management Issues to Support Software Maintenance Process

Software maintenance is a knowledge intensive activity. Maintainers need knowledge of the application domain, of the organization using the software, of past and present software engineering practices, of different programming languages (in their different versions), programming skills, etc. Concurrently to this knowledge need, a recurring problem of software maintenance is the lack of system documentation. Studies report that 40% to 60% of the software maintenance effort is devoted to understanding the system (Pfleeger, 2001, p. 475; Pigoski, 1996, p.35). This scenery motivated us to investigate which knowledge could be useful for software maintenance and we organized it in an ontology (§ 4.2.1). This ontology was later used to support knowledge capture of software maintenance (§ 4.2.2).

4.2.1 Identifying and Formalizing the Knowledge Needed for Software Maintenance

To achieve our goal of identifying what knowledge is used in software maintenance, we defined a process composed of three steps (Figure 4.1).

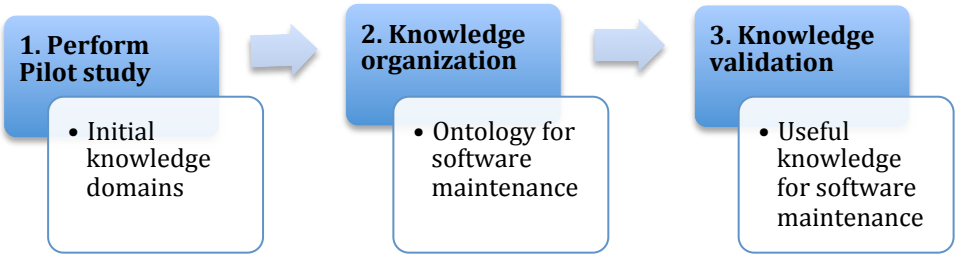


Figure 4.1. Steps for identification of knowledge for software maintenance

The first step was to observe software maintainers in their daily activities to get a first idea of what knowledge they use. In this pilot study, we tried to observe the software engineers with as little preconception as possible. Knowledge domains were not predefined but extracted from the results of the study. In the second step we formally modeled the knowledge needed for software maintenance using this pilot study, our experience and a literature review in a domain ontology (see Chapter 1, § 2.5.3). In the third step we validated this ontology with some field research to identify if the knowledge is really useful for maintenance. In this section, we present these steps.

In the pilot study (Figure 4.1 - step 1), we observed software engineers performing maintenance. We studied six software engineers in two different organizations (a bank and a public administration) during 13 sessions of 75 minutes on average. The sessions followed a protocol called think-aloud (Lethbridge *et al.*, 1996) where the maintainers were asked to say everything they did and why they did it. These sessions were recorded and later transcribed on paper to be analyzed. During the analysis, we tried to identify the kind of knowledge, which we called knowledge atoms that the software engineers were using at each moment. For example, from the session description presented in Figure 4.2, we concluded that: to perform the descriptions from 1 to 5 the maintainer knows the development environment s-he is working with; to know which method to modify (description 4) and what to remove (description 5) means the maintainer had some previous knowledge of the application implementation (i.e., the software system); and finally to start to write new code (description 6) means the maintainer knows the programming language. The "knowledge atoms" were then categorized in general knowledge domains.

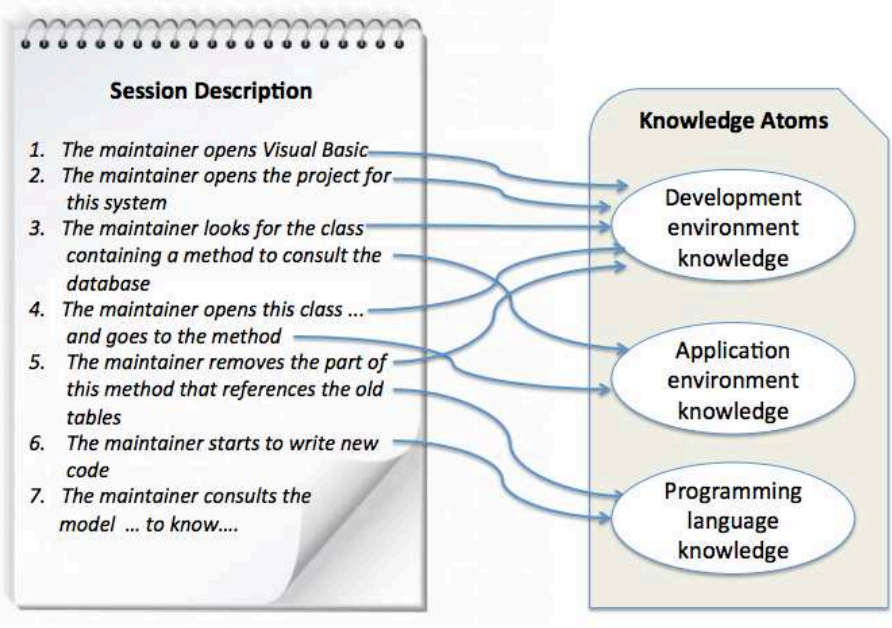


Figure 4.2. Knowledge identification

From the study of the 13 sessions we extracted the following domains of knowledge:

- Computer science knowledge - knowledge that typical software engineer would have; such as, knowledge about programming, about a specific programming language, software development/debugging, and design techniques;
- Business knowledge - knowledge that a typical member of the organization (not limited to computer scientist) would have; such as, about the application functionalities, the production environment of the system implementation, the problem of being analyzed for maintenance, and the organization itself;
- General knowledge - common sense knowledge that "anyone" could have; such as knowledge of foreign languages, web browser and email.

As already mentioned, we tried to start this pilot study with as little preconceptions as possible, and this classification was established bottom-up, grouping together knowledge items that were found in the sessions analyzed.

After this pilot study, we started a more formal approach. Based on the concepts identified in the pilot study, a literature review, and our own experience, we defined an ontology of the knowledge needed during maintenance (Figure 4.1 - step 2).

We started the ontology construction by looking for motivating scenarios where the knowledge captured would be useful. Some of those scenarios are: deciding who is the best maintainer to allocate to a modification request based on her-his experience of the technology and the system considered; learning about a system the maintainer will modify (which are its documents and components and where to find them); defining the software maintenance activities to be followed in a specific software maintenance, and also the resources necessary to perform those activities.

These scenarios induced us to organize the knowledge around five different aspects: knowledge about the *Software System* itself (also identified in the pilot study); knowledge about the maintainer's skills, that is *Computer Science Skills* (related to the computer science and general knowledge of the pilot study); knowledge about the maintenance activity, that is the *Modification Process* (not identified in the pilot study); knowledge about the *Organizational Structure* (related to the business knowledge domain of the pilot study); and knowledge about the *Application Domain* (related to the business knowledge domain of the pilot study). Each of these aspects was described in a sub-ontology. Figure 4.3 illustrates how the sub-ontologies combined in the general ontology. For each of the sub-ontologies we defined "competency questions", captured the necessary concepts to answer these questions, established relationships among the concepts, described the concepts in a glossary and validated them with experts.

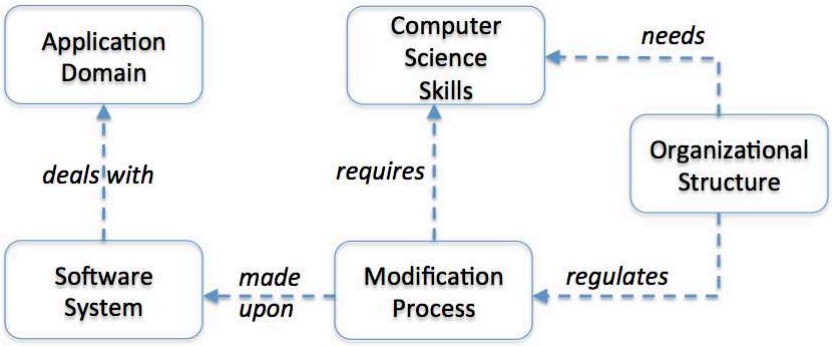


Figure 4.3. Sub-ontologies of the maintenance ontology (**Anquetil *et al.*, 2006**)

We will not give a detailed description of the whole ontology here (see (**Anquetil *et al.*, 2006**) for a complete definition), but we will focus on the *Software system* sub-ontology (Figure 4.4). The System sub-ontology is one of two sub-ontologies corresponding to the more computer science oriented knowledge. Knowledge about the system is also intuitively fundamental to software maintenance.

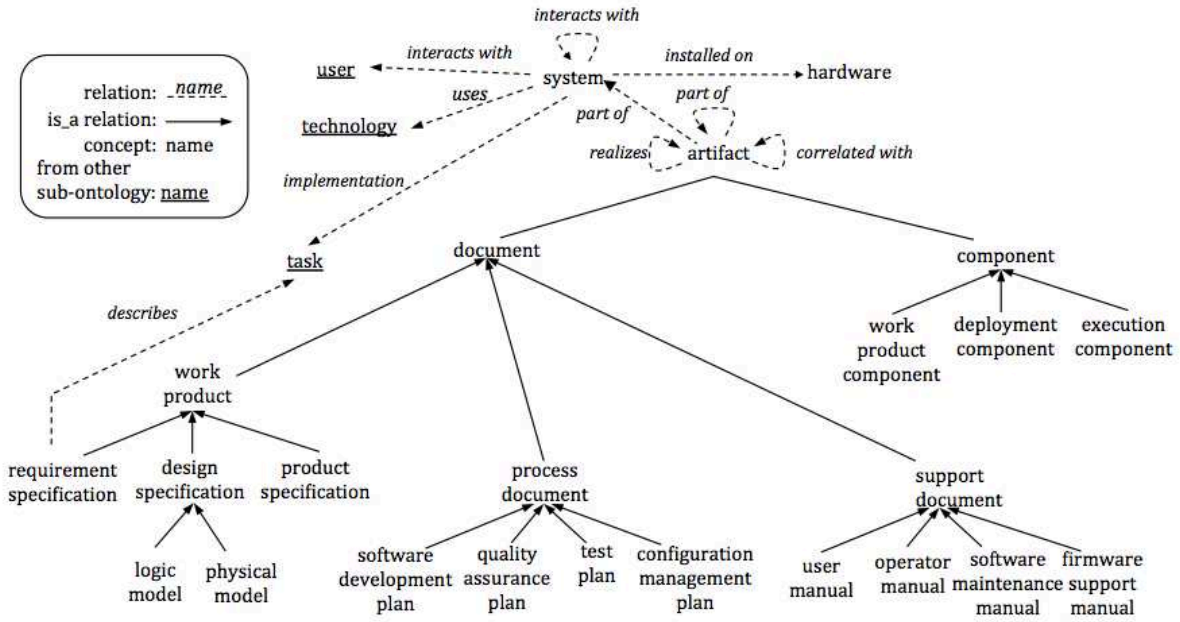


Figure 4.4. Software system sub-ontology (**Anquetil *et al.*, 2006**)

The competency questions for the System sub-ontology are:

- What are the artifacts of a software system?
- How do they relate to each other?
- Which technologies are used by a software system?
- Where is the system installed?
- Who are the software system users?
- Which functionalities from the application domain are considered by the software system?

Answering these questions led to a decomposition of the software system in artifacts, a taxonomy of these artifacts and the identification of the hardware where the system is installed, its users and the technologies that was used in its development.

The artifacts of a system can generally be decomposed in documentation and software components. Briand *et al.* (1994) considers three kinds of documentation:

- Work product, describing the system itself (i.e., software requirement specification, software design specification, and software product specification);
- Process documents, used to conduct software development and maintenance (i.e., software development plan, quality assurance plan, test plan, and configuration management plan);
- Support documents, helping to operate the system (i.e., user manual, operator manual, software maintenance manual, firmware support manual).

Considering that the software design specification proposed by Briand *et al.* (1994) should represent the behavior and structure of the system and that we can have different abstraction models, we refined the software design specification in logic and physical models.

Software components represent all the coded artifacts that compose the software program itself. Booch (2000) classify them in:

- Execution components, generated for the software execution;
- Deployment components, composing the executable program;
- Work product components, which are the source code, the data, and anything from which the deployment components are generated.

All those artifacts are, in some way, related one to the other. For example, a requirement is related to design specifications, which are related to deployment components. We call the first kind of relation realization, relating two artifacts of different abstraction levels. The second one is a correlation of artifacts at the same abstraction level. To express those constraints we defined a set of axioms (Figure 4.5 illustrates this one). In total we defined 21 axioms for the system sub-ontology and 53 for the whole ontology.

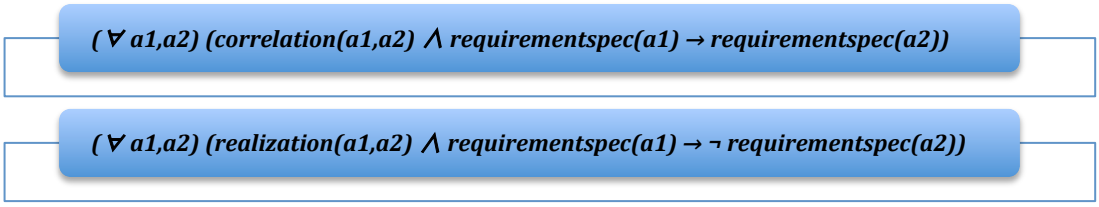


Figure 4.5. Example of formalization of an axiom

Other relations in this sub-ontology are: the software system is installed on some hardware, the user interacts with the software system, and finally, the software specification describes the domain tasks to be implemented (or functionalities).

In the last step (Figure 4.1 - step 3), we were interested in verifying if the knowledge organized in the ontology would be really useful for software maintenance. To do so, we instantiated the ontology with data from a real project; and, we performed observational studies with maintainers.

The ontology instantiation was performed from the documentation of a real software system. This documentation came from the development of the system as well as from past maintenances. As a result, 74% of the ontology concepts were instantiated. We consider this result acceptable since the other concepts were not specific for that project.

For the observational study, we realized two experiments:

i) Observing maintainers with the think-aloud protocol, as in the pilot study;

ii) Presenting the instantiated knowledge for a software system before the maintenance activity and then asking the software engineers what knowledge was used.

The think-aloud experiment (i) followed basically the same procedure as in the pilot study: the maintainers explained everything they were doing while maintaining a system, the sessions were taped, the tapes were transcribed, and finally, we identified and classified the knowledge atoms. The only difference from the pilot study was that this last step was based on the ontology. Two maintainers participated in this experiment, doing five sessions for a total of 132 minutes (26 minutes per session on average).

In the second experiment (ii), the ontology was presented and explained to the software maintainers and they were asked to fill in, every day, a questionnaire on the concepts they used. This form consisted of all the concepts we had instantiated previously and the list of their instances (as we identified them). The maintainers were simply asked to tick the instances they had used during the day. They could not add new instances. The experiment

was done with three maintainers and one manager. They filled 17 forms in 11 different days over a period of 10 weeks.

The results of these two experiments are given in Table 4.1. One may observe that there are a lot less concepts used in the first one than in the second. One reason for this is that there were fewer sessions in the first experiment and they were mostly short punctual maintenance (average of 26 minutes compared, for example, with the 75 minutes per session in the pilot study).

All the concepts detected in the first experiment were also found in the second one, it did not bring in any new instances. However, six concepts previously instantiated were not found here:

- Analysis Technique and Requirement Specification Technique (Skills sub-ontology) were not used because the maintenance operations were relatively simple and restricted to small modification to the source code. Therefore, no high level analysis was required;
- For the same reason, the Requirement Specifications (System sub-ontology) was neither studied nor modified;
- The creation, modification and distribution of all Support Documentation (including User Manual and Operation Manual, the three of them being in the System ontology) were under the responsibility of another unit; therefore the software engineers we studied did not know about them or use them.

Table 4.1. Number of concepts used in two studies (**Anquetil *et al.*, 2003**)

	Ontology	Think-aloud		Questionnaire	
	#	#	%	#	%
Computer science Skill	38	15	39%	26	68%
Application domain	4	1	25%	2	50%
Modification process	30	16	53%	23	77%
Software System	23	9	39%	13	57%
Organizational Structure	3	2	67%	3	100%
Total	98	43	44%	67	68%

Details of each part of this research can be found in several publications (**Anquetil *et al.*, 2007; Anquetil *et al.*, 2006; Anquetil *et al.*, 2003; Dias *et al.*, 2003**).

4.2.2 Capturing Knowledge for Maintenance Using Ontologies

We argue that software maintenance is a knowledge intensive activity. As presented in the previous section, different kinds of knowledge are required for performing a maintenance. One could argue that software development suffers from the same knowledge needs, however these needs are more difficult to be answered during maintenance. For example, it is not uncommon in software maintenance to have a very vague knowledge of what were the exact requirements for the system, whereas during software development, one is expected to have access to the requirements relatively easily. For example, in the study presented in chapter 3 (§ 3.2.1) we noted that the documentation of the ten systems, if any, was outdated and incomplete. Besides this, it is common that the maintenance is done by people that do not develop the system. According to Pigoski (1996) 40% to 60% of the software maintenance effort is devoted to understanding the maintained system. For this reason, we decided to perform knowledge capture in software maintenance project not only for gathering lessons

learned to improve the software process (which is the traditional use in software engineering), but also to discover knowledge about the software system itself to help in future maintenances.

To address this problem we adapted Post-Mortem Analysis (PMA) (presented in Chapter 1, § 2.5.2.1), a knowledge externalization tool usually used for software process improvement at the end of software projects.

Thus, to adapt PMA as an externalization technique in maintenance projects, we need to define how to uncover not only knowledge on the maintenance process (e.g. how it was executed, what tools or techniques worked better), but also to register knowledge on the system maintained (e.g. how subsystems are organized, or what components implement a given requirement). This section presents how we adapted PMA and its application in project from industry.

4.2.2.1 PMA for software maintenance

To define this new PMA approach, we had to consider three important aspects (**Anquetil *et al.*, 2006; Souza *et al.*, 2004**):

i) When to insert PMA during the execution of a typical maintenance process;

ii) What knowledge we should look for;

iii) How to extract this knowledge from the software maintainers.

To address the first aspect ((*i) when to insert PMA during the execution of a typical maintenance process*), we had to have in mind that software maintenance projects may be of widely varying size, they may be short in the correction of a localized error, or very long in the implementation of a new complex functionality, or correction of a very diluted problem. For small projects, one may easily conduct a PMA at the end of the project without the risk of loosing (forgetting) important information, but for larger projects, it is best to conduct several PMAs during the project (as proposed by Yourdon (2001)) so as to capture important knowledge before it becomes so internalized in the participants' mental models that they cannot clearly remember the details.

To identify the points, in a software maintenance project, where we could perform PMA, we used the ISO/IEC 14764 (1999) maintenance process that is composed of the following phases: process implementation, problem and modification analysis, modification implementation, maintenance review/acceptance, migration, and software retirement. Modification implementation is, usually, the longest phase, including tasks to implement the modification such as requirements analysis, architectural design, detailed design, coding, and testing. Therefore, we set three points to perform PMAs (see Figure 4.6):

- After the analysis of the modification, which includes the first two activities (Process implementation, Problem and modification analysis) and the initial task of the third activity (Modification implementation: requirement analysis);
- After the implementation of the modification, which includes the rest of the third activity (Modification implementation);
- At the end of the project in order to review all its aspects and the most recent activities not yet considered in the intermediary PMAs.

To address the second aspect (*(ii) what knowledge we should look for*)), we identify that depending on the specific scope of each PMAs, we may hope to explicit different kinds of knowledge. For example, information on the testing techniques used will be best discovered during the second PMA (post-implementation) just after the tests have been performed. We

recall that we wished not only to discover lessons learned from the execution of process activities but also to discover new knowledge learned on the system, its application domain, or other issues not related to the maintenance process.

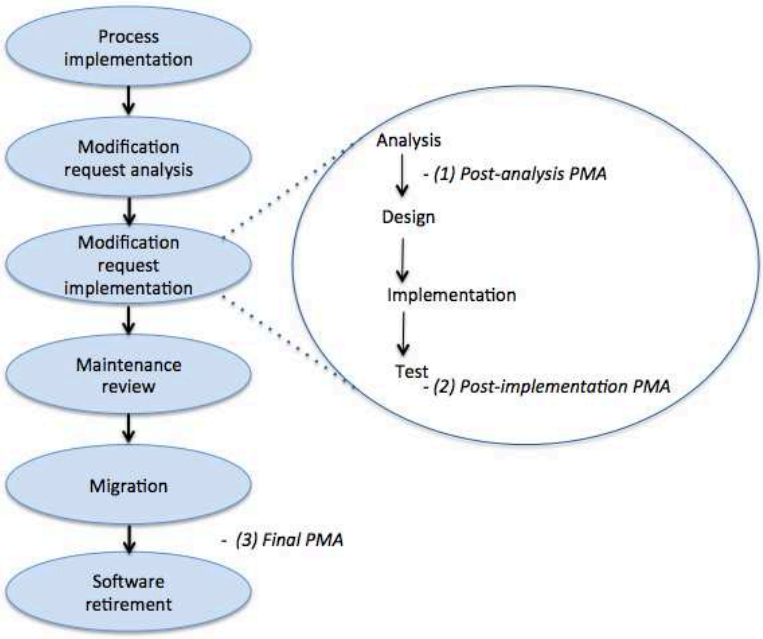


Figure 4.6. ISO14764 maintenance process with the intermediary (1 and 2) and final (3) PMAs (**Anquetil *et al.*, 2006**)

To identify what information we could hope to discover in each PMA, we mapped the concepts defined in our ontology for software maintenance (presented in § 4.2.1) to the particular tasks reviewed in each PMAs. For example, the first PMA (post-analysis) occurs after the following activities: the process implementation activity, modification request analysis activity, and the requirements analysis task from the modification request implementation activity. Let's take as an example the first activity: process implementation.

In the *process implementation* activity, the only task typically performed for each new maintenance project is to develop plans and procedures for conducting the activities of the project based on the modification request. To execute process implementation tasks the project manager usually takes into account his/her experience from previous projects with a similar domain, size, and team. Therefore, the type of knowledge related to this activity is about the process execution, what maintenance activities are needed and may be what specific technology will be required (concepts from the software maintenance ontology). This implies that the first PMA should look for this particular type of knowledge, named details on the modification request (see Table 4.2). We did the same analysis for all tasks reviewed in each PMA. We arrived to the set of type of knowledge for each PMA presented in Table 4.2.

Finally, to address the third aspect *((iii) how to extract this knowledge from the software engineers*), we elaborated the questionnaires that should be distributed among the software engineers that will participate in the PMA session. The questionnaires are composed of one or more questions for each type of knowledge identified for that PMA. Questions are designed to instantiate the concepts defined in our ontology.

Figure 4.7 shows some questions from the PMA post-analysis questionnaire. In this example, application domain questions are intended to instantiate the Application domain

sub-ontology concepts. There are two possible uses of the questionnaire. First, it may be used only to revive the memory of the PMA participants on the various types of knowledge sought. Second, the answers to the questionnaires are used to help the facilitator to focus the PMA session on the topics that appear most likely to bring new knowledge.

Table 4.2. The three maintenance PMAs and the types of knowledge they focus on

PMA	Type of knowledge
(1) Post-analysis	Details on the modification request Organizational structure using the software Options for implementing the modification Effort estimation for the modification Negotiation of time frame to do the modification Documents modified Requirement elicitation technique used Tools used Application domain Details on the requirements
(2) Post-implementation	Programming languages and tools used Programming techniques used Software components modified Systems interrelationship Analysis/design inconsistencies Re-engineering opportunities detected Artifacts traceability Database design Design patterns used Testing technique used Process and support documentation modified
(3) Final	Negotiations with other technological departments Modification monitoring Maintenance process

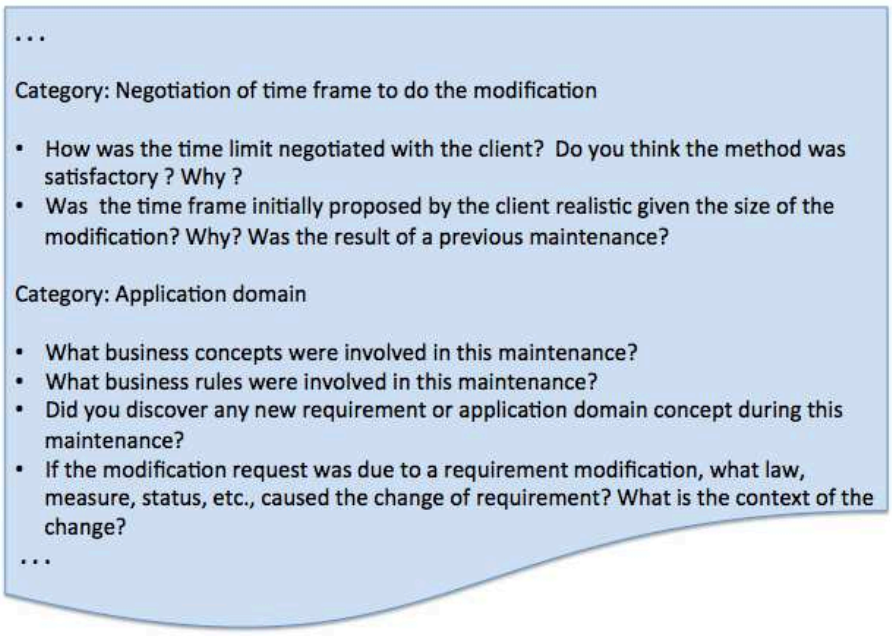


Figure 4.7. Extract of the post-analysis questionnaire

4.2.2.2 Applications in industry

This approach of PMA for maintenance was applied to six maintenance projects from a public organization in Brazil: four small maintenance projects (about 1 man/week work), and two larger projects (more than two months). The methodology was tested on a specific group of 15 people, responsible for the maintenance of 7 legacy software systems.

During and after each project, questionnaires were distributed to the maintainers to revive the important points of the projects in the mind of the participants and help them focus on the topics of interest. For the small maintenance projects the PMAs facilitator met each maintainer to interview him/her. For the two large maintenance projects with more maintainers involved (more than 10 in our experiments), half-day meeting sessions were organized to capture positive and negative points about the maintenance (the lessons learned).

In Table 4.3, we present an overview of the number of concepts that could be instantiated during the PMAs; that means the knowledge acquired about the software systems. We note that:

- From the 23 concepts in the Software System sub-ontology, 11 were instantiated with many instances (80), which means that at least one concrete example of these concepts was mentioned during the PMAs as something that was learned and worthy of remembering;
- Because of the typical conditions of legacy software systems (foremost the lack of system documentation), many concepts from the Software System sub-ontology could not be instantiated. This is the case of many document sub-concepts (there are 16 in the sub-ontology);
- The Process sub-ontology is the one that was the most instantiated, in number of concepts (21) and number of instances (135);
- All concepts from Application Domain and the Organizational Structure sub-ontologies were instantiated and, more importantly, many instances were found, especially in the case of the application domain sub-ontology (68 instances). This is a good result since application domain knowledge is considered very important by some authors (e.g. Biggerstaff *et al.,* 2002);
- Very few concepts were instantiated for the Computer Science Skills sub-ontology. It is natural that experienced software engineers discover less new knowledge about computer science techniques or CASE tools, and we do not see this as a problem with our approach. Computer science skills are considered background knowledge that all software engineers should have.

Table 4.3. Captured knowledge about the system: instantiated concepts from the ontology

Sub-ontologies	Number of concepts	Instantiated concepts	Number of instances
Software System	23	11	80
Modification Process	30	21	135
Computer science skills	38	05	09
Organizational structure	03	03	22
Application domain	04	04	68

Some examples of instances captured with the PMA are:

- Business knowledge instantiated as concepts in the Application Domain sub-ontology. For example, for one of the system that deals with monitoring of government educational projects, some of the instantiated concepts were about the publication and execution of agreements, and extension of the expiry date, including business rules of how to perform these extensions;
- Update of some use cases in all maintenance projects (in Software System sub-ontology);
- Technical details of implemented components (in Software System sub-ontology). For example, for the same system of monitoring of government educational projects, some tables in the database (example, the one with data about project date estimates) was described and all their components (code source, files, specification) were associated in the same component;
- Identification of who asked the modification, who participated in the maintenance of a specific requirement, and which requirements were changed (in Modification Process sub-ontology).

Figure 4.8 presents some examples of lessons learned, that is, the knowledge acquired about the maintenance process.

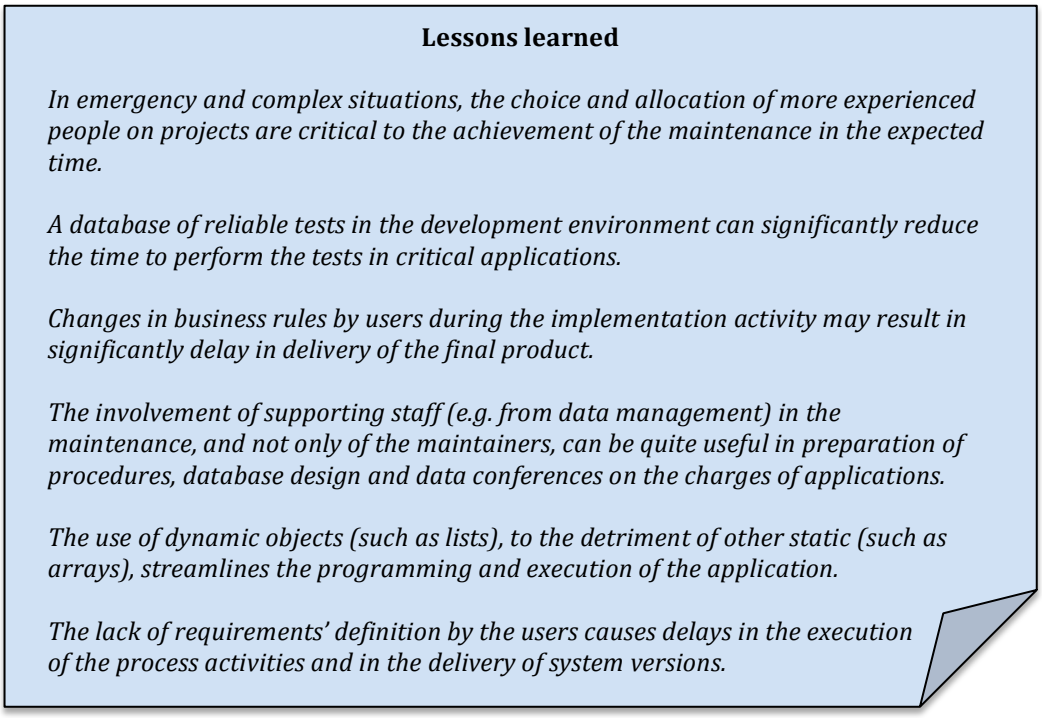


Figure 4.8. Captured knowledge about the software process execution

As a result of those experiences, the organization made some improvement, such as:

- Redefinition of people roles in the project teams;
- Increase the involvement of some internal areas;
- Correction of the reference documents used by technicians;
- Improving the process for approving applications;
- Conducting trainings to correct deficiencies;
- Reorganization of the teams' responsibilities.

More details of this study can be found in (**Anquetil *et al.*, 2006, Souza *et al.*, 2004**).

4.3 Capturing and Disseminating Knowledge with Experience Factory

It is well known that to produce good software systems using the adequate resources and cost is a difficult task. Appropriate effort estimate is an important part of this task. It is particularly important to the organizations, because too high an estimate may result in loosing a contract to a competitor, whereas too low an estimate could result in a loss for the organization (Agarwal *et al.*, 2001).

With the need of providing better estimates for each new software development project and inspired by the idea of continuous improvement with a knowledge management infra-structure of experience factories (see Chapter 1, § 2.5.1.1), we worked on the definition and use of an experience factory for software effort estimation (**Kosloski and Oliveira, 2005**). To define this experience factory we had to attend each one of the steps of the Quality Improvement Paradigm (described in Chapter 1, § 2.5.1.1), as follows:

i) **Characterize** – based on the literature, we identified thirty-three characteristics that could impact the efforts estimates and, therefore, would be useful to characterize the projects. Some examples of the characteristics are: application type, software complexity, business type, programming language, and development team size;
ii) **Set goals** – we used Goal-Question Metric to define measures with the goal of improving the accuracy of effort estimates. Some of the measures are: estimated size error, initial and final productivity, schedule estimation constraint;
iii) **Choose process** – we defined a software effort estimate process based on (Agarwal *et al.*, 2001; PSM, 2002);
iv) **Execute** – we applied the approach to seven projects of a large organization in Brazil;
v) **Analyze** – we analyzed the results of each project and captured lessons learned;
vi) **Package** – we extended the organization's effort estimation tool including an experience base that registers the projects with their thirty three characteristics and the captured lessons learned.

We will not present here these steps in detail, since a significant part for the experience factory definition is the three first steps for the definition of how to characterize and define the measurements for improvement. These steps were supported by the literature review and the Goal-Question Metric approach that was largely exemplified in Chapter 1. These steps are described in (**Kosloski and Oliveira, 2005, 2006**).

Nevertheless, we would like to detail the analysis phase with regard to the capture of lessons learned (the interest of this chapter). After the execution of each project, a session meeting with development team, estimators and managers was conducted to analyze the main problems with the project estimation and identify lessons learned. These meetings took around of two hours and were supported by all the documentation of the project (the thirty three characteristics) and the collected measures. Based on this documentation the participants discussed and defined lessons learned and recommendations. After that each lesson learned was registered in the experience based and classified in one of the following types:

- Initial - lesson learned identified as a trend and for which there are already some case studies, but still need further confirmations;
- Defined - lesson learned identified as a trend and for which there are several case studies, that confirmed the observed trend;
- Generalized - lesson learned identified in a lot of case studies, and institutionalized as a best practice in the organization.

One important aspect is the definition of the number of case studies necessary to define a lesson learned in an organization. This decision is particular for each organization. In the organization we performed the case study, it was defined the need of more than five projects to classify a lesson learned as *defined* and more than fifteen to classify as *generalized*.

Table 4.4 presents some examples of lessons learned captured after the execution of the seven projects. It shows also the recommendations defined for each lesson learned. All these lessons learned were classified as initial since they were observed in less than five of the seven projects.

Table 4.4. Examples of lessons learned for software process estimation

Lessons learned	Recommendation
Greater experience on requirements definition tends to generate more stable functional scopes for the software, providing more precise estimates of size.	Choose software engineers expert on the business goal
Higher levels of reuse tend to improve the productivity of software development	Try as much as possible to reuse from other similar solutions already developed.
Most experienced development teams tend to perform better productivity in software development.	It is important to balance technical and business handled by the organization in order to homogenize the team's experience and optimize the productivity of software development.
The programming language influences the productivity of software development	Evaluate the productivity of similar systems developed in the same programming language to extract values of productivity in new effort estimates.
The rate of turnover of development team affects the productivity of development so that higher rates of turnover tend to result in lower productivity.	Avoid high turnover of people on the development team
The speed of establishing estimates influences the accuracy of the results, that is, faster estimates tend to give less accurate results.	Verify similar projects with their speed estimates and associated errors, in order to identify an appropriate speed to the new project to be estimated. The time limit for estimating should be established taking into account this information along with the number of experts available to estimate.

4.4 Pro-active Dissemination of Knowledge

In the previous section we presented two approaches of capturing and dissemination of knowledge in software organizations with the use of *post-morten* analysis (§ 4.2.2.1) and experience factory (§ 4.3).

From our own experience, we identified some problems with these techniques: experience factory requires a sizeable investment (in time, people, or process) being more suitable to larger organizations; both techniques focus on explicit knowledge (e.g. in the form of lessons learned or guidelines); and they require the users to “pull” (look for) the knowledge from a repository. That means the software engineer should “pull” the specific lesson learned, guidelines or best practice when s-he needs.

Based on these findings, we proposed an approach adapted to small and medium organizations (**Torres *et al.*, 2006**). It is based on the Dixon's process proposition (2000) (see Chapter 1, § 2.5) for pro-active knowledge dissemination and it uses the Learning History (see Chapter 1, § 2.5.2.2) that allows richer knowledge transfer. Proactive approach means for us that the knowledge is "pushed" to the members of the organization instead of being "pulled" by them when need arises. However, push knowledge does not exclude the possibility of pulling knowledge when needed.

In the process proposed by Dixon (2000) (Chapter 1, Figure 2.16), one of the steps of the knowledge cycle is the selection of a transfer system. Given the characteristics of software projects and their executing team (each project is unique, transfer is from one team to another team doing a similar, non-routine, task in different contexts), we chose to work with the so called "far transfer" system (Chapter 1, § 2.5.1). The far transfer system targets transfer of tacit knowledge between two teams performing similar non-routine tasks in differing contexts.

Figure 4.9 illustrates how we mapped the learning history process (presented in Chapter 1, Figure 2.19) to Dixon's process (Chapter 1, Figure 2.16). Concretely, our proposal is the following:

i) A team realizes a software project (e.g. development of a new system);
ii) Some significant result is obtained that justifies preserving the knowledge gained;
iii) interviews of the relevant project members are planned and conducted, this corresponds to Dixon's activity where the team reflects on the relation between its actions and the results, it is also the 1st and 2nd steps in the learning history process (planning and reflexive research);
iv) Knowledge is converted in a form suitable for dissemination in the next 3 steps of the learning history process (distillation, writing and validation);
v) At the beginning of a new project by another team, the story is recovered (and the receiving team adapts the knowledge to its own necessities during a dissemination meeting (see below);
vi) The history may eventually be published so that it is available to other teams on a repository (for "pulling" knowledge).

To complete the description of our proposal, we need to explain how the dissemination occurs: before starting a new project, a learning history of interest is selected from a repository and presented and discussed within the receiving team. For this, one must first find a story of interest, usually registered in a repository, for a given team and project, and second, concretely use it for knowledge dissemination.

The dissemination is done in a workshop (meeting) structured as follows. First, a week in advance, all members of the receiving team receive a copy of the learning history and some instruction on how to prepare themselves for the dissemination meeting: the story will be discussed during the meeting and, as such, they should read it beforehand and analyze the historian comments; they should write down questions, reactions, comments in the story; they must compare the setting of the project described with their experience; and, finally, they may not discuss the story between them before the meeting as such discussion should happen with all members present.

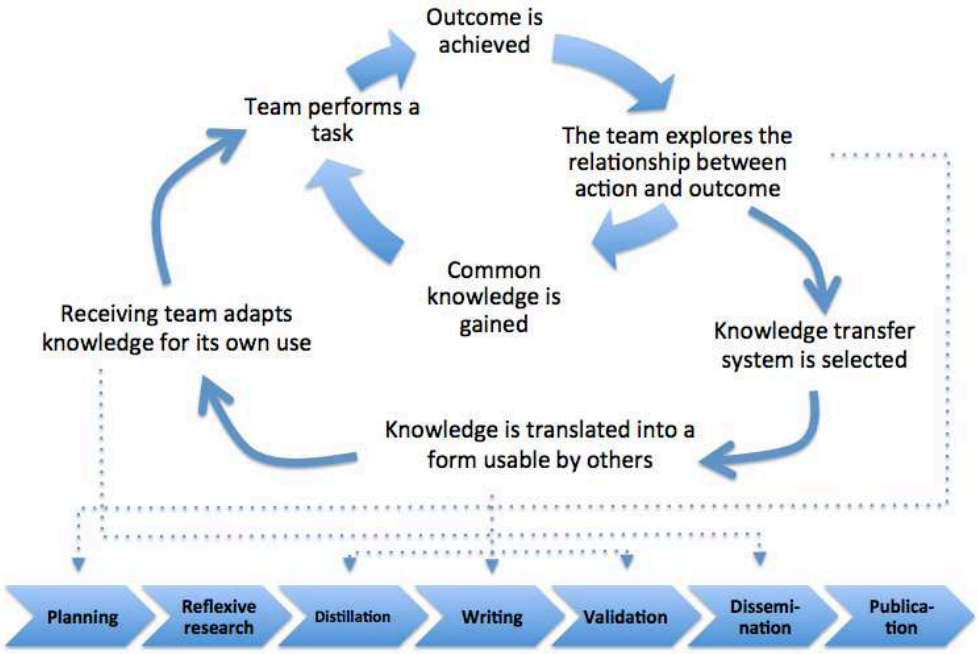


Figure 4.9. Mapping of the learning history process to Dixon's knowledge cycle

The meeting in itself includes two parts. First, the story is discussed to check whether there are comments or questions about it and what is understood about it. Second, the receiving team adapts the knowledge (contained in the story) to its own necessities by answering the following questions:

- What is similar between the project described in the story and the project about to start?
- What good result could be replicated or what mistake should be avoided?
- How the story may impact this project?

As a case study, we created a first learning history from a project and realized a dissemination workshop to another team about to start a similar project. The particular project of the case study was selected because: it was an innovative project (it used a new process and the team was mostly freshly hired); it was expected to be the first of a series of others occurring in similar conditions; it was rich in notable results (both positive and negative); we had easy access to its participants; and we knew well the project in itself (although none of us participated in it). This project lasted 7 months and involved 2 business analysts, the head of the department, one project manager, 2 (internal) clients, and some users consulted when more details were needed on some particular area.

We conducted five interviews (1 manager, 2 clients, 2 business analysts) for a total of 3 hours of interview. The cost of the total case study is summarized in Table 4.5 and was judged acceptable given the size of the project. A small part of the resulting story (which has 20 pages) is presented in Figure 4.10. The dissemination was conducted in a 3 hours meeting with a team of 5 persons, about to start a project in similar conditions. The project manager (of the new project) was the moderator during the dissemination meeting. The overhead of the initiative is very small, about 1 week (39 man/hour) to build the history for a 7 months project with five interviewed of the first team (the actual team was larger than that), and 18 man/hours for the dissemination for the new project (the dissemination is not dependent on the length of the new project, but on the size of the new team).

Table 4.5. Cost of a knowledge cycle using learning history

Activities	Duration	Participants	Total (man/hour)
Planning	1 h	Historian	1 h
Interviews	3 h	Historian & team members	6 h
Story transcription	12 h	Secretary	12 h
Document conception	20 h	Historian	20 h
Story reading	1 h	New team	4 h
Dissemination workshop	3 h	Knowledge manager & new team	18 h

Beginning of all: Information	
The [organization] produces information on agribusiness and supply chain. Produced indicators were treated by different departments, historically with communication problems. As a result, each department created its own reports with redundant information and differing formats (from the other). The project aims at correcting this problem.	
System's goals: Organize and make accessible information in order to fulfill the strategic goals of the organization	SUPER-INTENDANT: Based on the [organization]'s strategic planning, some goals were defined, which were exactly that the [organization] would become a reference in terms of providing information and knowledge in the agricultural supply chain sector. So, to actually fulfill this goal, an internal reorganization was needed. This is where agribusiness and supply chain fits, and also the reason why it was set as a priority. MANAGER: In the 90's, we developed a system for pricing, that is now called "agripricing". Later new tools appeared on the market, the [organization] went from mainframe to client/server, and we would always talk about the necessity to have a project where we could map all the information on the agribusiness, not only prices.

Figure 4.10. Extract of a learning history (translated from Portuguese)

The meeting was evaluated very positively by the participants, one of them actually proposed that the organization should generalize this kind of proactive knowledge dissemination action (he did not know the details of the research and that this was our goal from the very beginning). Another evaluation showed that some conditions necessary to learning were actually fulfilled. These conditions are (von Krogh *et al.*, 2000): mutual trust, positive empathy, assistance access, kind judgment, and courage. We found that three of these conditions were fulfilled, and that we lacked data to reach any conclusion in two. We evaluate that this lack of data was due to the first part of the meeting (understanding of the story) having taken too much time. During this part, little knowledge is actually created. The second part (adaptation of the story to the team's reality) had to be interrupted before it was actually completed. This is something that we plan to correct in the next dissemination meeting.

Although we still lacked one small part of the model (indexation of the stories and retrieval from a repository), this first experiment showed that the capture and dissemination of the knowledge actually happen at a relatively low cost.

A second learning history and its dissemination can be found in (**Torres *et al.*, 2009**). This book also presents an analysis of the presented history using the document of learning history itself.

4.5 Using Ontologies to Improve Usability of Personalized User Interfaces

Usability is one of the quality characteristics more explored in literature since it has been studied by two research communities: software engineering and human-computer interaction. Several standards, quality models, tools and methods have been proposed to support the usability quality assurance – see, for example (Sears and Jacko, 2009) for a large discussion about methods and tools for usability evaluation; and (Seffah *et al.*, 2006) about quality models and measurements. Usability is defined as the extent to which a product can be used by specified users to achieve specific goals with effectiveness (the accuracy and completeness with which users achieve specified goals), efficiency (the resources expended in relation to the accuracy and completeness with which users achieve goals) and satisfaction (the comfort and acceptability of use) in a specified context of use (ISO 9241-11, 1998).

Recognizing that ensuring the usability cannot be postponed and only evaluated when the final user interface is developed, various methods and approaches have been proposed to introduce usability issues throughout the development of an interactive system (some examples can be found in: Abrahão and Insfran, 2006; Sears and Jacko, 2009; Anderson *et al.*, 2001; Fink and Obendorf, 2008). In this context, issues related to personalization emerge in search of enhance the usability of interactive systems (e.g. Vora and Bojewar, 2011; Vuljanic *et al.*, 2010; Glavinic *et al.*, 2008; Kumari *et al.*, 2012).

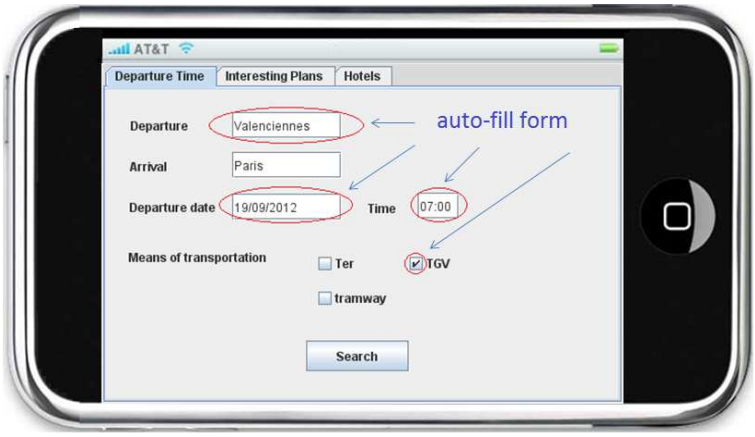
Personalization has been defined as:

- "the ability to provide content and services that are tailored to individuals based on knowledge about their preferences and behavior" (Hagen *et al.*, 1999);
- "Delivering to a group of individuals, relevant information that is retrieved, transformed, and/or deduced from information sources" (Won, 2002);
- "adaptation towards a named user for which an internal and individual model is needed" (Garía-Barrios *et al.*; 2005);
- "the dynamic adaptation of the interface to the profile" (Simonin and Carbonell, 2006).

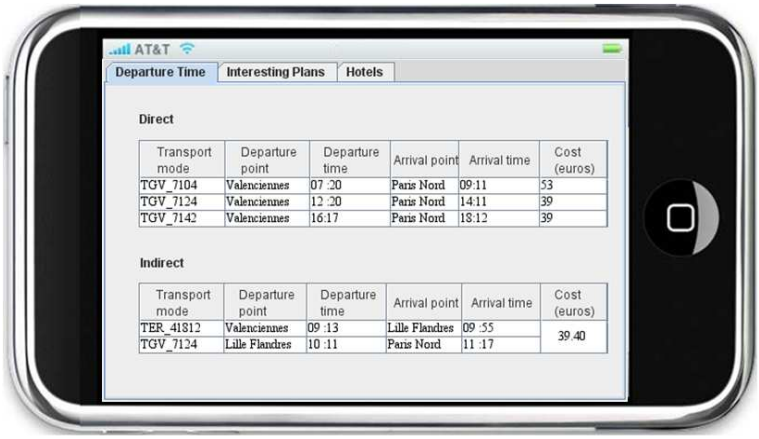
In general, personalization deals with the ability of adapting a user interface (UI) considering some information related to this user. Personalization can take many features into account and can be applied to many levels (Kobsa, 2001; Chevalier and Julien, 2003), as follows: personalization of the presentation (containers), that aims to adapt the style and format of interaction interface components (e.g. buttons, text fields) based on the user needs and context; personalization of the structure, applied to the links between a website's pages; personalization of functionalities, to make available only the functions necessary for a specific user to answer a task by automatically adapting the system; personalization of the navigation, that guides the user to the right information, by avoiding irrelevant pages; and personalization of the content that works on the selection and adaptation of the input/output information, according to the user, his/her preferences and context.

Figure 4.11 illustrates examples of the personalization of the content and containers of a travel planning system (**Oliveira *et al.*, 2013**). For instance, the adaptation of the size of interface elements, such as fonts and widgets, represents a personalization of the container presentation based on the specific platform/device (i.e., iPhone) used by the user. Information about the departure city and departure date (Figure 4.11(a)) are examples of personalization of content. Departure city is automatically filled based on the user's current location. Departure date is the current day. Another example of personalization of content is presented in Figure 4.11(b). In this example, the user is unable to walk; thus, the system will propose direct

itineraries with a reduced price, according to his/her age, or indirect itineraries with the connection in the same train station.



(a)



(b)

Figure 4.11. Examples of user interface personalization in a transportation system (**Oliveira *et al.*, 2013**)

Analyzing the types of customization, we realize that the personalization of content can directly impact on usability by improving efficiency (thanks to the auto-fill form of information that the system already knows, and therefore allows the user to save time in performing the activities) and user satisfaction (the user feels that the system helps in his/her activity). Thus, we decided to investigate how to develop interactive systems providing content personalization.

To address this idea, we explicitly define a context model to capture all relevant information related to the user (**Bacha *et al.*, 2011a**). This context model is generic and it can be used in the design of any user interface. To adapt such model to a specific application domain, we propose that the context model concepts are mapped to the concepts of an ontology that captures the knowledge of an application domain. In this way, it is possible to indicate which specific application domain information should be provided as input/output in the user interface based on the context.

Furthermore, we noted that we should work even during design time and should consider that an interactive system can be developed to run into several platforms (e.g. tablets, desktop,

mobile phones) and medias (e.g. audio, video, image, text), therefore, being necessary to adapt its UI. Model-driven architecture (MDA) (OMG, 2003) has been shown to be an appropriate approach for the design and code of the software system and their UI to address those challenges since we could specify the personalization issues in high-level of abstraction independent of the platform and it would be applied in the final application. In MDA, models play a more direct role in software production, manipulation and transformation by machines. UI can be specified with a high level of abstraction, and the final UI are transformed for different platforms. We, therefore, embed the context model and the domain ontology in a MDA framework to allow semi-automatic generation of personalized user interfaces (Bacha, 2013).

In this section, we detail how the ontology is applied to make possible to perform content personalization. Details about the UI generation using MDA can be found in several papers (**Bacha *et al.,* 2011a, 2011b, 2011c, 2010**). We applied this approach in two application domains: transportation (**Oliveira *et al. 2013;* Bacha *et al.,* 2011a, 2011d**) and medical domain (**Bacha *et al., 2013, 2011b***). In this section, we will use the transportation domain to illustrate the proposition.

4.5.1.1 Context Model Definition

Our first step for performing content personalization was the definition of the context model. In general, the research in UI design considers the context is composed of three classes of entities (Calvary et al., 2003): the system's user profile; the platform (i.e., hardware and software), which is used for interacting with the system; and the physical environment in which the interaction takes place.

To define the information about those elements, we have done a literature review (**Bacha *et al.,* 2011a**). Although eighteen context model propositions were found, none of them was considered complete. Some propositions considered only one of the context dimensions: user, platform or environment. Others were particular for a specific domain (e.g. smart phones, e-commerce) and/or not detailed enough. Thus, we decided to integrate the main information from all propositions in a context model covering the three parts (i.e., the user profile, platform and environment). We noted that some information about user (like his/her preferences, abilities and activities) should be particularized to each domain for a better definition of the context, and other (like demographic and contact information of the user, and all information about the environment and platform) is generic to any domain. Therefore, the context model was defined with some classes that should be instantiable for particular domains and others generic being applicable to any domain. Figure 4.12 presents the part of the user profile context model for the transportation domain. The complete context model can be found in (**Bacha *et al.* 2011; Oliveira *et al.,* 2013**).

The user profile model is divided it into five major parts, specifying the user when interacting with the final interface: Contact information, which contains personal data; Demographic information, which contains basic and unchanged user data; Preference, which describes user interests and preferences; User State, which describes the physiological user state and user activity; and Ability and Proficiency, which specifies the user skills and abilities.

Once a context model is specified, the next question to address is how it could be used in a specific application domain. For example, with respect to Figure 28, how can we set that "departure" is the city where the user is at the time of using the system or that, for the proposed itineraries, we should consider the age and the ability of the user? To address this

problem we propose to map the context related concepts to the specific concepts of the application domain.

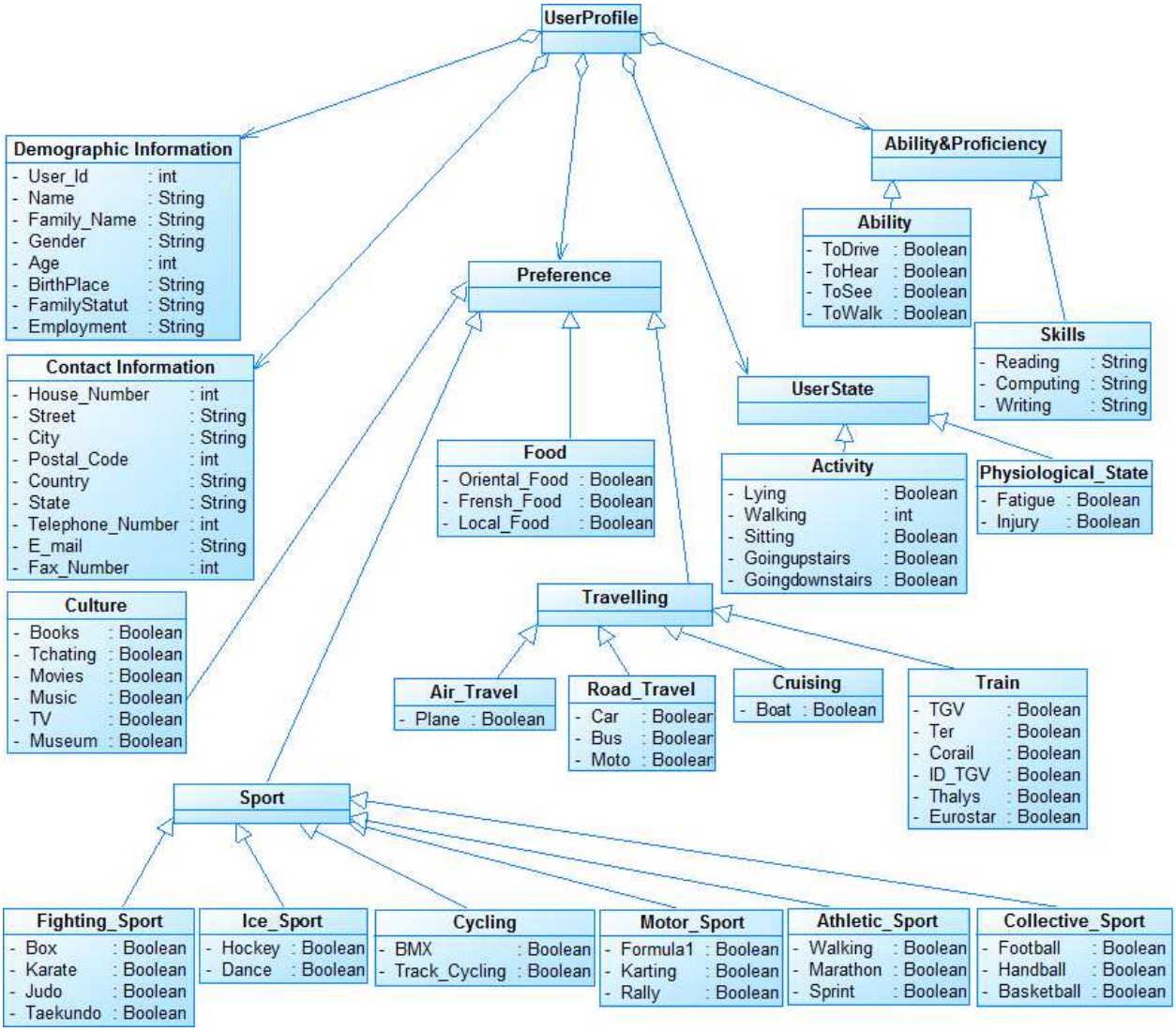


Figure 4.12 The context model: user profile (**Bacha *et al.*, 2011a**)

4.5.1.2 Mapping Context Model and Domain Ontology

To establish the mappings we assume that the vocabulary of an application domain is defined in a domain ontology. We chose to work with ontologies since our previous experience (**Oliveira *et al.*, 2006; Anquetil *et al.*, 2006; Oliveira *et al.*, 2004; Leão *et al.*, 2004; Oliveira *et al.*, 1999a, 1999b; Maidantchick *et al.*, 2000**) showed us that ontology is very useful to organized the knowledge of an application domain independently of a particular software system and of how it will be used in the development of that system. Therefore, the ontology can be applied in the development of various applications of the same domain.

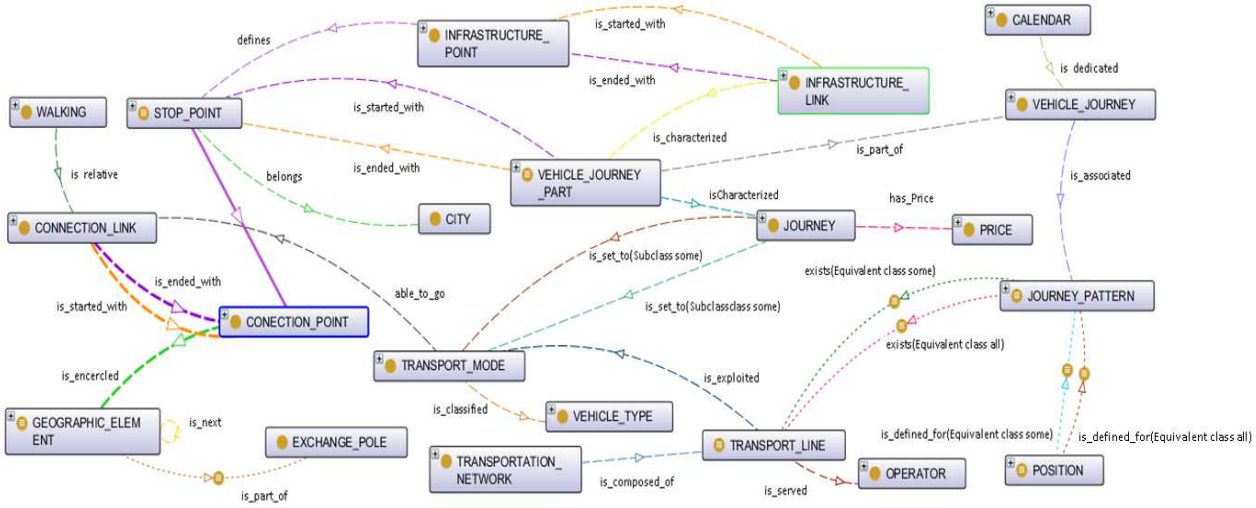


Figure 4.13 shows the top-level transportation ontology we developed for the application of this approach in the transportation domain.

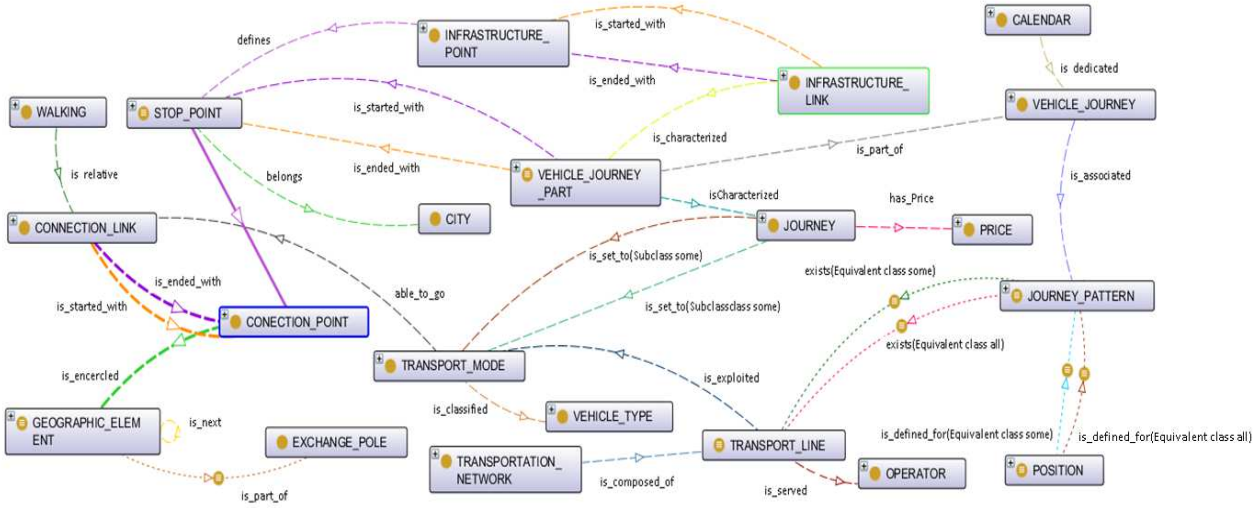


Figure 4.13. The transportation ontology global view (**Oliveira *et al.*, 2013**)

The transportation ontology was formalized in OWL1.0 and Protégé16. A detailed description of this ontology can be found in (Mnasser et al., 2010; Oliveira et al., 2013).

Considering the elements of a domain ontology (i.e., concepts, properties, relationships, and axioms) and those of a context model (i.e., concepts, attributes and relationships), we propose a meta-model (see Figure 4.14) that sets a mapping between any element from a context model and any element from a domain ontology, except for the constraints. The constraints express rules that are used with the mappings for the personalization. Such meta-model exploits three types of mappings:

- Direct mapping, when some information of the domain is directly associated with the information modeled in the context in that they have the same meaning. For example,

[16] Available at http://protege.stanford.edu (Accessed April 10, 2014)

any information to identify the user in the domain ontology is directly associated with the name of the user in the context. The departure city name in Figure 4.11(a) is another example of direct mapping. To define a direct mapping the designer just look in the domain ontology if there is any information that is the same as some information defined in the context model;

- Indicative mapping - when some information of the context indicates the presence or absence of some information in the domain. For example, the information about a user interest, such as s-he practices cycling as sport, can indicate the kind of book s-he can be interested in a domain of bookstores. The preferred transportation mode in Figure 4.11(a) is also an example of indicative mapping. To define an indicative mapping the designer should verify for each context element defined as a Boolean attribute if there is some concept in the domain that represents that context element;
- Indirect mapping - when some information of the domain ontology is influenced by some information in the context model. For example, the price of travelling for seniors or students is indirectly associated with demographic information about age that will influence the search for prices. The proposed itineraries in Figure 4.11(b) are another example of indirect mapping where the itineraries are indirectly associated with the age and abilities of the user. To define an indirect mapping the designer should verify if there is any information in the domain ontology that could vary depending on some personal information modeled in the context.

Note that in Figure 4.14 any element of an ontology (concepts or attributes) can be mapped onto any element of the context of use (concepts or attributes). For each mapping, the type (direct, indicative or indirect) should be set. Note also that we can have domain concepts without any mapping, or with more than one mapping.

Figure 4.15 shows some of the mapping examples from our transportation ontology and the context model elements.

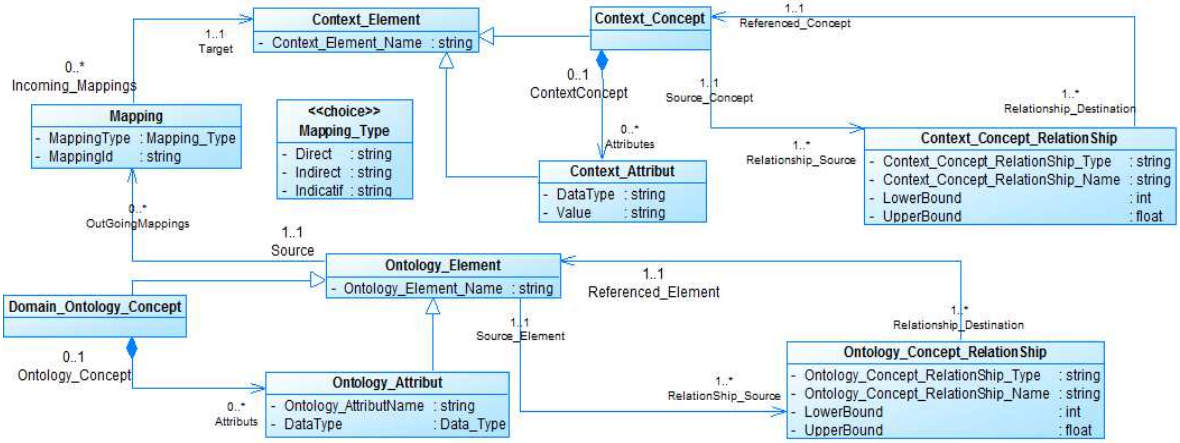


Figure 4.14. Meta-model for mapping context and ontology elements (**Bacha *et al.* 2011a**)

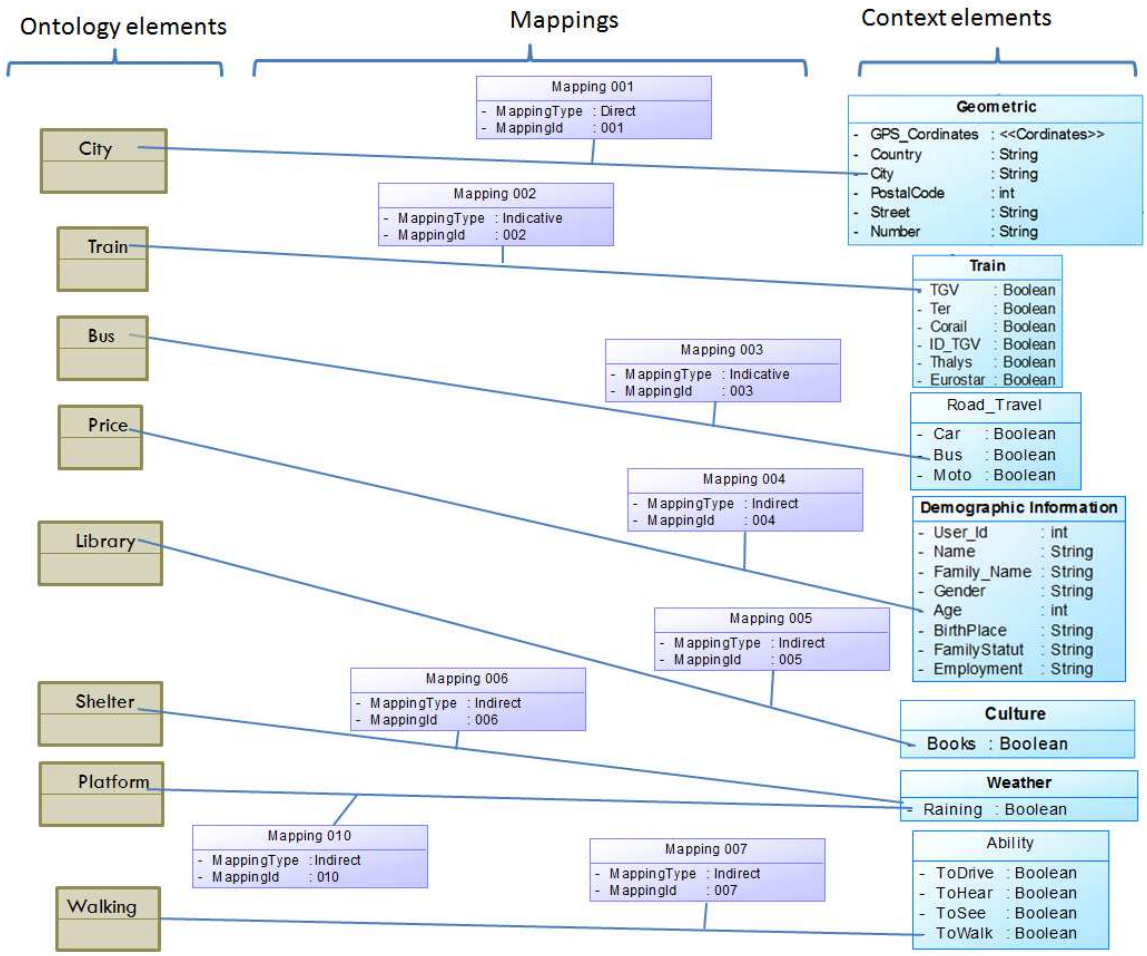


Figure 4.15. Mapping examples (**Oliveira *et al.*, 2013**)

4.5.1.3 Application: Using Domain Ontology for Content Personalization of User Interface

With all the mappings defined, while the UI designer specifies the UI using the domain vocabulary represented in the ontology, s-he is also embedding the context information that should be presented in the final UI at the runtime. To do so, our MDA approach uses the mappings, ontology concepts and axioms to pre-define the functions and the information that should be retrieved from the context and the domain ontology. To better explain this, we suppose we need to develop a system for planning a journey. In this scenario, the user connects to the system to be identified. Thus, all information about the user's context is already registered. Next, the system shows the form with the required information (e.g. departure city, destination city, dates, and user preferences of transport mode) and, then, the result of the user's query. Figure 4.11 shows an example of UI for this system. Generating personalized UI means dynamically generating a UI for the platform, which the user is using the system with all the user information and the domain already filled in the correct input fields. Figure 4.16 shows the UI specification (high-level specification in a MDA structure, i.e., so called computation-independent model) related to this part of the process.

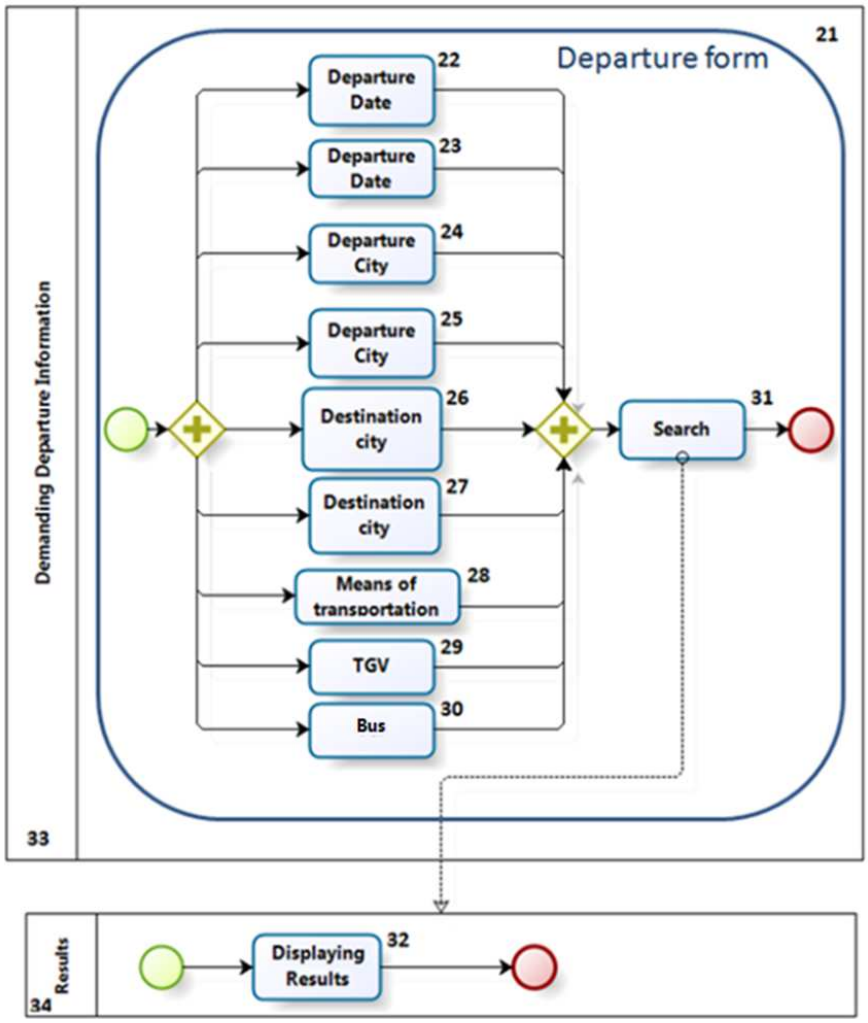


Figure 4.16. Example of a UI specification

As shown in Figure 4.17, each element in the specification is defined by a ID, its type (i.e., "user task" if the element represents an user interaction; "system task" if the element represents the result of an execution of the system functionalities; or "subprocess" if the element will be decomposed in other elements until the elementary elements that represent the "user task" and the "system task"), the kind of interaction element is presented (e.g. the field, "DepartureCity", will be presented in a field that should be filled in manually by the user), the domain vocabulary related to this field (i.e., the ontology concept associated), and the kind of mapping, if applicable.

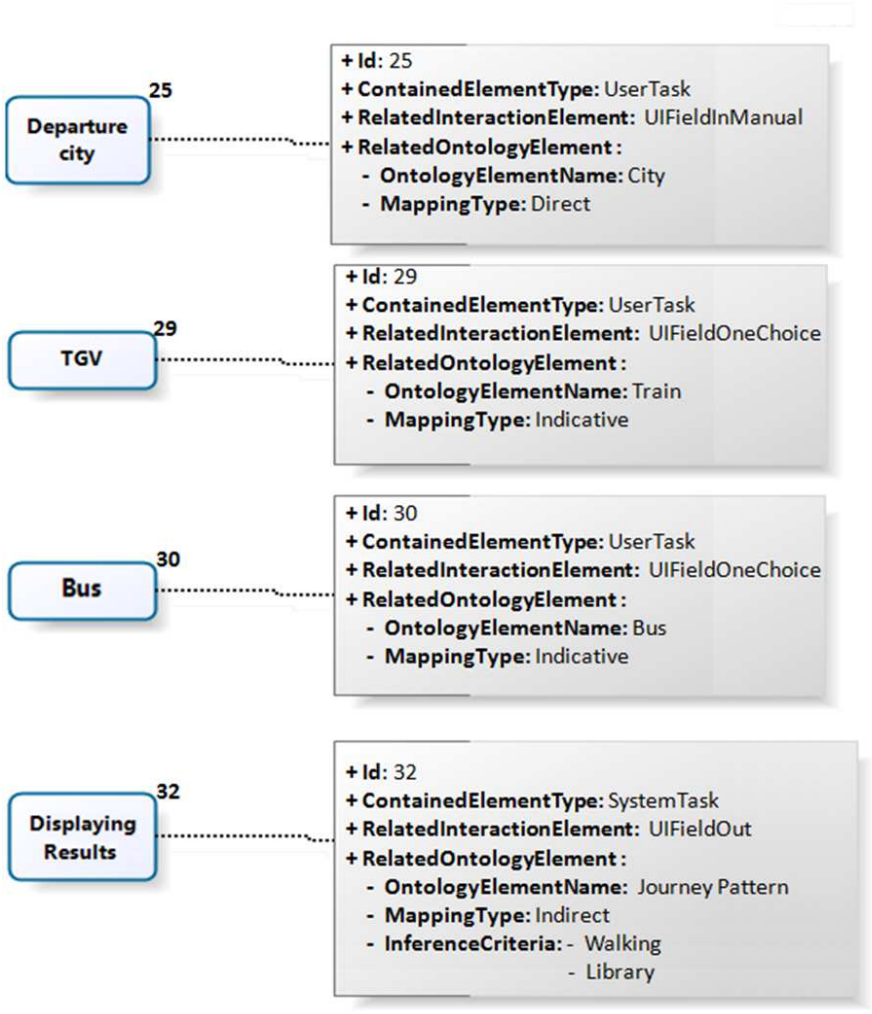


Figure 4.17. Annotation of task model elements (**Bacha *et al.*, 2011d**)

To provide content personalization, the departure city and user preferences for the transport mode should be already filled in by the system. Although the user can change this information, the system should provide the form with all content collected based on information collected from the context. The results of the query should also consider the domain knowledge and the context. In this example, the user is unable to walk. So, the system should propose direct itineraries with a reduced price, according to the user's age. In case of indirect itineraries, the system should propose only protected journey patterns containing shelters since it is raining, information being obtained from the context. After defining the annotated UI task specification, the designer can launch the MDA transformation process (see (**Bacha *et al.*, 2011b**) for details about the transformation rules). The specification of the low-levels in the MDA structure was written in UIML (User Interface Markup Language) (Helms *et al.*, 2009), a general language for describing UI that facilitates the work with dynamic information. In this transformation, the content personalization of the UI is done by two methods (Doucet *et al.*, 2004; Ioannidis and Koutrika, 2005; Abbas, 2008): auto-fill forms and query enrichments.

Auto-fill forms are performed for all the fields to which we defined the direct or indicative mapping in the UI task model (Figure 4.16 and Figure 4.17). To that end, we search for what concept from the transportation ontology was used and what context element it was mapped with. For direct mapping, the UIML property that contains the content (g:text) will be filled in

automatically by the value taken from the instance of the context model by invoking a method that has as parameter the name of the ontology element that is related to the task.

In the same way, it is verified the information related to the concept of the ontology, which had an indicative mapping. In order to decide the selection of the concerned element, the value of the context element (i.e., true/false) is verified in the condition part in the UIML code, for the pre-selection of the right alternative.

For an indirect mapping, the transformation generates a *call* type statement from UIML. It represents a method that is coded by the software designer to search the required information based on the given parameters. This method is equivalent to a searching query, in which the *DomainElementName* annotation attribute presents the searched element and the method's first argument, and the *InferenceCriteria* presents the parameters to take into account when searching them. Since we were interested in providing personalized content to the user, this query was enriched by including automatically other context elements during the search process. We used the ontology relationships and axioms for this enrichment. The idea is to exploit the ontology classes that implement an indirect mapping, which the designer has not explicitly specified as inference criteria in the UI design. Assuming that the *DomainElementName* is named "A" and that "B" belongs to its *InferenceCriteria*, each ontology class that has an indirect mapping and that is connected, threw a functional relationship, to a class on the path from "A" to "B" will be integrated among the *InferenceCriteria*. To that end, the properties of the concepts are explored. As a result, it will enrich the query.

Table 4.6 shows an example of the use of ontology properties for the query enrichment. Details of the UIML code generation by transformation can be found in (**Oliveira *et al.*, 2013**).

Table 4.6. Using ontology properties in path finding

Relationship characteristic	Definition	How to be used
Functional Property	If a property, P, is tagged as functional, then for all class C, C', and C'': P(C,C') and P(C,C'') implies C' = C''. In other words If a property is functional, for a given individual, there can be at most one individual that is related to the individual via the property.	Assuming that "B" is an *InferenceCriteria* for searching "A". Each ontology class that has an indirect mapping and that is connected threw a functional relationship, to a class that belongs to the path from A to B, will be integrated among the *InferenceCriteria* since this class is considered strongly pertinent for searching "A". A class related with a functional property means that there is at most one instance that is related to the other class, therefore, it restricts the search.

As previously mentioned, we also applied this approach to the medical domain, in particular for medicine recommendation (**Bacha *et al.*, 2013**).

We confirm with the two case studies (for transportation and for medical domain) that a domain ontology is very useful in the process of personalization since the design time. The ontology was used to automatically identify the information to perform query enrichment and auto-fill forms to the runtime when the inference rules can be integrated in traditional methods to provide the required result of the software application. However, in order to be better applied, support tools should be developed to help the software designers to use this approach.

This research was subject of different publications (Bacha, 2013**; Oliveira *et al.*, 2013, Bacha *et al.*, 2011a, 2011b, 2011c, 2011d, 2010**).

4.6 Conclusion

This chapter has presented some works that use knowledge management practices in order to provide better quality for software process and products. From these works, we can quote in general the following main contributions:

- Definition of an approach to improve software maintenance and software maintenance process using domain ontologies;
- Adaptation of PMA for the context of software maintenance projects. For our knowledge this technique was more applied to software development projects than for maintenance. It was concerned only with the knowledge acquired in the project and not about the software system itself like we defined for software maintenance;
- Definition of an approach for proactive knowledge dissemination in software projects adapted for small and medium organization. This approach combines learning histories (technique usually used in context different than software) with the knowledge cycle proposed by Dixon (2000);
- Definition of and experience factory for effort estimation;
- Definition of a MDA approach based on ontologies for content personalization of user interface aiming to improve the usability of interactive systems.

Based on the learning experience of applying knowledge management practices and also the experience and knowledge about software quality assurance presented in the previous chapter, we concluded that the software quality assurance is a knowledge-rich activity including explicit documented knowledge in standards, quality measures and techniques as well as tacit knowledge of each individual that participates in the quality evaluation process (evaluators, users and all stakeholders in the process). These people are carriers of knowledge and this knowledge directly affects the process of quality assessment. We believe it is important to identify this knowledge and analyze how to take advantage of it while performing software quality assurance. This belief brings the following questions:

- What explicit knowledge is used and shared in every moment?
- How to integrate the process of using tacit knowledge in the software quality assurance process?
- How to transfer and maintain this knowledge to be useful for future quality evaluation?

Research perspectives to answer those questions are presented in the next chapter.

Chapter 5 Synthesis and Research Perspectives

5.1 Introduction

Software quality assurance is gaining wide importance in the current market that seeks better customer satisfaction. Organizations recognize that it is only by offering better quality products that they will have an edge over their competitors and thus they can ensure customer loyalty. In this context, the research on software quality has proposed methodologies, techniques and different approaches to support this activity and its effective use in industry.

Precisely, this report has presented the research that I have conducted in that direction. Since my master degree (1993), I have been working in research of software quality assurance. I explored in theory and in practice, evaluation methods, measurements and knowledge management procedures. In the next section (§ 5.2) I present a synthesis of my contributions analyzing what was done and what is still to be done. Based on this analysis, I propose some research perspectives that I plan to work on (§ 5.3).

5.2 Synthesis of Contributions

To summarize my contribution to the domain of software quality assurance, I come back to the four challenges presented in Chapter 1 (§ 1.2). For each challenge, I recap the research performed to address it, examine the results objectively, and propose some possible further investigation.

Challenge 1. Definition of quality measures and their interdependencies

I have worked on the definition and application of measures for different types of software systems and software artifacts, as follows:

- Expert system – measures for evaluating expert system quality were defined and applied in practice for the different artifacts produced in the software development (software requirements, models and final product) (**Rabelo *et al.,* 1997**);
- Web application – qualitative and quantitative measures were defined and applied to ensure the quality of web sites (**Lima *et al.,* 2000**) and web application (**Lima *et al.,* 2009*;* Valentin *et al.,* 2012**);
- Legacy systems – technical (**Ramos *et al.*, 2004, 2005**) and managerial (**Queiroz *et al.*, 2009**) measures were defined and applied in large legacy systems from industry supporting the managers' decision-making;
- Ubiquitous systems – a large review about what should be explored in this new domain was performed and measures were defined and applied (**Santos *et al.*, 2013;** Santos, 2014);
- Human-computer interaction (HCI) – measures for the well-known criteria used by the HCI community were defined and applied, showing that quantitative values can

support qualitative values defined by the subjective evaluations (**Gabillon *et al.*, 2013*;* Oliveira, 2010**);
- Software process – measures for software process evaluation were defined and applied in real projects (**Gomes *et al.* 2000*;* Sousa *et al.*, 2003**). Indicators were also defined integrating a set of measures to support process performance analysis (**Monteiro and Oliveira, 2011**).

However, although I recognize that measures are interdependent and that the result of a measure impacts on others, I did not specifically explore these interdependences. In all research I performed, this analysis was executed by looking at the results of the assessment for each one of the software artifacts under evaluation.

For all studies performed for the definition and application of measures, specific methods for software quality assurance were explored as follows:

- Adaptation of phased inspections to the evaluation of expert system specifications and final code (**Rabelo *et al.*, 1997**)[17];
- Use of walkthrough in the evaluation of legacy systems specifications and design models (**Ramos *et al.*, 2004, 2005**).
- Combination of existing methodologies for the definition of reading based techniques to be used in the evaluation of usability in early stages of software development (**Valentin *et al.*, 2012**);
- Definition of an approach for software testing planning by establishing the quality characteristics and software testing techniques to be used (**D'Oliveira *et al.*, 2003**)[10]. We also used testing techniques for the formal evaluation of expert system and ubiquitous systems (**Santos *et al.*, 2013*;* Rabelo *et al.*, 1997**).

Finally, in all practical applications I performed SQA, the presence of an expert for the final judgment was required. Even when we had a threshold well established (**Ramos *et al.*, 2004, 2005; Queiroz *et al.*, 2009**) the qualitative analysis of the quantitative data was required. I do believe that we need historical data for a better support in the analysis and interpretation of the results. Suggestions for analysis and interpretation can be drawn based on previous results of similar projects in similar contexts. Another idea would be to investigate knowledge management issues to support SQA. These ideas are presented as research perspectives in the next sections (see § 5.3.1 and § 5.3.2).

Challenge 2. Definition of experimental protocols for the SQA

All the case studies I performed were in the context of specific projects as proof of concept. I recognize the importance of doing formal experimentation for the generality of the results. I worked on two evaluations with formal definition of experimental protocols and statistical analysis of the results (**Valentin *et al.*, 2012; Santos and Oliveira, 2009**). Those studies showed me that despite the difficult to perform formal experimentation, we have much more confidence in the results. Therefore, much more should be done in this direction. A relevant idea proposed by (Sayeb *et al.*, 2012) is to define patterns of experimental protocols.

[17] For reason of space, this research was not presented in this document.

Challenge 3. Implementation of continuous software process improvement

In the previous chapters, I presented my experience of the definition, evaluation and improvement of software process, as follows:

- Definition of software processes for specific domains and considering standard software processes (**Machado *et al.*, 2000; Villela *et al.*, 2002; Macedo *et al.*, 2006**);
- Definition of a measurements (**Gomes *et al.*, 2000**; **Weber *et al.*, 2004**) and a catalog of indicators (**Monteiro and Oliveira, 2011**) for the evaluation and improvement of software process;
- Adaptation of project reviews approaches to collect lessons learned for continuous software process improvement (**Anquetil *et al.*, 2006; Torres *et al.*, 2006, 2009**). Nevertheless, the cycle of quality improvement was not completed being necessary the improvement of the software processes based on the identified issues and performing their re-evaluation.

Such research experience was essential for my participation in the definition of the Brazilian Improvement Software Process Standard (MPS.BR). I worked with the technical team that defined the standard and institutionalized it in the Brazilian enterprises. I was one of the editors of the second version of the standard in 2007, being responsible for the revision and integration of modifications required by the consultants who implemented the model. Currently, more than 500 appraisals were performed in this standard. MPS.BR has become the most important model of software process improvement in Brazil.

Challenge 4. Definition and use of knowledge management procedures for SQA

Chapter 4 presented my experience on applying knowledge management issues for organization, capture, dissemination and use of knowledge to support quality assurance, summarized as follows:

- Knowledge organization using domain ontologies to produce software with better usability (**Oliveira *et al.*, 2013; Bacha *et al.*, 2013**) and support the maintenance of legacy systems (**Anquetil *et al.* 2006, 2007**);
- Knowledge capture and dissemination with post-mortem (***Anquetil et al.*, 2006**) and learning history (**Torres *et al.*, 2006, 2009**) techniques providing a way of producing better products and improving software process activities based on lessons learned from the past projects (for software development and maintenance);
- Definition and implementation of a knowledge management structure to support the improvement based on past experiences with the development of an experience factory for continuous software effort estimation (**Kosloski and Oliveira**, **2005**).

Moreover, in my Doctorate research, not presented in this document, I defined how to use domain knowledge (formalized with ontologies) to support activities of the software development improving the traditional software processes (**Oliveira *et al.*, 1999a; Oliveira *et al.*, 2004**).

All the research I did on this topic was focused to support the SQA activities defining and applying propositions to identify problems to allow the improvement of the quality of software products and processes. However, I did not explore how to use knowledge management issues to manage the knowledge of the SQA activity itself. I argue that SQA is a knowledge intensive activity performed by individuals that have their own knowledge and experience, which can impact in the quality evaluation results. Based on this belief, I propose a research perspective in this direction (see § 5.3.3).

All those challenges were intensively discussed in the context of a French research action I coordinated during two years. Several researchers from nine laboratories participated in the meetings. Besides the rich discussions we had, we also produced several papers (**Dupuy-Chessa *et al,* 2014**; Arduin *et al*, 2012, 2013**; Oliveira *et al., 2012***; **Kolski *et al.*, 2012**), a special issue in a national journal (**Oliveira and Rosenthal-Sabroux, 2013**) and the definition of the new frontiers for information system evaluation in a collective paper from the French computer science community (**INFORSID, 2012**).

5.3 Research Perspectives

Two main concerns guide my research perspectives: quality evaluation based on measurements for interactive systems, and knowledge management of software quality assurance activity.

The first concern is related to the use of measurements for the continuous evaluation of software products. Being integrated in the research group of DIM (Decision, Interaction and Mobility) of the LAMIH research laboratory since 2009, I planned to explore this issue for the evaluation of human-computer interaction (HCI).

When talking about quality of HCI, one relates it directly to the usability evaluation. A wide variety of studies and projects have demonstrated that a commitment to usability and user's needs in the early design phases may have an impact on different areas such as user satisfaction, performance, safety as well as commercial viability (see, for example Mayhew (1999)). However, the evaluation methods in the context of HCI usually require either a fully functional prototype that end users can play with or an early prototype that only the evaluator can present to the end users or use in an inspection-based evaluation. The tests are most often conducted after the development and deployment of the interactive software. They also require specific usability labs, which may result in significant maintenance costs. Furthermore, with the diversity of computing platforms, interaction devices and styles, the (semi) automatic generation of user interface (UI) from design models has also seen significant progress. Different UI design models have been proposed to support the derivation of a concrete UI from a high level abstraction of the UI (model). A derivation consists to transform UI models at different level of abstraction (see for example, Vanderdonkt, 2005; Brossard *et al.*, 2011; Sottet *et al.*, 2007). The question addressed is, can we – software developers, usability expert, HCI designer - predict usability from these early UI design artifacts and models?; in other words, can we define predictive measures to evaluate usability in the early design stages and without a fully functional prototype? My first research perspective aims to answer these questions with the definition of an approach for predictive usability evaluation (§ 5.3.1).

In a previous study (**Santos *et al.,* 2013**) (see § 3.3.3) we concluded that issues related to quality evaluation of HCI in ubiquitous systems are broader than usability aspects. We identified a set of quality characteristics and we started to define some measures. By applying these measures in the evaluation of three systems, we noted that formal procedures about how to evaluate the quality of HCI of this type of systems should be defined. Moreover, we noted that the use of measures was really helpful for the identification of UI problems and therefore the definition of improvement actions. We plan therefore to continue the research on quality evaluation of HCI by defining measures for all quality characteristics proposed in section 3.3.3, looking for their correlation, and defining a methodology for quality evaluation of HCI in ubiquitous systems based on measurements (see § 5.3.2).

Finally, the second concern is related to the software quality assurance activity itself. That means to investigate what is the tacit and explicit knowledge behind a quality evaluation. To

that end, I intend to come back to the basis of knowledge management issues and to explore the idea that considers knowledge not as an object but the result of a process (see § 5.3.3).

5.3.1 Predictive Usability Evaluation

During the last two decades, usability evaluation has been intensively treated by HCI researchers and usability professionals. A large number of usability evaluation methods and models have been proposed (Albert *et al.*, 2010; Cockton *et al.*, 2009; Abran *et al.*, 2003). An exhaustive account of these methods can be found in (Ivory et al., 2001). Several models and standards have been established to measure usability as a quality construct. Mainly these models break down usability into measurable attributes. They also provide a list of measures that are generally in conjunction with usability evaluation methods to qualitatively and quantitatively estimate usability attributes ISO 9241-11 (1998) and ISO/IEC 25023 (2011). Building on or contributing to these standards and usability evaluation methods, several European projects suggested specific frameworks to measure usability; representative examples include SUMI (Software Usability Measurement Inventory) and MUSiC (Metrics for Usability Standards in Computing) developed by the Human Factors Research Group, University College, Cork. Trying to integrate different models in a same structure from a literature review, Seffah *et al.* (2006) proposed a consolidated model, named QUIM (Quality in Use Integrated Measurement). QUIM is a unifying model that assesses usability by using 12 factors, 29 criteria and more that 120 measures. Another significant research is the Single Usability Measure suggested to standardize usability measure into a single global score (Sauro and Kindlund, 2005). Moreover, guidelines and ergonomic criteria for usability evaluation were also proposed (e.g. Scapin and Bastien, 1997; and, Vanderdonkt, 1994).

However, most of those projects do not address explicitly usability prediction from early UI design models. This is because the calculation of the proposed measures in these projects requires a fully functional prototype. Among the few initiatives, one can mention, the European action COST294 that highlighted the need for theoretical frameworks defining the nature of interaction quality. This includes techniques and measures for the evaluation of the interaction quality during design. Another initiative is the proposition of Panach *et al.* (2013) to represent usability features related to functionality abstractly in a conceptual model, and than to carry out automaticaly their implementation. To that end, they proposed a method to define the functional usability features (e.g. feedback support, user input error prevention, go back) in a model-driven approach using what they call conceptual primtives. This method includes the indentificaion of usability mecanisms, changes in conceptual models and in the compilers. Measures are also proposed for early usability evaluation considering these conceptual primitives (Panach *et al*. 2011). To interpret the result of the measures, they assign categorical values (e.g. very good, good, bad) for range of numerical values based on guidelines in the literature. However, the classification of measure results into categories still remains intuitive and is not correlated with the end user view. Considering this scenery, we plan to work on a definition of a framework for predictive usability evaluation from early design models based on measures, that can be interpreteds using the perception of the users about the usability of final user interfaces. No extra effort is required in the UI design to allow the quality evaluation.

Next sections present some ideas about how to perform predictive usability evaluation and a preliminary investigation on this direction.

5.3.1.1 Definition of an Approach for Predictive Usability Evaluation

The context we envision is that UI designers can, during UI modeling and early design phases, apply the specific measures related to that model and have some indication of the level of usability of the final UI using pre-defined correlations. The new UI being developed and evaluated can later also have its measures integrated in the historical base of evaluations, thus improving the correlations.

To that end, we have planned to define a framework that consists of a set of measures applicable to different UI design models from a high abstraction level to a concrete level, that is the implementation (**Oliveira, 2010**). The prediction of usability and the analysis of measures results are based on the possible correlations between predictive usability measures (on the design artifacts) and the results of usability tests performed by users (on the final UI). The correlations will be defined with several empirical studies.

Figure 5.1 portrays the proposed predictive usability prediction approach as well as the way it can supplement inspection-based evaluations. Predictive usability evaluations are performed on various design models using usability predictive measures. The predictive usability evaluation uses a database including the correlations gathered from previous empirical experiments. Each experiment defines and refines continuously the correlations between the results of usability measures collected from early UI design models and the results of inspection-based evaluations made on the final UI. A predictive usability analyzer tool (PredictUse Analyzer) will support the measures collection and calculation as well as the usability prediction that uses the database.

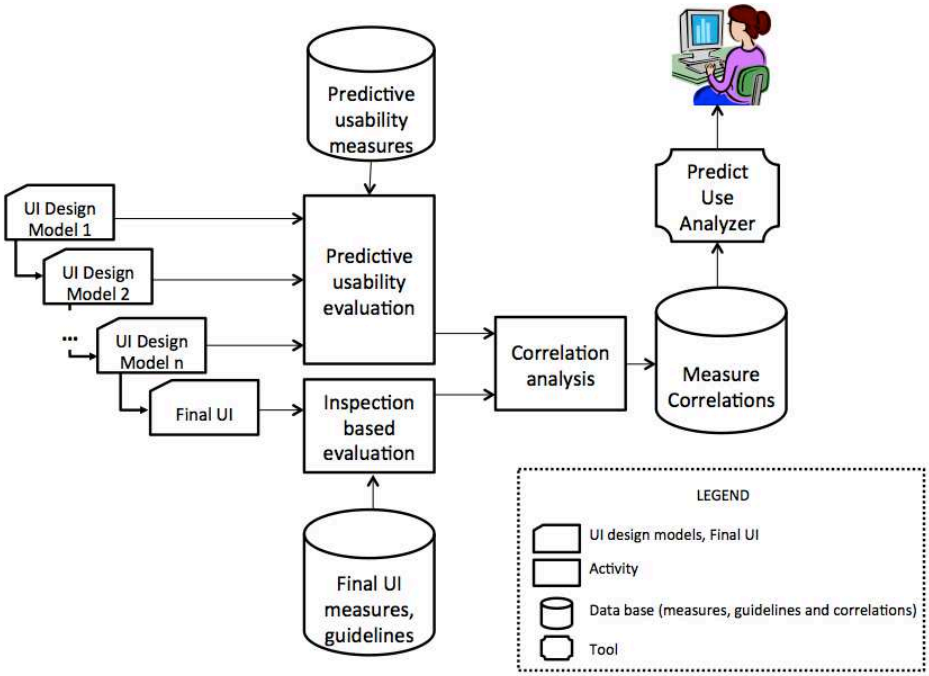


Figure 5.1. Predictive usability evaluation

To address this goal, I plan the following research studies:

- Selection of usability criteria from literature that will be used for predictive usability evaluation. Moreover, selection of the design models to be evaluated and then identify which criteria can be evaluated for each design model. We are aware that not all criteria can be evaluated for all UI design models;

- Definition of which usability evaluation method (survey, model based method, inspection based method and usability tests (see Sears and Jacko, 2009)) should be used for each criteria and each UI design model;
- Definition of formal experimentation protocols to be used in a context of evaluation of software products. These protocols should be defined in a way that they could be specialized considering the specific products (UI design model and final user interface) under evaluation;
- Realization of several empirical studies to collect measures and establish the measures correlations;
- Use of statistical methods to identify correlations between measures from high-level of abstraction to low-level. These correlations will be useful later to define the thresholds to support decision-making in future projects;
- Definition of interdependences between the proposed measures to evaluate specific products and, definition of procedures to support the analysis of the results. At the beginning, interview with experts and literature review can be used. Nevertheless, with data resulting from software evaluation I plan to use ground theory (Strauss and Corbin, 1998) to formally define a theory about this interdependences. Grounded theory method is a systematic methodology in the social sciences involving the discovery of theory through the analysis of data.

I have started to explore this idea with a colleague in an introduction course of HCI in the two-years formation in Computer Science at the University of Valenciennes. The results are presented in next section.

5.3.1.2 A preliminary investigation

We had surveyed different quality models that propose a list of measures. Our starting point was the QUIM model (Seffah *et al.*, 2006) since it is defined considering several quality models and it organizes a set of usability measures. While investigating those measures, we noted that most of the measures were applicable only to final UI since the main goal of QUIM is the definition of quality in use that means, while using the UI. We decided therefore to analyze also the internal measures of usability (applied to a non-executable software product) from ISO9126. The ISO9126 external measures (applied to an executable software or system) has already been included in QUIM. Finally, we also analyzed the ergonomic criteria from the most cited usability model, defined by Scapin and Bastien (1997).

Our procedure was first to read the measure definition (description, formula and scales when available). Next, considering the features of each design model, we identified if that measure could be applied to that design model even that the formula customization would be needed. Finally, we analyzed the measures against the three UI design models proposed in the CAMELEON framework (Calvary *et al.*, 2003):

- Task model - that describes the various users' tasks to be carried out with a UI. Concur Task Tree (CTT) (Paterno *et al.*, 1997) is the most used notation to define this model;
- The abstract User Interface (AUI) - that defines the rendering of the task model independently from any style of interaction (e.g. graphical interaction, vocal interaction, speech synthesis and recognition, video-based interaction, virtual, augmented or mixed reality) as well as the capability and constraints of the platform;
- The concrete user interface (CUI) – that concretizes an abstract UI into an interactor-dependent expression for a given context of use;
- Final UI – that is the operational user interface running in a specific platform.

To do the analysis, we took into account the features of each one of the UI models. For instance, the task model defined using CTT, is composed basically of a hierarchy of different kind of tasks (abstract, interaction, user and application tasks) and temporal relationships among tasks (interleaving, synchronization, enabling, enabling with information passing, choice, deactivation, iteration and optional).

Table 5.1 presents some measures extracted from QUIM. We discovered that some measures require the execution of the UI (e.g. temporal efficiency, error frequency), therefore, it will be applied to the final UI only. Others are related to the UI design itself (e.g. layout complexity, local density), are thus related to the CUI and some of them to AUI. Finally, some measures are related to the information to be presented in the UI (e.g. visual coherence, number of commands/tasks whose terminology is familiar to the user), thus they can be applied in all UI design models from the highest level of abstraction (the task model).

Table 5.1. Example of analysis of measures

Measures	Description	Task model	AUI	CUI	Final UI
Essential Efficiency (Constantine, Lockwood, 1999)	It is a measure of how closely a given user interface design approximates the ideal expressed in the essential use case.	✓		✓	✓
Function understandability (ISO/IEC 9126-2,2001)(ISO/IEC 9126-3,2001)	What proportion of the product functions will the user be able to understand correctly?	✓		✓	✓
Human Efficiency (Bevan and Macleod, 1994)	It is a measure of the human efforts (either mental or physical) expended in relation to the effectiveness	✓		✓	✓
Interface Appearance Customizability (ISO/IEC 9126-2,2001)	What proportion of interface elements can be customized in appearance to the user's satisfaction?			✓	✓
Interface shallowness (Yamada et al., 1995)	It measures the degree of heaviness of the cognitive load on users.	✓	✓	✓	✓
Layout Appropriateness (Sears, 1993)	It measures the appropriateness of a given layout and is computed by weighting the cost of each sequence of actions by how frequently the sequence is performed.		✓	✓	✓
Layout Complexity (Tullis, 1984)	It is the extent to which the arrangement of items on a screen follows a predictable visual scheme.		✓	✓	✓
Local Density (Tullis, 1984)	It measures the percentage of the space used within each individual group of items.		✓	✓	✓
Longest Depth	Depth is the measure of the path traversed in the information hierarchy from one point to other.	✓	✓	✓	✓
Temporal Efficiency (Bevan, Macleod, 1994) or Task Productivity (ISO/IEC 9126-4,2001)	It is a measure of the user time spent to achieve certain level of effectiveness or is the measure of how productive is the user.				✓
Visual Coherence (Constantine, Lockwood, 1999)	It measures how well a user interface keeps related things together and unrelated things apart. It is based on the principle that well structured interfaces group together components that present closely related concepts.	✓	✓	✓	✓

After analyzing measurable criteria (considering QUIM measures, SQUARE standard, and Scapin and Bastien criteria), we started the customization of them to evaluate each one of the models. Table 5.2 presents the analyses of some measures applicable to task model. We note that we have qualitative and quantitative measures.

The qualitative measures are collected by inspections with UI designers. Those inspections are cognitive walkthrough based (Polson *et al.*, 1992; Mahatody *et al.,* 2010), since the assessment is done by experts who should wonder the potential perception of the user in the use of UI implemented based on that task model.

Table 5.2. Some proposed measures

Measure	Purpose	Measurement
Function understandability	What proportion of the product functions will the user be able to understand correctly?	Subjective evaluation with five point ordinal scale (1-5)
Longest Depth	What is the maximum path to execute a task?	Computes the number of nodes in the CTT hierarchy from the root till the primitive task
Terminology familiarity	What is the number of commands/tasks whose terminology is familiar to the user?	Objective: 100 - number of tasks which terminology is unfamiliar to evaluator)/number of tasks % Subjective: 1(non Familiar) – 5 (Familiar)
Percentage grouping	What is the percentage of decomposed tasks?	(Number of tasks decomposed in at least one primitive task) / total number of tasks * 100
Size of group	What is the average size of group of tasks?	Average of (total of primitive task / Number of tasks decomposed in at least one primitive task)
Visual Coherence	How well grouped are the tasks in the abstract tasks?	100 - Number of primitive task that are not closely related in the same high level task / number of primitive tasks defined * 100
Evident functions	What proportion of the product functions the UI designer expert believes will be evident to the user?	Subjective evaluation with a five point ordinal scale
Immediate Feedback	What proportion of tasks represents the feedback?	(Number of application tasks or Number of decomposed application tasks) + Number of interactive task for feedback) / Number of tasks decomposed in at least one primitive task *100
Legibility	How clear is the text of information that will be required or displayed?	Subjective evaluation with a five point ordinal scale (1-5)
Minimal Actions	What is the minimum path of access to primitive task?	Compute the minimum path of access to primitive task
Information Density	What is the information density?	Maximum and minimal number of input information needed per abstract task
		Average percentage of tasks of order independent among the decomposed tasks that has order independence relationships
Consistency	How consistent are the terms used in different tasks?	100-Number of terms used with different meanings in the definition of tasks / number of terms used in the definition of tasks*100
Significance of Codes	What proportion of codes has a clear meaning?	(100 - Number of codes with no clear meaning) /Number of codes * 100
Input validity data	What proportion of input data is validated?	(Number of sequence of "Enabling with information passing" + number of tasks explicitly defined to validate the data)/ Number of tasks is decomposed in at least one primitive task
User Control	How well the user can control his/her interaction with the system?	Number of temporal relation used for navigability among decomposed tasks (Enabling with information passing, enabling, deactivation) / Number temporal relationships * 100. We consider that the use of a temporal relation in one direction means backward and forward
		Number of temporal relationships that allows free choice to the user (independent concurrence, order independence, concurrence with exchange, choice) / Number temporal relationships * 100

The quantitative measures can be calculated from the data obtained from task models tools (e.g. IdealXML and CTTE).

It is important to highlight that each one of those measures is not complete, that means, each measure is defined to measure what can be measured in the task model considering only the elements of this model. However, in other models and in the final UI other aspects of the same measure should be evaluated. For example, the measure defined for information density considers only the aspect number of input/output in the same task since this task will probably be designed in the same panel/window (if we consider graphical UI). However, when evaluating this measure in an AUI or in CUI we can look for the organization of those input/output related to the whole space of the panel/window.

We also highlight that some transversal features cannot be evaluated in a task model since they are usually not modeled. These transversal features include: help, interface customization aspects, specific aspects of navigability such as home, exit, etc.

Finally, we note that our idea is not to define an interpretation of each measure. This will be done by comparing the result of evaluation for a specific project with other results plotted in a graph of correlation points. This graph correlates measures of models with the evaluation of final UI.

Table 5.3 shows the results of the measures collected for the task model published in (Molina *et al.,* 2005) (Figure 5.2).

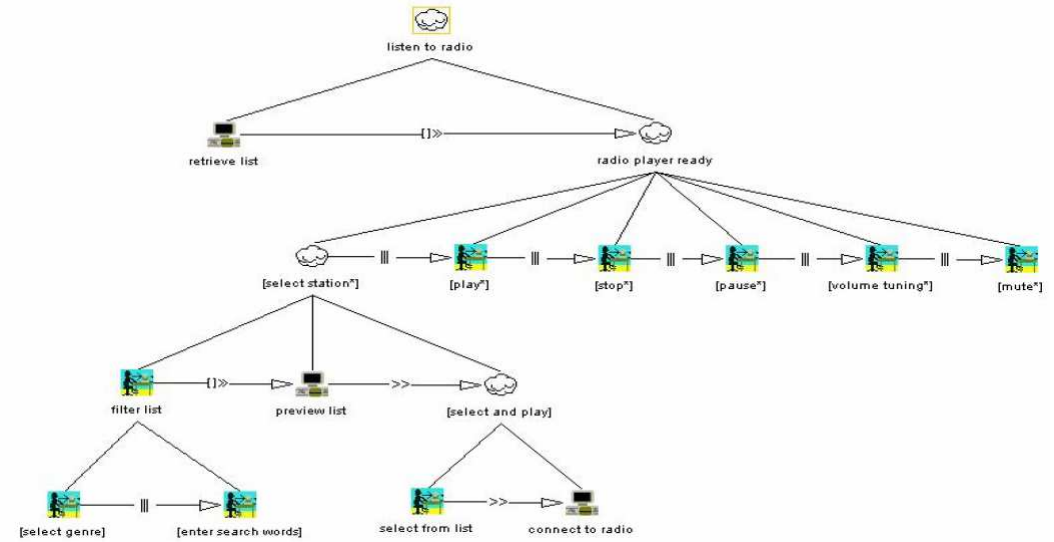


Figure 5.2. Task model for the radio player (Molina *et al.*, 2005)

To allow the prediction evaluation we need to compare these results with historical data of previous evaluations where correlations were discovered between the evaluation of the task models and the final UI.

The definition of a database of correlation takes time since we need several UI designs of several projects, the definition of the UI based on those models and the evaluation from UI design models and final UI.

Table 5.3. Resuls of measures for the “Radio Player”

Measure	Value
Function understandability	5-All
Longest Depth	4
Terminology familiarity	Objective : 100% 5 – Familiar to the domain
Percentage grouping	5/16*100 = 31.25 %
Size of group	11/2 = 5.5
Visual Coherence	100-0/11= 100%
Evident functions	5-All
Immediate Feedback	(3+0)/5*100= 60%
Legibility	Input : 5- Very easy Output: 5-Very easy
Minimal Actions	1
Information density:	
Max number of Inputs	5
Minimum number of Inputs	1
Average of order independence task	5
Consistency	100-0= 100%
Significance of Codes	0
Input validity data	2/5=0.4
User Control:	
Navigability	4/10*100 = 40%
Flexibility	6/10 * 100 = 60%

However, to evaluate the feasibility of our idea, we developed a case study with students of the second year of the two-year graduate course on computer science at the University of Valenciennes. Although we had very little data (collected from 14 projects done by students during the course) for an adequate predictive evaluation, we would like to show an example of the proposed approach. Figure 5.3 presents a graph that shows the evaluation of the task model (x axis) for the measure input validity data and the evaluation of the same measure in a final UI (y axis) (we used a scale from 1 (input never validated) to 5 (input always validated)). It shows that when the task model has a good evaluation the final UI tends to have a better evaluation too. We obtained a value of Spearman correlation equals to 0.46, which is not very good, but that we considered acceptable since we do not have a lot of points (only 14 projects) and the subjects (i.e., the students) had very little experience in task modeling and programming. By looking at the graph, we could infer that the measure result for the task model evaluation for the “Radio Player” (input validity data = 0.4) should give a final user perception around 2.

This preliminary study showed us that there a lot of work need to be perform to effectively validate this approach since we must have UI design models and the final UI developed based on those models. That means, that many controlled experiments have to be performed to generate enough data for statistical analysis of correlations.

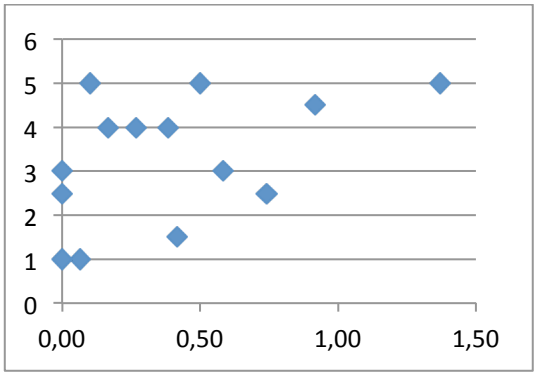


Figure 5.3. Evaluation of UI design model x Final UI for input validity data

5.3.2 Quality Evaluation of HCI in Ubiquitous Systems

As previously defined (chapter 3, § 3.3.3), we have noted in our research on quality evaluation of HCI in ubiquitous systems, that many other aspects are important in the evaluation of HCI than the traditional ones concerned about usability.

Hassanein and Head (2003) proposed that the usability for ubiquitous systems is determined by the elements about its user, environment, task and interface. This idea comes from the intrinsic characteristic of being context-aware. Ubiquitous systems must be able to capture the context and use it to guide their behavior and assist users in performing their tasks. Abowd and Mynatt (2000) stated that the context should consider: *who* is the user and the other people in the environment where the system is used; *what* to consider about human activity to be able to provide useful information; where the user is while using the system; *when* the system is used and the relative changes in time as an aid for interpreting human activity; and, the *why* the user performs certain actions, understanding "why" that person is doing it. According to (Hassanein and Head, 2003), new characteristics for usability evaluation should be taken into account considering the types and limitations of the user, environment, task and interface, as follows:

- The user - it consist of the analyses of novice and experts, and their capacity of performing sequential or multi-tasking. The user limitation is related to his/her memory, visual and motor skill capacities;
- The environment - the influences of the static/dynamic environment, that considers if the user is using an application while being stationary or while moving around; and the quiet/noisy environment, which implies the audio/visual interferences. Regarding environment limitations, the authors quoted the distractions due to various stimuli that will compete with the user's attention, the ambient conditions (e.g. poor/high luminance, extreme temperatures) and time (e.g. time pressure);
- The task – that can be opened/closed or accessing/authoring tasks. Open tasks are more explanatory, vague and non-specific than closed tasks that have a specific objective. Task limitations that influence ubiquitous systems are the complexity and interaction level required by the tasks;
- The interface – several interfaces types (hierarchical menu, short-cut codes, tree-based, table-based, speech based) and their limitations (screen size and quality, input methods and varied devices) have to be considered.

According to Postlad (2009), the complexity of HCI for ubiquitous systems increases because of the following reasons:

- The need to use tasks as part of activities requiring access services that are in various devices;
- The use of the devices by different kind of users;
- Users are engaged in multiple concurrent activities;
- Users are engaged in activities that may occur in various physical environments;
- The activities may be shared among the participants;
- The activities may occasionally be suspended and resumed; and,
- The complexity of determining correctly the user's context

Thus, new challenges for HCI emerge when taking into account the explicit and implicit interaction in ubiquitous environment (Abowd, 1999; Postlad, 2009), such as:

- Ubiquitous system support with information from a limited context in order to improve user experience and satisfaction;
- Integration between the ubiquitous devices, the environment and the user;
- Development of a process for user-centered systems;
- Optimization of energy in ubiquitous devices;
- Minimization of the explicit user interaction with the system;
- Better system adaptation to provide users with information and services relevant to their needs at the desired time;
- Implementation of mechanisms for automatic acquisition of contextual information.

Furthermore, an important difficulty of testing a context-aware application is that part of its behavior depends of the context (Tse *et al.*, 2004); therefore, it is necessary to preview all relevant changes in the context and determine when those changes can impact the behavior of the system application (Wang *et al.*, 2007). Lu (2009) proposed four challenges for testing context-aware applications:

- Rich formats - context data can be captured dynamically from a variety of different sources. The data are then represented in formats and rich semantic hierarchies. Testers need to carefully replicate or simulate context values in different formats supported by the application design as test inputs;
- Volatile contexts – the context can vary rapidly. So when running a software system using a particular context to derive its response to the user, the context may have already been replaced by a new context. Thus, unlike a traditional application in which is produced the same result for the same input value, the input from a context-awareness application can generates different executions according to the different contexts;
- Sensitivity to egocentric context - computational entities are usually modeled as services and have egocentric behavior, that means they take into account only the context information they need;
- Soft Computing - physical contexts are by nature imprecise. When a software application generates a result, it is difficult to determine if it is (i) correct, (ii) wrong due to software failure, or (iii) wrong due to inaccuracy of context information.

On the other hand, in their essence, ubiquitous systems are open distributed systems (Baresi *et al.*, 2006), volatile (Kindberg and Fox, 2002; Coulouris *et al.*, 2011), heterogeneous (Satyanarayanan, 2001) and focused on users (Weiser, 1991) with two main characteristics: spontaneous interoperability and integration with the physical world (Kindberg and Fox, 2002). This physical word includes a large diversity of computing devices range from simple sensors to large computers, through watches, smartphones, tablets and laptops, which have different features (e.g. accelerometer, camera and global positioning system) and varied

computational capabilities (e.g. memory size, processing power and screen resolution). Besides the heterogeneity of computational devices, there is heterogeneity of development platforms (e.g. Sun Microsystems JME, Google Android, .NET Compact Framework and Apple iOS SDK), programming language (e.g. Java, C#, Objective-C), execution platforms (e.g. Linux, Windows Mobile and iOS) and communication technology (e.g. Bluetooth, Zigbee, RFID, Wi-Fi). Additionally, these computing devices, platforms, programming languages and communication technologies are provided by different vendors, who in their great majority adopt different standardizations.

In summary, performing quality assessment of ubiquitous systems is not a trivial task. We should take into account all the particularities of these kinds of system previously described (e.g. context information, diversity of devices, different contexts of use). A specific methodology for the evaluation of these systems is, therefore, necessary.

In our experience of evaluating ubiquitous systems (**Santos *et al.*, 2013**; Santos 2014), we used several procedures to collect and interpret the results, as follows:

- Set log files for each system to be evaluated to collect the data to compute some of the measures defined; that means, to collect the automatic and quantitative data;
- Define questionnaire forms to evaluate the qualitative information to be answered by the users after the use of the application system; that means, applying subjective methods to obtain the users' appreciation of the system (qualitative data);
- Define questionnaires for the software developers to learn about the system information to be developed;
- Observation of the users while executing the tasks to evaluate the system; and,
- Define test cases to be executed to allow the quality assessment.

Table 5.4 presents some examples of measures used in our case studies and the method we used to collect the measures to support the quality evaluation. To collect these measures, specific test scenarios were defined and executed.

Table 5.4. Examples of measures and methods used for quality evaluation

Quality characteristic	Measure	Description	Method used for evaluation
Context awareness	Correction of adaptation	The extent to which the adaptation is correct for the current context of the user	Automatic by the log that register the adaptation performed + Observation to confirm if the adaptation performed is the one expected for that context
	Mean time between context changes	Time frequency between context changes?	Automatic collection by the log files
Calmness	Courtesy	The extent to which interaction occurs in the best possible ways.	Questionnaire answered by users
Transparency	Application proactiveness	The extent to which the application can override the user's actions with the use of sensors?	Questionnaire answered by developers

Considering this previous experience and the specificities of ubiquitous systems, we envisage the definition of a methodology as presented in Figure 5.4 portraits the overview of what we plan to define in the methodology to evaluate HCI for ubiquitous applications. This methodology will be the combination of the existing methods (inspection-based methods, usability questionnaires, usability tests), in a well-defined process that establishes all the activities to be performed, techniques and methods to be used, the outcomes of activities and guidelines about how to integrate the whole information to support the decision-making about the quality of the HCI of the ubiquitous system in evaluation. The basis of this methodology is the definition of formally planned tests that consider the information of different situations of context and allow to collect the data to obtain measures.

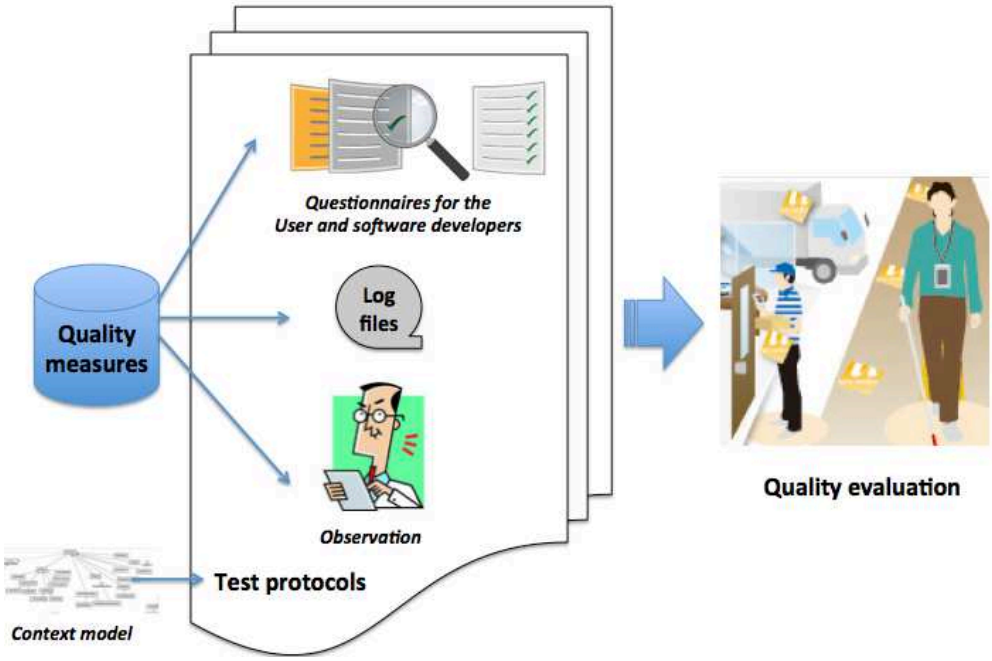


Figure 5.4. Overview of quality evaluation of HCI in ubiquitous systems

First of all, we plan to define measures for the set of quality characteristics defined in previous research (see chapter 2, § 3.3.3, Table 3.11). To that end, we will organize the 26 characteristics identified in a hierarchical quality model as defined by SQuaRE (ISO/IEC 25000, 2005, see chapter 2, § 2.4). Then, measures will be defined for the sub-characteristics. For each measure, procedures for collection and interpretation will be set. For while, considering our previous experience, we plan to collect the data for these measures with:

- Questionnaires to be answered by the users and software developers (for instance, user's satisfaction; or, number of functions the system can perform without the explicit user's interaction);
- Quantitative data obtained in the structured log files that register all inputs, output and changes that the system performs during the whole time of HCI;
- By observation of the use of system.

All these methods to collect data should be correctly planned in the test protocols. These protocols should set the potential contexts of use of the application to run the tests. To that end, a context model should be used. From the context model previously defined (see chapter 3, § 4.5.1.1), we plan to define a context model specific for ubiquitous systems applications. Moreover, the protocol should also specify test requirements, schedule, responsible, and introduce which measures will be collected in each situation, where to record the results and

how to support their analysis. Once all these aspects are defined, the quality evaluation can be performed by asking users to use the system.

Ubiquitous systems developed in LAMIH will be used in the case studies. Currently, two systems have already be developed: CATS (Context-Aware Transportation Services) and a road traffic simulator. The CATS proposes a service-oriented architecture able to compose applications out of services. This makes our framework flexible and allows for easier adaptation to context changes through the use of a Context Manager for all services (instead of having each service or application monitor the context) (Popovici et al., 2012). The road traffic simulator is available on *TangiSense* tabletop in two versions: centralized (on one tabletop), distributed (on several tabletops) (Kubicki *et al*, 2013). This tabletop allows different users to manipulate virtual and tangible objects. The tangible objects are equipped with RFID tags (Kubicki *et al.*, 2012).

The complete definition of this envisioned methodology with empirical validation is a long-term project and requires several research studies.

5.3.3 Towards Knowledge Management *of* and *for* Software Quality Assurance

Based on the learning experience of applying knowledge management practices for SQA presented in Chapter 1, we have concluded that SQA is a knowledge-rich activity including explicit documented knowledge in standards, quality measures and techniques as well as tacit knowledge of each individual who participates in the quality evaluation process (evaluators, users and other stakeholders in the process).

By analyzing my research, I see two important aspects. First, most of the work I have done in the knowledge management area is related to the capture, organization (as ontology concepts, lessons learned, best practices, etc.), dissemination and use of the knowledge to support SQA activities, mainly, quality improvement. In general, I have worked on the "explication" of knowledge. Only the work with learning histories (see Chapter 4, §4.4) tried to capture and disseminate the knowledge considering it as resulting of some action that should be interpreted in new contexts, considering knowledge as a process.

Second, as already discussed, the work I performed was focused in using knowledge management issues for SQA, that is, to support the identification of problems, correct them and improve the software product and processes. In all the work I performed, I considered the individuals (stakeholders) as either (i) "owners" of the domain knowledge to what the system is being developed (e.g. cardiologists, experts in transportation system), or (ii) "owners" of the experiences performing activities of software development and maintenance (for instance, software designers and programmers). In the first case, I tried to elicit the knowledge and organize it in domain ontologies to support the generation of better products focusing on some quality aspects (e.g. ontology for cardiology (**Oliveira *et al.,* 2004**), for transportation (**Oliveira *et al.,* 2013***)*, for telecommunication (**Maidantchick *et al.,* 2000**)). In the second case, I tried to collect what the individuals learned about the software process activities and about the software products, to be possible to make better next time (lessons learned about effort estimate (**Kolsloski and Oliveira, 2005**) and in maintenance projects (**Anquetil *et al.,* 2007**)).

Nevertheless, I did not look for these individuals as "owners" of knowledge about the SQA domain itself. I did not consider that the stakeholders of the SQA activities carry their own knowledge, that their experiences in quality evaluation, their experiences in the organization, the scope of their expertise, and their personal knowledge could also influence the quality assessment.

We know that even while evaluating using measures, the thresholds are not enough and the analysis of an expert for their interpretation is required (**Oliveira *et al.*, 2012**). Both the interpretation of measures and the evaluation according to quality characteristics can vary greatly from evaluator to evaluator, even when the evaluators have the same level of expertise. This is not only because of the subjective nature of an evaluation but also due to differences of tacit knowledge of each evaluator. Thus, some questions emerged:

- How to explain the differences in the quality evaluations performed by the stakeholders? What do they take into account in their decisions?
- Is it possible to capture/transfer knowledge from experts in evaluation to novices?
- How to take into account the knowledge brought by evaluators to support the SQA activities?

These questions are not easy to answer and involve a deep analysis about the activity of performing an assessment itself before analyzing the assessment of software product and processes. Furthermore, we believe we should also analyze the idea of knowledge creation ability introduced by Tsuchiya (1993). According to him, tacit knowledge is the interpretation of information by an individual by activating his/her interpretative framework. Next section presents some concepts related to the notion of tacit knowledge creation before describing some ideas about this research perspective.

5.3.3.1 Tacit Knowledge and Interpretative Framework

As presented in chapter 2 (§ 2.5.1) we usually find in the literature the definition of knowledge by differentiating data (or datum), information and knowledge. Following this trend, Tsuchiya (1993) defines that:

> "When datum is *sense-given* through *interpretative framework*, it becomes information, and when information is *sense-read* through interpretive framework, it becomes knowledge (p. 88)"

This definition highlights three concepts: sense-giving, sense-reading and interpretative framework.

Sense-giving and sense-reading were introduced by Polany in 1967 (Polany, 1967). According to Polany, when one uses "words" to describe an experience, s-he performs a sense-giving action. One's experience, perception and tacit knowledge are expressed as explicit knowledge in the "words". On the other hand, when someone reads these words and interprets them, s-he performs a *sense-reading* action, eventually, generating new tacit knowledge. Tsuchiya (1993) states that the execution of these actions uses one's interpretative framework.

Arduin (2013) organized these concepts as presented in Figure 5.5. When a person P1 structures his/her tacit knowledge and transfers it, P1 creates information. When a person P2, collects data from that information and absorbs it, P2, potentially, creates a new tacit knowledge. Therefore, knowledge is a result of the interpretation of some information by an individual. This interpretation is made through an interpretative framework, which filters the data contained in the information, and with the use of pre-existing tacit knowledge considering the context, the situation and intentions. In this way, tacit knowledge is transferred from one person to another.

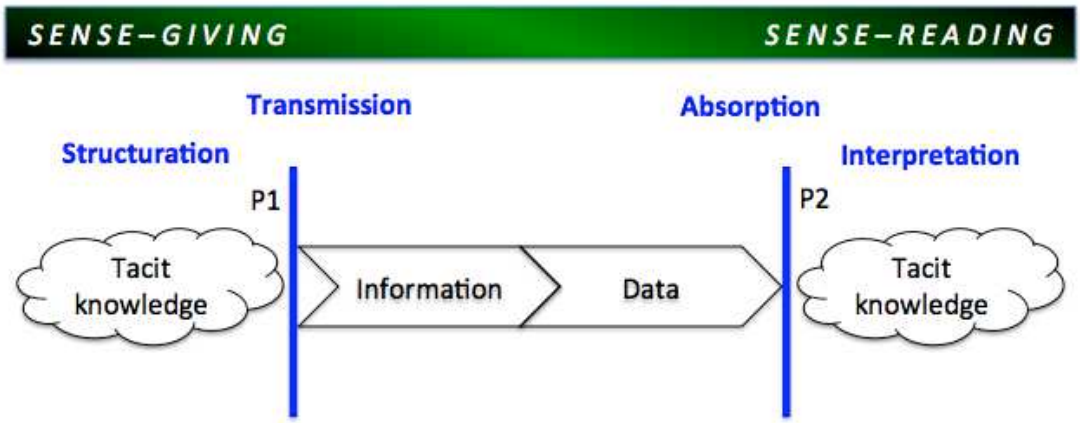


Figure 5.5. Tacit Knowledge transfer (Arduin, 2013)

Once we recognize the existence of an interpretative framework, it is important to understand the concept of commensurability of the interpretative frameworks. Grundstein (2012) defined commensurability as "the common space of the set of interpretative frameworks of each member (e.g. cognitive models, mental models directly forged by education, experience, beliefs, and value systems)". In other words, if the probability that two individuals give the same meaning to the same information is strong, we say that their interpretative frameworks have a strong commensurability or are commensurable. Otherwise, if the probability is low, we say that their interpretation schemes have low commensurability or are incommensurable (Arduin *et al.*, 2013). According to Tsuchiya (1993) commensurability of interpretative frameworks of members is essential for tacit knowledge sharing.

5.3.3.2 Exploring knowledge about SQA

As previously presented, this research perspective looks for exploring knowledge management concepts in benefit of the quality evaluation activity itself. In other words, we would like to investigate how to take into account the stakeholders' knowledge used in the quality evaluation to support the SQA.

We are aware that to answer this question we should work with the two main categories of knowledge:

i) Explicit knowledge - the know-how about SQA, the knowledge that can be formalized in documents, artifacts or in a repository. Some proposals in that direction consider the definition of a software quality ontology (Kayed *et al.*, 2009; Bertoa *et al.*, 2006) and software process ontology (Palomäki and Keto, 2006; Wongthongtham *et al.*, 2006);
ii) Tacit knowledge – embodied in stakeholders' minds and that can come from routines performed in the organization (collective knowledge) and personal knowledge (skills, knowledge of the history of the organization, etc.) when doing SQA activities. How to profit of this knowledge to assist in the SQA itself?

For the first category ((i) explicit knowledge), it is necessary to study the process of knowledge management in terms of quality assurance in order to answer what knowledge to acquire, where to get them, how to use and maintain, how to transfer the knowledge from quality evaluation, and what happens during the SECI (socialization, externalization, combination and internalization) cycle (see Chapter 2, §2.5.1) in the quality assurance process.

We should also, identify the knowledge that can be explained and formalized; that is, organize the knowledge used in the quality assurance and, analyze how it is performed by understanding the reasoning behind the evaluation activity. The organization of this knowledge can be done using domain ontology(ies), or knowledge repositories with guidelines, lessons learned, measures, etc. (in connection with the other research perspectives previously described).

However, the most difficult challenge of this research perspective is to investigate the second category ((ii) tacit knowledge) to really understand how knowledge can be transferred to support SQA activities. To address this challenge, we identify two branches of research.

The first one is to investigate what the stakeholders take into account while performing a qualitative evaluation (for example, from qualitative measures) or while interpreting results of quantitative evaluation (from quantitative measures). In this context, we should investigate why we can have differences even when the evaluators have the same level of expertise in quality evaluation procedures. An idea in this direction is to investigate potential differences in their interpretative framework (as defined in previous section).

To that end, several empirical studies of quality evaluation should be performed (here, again the idea is to take advantage of the experiments planned in the previous research perspectives). Evaluators with different levels of expertise, and different levels of experience will participate in these studies. Then, we should analyze the results of the subjective evaluations and the interpretation of quantitative data collected for the objective measures trying to understand the interpretative framework used by each one. This is an exploratory research aiming at gathering preliminary information that will help suggest hypothesis for further investigations.

The second branch of research is to understand how we could transfer the knowledge of the experts in evaluation to novices. It seems that when we identify a high commensurability in the empirical studied performed, it could be possible to efficiently share the knowledge used by the evaluators to perform the evaluation. On the other hand, low commensurability could help to understand what should be taken into account in an evaluation, and what is needed to perform it.

The main result of this research is the definition of an approach that integrates knowledge management of SQA to support SQA activities.

5.4 Conclusion

This document has presented the main research I have developed after finishing my doctorate. Working in different countries and universities I had the opportunity to collaborate with professionals from different domains (e.g. computer science, management, psychology). All work carried out have aimed to search for software quality assurance, a research area that interested me since my Master in computer science.

SQA is a dynamic area that must respond to the continuous evolution of technology to ensure the best customer satisfaction. Therefore, a lot of work has still to be done. I described in this document some of the research perspectives I plan to work in the next years. However, new challenges will also emerge, and new research will be necessary to address these challenges.

References

Abbas, K. Système d'accès personnalisé à l'information : application au domaine medical. Doctoral Thesis, INSA de Lyon, 2008.

Abi-Char, P. E., Mhamed, A., El-Hassan, B., Mokhtari, M. A Flexible Privacy and Trust Based Context-aware Secure Framework. 8th international conference on Smart homes and health telematics, Seoul, Korea, June 2010, pp. 17-23.

Abowd, G. D., Dey, A. K., Brown, P.J, Davies, N., Smth, M., Steggles, P. Towards a Better Understanding of Context and Context-Awareness. International symposium on Handheld and Ubiquitous Computing, Karlsruhe, Germany, September 1999, pp. 304-307.

Abowd, G., Mynatt, E.D. Charting Past, Present, and Future Research in Ubiquitous Computing. ACM Transactions on Computer-Human Interaction, v. 7, n. 1, March 2000, pp. 29-58.

Abrahão, S., Insfran, E. Early Usability Evaluation in Model-Driven Architecture Environments. 6th IEEE International Conference on Quality Software, Beijing, Chine, October 2006, pp. 287-294.

Abran, A., Khelifi, A., Suryn, W., Seffah, A. Usability Meanings and Interpretations in ISO Standards. Software Quality Journal, v. 1, n. 4, 2003, pp. 325-338.

Abran, A., Nguyenkim, H. Analysis of Maintenance Work Categories Through Measurement. Conference on Software Maintenance, Sorrento, Italy, October 1991, pp. 104-113.

Agarwal, R., Kumar, M. Yogesh, S., Mallick, S., Braradwaj, R. M., Anantwar, D. Estimating Software projects, ACM SIGSOFT, July 2001, pp. 60-67.

Agresti W. Lightweight Software Metrics: The P10 Framework. IT Professional, 8(5), 2006, pp. 12-16.

Al Balushi, T.H.,Sampaion, P.R.F. Loucopoulos, P. Eliciting and prioritizing quality requirements supported by ontologies: a case study using the ElicitO framework and tool. Expert systems, v. 30, issue 2, 2013, pp. 129 - 151.

Albert, B., Tullis, T., Tedesco, D. Beyond the usability lab. Amsterdam, The Netherlands: Morgan Kaufmann, 2010.

Anderson, J., Fleak, F., Garrity, K., Drake, F. Integrating usability techniques into software development. IEEE Software, v. 18, n. 1, January-February 2001, pp. 46-53.

Andrade, E. L. P., Oliveira, K. M. Aplicação de Pontos de Função e Pontos de Casos de Uso de Forma Combinada no Processo de Gestão de Estimativa de Tamanho de Projetos de Software Orientados a Objetos. IP. Informática Pública, v. 7, n. 1, 2005, pp.13-30.

Andrade, E.L.P., Oliveira, K. M. Uso Combinado de Análise de Pontos de Função e POntos de Caso de Uso na Gestão de Estimativa de Tamanho de Projetos de Software Orientado a Objetos. III Simpósio Brasileiro de Qualidade de Software, Brasilia, Brazil, June 2004, pp. 234-248.

Anquetil, N., Oliveira, K. M., Dias, M. Software Maintenance Ontology. Ontologies for Software Engineering and Software Technology ed.Heildelberg: Springer, 2006, v. 1, pp. 153-173.

Anquetil, N., Oliveira, K. M., Dias, M., Ramal, M., Menezes, R. Knowledge for Software Maintenance. Software Engineering and Knowledge Engineering, San Francisco, USA, July 2003, pp. 6-68.

Anquetil, N., Oliveira, K. M., Souza, K. D. De, Dias, M. Software Maintenance Seen as a Knowledge Management Issue. Information and Software Technology, v. 49 (5), 2007, pp. 515-529.

Arduin, P-E, Doctoral Thesis, Vers une métrique de la commensurabilité des schemas d'interprétation, Paris-Dauphine University, 2013.

Arduin, P.E., Doan, Q-M., Grigori, D., Grim, M., Grunstein, M., Negre, E., Rosenthal-Sabroux, C., Thion, V., Evaluation d'un système d'information et de connaissance - De l'importance de la prise en compte de la connaissance. INFORSID'2012, INFormatique des ORganisations et Systèmes d'Information et de Décision, Montpellier, France, May 2012, pp. 371-378.

Arduin, P.E., Grunstein, M., Rosenthal-Sabroux, Évaluer la prise en compte des connaissances tacites dans un système d'information : vers un système d'information et de connaissance, Ingénierie des Systèmes d'Information, 18(3), 2013, pp. 121-148.

Arnaut, W., Oliveira, K. Lima, F. OWL-Soa: A Service Oriented Architecture Ontology Useful During Development Time And Independent From Implementation Technology, 4th IEEE International Conference on Research Challenges in Information Systems, Nice, France, May 2010, pp. 527-536.

Asato, R., Mesquita, S. M., Costa, I., Farias, S. W. H. Alignment between the business strategy and the software processes improvement: A roadmap for the implementation. Portland International Conference on Management of Engineering & Technology, Portland, OR, October 2009, pp. 1066-1071.

Assila A., Ezzedine H., Oliveira K., Bouhlel M. Towards improving the subjective quality evaluation of Human Computer Interfaces using a questionnaire tool. International Conference on Advanced Logistics and Transport, Sousse, Tunisia, May 2013, pp. 275-283.

Assila, A., Oliveira, K.M.; H.Ezzedine, K.M. Towards qualitative and quantitative data integration approach for enhancing HCI quality evaluation, 16th International Conference on Human-Computer Interaction, Springer, 22-27, June 2014, Creta, Greece, June 2014. (accepted for publication)

Atterer, R. Where Web Engineering Tool Support Ends: Building Usable Websites. the 20th Annual ACM Symposium on Applied Computing, Santa Fe, New Mexico, March 2005, pp. 12-17.

Aurum, A., Jeffrey, R., Wohlin, C., Handzic, M. Managing Software Engineering Knowledge, Springer Verlag, Berlin, 2003.

Aversano, L., Esposito, R., Mallardo, T. e Tottorella, M. Supporting Decisions on the Adoption of Re-engineering Technologies. Eighth European Conference on Software Maintenance and Reengineering, Tampere, Finland, March 2004, pp. 95-104.

Bacha, F., Abed, M., Oliveira, K. Providing Personalized Information in Transport Systems: A Model Driven Architecture Approach. First IEEE International Conference on Mobility, Security and Logistics in Transport, Hammamet, Tunisia, June 2011d, pp. 452-459.

Bacha, F., Intégration de la personnalisation du contenu dans la conception et la génération des applications interactives basée sur une approche MDA, Doctoral Thesis, University of Valenciennes, May 2013.

Bacha, F., Oliveira K., Abed M. Foundations of a Model Driven Engineering Approach for Human-Computer Interface Focused on Content Personalization. The 11th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems, Valenciennes, France, September 2010.

Bacha, F., Oliveira, K., Abed, M. A model driven architecture approach for user interface generation focused on content personalization. Fifth IEEE International Conference on Research Challenges in Information Science, Guadeloupe - French West Indies, France, May 2011c, pp. 1-6.

Bacha, F., Oliveira, K., Abed, M. Context-Aware MDA Approach for Content Personalization in User Interface Development. Diaz V.G., Lovelle J.M.C., Garcia-Bistelo B.C.P, Martinez O.S., Progressions and Innovations in Model-Driven Software Engineering, IGI Global, 2013, pp. 88-105.

Bacha, F., Oliveira, K., Abed, M. Supporting models for the generation of personalized user interfaces with UIML. Software Support for User Interface Description Language, UIDL'2011, Interact'2011 workshop, Lisbon, Portugal, Septembre 2011b.

Bacha, F., Oliveira, K., Abed, M. Using Context Modeling and Domain Ontology in the Design of Personalized User Interface. International Journal on Computer Science and Information Systems (IJCSIS), 6, 2011a, pp. 69-94.

Barcellos, M.P., Falbo, R.; Rocha, A.R. A Well-Founded Software Process Behavior Ontology to Support Business Goals Monitoring in High Maturity Software Organizations, 14th IEEE International Enterprise Distributed Object Computing Conference Workshops, Vitória, Brazil, October 2010, pp. 253-262.

Baresi, L., Di Nitto, E., Ghezzi, C., Toward Open-World Software: Issue and Challenges, Computer , v. 39, n. 10, , October 2006, pp. 36-43.

Basili V.R., Caldiera G., Rombach H.D. The Goal Question Metric Approach, Encyclopedia of Software Engineering, Wiley, 1994a.

Basili, V., Caldiera, G.H. Rombach. D. The Experience Factory, Encyclopedia of Software Engineering, John Wiley & Sons, 1994b, pp 469–76,

Basili, V., Costa, P., Lindval, M., Mendona, M., Seaman, C., Roseanne, T., Zelkowitz, M.. An experience management system for a software engineering research organization. 26th Annual NASA Goddard Software Engineering Workshop. NASA Goddard Space Flight Center, 2001.

Bennett, K.; Ramage, M.; Munro, M. Decision Model for Legacy Systems. IEE Proceedings Software, v. 146, n. 3, June 1999, pp. 153-159.

Berander P, Jönsson P. A goal question metric based approach for efficient measurement framework definition. International Symposium on Empirical Software Engineering, Rio de Janeiro, Brazil, September 2006, pp. 316-325.

Bertoa, M., Vallecillo, A., An Ontology for Software Measurement. Ontologies for Software Engineering and Software Tecnology ed.Heildelberg, Springer 2006, pp. 175-196.

Biggerstaff, T.J., Mitbander, B.G, Webster, D. Program understandingand the concept assignment problem, Commun. ACM 37 (5), 1994, pp. 72–83.

Birk, A., Dingsoyr, T., Stalhane, T. Postmortem: Never Leave a Project Without It, IEEE Software, v. 19, n. 3, 2002, pp. 43-45.

Bjørnson, F.O., Dingsøyr, T. Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used, Information and Software Tecnology, v. 50, n. 11, October 2008, pp. 1055-1068.

Booch, G., Rumbaugh, J., Jacobson, I. UML - Guia do usuário. Editora CAMPUS, 2000.

Bradbury, H. and Mainemelis, C. Learning history and organizational praxis. Journal of Management Inquiry, 10(4) , 2001, pp. 340–357.

Briand, L. C., Basili, V., Kim, Y., Squier, D. R. A Change Analysis Process to Characterize Software Maintenance Projects. The International Conference on Software Maintenance, Victoria, Canada, September 1994, pp. 38-49.

Briand, L.C., Daly, J.W., Wust, J.K. A unified framework for coupling measurement in object-oriented systems, IEEE Transactions on Software Engineering, v. 25, 1999, pp. 91-121.

Brito, M. C. A., Nóbrega, G.M., Oliveira, K. M. Integrating instructional material and teaching experience into a teachers' collaborative learning environment. European Conference on Technology Enhanced Learning. Lecture Notes in Computer Science, n. 4227, Crete, Greece, October 2006, pp. 458-463.

Brito, M., Nóbrega, G.M., Oliveira, K. M. Capturando experiência docente para guiar o design instrucional colaborativo e contínuo. XVI Simpósio Brasileiro de Informática na Educação, Juiz de Fora, Brazil, 2005, pp. 147-157.

Brooke, C.; Ramage, M. Organisational scenarios and legacy systems. International Journal of Information Management, 2001, pp. 365-384.

Brossard, A., Abed, M., Kolski, C. Taking context into account in conceptual models using a Model Driven Engineering approach. Information and Software Technology, 53 (12), 2011, pp. 1349-1369.

Brown, A., Earl, A., McDermid, J., Software Engineering Environments: Automated Support for Software Engineering; McGraw-Hill Book Company, 1992.

Calazans, A.T.S., Oliveira, K. M., Santos, R. Medição de Tamanho para Sistemas de Data Mart. III Simpósio Brasileiro de Qualidade de Software, Brasília, Brazil, June 2004b, pp. 384-398.

Calazans, A.T.S., Oliveira, K. M., Santos, R. R. Adapting Function Point Analysis to Estimate Data Mart Size. 10th IEEE International Symposium on Software Metrics, Chicago, USA, September 2004a, pp. 300-311.

Caldwell, B., Cooper, M., Reider, L. G., Vanderheiden, G. Web Content Accessibility Guidelines 2.0: W3C Candidate Recommendation 11, December 2008. (http://www.w3.org/TR/WCAG20/) (Accessed April 10, 2014)

Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J. A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers, v. 15, n. 3, June 2003, pp. 289–308.

Campo, M. Why CMMI maturity level 5?, CrossTalk, v. 25, 2012, pp. 15-18.

Cappiello, I., Puglia, S., Vitaletti, A. Design and Initial Evaluation of a Ubiquitous Touch-Based Remote Grocery Shopping Process. International Workshop on Near Field Communication, Hagenberg, Austrian, February 2009, pp. 9-14.

Carazza, A., Mello, W., Oliveira, K. M. Avaliação da Qualidade de Software: um estudo de caso. Workshop de Qualidade, Rio de Janeiro, Brazil, June 2001, pp.183–188.

Card, S.K., Moran, T.P. and Newell, A. The Psychology of Human-Computer Interaction. Erlbaum, Hillsdale, NJ, 1983.

Chandrasekaran, B., Josephson, J. R., Benjamins V. R. What Are Ontologies, and Why Do We Need Them?. IEEE Intelligent Systems & their applications, v. 14, n. 1, 1999, pp. 20-26.

Chang, Y.-H., Lin, B.-S. An Inquiry-based Ubiquitous Tour System. International Conference on Complex, Intelligent and Software Intensive Systems, Seoul, July 2011, pp. 17-23.

Charfi, S., Trabelsi, A., Ezzedine, H., Kolski, C. Widgets dedicated to user interface evaluation. International Journal of Human-Computer Interaction, v. 30, n. 5, 2014, pp. 408-421.

Cheng, B.H.C., Lemos, R., et. al.: Software Engineering for Self-Adaptive Systems: A Research Roadmap. Software Engineering for Self-Adaptive Systems, Springer Berlin Heidelberg, 2009, pp. 1-26.

Cherfi, S. S., Ayad, S., Comyn-Wattiau, I. Improving Business Process Model Quality Using Domain Ontologies, Journal on Data Semantics, v. 2, 2013, pp. 75-87.

Chevalier, M. et Julien, C. Interface coopérative et adaptative pour la recherche d'information sur le web. Chrisment, C., editeur: BDA, 2003.

Chrissis M, Konrad M, Shrum S. CMMI for development: guidelines for process integration and product improvement. 2nd ed. Addison-Wesley, 2011.

Cockton, G., Woolrych, A., Lavery, D. Inspection-Based Evaluations. Sears, A. and Jacko, J. Human-Computer Interaction: Development Process, CRC Press, 2009, pp. 274-292.

Comella-Dorda, S., Wallnau, K., Seacord, R., Robert, J. A Survey of Legacy System Modernization Approaches. Technical Note CMU/SEI 2000-TN-003, April 2000.

Corrêa, G., Figueiredo, R., Oliveira, K. M., Araújo, José Marcelo Pereira de Diretrizes para a Melhoria da Gerência e Desenvolvimento de Requisitos em uma Empresa de Software. Simpósio Internacional de Melhoriade Processos de Software, São Paulo, Brazil, November 2004, pp.95 – 106.

Coulouris, G., Dollimore, J., Kindberg, T., Blair, G. Distributed Systems: Concepts and Design, Addison Wesley, 2011.

Crispim, E., Calixto, I., Brito, M., Nóbrega, G.M., Oliveira, K. M. Um ambiente Web para a captura de experiência docente baseado em Objetos de Aprendizagem e na colaboração: da concepção à prototipagem VIII Simpósio Brasileiro de Informática na Educação, Sào Paulo, Brazil, 2007, pp. 318-327.

Cristal, M., Reis, J. Leveraging lessons learned for distributed projects through Communities of Practice. International Conference on Global Software Engineering, Florianóplois, Brazil, October 2006, pp. 239-240.

Crosby, P.B., Quality is free, McGraw-Hill, New York, 1979.

D'Oliveira, F.M., Oliveira, K. M., Figueiredo, Adelaide. Priorização de Testes de Software: Uma abordagem orientada ao cliente. Simpósio Brasileiro de Qualidade de Software, Fortaleza, Brazil, 2003, pp. 218-232.

Damián-Reyes, P., Favela, J., Contreras-Castillo, J. Uncertainty Management in Context-aware Applications: Increasing Usability and User Trust. Wireless Personal Communications, 2011, v. 56, n. 1, pp. 37–53.

Davenport, T.H., Prisak, L. Working Knowledge: How Organizations Manage What They Know, Harvard Business School Press, Boston, 1998.

De Lucia, A., Fasolino, A.R. e Pompelle, E. A Decisional Framework for Legacy System Management, 2001, pp. 642-651.

De Marco, L., Ferrucci, F., Gravino, C., Sarro, F., Abrahao, S., Gomez, J. Functional versus design measures for model-driven Web applications: A case study in the context of Web effort estimation. 3rd International Workshop on Emerging Trends in Software Metrics, Zurich, Switzerland, June 2012, pp.21-27.

De Moor, K., Ketyko, I., Joseph, W., Deryckere, T., Marez. L., Martens, L., Verleye, G. Proposed Framework for Evaluating Quality of Experience in a Mobile, Testbed-oriented Living Lab Setting. Mobile Networks and Applications, v. 15, n. 3, 2010, pp. 378-391.

Delgado, A., Weber, B., Ruiz, F., Guzman, I.G., Piattini, M. An integrated approach based on execution measures for the continuous improvement of business processes realized by services, Information and Software Technology, vol 56, Issue 2, February 2014, pp 134-162.

Dias, M., Anquetil, N., Oliveira, K. M. Organizing the Knowledge Used in Software Maintenance. Journal of Universal Computer Science, v. 9, 2003, pp.641-658.

Dingsøyr , T. Moe, N.B., Moe, N. Augmenting experience reports with lightweight postmortem reviews. PROFES, Lecture Notes in Computer Science, n. 2188, Kaiserslautern, Germany, September 2001, pp.167-181.

Dingsøyr, T. Conradi, R. A survey of case studies of the use of knowledge management in software engineering. International Journal of Software Engineering and Knowledge Engineering, 12 (4), 2002, pp. 391–414.

Dixon, N.M. Common Knowledge: How Companies Thrive by Sharing What They Know. Harvard Business School Press, 2000.

DOT, DOT/SSA/CT-91/1, Software Quality Metrics, US Department of Transportation, August 1991.

Doucet, A., Lumineau, N., Berrut, C., Denos, N., Rumpler, B., Rocacher, D., Boughanem, M., Soule-Dupuy, C., Mouaddib, N. et Kostadinov, D. Action Specifique sur la Personnalisation de l'Information. Groupe MRIM - CLIPS-IMAG, 2004.

Dumas, J.S., Fox, J.E. Usability Testing: Current Practice and Future Direcstions. Sears, A., Jacko, J., Human-Computer Interaction: Development Process, CRC Press, 2009, pp. 231-251.

Dupuy-Chessa, S., Oliveira, K.M, Si-Said Cherfi, S. Qualité des modèles : retour d'expériences. INFORSID'2014, INFormatique des ORganisations et Systèmes d'Information et de Décision, Lyon, France, May 2014. (accepted for publication).

Duran, R., Marinho, M., Figueiredo, R., Oliveira, K. M. Institucionalização da Gerência de Configuração no Desenvolvimento de Software de uma Organização. VII Simposio Internacional de Melhoria de Processo de Software, São Paulo, Brazil, Novembro 2005, pp. 60-85.

Emam, K. E., Drouin, J. N., Melo, W. SPICE – The Theory and Practice of Software Process Improvement and Capability Determination, IEEE Computer Society, Edwards Brothers Inc., Estados Unidos, 1998.

Emam, K. The ROI from Software Quality, 1a ed.: Auerbach Publications, 2005.

Erdinc, O., Lewis, J.R. Psychometric Evaluation of the T-CSUQ: The Turkish Version of the Computer System Usability Questionnaire. International Journal of Human and Computer Interaction, vol 29(5), 2013, pp. 319-326.

Evers, V. et al. Interacting with Adaptive Systems. Interactive Collaborative Information Systems, Springer, 2010, pp. 299–325.

Fagan, M.E., Advances in Software Inspections, IEEE Transactions on Software Engineering, v. SE-12, n. 7, July 1986, pp. 744-751.

Fenton, N., Pfleeger, S. Software Metrics A Rigorous & Practical Approach, 2nd. Ed., PWS Publishing Company, 1997.

Ferreira, A.I.F., Santos, G., Cerqueira, R., Montoni, M., Barreto, A., Rocha, A. R., Barreto, A. O. S, Silva Filho, R. C. ROI of software process improvement at BL informática: SPIdex is really worth it, Software Process Improvement and Practice, v. 13, 2008, pp. 311-318.

Fikes, R., Farquhar, A. Distributed Repositories of Highly Expressive Reusable Ontologies. IEEE Intelligent Systems & their applications, v. 14, n. 2, March/April 1999, pp. 73-79.

Fink, M., Obendorf, H. Scenario-based usability engineering techniques in agile development processes. ACM - CHI Extended Abstracts on Human Factors in Computing Systems, Florence, Italy, April 2008, pp. 2159-2166.

Gabillon Y., Lepreux S., Oliveira K. Towards ergonomic User Interface composition: a study about information density criterion. M. Kurosu, 15th International Conference on Human-Computer Interaction, HCI International, Lecture Notes in Computer Science, Las Vegas, USA, July 2013, pp. 211-220.

Gabillon, Y., Petit, M., Calvary, G., and Fiorino, H. Automated planning for user interface composition. 2nd International Workshop on Semantic Models for Adaptive Interactive Systems (SEMAIS'11) of the International Conference on Intelligent User Interfaces, Palo Alto, CA, USA, 2011.

Galin, D. Software Quality Asssurance. Pearson Addison-Wesley, UK, Harlow, 2003.

Galinac, T. Empirical evaluation of selected best practices in implementation of software process improvement," Information and Software Technology, v. 51, 2009, pp. 1351-1364.

García, J., Amescua, A., Sanchez-M-I, Bermón, L., Design guidelines for software processes knowledge repository development, Information and Software Technology, v. 53, issue 8, August 2011, pp. 834–850.

Garía-Barrios, V., Mödritscher, F. Gütl, C. Personalisation versus Adaptation? A User-centred Model Approach and its Application. In: K. Tochtermann, & H. Maurer (eds.). International Conference on Knowledge Management, Graz, Austria, 2005, pp. 120-127.

Glavinic, V., Ljubic, S., Kukec, M. A Holistic Approach to Enhance Universal Usability in m-Learning. The Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, Valencia, Spain, September/October 2008, pp. 305-310.

Gomes, A., Machado, L. F., Oliveira, K. M., Rocha, A.R., Software Process Improvement Through Measurement And Experts Judgement. The 3rd European Software Measurement Conference - (Federation of European Software Metrics Associations), 2000.

Gomes, A., Oliveira, K. M., Rocha, A.R. Métricas para Medição e Melhoria de Processos de Software. 4o Encontro de Qualidade nas Tecnologias de Informação e Comunicações, 2001b, Lisboa, Portugal, March 2001b, pp. 71- 78.

Gomes, A., Oliveira, K. M., Rocha, A.R., Avaliação de Processos de Software baseada em Medições. XV Simposio Brasileiro de Engenharia de Software, Rio de Janeiro, RJ, October 2001a, pp. 84-99.

Grammenos, D., Akoumianakis, D., Stephanidis, C. Integrated support for working with guidelines: the Sherlock guideline management system. Interacting with Computers, vol 12(3), 2000, pp. 281 – 311.

Gruber, T., 1993, A Translation Approach to Portable Ontology Specifications, Knowledge Acquisition, 5(2), 1993, pp. 199-220.

Grundstein, M. Three Postulates That Change Knowledge Management Paradigm, New Research on Knowledge Management Models and Methods, Huei-Tse Hou (Ed.), InTech, 2012.

Guarino, N. Formal Ontology and Information System, In: Guarino, N. (ed.) Formal Ontology in Information System, 3-15, IOS Press, 1998.

Haapalainen, E., Kim, S., Forlizzi, J. F., Dey, A. K. Psycho-Physiological Measures for Assessing Cognitive Load. ACM International Conference on Ubiquitous computing, Copenhagen, Denmark, September 2010, pp. 301-310.

Hagen P., Manning, H. Souza, R. Smart Personalization. Forrester Research, USA, 1999.

Hall, T., Fenton N. Implementing effective software metrics programs. IEEE Software, v. 14, n. 2, 1997, pp. 55-65.

Hardin, M., Hom, D., Perez, R., Williams, L. Quel diagramme ou graphique vous convient le mieux? Copyright Tableau Software, Inc. 2012.

Harris, K., Mcroy, L. Use a Health Check to Determine Your Application's 'Fitness for Duty. Gartner Research. 2006.

Harrison, R., Counsell, S.J., Nithi, R.V. An evaluation of the MOOD set of object-oriented software metrics. IEEE Transactions on Software Engineering, v. 24, n. 6, 1998, pp. 491- 496.

Harrison, W., Cook, C. Insights on Improving the Maintenance Process Through Software Measurement. Conference on Software Maintenance, San Diego, USA, November 1990, pp. 37-45.

Hassanein, K., Head, M. Ubiquitous Usability: Exploring Mobile Interfaces within the Context of a Theoretical Model. Workshop on Ubiquitous Mobile Information and Collaboration Systems Workshop, The 15th Conference on Advanced Information Systems Engineering (CAiSE), Klagenfurt/Velden, Austria, June 2003.

Helms, J., Schaefer, R., Luyten, K., Vermeulen, J., Abrams, M., Coyette, A. Vanderdonckt, J. Human-Centered Engineering with the User Interface Markup Language. Seffah, A., Vanderdonckt, J., Desmarais, M. (eds.). Human-Centered Software Engineering, Chapter 7, HCI Series, 2009, pp. 141-173.

Henderson-Sellers, B., Gonzalez-Perez, C., Mcbride, T., Low, G. An ontology for ISO software engineering standards: 1) Creating the infrastructure. Journal on Computer Standards & Interfaces archive, v. 36, n. 3, March 2014, pp. 563-576.

Henry, S. L. 2005. Introduction to Web Accessibility. http://www.w3.org/ WAI/intro/accessibility.php (Accessed April 10, 2014).

Heravi, B.R., Lycett, M., Cesare, S. Ontology-based standards development: Application of OntoStanD to ebXML business process specification schema. International Journal of Accounting Information Systems, 2014.

Holzinger, A. Usability Engineering Methods for Software Developers. Communications of the ACM, v. 48, n. 1, 2005, pp. 71-64.

Holzinger, A. Usability Engineering Methods for Software Developers. Communications of the ACM, v. 48, n. 1, January 2005, pp. 71-64.

Hornbæk, K., Høegh, R. T., Pedersen, M. B., Stage, J. Use Case Evaluation (UCE): A Method for Early Usability Evaluation in Software Development. 11th International Conference on Human-Computer Interaction, Rio de Janeiro, Brazil, September 2007, pp. 578-591.

Ianzen, A., Mauda, E.C., Paludo, M.A., Reunehr, S., Malucelli, Software process improvement in a financial organization: An action research approach, Computer Standards & Interfaces, v. 36, issue 1, November 2013, pp. 54–65.

IEEE, IEEE Std 610.12-1990 – IEEE Standard Glossary of Software Engineering Terminology, the Institute of Eletrical and Electronics Engineers, USA, New York, 1990.

INFORSID, La recherche en systèmes d'information et ses nouvelles frontières. Ingénierie des Systèmes d'Information, v. 17/3, 2012, pp. 9-68.

Ioannidis, Y. E., Koutrika, G. Personalized systems: Models and methods from an ir and db perspective. In Bohm, K., Jensen, C. S., Haas, L. M., Kersten, M. L., Larson, P.A. and Ooi, B. C., editeurs, VLDB, 1365, 2005.

Iqbal, R. Sturm, J.A., Terken, J.M.B. Wang, C. User-centred design and evaluation of ubiquitous services. the 23rd annual international conference on Design of Communication: Documenting and Designing for Pervasive Information, ACM SIGDOC, 2005, pp. 138-145.

ISO 9241-11. Ergonomic requirements for office work with visual display terminals - Part 11: Guidance on Usability, 1998.

ISO 9241-171. Ergonomics of human-system interaction — Part 171: Guidance on software accessibility, 2008.

ISO 9241-151. Ergonomics of human-system interaction — Part 151: Guidance on world wide web user interfaces, 2008.

ISO 9241-920:2009. Ergonomics of human-system interaction -- Part 920: Guidance on tactile and haptic interactions, 2009.

ISO/IEC 12207. Systems and Software Engineering - Software Life Cycle Processes, 2008.

ISO/IEC 14598. Information Technology: Software Product Evaluation – Part 1: General Overview, 1999.

ISO/IEC 14764. Information technology – Software Maintenance. Technical Report 14764, Joint Technical Commitee International Standards Organization/International Electrotechnique Commission, 1999.

ISO/IEC 15504-1. Information technology - Process assessment - Part 1: Concepts and Vocabulary, November 2004.

ISO/IEC 15504-2. Information technology — Process assessment — Part 2: Performing an assessment, October 2003.

ISO/IEC 15504-3. Information technology — Process assessment — Part 3: Guidance on performing an assessment. January 2004.

ISO/IEC 15939. System and Software Engineering – Measurement Process, second edition, 2007.

ISO/IEC 25000. ISO/IEC FDIS 25000, Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE), January 2005.

ISO/IEC 25022. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of quality in use, July 2012.

ISO/IEC 25023. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of system and software product quality, August 2011.

ISO/IEC 9126. Software engineering - Product quality - Part 1: Quality model, 2001.

ISO9000:2000. Systèmes de management de la qualité -- Principes essentiels et vocabulaire, Quality management systems – Fundamentals and vocabulary, December, 2000.

Itaborahy, A., Oliveira, K. M., Santos, R. Value-Based Software Project Management: a business perspective on software projects, 10th International Conference on Enterprise Information Systems, Bracelona, Spain, June 2008, pp. 218-225.

Itaborahy, A., Radis, E., Longhi, F., Oliveira, K. M., Figueiredo, R. Aplicação do método SCAMPI para avaliação do processode gerenciamento de projetos de software numa instituiçãofinanceira. VII Simposio Internacional de Melhoria de Processo de Software, São Paulo, Brazil, November 2005, pp.1-26.

Ivory, M. Y., Hearst, M. A. The state of the art in automating usability evaluation of user interfaces. ACM Computing Surveys, v. 33, n. 4, December 2001, pp. 470-516.

Jafari, S., Mtenzi, F., O'Driscoll, C., Fitzpatrick, R., O'Shea, B. Privacy Metrics in Ubiquitous Computing Applications. International Conference for Internet Technology and Secured Transactions (ICITST), London, UK, November 2010, pp. 1-2.

Jermakovics, A., Scotto, M., Succi, G. Visual Identification of Software Evolution Patterns. Ninth international workshop on Principles of software evolution: in conjunction with the 6th ESEC/FSE joint meeting, Dubrovnik, Croatia, 2007, pp. 27-30.

Ji, N. Continuous Quality Improvement (CQI) Model Based on Knowledge Management for Software International Conference on Enterprise. Information Management, Innovation Management and Industrial Engineering, Shenzhen, China, November 2011, pp. 247-251.

Jia, L., Collins, M., Nixon, P. Evaluating Trust-Based Access Control for Social Interaction. International Conference on Mobile Ubiquitous Computing, Systems, Services, and Technologies, Sliema, Malta, October 2009, pp. 277-282.

Juran, J.M., Godfrey, A.B. Juran's Quality Control Handbook, McHraw-Hill, New York, 1990.

Kallepalli, C., Tian, J., Measuring and modeling usage and reliability for statistical Web testing. IEEE Transactions on Software Engineering, v. 27, n. 11, November 2001, pp.1023-1036.

Kaplan, R.S, Norton, D. P. The Balanced Scorecard: measures that drive performance. Harvard Business Review, January/February 1992, pp. 71–80.

Kayed, A. Hirzalla, N. Samhan, A.A. Alfayoumi, M. Towards an Ontology for Software Product Quality Attributes. Fourth International Conference on Internet and Web Applications and Services, Venice/Mestre, Italy, May 2009, pp. 200-204.

Kemp, E. A., Thompson, A-J., Johnson, R. S. Interface Evaluation for Invisibility and Ubiquity: An Example from E-learning. 9th ACM SIGCHI New Zealand Chapter's International Conference on Human-Computer Interaction: Design Centered HCI, 2008, pp. 31-38.

Kerzazi, N., Lavallée, M. Inquiry on usability of two software process modeling systems using ISO/IEC 9241. 24th Canadian Conference on Electrical and Computer Engineering (CCECE), Niagara Falls, Canada, May 2011, pp. 773-776.

Khoshgoftaar, T.M., Allen, E.B., Halstead, R., Trio, G.P., Flass, R.M. Using process history to predict software quality, Computer, v. 31, n. 4, April 1998, pp. 66-72.

Khurshid, N., Bannerman, P. L. and Staples, M. Overcoming the first hurdle: Why organizations do not adopt CMMI. International Conference on Software Process, Lecture Notes in Cumputer Science, n. 5543, Vancouver, Canada, May 2009, pp. 38-49.

Kim, G., Lee, M., Lee, J., Lee, K., Design of SPICE experience factory model for accumulation and utilization of process assessment experience. Third ACIS International Conference on Software Engineering Research, Management and Applications, 2005, August 2005, pp. 368-374.

Kim, H. J., Choi, J. K., JI, Y. Usability Evaluation Framework for Ubiquitous Computing Device. 3rd International Conference on Convergence and Hybrid Information Technology, Busan, South Korea, November 2008, pp. 164-170.

Kindberg, T., Fox, A. System software for ubiquitous computing. IEEE Pervasive Computing, v. 1, issue 1, January/March 2002, pp. 70-81.

Kitchenham, B., Dyba, T., Jorgensen, M. The value of mapping studies – A participant-observer case study. In 14th International Conference on Evaluation and Assessment in Software Engineering, Keele, UK, April 2010, pp. 1-9.

Kitchenham, B.A., Pfleeger, S.L., Pickard, L.M., Jones, P.W., Hoaglin, D.C., El Emam, K.; Rosenberg, J., Preliminary guidelines for empirical research in software engineering, IEEE Transactions on Software Engineering, v. 28, n. 8, August 2002, pp.721-734

Kitchenham, K., Charters, S. Guidelines for Performing Systematic Literature Reviews in Software Engineering. EBSE Technical Report, 2007.

Kleiner, A., and Roth, G.L. Learning about Organizationl Learning – Creating a Learning History. MIT Center for Organizational Learning, 1995.

Knight, J. and Myers, E.A. Improved inspection technique, Communications of the ACM, 1993, pp. 51- 61.

Ko, I.-Y., Koo, H.-M. Jimenez-Molina, A. User-centric Web Services for Ubiquitous Computing. Advanced Techniques in Web Intelligence, Springer, 2010, pp. 167–189.

Kobsa, A. Generic user modeling systems. User Modeling and User-Adapted Interaction, 11(1), 2001, pp. 49-63.

Kolski, C., Ezzedine, H, Gervais, M.P, Oliveira, K.M., Seffah, A. Evaluation des SI; Besoins en méthodes et outils provenant de l'ergonomie et de l'IHM, INFORSID'2012, INFormatique des ORganisations et Systèmes d'Information et de Décision, Montpellier, France, May 2012, pp. 395-410.

Kolski, C., Uster, G., Robert, J., Oliveira, K., David, B. Interaction in mobility: the evaluation of interactive systems used by travellers in transportation contexts. International Conference, HCI International, Lecture Notes in Computer Science, n. 6763, Orlando, USA, July 2011, pp. 301-310.

Koskinen, J., Ahonen, J., Sivula, H., Tilus, T, Lintinen, H. e Kankaanpaa, I. Software Modernization Decision Criteria: An Empirical Study. Ninth European Conference on Software Maintenance and Reengineering, Manchester, UK, March 2005, pp. 324-331.

Kosloski, R.A., Oliveira, K. M. An Experience Factory to Improve Software Development Effort Estimates. 6th International Conference Product Focused Software Process Improvement, Lecture Notes in Computer Science, n. 3597, Oulu, Finland, June 2005, pp. 560-573.

Kosloski, R.A.D., Oliveira, K. M. Melhoria Contínua de Estimativa de Esforço para o Desenvolvimento de Software: Uma Abordagem sobre Produtividade. V Simpósio Brasileiro de Qualidade de Software, Vila Velha, Brazil, June 2006, pp. 409-423.

Kourouthanassis, P. E., Giaglis, G. M., Karaiskos, D. C. Delineating the Degree of Pervasiveness in Pervasive Information Systems: An assessment framework and design implications. Pan-Hellenic Conference on Informatics, Samos, Grece, August 2008, pp. 251-255.

Kroeger, T.A., Davidson, N.J., Cook, S.C., Understanding the characteristics of quality for software engineering processes: A Grounded Theory investigation. Information and Software Technology, vol 56, issue 2, pp. 252-271, February 2014.

Kryvinska, N., Strausss, C., Zinterhof, P. Variated Availability Approach to the Services Manageable Delivering. Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), Seoul, Korea, June/July 2011, pp. 106-113.

Kubicki S., Lebrun Y., Lepreux S., Adam E., Kolski C., Mandiau R. Simulation in Contexts Involving an Interactive Table and Tangible Objects. Simulation Modelling Practice and Theory, 31, 2013, pp. 116–131.

Kubicki, S., Lepreux, S., Kolski ,C. RFID-driven situation awareness on TangiSense, a table interacting with tangible objects. Personal and Ubiquitous Computing, 16 (8), 2012, pp. 1079-1094.

Kumari, K.A., Saranya, K., Babu, B., Sathasivam, G.S., Ontology for semantic web using personalization, International Conference on Advances in Engineering, Science and Management, Nagapattinam, Tamil Nadu, March 2012, pp.688-693.

Lazić, L., Mastorakis, N. E., Software economics: Quality-based return-on-investment model, Iasi, 2010, pp. 25-39.

Leão, P.R., Oliveira, K. M., Moresi, E.. Ontologia para gestão de competências dos profissionais em Tecnologia da Informação. IV Jornadas Iberoamericanas en Engenieria del Softfware e Ingenieria del Conocimento, 2004, v. 2, pp. 651-655.

Lee, J., Song, J., Kim, H., Choi, J. Yun, M.H. A User-Centered Approach for Ubiquitous Service Evaluation: An Evaluation Metrics Focused on Human-System Interaction Capability. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), n. 5068, 2008, pp. 21–29.

Lee, J., Yun, M. H.: Usability Assessment for Ubiquitous Services: Quantification of the Interactivity in Inter-personal Services. International Conference on Management of Innovation and Technology, Sanur, Bali, June 2012, pp. 718-724.

Lehaney, B., Clarke, S., Coakes, E., Jack, G. Beyond Knowledge Management. Hershey: Idea Group Publishing. 2004.

Lepreux S., Vanderdonckt J., Kolski C. User Interface Composition with UsiXML. Faure D., Vanderdonckt J., Workshop on User Interface Extensible Markup Language UsiXML, Berlin, Germany, June 2010, pp. 141-151.

Lepreux, S., Vanderdonckt, J., and Michotte, B. Visual Design of User Interfaces by (De)composition. 13th Int. Workshop on Design, Specification, and Verification of Interactive Systems DSV-IS'2006, G. Doherty, A. Blandford (eds.). Lecture Notes in Computer Science, n. 4323, Sublin, Ireland, July 2006, pp.157-170.

Lethbridge, T. C., Sim, S. E., Singer, J. Software Anthropology: Performing Field Studies in Software Companies. Consortium for Software Engineering Research (CSER), 1996.

Lewis, J.R. IBM computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use. International Journal of Human Computer Interaction, vol (7), 1995, pp. 57–78.

Liampotis, , Roussaki, I., Papadopoulou, E., Abu-Shaaban, Y., Williams, M.H., Taylor, N.K., McBurney, S.M., Dolinar, K. A privacy framework for personal self-improving smart spaces. 2th IEEE International Conference on Computational Science and Engineering, Vancouver, Canada, August 2009, pp. 444-449.

Lima, R., Sampaio, F., Oliveira, K. M., Rocha, A.R. Evaluating Web Sites For an Educational Environment Target For Cardiology. The 3rd European Software Measurement Conference - Federation of European Software Metrics Associations, 2000.

Lima, S., Lima, F., Oliveira, K. M. Evaluating the Accessibility of Websites to Define Indicators in Service Level Agreements, 11th International Conference on Enterprise Information Systems, Milan, Italy, May 2009, pp. 858-869.

Lima, S., Lima, F., Oliveira, K.M., Towards Metrics for Web Accessibility Evaluation. International Workshop on Design & Evaluation of e-Government Applications and Services, Rio de Janeiro, Brazil, 2007.

Lin, J., Wong, J., Nichols, J., Cypher, A., Lau, T.A. End-user programming of mashups with vegemite. 14th ACM international conference on Intelligent user interfaces, Florida, USA, February 2009, pp. 97-106.

Lindvall, M., Rus, I., Jammalamadaka, R., Thakker, R. Software Tools for Knowledge Management. Technical Reports, DoD Data Analysis Center for Software, Rome, NY, 2001.

Lu, H. A software testing framework for context-aware applications in pervasive computing, Doctoral Thesis, University of Hong Kong, Hong kong. 2009

Luna, E. R., Panach, J. I., Grigera, J., Rossi, G., Pastor, O. Incorporating usability requirements in a test/model-driven web engineering approach. In Journal of Web Engineering, 9, 2010, pp. 132-156.

Macedo, C. C., Lima, S., Rocha, A.R., Natali, A.C., Oliveira, K. M., Mian, P., Barreto, A., Barreto, A., Santos, G., Conte, T. Implantação de Melhoria de Processos de Software no Tribunal Superior Eleitoral. V Simpósio Brasileiro de Qualidade de Software, Vila Velha, Brazil, June 2006, pp. 351-358.

Machado, L. F., Oliveira, K. M., Rocha, A.R. Modelo para Definição de Processos de Software baseado na ISO/IEC12207, em Modelos de Maturidade e Características do Projeto. Terceiro Workshop Ibero-americano de Engenharia de Requisitos e Ambientes de Software, Cancun, Mexico, April 2000b, pp.73-84.

Machado, L. F., Oliveira, K. M., Rocha, A.R. Using Standards And Maturity Models For The Software Process Definition. 4th International Software Quality Week Europe, Brussels, Belgium, November 2000a.

Mahatody, T., Sagar, M., Kolski, C. State of the Art on the Cognitive Walkthrough method, its variants and evolutions. International Journal of Human-Computer Interaction, 26 (8), 2010, pp. 741-785

Maidantchick, C., Oliveira, K. M., Vidal, H., Masiero, M. L. Applying Management Model and Ontology on a Telecommunication Company. 13th International Conference Software & Systems Engineering and their Applications, Paris, France, November 2000.

Marinho, F.G., Andrade, R. M. C., Werner, C. , Viana, W. Maia, M. E. F. Rocha, L. S. Teixeira, E. Filho, J. B. F. Dantas, V. L. L. Lima, F. , Aguiar, S. MobiLine: A Nested Software Product Line for the domain of mobile and context-aware applications. Science of Computer Programming, v. 78 (2), December 2013, pp. 2381-2398.

Mayhew, D.J. The Usability Engineering Lifecycle: A Practitioner's Handbook for User Interface Design, Morgan Kaufman Publishers, 1999.

Mayr, A., Plosch, R., Klas, M., Lampasona, C., Saft, M., A Comprehensive Code-Based Quality Model for Embedded Systems: Systematic Development and Validation by Industrial Projects. 23rd IEEE International Symposium on Software Reliability Engineering, Dallas, USA, November 2012, pp.281-290.

McGarry J, Card D, Jones C, Layman B, Clark E, Dean J, Hall F. Practical Software Measurement: objective information for decision makers. 1st ed. Addison-Wesley: Boston, 2002.

McGibbon, T., Nicholls, D. Making the (Business) case for software reliability, Seattle, WA, 2002, pp. 285-292.

Melo, A. M., Baranauskas, M.C.C., and Bonilha, F.F.G. Avaliação de Acessibilidade na Web com a Participação do Usuário: Um Estudo de Caso. Simpósio Sobre Fatores Humanos em Sistemas Computacionais, Curitiba, Brazil, October 2004, pp. 181-184.

Mnasser, H., Khemaja, M., Oliveira, K., Abed, M. A public transportation ontology to support user travel planning, 4th International Conference on Research Challenges in Information Systems, Nice, France, May 2010, pp. 127-186.

Molina, F., Toval, A. Integrating usability requirements that can be evaluated in design time into Model Driven Engineering of Web Information Systems. Journal Advances in Engineering Software, Oxford, 40, 2009, pp. 1306-1317.

Molina, J.P., Vanderdonckt, J., Montero, F., González, P. Towards Virtualization of User Interfaces based on UsiXML. Web3D 2005 Symposium, 10th International Conference on 3D Web Technology, Bangor, UK, March-April 2005, pp. 169-178.

Monteiro L., Oliveira K.M. Defining a catalog of indicators to support process performance analysis. Journal of Software Maintenance and Evolution: Research and Practice, vol 23, Issue 6, 2011, pp. 395-422.

Montoni, M., Rocha, A.R, Weber, K. MPS.BR: a successful program for software process improvement in Brazil, Software Process: Improvement and Practice, vol 14, issue 5, September 2009, pp. 289-300.

Muller, N. J. Managing Service Level Agreements. International Journal of Network Management. v. 9, n. 3, May–June 1999, pp. 155-166.

Myers, B. Engineering more natural interactive programming systems: keynote talk. 1st ACM SIGCHI symposium on Engineering interactive computing systems, EICS, Pittsburgh, USA, July 2009, pp. 1-2.

Myers, G.J. The Art of Software Testing. John Wiley & Sons, Inc., Hoboken, New Jersey, 2n edition, 2004.

Nascimento, A. A., Knoth, C. R., Fagundes, J. S., Figueiredo, R., Oliveira, K. M. Aplicações de Práticas Selecionadas do Nível 2 do Modelo eSCM-SP v2 em um Órgão da Administração Pública Federal Brasileira. 5th Conference for Quality in Information and Communications Tecnhology, Porto, Portugal, October 2004, pp.153-159.

Nery, A., Bandeira, L., Lima, F., Oliveira, K. M. Qualidade de Software aplicada à navegabilidade na Web. Simpósio Brasileiro de Sistemas Multimídia e Web, Natal, Brazil, November 2006.

Nielsen, J. Heuristic evaluation, Jakob Nielsen, Mack, R. L. (eds), Usability inspection methods, Heurisitic Evaluation, New York, NY, John Wiley & Sons, Inc, 1994.

Nielsen, J. Usability engineering, Boston, Academic Press, 1993.

Nikov, A., Vassileva, S., Angelova, S. ,Tzvetanova, Stoeva, S. WebUse: An approach for web usability evaluation. 3rd Symposium on Production Research, Istanbul, Turkey, 2003, pp. 511-518.

Nonaka, I., and Takeuchi, H. The Knowledge-Creating Company, Oxford University Press, 1995.

OCG, Office for Government Commerce. ITIL – The Key to Manage IT Services: Service Delivery – Version 1.2; Crow, 2001.

Oliveira K., Thion V., Dupuy-Chessa S., Gervais M., Si-Said Cherfi S., Kolski C. Limites de l'évaluation d'un système d'information : une analyse fondée sur l'expérience pratique. INFORSID, INFormatique des ORganisations et Systèmes d'Information et de Décision, Montpellier, France, May 2012, pp. 411-427.

Oliveira, K. New Research Challenges for User Interface Quality Evaluation. Conference Internationale Francophone sur l'Interaction Homme-Machine, Luxembourg, Septembre 2010, pp. 145-148.

Oliveira, K. M., Gallota, C., Menezes, C., Travassos, G.H, Rocha, A.R. Defining and Building Domain-Oriented Software Development Environments. 12th International Conference Software & Systems Engineering and their Applications, Paris, France, November 1999b.

Oliveira, K. M., Menezes, C., Travassos, G.H, Rocha, A.R. Using Domain-Knowledge in Software Development Environments. Software Engineering and Knowledge Engineering, Kaiserslautern, Germany, June 1999a, pp. 180-187.

Oliveira, K. M., Model for the construction of Domain-Oriented Software Development Environments, Doctoral Thesis, Federal University of Rio de Janeiro, October 1999.

Oliveira, K. M., Rabelo Jr, A., Rocha, A.R., Souza, A., Ximenes, A., Andrade, C., Asanome, C., Onnis, D., Olivaes, I., Lobo, N., Ferreira, N., Werneck, V. An Experience of Software Quality Assurance for an Expert System in Cardiology. 2nd Medical Engineering Week of the World, Taipei, China, May 1996.

Oliveira, K. M., Villela, K., Rocha, A., Travassos, G.H. Use of Ontologies in Software Development Environments. Ontologies for Software Engineering and Software Technology, ed. Heidelberg: Springer, v.1, 2006, pp. 275-309.

Oliveira, K. M., Zlot, F., Rocha, A. R., Travassos, G. H., Gallota, C., Menezes, C. Domain-oriented software development environment. Journal of Systems and Software, v. 172, 2004, pp.145-161.

Oliveira, K., Rosenthal-Sabroux, C. Numéro spécial Évaluation des Systèmes d'information, Ingénierie des systèmes d'information (ISI), 18 (3), Hermes, Paris, ISBN 1633-1311, 2013.

Oliveira, K.M., Bacha, F., Mnasser, H., Abed, M. Transportation Ontology Definition and Application for the Content Personalization of User Interfaces. Expert Systems with Applications, 40 (8), 2013, pp. 3145-3159.

Olsina, L., Rossi, G., Measuring Web application quality with WebQEM, IEEE MultiMedia, v. 9, n.4, October-December 2002, pp. 20-29.

OMG, Model-Driven Architecture Guide Version 1.0.1. 2003.

Orenyi, B. A., Basri, S., Jung, L.T. Object-Oriented Software Maintainability Measurement in the past Decade. International Conference on Advanced Computer Science Applications and Technologies, Kuala, Lumpur, November 2012, pp. 257-262.

Page-Jones, M., Structured system design, Prentice Hall, 2nd edition, 1988.

Palomäki, J., Keto, H. A Process-Ontological Model for Software Engineering, 18th Conference on Advanced Information Systems Engineering - Trusted Information Systems, Workshops and Doctoral Consortium, Luxembourg, June 2006, pp. 720-726.

Panach, J. I., Condori-Fernández, N., Vos, T. E. J., Aquino, N., Valverde, F. Early Usability Measurement in Model-Driven Development: Definition and Empirical Evaluation, International Journal of Software Engineering and Knowledge Engineering, v. 21, n. 3, 2011, pp. 339-365.

Panach, J. I., Juzgado, N. J., Pastor, O. Including functional usability features in a model-driven development method. Computer Science and Information System Journal, v.10, n.3, 2013, pp. 999-1024.

Papazoglou, M. P. Service-oriented computing: Concepts, characteristics and directions. Fourth International Conference on Web Information Systems Engineering, December 2003, pp. 3-12.

Parasuraman, A., Zeithaml, V.A., Berry, L.L. SERVQUAL: A Multiple-Item Scale for Measuring Consumer Perceptions of Service Quality. Journal of Retailing, v. 64, n. 1, 1988, pp. 12-40.

Park R.E., Goethert W.B. e Florac W.A. Goal Driven Software Measurement – a Guidebook, CMU/SEI-96-BH-002, Software Engineering Institute, Carnegie Mellon University, August 1996.

Paterno, F., Mancini, C., Meniconi, S. ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models, INTERACT '97, International Conference on Human-Computer Interaction, Sydney, Australia, July 1997, pp. 362-369.

Pearse, T. Oman, P. Maintainability measurement on industrial source code maintenance activities, International Conference on Software Maintenance, Opio, France, October 1995, pp. 295-303

Peixoto, D.C.C., Bastista, V. A., Resende, R. F. and Pádua, C. I. P. S. A case study of Software Process Improvement implementation, Redwood City, CA, 2010, pp. 716-721.

Pfleeger, S. L. Software Engineering: Theory and Practice. 2nd Edition. New- Jersey: Prentice Hall, 2001.

Pigoski, T.M. Practical Software Maintenance: Best Practices for Software Investment, John Wiley & Sons, Inc., 1996.

Pinna-Dery, A. M., Fierstone, J., Picard E. Component model and programming: a first step to manage human computer interaction adaptation. 5th International Symposium on Human- Computer Interaction with Mobile Devices and Services (Mobile HCI), Lecture Notes in Cumputer Science, n. 2795, Udine, Italy, September 2003, pp. 456-446.

Pino, F., Pardo, C., García, F., Piattini. Assessment methodology for software process improvement in small organizations. Information and Software Technology, v. 52, n.10, October, 2010, pp 479-500.

Polany, Sense-giving and sense-Reading. Journal of the Royal Institute of Philosophy, v. 42, n. 162, 1967, pp. 301-323.

Polson, P., Lewis, C., Rieman, J., & Wharton, C. Cognitive Walkthrough: A method for theory-based evaluation of user interface. International Journal of M Polson an-Machine Studies, 36, 1992, pp. 741–773.

Popovici D., Desertot M., Lecomte S., Delot T. A framework for mobile and context-aware applications applied to vehicular social networks. Journal of Social Network Analysis and Mining, Springer, 2012.

Porto, D., Souza, L., Martinez, M., Anquetil, N., Oliveira, K. M. Avaliação da Confiabilidade de um Software utilizando Aspectos. V Simpósio Brasileiro de Qualidade de Software, Vila Velha, Brazil, June 2006, pp. 334-342.

Poslad, S. Ubiquitous Computing Smart Devices, Smart Environments and Smart Interaction. Wiley. 2009.

Pranata, I., Athauda, R., Skinner, G. Determining Trustworthiness and Quality of Mobile Applications, Mobile Wireless Middleware, Operating Systems, and Applications. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, v. 65, 2013, pp. 192-206.

Probst, G,; Raub, S., Romhardt, K. Managing knowledge: building block for success. Jonh Wiley & Sons. 2000.

Queiroz, A., Anquetil, N. Oliveira, K.M. Avaliação pró-ativa da deterioração de sistemas de informação por meio de medidas de gestão, Revista de Informática Téorica e Aplicada, v. 16, n. 1, 2009, pp. 45-68.

Rabelo Jr, A., Rocha, A.R., Oliveira, K. M., Ximenes, A., Souza, A., Andrade, C., Onnis, D., Lobo, N., Ferreira, N., Werneck, V. An Expert System for the Diagnosis of Acute Myocardial Infarction with EKG Analysis. Artificial Intelligence in Medicine, v. 1, 1997, pp. 75-92.

Ramos C., Oliveira K. M., Rocha A. Towards a strategy for analysing benefits of Software Process Improvement programs. The 25th International Conference on Software Engineering and Knowledge Engineering, Boston, USA, June 2013.

Ramos, C. S., Oliveira, K. M., Anquetil, N. Legacy Software Evaluation Model for Outsourced Maintainer. 8th IEEE European Conference on Software Maintenance and Reengineering, Tampere, Finland, March 2004, pp. 48-57.

Ramos, C., Oliveira, K. M., Rocha, A. Uma Abordagem para Análise de Retorno sobre Investimentos em Programas de Melhoria de Processos de Software. WAMPS, Workshop Anual do MPS.BR, Campinas, Brazil, October 2012, pp. 43-54.

Ramos, C.S., Oliveira, K. M., Anquetil, N. Avaliação de Sistemas Legados. IV Simpósio Brasileiro de Qualidade de Software, Porto Alegre, Brazil, June 2005, pp. 139-144.

Ranganathan, A., Al-Muhtadi, J., Biehl, J., Ziebart, B., Campbell, R.H., Bailey, B. Towards a Pervasive Computing Benchmark. International Conference on Pervasive Computing and Communications Workshops, Kauai Island, USA, March 2005, p. 194-198.

Ransom, J., Sommerville, I. e Warren, I. A method for assessing legacy systems for evolution. 2nd Euromicro Conference on Software Maintenance and Reengineering, Florence, Italy, March 1998, pp. 128-134.

Rico, D.F. ROI of Software Process Improvement: Metrics for Project Managers and Software Engineers: J. Ross Publishing, Inc, 2004.

Rising, L., Patterns in postmortems, 23rd IEEE Annual International Computer Software and Applications Conference, Phoenix, USA, October 1999, pp. 314–315.

Rocha, C., Oliveira, K. M., Rocha, A.R., Montoni, M., Timbo, A., Sampaio, L. C., Rabelo Jr., A. CardioSurgery: An Environment to Support Surgical Planning and Follow-up in Cardiology. WebNet - World Conference on the WWW and Internet, San Antonio, USA, 2000, pp. 459-463.

Rocha, L.S., Filho, J. B. F., Lima, F.F.P., Maia, M.E.F., Viana, W., Castro, M.F., Andrade, R.M.C. Past, Present, and Future Perspectives on Ubiquitous Software Engineering. Journal of Systems and Software, 2012.

Ross, T.; Burnett, G. Evaluating the Human–Machine Interface to Vehicle Navigation Systems as An Example of Ubiquitous Computing. International Journal of Human-Computer Studies, v. 55, n. 4, October 2001, pp. 661–674.

Rubio, J. M. L., Bozo, J. P. Approach to a Quality Process for the Ubiquitous Software Development. Electronics, Robotics and Automotive Mechanics Conference, Morelos, Mexico, September 2007, pp. 701-705.

Rus, M. Lindvall, M. Knowledge management in software engineering, IEEE Software. 19 (3), 2002, pp. 26–38.

Santos R., Oliveira K., Andrade R., Santos I., Lima E. A Quality Model for Human-Computer Interaction Evaluation in Ubiquitous Systems. 6th Latin American Conference on Human Computer Interaction, Lecture Notes in Cumputer Science, n. 8278, Guanacaste, Costa Rica, December 2013, pp. 63-70.

Santos, R. P. Dos, Oliveira K. M. De, Silva, W. P. da., Evaluating the Service Quality of Software Providers appraised in CMM/CMMI, Software Quality Journal, Springer Verlag, v. 17 (3), September 2009, pp. 283-301.

Santos, R. P., Oliveira, K. M., Silva, W. P. Percepção dos Clientes sobre a Qualidade do Serviço de Provedores Formalmente Avaliados nos Modelos CMM/CMMI. Simpósio Brasileiro de Qualidade de Software, Fortaleza, Brazil, Junho 2007, pp. 203-217.

Santos, R. P., Souza, S. M., Lima, S. T., Oliveira, K. M., Figueiredo, R., Knoth, C. R. Definição de SLA para Processos de Testes de Software - Um estudo de caso em unidade de teste de integração de sistemas. 5th Conference for Quality in Information and Communications Tecnhology, Porto, Portugal, October 2004, pp. 127-134.

Santos, R., Caracteísticas e Medidas de Software para a Avaliação da Qualidade da Interação Humano-Computador em Sistemas Ubíquos, Masters Report, Federal Universy of Ceará, February 2014.

Satyanarayanan, M. Pervasive Computing: Vision and Challenges. IEEE Personal Communications, v. 8, 2001, pp. 10-17.

Sauro, J., Kindlund E. A method to standardize usability metrics into a single score. ACM-CHI conference on Human Factors in Computing Systems, Portland, USA, April 2005, pp. 401-409.

Sayeb, K, Rieu, D., Mandran N, Dupuy-Chessa. S. Qualité des langages de modélisation et des modèles : vers un catalogue des patrons collaboratifs. INFORSID, INFormatique des ORganisations et Systèmes d'Information et de Décision, Montpellier, France, May 2012, Montpellier, France, 2012, pp. 429-446.

Scapin, D.L. and Bastien, J.M.C. Ergonomic criteria for evaluating the ergonomic quality of interactive systems. Behaviour & Information Technology, v. 16, n.4, 1997, pp. 220-231.

Schalkwyk, J., Beeferman, D.m Beaufays, F., Byrne, B. *et al*. "Your Word is my Command": Google Search by Voice: A Case Study. Advances in Speech Recognition, Springer, 2010, pp. 61–90.

Scholtz, J.; Consolvo, S. Toward a Framework for Evaluating Ubiquitous Computing Applications. IEEE Pervasive Computing, v. 3, n. 2, April 2004, pp. 82-88.

Scott, L., Jeffery, R., Carvalho, L., D'Ambra, J., Rutherford, P. Practical software process improvement - the IMPACT project. Australian Software Engineering Conference, Camberra, Australie, August 2001, pp. 182-189.

Seacord, Robert C., Plakosh, Daniel, Lewis, Grace A. Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices, Addison-Wesley, 2003.

Sears, A. Layout Appropriateness: A Metric for Evaluating User Interface Widget Layout. IEEE Transaction on Software Engineering, n. 19, July 1993, pp. 707-719.

Sears, A., Jacko, J. Human-Computer Interaction: Development Process, CRC Press, 2009.

Seffah, A., Donyaee, M., Kline, R. Padda, H. Usability measurement and measures: A consolidated model, Software Quality Journal, v. 14, 2006, pp. 159–178.

SEI – Software Engineering Institute. CMMI® for Development, Version 1.3. 2010.

Senge, P., Kleiner, A., Roberts, C., Ross, R., Roth, G. Smith, B.. The dance of change: the challenges of sustaining momentum in learning organizations. Nicholas Brealey Publishing, London, 1999.

Simonin J., Carbonell N. Interfaces adaptatives : adaptation dynamique à l'utilisateur courant. In Saleh, I. and Regottaz, D., Interfaces numériques, Paris : Hermes Lavoisier (coll. Information, hypermédias et communication), 2006.

Sneed, H. Planning the reengineering of legacy systems. IEEE Software, v. 12, n. 1, 1995, pp. 24-34.

SOFTEX, Melhoria de Processo do Software Brasileiro – Guia Geral MPS de Software, (editors: Souza, G.S.; Rocha, A. R.; Machado, C.A.F.), August 2012.

SOFTEX, Melhoria de Processo do Software Brasileiro – Guia Geral MPS de Software, (editors: Oliveira, K.M., Rouiller, A.C, Rocha, A.R.), 2006.

Solingen, R. van, Berghout, E.. The Goal/Question/Metric Method: A practical guide for quality improvement of software development. McGraw-Hill, 1999.

Sottet, J-S., Calvary, G., Coutaz, J., Favre, J-M. A Model-Driven Engineering Approach for the Usability of Plastic User Interfaces, ACM SIGCHI symposium on Engineering interactive computing systems EICS, 2007, Lecture Notes in Cumputer Science, n. 4940, 2008, pp. 140-157.

Sousa, B., Pentikousis, K., Curado, M. UEF: Ubiquity Evaluation Framework. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), , n. 6649, 2011, pp. 92–103.

Sousa, K.D, Anquetil, N., Oliveira, K.M. Learning software maintenance organizations, in: Grigori Melnik, Harald Holz (Eds.), Advances in Learning Software Organizations – 6th International Workshop, LSO 2004, Lecture Notes in Cumputer Science, n. 3096, June 2004, pp. 67–77.

Sousa, M., Sândi, V.T., Oliveira, K. M., Figueiredo, R. Processo de Aquisição de Produtos e Serviços de Softwarepara uma Instituição Bancária. VII Simposio Internacional de Melhoria de Processo de Software, São Paulo, Brazil, November 2005, pp. 37-53.

Souza, K.D., Anquetil, N., Oliveira, K. M. Learning Software Maintenance Organizations. International Workshop on Learning Software Organizations, Lecture Notes in Cumputer Science, n. 3096, Banff, Canada, June 2004, pp. 67-77.

Souza, K.D., Oliveira, K. M., Anquetil, N. Aplicação do GQM para Avaliação de Processo de Manutenção de Software. III Jornadas Iberoamericana de Ingenieria del Software e Ingenieria del Conocimento, 2003, pp. 65-74.

Souza, S. C. B., Anquetil, N., Oliveira, K. M. Which documentation for software maintenance? Journal of the Brazilian Computer Society, v. 3, 2007, pp. 31-44.

Souza, S.C.B., Anquetil, N., Oliveira, K. M.A Study of the Documentation Essential to Software Maintenance. International Conference on Design of Communication (SIGDOC), Conventry, Engalnd, September 2005, pp. 68–75.

Stålhane, T. Dingsøyr, T. Hanssen, G.K. and Post, N.N.M. Postmortem - an assessement of two approaches. In Empirical Methods and Studies in Software Engineering, Lecture Notes in Cumputer Science, n. 2765, 2003, pp. 129-141.

Strauss, A., Corbin, J. Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. 2 ed. London, SAGE Publications, 1998.

Sulayman, M., Urquhart, C., Mendes, E., Seidel, S. Software process improvement success factors for small and medium Web companies: A qualitative study, Information and Software Technology, vol 54, n. 5, May 2012, pp. 1044-1061.

Sun, T., Denko, M. K.: Performance Evaluation of Trust Management in Pervasive Computing. International Conference on Advanced Information Networking and Applications, Okinawa, March 2008, pp. 386-394.

Surrephong, Pradorn Sureephong, P. , Chakpitak, N., Ouzrout, Y., Bouras, A. An Ontology-based Knowledge Management System for Industry Clusters, in Global Design to Gain a Competitive Edge, 2008, pp. 334-342.

Szyperski, C. Gruntz, D., Murer. S. Component software: beyond object-oriented programming. Addison-Wesley Professional, 2002.

Tan, D.S., Meyers, B. and Czerwinski, M. Wincuts: manipulating arbitrary window regions for more effective use of screen space. ACM CHI 2004 Conference on Human Factors in Computing Systems, v. 2, Vienna, Austria, April 2004, pp. 1525–1528.

Tang, L., Yu, Z., Zhou, X., Wang, H., Becker, C., Supporting rapid design and evaluation of pervasive applications: challenges and solutions. Personal Ubiquitous Computing, v. 15, n. 3, March 2011, pp. 253-269.

Thompson, S. G., Azvine, B. No Pervasive Computing Without Intelligent Systems. BT technology journal, v. 22, n. 3, 2004, pp. 39–49.

Tiwana, A. The knowledge management toolkit: practical techniques for building a knowledge management system, Prentice Hall PTR Upper Saddle River, USA, 2000.

Toch, E. Super-Ego: a framework for privacy-sensitive bounded context-awareness. ACM International Workshop on Context-Awareness for Self-Managing Systems, Beijing, China, September 2011, pp. 24-32.

Torres, A. H. S., Anquetil, N., Oliveira, K. M. Pro-active dissemination of knowledge with learning histories. 8th International Workshop on Learning Software Organization, Rio de Janeiro, Brazil, September 2006, pp. 19-27.

Torres, A.H., Oliveira, K.M., Anquetil, N., Lucena, G. Historias de aprendizagem em projetos de software: da teoria à prática. Editora Universa, Brasília, DF, Brasil, 2009.

Tran, C., Ezzedine, H., Kolski, C. EISEval, a Generic Reconfigurable Environment for Evaluating Agent-based Interactive Systems, International Journal of Human-Computer Studies, v. 71, n. 6, 2013, pp. 725-761.

Traue, T. G., Kobayashi, G. A discussion about Human-Computer interaction requirements for ubiquitous systems", Fourth International Conference on Ubi-Media Computing, 2011, pp. 134-137.

Travassos, G. H., Shull, F., Fredericks, M., Basili, V. Detecting defects in object-oriented designs: using reading techniques to increase software quality. 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, 34, Denver, USA, November 1999, pp. 47-56.

Tse, T. H., Yau, S. S., Chan, W. K., Lu, H., Chen, T. Y. Testing Context-Sensitive Middleware-Based Software Applications. International Computer Software and Applications Conference - COMPSAC'04, Washington, USA, September 2004, pp. 458-466.

Tsuchiya, S. Improving knowledge creation ability through organization learning, International Symposium on the Management of Industrial and Corporate Knowledge, 1993, pp. 87-95.

Tullis, T., Predicting the usability of alphanumeric displays. Doctoral Dissertation. Dept of Psychology, Houston, TX, Rice University. 1984.

Unterkalmsteiner, M., Gorschek, T., Islam, A., Cheng, C. K., Permadi, R. B., Feldt, R., Evaluation and measurement of software process improvement-A systematic literature review, IEEE Transactions on Software Engineering, v. 38, 2012, pp. 398-424.

Valentim, N.M.C., Oliveira K., Conte, T. Definindo uma abordagem para inspeção de usabilidade em modelos de projeto por meio de experimentação. IHC '12, Proceedings of the 11th Brazilian Symposium on Human Factors in Computing Systems, Cuiaba, Brazil, Novembre 2012, pp. 165-174.

Vanderdonckt, J. Guide ergonomique des interfaces homme-machine. Presses Universitaires de Namur, 1994.

Vanderdonkt, J. A MDA-Compliant Environment for Developing User Interfaces of Information Systems. CAISE, Lecture Notes in Cumputer Science, n. 3520, 2005, pp. 16-31.

Villela, K., Oliveira, K. M., Travassos, G.H., Rocha, A. R. The Definition and Automated Support of Software Processes, taking Domain Knowledge and Organizational Culture into Consideration. Workshop on Software Quality – ICSE, Orlando, USA, May 2002.

Viravan, C., Lessons learned from applying the spiral model in the software requirements analysis phase. Third IEEE International Symposium on Requirements Engineering, Annapolis, USA, January 1997.

Visaggio, G. Assessment of a Renewal Process Experimented in the Field, The Journal of Systems and Software v. 45, n. 1, 1999, pp. 3–17.

Visaggio, G. Value-based decision model for renewal processes in software maintenance. Annals of Software Engineering 9, 2000, pp. 215-233.

von Krogh, G., Ichijo, K., and Nonaka, I.. Enabling knowledge creation. Oxford University Press, New York, 2000.

Vora, V., and S. Bojewar. Design of a tool using statistical approach for personalization and usability improvement. International ACM Conference & Workshop on Emerging Trends in Technology, Mumbai, India, February 2011, pp. 228-229.

Vuljanić, D., Rovan, L., Baranović, M. Semantically enhanced web personalization approaches and techniques. 32nd International Conference on Information Technology Interfaces, Cavtat, Dubrovnik, June 2010, pp. 217-222.

Wagner, S., Toftegaard, T., Bertelsen, O. Requirements for an Evaluation Infrastructure for Reliable Pervasive Healthcare Research. 6th International Conference on Pervasive Computing Technologies for Healthcare, San Diego, USA, May 2012, pp. 260-267.

Waibel, A., Stiefelhagen, R. (eds) Computers in the Human Interaction Loop. Handbook of Ambient Intelligence and Smart Environments, Springer, 2010.

Wang, Q., Li, M. Measuring and improving software process in China. International Symposium on Empirical Software Engineering, November 2005.

Wang, Z., Elbaum, S.; Rosenblum, D. S. Automated Generation of Context-Aware Tests. International conference on Software Engineering, Minneapolis, USA, May, 2007, pp. 406-415.

Warnock, D. A Subjective Evaluation of Multimodal Notifications. 5th International Conference on Pervasive Computing Technologies for Healthcare, Dublin, Irealnd, May 2011, pp. 461-468.

Warren, I., Ransom, J. Renaissance: A Method to Support Software System Evolution. 26th Annual International Computer Software and Applications Conference, Oxorf. UK, August 2002, pp.415-420

Weber, K.C., Rocha, A. R., Rouiller, A. C., Alves, A., Crespo, A., Goncalves, A., Paret, B., Vargas, C., Salviano, C., Oliveira, K. M. Uma Estratégia para Melhoria de Processo de Software nas Empresas Brasileiras. 5th Conference for Quality in Information and Communications, Porto, Portugal, October 2004, pp. 73-78.

Weihong-Guo, A. et al. Using Immersive Video to Evaluate Future Traveller Information Systems. IET Intelligent Transport Systems, v. 2, n. 1, March 2008, pp. 38-46.

Weiser, M. The computer for 21st century, Scientific American, 1991.

Werneck, V., Oliveira, K. M., Rabelo Jr, A., Rocha, A.R. A Software Development Process for Expert Systems. 10th International Symposium on Methodologies for Intelligent Systems, Charlotte, USA, 1997, pp. 1-10.

Wholin, C., Runeson, P., Host, M. et al., Experimentation in Software Engineering, Kluwer Academic Publishers, 2000.

Won K. Personalization: Definition, Status, and Challenges ahead. Journal of Object Technology, 2002, pp. 29-40.

Wongthongtham, P., Chang, E., Dillon, T. Software Design Process Ontology Development, OTM 2006 Workshops, Lecture Notes in Cumputer Science, n. 4278, 2006, pp. 1806-1813.

Wu, C.-L. C.-L., Fu, L.-C. Design and Realization of a Framework for Human-System Interaction in Smart Homes. IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans, v. 42, n. 1, January, 2012, pp. 15–31.

Yourdon, E. Minipostmortems. Computerworld, March19, 2001.

Zhang, Y., Zhang, S., Tong, H. Adaptive Service Delivery for Mobile Users in Ubiquitous Computing Environments. International conference on Ubiquitous Intelligence and Computing, Lecture Notes in Cumputer Science, n. 4159, Wuhan, China, September, 2006, pp. 209-218.

Zlot, F., Oliveira, K. M., Rocha, A. Modeling Task Knowledge to Support Software Development. Software Engineering and Knowledge Engineering, Ischia, Italy, July 2002, pp. 35-42.

Appendix A. Defect Density Indicator

This appendix presents the specification of an indicator for software process performance analysis *(***Monteiro and Oliveira, 2011***)*. This specification is part of the research presented in part II, Chapter 3, § 3.2.2.

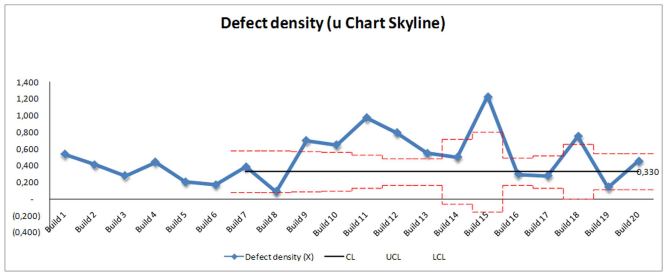
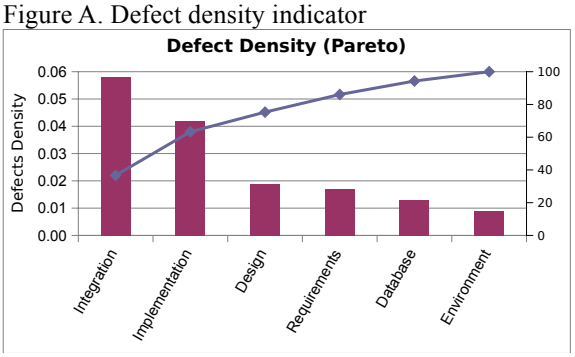
Information Need	What is the quality of the software product being developed?
Information Category	Quality

Measurement Objective	
Measurement Objective	To evaluate the performance of technical processes of software development based on the quality of the resulting product of these processes.

Entities and Attributes	
Relevant Entities	Project product (documentation or unit of software).
Attributes	Number of defects Size

Base Measure Specification	
Base Measures	Number of defects Total number of defects Number of defects by type Product size: Source lines of code Number of function points Number of requirements Document page count
Measurement Methods	Count the total number of defects found in tests of a package or unit of software. Count the number of defects found in tests of a package or unit of software, classified by type of defect or cause of the defect (Implementation, Database, Analysis, Design, Requirement, Integration, Environment, etc.). Assess the size of the product. Count the number of source lines of code. Count the number of function points. Count the number of functional requirements of the software. Count the number of pages of the document or other project product.
Type of Method	Objective Objective Objective
Scale	Integers, from zero to infinity Integers, from zero to infinity Integers, from zero to infinity Decimals, from zero to infinity Integers, from zero to infinity Integers, from zero to infinity

Type of Scale	Absolute Nominal Absolute Rational Absolute Absolute
Unit of Measurement	Defects Defects Source lines of code Function points Requirements Number of pages

Derived Measure Specification	
Derived Measure	Total defect density Defect density by type
Measurement Function	$\frac{\text{Total number of defects}}{\text{Product size}}$ $\frac{\text{Number of defects by type}}{\text{Product size}}$
Indicator Specification	
Indicator Description and Sample[18]	**Defect density**: This indicator shows the ratio between the number of defects found during the tests and the size of the product verified (Figure A). The indicator also presents the classification of defects by type to identify the processes that caused the most defects in the project product Figure B).  Figure A. Defect density indicator  Figure B. Classification of defects
Analysis Model	The measure of "total defect density" is presented in the form of a control chart in order to evaluate the stability of the software development process in relation to the quality of the product created. The u Chart Control presents the evolution of the defect density in different products produced and verified by the project team. The analysis of this chart should look for signs of deviations, or exceptional variations of the process, that require detailed

	research in order to identify and correct the causes of variation. The centerline (CL) represents the defect density average, and the upper and lower control limits (UCL and LCL) are not fixed. That is, for each observation of the measure, its limits are recalculated from the average of all defect densities observed and the relationship between the densities of defects versus the size of each observation of the measure. This relationship involving individual averages is what adds flexibility to the control limits. The measure of "defect density by type" is presented in a Pareto chart. This diagram shows the main causes of defects found, sorted according to their defect densities. The goal of the Pareto diagram is to support the prioritization of actions to resolve the causes that have the highest defect densities.
Decision Criteria	The decision criteria of the "total defect density" measure follows the rules of process instability for each of the graphs that represent the indicator. For every occurrence of one of the rules listed below, the process should be examined in order to identify and eliminate the causes of variation. For the u control chart, one or more points outside the upper control limit (CL + 3 sigma) and lower control limit (CL – 3 sigma) For the Pareto chart, the defect density by type has no decision criteria. The main causes of variation should be primarily dealt with.
Indicator Interpretation[19]	The control chart in the example above shows that Builds 9, 10, 11, 12, 13, 15, and 18 show signs of diversion, as they demonstrate values outside the upper and lower control limits. The Pareto chart of the above example shows that the most common causes of defects in the product are: Integration, followed by Implementation, Design, Requirements, Database and Environment. Following the Pareto rule that 20% of causes are responsible for 80% of the problems, actions should be implemented primarily in the processes of Integration, Implementation, Design and Requirements, because they will be most effective in improving product quality.

Data Collection Procedure (for Each Base Measure)	
Frequency of Data Collection	Each package or unit of software delivered to the customer (Build). Collection of the total number of defects of the package or unit of software. Each package or unit of software delivered to the customer (Build). Collection of number of defects by type of the package or unit of software. Each package or unit of software delivered to the customer (Build). Collection of size of the package or unit of software.
Responsible Individual	The Test analyst collects the number of defects and defects by type; the Metrics analyst obtains the size of the package or unit of software tested and consolidates the measurement information.
Phase or Activity in Which Collected	In construction and transition phases of the project life cycle, measurement continues during the operation of the product: data should be collected from the time the first tests are carried out. Optionally, the measure can continue being collected during the operation or warranty of the product in a production environment.
Tools Used in Data Collection	Defect recording tool Defect recording tool Counting lines of code tool Function point recording tool List of requirements
Verification and Validation	Audits on the records of defects and product size.
Repository for Collected Data	Measurement repository of the organization.

Data Analysis Procedure (for each Indicator)	
Frequency of Data Reporting	For each product or software package delivered to the customer.
Responsible Individual	Measurement Analyst and Project Manager (project level) Organizational Measurement Analyst (organizational level)
Phase or Activity in Which Analyzed	In construction and transition phases of the project life cycle, and to continue during the operation of the product: data should be collected from the time the first tests are carried out. Optionally, the measure can continue being collected during the operation or warranty of the product in a production environment.
Source of Data for Analysis	Measurement repository of the organization
Tools Used in Analysis	Measurement repository of the organization
Review, Report, or User	Project team, Project Manager, and Portfolio Manager (Project level) Portfolio Manager and Senior Manager (organizational level)

Additional Information	
Additional Analysis Guidance	Analysis of the evolution of the defect density by type may be carried out through the evaluation of the different Pareto diagrams produced each month.
Implementation Considerations	**Phase or activity of the life cycle**: The organization must adapt the timing of data collection and indicator analysis according to the phases and activities defined in the life cycle of its projects (e.g. waterfall or iterative). **Minimum quantity of data:** To start the analysis of variability are required at least 20 points in the graph. Below this number, the chart can be prepared, but is not suitable for analysis of variability.