COPPE/UFRJ

APOIO À ESPECIFICAÇÃO E VERIFICAÇÃO DE REQUISITOS FUNCIONAIS DE UBIQUIDADE EM PROJETOS DE SOFTWARE

Rodrigo Oliveira Spínola

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientador: Guilherme Horta Travassos

Rio de Janeiro

Novembro de 2010

APOIO À ESPECIFICAÇÃO E VERIFICAÇÃO DE REQUISITOS FUNCIONAIS DE UBIQUIDADE EM PROJETOS DE SOFTWARE

Rodrigo Oliveira Spínola

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

_____

Prof. Guilherme Horta Travassos, D.Sc.

_____

Profa. Cláudia Maria Lima Werner, D.Sc.

_____

Prof. Jano Moreira de Souza, Ph.D.

_____

Profa. Karin Koogan Breitman, D.Sc.

_____

Prof. Orlando Gomes Loques Filho, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
NOVEMBRO DE 2010

Spínola, Rodrigo Oliveira

Apoio à Especificação e Verificação de Requisitos Funcionais de Ubiquidade em Projetos de Software / Rodrigo Oliveira Spínola – Rio de Janeiro: UFRJ/COPPE, 2010.

XVIII, 289 p.: il.; 29,7 cm.

Orientador: Guilherme Horta Travassos.

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2010.

Referências Bibliográficas: p. 189-204.

1. Computação Ubíqua. 2. Engenharia de Requisitos. 3. Especificação e Verificação de Requisitos. 4. Engenharia de Software Experimental. I. Travassos, Guilherme Horta II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Aos meus Pais, Nil e Edvaldo, meus exemplos de vida e
de onde tiro minha inspiração.
À minha Esposa, Joice.
Aos meus Irmãos, Gustavo e Eduardo.

Agradecimentos

Ao meu Pai, Edvaldo (em memória), pelo carinho, força e por todo exemplo de vida que sempre me deu.

À minha Mãe, Nil, pelo amor, compreensão e apoio irrestrito nas horas mais difíceis.

À minha Esposa, Joice, pelo amor e carinho. Sua preocupação e força dada a todo o momento foram muito importantes.

Aos meus Irmãos, Gustavo e Eduardo, pela força e por estarem sempre presentes nesta caminhada.

A Sr. José e Keu por sempre me acolherem bem e pelo apoio. Aos meus Cunhados, Jeane e Joan, pelo carinho e torcida constantes. À minha Sobrinha Giovanna, por ser parecida com a tia e pelo carinho.

Aos meus Avôs, Edite, Nair (em memória), Nelson (em memória), e Narcizo (em memória). Aos meus Tios Nelson, Bárbara, Jorge, Nilzon, Nancy, Lafayet, Spínola, Dete, Nilzete, Joel e Nelci. Aos meus Primos Dudu, Jaque, Juli, Joelma, Roland, Junior, Cissa, Ticiane, Junior, Adriano e Victor. Obrigado a todos pela torcida e apoio.

Ao meu Orientador, Guilherme Travassos, pela grande dedicação, conselhos e motivação durante todo este período, o que contribuiu não apenas para minha formação acadêmica, mas também minha formação como pessoa. Agradeço também pela orientação, incentivo e por me conduzir durante este trabalho e, por acreditar em mim e no meu trabalho.

Aos professores Cláudia Werner, Jano Souza, Karin Breitman e Orlando Loques por participarem de minha banca de defesa de doutorado.

Aos meus Amigos Anderson, Arilo, Caetano, Ely, Glênio, Gustavo, João Carlos, Leandro, Marcelo, Márcio, Marcos, Nilda, Rafael, Scheila e Thomaz, sempre presentes e na torcida.

Aos Companheiros da COPPE Beto, Breno, Karen, Leonardo, Lucas, Marco Antônio, Paulo Sérgio, Priscila, Taísa, Tayana, Verônica, Victor Vidigal, Vitor Farias, Vitor Lopes e Wallace pela amizade, sugestões e ajuda nos momentos que precisei. Em especial ao Jobson e Felipe pela ajuda e contribuições nesta pesquisa.

Aos Amigos da UNIFACS Atta, Isaac e Manoel aos quais sempre serei grato.

À CAPES pelo apoio financeiro. À COPPE por prover a infra-estrutura.

A Deus, por estar comigo em todos os momentos e por ter me dado saúde e o dom da sabedoria. Nada seria possível sem Ele.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

APOIO À ESPECIFICAÇÃO E VERIFICAÇÃO DE REQUISITOS FUNCIONAIS DE UBIQUIDADE EM PROJETOS DE SOFTWARE

Rodrigo Oliveira Spínola

Novembro/2010

Orientador: Guilherme Horta Travassos

Programa: Engenharia de Sistemas e Computação

Este trabalho apresenta uma abordagem para auxiliar o engenheiro de software durante as atividades de definição e verificação dos requisitos funcionais de ubiquidade do projeto considerando dois aspectos: (1) conjunto de características da computação ubíqua considerado pertinente e relevante na caracterização de projetos de software, e; (2) definição de guias de apoio às atividades de definição e verificação dos requisitos de ubiquidade em projetos de software de forma a atender às necessidades específicas da engenharia de requisitos no contexto da computação ubíqua.

A definição do conjunto de características assim como a avaliação sobre sua pertinência e relevância foi efetuada através de estudos secundários (revisões sistemáticas) e primários (*survey*). A partir deste conhecimento, foram elaborados os guias de apoio às atividades de definição e verificação de requisitos de ubiquidade.

Resultados de uma avaliação experimental indicam que a abordagem apresentada contribui para melhoria na efetividade e eficiência do processo de definição e verificação de requisitos funcionais de ubiquidade em projetos de software quando comparada ao uso de uma abordagem *ad-hoc* na execução destas atividades.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

SUPPORTING THE DEFINITION AND VERIFICATION OF UBIQUITY FUNCTIONAL REQUIREMENTS IN SOFTWARE PROJECTS

Rodrigo Oliveira Spínola

November/2010

Advisor: Guilherme Horta Travassos

Department: Computer Science and Systems Engineering

This Thesis presents an approach to support the functional requirements definition and verification by software engineers in the context of ubiquitous software projects considering two aspects: (1) the set of ubiquitous computing characteristics that are considered relevant and pertinent to characterize software projects, and; (2) the definition of guidelines to support the activities of definition and verification of ubiquity requirements in software projects in order to promote specific requirements engineering needs in the context of ubiquitous computing.

The identification of the ubiquitous computing characteristics as well as the evaluation of their relevance and pertinence had been achieved through secondary (systematic reviews) and primary (survey) studies. From such organized knowledge, guidelines to support the definition and verification activities of ubiquity requirements were defined.

Results of an empirical evaluation indicate that the proposed approach can allow the increase on effectiveness and efficiency in defining and verifying ubiquity functional requirements when compared to an *ad-hoc* approach.

ÍNDICE

ÍNDICE DE FIGURAS ..... **XIV**

ÍNDICE DE TABELAS ..... **XVII**

CAPÍTULO 1 – INTRODUÇÃO **1**

1.1 – INTRODUÇÃO 1

1.2 – CONTEXTO E MOTIVAÇÃO 2

1.3 – QUESTÕES DE PESQUISA 4

1.4 – OBJETIVOS 7

1.5 – METODOLOGIA E HISTÓRICO DA PESQUISA 8

1.5.1 Fase de Concepção da Tecnologia 10

1.5.2 Fase de Avaliação da Tecnologia 12

1.6 – ORGANIZAÇÃO DA TESE 16

CAPÍTULO 2 – COMPUTAÇÃO UBÍQUA: CARACTERÍSTICAS E FATORES **19**

2.1 - INTRODUÇÃO 19

2.2 – CARACTERÍSTICAS DA COMPUTAÇÃO UBÍQUA 20

2.2.1 Planejamento e Execução da Revisão Sistemática 21

2.2.2 Definição de Computação Ubíqua 23

2.2.3 Características da Computação Ubíqua 24

2.3 – FATORES FUNCIONAIS E RESTRITIVOS ASSOCIADOS ÀS CARACTERÍSTICAS DA COMPUTAÇÃO UBÍQUA 27

2.3.1 Planejamento e Execução da Revisão Sistemática 27

2.3.2 Fatores Funcionais e Restritivos 30

2.4 – CONSIDERAÇÕES FINAIS 41

CAPÍTULO 3 – CARACTERIZAÇÃO DE PROJETOS DE SOFTWARE UBÍQUO **43**

3.1 - INTRODUÇÃO 43

3.2 - ABORDAGEM PARA CARACTERIZAÇÃO DE PROJETOS DE SOFTWARE UBÍQUO BASEADA EM *CHECKLIST* 43

3.3 - CARACTERIZAÇÃO DE PROJETOS DE SOFTWARE UBÍQUO 46

3.3.1 Projeto de software discutido em (KINDBERG *et al*. 2000) 46

3.3.2 Projeto de software apresentado em (JOEL *et al.,* 2004) 47

3.3.3 Projeto de software apresentado em (ALI *et al.,* 2004) 48

3.3.4 Projeto de software descrito em (TAHTI *et al.,* 2004) 48

3.3.5 Projeto de software proposto em (BOSSEN and JORGENSEN, 2004) 49

3.3.6 Projeto de software discutido em (LEE *and* CHUNG, 2004) 49

3.3.7 Projeto de software apresentado em (HATALA *et al.,* 2005) 50
3.3.8 Projeto de software descrito em (ZHOU *et al.* 2005) 51
3.3.9 Projeto de software apresentado em (KIENTZ *et al.,* 2005) 51
3.3.10 Projeto de software apresentado em (VAINIO *et al.,* 2006) 52
3.3.11 Projeto de software apresentado em (NAWYN *et al.,* 2006) 52
3.3.12 Projeto de software proposto por (O'NEILLl *et al.,* 2006) 53
3.3.13 Projeto de software proposto por (SHIN *et al.,* 2007) 53
3.3.14 Projeto de software apresentado em (KIENTZ *et al.,* 2007) 54
3.3.15 Projeto de software apresentado em (SU *et al.,* 2009) 55
3.3.16 Projeto de software apresentado em (LEE e PARK, 2009) 55
3.3.17 Projeto de software apresentado em (YANG *et al*., 2009) 56
3.4 – CONSIDERAÇÕES FINAIS 58

CAPÍTULO 4 – CARACTERÍSTICAS DA COMPUTAÇÃO UBÍQUA: PERTINÊNCIA E RELEVÂNCIA 59

4.1 - INTRODUÇÃO 59
4.2 - AVALIAÇÃO INICIAL DO CORPO DE CONHECIMENTO 60
4.2.1 Instrumentação e População 62
4.2.2 Planejamento da Análise de Dados 63
4.2.3 Resultados 64
4.3 - PERTINÊNCIA E RELEVÂNCIA DAS CARACTERÍSTICAS DA COMPUTAÇÃO UBÍQUA 67
4.3.1 Instrumentação e População 69
4.3.2 Planejamento da Análise de Dados 72
4.3.2.1 Peso dos Participantes 72
4.3.2.2 Pertinência das Características 73
4.3.2.3 Relevância das Características 74
4.3.3 Resultados 74
4.3.3.1 Análise da Pertinência das Características da Computação Ubíqua 75
4.3.3.2 Análise da Relevância das Características da Computação Ubíqua 76
4.4 - ATUALIZAÇÃO DO CORPO DE CONHECIMENTO 77
4.4.1 Protocolo de Atualização do Corpo de Conhecimento 78
4.4.2 Atualização do Corpo de Conhecimento 80
4.5 - CONSIDERAÇÕES FINAIS 82

CAPÍTULO 5 – ENGENHARIA DE REQUISITOS E COMPUTAÇÃO UBÍQUA 83

5.1 – INTRODUÇÃO 83
5.2 – TRABALHOS RELACIONADOS 84
5.2.1 Planejamento e Execução da Revisão da Literatura 84
5.2.2 Trabalhos Relacionados Identificados 88
5.2.2.1 Casos de Uso Executáveis 89
5.2.2.2 Modelagem de Requisitos com a Linguagem KAOS 89
5.2.2.3 Metodologia para Definir Requisitos de Sistemas Adaptáveis e Sensíveis ao Contexto 90

5.2.2.4 Definição de Requisitos do Usuário com base em uma Ontologia 92
5.2.2.5 Mecanismo para Definição de Requisitos de Serviços 93
5.2.2.6 Exemplos de Requisitos de Ubiquidade do Projeto ENQUETE-BAISE 94
5.2.2.7 Exemplos de Requisitos de Ubiquidade de um Sistema Multimodal Multimedia 95
5.2.2.8 Processo Sistemático para Análise de Requisitos de Segurança em Sistemas Embarcados 96
5.2.2.9 Processo para Definição de Requisitos com base nas Intenções dos Usuários 96
5.2.3 Análise das Abordagens Identificadas na Literatura 97
5.3 – CONSIDERAÇÕES FINAIS 98

CAPÍTULO 6 – *U*-RDV: ABORDAGEM DE APOIO À DEFINIÇÃO E VERIFICAÇÃO DE REQUISITOS FUNCIONAIS DE UBIQUIDADE EM PROJETOS DE SOFTWARE 99

6.1 - INTRODUÇÃO 99
6.2 – VISÃO GERAL DA ABORDAGEM 100
6.3 – CONFIGURAÇÃO DA ABORDAGEM 106
6.3.1. Elaborar Modelos de Ubiquidade 107
6.4 – CARACTERIZAÇÃO DO PROJETO DE SOFTWARE UBÍQUO 110
6.4.1. Identificar Características e Fatores de Ubiquidade Relevantes ao Projeto 111
6.5 – DEFINIÇÃO DE REQUISITOS DE UBIQUIDADE - *UBICHECK* 113
6.5.1. Elaborar Guia Geral de Definição de Requisitos de Ubiquidade 115
6.5.1.1 Elaborar Árvore de Elementos 115
6.5.1.2 Elaborar Perguntas 120
6.5.2. Especializar Guia Geral de Definição de Requisitos de Ubiquidade 122
6.5.3. Definir Requisitos de Ubiquidade do Projeto 126
6.6 – VERIFICAÇÃO DE REQUISITOS DE UBIQUIDADE - *UBIVERI* 127
6.6.1. Elaborar Guia Geral de Verificação de Requisitos de Ubiquidade 127
6.6.2. Especializar Guia Geral de Verificação de Requisitos de Ubiquidade 131
6.6.3. Verificar Requisitos de Ubiquidade 132
6.7 – CONSIDERAÇÕES FINAIS 132

CAPÍTULO 7 - AVALIAÇÃO DE *UBICHECK* 135

7.1 - INTRODUÇÃO 135
7.2 - AVALIAÇÃO DO USO DE *UBICHECK* 135
7.2.1 Definição do Estudo Experimental 135
7.2.2 Planejamento do Estudo Experimental 136
7.2.3 Operação 143
7.2.4 Análise e Interpretação dos Dados 144
7.3 – *UBICHECK* - VERSÃO 2.0 146
7.3.1. Atividades Alteradas e Inseridas em *UbiCheck* 2.0 148
7.3.1.1 Elaborar Guia Geral de Definição de Requisitos de Ubiquidade 148
7.3.1.2 Elaborar Direcionamento da Resposta 149
7.3.1.3 Elaborar Glossário 150

7.3.1.4 Definir Requisitos de Ubiquidade do Projeto 151
7.4 – CONSIDERAÇÕES FINAIS 151

CAPÍTULO 8 - AVALIAÇÃO DE *UBIVERI* 152

8.1 - INTRODUÇÃO 152
8.2 – AVALIAÇÃO DO USO DE *UBIVERI* 152
8.2.1 Definição do Estudo Experimental 152
8.2.2 Planejamento do Estudo Experimental 153
8.2.3 Operação 163
8.2.4 Análise e Interpretação dos Dados 165
8.2.4.1 Análise da Variável Número de Defeitos 165
8.2.4.2 Análise da Variável Número de Discrepâncias 169
8.2.4.3 Análise da Variável Número de Falso Positivos 172
8.2.4.4 Análise da Variável Duração da Revisão 176
8.2.4.5 Análise da Variável Satisfação do Usuário 179
8.2.5 Resumo dos Resultados 181
8.3 – CONSIDERAÇÕES FINAIS 182

CAPÍTULO 9 – CONSIDERAÇÕES FINAIS 183

9.1 – CONSIDERAÇÕES FINAIS 183
9.2 – RESULTADOS OBTIDOS 184
9.3 – CONTRIBUIÇÕES DA PESQUISA 184
9.3.1 Corpo de Conhecimento em Computação Ubíqua 185
9.3.2 Abordagem de Apoio à Definição de Requisitos de Ubiquidade: *UbiCheck* 185
9.3.3 Abordagem de Apoio à Verificação de Requisitos de Ubiquidade: *UbiVeri* 185
9.3.4 Guias de Apoio às atividades de Definição e Verificação de Requisitos Funcionais de Ubiquidade .. 186
9.3.5 Metodologia Científica que apóia a Concepção de Tecnologias de Software a partir da condução de Estudos Secundários e Primários 186
9.4 – LIMITAÇÕES 186
9.5 – TRABALHOS FUTUROS 187

REFERÊNCIAS BIBLIOGRÁFICAS 189

APÊNDICE A – ARTIGOS UTILIZADOS NAS REVISÕES SISTEMÁTICAS 205

A.1 – REVISÃO SISTEMÁTICA: CARACTERÍSTICAS DA COMPUTAÇÃO UBÍQUA 205
A.2 – REVISÃO SISTEMÁTICA: FATORES FUNCIONAIS E RESTRITIVOS 207

APÊNDICE B – *CHECKLIST* DE CARACTERIZAÇÃO 211

B.1 – *CHECKLIST* DE CARACTERIZAÇÃO 211

APÊNDICE C – PROTOCOLO DE ATUALIZAÇÃO DO CORPO DE CONHECIMENTO 216

C.1 – PROTOCOLO DE ATUALIZAÇÃO 216

APÊNDICE D – MODELOS CONCEITUAIS 221

D.1 – CAPTURA DE EXPERIÊNCIA 221
D.2 – COMPORTAMENTO ADAPTÁVEL 222
D.3 – HETEROGENEIDADE DE DISPOSITIVOS 222
D.4 – INTEROPERABILIDADE ESPONTÂNEA 223
D.5 – ONIPRESENÇA DE SERVIÇOS 224
D.6 – SENSIBILIDADE AO CONTEXTO 225

APÊNDICE E – QUESTIONÁRIO DE CARACTERIZAÇÃO 226

E.1 – QUESTIONÁRIO DE CARACTERIZAÇÃO – ETAPA 1 226
E.2 – QUESTIONÁRIO DE CARACTERIZAÇÃO – ETAPA 2 228

APÊNDICE F – GUIA GERAL DE DEFINIÇÃO DE REQUISITOS DE UBIQUIDADE 233

F.1 – CARACTERÍSTICA CAPTURA DE EXPERIÊNCIA 233
F.2 – CARACTERÍSTICA COMPORTAMENTO ADAPTÁVEL 234
F.3 – CARACTERÍSTICA HETEROGENEIDADE DE DISPOSITIVOS 236
F.4 – CARACTERÍSTICA INTEROPERABILIDADE ESPONTÂNEA 237
F.5 – CARACTERÍSTICA ONIPRESENÇA DE SERVIÇOS 238
F.6 – CARACTERÍSTICA SENSIBILIDADE AO CONTEXTO 239

APÊNDICE G – GUIA GERAL DE VERIFICAÇÃO DE REQUISITOS DE UBIQUIDADE 242

G.1 – GUIA GERAL DE VERIFICAÇÃO DE REQUISITOS DE UBIQUIDADE 242
G.2 – GLOSSÁRIO DE TERMOS EM COMPUTAÇÃO UBÍQUA 247

APÊNDICE H – INSTRUMENTOS DO ESTUDO EXPERIMENTAL PARA AVALIAÇÃO DE UBICHECK 252

H.1 – CONSENTIMENTO DE PARTICIPAÇÃO 252
H.2 – CARACTERIZAÇÃO DOS PARTICIPANTES 253
H.3 – ORIENTAÇÕES PARA A OPERAÇÃO DO TRABALHO 255
H.3.1 – Versão Entregue aos Participantes da Abordagem *Ad hoc* 255
H.3.2 – Versão Entregue aos Participantes que utilizaram *UbiCheck* 256
H.4 - QUESTIONÁRIO DE ACOMPANHAMENTO 257

APÊNDICE I – CENÁRIOS DE UBÍQUIDADE 258

I.1 – CENÁRIO BICICLETÁRIO 258
I.2 – CENÁRIO ITEM RASTREÁVEL 259

APÊNDICE J – INSTRUMENTOS DO ESTUDO EXPERIMENTAL PARA AVALIAÇÃO DE UBIVERI 262

J.1 – CONSENTIMENTO DE PARTICIPAÇÃO 262

J.2 – CARACTERIZAÇÃO DOS PARTICIPANTES 263
J.3 – ORIENTAÇÕES PARA A OPERAÇÃO DO TRABALHO 264
J.3.1 – Versão Entregue aos Participantes da Abordagem Ad hoc 264
J.3.2 – Versão Entregue aos Participantes da Abordagem Proposta 265
J.4 - QUESTIONÁRIO DE ACOMPANHAMENTO 267
APÊNDICE K – GUIA GERAL DE DEFINIÇÃO DE REQUISITOS DE UBIQUIDADE COM DIRECIONAMENTO 269
K.1 – GUIA GERAL COM DIRECIONAMENTO 269

ÍNDICE DE FIGURAS

Figura 1.1. Paradigmas da computação..... 2
Figura 1.2. Metodologia de Pesquisa adotada 10
Figura 1.3. Fase de Concepção da Tecnologia 10
Figura 1.4. Resumo dos Estudos conduzidos ao longo desta pesquisa 14
Figura 1.5. Distribuição das atividades realizadas em uma linha de tempo 15
Figura 3.1. Fragmento do checklist 49
Figura 3.2. Nível de aderência do projeto apresentado em (KINDBERG et al. 2000)..... 50
Figura 3.3. Nível de aderência do projeto apresentado em (JOEL et al., 2004)..... 50
Figura 3.4. Nível de aderência do projeto apresentado em (ALI et al., 2004)..... 51
Figura 3.5. Nível de aderência do projeto apresentado em (TAHTI et al., 2004).... 52
Figura 3.6. Nível de aderência do projeto apresentado em (BOSSEN and JORGENSEN, 2004)..... 52
Figura 3.7. Nível de aderência do projeto descrito em (Lee and Chung, 2004)..... 53
Figura 3.8. Nível de aderência da aplicação apresentada por (HATALA et al., 2005)..... 53
Figura 3.9. Nível de aderência do projeto apresentado em (ZHOU et al. 2005)..... 54
Figura 3.10. Nível de aderência do projeto apresentado em (KIENTZ et al., 2005)..... 55
Figura 3.11. Nível de aderência do projeto descrito em (VAINIO et al., 2006)..... 55
Figura 3.12. Nível de aderência do projeto apresentado em (NAWYN et al., 2006)..... 56
Figura 3.13. Nível de aderência do projeto descrito em (O'NEILLl et al., 2006)..... 56
Figura 3.14. Nível de aderência do projeto descrito em (SHIN et al., 2007)..... 57
Figura 3.15. Nível de aderência do projeto apresentado em (KIENTZ et al., 2007)..... 57
Figura 3.16. Nível de aderência do projeto apresentado em (SU et al., 2009)..... 58
Figura 3.17. Nível de aderência do projeto descrito em (LEE e PARK, 2009)..... 59
Figura 3.18. Nível de aderência do projeto descrito em (YANG et al., 2009)..... 59
Figura 3.19 Nível de aderência dos projetos investigados em relação às características de ubiquidade..... 60
Figura 4.1. Evolução do conjunto de características de ubiquidade..... 69
Figura 4.2. Caracterização do participante..... 72
Figura 4.3. Identificação da pertinência das características..... 73

Figura 4.4. Identificação da relevância das características..... 74
Figura 4.5. Pertinência das características..... 78
Figura 4.6. Relevância das características..... 79
Figura 5.1. Abordagem de apoio à identificação de requisitos (CHIU et al., 2007). 95
Figura 5.2. Abordagem de apoio à identificação de requisitos. Adaptada de (BO et al., 2007)..... 96
Figura 6.1. Visão geral da abordagem de apoio à definição e verificação de requisitos de ubiquidade em projetos de software ubíquo..... 105
Figura 6.2. Processo para definição e verificação de requisitos de ubiquidade..... 106
Figura 6.3. Meta-modelo das características de software ubíquo..... 111
Figura 6.4. Exemplo de modelo de ubiquidade de parte da característica Sensibilidade ao Contexto..... 113
Figura 6.5. Fragmento do formulário para apoiar a identificação das características de ubiquidade que estarão contempladas no projeto de software.. 115
Figura 6.6. Fragmento do checklist para apoiar a identificação dos fatores funcionais que deverão estar contempladas no projeto de software..... 117
Figura 6.7. Visão geral de UbiCheck..... 118
Figura 6.8. Sub-atividades para Elaborar o Guia para Definição de Requisitos de Ubiquidade..... 119
Figura 6.9. Trecho do Modelo de Sensibilidade ao Contexto..... 119
Figura 6.10. Exemplo de representação de uma árvore de elementos de um trecho do modelo da característica Sensibilidade ao Contexto..... 120
Figura 6.11. Visão geral de UbiVeri..... 128
Figura 6.12. Fragmento da Guia Geral de Verificação de Requisitos de Ubiquidade..... 131
Figura 6.13. Fragmento do Formulário de Relato de Discrepâncias..... 132
Figura 7.1. Visão geral de UbiCheck 2.0..... 147
Figura 8.1. Análise de outlier para a variável Defeito..... 165
Figura 8.2. Análise de distribuição normal utilizando levene..... 165
Figura 8.3. Resultado do uso de Wilcoxon..... 166
Figura 8.4. Análise de outlier para a variável Defeito..... 167
Figura 8.5. Resultado do uso de ANOVA..... 167
Figura 8.6. Análise de outlier para a variável Discrepância – Cenário Bicicletário. 168
Figura 8.7. Análise de distribuição normal utilizando levene..... 169
Figura 8.8. Resultado do uso de ANOVA..... 169
Figura 8.9. Análise de outlier para a variável Discrepância – Cenário Bicicletário

antes e depois da análise de outlier..... 170
Figura 8.10. Análise de distribuição normal utilizando levene..... 170
Figura 8.11. Resultado do uso de ANOVA..... 171
Figura 8.12. Análise de outlier para a variável Falso Positivo – Cenário Bicicletário..... 172
Figura 8.13. Análise de distribuição normal utilizando levene..... 172
Figura 8.14. Resultado do uso de ANOVA..... 173
Figura 8.15. Análise de outlier para a variável Falso Positivo – Cenário Bicicletário antes e depois da análise de outlier..... 173
Figura 8.16. Análise de distribuição normal utilizando levene..... 174
Figura 8.17. Resultado do uso de ANOVA..... 174
Figura 8.18. Análise de outlier para a variável Duração da Revisão – Cenário Bicicletário..... 175
Figura 8.19. Análise de distribuição normal utilizando levene..... 176
Figura 8.20. Resultado do uso de ANOVA..... 176
Figura 8.21. Análise de outlier para a variável Duração da Revisão – Cenário Bicicletário..... 177
Figura 8.22. Análise de distribuição normal utilizando levene..... 177
Figura 8.23. Resultado do uso de ANOVA..... 178

ÍNDICE DE TABELAS

Tabela 2.1. Lista de fatores funcionais e restritivos identificados para a característica Captura de Experiência..... 31
Tabela 2.2. Lista de fatores funcionais e restritivos identificados para a característica Comportamento Adaptável..... 32
Tabela 2.3. Lista de fatores funcionais e restritivos identificados para a característica Composição de Funcionalidade..... 34
Tabela 2.4. Lista de fatores funcionais e restritivos identificados para a característica Descoberta de Serviços..... 36
Tabela 2.5. Lista de fatores funcionais e restritivos identificados para a característica Heterogeneidade de Dispositivos..... 37
Tabela 2.6. Lista de fatores funcionais e restritivos identificados para a característica Interoperabilidade Espontânea..... 38
Tabela 2.7. Lista de fatores funcionais e restritivos identificados para a característica Invisibilidade..... 39
Tabela 2.8. Lista de fatores funcionais e restritivos identificados para a característica Onipresença de Serviços..... 40
Tabela 2.9. Lista de fatores funcionais e restritivos identificados para a característica Sensibilidade ao Contexto..... 44
Tabela 4.1. Caracterização dos participantes do survey..... 67
Tabela 4.2. Estado atual do Corpo de Conhecimento organizado em computação ubíqua..... 69
Tabela 4.3. Pertinência e relevância das características de ubiquidade..... 80
Tabela 4.4. Estado atual do Corpo de Conhecimento organizado em computação ubíqua..... 84
Tabela 5.1. Abordagens identificadas..... 91
Tabela 6.1. Descrição do processo..... 108
Tabela 6.2. Exemplo de representação EM TABELA de uma árvore de elementos de um trecho do modelo da característica Sensibilidade ao Contexto..... 120
Tabela 6.3. Assunto, Escopo e Tipo da Pergunta..... 121
Tabela 6.4. Tipos de Perguntas..... 122
Tabela 6.5. Exemplos de Perguntas..... 123
Tabela 6.6. Trecho do Guia Geral de Definição de Requisitos de Ubiquidade da característica Sensibilidade ao Contexto..... 124

Tabela 6.7. Questionamentos associados às perguntas específicas e genéricas.. 129
Tabela 7.1. Alocação das abordagens de verificação por grupos e dos cenários de ubiquidade..... 140
Tabela 7.2. Procedimento Experimental..... 141
Tabela 7.3. Cronograma do Experimento..... 142
Tabela 7.4. Tempo gasto por cada equipe para definir os requisitos de ubiquidade..... 145
Tabela 8.1. Alocação das abordagens de verificação por grupos e dos cenários de ubiquidade..... 159
Tabela 8.2. Procedimento Experimental..... 160
Tabela 8.3. Cronograma do Experimento..... 160
Tabela 8.4. Operação do estudo..... 163
Tabela 8.5. Resumo dos Resultados..... 180

Capítulo 1 – Introdução

Neste capítulo são apresentados o contexto do trabalho, o que motivou esta pesquisa e a questão de investigação. São também apresentados os objetivos, a metodologia de pesquisa utilizada e como este texto está estruturado.

1.1 – Introdução

A engenharia de software apóia o uso de abordagens sistemáticas, econômicas e quantificáveis para o desenvolvimento, operação e manutenção de software de qualidade. Para isso, define um conjunto de metodologias de apoio à construção de software.

Parte destas metodologias é elaborada a partir do conhecimento prático e dos problemas que engenheiros de software vivenciam em suas atividades diárias. Além disso, também é possível desenvolver novas abordagens a partir do surgimento de demandas no desenvolvimento de projetos de software. Em ambos os casos, o uso do paradigma experimental para avaliar os resultados alcançados é fundamental para se ter uma visão crítica e mais segura sobre os reais benefícios e viabilidade de uso das tecnologias desenvolvidas.

Ao analisar-se o cenário atual de desenvolvimento de software, percebe-se que o avanço nas tecnologias de infraestrutura, desenvolvimento e especialização de demandas organizacionais traz a necessidade da elaboração de novas abordagens de apoio ao desenvolvimento do software (SAKAMURA, 2006).

Mais especificamente, observou-se ao longo da última década outra modificação no cenário de desenvolvimento de software. Este passou a ser também caracterizado pela proliferação de diferentes dispositivos com poder de processamento e redes de comunicação sem fio. Isto traz novas restrições, requisitos e premissas que ainda não são tratados pelas abordagens convencionais encontradas na área da engenharia de software. Este é o cenário encontrado quando se considera a computação ubíqua (ABOWD, 1999) (BANAVAR e BERNSTEIN, 2002) (KINDBERG e FOX, 2002) (DUCATEL *et al.*, 2003) (NIEMELA e LATVAKOSKI, 2004) (SAKAMURA, 2006). Como resultado destas demandas, pode-se observar nichos de pesquisas em diversas áreas como software embarcado, software para dispositivos móveis, ambientes inteligentes, dentre outros.

É importante observar que estas áreas surgiram a partir de demandas reais resultantes do rápido avanço tecnológico pelo qual se passa atualmente. E como demandas reais, devem ser tratadas de forma que os serviços disponibilizados pelos sistemas de software sejam desenvolvidos dentro de prazos, custos e níveis de qualidade previamente acordados e usualmente esperado quando se “engenheira” software. Entretanto, para se alcançar estes objetivos, o avanço das demandas também deve ser acompanhado por avanços na engenharia de software.

1.2 – Contexto e Motivação

Nos últimos anos, temos observado o surgimento e a evolução de um novo paradigma da computação (terceiro), a computação ubíqua. Esta evolução pode ser dividida em três grandes etapas (WEISER, 1991) conforme visto na Figura 1.1:

- Primeiro Paradigma: caracteriza-se por uma relação de 1 dispositivo com poder de processamento para N usuários. O destaque deste período é o uso de *mainframes*;
- Segundo Paradigma: caracteriza-se por uma relação de 1 dispositivo com poder de processamento para 1 usuário. O possibilitador deste paradigma foi o surgimento dos PCs (computadores pessoais);
- Terceiro Paradigma: caracteriza-se por uma relação de N dispositivos com poder de processamento para 1 usuário. Neste paradigma, temos como principais possibilitadores o avanço no desenvolvimento de dispositivos móveis com poder de processamento e a melhoria das redes de comunicação.

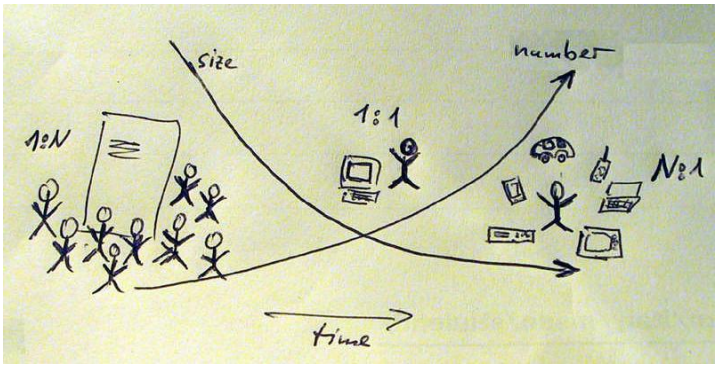


Figura 1.1. Paradigmas da computação (WEISER, 1991).

De forma geral, o objetivo da computação ubíqua é tornar os serviços computacionais embutidos no ambiente que nos cerca de forma que seu uso seja não intrusivo para os usuários (WEISER, 1991) (ABOWD, 1999).

A computação ubíqua tornou-se possível, em parte, devido aos avanços tecnológicos nas áreas de dispositivos móveis e redes de comunicação. Este avanço permite criar artefatos e ambientes que disponibilizam para o usuário recursos de comunicação e de processamento. A partir desta infraestrutura, é possível iniciar a construção de um novo conjunto de sistemas no sentido de disponibilizar diferentes tipos de serviços para os usuários, os sistemas ubíquos.

Estes carregam consigo um conjunto de características que definem a computação ubíqua. Estas características tendem a aumentar a complexidade do software construído assim como trazer desafios ao seu desenvolvimento. Estes desafios podem ser observados na necessidade de investigação de técnicas ou métodos de engenharia de software para apoiar a construção deste tipo de aplicação como: definição dos requisitos, desenvolvimento de metodologias, processos de software, abordagens de teste e técnicas de garantia da qualidade (DUCATEL *et al.*, 2003) (NIEMELA e LATVAKOSKI, 2004) (SAKAMURA, 2006).

Entretanto, enquanto pesquisas em computação ubíqua têm produzido resultados no campo de dispositivos, infraestrutura e aplicações, ainda existe uma demanda não atendida por investigações sobre como a engenharia de software pode apoiar o desenvolvimento de sistemas ubíquos (SPÍNOLA *et al*., 2006). Isto aumenta a complexidade e os riscos associados às atividades do desenvolvimento desta categoria de software e indica que as técnicas convencionais de engenharia de software podem não ser eficazes/eficientes para este tipo de software.

Desta forma, é importante que novas pesquisas sejam realizadas na engenharia de software considerando a computação ubíqua e seus desafios associados. Estes desafios podem estar distribuídos ao longo das diferentes etapas do ciclo de vida do software, por exemplo:

- Definição do Processo
    - Quais atividades devem ser acrescentadas ao processo para atender às necessidades advindas de cada característica da computação ubíqua?
- Planejamento do Projeto
    - Quais são os riscos associados a cada característica da computação ubíqua?
    - Como diminuir os riscos associados ao desenvolvimento de sistemas ubíquos?
- Requisitos
    - Como apoiar a captura dos requisitos de sistemas ubíquos?

    - Quais influências cada uma das características da computação ubíqua exerce sobre as atividades de levantamento e definição dos requisitos?
    - Qual abordagem utilizar para apoiar a verificação dos requisitos associados às características da computação ubíqua?
- Projeto
    - Quais são os impactos deste tipo de software em sua arquitetura?
    - Como avaliar a qualidade da arquitetura projetada?
- Codificação
    - Qual tecnologia ou conjunto de tecnologias utilizar?
- Testes
    - Quais abordagens de teste podem ser utilizadas?
    - Como garantir a qualidade das funcionalidades disponibilizadas no ambiente?

Esta listagem inicial nos permite observar que as oportunidades de pesquisa nesta área podem ser inúmeras e novas oportunidades podem ser apresentadas caso seja feita uma análise mais detalhada do assunto. A lista apresentada teve apenas o objetivo de ilustrar algumas das oportunidades de pesquisa identificadas ao longo deste trabalho.

Desta forma, a falta de respostas para estas perguntas representam lacunas a serem preenchidas. À medida que a computação ubíqua estiver sendo mais aplicada para apoiar a solução de problemas críticos do mundo real, técnicas de engenharia de software eficazes e eficientes para esta área de aplicação irão se tornar cada vez mais vitais para o sucesso dos produtos desenvolvidos.

1.3 – Questões de Pesquisa

O aumento da complexidade no desenvolvimento de projetos de software ao se considerar o cenário da ubiquidade computacional faz com que as técnicas tradicionais da engenharia de software tenham sua eficácia reduzida. Isto por que projetos de software ubíquos trazem novos desafios relacionados às características da computação ubíqua e estes desafios não têm sido considerados pelas técnicas tradicionais (ABOWD, 1999) (BANAVAR e BERNSTEIN, 2002) (KINDBERG e FOX, 2002).

É preciso lidar com estes riscos para que se possa ter mais segurança na execução das atividades do projeto de forma que este seja entregue com qualidade e dentro de prazos e custos estabelecidos. Considerando o cenário da ubiquidade

computacional, vários fatores podem auxiliar a mitigar os riscos do projeto, por exemplo:

- Definição de um processo adequado às características de ubiquidade do software;
- Apoiar a definição de requisitos para um domínio específico de conhecimento (neste caso, de ubiquidade);
- Efetuar atividades de verificação dos requisitos de ubiquidade;
- Utilizar uma tecnologia adequada a este novo cenário de desenvolvimento;
- Definir estratégias e técnicas de testes eficientes para lidar com as diferentes características de ubiquidade.

Ao analisar os trabalhos de (BOEHM, 1981) (WHEELER *et al*., 1996) (BOEHM e BASILI, 2001), pode-se considerar que uma das principais formas de minimizar impactos negativos em fases avançadas do desenvolvimento de um projeto de software (e como consequência, diminuir seus riscos) é reduzir a inserção de defeitos nas fases iniciais do desenvolvimento e criar mecanismos que possam identificar os defeitos nas fases em que são inseridos de forma que não sejam encontrados em fases posteriores do desenvolvimento, o que traria como consequência aumento de custo para sua correção e retrabalho.

Um defeito (*fault*) é um ato inconsciente cometido por um indivíduo ao tentar entender uma determinada informação, resolver um problema ou utilizar um método ou uma ferramenta (IEEE 610, 1990).

Diferentes tipos de defeitos podem ser identificados ao longo do desenvolvimento do software. A partir das constatações reportadas em (BOEHM, 1981) (WHEELER *et al*., 1996) (BOEHM e BASILI, 2001), nesta pesquisa optou-se por considerar defeitos associados aos requisitos. Desta forma, a taxonomia de defeitos definida em (SHULL *et al.,* 2000) será utilizada:

- Omissão: (1) Algum requisito importante relacionado à funcionalidade, ao desempenho, às restrições de projeto, ao atributo, ou à interface externa não foi incluído; (2) não está definida a resposta do software para todas as possíveis situações de entrada de dados; (3) faltam seções na especificação de requisitos; (4) faltam referências de figuras, tabelas, e diagramas; (5) falta definição de termos e unidades de medidas.
- Ambiguidade: Um requisito possui várias interpretações devido a diferentes termos utilizados para uma mesma característica ou vários significados de um termo para um contexto em particular.

- Inconsistência: Dois ou mais requisitos são conflitantes.
- Fato Incorreto: Um requisito descreve um fato que não é verdadeiro, considerando as condições solicitas para o sistema.
- Informação Estranha: As informações fornecidas no requisito não são necessárias ou mesmo usadas.

Neste contexto, esta pesquisa apresenta uma abordagem para apoiar a **definição** e **verificação** (garantia da qualidade) de **requisitos funcionais de ubiquidade** em projetos de software. Entendemos que é *possível minimizar os riscos envolvidos com a construção de projetos de software ubíquo* ***reduzindo*** *problemas de omissão, fato incorreto, ambiguidade, informação estranha e inconsistência nos requisitos funcionais de ubiquidade do projeto e* ***aumentando*** *a satisfação do usuário ao executar atividades associadas à definição dos requisitos funcionais de ubiquidade através do uso de uma abordagem de apoio à definição e verificação de requisitos de ubiquidade.*

Para mostrar esta possibilidade, os sequintes questionamentos necessitam ser respondidos:

Q1: O uso de uma abordagem de apoio à definição dos requisitos funcionais de ubiquidade reduz o número de defeitos (omissão, fato incorreto, ambiguidade, informação estranha e inconsistência) inseridos nos requisitos funcionais de ubiquidade e aumenta a satisfação do usuário ao executar esta atividade?

Q2: O uso de uma abordagem de apoio à verificação dos requisitos funcionais de ubiquidade faz com que o número de defeitos (omissão, fato incorreto, ambiguidade, informação estranha e inconsistência) identificados ao efetuar a revisão dos requisitos funcionais de ubiquidade do projeto seja aumentado ao mesmo tempo em que aumenta a satisfação do usuário ao executar esta atividade?

Reconhecemos a importância das características restritivas para o desenvolvimento de software. Entretanto, a decisão em apoiar nesta pesquisa a definição e verificação dos requisitos funcionais de ubiquidade em projetos de software está fundamentada no fato de:

- trabalhos anteriores (SPÍNOLA e*t al.*, 2007) terem observado que pesquisas no campo da computação ubíqua têm sido mais focadas nos requisitos funcionais deste tipo de software. Este conjunto de conhecimento tende a ser mais consolidado, minimizando os riscos da pesquisa;
- existirem diferentes técnicas tradicionais da engenharia de software de apoio às atividades de verificação de requisitos funcionais já avaliadas experimentalmente (SILVA *et. al.*, 2004) (MAFRA e TRAVASSOS, 2006).

Este conjunto de técnicas pode ser utilizado como base para apoiar a definição de abordagens de verificação de requisitos de ubiquidade.

Sendo assim, **não** serão tratados ao longo desta pesquisa:

- Definição de requisitos não funcionais de ubiquidade;
- Verificação de requisitos não funcionais de ubiquidade;

Além disso, considerando as atividades base da engenharia de requisitos (elicitação, modelagem, análise, especificação ou definição, verificação, validação), serão tratadas apenas as atividades de definição e verificação dos requisitos.

1.4 – Objetivos

Desta forma, a meta deste trabalho consiste em definir uma abordagem que auxilia o engenheiro de software durante as atividades de definição e verificação dos requisitos funcionais de ubiquidade do projeto através de um conjunto de guias baseadas em conhecimento do domínio da computação ubíqua. Para isto, esta pesquisa seguirá duas linhas de trabalho: (1) organização de um corpo de conhecimento em computação ubíqua baseado em resultados de estudos experimentais, e; (2) definição de guias de apoio às atividades de definição e verificação dos requisitos funcionais de ubiquidade em projetos de software de forma a atender às necessidades específicas da engenharia de requisitos no contexto da computação ubíqua.

Esta meta pode ser decomposta em sete objetivos:

1. **Organizar um corpo de conhecimento em computação ubíqua:** estruturar através do uso de revisões sistemáticas um conjunto de características que definem a computação ubíqua;
2. **Qualificar o corpo de conhecimento em computação ubíqua:** identificar através do uso de *survey* a abrangência, pertinencia e relevância das características que compõem o corpo de conhecimento;
3. **Definir um processo de apoio à definição e verificação de requisitos de ubiquidade:** estabelecer atividades de apoio à definição e verificação de requisitos assim como os artefatos consumidos e produzidos;
4. **Elaborar modelos conceituais representando os fatores associados a cada uma das características de ubiquidade:** definir os principais conceitos contidos no corpo de conhecimento sobre computação ubíqua e como eles estão relacionados;

5. **Apoiar a especificação de requisitos funcionais de ubiquidade:** apoiar, a partir do uso do corpo de conhecimento em computação ubíqua, a especificação de requisitos funcionais de ubiquidade do software. Este apoio ocorre em dois momentos: direcionamento para identificação das informações necessárias e especificação das informações capturadas;
6. **Apoiar a verificação de requisitos funcionais de ubiquidade:** definir uma técnica para apoiar o engenheiro de software na atividade de verificação dos requisitos de ubiquidade do projeto e que auxilie a identificação de defeitos em fases iniciais do desenvolvimento;
7. **Apoiar a evolução do corpo de conhecimento em computação ubíqua:** estabelecer um protocolo de evolução que permita manter o corpo de conhecimento baseado na re-execução simplificada da revisão sistemática.

1.5 – Metodologia e Histórico da Pesquisa

De acordo com Kitchenham *et al.* (2004), a utilização de processos com qualidade por engenheiros de software não é condição suficiente para a melhoria da qualidade no desenvolvimento. Isto por que o desenvolvimento é dependente de tecnologias que muitas vezes não apresentam evidências suficientes sobre seus potenciais benefícios, limitações, custo de implantação e riscos associados. Para lidar com isto, Kitchenham *et al*. (2004) indicam que o uso de evidências permitiria a caracterização de uma tecnologia antes de sua adoção em projetos de software na indústria de forma que fosse possível determinar com níveis razoáveis de segurança a viabilidade de sua utilização considerando cenários específicos de uso.

Desta forma, minimizar os riscos na introdução de novas tecnologias na indústria pode ser considerado um fator importante para a evolução da engenharia de software como ciência. Neste contexto, Shull *et al.* (2001) propuseram uma abordagem para a introdução de processos de software na indústria através dos quais novas tecnologias podem ser inseridas. Tal abordagem apresenta uma série de questões que devem ser tratadas durante a avaliação dos processos de software, assim como os tipos de estudos experimentais que contemplam essas questões. Sua aplicação possibilita uma introdução gradativa da tecnologia em aplicações industriais, o que permite a realização de estudos intermediários e com diferentes níveis de controle e, consequentemente, de abrangência de seus resultados.

Entretanto, esta metodologia não contempla a etapa da definição de uma tecnologia de software. Ela parte do princípio de que já existe uma versão inicial da tecnologia a ser avaliada. Neste sentido, Mafra *et al.* (2006) estenderam a metodologia

proposta por Shull *et al.* (2001) através da introdução de uma abordagem formal baseada em estudos secundários para a identificação de evidência existente na literatura técnica como apoio à concepção de novas tecnologias de software. Desta forma, através da condução de revisões sistemáticas, seria possível identificar e caracterizar o grau de evidência experimental existente na área, e como consequência, eventuais dificuldades e incertezas que povoam o processo de definição de uma nova tecnologia poderiam ser minimizadas.

Embora o uso de estudos secundários (revisões sistemáticas) traga benefícios ao desenvolvimento de novas tecnologias através da identificação das evidências geradas, caracterizando assim o estado da arte em uma determinada área de pesquisa (KITCHENHAM *et al*., 2004), ainda não permite capturar um questionamento importante a respeito de tal evidência: “Qual a opinião de especialistas da área em relação às evidências obtidas?”. A resposta para essa pergunta é um ponto essencial para a continuidade do desenvolvimento de tal tecnologia. Para lidar com esta questão, a abordagem definida em (MAFRA *et al.*, 2006) foi evoluída de duas formas a fim de se obter ao final um corpo de conhecimento mais consolidado e abrangente para apoiar pesquisas em um campo de conhecimento:

- transformar a execução dos estudos secundários em um processo mais interativo onde, ao término da análise da cada estudo, é feita uma avaliação sobre a necessidade de refinamento e execução de uma revisão da literatura complementar;
- utilizar estudos primários na avaliação do conhecimento organizado a partir dos resultados obtidos com a execução de estudos secundários através da participação de pesquisadores externos que atuam na mesma linha de pesquisa.

Neste contexto, Spínola *et al*. (2008) propuseram uma abordagem definida a partir da proposta presente em (SHULL *et al*., 2001) e (MAFRA *et al.*, 2006) que faz uso de estudos secundários e primários no apoio à concepção de novas tecnologias.

Sendo assim, a metodologia de pesquisa seguida para o desenvolvimento deste trabalho foi fundamentada na abordagem para desenvolvimento de novas tecnologias de software baseada em estudos primários e secundários. Ela se divide em duas fases: concepção (definida em SPÍNOLA *et al*. (2008) e aprimorada em DIAS-NETO *et al*. (2010)) e avaliação da abordagem proposta (definida em SHULL *et al.* 2001), conforme apresentado na Figura 1.2.

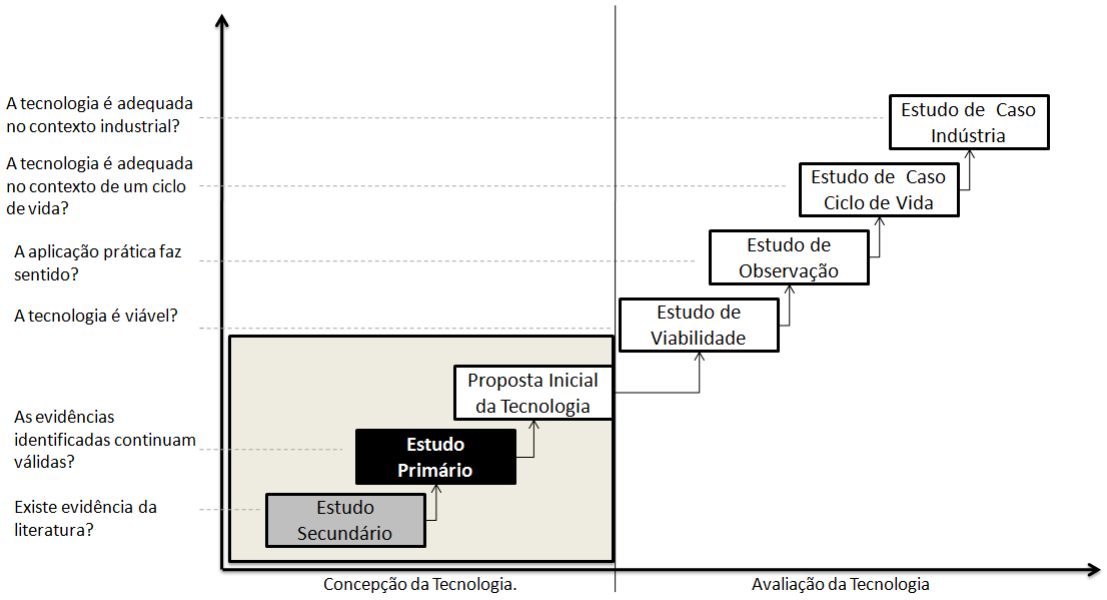


Figura 1.2. Metodologia de Pesquisa adotada. Adaptado de (DIAS-NETO *et al.*, 2010).

1.5.1 Fase de Concepção da Tecnologia

A fase de concepção da tecnologia considera alguns passos e envolve a execução de estudos secundários e/ou primários com o objetivo de se obter uma proposta inicial da tecnologia conforme pode ser observado na Figura 1.3.

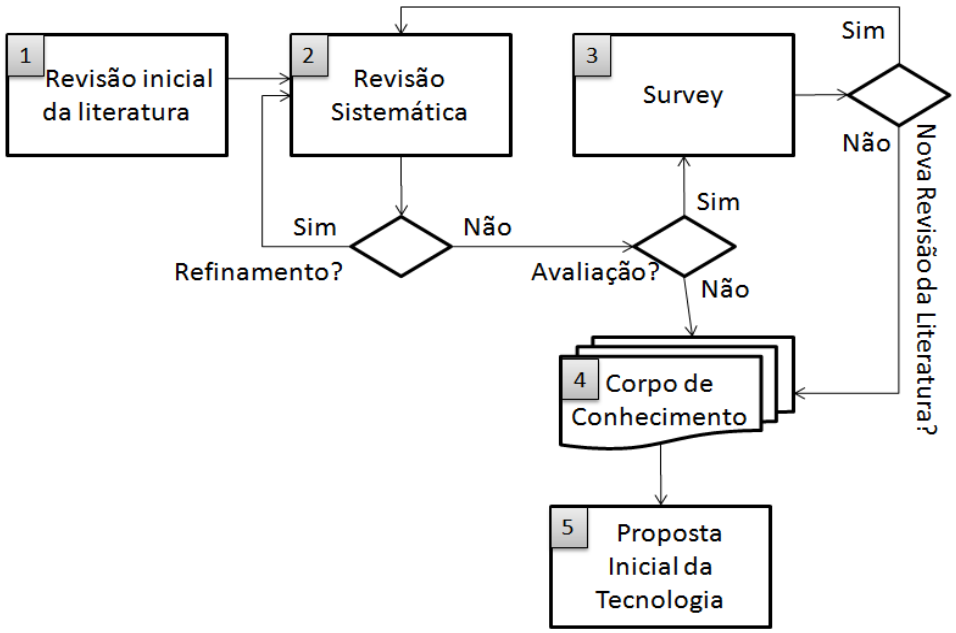


Figura 1.3. Fase de Concepção da Tecnologia. Adaptado de (DIAS-NETO *et al.*, 2010).

A seguir, tem-se uma breve descrição das atividades desempenhadas neste trabalho considerando os passos presentes na Figura 1.3. O resultado final da

execução deste conjunto de passos é a definição da abordagem proposta neste trabalho:

- **Revisão Inicial da Literatura:** executada no primeiro semestre de 2005, nesta atividade realizou-se uma revisão *ad hoc* da literatura sobre computação ubíqua. O foco desta revisão inicial foi entender os conceitos básicos desta área de pesquisa;
- **Identificação das Características da Computação Ubíqua (revisão sistemática):** nesta atividade realizou-se uma revisão sistemática da literatura. Os fundamentos da revisão sistemática (estudo secundário) auxiliam na obtenção de resultados mais justos e precisos[1] sobre o assunto que se pretende pesquisar (BIOLCHINI *et al.*, 2005). Executada no segundo semestre de 2005, o foco desta revisão foi investigar o estado da arte e da prática da computação ubíqua identificando sua definição, suas características e exemplos de sistemas ubíquos desenvolvidos. Os resultados deste estudo serão apresentados no Capítulo 2 deste trabalho;
- **Identificação dos Fatores Funcionais e Restritivos das Características (revisão sistemática):** em complemento à primeira revisão sistemática, foi considerada neste trabalho a execução de uma segunda revisão cujo objetivo foi identificar os fatores funcionais e restritivos associados a cada uma das características da computação ubíqua identificadas no passo anterior de forma a possibilitar a caracterização de projetos de software segundo as características e respectivos fatores de ubiquidade. Este estudo foi executado no primeiro semestre de 2006 e seus resultados serão apresentados no Capítulo 2 deste trabalho;
- **Caracterização de Projetos de Software Ubíquo:** no primeiro semestre de 2007, o conjunto de características e fatores da computação ubíqua foi organizado em um corpo de conhecimento e utilizado no apoio à caracterização de projetos de software. O resultado desta caracterização, apresentada no Capítulo 3, nos levou à necessidade de avaliar o corpo de conhecimento organizado considerando outros pesquisadores da área. Além disso, baseado nas duas revisões sistemáticas da literatura, foi elaborado um protocolo simplificado de atualização do corpo de

[1] Os termos justo e preciso indicam que revisões sistemáticas são elaboradas de forma a evitar que os resultados da execução da revisão seja direcionado apenas pelo conhecimento do autor da revisão. Além disso, indicam também que estas revisões são planejadas de forma que possam ser replicadas por outros pesquisadores levando a resultados semelhantes.

conhecimento. Sua definição e os resultados de sua execução no primeiro semestre de 2010 estão apresentados no Capítulo 4;

- **Avaliação Inicial (*Survey I*):** executada no primeiro semestre de 2008, o objetivo desta atividade foi analisar as características da computação ubíqua, seus fatores e grupos de fatores extraídos da literatura técnica com o objetivo de caracterizá-los com respeito a sua aplicabilidade e escopo no contexto de projetos de software ubíquo. O resultado de sua execução permitiu a evolução do corpo de conhecimento organizado sobre computação ubíqua. Os resultados deste estudo serão apresentados no Capítulo 4 deste trabalho. Ao final, foi identificada a necessidade de se realizar uma avaliação mais abrangente do corpo de conhecimento;
- **Avaliação do Corpo de Conhecimento (*Survey II*):** executada no primeiro semestre de 2009, o objetivo desta atividade foi analisar as características da computação ubíqua extraídas da literatura técnica e aprimoradas na avaliação inicial, com o objetivo de caracterizá-las, com respeito à sua pertinência e relevância na caracterização de projetos de software sob o ponto de vista de pesquisadores em computação ubíqua. O resultado de sua execução levou à estruturação final do corpo de conhecimento. Os resultados deste estudo também serão apresentados no Capítulo 4 deste trabalho;
- **Identificação de Trabalhos Relacionados:** executada no primeiro semestre de 2008, o foco desta revisão da literatura (baseada em princípios da revisão sistemática) foi identificar abordagens que apóiem a definição e a verificação de requisitos de ubiquidade em projetos de software. Os resultados desta atividade estão descritos no Capítulo 5;
- **Definição da Proposta:** executada durante os anos de 2008 e 2009, esta atividade foi desenvolvida de forma incremental e utilizou como base o corpo de conhecimento definido ao longo deste período. Ao final, chegou-se à abordagem para apoiar a definição e garantia da qualidade de requisitos funcionais de ubiquidade em projetos de software. Os resultados desta etapa serão apresentados no Capítulo 6 deste trabalho.

1.5.2 Fase de Avaliação da Tecnologia

Concluída a fase de concepção, passou-se à etapa de avaliação da abordagem proposta. Para tal, foram realizados dois estudos de viabilidade considerando os dois pilares da abordagem proposta: definição de requisitos e verificação de requisitos.

- **Estudo de Viabilidade – Definição de Requisitos:** executado no final do primeiro semestre de 2009, este estudo teve como objetivo analisar a abordagem de apoio à definição de requisitos de ubiquidade com o propósito de caracterizá-la com respeito à sua aplicabilidade no apoio às atividades de definição de requisitos funcionais de ubiquidade em projetos de software. Os resultados deste estudo serão apresentados no Capítulo 7 deste trabalho.
- **Estudo de Viabilidade – Verificação de Requisitos:** executado no primeiro semestre de 2010, este estudo teve como objetivo analisar a abordagem de apoio à verificação dos requisitos de ubiquidade com o propósito de caracterizar, com respeito à redução do esforço (em tempo), aumento do número de defeitos identificados e satisfação do usuário no contexto da verificação de requisitos funcionais de ubiquidade em projetos de software. Os resultados deste estudo serão apresentados no Capítulo 8 deste trabalho.

A Figura 1.4 apresenta um resumo das etapas que compõem a metodologia de pesquisa adotada neste trabalho, assim como os tipos de estudos conduzidos e os resultados obtidos. Já a Figura 1.5 representa as atividades desempenhadas nesta pesquisa na linha do tempo.

A partir da realização das atividades apresentadas na Figura 1.4, os seguintes resultados foram obtidos:

- Entendimento inicial sobre a computação ubíqua:
    - SPÍNOLA, R.O., SILVA, J.L.M., TRAVASSOS, G.H. (2006) “Towards a Conceptual Framework to Classify Ubiquitous Software Projects”. Proceedings of the 8th International Conference on Software Engineering and Knowledge Engineering (SEKE), San Francisco, USA.
- Definição e Características da Computação Ubíqua e abordagem para caracterização de projetos de software:
    - SPÍNOLA, R.O., SILVA, J.L.M., TRAVASSOS, G.H., 2007. “Checklist to Characterize Ubiquitous Software Projects”. Proceedings of the XXI Brazilian Symposium on Software Engineering, João Pessoa, Brasil.
    - SPÍNOLA, R.O., SILVA, J.L.M., TRAVASSOS, G.H., 2007. Characterizing Ubicomp Software Projects through a Checklist. In

Proceedings of the I Workshop on Pervasive and Ubiquitous Computing, Gramado, Brasil.

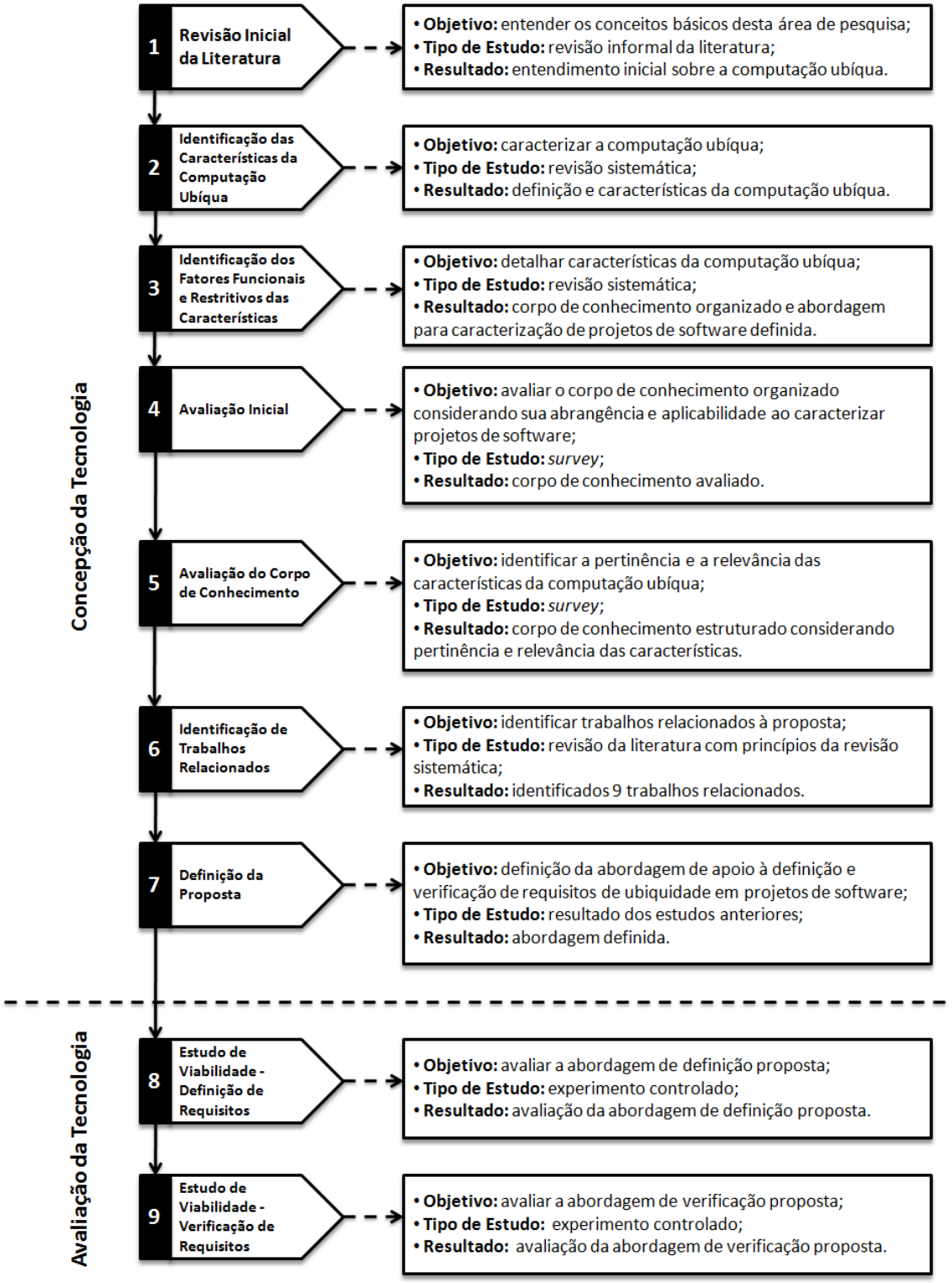


Figura 1.4. Resumo dos Estudos conduzidos ao longo desta pesquisa.

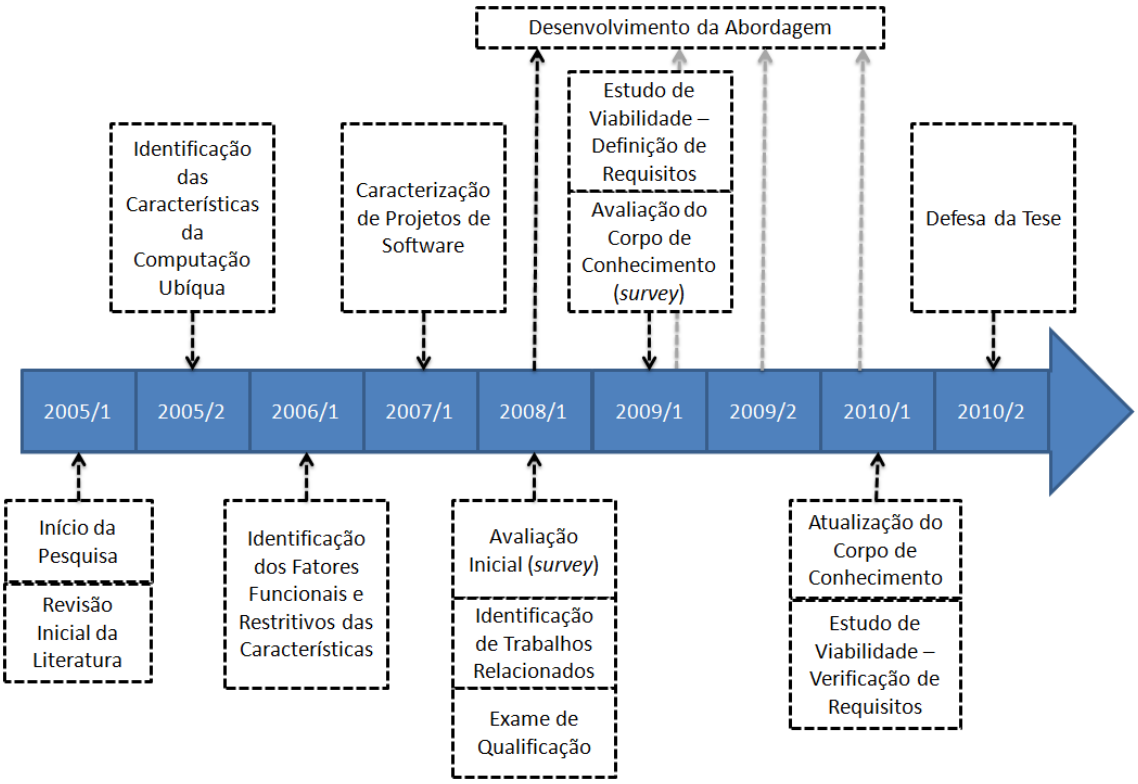


Figura 1.5. Distribuição das atividades realizadas na linha de tempo.

- Abordagem de apoio à definição e verificação de requisitos de ubiquidade em projetos de software:
    - SPÍNOLA, R. O.; PINTO, F.R.; TRAVASSOS, G.H., 2008. Apoio às Atividades de Especificação e Verificação de Requisitos Funcionais de Ubiquidade em Projetos de Software. In: International Information and Telecommunication Technologies Symposium - I2TS'2008, Foz do Iguaçu.
    - SPÍNOLA, R. O. ; TRAVASSOS, G.H., 2008. Arcabouço para Apoiar a Definição e a Garantia de Qualidade de Requisitos de Ubiquidade em Projetos de Software. In: II Workshop on Pervasive and Ubiquitous Computing, 2008, Campo Grande.
    - SPÍNOLA, R. O., PINTO, F. C. R., TRAVASSOS, G.H., 2008. Supporting Requirements Definition and Quality Assurance in Ubiquitous Software Project. In: 3rd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation, 2008, Kassandra, Chalkidiki, Greece.
- Corpo de conhecimento em computação ubíqua estruturado:
    - SPÍNOLA, R.O., and TRAVASSOS, G.H., 2010. Characteristics of Ubiquitous Software Projects: Pertinence, Relevance, and Use. In

Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering (SEKE 2010), San Francisco Bay, USA.

- Avaliação experimental da abordagem de apoio à definição de requisitos de ubiquidade em projetos de software ubíquo:
    - SPÍNOLA, R.O.; PINTO, F.C.R.; TRAVASSOS, G.H., 2010. *UbiCheck*: An Approach to Support Requirements Definition in the Ubicomp Domain. In: 25th Symposium On Applied Computing, 2010, Sierre. 25th Symposium On Applied Computing, 2010. p. 306-310, Sierre, Suiça.

1.6 – Organização da Tese

Este capítulo apresentou algumas das idéias que motivaram o desenvolvimento desta tese, a metodologia de pesquisa utilizada para definição da proposta e a hipótese que direciona esta pesquisa. Estes tópicos serão refinados ao longo dos próximos capítulos.

A organização do texto deste trabalho segue a linha de tempo em que as atividades de pesquisa foram realizadas (Figura 1.5) e está estruturada segundo os itens abaixo:

- **Capítulo 2 – Computação Ubíqua: Características e Fatores:** apresenta os conceitos relacionados à ubiquidade computacional, o conjunto de características que a definem de acordo com a investigação da literatura técnica e o conjunto de fatores funcionais e restritivos associados a estas características;
- **Capítulo 3 – Caracterização de Projetos de Software Ubíquo:** apresenta uma abordagem para caracterização de projetos de software e seu uso na caracterização de 17 projetos;
- **Capítulo 4 – Características da Computação Ubíqua: Pertinência e Relevância:** apresenta a avaliação do corpo de conhecimento em computação ubíqua organizado no Capítulo 2. Para isso, foram planejados e executados dois questionários envolvendo pesquisadores desta linha de pesquisa. O resultado desta avaliação é a definição da pertinência e relevância de cada uma das características;
- **Capítulo 5 – Engenharia de Requisitos e Computação Ubíqua:** apresenta o planejamento e os resultados alcançados com a execução de uma revisão da

literatura cujo objetivo foi identificar abordagens de apoio à definição e garantia de qualidade de requisitos de ubiquidade em projetos de software;

- **Capítulo 6 – Abordagem de Apoio à Definição e Verificação de Requisitos Funcionais de Ubiquidade em Projetos de Software:** apresenta a abordagem para apoiar a definição (*UbiCheck*) e verificação (*UbiVeri*) de requisitos funcionais de ubiquidade em projetos de software. A abordagem é fundamentada em três elementos: o corpo de conhecimento em computação ubíqua, um processo de apoio às atividades de definição de requisitos e um processo de apoio às atividades de verificação de requisitos;
- **Capítulo 7 – Avaliação de *UbiCheck*:** descreve o plano e o resultado do estudo para avaliar os benefícios que o uso da abordagem de apoio à definição de requisitos funcionais de ubiquidade em projetos de software traz. Ao final, é apresentada a evolução da abordagem a partir dos resultados obtidos no estudo;
- **Capítulo 8 – Avaliação de *UbiVeri*:** descreve o plano e o resultado do estudo para avaliar os benefícios que o uso da abordagem de apoio à verificação de requisitos funcionais de ubiquidade em projetos de software traz;
- **Capítulo 9 – Considerações Finais:** apresenta as considerações finais sobre esta pesquisa, seus resultados obtidos, as limitações do trabalho e futuros direcionamentos para a continuidade desta pesquisa.

Além disto, este trabalho contém 11 apêndices:

- **APÊNDICE A – ARTIGOS UTILIZADOS NAS REVISÕES SISTEMÁTICAS:** apresenta a lista de artigos retornados pelas revisões sistemáticas executadas neste trabalho e que foram descritas no Capítulo 2.
- **APÊNDICE B – CHECKLIST DE CARACTERIZAÇÃO:** apresenta o *checklist* elaborado para caracterizar projetos de software ubíquo de acordo com a sua aderência às características e fatores de ubiquidade.
- **APÊNDICE C – PROTOCOLO DE ATUALIZAÇÃO DO CORPO DE CONHECIMENTO:** apresenta o protocolo de atualização do corpo de conhecimento em computação ubíqua.
- **APÊNDICE D – MODELOS CONCEITUAIS:** apresenta os modelos conceituais elaborados para cada uma das características funcionais de ubiquidade.
- **APÊNDICE E – QUESTIONÁRIO DE CARACTERIZAÇÃO:** apresenta o questionário de caracterização de projetos de software ubíquo.

- **APÊNDICE F – GUIA GERAL DE DEFINIÇÃO DE REQUISITOS DE UBIQUIDADE:** apresenta o Guia Geral de Definição de Requisitos de Ubiquidade.
- **APÊNDICE G – GUIA GERAL DE VERIFICAÇÃO DE REQUISITOS DE UBIQUIDADE:** apresenta o Guia Geral de Verificação de Requisitos de Ubiquidade.
- **APÊNDICE H – INSTRUMENTOS DO PLANO DO EXPERIMENTO PARA AVALIAÇÃO DE UBICHECK:** descreve os instrumentos usados no estudo experimental realizado neste trabalho que avaliou a abordagem *UbiCheck*, de apoio à definição de requisitos de ubiquidade em projeto de software.
- **APÊNDICE I – CENÁRIOS DE UBÍQUIDADE:** apresenta os cenários de ubiquidade utilizados nos estudos descritos nos Capítulos 7 e 8.
- **APÊNDICE J – INSTRUMENTOS DO PLANO DO EXPERIMENTO PARA AVALIAÇÃO DE UBIVERI:** descreve os instrumentos usados no estudo realizado ao longo deste trabalho que avaliou a abordagem *UbiVeri*, de apoio à verificação de requisitos de ubiquidade em projeto de software ubíquo.
- **APÊNDICE K – GUIA GERAL DE DEFINIÇÃO DE REQUISITOS DE UBIQUIDADE COM DIRECIONAMENTO:** apresenta a versão do Guia Geral de Definição de Requisitos de Ubiquidade presente em *UbiCheck* 2.0.

Capítulo 2 – Computação Ubíqua: Características e Fatores

Neste capítulo são apresentados conceitos relacionados à ubiquidade computacional, o conjunto de características que a definem de acordo com a investigação da literatura técnica e o conjunto de fatores funcionais e restritivos associados a estas características.

2.1 - Introdução

As tecnologias mais avançadas são aquelas que desaparecem sob a perspectiva de seus usuários. Na computação ubíqua, os computadores estão embutidos no ambiente que nos cerca, criando um novo paradigma de acesso e manipulação de informação (WEISER, 1991). Dessa forma, a computação ubíqua pressupõe forte integração com o mundo real, mantendo alta transparência e o foco no usuário.

A proliferação de vários tipos de dispositivos computacionais, muitos deles com possibilidade de conexão sem fio, permite o surgimento desta nova área de aplicação que transcende as características dos sistemas atualmente em uso (LI e LANDAY, 2008). Neste contexto, a computação ubíqua demanda abrir diversas frentes de pesquisas. É uma área multidisciplinar envolvendo pesquisas nas áreas de redes de computadores, otimização, processamento de sinais e inteligência artificial, dentre outros (RUSSEL *et al.*, 2005). Como fruto desta multidisciplinaridade, o termo ubiquidade computacional passou a ser usado de forma bastante abrangente.

Entretanto, embora a questão da aplicação da computação ubíqua venha sendo explorada em diferentes contextos, não foi identificado de forma centralizada um trabalho contendo sua definição considerando as características[2][3] e fatores[4] que

[2] As definições para as palavras características e fatores foram extraídas do dicionário inglês Cambridge. Isso se deve ao fato destas palavras terem sido identificadas em inglês para comporem o conjunto de palavras chave utilizadas nas revisões sistemáricas apresentadas neste capítulo.

[3] De acordo com o dicionário Cambridge, característica: é uma qualidade de alguém ou alguma coisa que pode ser notada. Sendo assim, no contexto deste trabalho iremos considerar característica da computação ubíqua como sendo uma qualidade de software ubíquo que pode ser notada.

sistemas voltados para a computação ubíqua devam contemplar (SPÍNOLA *et. al*., 2006).

Desta forma, visando conhecer uma definição mais abrangente e atual sobre computação ubíqua, optou-se por executar uma revisão sistemática da literatura. Esta revisão teve o objetivo de investigar o estado da arte e da prática da computação ubíqua contemplando três questionamentos (SPÍNOLA *et. al*., 2006):

- O que é computação ubíqua?
- Como a computação ubíqua se apresenta atualmente?
- Que características definem aplicações para a computação ubíqua?

Dentre alguns resultados, obteve-se:

- uma definição, a princípio, mais completa e menos ambígua para computação ubíqua;
- uma definição para sistemas ubíquos;
- um conjunto inicial de características de ubiquidade que atualiza a definição de computação ubíqua.

Neste momento optou-se pela execução de uma nova revisão sistemática da literatura com objetivo de detalhar cada uma das características de ubiquidade atentando para seus fatores funcionais e restritivos. A seguir serão apresentados os resultados dos dois Estudos Secundários executados.

2.2 – Características da Computação Ubíqua

Considerando o cenário apresentado na introdução deste capítulo, foi planejada e executada uma revisão sistemática com o objetivo de elaborar uma definição de computação ubíqua e identificar suas características a partir do conhecimento disponível na literatura técnica.

Para isso, um protocolo para busca e análise foi adotado. O planejamento desta revisão, assim como os resultados alcançados estão apresentados nas próximas subseções.

[4] *De acordo com o dicionário Cambridge, fator é um fato ou situação que influencia o resultado de algo. No contexto deste trabalho, consideramos fator como sendo um fato ou situação que influencia uma determinada característica de software ubíquo.*

2.2.1 Planejamento e Execução da Revisão Sistemática

Os passos que envolvem o planejamento de uma revisão sistemática são: definição do objetivo e questões de pesquisa, seleção das fontes de busca, definição das *strings* de busca e dos critérios de inclusão/exclusão dos artigos, e definição das estratégias de classificação e informações a serem extraídas de cada artigo. Usando a abordagem descrita em (BIOLCHINI *et al*., 2005), esses passos estão sumarizados a seguir:

- Objetivo: investigar o estado da arte e da prática da computação ubíqua;
- Questões de pesquisa:

  P0: O que é computação ubíqua?

  P1: Como a computação ubíqua se apresenta atualmente?

  P2: Quais são as características básicas que definem sistemas ubíquos?
- Fontes de busca: cinco bibliotecas digitais (IEEEXplorer, ACM Portal, INSPEC, Compendex IE, Web of Science) e anais de conferências;
- Critérios de Inclusão e Exclusão de artigos.
  - Os artigos devem estar disponíveis na web;
  - Os artigos devem estar escritos em inglês;
  - Os artigos devem contemplar a definição de computação ubíqua (exclusivo para **P0**);
  - Os artigos devem relatar aplicações atuais que utilizam conceitos de computação ubíqua (exclusivo para **P1**);
  - Os artigos devem conter projetos de software ubíquo. Assim, não serão consideradas aplicações que estejam relacionadas a software básico (sistemas operacionais, sistemas de rede, compiladores, protocolos) (exclusivo para **P1**);
  - Os artigos devem apresentar características que definem sistemas ubíquos. Assim, não serão considerados artigos que trazem somente características ligadas ao domínio da aplicação (exclusivo para **P2**).
- *Strings* de busca:
  - Para **P0**:
    - (ubiquitous computing <or> pervasive computing) <and> (definition <or> characterization <or> concept)
  - Para **P1**:
    - ubiquitous application <or> ubiquitous system <or> ubiquitous software

        - pervasive application <or> pervasive system <or> pervasive software
    - Para **P2**:
        - (ubiquitous computing <or> pervasive computing) <and> (feature <or> requirement <or> characteristic)
        - (ubiquitous application <or> ubiquitous system <or> ubiquitous software) <and> (feature <or> requirement <or> characteristic)
        - (pervasive application <or> pervasive system <or> pervasive software) <and> (feature <or> requirement <or> characteristic)

A partir deste protocolo, a revisão da literatura foi executada. Somados os resultados, obteve-se um total de 751 artigos retornados. Dois pesquisadores deram início ao processo de filtragem (todos os artigos foram analisados pelos dois pesquisadores). Aplicada a filtragem considerando os critérios de inclusão e exclusão previamente definidos, chegou-se a uma quantidade de 112 artigos para leitura e extração dos resultados. Ainda assim, foram eliminados alguns artigos em decorrência de estarem repetidos em *strings* de busca diferentes de uma mesma pergunta e por não atenderem aos critérios de inclusão durante a leitura integral do artigo. Desta forma, chegou-se a um número final de 57 artigos (listados no Apênice A.1), divididos entre os dois pesquisadores participantes da revisão sistemática para extração dos resultados e análise para:

- **elaborar uma definição para computação ubíqua e sistemas ubíquos:** estas foram formalizadas a partir do conjunto de definições extraídas de cada artigo;
- **identificar aplicações onde os conceitos de ubiquidade estão sendo usados:** foram extraídos diretamente dos artigos as descrições das aplicações ubíquas, suas características e contexto de utilização;
- **definir as características básicas que definem sistemas ubíquos:** foi extraído de cada artigo uma lista de características com as respectivas definições. Posteriormente, essas definições foram analisadas para eliminar redundâncias e obter um conjunto final de características.

2.2.2 Definição de Computação Ubíqua

Mark Weiser (WEISER, 1991) define computação ubíqua como sendo o uso do computador através de sua disponibilidade no meio físico, tornando-o efetivamente invisível para o usuário. Ou seja, o computador está de tal forma integrado ao ambiente que sua utilização se dá de forma não intrusiva. Esta é a definição que marcou a origem do termo Computação Ubíqua e, embora tenha grande importância por apresentar um novo paradigma da computação, não é completa. A falta de completude está evidenciada ao não explicitar quais características um software deve contemplar para ser considerado ubíquo e isto reflete, de certa forma, a proposta inovadora e também as limitações tecnológicas encontradas na época em que a definição foi proposta.

Partindo deste ponto, um conjunto de artigos foi analisado em busca de uma definição mais abrangente para o tema. O que se encontrou foi um conjunto de variações sobre a definição elaborada por (WEISER, 1991). Alguns exemplos dessas variações são:

- Computação ubíqua se apresenta quando a computação é não intrusiva, móvel, flexível e altamente integrada ao ambiente de trabalho (DISZ *et al.*, 1997);
- Computação ubíqua representa o conceito de computação em todo o lugar, fazendo com que o uso da computação e a comunicação sejam transparentes para o usuário (YAU *et al.*, 2002);
- Computação ubíqua envolve a integração da computação no mundo real. Assim, computação ubíqua é, principalmente, preocupar-se com o entrelaçamento entre sistemas de informação e o mundo real (COUDERC e BANATRE, 2003).

Pode-se perceber que, embora muito tempo tenha se passado desde a proposta inicial, estas definições ainda não tornam explícito o que um software deve conter para ser considerado ubíquo.

Neste contexto, buscou-se uma definição mais formal e detalhada para computação ubíqua. Desta maneira, baseado nas indicações obtidas a partir dos resultados da revisão sistemática, adotaremos a seguinte definição:

- **Computação ubíqua:** se faz presente no momento em que os serviços ou facilidades computacionais são disponibilizados às pessoas de forma que o computador não seja uma ferramenta visível ou imprescindível para acesso a esses serviços. Ou seja, esses serviços ou facilidades podem se

materializar em qualquer momento ou lugar, de forma transparente, através do uso de dispositivos de uso comum no dia-a-dia. Neste contexto, os sistemas que compõem o ambiente podem contemplar, total ou parcialmente, as seguintes características: onipresença dos serviços, invisibilidade, sensibilidade ao contexto, comportamento adaptável ou dinamismo de tarefas, captura de experiências, descoberta de serviços, composição de funcionalidades, interoperabilidade espontânea, heterogeneidade de dispositivos e tolerância a falhas (SPÍNOLA *et al.*, 2006).

Pode-se notar que a definição de computação ubíqua representa a "filosofia" deste novo paradigma da computação. Como tal, ela define as condições ideais nas quais teremos um ambiente onde os recursos computacionais possam ser acessados de forma ubíqua. Já a definição de sistema ubíquo possui um escopo mais bem definido e está relacionado com as diferentes características que compõem a computação ubíqua. Isto porque ubiquidade é uma propriedade do sistema e como tal, pode ser atendida em sua plenitude ou parcialmente. Esta variação está relacionada com o fato de um sistema implementar ou não as funcionalidades associadas às características da computação ubíqua (SPÍNOLA *et al*., 2006), que serão descritas a seguir.

2.2.3 Características da Computação Ubíqua

Do ponto de vista das características da computação ubíqua, ao se efetuar a análise dos diferentes artigos retornados a partir das buscas efetuadas na revisão sistemática, pôde-se perceber que eles apontam na direção de um conjunto inicial de características que uma aplicação ubíqua pode conter. Na sequência temos a definição das características junto com a descrição de um cenário que exemplifica sua aplicação em projetos de software (SPÍNOLA *et al.*, 2007a):

- **Onipresença dos serviços:** permitir a movimentação física do usuário, dando a ele a percepção de estar levando consigo os serviços computacionais;
    - **Cenário:** o usuário, ao se deslocar, continua acessando seus serviços nos diferentes ambientes em que se encontra.
- **Invisibilidade:** capacidade de estar presente nos objetos de uso do dia-a-dia, descaracterizando, do ponto de vista do usuário, a "utilização" de um computador e acentuando a percepção de objetos ou dispositivos que provêem serviços ou algum tipo de "inteligência". Com isto, procura-se

permitir alternativas apropriadas para as interfaces gráficas tradicionais utilizadas nas soluções desktop, de forma a privilegiar formas mais naturais de entrada de dados tais como reconhecimento de escrita, fala, gestos, expressões faciais, ou movimentos ou ainda integrar essas formas de tal modo que a interface seja minimamente percebida pelo usuário;

    - **Cenário:** ao passar em frente a um cartaz de um filme, o indivíduo recebe no celular maiores informações sobre o filme em questão como trilha sonora, diretor, duração, elenco e o resumo.

- **Sensibilidade ao Contexto:** capacidade de coletar informações sobre o ambiente onde está sendo utilizado;
    - **Cenário:** um sistema para controle de temperatura de um frigorífico deve estar constantemente monitorando a temperatura para manter o ambiente no estado ideal para manutenção dos produtos.
- **Comportamento Adaptável:** capacidade de, dinamicamente, adaptar os serviços disponíveis ao ambiente onde está sendo utilizado dentro de suas limitações;
    - **Cenário:** imaginemos um software para gerência de vídeo conferência. Ao identificar a redução de largura de banda a ponto de prejudicar a transmissão do áudio e vídeo na qualidade atual, o software deve reduzir a qualidade do áudio e do vídeo de forma a permitir que a comunicação entre os participantes continue fluindo normalmente sem interrupções.
- **Captura de Experiências:** capacidade de capturar e registrar experiências para uso posterior;
    - **Cenário:** imaginemos um software para gerenciamento de uma casa. Ele pode identificar comportamentos comuns do morador, por exemplo: o morador ao chegar em casa liga a lâmpada da sala, esquenta a água para o café, liga a banheira e define a temperatura da água para 28º C. Ao perceber que este comportamento se repete (uma experiência), o software poderá gerenciar estas atividades quando o morador chegar em casa sem a necessidade deste desempenhar diretamente estas atividades.
- **Descoberta de Serviços:** construir serviços proativamente de acordo com o ambiente em que se encontra. A aplicação deve interagir com o ambiente e permitir que o usuário também o faça a fim de descobrir novos serviços ou informações para atingir o objetivo desejado;

    - **Cenário:** ao entrar no supermercado, o software pode identificar serviços disponibilizados pelo supermercado para apoiar a compra dos produtos pelo usuário; por exemplo: um mapa de promoções e localização de produtos.
- **Composição de Funcionalidades:** capacidade de, a partir de serviços básicos, montar uma determinada funcionalidade requerida pelo usuário;
    - **Cenário:** imagine uma situação onde o usuário esteja precisando visualizar uma planilha e gerar um arquivo PDF com o resultado da análise e estes serviços não estejam disponíveis em sua estação de trabalho, o software pode identificar os serviços necessários e torná-los disponíveis para o usuário.
- **Interoperabilidade Espontânea:** capacidade de interagir com outros dispositivos durante a sua operação conforme a sua movimentação;
    - **Cenário:** imagine que você esteja se deslocando e o software, executando no PDA, esteja desempenhando uma tarefa intensiva em processamento. Durante o percurso, o software poderá interagir com outros dispositivos para alocação temporária de processamento.
- **Heterogeneidade de Dispositivos:** prover mobilidade da aplicação através de dispositivos heterogêneos;
    - **Cenário:** imagine um software para monitoramento do mercado de ações. Ele possui suas funcionalidades totais para acesso via desktop. Entretanto, no meio de suas atividades você tem que se deslocar para outra unidade da organização tendo disponível neste meio tempo um PDA. Nesta situação, o software deveria migrar parte de suas funcionalidades para serem acessadas pelo PDA e você continuar tendo acesso às informações necessárias.
- **Tolerância a Falhas:** capacidade de se adaptar diante de falhas no ambiente (por exemplo, disponibilidade on-line/off-line).
    - **Cenário:** sistemas ubíquos estão sujeitos a uma grande quantidade de fatores que podem provocar falha, por exemplo: sensores com problema de hardware, queda de rede, dentre outros.

2.3 – Fatores Funcionais e Restritivos Associados às Características da Computação Ubíqua

Embora fundamentais para o entendimento sobre como se comportam os projetos de software ubíquos, estas características ainda nos dizem pouco sob como cada uma delas pode estar contemplada em projetos de software. O ideal é identificar de forma mais precisa como cada característica pode estar contemplada em projetos de software.

Para isso, um protocolo de busca e análise foi adotado (SPÍNOLA *et al.*, 2007). O planejamento desta segunda revisão, assim como os resultados atingidos, estão apresentados nas próximas subseções.

2.3.1 Planejamento e Execução da Revisão Sistemática

Novamente utilizando os passos descritos em (BIOLCHINI *et al.*, 2005), temos um resumo do protocolo elaborado para esta revisão:

- Objetivo: investigar quais fatores funcionais e restritivos estão associados às características onipresença dos serviços, invisibilidade, sensibilidade ao contexto, comportamento adaptável, captura de experiências, descoberta de serviços, composição de funcionalidades, interoperabilidade espontânea, heterogeneidade de serviços e tolerância a falhas.
- Questões de pesquisa:
    - 1: Quais fatores funcionais e restritivos caracterizam a característica onipresença dos serviços?
    - 2: Quais fatores funcionais e restritivos caracterizam a característica invisibilidade?
    - 3: Quais fatores funcionais e restritivos caracterizam a característica sensibilidade ao contexto?
    - 4: Quais fatores funcionais e restritivos caracterizam a característica comportamento adaptável?
    - 5: Quais fatores funcionais e restritivos caracterizam a característica captura de experiências?
    - 6: Quais fatores funcionais e restritivos caracterizam a característica descoberta de serviços?
    - 7: Quais fatores funcionais e restritivos caracterizam a característica composição de funcionalidades?

    - 8: Quais fatores funcionais e restritivos caracterizam a característica interoperabilidade espontânea?
    - 9: Quais fatores funcionais e restritivos caracterizam a característica heterogeneidade de serviços?
    - 10: Quais fatores funcionais e restritivos caracterizam a característica tolerância a falhas?
- Fontes de busca: duas bibliotecas digitais (IEEEXplorer, ACM Portal) e anais de conferências. A quantidade de bibliotecas digitais foi reduzida nesta segunda revisão devido à grande quantidade de *strings* de busca utilizadas. Isto dificultaria a adaptação destas diferentes *strings* a cada máquina de busca e tornaria a análise dos resultados excessivamente demorada.
- Critérios de Inclusão e Exclusão de artigos.
    - Os artigos devem estar disponíveis na web;
    - Os artigos devem estar descritos em inglês;
    - Os artigos devem contemplar requisitos funcionais e não funcionais pertinentes a cada uma das características de ubiquidade para a qual a *string* de busca foi formulada.
- *Strings* de busca:
    - Para a primeira pergunta:
        - (ubiquitous computing <or> pervasive computing) <and> ((functional requirement <or> functionality <or> feature <or> characteristic) <or> (no functional requirement <or> quality requirement)) <and> (computer everywhere)
    - Para a segunda pergunta:
        - (ubiquitous computing <or> pervasive computing) <and> ((functional requirement <or> functionality <or> feature <or> characteristic) <or> (no functional requirement <or> quality requirement)) <and> (invisibility)
    - Para a terceira pergunta:
        - (ubiquitous computing <or> pervasive computing) <and> ((functional requirement <or> functionality <or> feature <or> characteristic) <or> (no functional requirement <or> quality requirement)) <and> (context awareness)
    - Para a quarta pergunta:
        - (ubiquitous computing <or> pervasive computing) <and> ((functional requirement <or> functionality <or> feature <or>

characteristic) <or> (no functional requirement <or> quality requirement)) <and> (adaptability)

- Para a quinta pergunta:
  - (ubiquitous computing <or> pervasive computing) <and> ((functional requirement <or> functionality <or> feature <or> characteristic) <or> (no functional requirement <or> quality requirement)) <and> (automated capture <or> experience capture)
- Para a sexta pergunta:
  - (ubiquitous computing <or> pervasive computing) <and> ((functional requirement <or> functionality <or> feature <or> characteristic) <or> (no functional requirement <or> quality requirement)) <and> (service discovery)
- Para a sétima pergunta:
  - (ubiquitous computing <or> pervasive computing) <and> ((functional requirement <or> functionality <or> feature <or> characteristic) <or> (no functional requirement <or> quality requirement)) <and> (service composition <or> functionality composition)
- Para a oitava pergunta:
  - (ubiquitous computing <or> pervasive computing) <and> ((functional requirement <or> functionality <or> feature <or> characteristic) <or> (no functional requirement <or> quality requirement)) <and> (service heterogeneity)
- Para a nona pergunta:
  - (ubiquitous computing <or> pervasive computing) <and> ((functional requirement <or> functionality <or> feature <or> characteristic) <or> (no functional requirement <or> quality requirement)) <and> (spontaneous interoperability)
- Para a décima pergunta:
  - (ubiquitous computing <or> pervasive computing) <and> ((functional requirement <or> functionality <or> feature <or> characteristic) <or> (no functional requirement <or> quality requirement)) <and> (fault tolerance)

Na execução deste protocolo, obteve-se um total de 599 artigos retornados. Um pesquisador deu início ao processo de filtragem (todos os artigos foram

analisados). Aplicada a filtragem considerando os critérios de inclusão e exclusão, chegou-se a uma quantidade de 59 artigos para leitura e extração dos resultados (listados no Apêndice A.2).

Na sequência os resultados foram tabulados e foram realizadas análises para definir os fatores funcionais e restritivos para cada uma das características de ubiquidade. Posteriormente, essas definições foram analisadas para eliminar redundâncias e obter o conjunto final de fatores.

2.3.2 Fatores Funcionais e Restritivos

Os fatores funcionais e restritivos identificados foram organizados por característica de ubiquidade. Além disso, como a quantidade de fatores identificados foi grande e alguns deles são complementares, foi definido, quando possível, um agrupamento dos fatores para cada característica de ubiquidade. Estas informações estão apresentadas a partir da Tabela 2.1 até a Tabela 2.10 organizadas pelas características de ubiquidade identificadas até o momento. Estes fatores refletem com maior clareza como as diferentes características de ubiquidade podem estar contempladas em projetos de software ubíquos.

Para a característica Captura de Experiência (Tabela 2.1), foi identificado um total de 10 fatores de ubiquidade organizados em três grupos de fatores: captura de informação, interpretação da informação e gerência da informação.

Tabela 2.1. Lista de fatores funcionais e restritivos identificados para a característica Captura de Experiência.

Captura de Experiência		
Código	**Descrição Original**	**Descrição no Novo Formato**
Grupo de Fator: Captura de Informação		
CE01	Captura de informações sobre a interação do usuário.	O sistema deve capturar as informações das interações do usuário.
CE02	Armazenar informações capturadas.	O sistema deve armazenar as informações das interações do usuário.
CE03	Captura automática de experiências.	O sistema deve capturar experiências do usuário.
Grupo de Fator: Interpretação da Informação		
CE04	Análise de padrões que ocorrem nas interações de usuário capturadas.	O sistema deve analisar padrões nas interações do usuário.
CE05	Criar mecanismos de análise de relacionamentos entre experiências privadas e públicas.	O sistema deve analisar os relacionamentos entre experiências.
CE06		O sistema deve considerar experiências do tipo privada.
CE07		O sistema deve considerar experiências do tipo público.

Captura de Experiência		
Código	**Descrição Original**	**Descrição no Novo Formato**
Grupo de Fator: Gerência da Informação		
CE08	Possuir um mecanismo de representação de atividades desempenhadas pelo usuário.	O sistema deve representar as atividades do usuário.
CE09	Prover uma representação das necessidades do usuário e suas preferências.	O sistema deve representar as necessidades do usuário.
CE10		O sistema deve representar as preferências do usuário.

Para a característica Comportamento Adaptável (Tabela 2.2), foi identificado um conjunto de 25 fatores de ubiquidade organizados em dois grupos de fatores: gerência de adaptações e adaptação.

Tabela 2.2. Lista de fatores funcionais e restritivos identificados para a característica Comportamento Adaptável.

Comportamento Adaptável		
Código	**Descrição Original**	**Descrição no Novo Formato**
Grupo de Fator: Gerência de Adaptações		
CA01	Tratabilidade: manter informações sobre as adaptações efetuadas para prover ao usuário informações que permitam a ele saber por que determinada decisão foi tomada pelo sistema.	O sistema deve considerar informações sobre adaptações.
CA02		O sistema deve prover informações sobre adaptações.
CA03	O software deve agir proativamente, ou seja, analisar as variáveis e a partir dela tomar decisões que antecipem uma determinada situação.	O sistema deve tomar decisões de acordo com as informações do ambiente.
CA04		O sistema deve analisar informações do ambiente.
CA05	Selecionar a alternativa correta de adaptação.	O sistema deve selecionar a alternativa de adaptação.
CA06	Prever alterações no ambiente e se adaptar ou preparar-se para adaptações antes que as alterações ocorram no ambiente.	O sistema deve prever alterações no ambiente.
CA07		O sistema deve adaptar suas funcionalidades de acordo com as informações do ambiente.
CA08	Gerenciar adaptações considerando preferências do usuário e condições do ambiente.	O sistema deve gerenciar adaptações de acordo com as preferências do usuário.
CA09		O sistema deve gerenciar adaptações de acordo com as informações do ambiente.
CA10	Calcular a configuração ótima de uma aplicação e seus requisitos de qualidade dada as necessidades do usuário e	O sistema deve configurar uma aplicação de acordo com as necessidades do usuário.

Comportamento Adaptável		
Código	**Descrição Original**	**Descrição no Novo Formato**
CA11	informações do ambiente.	O sistema deve configurar uma aplicação de acordo com as informações do ambiente.
CA12	Antes de executar um serviço, verificar: • Se o número de instâncias de serviços sendo executadas no momento é menor que o número máximo permitido; • O status da execução e localização de cada serviço; • O tempo requerido para uso do serviço em relação ao tempo disponível da próxima instância do serviço.	O sistema deve executar serviços. Para isso deve verificar: se o número de instâncias de serviços sendo executadas no momento é menor que o número máximo permitido; o status da execução e localização de cada serviço; e o tempo requerido para uso do serviço em relação ao tempo disponível da próxima instância do serviço.
CA13	Gerenciar a disponibilidade de recursos para execução dos serviços.	O sistema deve gerenciar a disponibilidade de recursos.
CA14	O gerente de adaptações ao decidir sobre adaptações a serem realizadas deve considerar o impacto que estas adaptações terão no sistema como um todo, ou seja, verificar se outras aplicações relacionadas não sofrerão um impacto relevante depois da adaptação.	O sistema deve gerenciar adaptações de acordo com o seu impacto. Nesse sentido, o sistema deve verificar se outras aplicações relacionadas não sofrerão um impacto relevante depois da adaptação.
CA15	Finalizar o uso de recursos, liberando-os para futuro uso.	O sistema deve desalocar recursos.
CA16	Decidir onde alocar os processamentos necessários.	O sistema deve alocar recursos.
Grupo de Fator: Adaptação		
CA17	Adaptar automaticamente as funcionalidades baseado no contexto atual e passado.	O sistema deve adaptar as funcionalidades de acordo com o contexto.
CA18	Adaptação automática de seção de usuário.	O sistema deve adaptar a seção do usuário.
CA19	Adaptação automática de conteúdo com base no contexto.	O sistema deve adaptar conteúdo com base no contexto.
CA20	Selecionar parte do código a ser executado com base no dispositivo e contexto em que a aplicação se encontra.	O sistema deve executar código de acordo com o dispositivo.
CA21		O sistema deve executar código de acordo com o contexto.
CA22	Interface gráfica adaptável.	O sistema deve adaptar a interface gráfica.
CA23	Minimizar alterações em resultados de operações providos aos usuários como consequência de alterações no ambiente.	O sistema deve minimizar as adaptações.
CA24	Adaptar a interface com usuário a diferentes dispositivos em tempo de projeto ou execução. A adaptação é feita com base em informações do dispositivo e preferências do usuário.	O sistema deve adaptar a interface com o usuário de acordo com o dispositivo.

Comportamento Adaptável		
Código	**Descrição Original**	**Descrição no Novo Formato**
CA25	Efetuar ajustes de conteúdo e formato nos dados gerados pelos serviços.	O sistema deve adaptar serviços.

Já para a característica Composição de Funcionalidades (Tabela 2.3), foi identificado um total de 20 fatores de ubiquidade organizados em três grupos de fatores: composição, gerência de serviços e integração de serviços.

Tabela 2.3. Lista de fatores funcionais e restritivos identificados para a característica Composição de Funcionalidade.

Composição de Funcionalidade		
Código	**Descrição Original**	**Descrição no Novo Formato**
Grupo de Fator: Composição		
CF01	Compor componentes menores para formar a aplicação.	O sistema deve compor funcionalidades a partir do uso de componentes.
CF02	Gerenciar a composição de funcionalidades.	O sistema deve gerenciar funcionalidades.
CF03	Efetuar *deploy* das funcionalidades geradas (instalação, ativação, atualização, remoção).	O sistema deve implantar funcionalidades.
CF04	Compor funcionalidades a partir de serviços disponibilizados.	O sistema deve compor funcionalidades a partir do uso de serviços.
CF05	Compor funcionalidades específicas para uma plataforma.	O sistema deve compor funcionalidades de acordo com a plataforma.
CF06	Permitir que funcionalidades sejam retiradas ou acrescentadas em tempo de execução (manutenção).	O sistema deve acrescentar funcionalidades .
CF07		O sistema deve remover funcionalidades.
CF08	Composição é executada sem interação do usuário.	O sistema deve compor automaticamente funcionalidades.
CF09	Compor funcionalidades baseadas em necessidades do usuário e informações do contexto.	O sistema deve compor funcionalidades de acordo com a necessidade do usuário.
CF10		O sistema deve compor funcionalidades de acordo com as informações de contexto.
CF11	Considerar a semântica no processo de composição de funcionalidades.	O sistema deve considerar semântica da funcionalidade.
CF12	Ao final do processo de composição, solicitar permissão de execução do novo serviço para o usuário.	O sistema deve executar funcionalidades de acordo com a autorização do usuário.
Grupo de Fator: Gerência de Serviços		

Composição de Funcionalidade		
Código	**Descrição Original**	**Descrição no Novo Formato**
CF13	Prover informações sobre os componentes utilizados em tempo de execução.	O sistema deve considerar informações dos componentes.
CF14		O sistema deve considerar serviços como um tipo de componente.
CF15	Permitir que os estados dos componentes sejam mantidos durante e após a adaptação do software.	O sistema deve considerar o estado dos componentes.
CF16	Monitorar a execução dos serviços criados para verificar se eles estão desempenhando suas atividades de acordo com suas especificações.	O sistema deve monitorar as funcionalidades.
CF17	Gerenciar componentes remotos (criar, destruir, carregar, descarregar e transferir).	O sistema deve gerenciar componentes.
Grupo de Fator: Integração de Serviços		
CF18	Lidar com incompatibilidade entre interfaces de serviços.	O sistema deve considerar a compatibilidade dos componentes.
CF19	Integrar dados trocados entre os serviços	O sistema deve integrar os dados dos componentes.
CF20	Fazer uso de mediadores no mecanismo de integração. Os mediadores devem ser móveis e acopláveis a diferentes serviços. Assim, não fazem parte da estrutura de um serviço, mas sim, utilizados por eles.	O sistema deve integrar funcionalidades a partir do uso de mecanismos de integração.

Para a característica Descoberta de Serviços (Tabela 2.4), foi identificado um total de 13 fatores de ubiquidade organizados em três grupos de fatores: descoberta de serviços, conectar-se a serviços e seleção de serviços.

Tabela 2.4. Lista de fatores funcionais e restritivos identificados para a característica Descoberta de Serviços.

Descoberta de Serviços		
Código	**Descrição Original**	**Descrição no Novo Formato**
Grupo de Fator: Descoberta de Serviços		
DS01	Procurar serviços que desempenhem a mesma funcionalidade de outro já utilizado.	O sistema deve procurar serviços de acordo com a sua funcionalidade.
DS02	Identificar serviços semelhantes projetados para dispositivos diferentes.	O sistema deve identificar serviços de acordo com a sua funcionalidade.
DS03	Lidar com ambiguidade na definição de serviços.	O sistema deve considerar ambiguidade na definição dos serviços.
DS04	Lidar com a liberação de um serviço/componente que tem sido utilizado.	O sistema deve liberar serviços.
DS05	Interagir com serviços de endereçamento de serviço.	O sistema deve interagir com serviços de endereçamento de serviços.

Descoberta de Serviços		
Código	**Descrição Original**	**Descrição no Novo Formato**
DS06	Ao descobrir serviços, apresentar como sugestões de uso para o usuário.	O sistema deve procurar serviços.
DS07		O sistema deve sugerir serviços.
DS08	Cooperar com outros dispositivos que estejam em busca de um mesmo serviço.	O sistema deve cooperar com outros dispositivos.
DS09	Serviços devem ser descobertos baseados na localização do dispositivo cliente seguindo duas abordagens: baseada em distância entre o cliente e o servidor; baseada no escopo de atuação do serviço.	O sistema deve descobrir serviços de acordo com a localização do dispositivo do usuário.
Grupo de Fator: Conectar-se a Serviços		
DS10	Conectar-se dinamicamente a serviços.	O sistema deve conectar-se a serviços.
Grupo de Fator: Seleção de Serviços		
DS11	Possuir mecanismos baseados em critérios para selecionar um serviço dentre um conjunto de serviços semelhantes disponíveis.	O sistema deve selecionar serviços de acordo com a sua funcionalidade.
DS12	Selecionar conjunto de serviços para atenderem a necessidades do usuário.	O sistema deve selecionar serviços de acordo com a necessidade do usuário.
DS13	Selecionar conjunto de serviços em resposta a alterações no ambiente.	O sistema deve selecionar serviços de acordo com alterações no ambiente.

Para a característica Heterogeneidade de dispositivos (Tabela 2.5), foi identificado um total de 10 fatores de ubiquidade organizados em três grupos de fatores: busca por dispositivos, migração da aplicação e compartilhamento de recursos.

Tabela 2.5. Lista de fatores funcionais e restritivos identificados para a característica Heterogeneidade de Dispositivos.

Heterogeneidade de Dispositivos		
Código	**Descrição Original**	**Descrição no Novo Formato**
Grupo de Fator: Busca por Dispositivos		
HD01	Buscar dispositivo compatível com um determinado componente.	O sistema deve buscar dispositivos de acordo com a sua compatibilidade.
HD02	Identificar dispositivos disponíveis no ambiente.	O sistema deve identificar dispositivos de acordo com a sua disponibilidade.
Grupo de Fator: Migração de Aplicação		
HD03	Lidar com migração de código entre dispositivos heterogêneos.	O sistema deve considerar dispositivos heterogêneos.
HD04	Possibilitar a migração de uma aplicação inteira entre dispositivos ou parte de seu código.	O sistema deve migrar aplicações.
HD05		O sistema deve migrar código de

Heterogeneidade de Dispositivos		
Código	**Descrição Original**	**Descrição no Novo Formato**
		aplicações.
HD06	Controlar a migração de aplicações através do controle de versionamento e estado.	O sistema deve migrar aplicações de acordo com a versão.
HD07		O sistema deve migrar aplicações de acordo com estado.
HD08	Prover o gerenciamento de sessões quando o usuário troca de dispositivos.	O sistema deve gerenciar a sessão do usuário de acordo com o dispositivo.
HD09	Manter o estado do conjunto de aplicações para estas poderem ser migradas sem perdas para o usuário.	O sistema deve considerar o estado das aplicações.
Grupo de Fator: Compartilhamento de Recursos		
HD10	Permitir o compartilhamento de recursos	O sistema deve compartilhar recursos de dispositivos.

Já para a característica Interoperabilidade Espontânea (Tabela 2.6), foi identificado um total de 12 fatores de ubiquidade organizados em dois grupos de fatores: gerência de informações e interoperabilidade.

Tabela 2.6. Lista de fatores funcionais e restritivos identificados para a característica Interoperabilidade Espontânea.

Interoperabilidade Espontânea		
Código	**Descrição Original**	**Descrição no Novo Formato**
Grupo de Fator: Gerência de Informações		
IE01	Manter histórico de relacionamentos estabelecidos com dispositivos.	O sistema deve considerar o histórico dos relacionamentos com dispositivos.
IE02	Manter mecanismos de seleção e limpeza de informações históricas possivelmente não mais relevantes.	O sistema deve selecionar informações de acordo com a sua relevância.
IE03		O sistema deve limpar o histórico.
Grupo de Fator: Interoperabilidade		
IE04	Interagir com aplicações existentes para uso de serviços.	O sistema deve interagir com aplicações a partir do uso de serviços.
IE05	Componentes devem interagir de forma espontânea através da disponibilização de informações que descrevam o componente e funcionalidades.	O sistema deve interagir com aplicações a partir do uso de componentes.
IE06	Possuir mecanismos para possibilitar a comunicação entre aplicações que possuam interfaces de comunicação heterogêneas.	O sistema deve comunicar com aplicações a partir do uso de interfaces de comunicação heterogêneas.
IE07	Possuir mecanismos para integração de dados.	O sistema deve integrar os dados das informações.

Interoperabilidade Espontânea		
Código	**Descrição Original**	**Descrição no Novo Formato**
IE08	Cooperar com outros dispositivos.	O sistema deve cooperar com dispositivos.
IE09	Acessar outros dispositivos desde que o acesso seja permitido pela política de segurança.	O sistema deve acessar dispositivos de acordo com a política de segurança.
IE10	Identificar características dos dispositivos descobertos (capacidade de processamento e armazenamento, por exemplo).	O sistema deve identificar características dos dispositivos.
IE11		O sistema deve descobrir dispositivos.
IE12	Conexão dinâmica: conexões com dispositivos imersos no ambiente podem ser perdidas temporariamente, é necessário possibilitar seu restabelecimento.	O sistema deve conectar com dispositivos.

Para a característica Invisibilidade (Tabela 2.7) foi identificado um total de 8 fatores de ubiquidade organizados em três grupos de fatores: gerência de entidades, interface com usuário e redução do nível de interatividade com o usuário.

Tabela 2.7. Lista de fatores funcionais e restritivos identificados para a característica Invisibilidade.

Invisibilidade		
Código	**Descrição Original**	**Descrição no Novo Formato**
Grupo de Fator: Gerência de Entidades		
IN01	As entidades que compõem o sistema devem possuir um identificador.	O sistema deve considerar a identidade de suas entidades.
IN02	O registro da existência de cada identificador deve ser mantido.	O sistema deve registrar a identidade de suas entidades.
Grupo de Fator: Interface com Usuário		
IN03	Prover mecanismos diferenciados para que o usuário entre com informações (comando de voz, aproximação do usuário).	O sistema deve considerar dispositivos do tipo de entrada e saída de informação. Por exemplo, comando de voz e aproximação do usuário.
IN04	Prover mecanismos diferenciados para apresentar alguma saída ao usuário.	O sistema deve representar informações de acordo o dispositivo.
Grupo de Fator: Redução do Nível de Interatividade com o Usuário		
IN05	Pró-atividade: baseado nas informações coletadas, tomar uma decisão sem que haja necessidade de interação com usuário.	O sistema deve tomar decisões. Esse conceito é considerado pró-atividade, ou seja, o sistema age sem a necessidade de interação com o usuário.
IN06	Minimizar necessidade de configuração de dispositivos por parte do usuário.	O sistema deve minimizar a configuração de dispositivos.
IN07	Tornar a adaptação das aplicações em	O sistema deve adaptar as

Invisibilidade		
Código	**Descrição Original**	**Descrição no Novo Formato**
	tempo de execução o menos intrusiva possível.	aplicações de acordo com o dispositivo.
IN08	Reduzir a interatividade com o usuário.	O sistema deve minimizar a interatividade do usuário.

Para a característica Onipresença de Serviços (Tabela 2.8) foi identificado um total de 8 fatores de ubiquidade organizados em três grupos de fatores: mobilidade, gerência de serviços e divulgação de serviços.

Tabela 2.8. Lista de fatores funcionais e restritivos identificados para a característica Onipresença de Serviços.

Onipresença de Serviços		
Código	**Descrição Original**	**Descrição no Novo Formato**
Grupo de Fator: Mobilidade		
OS01	Gerenciar seções do usuário.	O sistema deve gerenciar a seção do usuário.
OS02	Ao deslocar os serviços, estes devem continuar operando a partir do ponto em que seu processamento foi interrompido para a migração da funcionalidade.	O sistema deve desalocar serviços.
OS03	Suportar a mobilidade entre domínios e dentro de um mesmo domínio.	O sistema deve considerar a mobilidade do usuário.
Grupo de Fator: Gerência de Serviços		
OS04	Cada dispositivo deve conter uma estrutura apropriada para alocar serviços.	O sistema deve alocar serviços.
OS05	Cada dispositivo deve gerenciar os serviços alocados em seu container.	O sistema deve gerenciar os serviços de acordo com o container do serviço.
OS06	Organizar os serviços segundo o contexto.	O sistema deve organizar serviços de acordo com o contexto do serviço.
Grupo de Fator: Divulgação de Serviços		
OS07	Divulgar a existência do serviço para outros dispositivos/aplicações.	O sistema deve divulgar serviços.
OS08	Manter o registro de serviços divulgados em cache para aumentar o desempenho em uma nova divulgação de serviços.	O sistema deve manter em memória os últimos serviços utilizados para aumentar o desempenho.

Para a característica Sensibilidade ao Contexto (Tabela 2.9), foi identificado um total de 26 fatores de ubiquidade organizados em cinco grupos de fatores: captura de informações, controle dos sensores, gerência de informações de contexto, interpretação da informação e compartilhamento da informação.

Tabela 2.9. Lista de fatores funcionais e restritivos identificados para a característica Sensibilidade ao Contexto.

Sensibilidade ao Contexto		
Código	**Descrição Original**	**Descrição no Novo Formato**
Grupo de Fator: Captura de Informações		
SC01	Permitir identificar a identidade, localização ou atividade de um usuário quando este estiver no contexto de funcionamento do sistema.	O sistema deve identificar a identidade do usuário.
SC02		O sistema deve identificar a localização do usuário.
SC03		O sistema deve identificar a atividade do usuário.
SC04	Temporalidade: considerar a variável Tempo para a informação de contexto capturada.	O sistema deve considerar a temporalidade das informações de contexto.
SC05	Considerar informações de contexto de sistema. Aquelas relacionadas aos dispositivos, infra-estrutura do ambiente (informações de hardware utilizado - CPU, memória, tamanho da tela, energia -, largura de banda disponível e dispositivos acessíveis) e outras aplicações que estão relacionadas para compor o sistema.	O sistema deve considerar informações de contexto do sistema. Aquelas relacionadas aos dispositivos, infra-estrutura do ambiente (informações de hardware utilizado - CPU, memória, tamanho da tela, energia -, largura de banda disponível e dispositivos acessíveis) e outras aplicações que estão relacionadas para compor o sistema.
SC06	Considerar informações de contexto de infra-estrutura. Neste caso, as informações dizem respeito à validade das informações dos outros tipos de contexto considerados. A importância deste tipo de informação de contexto está na diversidade de mecanismos de captura de informações.	O sistema deve considerar informações de contexto de infra-estrutura. Neste caso, as informações dizem respeito à validade das informações dos outros tipos de contexto considerados. A importância deste tipo de informação de contexto está na diversidade de mecanismos de captura de informações.
SC07	Considerar informações de contexto físico. Neste caso, buscam-se informações que dizem respeito à interação entre dispositivos e ambiente. Por exemplo, informações de mobilidade, localização, tempo, condições de iluminação e barulho, temperatura.	O sistema deve considerar informações de contexto físico. Neste caso, busca-se informações que dizem respeito à interação entre dispositivos e ambiente. Por exemplo, informações de mobilidade, localização, tempo, condições de iluminação e barulho e temperatura.
SC08	Considerar informações sobre o usuário: perfil do usuário (preferências, objetivo, calendário do usuário, informações pessoais).	O sistema deve considerar informações de contexto do usuário. Por exemplo: perfil do usuário,preferências, objetivo, calendário do usuário e informações pessoais.
Grupo de Fator: Controle de Sensores		
SC09	Controlar os sensores (ativo/em espera).	O sistema deve controlar sensores.
SC10		O sistema deve considerar fontes de dados do tipo sensor.

Sensibilidade ao Contexto		
Código	**Descrição Original**	**Descrição no Novo Formato**
Grupo de Fator: Gerência de Informações de Contexto		
SC11	Contextualizar as informações obtidas.	O sistema deve contextualizar informações.
SC12	Armazenar as informações consideradas úteis.	O sistema deve armazenar informações de acordo com a sua utilidade.
SC13		O sistema deve avaliar a utilidade das informações de contexto.
SC14	Consolidar as informações originadas de diferentes sensores.	O sistema deve consolidar as informações de contexto de acordo com sua fonte de dados.
SC15	Tratar alto acoplamento: existe relacionamento entre informações contextuais em diferentes níveis de abstração. É preciso gerenciar estes relacionamentos.	O sistema deve gerenciar relacionamentos entre informações de contexto.
SC16	Integração: é preciso criar mecanismos que possibilitem a integração de informações contextuais originadas em diferentes dispositivos com formatos distintos de representação.	O sistema deve integrar informações de contexto.
SC17	Categorizar contexto com base em sua fonte de dados (informações de rede, dispositivo e interação do usuário).	O sistema deve categorizar contexto de acordo com sua fonte de dados de suas informações.
SC18	Possuir mecanismos de representar informações de contexto considerando duas variáveis: assunto e estado. O assunto, por sua vez, é dividido em três outras variáveis: identidade, tempo e localização.	O sistema deve representar informações de contexto.
SC19	Considerar semântica na organização e captura das informações de contexto.	O sistema deve organizar as informações de contexto de acordo com a sua semântica.
Grupo de Fator: Interpretação da Informação		
SC20	*Learning* e *Reasoning*: a partir das informações contextuais capturadas e de seus relacionamentos, derivar outras informações contextuais.	O sistema deve derivar informações de contexto.
SC21	Transformação da informação: os dados capturados podem ser transformados para gerar uma informação para o software. Deve existir um rastro indicando como a transformação foi efetuada e de qual dado uma determinada informação foi gerada.	O sistema deve transformar informações de contexto.
SC22	Contextualizar e personalizar as informações capturadas de acordo com preferências do usuário.	O sistema deve contextualizar as informações de contexto de acordo com as preferências do usuário.
SC23		O sistema deve personalizar as informações de contexto de acordo com as preferências do usuário.

Sensibilidade ao Contexto		
Código	**Descrição Original**	**Descrição no Novo Formato**
SC24	Considerar semântica na interpretação de informação de contexto.	O sistema deve interpretar as informações de contexto de acordo com a sua semântica.
SC25	Associar dados do sistema com informações do contexto.	O sistema deve considerar os dados das informações de contexto.
Grupo de Fator: Compartilhamento da Informação		
SC26	Compartilhar informações de contexto com usuários e outros dispositivos. Disponibilizar para o usuário as informações de contexto como um recurso a ser utilizado para, por exemplo, efetuar configurações no software.	O sistema deve compartilhar informações de contexto.

Por fim, para a característica Tolerância a Falhas (Tabela 2.10), foi identificado um total de 3 fatores de ubiquidade organizados em dois grupos de fatores: falhas de software e falhas de sistema.

Tabela 2.10. Lista de fatores funcionais e restritivos identificados para a característica Tolerância a Falhas.

Tolerância a Falhas		
Código	**Descrição Original**	**Descrição no Novo Formato**
Grupo de Fator: Falhas de Software		
TF01	Lidar com falha de rotinas.	O sistema deve lidar com falhas de rotinas.
TF02	Criar mecanismos para armazenar informações de estado da aplicação para as falhas consideradas mais comuns de acontecer.	O sistema deve armazenar informações de estado da aplicação para as falhas consideradas mais comuns de acontecer.
Grupo de Fator: Falhas de Sistema		
TF03	Lidar com falhas no ambiente (conexão perdida, por exemplo).	O sistema deve lidar com falhas no ambiente.

2.4 – Considerações Finais

Este capítulo apresentou os resultados de duas revisões sistemáticas da literatura relacionadas à computação ubíqua. Através da execução das revisões foi possível obter como resultados:

- Uma definição atualizada da computação ubíqua;
- A identificação de um conjunto inicial de 10 características da computação ubíqua: onipresença dos serviços, invisibilidade, sensibilidade ao contexto, comportamento adaptável, captura de experiências, descoberta de

serviços, composição de funcionalidades, interoperabilidade espontânea, heterogeneidade de dispositivos e tolerância a falhas;

- A identificação de um conjunto de fatores funcionais e restritivos associados a cada uma das características da computação ubíqua que indicam em termos de funcionalidades como as características têm sido trabalhadas.

Este conjunto de conceitos compõe o corpo de conhecimento em computação ubíqua que será evoluído e utilizado ao longo deste trabalho.

Estas definições refletem o conhecimento técnico explicitado sobre computação ubíqua disponibilizado na literatura técnica. Entretanto, sua abrangência pode ser de certa forma limitada pelo escopo definido nas fontes de busca utilizadas.

No próximo capítulo será apresentada uma abordagem para a caracterização de projetos de software ubíquo elaborada a partir do corpo de conhecimento definido. Também será descrita a aplicação desta caracterização em um conjunto de 17 projetos de software identificados na literatura.

Capítulo 3 – Caracterização de Projetos de Software Ubíquo

Neste capítulo é apresentada uma abordagem para caracterização de projetos de software ubíquo e seu uso na caracterização para avaliar a aderência de 17 projetos de software ubíquo.

3.1 - Introdução

Nos últimos anos, tem-se observado um aumento na quantidade de projetos em desenvolvimento que consideram no seu conjunto de funcionalidades requisitos relativos à percepção inicial da computação ubíqua proposta por Weiser (1991). SPINOLA *et al*. (2007) observaram que o processo de aquisição de informação para reduzir os riscos envolvidos com o desenvolvimento de projetos de software ubíquo deve começar com a identificação do impacto das características de ubiquidade sobre projetos de software. Com isso, acreditamos que o conjunto de características de ubiquidade e seus fatores apresentados no capítulo anterior podem ser utilizados para a caracterização de projetos de software ubíquo.

Neste contexto, este capítulo apresenta uma abordagem baseada em *checklist* para apoiar a caracterização de projetos de software ubíquo. Esta abordagem foi aplicada inicialmente na caracterização de 12 projetos identificados na literatura até o ano de 2006. O resultado desta caracterização inicial indicou a necessidade da realização de ajustes no corpo de conhecimento organizado no Capítulo 2. Além disso, para a escrita desta Tese, outros 5 projetos de software ubíquo foram adicionados a esta análise considerando projetos entre 2007 e 2010. Este complemento reforçou os resultados já obtidos na análise inicial. Os resultados da caracterização dos 17 projetos também serão apresentados nas próximas seções.

3.2 - Abordagem para Caracterização de Projetos de Software Ubíquo baseada em *Checklist*

Baseado nos resultados obtidos com a execução das revisões sistemáticas apresentadas no Capítulo 2, foi notado que a computação ubíqua se faz presente em sua totalidade quando suas 10 características podem ser observadas nos softwares

analisados. Assim, numa primeira análise, para ser considerado ubíquo, um projeto de software deveria contemplar os diferentes fatores de cada característica de ubiquidade.

No entanto, podemos ter projetos de software com diferentes níveis de aderência às características de ubiquidade. Estes níveis podem ser consequência do domínio da aplicação ou dos requisitos do projeto, por exemplo.

Dessa forma, considerando o conjunto de definições da computação ubíqua apresentado no Capítulo 2, foi concebido um *checklist* para caracterizar projetos de software de acordo com a sua aderência às características e fatores de ubiquidade. É importante notar que o objetivo da abordagem de caracterização não é definir se um projeto de software é mais ubíquo do que outro. Seu objetivo inicial é entender como estes conceitos têm sido aplicados na prática e identificar possíveis desafios trazidos por estas características no contexto da Engenharia de Software.

A abordagem de caracterização compreende três etapas:

(1) verificar a presença dos fatores funcionais e restritivos de cada característica;
(2) calcular o nível de aderência do projeto de software para cada característica baseado na presença / ausência de cada fator restritivo e funcional;
(3) gerar um gráfico que irá representar o nível de aderência da aplicação frente às características de ubiquidade usando os valores calculados na etapa 2.

Para apoiar estas atividades foi elaborada uma planilha para realizar o cálculo do nível de aderência de cada característica. A Figura 3.1 apresenta um fragmento da planilha que permite aos engenheiros de software capturar as seguintes informações:

- Característica: apresenta as características de computação ubíqua definidas no Capítulo 2;
- Nível de Aderência à Característica: apresenta o percentual da aderência de cada característica de ubiquidade do projeto de software considerando os valores definidos na coluna Status da planilha. O cálculo é dado pela seguinte expressão definida na planilha de caracterização:

$$\text{Nível de Aderência à Característica} = \sum \frac{\text{Fatores contemplados} \times 100}{\text{Número de Fatores}}$$

Onde:

- Fatores contemplados são aqueles cujo valor da coluna Status da planilha foi marcado com o símbolo ✔ para uma determinada característica.
- Número de fatores se refere à quantidade total de fatores identificados para uma determinada característica.

- Grupo de Fatores: apresenta os grupos de fatores identificados a partir dos resultados da segunda revisão sistemática apresentada na Seção 2.3;
- Nível de Aderência ao Grupo de Fator: apresenta o percentual da aderência do projeto de software referente a um grupo de fator de uma característica de ubiquidade considerando os valores definidos na coluna Status. O cálculo é dado pela seguinte expressão:

$$\text{Nível de Aderência ao Grupo de Fator} = \frac{\sum \text{fatores contemplados} \times 100}{\text{Número de Fatores}}$$

Onde:

- Fatores contemplados são aqueles cujo valor da coluna status foi marcado com o símbolo ✔ para um determinado grupo de fator.
- Número de fatores se refere à quantidade total de fatores identificados para um determinado grupo de fator.

- Fatores: apresenta os fatores funcionais e restritivos identificados a partir dos resultados da segunda revisão sistemática apresentada na Seção 2.3;
- Status: coluna onde é definido se o fator está presente (1 → ✔) ou ausente (0 → ✘). É preenchida pelo engenheiro de software.

Uma versão completa deste *checklist* pode ser encontrada no Apêndice B.

À medida que o engenheiro de software preenche a coluna Status, a coluna Nível de Aderência à Característica (percentual dos fatores marcados em relação ao total de fatores da característica) são calculadas de forma automática. Além disso, à medida que os níveis de aderência são definidos, um gráfico que representa o nível de aderência de um projeto de software frente às características de ubiquidade é gerado.

Característica	Nível de Aderência à Característica	Grupo de Fatores	Nível de Aderência ao Grupo de Fator	Fatores	Status
Onipresença de Serviços	70%	Mobilidade	50%	Gerenciar seções do usuário	✔
				Lidar com a mobilidade do usuário tornando o serviço utilizado pelo usuário disponível onde ele esteja	✖
				Ao deslocar os serviços, estes devem continuar operando a partir do ponto em que seu processamento foi interrompido para a migração da funcionalidade	✔
				Suportar a mobilidade entre domínios e dentro de um mesmo domínio	✖
		Gerência de Serviços	67%	Cada dispositivo deve conter um container para alocar serviços	✔
				Cada dispositivo deve gerenciar os serviços alocados em seu container	✖
				Organizar os serviços segundo o contexto	✔
		Divulgação de Serviços	100%	Divulgar a existência do serviço para outros dispositivos/aplicações	✔
				Manter o registro de serviços divulgados em cache para aumentar o desempenho em uma nova divulgação de serviços	✔
		Fator Restritivo	100%	Cada serviço deve ser genérico o suficiente para continuar operando enquanto ocorrem alterações no ambiente	✔

Figura 3.1. Fragmento do *checklist* de caracterização

3.3 - Caracterização de Projetos de Software Ubíquo

A partir de agora serão apresentados os resultados da caracterização de 17 aplicações ubíquas (sendo os 12 primeiros referentes à caracterização inicial – até a subseção 3.3.12 – e os 5 últimos referentes à caracterização complementar). Para cada aplicação analisada, será apresentada uma breve descrição de seu escopo e o gráfico resultante da aplicação do *checklist* considerando as características: (IN) invisibilidade, (OS) onipresença de serviços, (SC) sensibilidade ao contexto, (CA) comportamento adaptável, (CE) captura de experiência, (DS) descoberta de serviços, (CF) composição de funcionalidades, (IE) interoperabilidade espontânea, (HD) heterogeneidade de dispositivos e (TF) tolerância a falhas.

3.3.1 Projeto de software discutido em (KINDBERG *et al*. 2000)

Em (KINDBERG *et al*. 2000) é apresentado o projeto Cooltown. Este integra serviços web com o objetivo de aprimorar a comunicação entre as pessoas através da disponibilização de serviços de acordo com o ambiente em que o usuário se encontra e da interação com objetos identificados no ambiente. Um exemplo de uso do sistema é apresentado no contexto de um museu, onde ao interagir com determinadas obras da exposição o sistema (que executa em PDAs) recupera e disponibiliza conteúdo específico relacionado à obra em questão. Para isto, o sistema faz uso de sensores para permitir a comunicação entre o PDA e os objetos presentes no museu.

A partir das informações apresentadas no artigo, identificou-se o conjunto de funcionalidades presentes. A partir destas funcionalidades e do uso do *checklist* de caracterização foi possível identificar os fatores funcionais e restritivos associados às

características de ubiquidade, definindo desta forma o nível de aderência da aplicação analisada aos conceitos de ubiquidade. Ao final da aplicação do *checklist*, o gráfico apresentado na Figura 3.2 foi gerado. Podemos indicar que quanto maior for a área do gráfico, mais próxima está a aplicação de contemplar o conjunto de características e fatores identificados na literatura como importantes de serem considerados em software ubíquos.

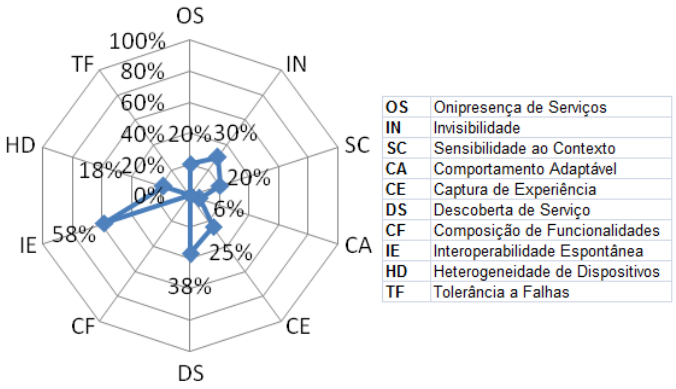


Figura 3.2. Nível de aderência do projeto apresentado em (KINDBERG *et al.* 2000).

3.3.2 Projeto de software apresentado em (JOEL *et al.*, 2004)

Joel *et al*. (2004) apresentam um projeto de software ubíquo para apoiar atividades de educação à distância. O apoio se dá através de atividades de pesquisa em campo, onde PDAs auxiliam em atividades de envio de relatório e interação com o professor orientador durante a pesquisa.

A partir das informações apresentadas no artigo, foi possível listar o conjunto de funcionalidades presentes e executar o *checklist* de caracterização. Ao final de sua aplicação, o gráfico apresentado na Figura 3.3 foi gerado.

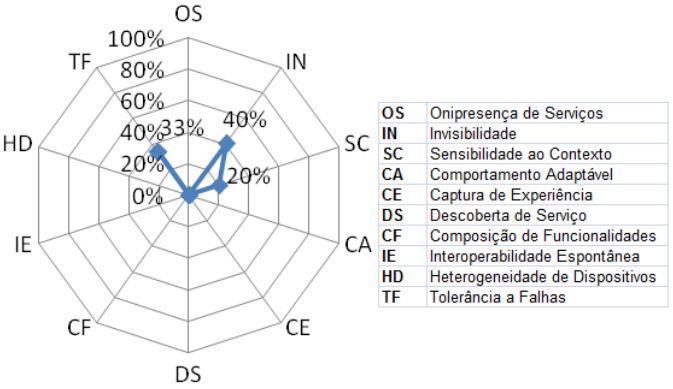


Figura 3.3. Nível de aderência do projeto apresentado em (JOEL *et al.,* 2004).

3.3.3 Projeto de software apresentado em (ALI *et al.,* 2004)

Ali *et al*. (2004) apresentam um projeto de software ubíquo com o objetivo de disponibilizar serviços no contexto de uma cozinha. Serviços como monitoramento das condições em que os alimentos se encontram e necessidade de compra de itens que estão acabando são descritos.

A partir das informações apresentadas no artigo, foi possível listar o conjunto de funcionalidades presentes. Com isto, foi possível identificar os fatores funcionais e restritivos associados às características de ubiquidade, definindo desta forma o nível de aderência da aplicação analisada aos conceitos de ubiquidade. Ao final da aplicação do *checklist* o gráfico apresentado na Figura 3.4 foi gerado.

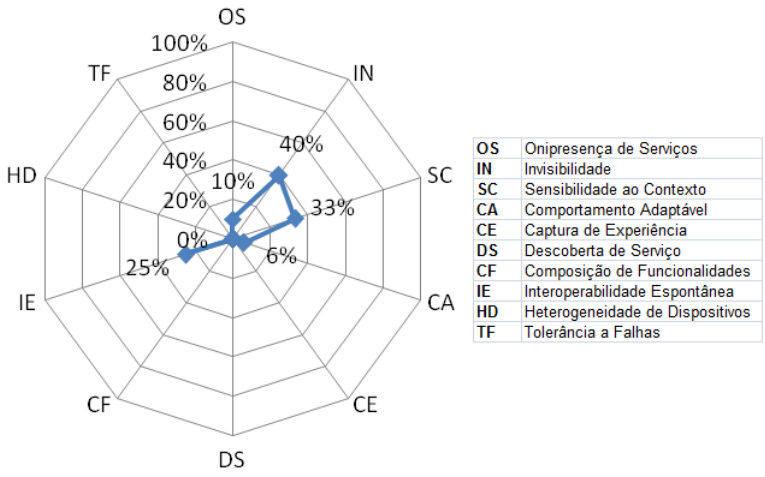


Figura 3.4. Nível de aderência do projeto apresentado em (ALI *et al.,* 2004).

3.3.4 Projeto de software descrito em (TAHTI *et al.,* 2004)

Em (TAHTI *et al.*, 2004) é apresentado um projeto de software ubíquo cujo objetivo é ter uma aplicação sensível ao contexto para apoiar a execução de atividades desempenhadas em escritórios como uso de impressoras, por exemplo. A aplicação consiste de um “assistente pessoal” composto por funcionalidades básicas e uma série de serviços disponibilizados a depender do contexto em que o usuário se encontre.

Ao final da aplicação do *checklist*, o gráfico apresentado na Figura 3.5 foi gerado.

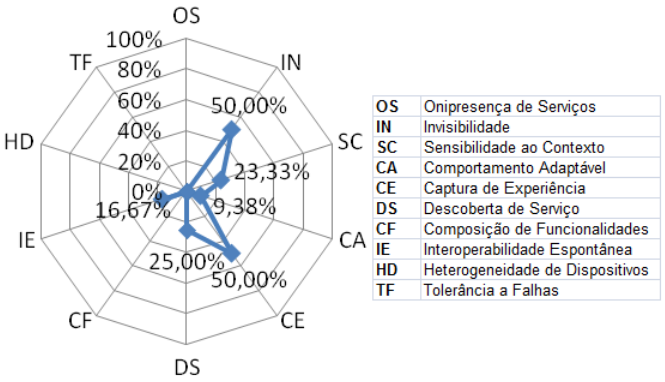


Figura 3.5. Nível de aderência do projeto apresentado em (TAHTI *et al.,* 2004).

3.3.5 Projeto de software proposto em (BOSSEN and JORGENSEN, 2004)

Bossen *and* Jorgensen (2004) apresentam um projeto de software ubíquo para apoiar atividades na área de saúde. O software construído permite o controle de dados do paciente armazenados em celulares ou PDAs, o que agiliza questões associadas a atendimento médico em clínicas e hospitais.

A partir das informações apresentadas no artigo, foi possível listar o conjunto de funcionalidades presentes e executar o *checklist* de caracterização. Ao final de sua aplicação, o gráfico apresentado na Figura 3.6 foi gerado.

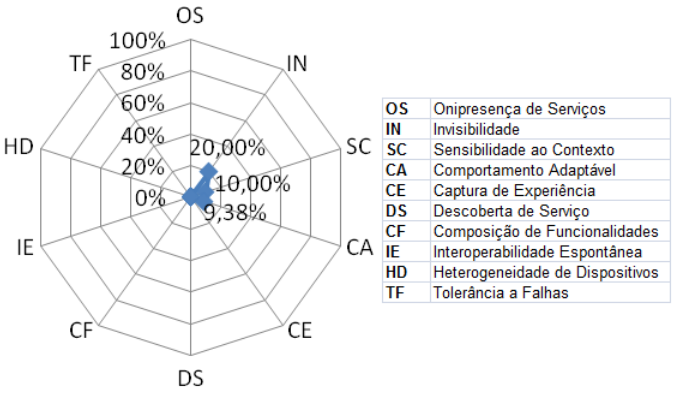


Figura 3.6. Nível de aderência do projeto apresentado em (BOSSEN and JORGENSEN, 2004).

3.3.6 Projeto de software discutido em (LEE *and* CHUNG, 2004)

Lee *and* Chung (2004) apresentam possíveis aplicações no contexto da computação ubíqua, associando o conjunto de requisitos às tecnologias necessárias para sua implementação. Estas aplicações são definidas considerando um conjunto de características de ubiquidade em comum e, desta forma, foram analisadas em conjunto.

Depois de aplicado o *checklist* às características identificadas, chegou-se ao resultado apresentado no gráfico da Figura 3.7.

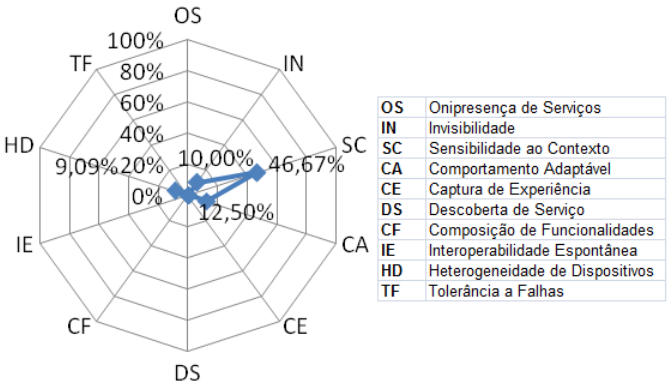


Figura 3.7. Nível de aderência do projeto descrito em (Lee and Chung, 2004).

3.3.7 Projeto de software apresentado em (HATALA *et al.,* 2005)

Hatala *et al.* (2005) apresentam um projeto de software ubíquo cujo objetivo é aprimorar a experiência de usuários que visitam museus. Neste contexto, o projeto possibilita a integração de diferentes tipos de mídia (vídeo, som e imagem) através de uma rede semântica de conhecimento fundamentada em ontologias. Este conteúdo é disponibilizado para o usuário em seu celular ou PDA quando este visita o museu, e o conteúdo (vídeo, som e imagem) é disponibilizado baseado no contexto de visitação em que o usuário se encontra em um dado momento.

Depois de aplicado o *checklist* às características identificadas para esta aplicação, chegou-se ao resultado apresentado no gráfico da Figura 3.8.

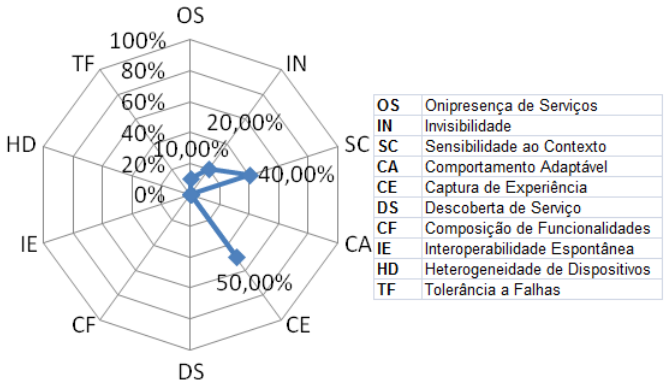


Figura 3.8. Nível de aderência da aplicação apresentada por (HATALA *et al.,* 2005)

3.3.8 Projeto de software descrito em (ZHOU *et al.* 2005)

Em (ZHOU *et al.* 2005) é apresentada uma aplicação ubíqua que permite ao usuário com celular ou PDA efetuar chamadas a taxis disponíveis em uma região próxima ao que o usuário se encontra. O sistema efetua a descoberta dos taxis disponíveis (aqueles que não estão transportando passageiros no momento) e efetua a chamada. Este projeto foi desenvolvido com intuito de apoiar os serviços de taxi em grandes centros urbanos em que localizar um taxi disponível pode não ser uma tarefa rápida de ser executada. Os autores exemplificam o uso do sistema na cidade de Nova York.

A partir das informações apresentadas no artigo, foi possível listar o conjunto de funcionalidades presentes e executar o *checklist* de caracterização de projetos de software ubíquo. Ao final da aplicação, o gráfico apresentado na Figura 3.9 foi gerado.

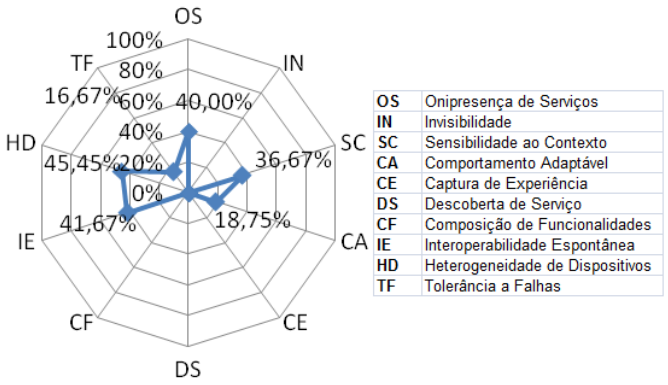


Figura 3.9. Nível de aderência do projeto apresentado em (ZHOU *et al.* 2005).

3.3.9 Projeto de software apresentado em (KIENTZ *et al.*, 2005)

Kientz *et al.* (2005) apresentam um exemplo de aplicação de captura automática de comportamentos que fornece apoio às atividades de tratamento e acompanhamento de crianças autistas.

Depois de aplicado o *checklist* às características identificadas para esta aplicação, chegou-se ao resultado apresentado no gráfico da Figura 3.10.

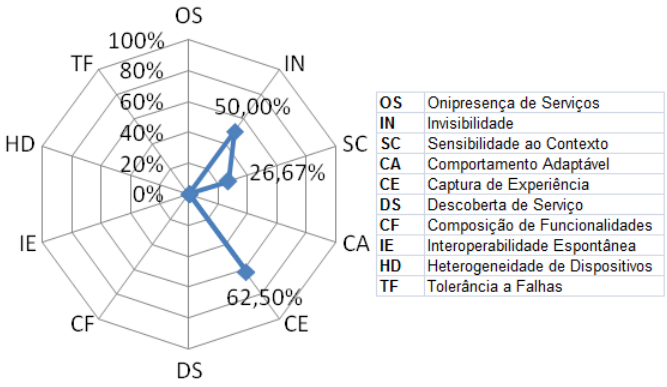


Figura 3.10. Nível de aderência do projeto apresentado em (KIENTZ *et al.,* 2005).

3.3.10 Projeto de software apresentado em (VAINIO *et al.,* 2006)

Vainio *et al.* (2006) apresentam um projeto de software ubíquo cujo objetivo é utilizar facilidades provenientes da sensibilidade ao contexto e conceitos da lógica *fuzzy* para permitir o controle de casas inteligentes de forma proativa.

Depois de aplicado o *checklist* às características identificadas para esta aplicação, chegou-se ao resultado apresentado no gráfico da Figura 3.11.

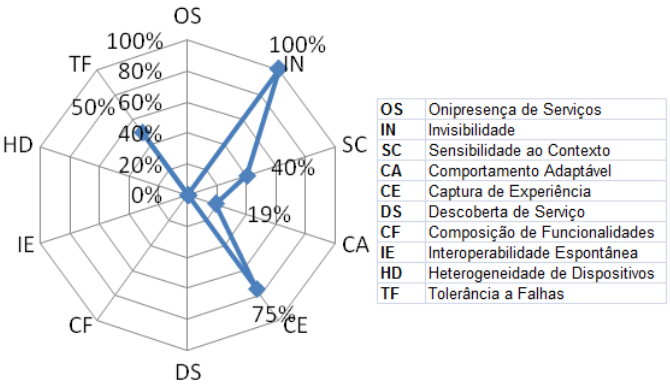


Figura 3.11. Nível de aderência do projeto descrito em (VAINIO *et al.,* 2006).

3.3.11 Projeto de software apresentado em (NAWYN *et al.,* 2006)

Nawyn *et al.* (2006) apresentam ViTo, um sistema de auxílio à modificação de comportamento em dispositivos eletrônicos embarcados. Esta mudança de comportamento se dá através de conceitos da captura de experiência onde, a partir de um conjunto de interações entre o usuário com o dispositivo, o dispositivo possui informações suficientes para personalizar seu funcionamento com base no perfil de uso.

Depois de aplicado o *checklist,* chegou-se ao resultado apresentado no gráfico da Figura 3.12.

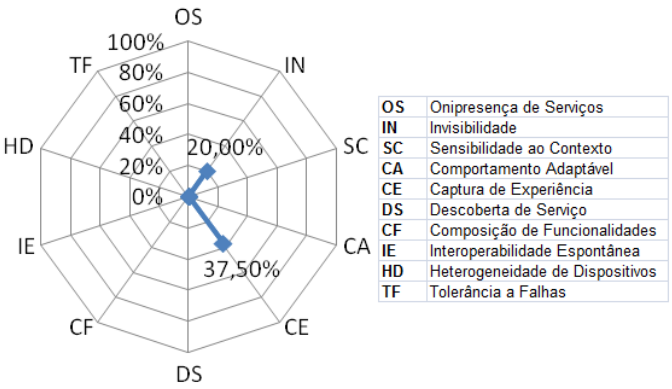


Figura 3.12. Nível de aderência do projeto apresentado em (NAWYN *et al.,* 2006).

3.3.12 Projeto de software proposto por (O'NEILLI *et al.,* 2006)

O'Neill *et al*. (2006) apresentam o projeto de uma aplicação ubíqua imersa no ambiente urbano. O sistema projetado visa apoiar estudos de comportamento humano em ambientes reais através do uso de métodos avançados de observação, captura, modelagem e análise de variáveis associadas aos indivíduos, à cidade e a elementos físicos (por exemplo, o tempo).

Depois de aplicado o *checklist* às características identificadas para esta aplicação, chegou-se ao resultado apresentado no gráfico da Figura 3.13.

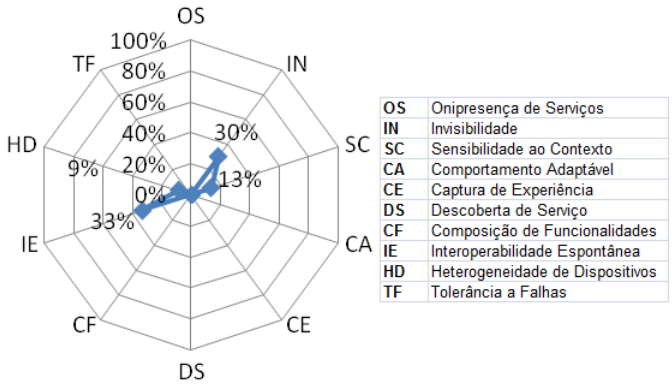


Figura 3.13. Nível de aderência do projeto descrito em (O'NEILLI *et al.,* 2006).

3.3.13 Projeto de software proposto por (SHIN *et al.,* 2007)

Shin *et al*. (2007) sugerem um sistema para cuidados com a saúde através do monitoramento ubíquo do paciente. Para isso, são utilizados mecanismos de monitoramento de dois tipos: software embarcado e sensores. Os primeiros estão

embutidos no ambiente (cama, sofá e assento). Já o segundo tipo trata-se de sensores que são colocados em uma cozinha e em todos os quartos para detectar os movimentos do paciente e suas atividades. Todas as informações coletadas pelos dispositivos ubíquos são transmitidas para uma central através de uma rede sem fio e enviadas para um laboratório do hospital após um processo de análise.

Depois de aplicado o *checklist* às características de ubiquidade identificadas para esta aplicação, chegou-se ao resultado apresentado no gráfico da Figura 3.14.

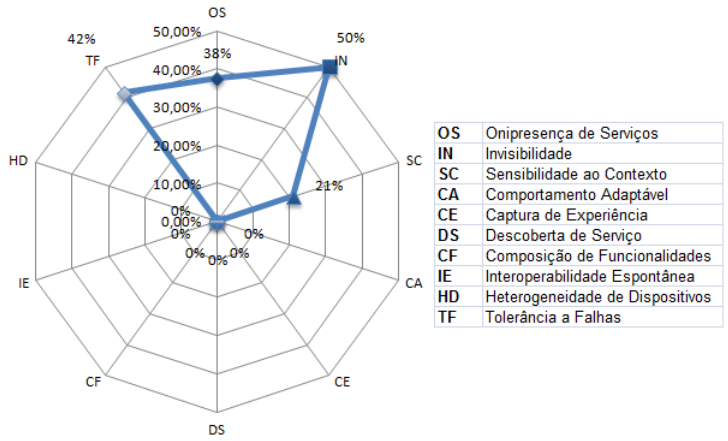


Figura 3.14. Nível de aderência do projeto descrito em (SHIN *et al.,* 2007).

3.3.14 Projeto de software apresentado em (KIENTZ *et al.,* 2007)

Kientz *et al.* (2007) apresentam um conjunto de serviços de apoio a crianças com autismo. O sistema é composto de funcionalidades de monitoramento dos pacientes ao mesmo tempo em que auxilia os cuidadores em tomadas de decisão. Este sistema trata-se de uma evolução do trabalho apresentado em (Kientz *et al.*, 2005).

Depois de aplicado o *checklist*, chegou-se ao resultado apresentado no gráfico da Figura 3.15.

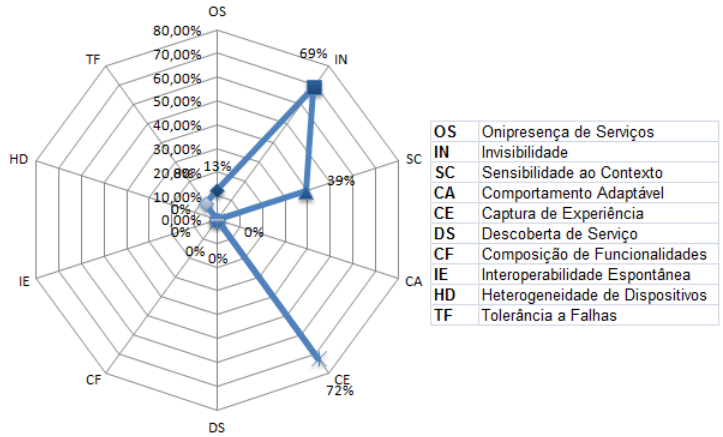


Figura 3.15. Nível de aderência do projeto apresentado em (KIENTZ *et al.,* 2007).

3.3.15 Projeto de software apresentado em (SU *et al.,* 2009)

Su *et al.* (2009) desenvolveram um sistema ubíquo de apoio a cuidados com a saúde (*u-Health*). Este faz uso de sensores, redes sem fio e sistemas embutidos para cuidar de pacientes com doenças crônicas. Com isto, o *u-Health* traz um conjunto de serviços que permite a monitoração remota de pacientes.

Depois de aplicado o *checklist*, chegou-se ao resultado apresentado no gráfico da Figura 3.16.

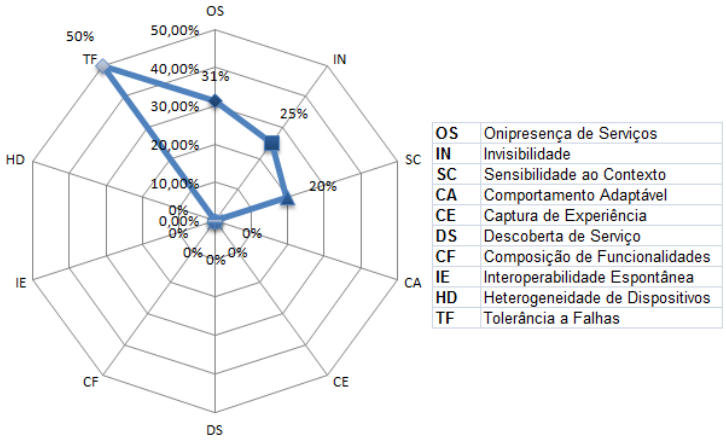


Figura 3.16. Nível de aderência do projeto apresentado em (SU *et al.,* 2009).

3.3.16 Projeto de software apresentado em (LEE e PARK, 2009)

Lee e Park (2009) apresentam o desenvolvimento de um sistema ubíquo de cuidados com a saúde através da coleta de indicadores vitais de pacientes. Para isto, o sistema é composto por três modulos: Monitoramento, Gestão e Análise. O módulo de monitoramento é responsável pela medição, transferência e recepção de dados vitais. Já o módulo de gerenciamento oferece servicos de acompanhamento para os médicos e gestores. Por fim, o módulo de análise auxilia na interpretação dos dados com o objetivo de prever doenças e calcular índices de saúde do paciente.

Depois de aplicado o *checklist* às características identificadas para esta aplicação, chegou-se ao resultado apresentada na Figura 3.17.

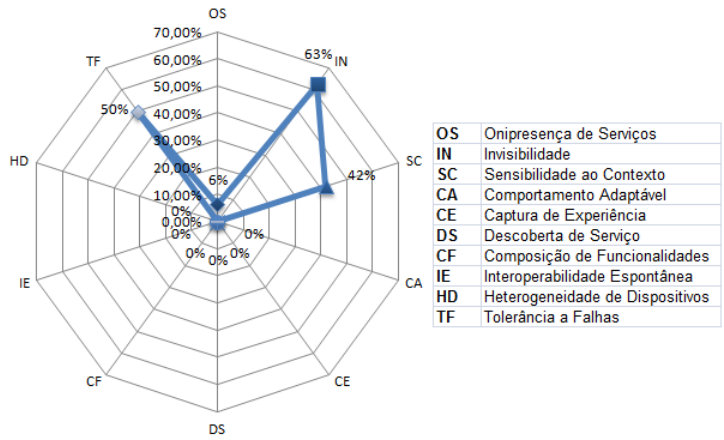


Figura 3.17. Nível de aderência do projeto descrito em (LEE e PARK, 2009).

3.3.17 Projeto de software apresentado em (YANG *et al*., 2009)

Yang *et al.* (2009) indicam inicialmente que soluções de *E-learning* estão ficando cada vez mais populares juntamente com a melhoria das redes sem fio e o avanço dos dispositivos ubíquos. Um exemplo de solução de *e-learning* é a experiência de aprendizado ao ar livre aliado ao desenvolvimento de jogos. Neste sentido, os autores do artigo propõem um jogo de auxilio a aprendizagem baseado em princípios da ubiquidade computacional. Para isso, utilizam sistemas de informação geogrático, GPS e tecnologias sem fio.

Depois de aplicado o *checklist* às características identificadas para esta aplicação, chegou-se ao resultado apresentado no gráfico da Figura 3.18.

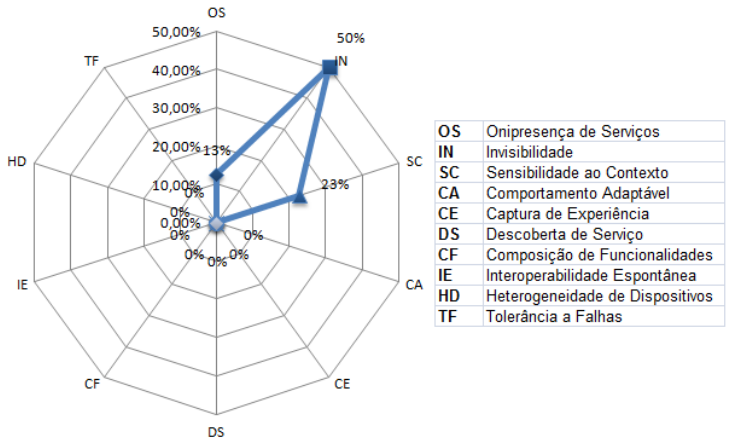


Figura 3.18. Nível de aderência do projeto descrito em (YANG *et al.,* 2009).

Aplicada a abordagem de caracterização, chegou-se a um conjunto de dados que, se analisados em conjunto, permitem inferir algumas observações interessantes. Para está análise, foi gerado o gráfico apresentado na Figura 3.19 (a ordem dos projetos indicados na figura reflete a ordem apresentada da seção 3.3.1 à 3.3.17).

Este representa o nível de aderência de cada aplicação investigada em relação às características de ubiquidade.

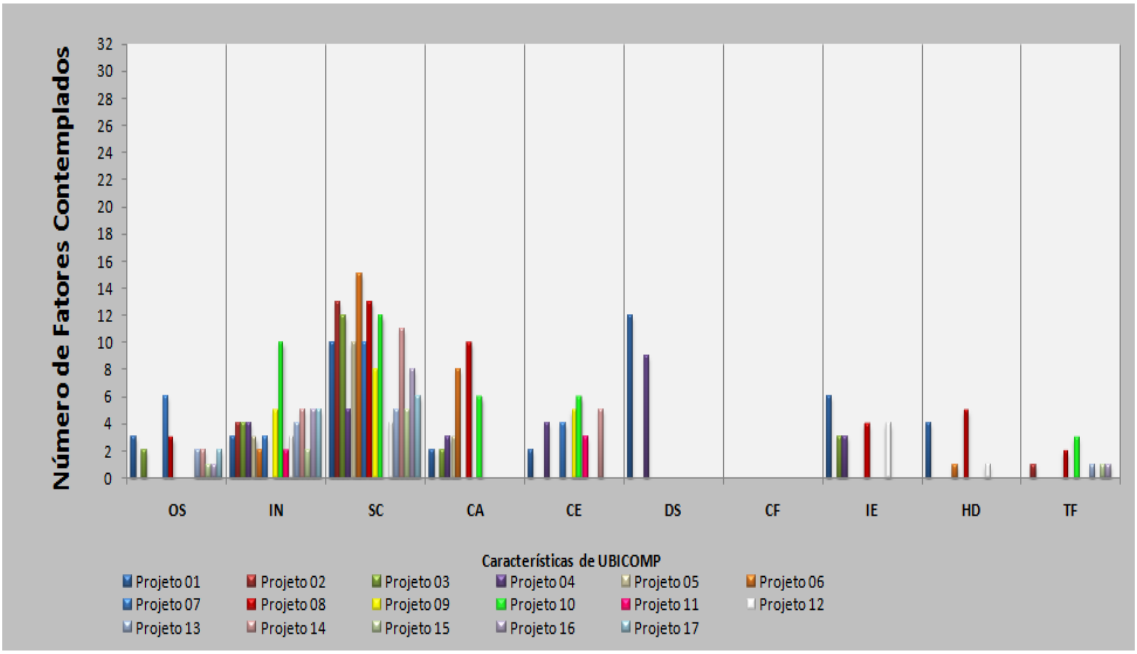


Figura 3.19 Nível de aderência dos projetos investigados em relação às características de ubiquidade.

Analisados os resultados obtidos, pôde-se observar que:

- Há uma maior concentração de esforço na implementação das características invisibilidade, sensibilidade ao contexto e comportamento adaptável.
- Destaca-se também o fato de que nenhuma das aplicações consideradas tenha contemplado fatores da característica composição de funcionalidades.
- Algumas características pouco foram consideradas: onipresença de serviços, descoberta de serviços, heterogeneidade de dispositivos e tolerância a falhas.
- Ao longo dos anos (2000 a 2009), embora conhecimento tenha sido produzido referente às diferentes características identificadas, isto ainda não pode ser identificado nos projetos de software que vem sendo desenvolvidos.

Desta forma, um comportamento observado ao finalizar a análise da aplicação do *checklist* está relacionado ao foco dado a determinadas características em projetos de software ubíquo. Enquanto invisibilidade, sensibilidade ao contexto e

comportamento adaptável têm sido constantemente consideradas nos projetos, as demais características aparecem como iniciativas isoladas, mesmo considerando projetos desenvolvidos entre 2000 e 2009, onde algumas evoluções tecnológicas ocorreram. Dois questionamentos podem ser feitos neste momento:

- **Este comportamento representa a distância natural entre o estado da arte e o estado da prática?** Ou seja, o conjunto de características de ubiquidade e seus fatores fazem sentido e a distância entre os conceitos definidos no estado da arte e aqueles aplicados no estado da prática é esperado.
- **O conjunto identificado de características de ubiquidade e seus fatores é pertinente ou possui definições que precisam ser ajustadas?** Ou seja, existem ajustes que devem ser realizados nas características de ubiquidade e seus fatores.

Para responder a estes questionamentos, optou-se nesta pesquisa pela realização de uma avaliação do corpo de conhecimento. O planejamento e a execução desta avaliação serão apresentados no próximo capítulo.

3.4 – Considerações Finais

Este capítulo apresentou uma abordagem para apoiar a caracterização de projetos de software ubíquo baseada nas características de ubiquidade e seus fatores funcionais e restritivos apresentados no Capítulo 2.

Conforme descrito ao longo deste capítulo, esta abordagem foi aplicada inicialmente na caracterização de 12 projetos identificados até o ano de 2006 e complementada com mais 5 projetos identificados entre os anos de 2007 e 2010 para a escrita desta tese. O resultado desta caracterização nos levou à necessidade de avaliação do corpo de conhecimento organizado.

Desta forma, no próximo capítulo serão apresentados o planejamento, a execução e os resultados desta avaliação.

Capítulo 4 – Características da Computação Ubíqua: Pertinência e Relevância

Neste capítulo é apresentada a avaliação do corpo de conhecimento em computação ubíqua organizado no Capítulo 2. Para isso, foram planejados e executados dois questionários envolvendo pesquisadores desta linha de pesquisa. O resultado desta avaliação é a definição da pertinência e relevância de cada uma das características.

4.1 - Introdução

Após a definição inicial da estrutura do corpo de conhecimento, foi identificada a necessidade de avaliá-lo considerando a opinião de outros especialistas em computação ubíqua. Seguindo a metodologia de pesquisa definida ao longo desta pesquisa e publicada em (DIAS-NETO, *et al.*, 2010), optou-se pela realização de um *survey* cujo planejamento e resultados obtidos serão apresentados neste capítulo.

Ao final desta avaliação inicial:

- observou-se que, embora o tamanho e a abrangência da população não pudessem ser considerados representativos considerando um cenário global em computação ubíqua, as observações reportadas durante o survey poderiam ser utilizadas para uma evolução inicial do corpo de conhecimento organizado em computação ubíqua;
- foi identificada uma propriedade adicional que deveria compor o corpo de conhecimento. Observou-se que cada característica que o compõe poderia ter um nível de relevância diferenciado no momento da caracterização de projetos de software ubíquo. No entanto, a definição dos níveis de relevância não poderia ser feita sem qualquer critério.

Motivado por estes dois fatores, a execução de um novo *survey* foi considerada e seu planejamento e resultados também serão apresentados neste capítulo.

Executados os dois *surveys,* chegou-se à configuração do corpo de conhecimento utilizado neste trabalho. Entretanto, este precisa ser constantemente

atualizado para que não se tenha um impacto negativo no seu uso. Por conta disso, um protocolo de atualização foi elaborado e está descrito ao final do capítulo junto com os resultados de sua execução.

4.2 - Avaliação Inicial do Corpo de Conhecimento

Esta avaliação inicial do corpo de conhecimento se baseou em um pacote de estudo primário com *survey* descrito em (FARIAS et al., 2003) para a área de riscos em projetos de software.

O objetivo do estudo foi **analisar** as características da computação ubíqua, seus fatores e grupos de fatores extraídos da literatura técnica (seções 2.2 e 2.3 desta tese), **com o objetivo de** caracterizar, **com respeito** a sua aplicabilidade e escopo, **no contexto de** projetos de software **sob o ponto de vista de** pesquisadores em engenharia de software que estejam trabalhando com pesquisa e desenvolvimento de projetos de software ubíquo.

As questões de pesquisa associadas a este estudo foram:

- Existe alguma característica de computação ubíqua que não está presente no conjunto inicial e que deveria ser incluída?
- Existe alguma característica de computação ubíqua presente no conjunto inicial que deveria ser excluída?
- Existe algum fator ou grupo de fatores associado a uma característica de computação ubíqua que não está presente no conjunto inicial e que deveria ser incluído?
- Existe algum fator ou grupo de fator associado a uma característica de computação ubíqua presente no conjunto inicial que deveria ser excluído?
- As características de computação ubíqua e seus fatores e grupos de fatores são úteis para caracterização de projetos de software ubíquo?

Alinhado a este objetivo, foram definidas as seguintes variáveis para este estudo:

Variáveis relacionadas às características da computação ubíqua

C_{CI} = Conjunto inicial de características da computação ubíqua.

C_{IN} = Características a serem incluídas em C_{CI}.

C_{EX} = Características a serem excluídas de C_{CI}.

C_F = Conjunto final de características da computação ubíqua.

Variáveis relacionadas aos grupos de fatores de características da computação ubíqua

GF_{CI} = Conjunto inicial de grupos de fatores das características da computação ubíqua.

GF_{IN} = Grupos de fatores a serem incluídos em GF_{CI}.

GF_{EX} = Grupos de fatores a serem excluídos de GF_{CI}.

GF_F = Conjunto final de grupos de fatores das características da computação ubíqua.

Variáveis relacionadas à aplicabilidade

AP = Aplicabilidade.

Foram definidas também três hipóteses nulas para este estudo que estão relacionadas, respectivamente, à (H0 1) análise da abrangência das características da computação ubíqua, (H0 2) análise da abrangência dos grupos de fatores das características, e (H0 3) aplicabilidade do *checklist* de caracterização de projetos de software. Elas estão descritas a seguir junto com suas hipóteses alternativas:

Hipótese nula 1 (H0): O conjunto inicial de características da computação ubíqua é abrangente, ou seja, não existem características a serem incluídas nem excluídas de C_{CI}.

$$H0: C_{IN} = C_{EX} = \varnothing; C_F = C_{CI}$$

Hipótese Alternativa (H1): O conjunto inicial não é abrangente e existem características da computação ubíqua a serem incluídas em C_{CI}.

$$H1: C_{IN} \neq \varnothing; C_F = C_{CI} + C_{IN}$$

Hipótese Alternativa (H2): O conjunto inicial é excedente e existem características da computação ubíqua a serem excluídas de C_{CI}.

$$H2: C_{EX} \neq \varnothing; C_F = C_{CI} - C_{EX}$$

Hipótese nula 2 (H0 2): O conjunto inicial de grupo de fatores de características da computação ubíqua é abrangente, ou seja, não há grupo de fatores a serem incluídos nem excluídos de GF_{CI}.

H0 2: $GF_{IN} = GF_{EX} = \varnothing$; $GF_{CI} = GF_F$

Hipótese Alternativa (H3): Existem grupos de fatores de características da computação ubíqua a serem incluídos em GF_{CI}.

H3: $GF_{IN} \neq \varnothing$; $GF_F = GF_{CI} + GF_{IN}$

Hipótese Alternativa (H4): Existem grupos de fatores de características da computação ubíqua a serem retirados de GF_{CI}.

H4: $GF_{EX} \neq \varnothing$; $GF_F = GF_{CI} - GF_{EX}$

Hipótese nula 3 (H0 3): O conjunto inicial de características da computação ubíqua e seus grupos de fatores não é aplicável à caracterização de projetos de software ubíquo.

H0 3: AP = Não

Hipótese Alternativa 5 (H5): Conjunto inicial de características da computação ubíqua é aplicável à caracterização de projetos de software ubíquo.

H5: AP = Sim

4.2.1 Instrumentação e População

O *survey* foi planejado e executado considerando a população de pesquisadores brasileiros. A escolha dos participantes foi baseada nos resultados de uma busca efetuada no Diretório dos Grupos de Pesquisa do CNPq[5] levando em conta grupos de pesquisa que consideram a computação ubíqua em seus trabalhos. Cerca de 60 pesquisadores foram convidados a participar do estudo via *e-mail*. Os questionários também foram enviados via e-mail e foram estruturados para serem preenchidos em três etapas:

(1) **Caracterização do participante.** Nesta etapa, os participantes foram questionados a respeito de suas informações de contato (nome e e-mail), formação acadêmica, nível de experiência em desenvolvimento de projetos de software (em anos), número de projetos de software executados por característica de ubiquidade;

[5] *O Diretório dos Grupos de Pesquisa no Brasil, projeto desenvolvido no CNPq desde 1992, constitui-se em bases de dados que contem informações sobre os grupos de pesquisa em atividade no País. O Diretório mantém uma base corrente, cujas informações são atualizadas continuamente pelos líderes de grupos, pesquisadores, estudantes e dirigentes de pesquisa das instituições participantes, e o CNPq realiza Censos bi-anuais, que são fotografias dessa base corrente.*

(2) **Identificação das características de computação ubíqua que deveriam ser incluídas, excluídas ou mantidas no conjunto inicial identificado.** Nesta etapa, o participante poderia informar novas características que não estavam presentes no conjunto inicial, mas que ele considerava importante ou, excluir qualquer uma das características presentes no conjunto inicial.

(3) **Identificação dos fatores e grupos de fatores associados às características da computação ubíqua que deveriam ser incluídos, excluídos ou mantidos no conjunto inicial identificado**. Para cada característica, o participante poderia informar novos fatores ou grupos de fatores que não estavam presentes no conjunto inicial, mas que ele considerava importante, ou excluir qualquer um deles.

4.2.2 Planejamento da Análise de Dados

Para a etapa de análise, foi definido que será considerado um peso diferente para cada participante de acordo com seu perfil e experiência. Pesquisadores com maior experiência em uma dada característica tiveram maior peso associado à análise de suas respostas para estas características. Depois desta definição, as respostas de todos os participantes foram analisadas para cada característica, grupo de fator e fator.

Os seguintes critérios foram utilizados para apoiar a análise dos resultados:

- Critérios de inclusão de característica / fator / grupo de fator:
    - Pelo menos um pesquisador com nível de experiência alto + análise do pesquisador responsável pelo *survey*;
    - Pelo menos dois pesquisadores com nível de experiência médio + análise do pesquisador responsável pelo *survey*;
- Critérios de exclusão de característica / fator / grupo de fator:
    - Pelo menos um pesquisador com nível de experiência alto + análise do pesquisador responsável pelo *survey*;
    - Pelo menos dois pesquisadores com nível de experiência médio + análise do pesquisador responsável pelo *survey*;

É importante observar que o critério “análise do pesquisador responsável pelo *survey*” é necessário, pois apenas ele possui uma visão completa sobre a opinião de cada pesquisador participante. Por exemplo, caso um pesquisador com experiência alta tenha solicitado a exclusão de uma característica e outros participantes também com experiência alta tenham julgado importante sua manutenção, o pesquisador

responsável pelo estudo deve tomar uma decisão baseada em seu conhecimento e no cenário global reportado pelos diferentes participantes.

4.2.3 Resultados

Este *survey* foi executado no ano de 2008 e ao final de sua execução 10 (cerca de 17% dos convidados) pesquisadores responderam ao questionário. A Tabela 4.1 apresenta um resumo da caracterização dos participantes envolvidos considerando sua titulação acadêmica e o nível de conhecimento para cada uma das características da computação ubíqua. Nesta tabela, as características da computação ubíqua consideradas são: **(OS)** Onipresença de Serviços, **(IN)** Invisibilidade, **(SC)** Sensibilidade ao Contexto, **(CA)** Comportamento Adaptável, **(CE)** Captura de Experiência, **(DS)** Descoberta de Serviço, **(CF)** Composição de Funcionalidade, **(IE)** Interoperabilidade Espontânea, **(HD)** Heterogeneidade de Dispositivos, **(TF)** Tolerância a Falhas.

Tabela 4.1. Caracterização dos participantes do *survey*.

Pesquisador	**Nível**		**Características da Computação Ubíqua**									
	M.Sc	D.Sc.	OS	IN	SC	CA	CE	DS	CF	IE	HD	TF
P01	X		A	M	A	M	A	M	M	B	-	B
P02	X		A	A	A	A	M	M	M	M	M	M
P03		X	B	B	M	M	B	M	M	M	-	B
P04		X	A	B	B	B	B	B	B	B	B	B
P05		X	A	M	A	A	B	M	M	B	A	-
P06		X	A	M	A	M	B	B	B	B	A	B
P07		X	M	M	A	A	A	M	B	M	M	B
P08		X	M	M	B	A	A	A	M	M	-	-
P09		X	B	B	A	A	A	-	M	M	M	M
P10		X	A	B	B	A	M	B	A	A	-	-

- **(A) Alto** = esteve envolvido em mais de dois projetos de software com esta característica ou áreas relacionadas.
- **(M) Médio** = esteve envolvido em até dois projetos de software com esta característica ou áreas relacionadas.
- **(B) Baixo** = pesquisa ou tem interesse nesta área, mas não esteve envolvido em nenhum projeto de software com esta característica ou áreas relacionadas.
- **(-) Nenhum** = não pesquisa e/ou não tem interesse em projetos de software com esta característica ou áreas relacionadas.

Ao observar os dados da Tabela 4.1, é importante notar que se alcançou um nível de cobertura abrangente frente às características de ubiquidade analisadas. Isto por que, com exceção da característica tolerância a falhas, todas as demais foram

analisadas por pelo menos 1 pesquisador com nível de conhecimento alto. Este fato é importante por que nos possibilita afirmar que, embora a quantidade de participantes que responderam ao *survey* seja baixa, a cobertura atingida pode ser considerada satisfatória e reforça a idéia da adequação dos resultados obtidos.

O resultado da aplicação do questionário nos permitiu aprimorar o conjunto inicial de características de ubiquidade e seus respectivos fatores e grupos de fatores através da:

- Inclusão de três novas características;
- Reorganização das características considerando duas perspectivas: funcional e restritiva;
- Exclusão de três fatores.

A Figura 4.1 apresenta como estava organizado o conjunto inicial de características da computação ubíqua, e mostra também sua nova estruturação obtida após a execução do *survey*.

Pode-se notar que antes da execução do *survey* 10 características de ubiquidade foram definidas. Os resultados do *survey* permitiram observar que estas características podeiam ser organizadas segundo duas perspectivas: funcional e restritiva. Esta nova categorização faz sentido uma vez que existem características que estão claramente relacionadas a propriedades não funcionais do software.

Além disso, 3 novas características relacionadas a requisitos não funcionais do software foram identificadas:

- Escalabilidade: indica a capacidade do sistema manipular uma quantidade crescente de trabalho de forma uniforme, ou estar preparado para o crescimento do mesmo;
- Qualidade de serviço: indica a capacidade do sistema em manter o funcionamento de seus serviços em níveis satisfatórios durante sua execução;
- Privacidade e confiança: indica a capacidade do sistema manter sob sigilo as operações executadas por um dado usuário e garantir que este não seja burlado no contexto do sistema.

Por último, a característica tolerância a falhas passou a fazer parte da perspectiva restritiva.

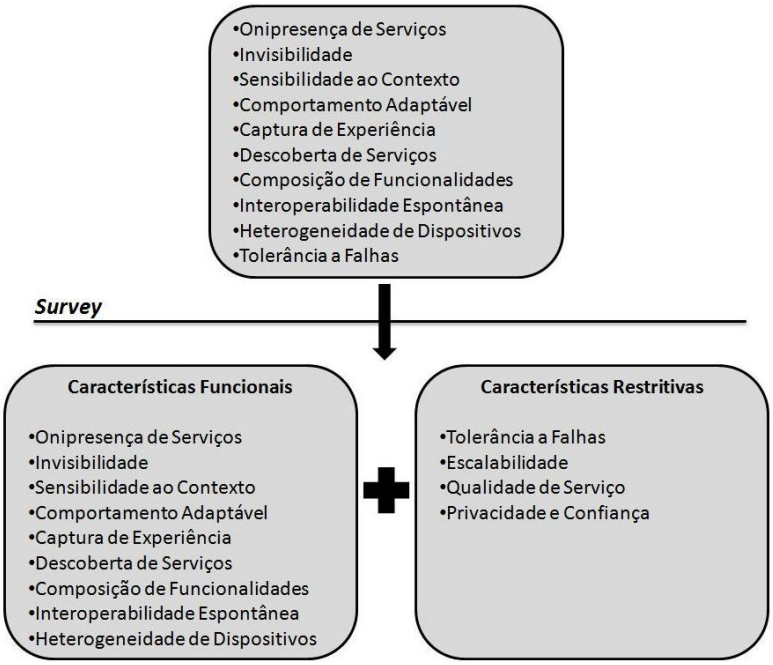


Figura 4.1. Evolução do conjunto de características de ubiquidade. Adaptado de (SPÍNOLA *et al.*, 2008)

Além destas contribuições, o *checklist* proposto (Capítulo 3) para caracterizar projetos de software ubíquo também foi avaliado. Ele foi considerado importante pelos pesquisadores para apoiar atividades no contexto da engenharia de software como engenharia de requisitos e planejamento de projeto.

Dessa forma, observou-se que as Hipóteses Nula 1, 2 e 3 (H0 1, H0 2 e H0 3) foram refutadas. Tabela 4.2 sumariza o corpo de conhecimento em computação ubíqua organizado até o momento.

Tabela 4.2. Estado atual do Corpo de Conhecimento organizado em computação ubíqua.

Característica	# Fatores	Funcional/ Restritivo
Onipresença de Serviço	10	F
Invisibilidade	10	F
Sensibilidade ao Contexto	30	F
Comportamento Adaptável	32	F
Captura de Experiência	7	F
Descoberta de Serviço	24	F
Composição de Funcionalidade	23	F
Interoperabilidade Espontânea	12	F
Heterogeneidade de Serviços	11	F
Tolerância a Falhas	6	R
Escalabilidade	*	R
Qualidade de Serviço	*	R
Privacidade e Confiança	*	R
* Não identificado.		

Embora o estudo apresentado nesta seção seja importante, nota-se que o tamanho da população pode ser considerado pequeno e não representativo considerando um cenário global em computação ubíqua. Desta forma, o resultado do *survey* não pôde ser utilizado como uma avaliação do corpo de conhecimento embora os resultados tenham sido importantes para sua evolução. Neste sentido, o planejamento e a execução de um segundo *survey* foi considerado e está apresentado na próxima seção.

4.3 - Pertinência e Relevância das Características da Computação Ubíqua

O objetivo deste estudo foi **analisar** as características da computação ubíqua extraídas da literatura técnica e aprimoradas na avaliação inicial, **com o objetivo de** caracterizar, **com respeito** à sua pertinência e relevância na caracterização de projetos de software **sob o ponto de vista de** pesquisadores em computação ubíqua **no contexto de** projetos de software ubíquo.

Para isto, foram definidas as seguintes questões de pesquisa:

- As características de ubiquidade extraídas da literatura técnica e aprimoradas na avaliação inicial são pertinentes para a caracterização de projetos de software?
- Existem características de ubiquidade descritas no corpo de conhecimento que foram incluídas indevidamente no conjunto de características de software ubíquo e que deveriam ser excluídas?
- Existem características de ubiquidade que devem ser incluídas no conjunto características de ubiquidade descritas no corpo de conhecimento?
- Qual é o nível de relevância das características de ubiquidade considerando a caracterização de projetos de software?

Neste estudo, o termo pertinência indica se cada característica é ou não importante para descrever ou definir um corpo de conhecimento em computação ubíqua considerando as características de ubiquidade. Já o termo relevância indica o quão importante uma determinada característica de ubiquidade é ao se caracterizar projetos de software, isto é, o peso da característica na caracterização de projetos de software.

Alinhado a este objetivo, foram definidas as seguintes variáveis para este estudo:

Variáveis relacionadas à pertinência das características de ubiquidade

C_{CI} = Conjunto inicial de características.

C_{IN} = Características a serem incluídas em C_{CI}.

C_{EX} = Características a serem excluídas de C_{CI}.

C_F = Conjunto final de características.

Variáveis relacionadas à relevância das características de ubiquidade

RE_i = nível de relevância da característica de ubiquidade "i" considerando o desenvolvimento de projetos de software, onde "i" se refere a um número de 1 a n (sendo n o numero total de características de ubiquidade).

Foram definidas também duas hipóteses nulas para este estudo que estão relacionadas, respectivamente, à (H0 1) análise da pertinência das características e (H0 2) suas relevâncias para a caracterização de projetos de software ubíquo. Elas estão descritas a seguir, junto com suas hipóteses alternativas:

Hipótese nula 1 (H0): O conjunto inicial de características da computação ubíqua é completo, ou seja, não existem características a serem incluídas nem excluídas de C_{CI}.

$$\textbf{H0: } C_{IN} = C_{EX} = \varnothing;\ C_F = C_{CI}$$

Hipótese Alternativa (H1): O conjunto inicial de características da computação ubíqua é incompleto, ou seja, existem características a serem incluídas em C_{CI}.

$$\textbf{H1: } C_{IN} \neq \varnothing;\ C_F = C_{CI} + C_{IN}$$

Hipótese Alternativa (H2): Existem características da computação ubíqua a serem excluídas de C_{CI}.

$$\textbf{H2: } C_{EX} \neq \varnothing;\ C_F = C_{CI} - C_{EX}$$

Hipótese nula 2 (H0 2): As características de ubiquidade possuem o mesmo nível de relevância.

$$\textbf{H0 3: } RE_1 = RE_2 = RE_3 = ... = RE_n$$

Hipótese alternativa 1 (H1): Existe pelo menos uma característica de ubiquidade que possui o nível de relevância diferente das demais.

$$\mathbf{H1:}\ \exists\ RE_i \mid RE_i \neq RE_j,\ i \neq j$$

(onde "i" e "j" são números entre 1 e n, e "i" ≠ "j")

4.3.1 Instrumentação e População

Para apoiar a execução deste *survey*, um questionário (localizado em inglês) foi desenvolvido e disponibilizado para preenchimento na *internet*. Seu preenchimento se dá em três etapas:

(1) **Caracterização do participante:** nesta etapa os participantes foram questionados a respeito de seus dados pessoais (nome e e-mail), formação acadêmica, nível de experiência em desenvolvimento de projetos de software (em anos), numero de projetos de software ubíquo executados. Um fragmento do questionário utilizado nesta etapa pode ser visto na Figura 4.2;

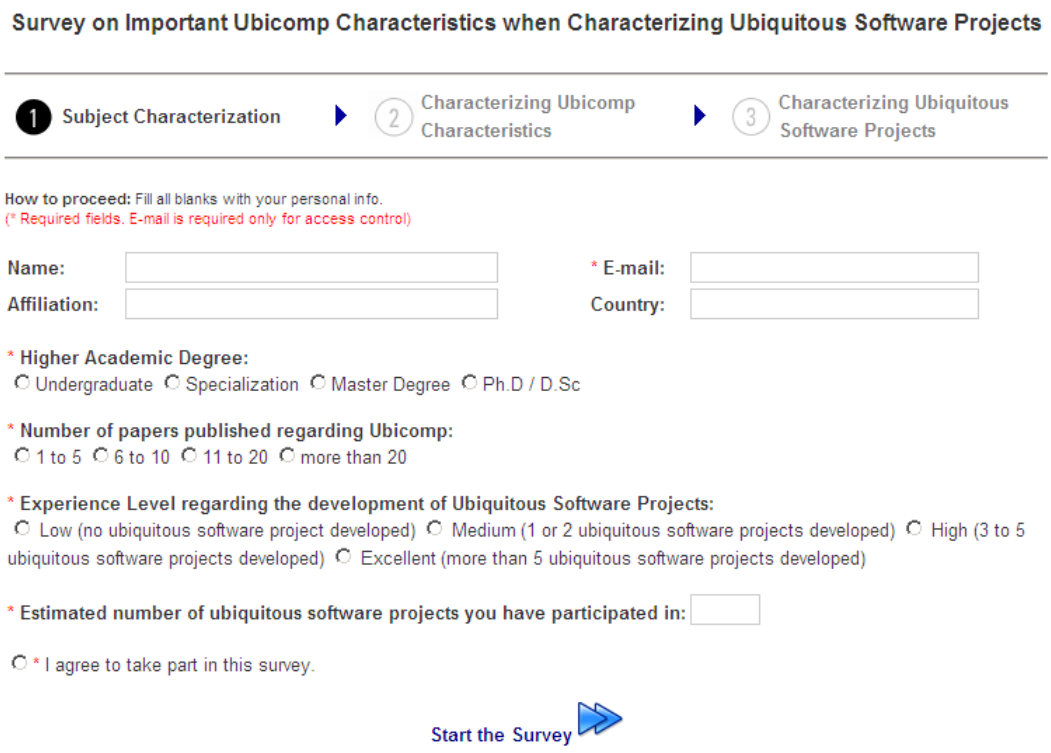
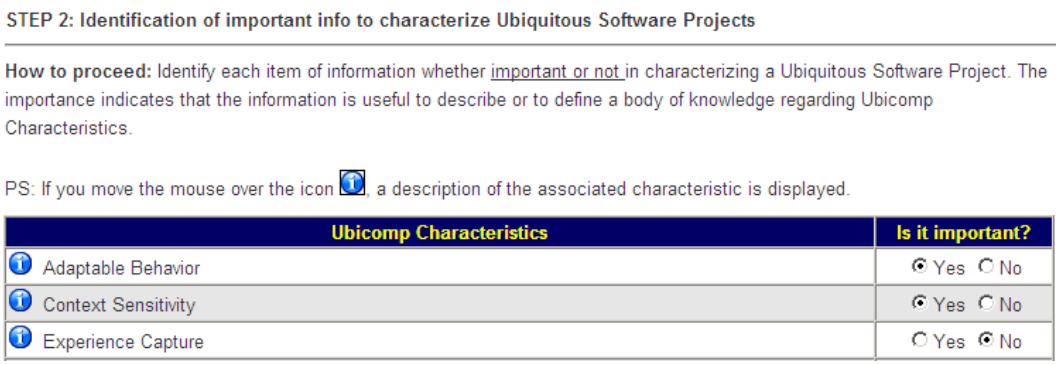


Figura 4.2. Caracterização do participante.

(2) **Identificação das características de ubiquidade pertinentes para apoiar a caracterização de projetos de software:** nesta etapa, os participantes eram solicitados a indicar quais características de ubiquidade eram ou não

pertinentes. Um fragmento do questionário utilizado nesta etapa pode ser visto na Figura 4.3;


STEP 2: Identification of important info to characterize Ubiquitous Software Projects

How to proceed: Identify each item of information whether important or not in characterizing a Ubiquitous Software Project. The importance indicates that the information is useful to describe or to define a body of knowledge regarding Ubicomp Characteristics.

PS: If you move the mouse over the icon, a description of the associated characteristic is displayed.

Ubicomp Characteristics	Is it important?
Adaptable Behavior	(•) Yes ( ) No
Context Sensitivity	(•) Yes ( ) No
Experience Capture	( ) Yes (•) No

Figura 4.3. Identificação da pertinência das características.

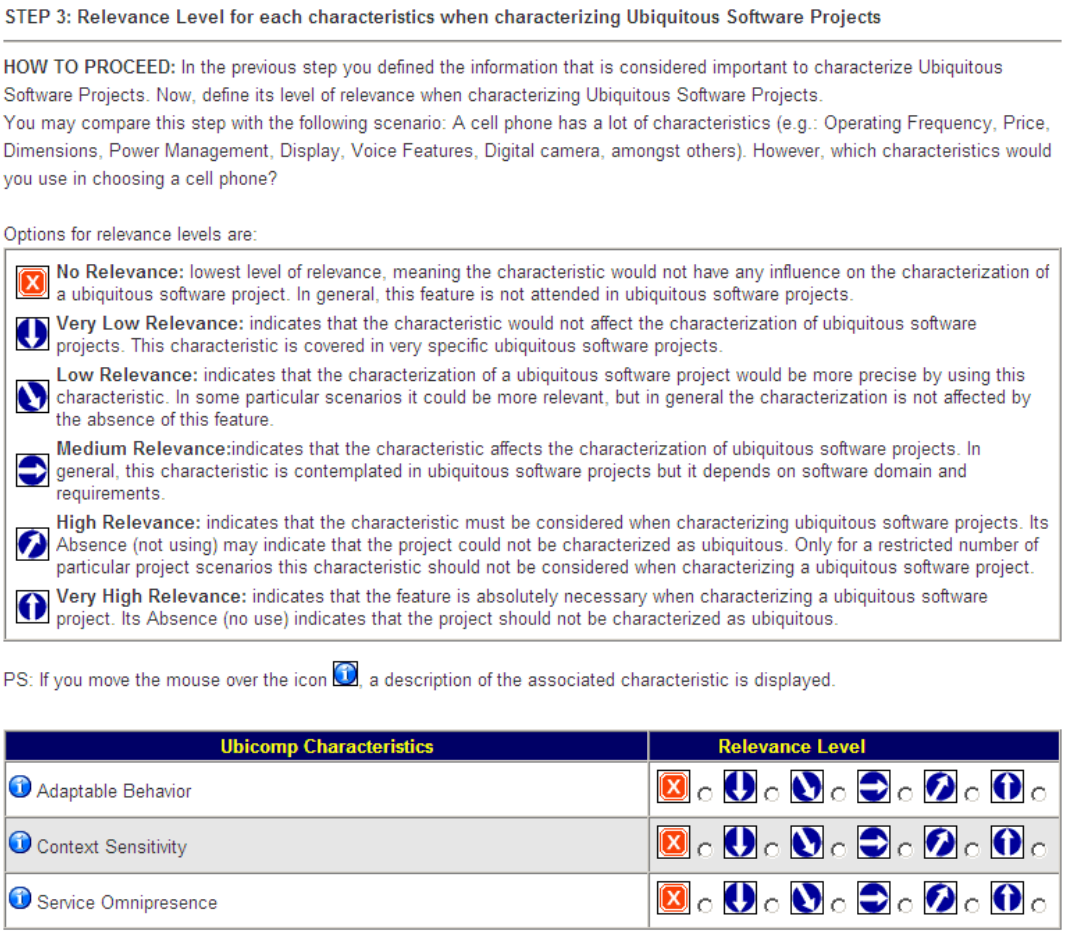
(3) **Definição do nível de relevância das características de ubiquidade para apoiar a caracterização de projetos de software ubíquo:** para esta etapa, foram definidos seis níveis de relevância (*Likert Scale*):

- **Irrelevante ( ):** é o nível mais baixo de relevância e indica que a característica da computação ubíqua não influencia na caracterização de projetos de software.
- **Relevância Muito Baixa ( ):** indica que a característica de ubiquidade pode não afetar a caracterização de projetos de software ubíquos. A característica está presente em projetos muito específicos.
- **Relevância Baixa ( ):** indica que a caracterização de projetos de software pode ser mais precisa utilizando esta característica da computação ubíqua. Em alguns cenários específicos a característica da computação ubíqua pode ser mais relevante, mas em geral a caracterização é pouco afetada pela sua ausência embora seja sempre afetada.
- **Relevância Média ( ):** indica que a característica da computação ubíqua afeta a caracterização de projetos de software ubíquo. Em geral, ela é contemplada em projetos desta natureza, mas isso depende do domínio e dos requisitos da aplicação.
- **Relevância Alta ( )**: indica que a característica da computação ubíqua deve ser considerada na caracterização de projetos de software ubíquos. Sua ausência pode indicar que o projeto pode não

ser caracterizado como sendo ubíquo. Apenas para um número restrito de situações esta característica pode não ser considerada ao se caracterizar projetos de software como sendo ubíquo.

- **Relevância Muito Alta ( ):** indica que a característica da computação ubíqua é absolutamente necessária ao se caracterizar projetos de software ubíquos. Sua ausência indica que o projeto não pode ser caracterizado como sendo ubíquo.

Um fragmento do questionário utilizado nesta etapa pode ser visto na Figura 4.4;



STEP 3: Relevance Level for each characteristics when characterizing Ubiquitous Software Projects

HOW TO PROCEED: In the previous step you defined the information that is considered important to characterize Ubiquitous Software Projects. Now, define its level of relevance when characterizing Ubiquitous Software Projects.
You may compare this step with the following scenario: A cell phone has a lot of characteristics (e.g.: Operating Frequency, Price, Dimensions, Power Management, Display, Voice Features, Digital camera, amongst others). However, which characteristics would you use in choosing a cell phone?

Options for relevance levels are:

No Relevance: lowest level of relevance, meaning the characteristic would not have any influence on the characterization of a ubiquitous software project. In general, this feature is not attended in ubiquitous software projects.

Very Low Relevance: indicates that the characteristic would not affect the characterization of ubiquitous software projects. This characteristic is covered in very specific ubiquitous software projects.

Low Relevance: indicates that the characterization of a ubiquitous software project would be more precise by using this characteristic. In some particular scenarios it could be more relevant, but in general the characterization is not affected by the absence of this feature.

Medium Relevance:indicates that the characteristic affects the characterization of ubiquitous software projects. In general, this characteristic is contemplated in ubiquitous software projects but it depends on software domain and requirements.

High Relevance: indicates that the characteristic must be considered when characterizing ubiquitous software projects. Its Absence (not using) may indicate that the project could not be characterized as ubiquitous. Only for a restricted number of particular project scenarios this characteristic should not be considered when characterizing a ubiquitous software project.

Very High Relevance: indicates that the feature is absolutely necessary when characterizing a ubiquitous software project. Its Absence (no use) indicates that the project should not be characterized as ubiquitous.

PS: If you move the mouse over the icon , a description of the associated characteristic is displayed.

Ubicomp Characteristics	Relevance Level
Adaptable Behavior	
Context Sensitivity	
Service Omnipresence	

Figura 4.4. Identificação da relevância das características.

A população deste estudo foi formada por autores que tiveram artigos: (i) identificados pelas duas revisões sistemáticas apresentadas no Capítulo 2, ou; (ii) publicados nos anais da UBICOMP – uma das mais importantes conferências da área. Os autores foram contactados por e-mail por onde também receberam acesso ao questionário na *internet*.

4.3.2 Planejamento da Análise de Dados

Para a etapa de análise dos dados foi necessário definir como as variáveis peso dos participantes, pertinência da característica e nível de relevância da caracteristica seriam calculadas (DIAS-NETO *et al*. 2007).

4.3.2.1 Peso dos Participantes

Para isto, serão consideradas quatro perspectivas para atribuição de pesos aos participantes do *survey*: formação acadêmica, quantidade de artigos publicados em computação ubíqua, experiência em participação de projetos de software ubíquo, quantidade de projetos de software ubíquo cujo participante esteve envolvido. A fórmula utilizada para atribuição de pesos aos participantes foi:

$$Weight(i) = f(i) + p(i) + e(i) + \frac{t(i)}{MedianTP}$$, onde:

- $Weight(i)$ é o peso atribuído ao participante *i*;
- $f(i)$ é a formação acadêmica. As opções para esta variável são:
  - $f(i)$ = 0, se o participante possui graduação;
  - $f(i)$ = 1, se o participante possui especialização;
  - $f(i)$ = 2, se o participante possui mestrado;
  - $f(i)$ = 3, se o participante possui doutorado;
- $p(i)$ indica a quantidade de artigos sobre computação ubíqua publicados pelo participante. As opções para esta variável são:
  - $p(i)$ = 0, se o número de artigos está entre 1 e 5;
  - $p(i)$ = 1, se o número de artigos está entre 6 e 10;
  - $p(i)$ = 2, se o número de artigos está entre 11 e 20;
  - $p(i)$ = 3, se o número de artigos é maior do que 20;
- $e(i)$ indica o nível de experiência do participante em projetos de software ubíquo. As opções para esta variável são:
  - $e(i)$ = 0, se o nível de experiência é baixo (não participou de nenhum projeto);
  - $e(i)$ = 1, se o nível de experiência é médio (participou de 1 ou 2 projetos);

- $e(i)$ = 2, se o nível de experiência é alto (participou de 3 a 5 projetos);
- $e(i)$ = 3, se o nível de experiência é excelente (participou acima de 5 projetos);

- $t(i)$ é o número total (estimado) de projetos de software ubíquo em que o participante esteve envolvido.
- $MedianTP$ é a mediana do total de projetos de software ubíquo considerando a resposta de todos os participantes.

4.3.2.2 Pertinência das Características

Para definir quais características de ubiquidade são pertinentes para caracterizar projetos de software como sendo ubíquo, é necessário primeiro somar as respostas de cada participante considerando seus respectivos pesos:

$Pertinence(j) = \sum_{i=1}^{M}(Answer(i,j) * Weight(i))$, onde:

- $Pertinence(j)$ é o valor total das respostas de todos os participantes (considerando seus respectivos pesos) sobre a pertinência da Característica de Ubiquidade *j* para caracterizar projetos de software ubíquo.
- $Answer(i,j)$ indica se a Característica *j* foi definida pelo participante *i* como pertinente (1) ou não pertinente (0).
- $Weight(i)$ é o peso atribuído ao participante *i*;
- *M* é o número total de participantes do *survey*.

A definição se uma característica da computação ubíqua é ou não pertinente para caracterizar projetos de software ubíquo é baseada em um ponto de corte (*threshold*) que indica se uma determinada característica é incluída (valor maior do que o *threshold*) ou não (valor menor do que o *threshold*) no conjunto final de características de ubiquidade. O valor definido como *threshold* foi 50% do valor máximo que uma característica *j* pode obter na variável $Pertinence(j)$ se todos os participantes a classificarem como pertinente.

$Threshold = 0{,}5 * \sum_{i=1}^{M} Weight(i)$, onde:

- $Weight(i)$ é o peso atribuído ao participante *i*;
- *M* é o número total de participantes do *survey*.

Dessa forma, os critérios são:

- se $Pertinence(j) \leq Threshold$ → característica *j* é classificada como "não pertinente" e deve ser removida do conjunto;
- se $Pertinence(j) > Threshold$ → característica *j* é classificada como "pertinente" e deve ser mantida no conjunto.

4.3.2.3 Relevância das Características

Por fim, para definir o nível de relevância de cada característica de ubiquidade classificada anteriormente como pertinente, é necessário primeiro somar a resposta de cada participante (multiplicada pelo seu respectivo peso).

$RLevel(j) = \sum_{i=1}^{N} (Scale(i, j) * Weight(i))$, onde:

- $RLevel(j)$ é o valor total das respostas de todos os participantes (multiplicadas pelos seus respectivos pesos) para a Característica *j*;
- $Scale(i, j)$ é a escala de nível de relevância (0-6) definida pelo participante *i* para a Característica *j*;
- *N* é o número total de participantes do *survey*.

Depois desta etapa, as características da computação ubíqua serão ordenadas do nível de relevância mais alto para o mais baixo. As características mais relevantes serão aquelas que possuírem maior valor para $RLevel(j)$.

4.3.3 Resultados

Este *survey* foi executado considerando um população de 280 participantes no ano de 2009. Deste total, 31 pesquisadores de diferentes regiões (América do Norte, Asia e Europa) responderam o questionário (cerca de 11%): 22 deles eram Ph.D., 7 Mestres e 2 graduados. Em média, os participantes já haviam participado de 7 projetos de software ubíquo.

Como resultado da análise dos dados, foi possível avaliar o corpo de conhecimento em computação ubíqua organizado através da identificação da pertiência e do nível de relevância de cada característica de ubiquidade.

4.3.3.1 Análise da Pertinência das Características da Computação Ubíqua

Aplicada a fórmula definida para avaliar a importância ou não de uma característica de ubiquidade apresentada na seção anterior, foram obtidos os resultados apresentados no gráfico da Figura 4.5.

O limite inferior para uma característica ser considerada pertinente é 46,74%. Esse critério foi adotado por ser o ponto médio na escala de nível de pertinência (que varia de 0% a 93,47%), seguindo a fórmula para cálculo do nível de pertinência adotada. Dessa forma, as características sensibilidade ao contexto, comportamento adaptável, onipresença de serviço, heterogeneidade de dispositivos e captura de experiência, interoperabilidade espontânea, escalabilidade, privacidade e confiança, tolerância a falhas e qualidade de serviço foram consideradas pertinentes enquanto que as características descoberta de serviços, invisibilidade e composição de funcionalidades foram descartadas.

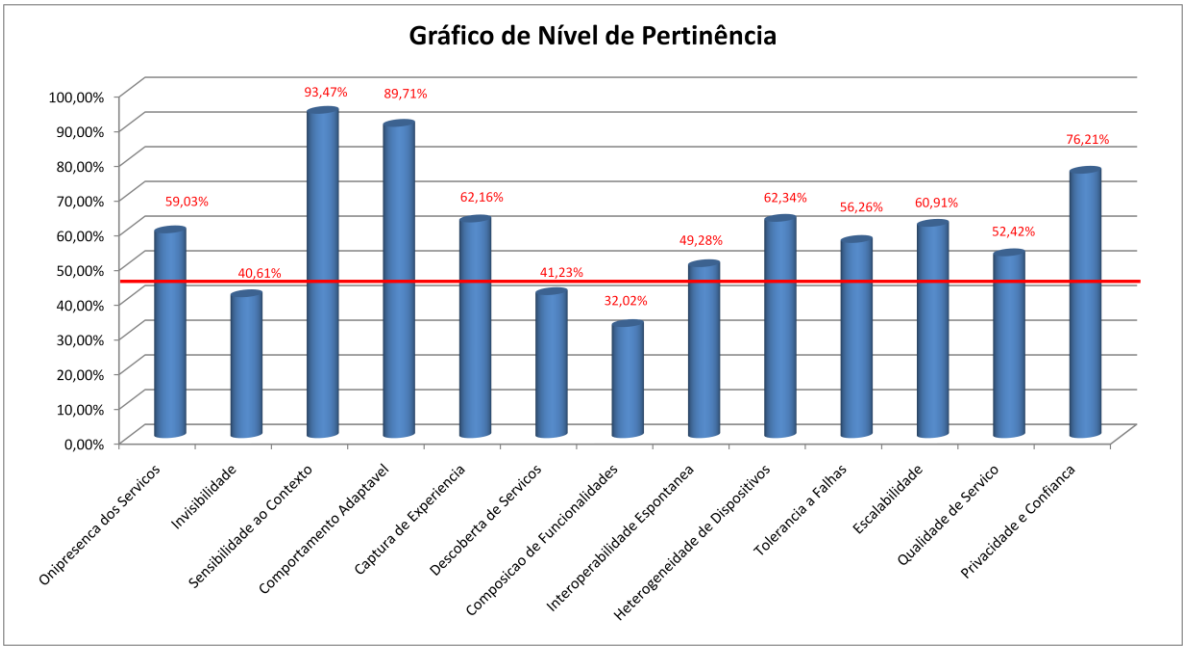


Figura 4.5. Pertinência das características.

Além disso, três pesquisadores indicaram a necessidade de inclusão de uma nova característica: usabilidade universal. Esta está associada ao fato de que o projeto de usabilidade deste tipo de software deve considerar bons padrões de usabilidade ao mesmo tempo em que considera diferentes perfis de usuários para o sistema.

Com isso, observa-se que a Hipótese Nula 1 (H0 1) foi refutada, pois existia uma característica da computação ubíqua sugerida pelos participantes que foi adicionada ao conjunto final inicial e três características que foram excluídas.

4.3.3.2 Análise da Relevância das Características da Computação Ubíqua

Após identificar as características da computação ubíqua que são consideradas pertinentes pelos participantes do estudo, o passo seguinte é a definição dos seus níveis de relevância para a caracterização de projetos de software ubíquo. Aplicada a fórmula para cálculo do nível de relevância apresentada na anteriormente, foram obtidos os resultados representados no gráfico da Figura 4.6.

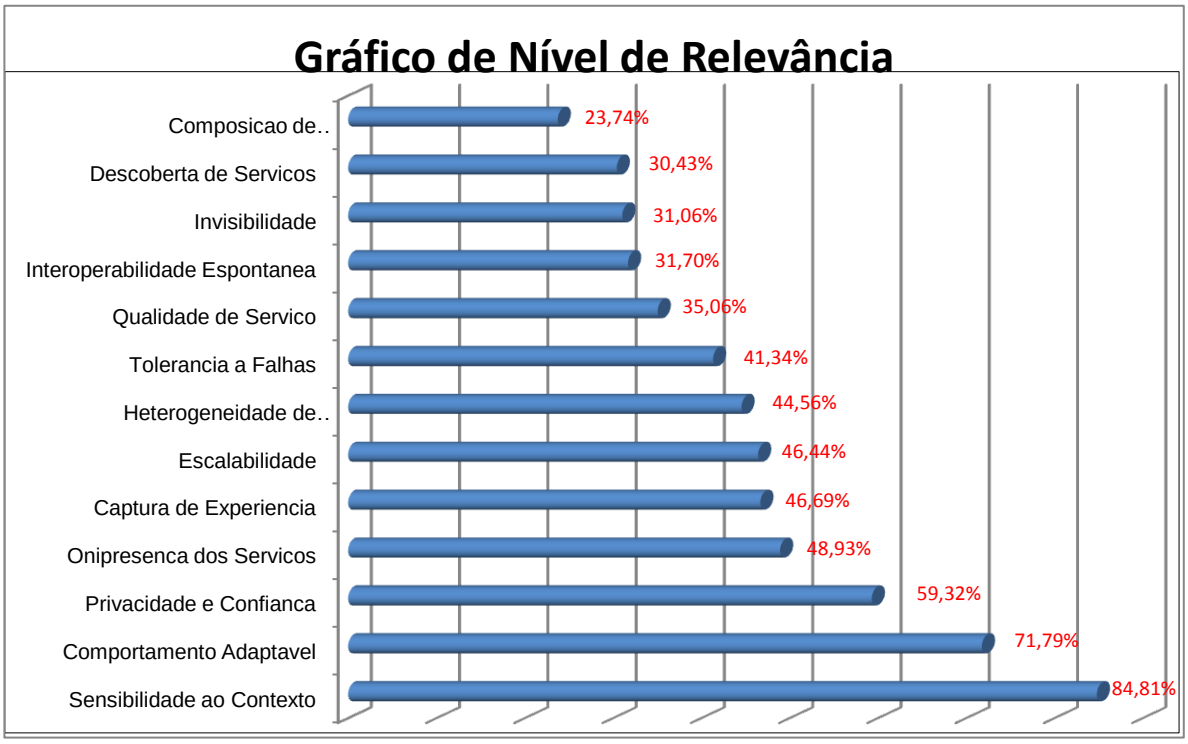


Figura 4.6. Relevância das características.

Dessa forma, observa-se que a Hipótese Nula 2 (H0 2) também foi refutada, uma vez que existiram características com diferentes níveis de relevância.

A Tabela 4.3 sumariza os resultados alcançados:

- As linhas destacadas em cinza indicam as características de ubiquidade consideradas no conjunto final. Esta seleção foi efetuada de acordo com o critério de inclusão definido no plano do *survey*;
- Sensibilidade ao contexto e comportamento adaptável são consideradas as características mais pertinentes e relevantes. É importante observar que estas características possuem uma relação de complemento entre elas;

Tabela 4.3. Pertinência e relevância das características de ubiquidade.

Característica	Pertinência	*Rank*	Nível de Relevância	*Rank*	Funcional / Restritivo
Sensibilidade ao Contexto	**93,47%**	**1**	**84,81%**	**1**	**F**
Comportamento Adaptável	**89,71%**	**2**	**71,79%**	**2**	**F**
Privacidade e Confiança	**76,21%**	**3**	**59,32%**	**3**	**R**
Heterogeneidade de Dispositivos	**62,34%**	**4**	**44,56%**	**7**	**F**
Captura de Experiência	**62,16%**	**5**	**46,69%**	**5**	**F**
Escalabilidade	**60,91%**	**6**	**46,44%**	**6**	**R**
Onipresença de Serviço	**59,03%**	**7**	**48,93%**	**4**	**F**
Tolerância a Falhas	**56,26%**	**8**	**41,34%**	**8**	**R**
Qualidade de Serviço	**52,42%**	**9**	**35,06%**	**9**	**R**
Interoperabilidade Espontânea	**49,28%**	**10**	**31,70%**	**10**	**F**
Usabilidade Universal	**-**	**-**	**-**	**-**	**R**
Descoberta de Serviço	**41,23%**	**12**	**30,43%**	**13**	**F**
Invisibilidade	**40,61%**	**13**	**31,06%**	**12**	**F**
Composição de Funcionalidade	**32,02%**	**14**	**23,74%**	**14**	**F**

- Existe um equilíbrio entre características funcionais (6/11) e restritivas (5/11). Isto pode indicar que características restritivas são críticas para esta categoria de software.

A partir destes resultados, foi possível também evoluir a definição de computacão ubíqua apresentada na seção 2.2.2 para: *a computação ubíqua se faz presente no momento em que os serviços ou facilidades computacionais podem se materializar em qualquer momento ou lugar, de forma transparente, através do uso de dispositivos de uso comum no dia-a-dia. Neste contexto, para que a computação ubíqua se faça presente, é preciso que os sistemas que compõem o ambiente contemplem (total ou parcialmente) as seguintes características:*

- *Funcional: sensibilidade ao contexto, comportamento adaptável, onipresença de serviço, heterogeneidade de dispositivos, captura de experiência e interoperabilidade espontânea;*
- *Restritiva: escalabilidade, privacidade e confiança, tolerância a falhas, qualidade de serviço e usabilidade universal.*

4.4 - Atualização do Corpo de Conhecimento

A organização do corpo de conhecimento apresentada no Capítulo 2 se deu entre os anos de 2005 e 2006. Este foi utilizado na caracterização de projetos de

software ubíquo identificados na literatura até o ano de 2007. Na sequência, o conhecimento organizado foi avaliado através da execucão de dois *surveys* entre os anos de 2008 e 2009. Como foi visto na seção anterior, esta avaliação nos permitiu chegar a um conjunto final de características da computação ubíqua assim como seus respectivos níveis de relevância.

Neste momento, baseado nesta lista final de características, definiu-se um protocolo de atualização do corpo de conhecimento. Esta definição foi motivada pela necessidade de:

- manutenção do corpo de conhecimento atualizado através da investigação de novas referências. Esta atualização é importante para que não se tenha um impacto negativo no uso do corpo de conhecimento;
- definição de um protocolo mais simplificado que permita a atualização constante do corpo de conhecimento com um esforço de execução do protocolo mais reduzido.

Desta forma, nas próximas sub-seções serão apresentados a descrição do protocolo de atualização do corpo de conhecimento e o resultado de sua aplicação considerando o período de janeiro de 2005 a fevereiro de 2010.

4.4.1 Protocolo de Atualização do Corpo de Conhecimento

O objetivo deste protocolo (apresentado no Apêndice C) é apóiar a execução de uma revisão sistemática para atualizar o corpo de conhecimento organizado em (SPÍNOLA and TRAVASSOS, 2010) considerando três questões de pesquisa:

- **P0:** Quais são as características básicas que definem softwares ubíquos?
- **P1:** Quais fatores funcionais caracterizam as características de ubiquidade (onipresença de serviços, sensibilidade ao contexto, comportamento adaptável, captura de experiência, heterogeneidade de dispositivos, interoperabilidade espontânea, escalabilidade, privacidade e confiança, tolerância a falhas, qualidade de serviço e usabilidade universal)?
- **P2:** Quais fatores restritivos caracterizam as características de ubiquidade (onipresença de serviços, sensibilidade ao contexto, comportamento adaptável, captura de experiência, heterogeneidade de dispositivos, interoperabilidade espontânea, escalabilidade, privacidade e confiança, tolerância a falhas, qualidade de serviço e usabilidade universal)?

Como fonte de artigos, foi definida a base eletrônica SCOPUS. O motivo de sua seleção se deve ao fato dela indexar as principais bases eletrônicas, incluindo ACM Digital Library e IEEE Explorer consideradas nas revisões da literatura apresentadas no Capítulo 3[6]. E como já descrito, serão considerados somente os artigos publicados a partir de 2005, sendo este o ano de realização das revisões sistemáticas que originaram o corpo de conhecimento base que será atualizado através da execução deste protocolo. Além deste filtro, outros critérios de inclusão foram definidos:

- Os artigos devem estar disponíveis na web;
- Os artigos devem estar em inglês;
- Os artigos devem apresentar características que definem softwares ubíquos (exclusivo para P0).
- Os artigos devem contemplar fatores funcionais pertinentes a cada uma das características de ubiquidade para a qual a *string* de busca foi formulada (exclusivo para P1).
- Os artigos devem contemplar fatores restritivos pertinentes a cada uma das características de ubiquidade para a qual a *string* de busca foi formulada (exclusivo para P2).

Um pesquisador deve aplicar a estratégia de busca para a identificação de potenciais artigos. Os artigos identificados serão selecionados pelo mesmo pesquisador através da leitura e verificação dos critérios de inclusão e exclusão estabelecidos. Para cada estudo selecionado, o pesquisador irá extrair as seguintes informações:

- Título do artigo
- Autores
- Fonte
- Característica identificada (exclusivo P0)
- Descrição das características (exclusivo P0 – deve ser preenchido caso a característica seja nova)
- Fatores funcionais (exclusivo P1)
- Fatores restritivos (exclusivo P2)

[6] A base SCOPUS não foi utilizada nas revises sistemáticas apresentadas no Capítulo 3 pelo fato dela possuir uma quantidade limitada de indexações de outras bases de artigos no período em que elas foram executadas. Contudo, desde então a base SCOPUS foi aprimorada e tem sido bastante utilizada no apoio à execução de revisões sistemáticas da literatura (CHEN *et al.*, 2010).

Ao final os resultados serão tabulados e serão realizadas análises para:

- definir se apareceram novas características de ubiquidade a serem acrescentadas ao conjunto inicial.
- definir os novos fatores funcionais para cada uma das características de ubiquidade.
- definir os novos fatores restritivos para cada uma das características de ubiquidade.

4.4.2 Atualização do Corpo de Conhecimento

A execução deste protocolo de atualização no ano de 2010 resultou na seleção de 48 artigos para análise. Desse total, 24 se referiam à identificação das características de ubiquidade, 14 estavam associados aos fatores funcionais das características e 10 estavam associados aos fatores restritivos. Como resultado:

- Não foram identificadas novas características de ubiquidade no período de 2005 a 2010;
- Foram identificados e acrescentados ao corpo de conhecimento 7 novos fatores funcionais:
    - (comportamento adaptável) Permitir que elementos estruturais (entidades) adicionais sejam associados a pontos específicos da aplicação que serão utilizados a depender do contexto inserido;
    - (comportamento adaptável) Permitir a definição de valores para as entidades de acordo com o contexto em que se encontra;
    - (comportamento adaptável) Fazer uso de workflow para orquestrar os serviços providos;
    - (comportamento adaptável) Associar atividades de captura de informações de contexto ao workflow;
    - (comportamento adaptável) Associar atividades de adaptação com base nas informações de contexto ao workflow;
    - (comportamento adaptável) A adaptação do workflow pode considerar 3 níveis: abstrato, concreto e de instância;
    - (comportamento adaptável) Esta adaptação envolve a adição, remoção e/ou substituição de tarefas do workflow.
- Foram identificados e acrescentados ao corpo de conhecimento 7 novos fatores restritivos:

- (escalabilidade) permitir a adição de novos recursos computacionais para lidar com a demanda crescente de uso do sistema;
- (escalabilidade) lidar com o aumento da carga de trabalho de forma transparente para o usuário;
- (qualidade de serviço) manter os níveis de serviços providos ao usuário dentro de padrões mínimos de qualidade;
- (privacidade e confiança) possuir mecanismos de criptografia de informações trafegadas pelo sistema;
- (privacidade e confiança) possuir mecanismos de controle de acesso aos dados trafegados de forma que apenas usuários autorizados podem ter acesso às informações;
- (usabilidade universal) possuir mecanismos que permitam o uso da aplicação em diferentes dispositivos;
- (usabilidade universal) considerar diferentes perspectivas de uso na interface do sistema.

A Tabela 4.4 apresenta o estado atual do conjunto de características e fatores associados.

Tabela 4.4. Estado atual do Corpo de Conhecimento organizado em computação ubíqua.

Característica	# Fatores	Funcional/ Restritivo
Onipresença de serviço	20	F
Sensibilidade ao contexto	30	F
Comportamento adaptável	39	F
Captura de experiência	7	F
Heterogeneidade de serviços	11	F
Interoperabilidade Espontânea	12	F
Tolerância a falhas	6	R
Escalabilidade	2	R
Qualidade de serviço	1	R
Privacidade e confiança	2	R
Usabilidade universal	2	R

Estes resultados nos permitem dizer que o corpo de conhecimento elaborado permanece até o momento válido, estável e atualizado considerando o cenário atual das pesquisas em computação ubíqua (SPÍNOLA and TRAVASSOS, 2010).

4.5 - Considerações Finais

Este capítulo apresentou a avaliação do corpo de conhecimento organizado no Capítulo 3. Esta avaliação ocorreu através do planejamento e execução de dois *surveys* cujo os resultados permitiram definir a pertinência e relevância das características da computação ubíqua na caracterização de projetos de software. Além disso, foi apresentado também o protocolo de atualização do corpo de conhecimento e os resultados de sua execução.

Considerando estes resultados, alguns questionamentos podem ser feitos:

- Conhecidas as características da computação ubíqua, como apoiar o desenvolvimento de projetos de software ubíquo?
- Existem abordagens que consideram estas caraterísticas no apoio ao desenvolvimento de projetos de software ubíquo?
- Como aprimorar a qualidade do software ubíquo a partir das características da computação ubíqua?

Nesta pesquisa, em particular, foi identificado que o conhecimento organizado poderia apoiar atividades associadas a definição e verificação de requisitos para projetos de software ubíquos. Entretanto, antes de iniciar o desenvolvimento de uma nova tecnologia nesta área, buscou-se identificar na literatura técnica trabalhos relacionados à engenharia de requisitos apoiando o desenvolvimento de projetos de software ubíquos.

Assim, no próximo capítulo serão apresentados os resultados de uma revisão de literatura que foi planejada com o objetivo de identificar abordagens de apoio à definição e garantia de qualidade de requisitos de ubiquidade em projetos de software ubíquo.

Capítulo 5 – Engenharia de Requisitos e Computação Ubíqua

Este capítulo apresenta o planejamento e os resultados alcançados com a execução de uma revisão da literatura cujo objetivo foi identificar abordagens de apoio à definição e garantia de qualidade de requisitos de ubiquidade em projetos de software ubíquo.

5.1 – Introdução

A engenharia de software pode ser definida como sendo o estudo ou aplicação de abordagens sistemáticas, econômicas e quantificáveis para o desenvolvimento, operação e manutenção de software de qualidade (PFLEEGER, 2007). Engenheiros de software devem adotar uma abordagem sistemática e organizada para seu trabalho e usar ferramentas e técnicas/métodos apropriados dependendo do problema a ser solucionado, das restrições de desenvolvimento e dos recursos disponíveis.

Neste contexto, a engenharia de requisitos, uma subárea da engenharia de software, estuda o processo de definição dos requisitos que o software deverá atender e como gerenciar estes requisitos (PFLEEGER, 2007). Ela aborda uma etapa crucial no ciclo de vida do desenvolvimento de software por tratar de conhecimentos não apenas técnicos, mas também gerenciais, organizacionais, econômicos e sociais. Segundo o Nuseibeh e Easterbrook (2000), é uma disciplina para desenvolver uma especificação completa, consistente e não ambígua, que sirva como base para um acordo entre todas as partes envolvidas, descrevendo o que o produto de software irá ou não fazer.

Nesta pesquisa, estamos particularmente interessados em abordagens para apoiar as atividades de definição e verificação de requisitos de ubiquidade em projetos de software considerando as características da computação ubíqua apresentadas no Capítulo 4. Requisitos bem especificados podem ser considerados um fator de sucesso para a entrega de produtos de software com qualidade e dentro de prazos e custo estabelecidos (PFLEEGER, 2007). Isto não deve ser diferente quando lidamos com projetos de software ubíquo.

Neste contexto, este capítulo apresenta o planejamento e os resultados de uma revisão da literatura que foi executada com o objetivo de identificar possíveis

abordagens para apoiar as atividades de definição e garantia de qualidade de requisitos de ubiquidade em projetos de software.

5.2 – Trabalhos Relacionados

Nesta seção, serão apresentadas as abordagens identificadas na literatura técnica que estão relacionadas à proposta apresentada neste trabalho, seja por apresentar uma abordagem de definição ou verificação de requisitos, ou simplesmente por apresentar exemplos de requisitos de ubiquidade definidos para um determinado projeto. A revisão da literatura foi **inspirada** em conceitos da revisão sistemática, cujo protocolo simplificado é apresentado na próxima seção.

5.2.1 Planejamento e Execução da Revisão da Literatura

Usando a abordagem descrita em (BIOLCHINI *et al*., 2005), o planejamento da revisão da literatura está sumarizado a seguir:

- **Objetivo:** identificar abordagens que apóiem a definição e a verificação de requisitos de ubiquidade em projetos de software.
- **Questões de pesquisa**: para a elaboração das questões de pesquisa, tomou-se como base o objetivo da revisão da literatura e associou-se este objetivo a cada uma das características **funcionais**[7] de ubiquidade e ao conceito de computação ubíqua. Uma abordagem semelhante para definição de questões de pesquisa foi utilizada por (ARAUJO e TRAVASSOS, 2008). Associado a cada questão de pesquisa, tem-se definidas as *strings* de busca elaboradas.
    - 1: Existem abordagens que apóiam a definição e a verificação de requisitos de ubiquidade em projetos de software? Se sim, como se caracterizam?
        - ((ubicomp <or> ubiquitous computing <or> pervasive computing <or> ambient intelligence <or> ubiquitous <or> pervasive) <and> (requirement definition <or> requirement elicitation <or> requirement identification <or> requirement analysis) <and> (approach <or> methodology <or> method <or> guideline <or> process <or> systematic))
        - ((ubicomp <or> ubiquitous computing <or> pervasive computing <or> ambient intelligence <or> ubiquitous <or>

[7] *Serão consideradas apenas as características funcionais no contexto desta revisão. Este limite de espoco se deve ao fato da proposta deste trabalho estar associada às características funcionais de ubiquidade.*

pervasive) <and> (requirement verification <or> requirement inspection <or> requirement review) <and> (approach <or> methodology <or> method <or> guideline <or> process <or> systematic)))

- 2: Existem abordagens que apóiam a definição e a verificação de requisitos de ubiquidade associadas à característica de ubiquidade sensibilidade ao contexto? Se sim, como se caracterizam?
  - ((context sensitivity <or> context aware <or>context awareness) <and> (requirement definition <or> requirement elicitation <or> requirement identification <or> requirement analysis))
  - ((context sensitivity <or> context aware <or>context awareness) <and> (requirement verification <or> requirement inspection <or> requirement review))
- 3: Existem abordagens que apóiam a definição e a verificação de requisitos de ubiquidade associadas à característica de ubiquidade comportamento adaptável? Se sim, como se caracterizam?
  - ((adaptability <or> adaptable behavior) <and> (requirement definition <or> requirement elicitation <or> requirement identification <or> requirement analysis))
  - ((adaptability <or> adaptable behavior) <and> (requirement verification <or> requirement inspection <or> requirement review))
- 4: Existem abordagens que apóiam a definição e a verificação de requisitos de ubiquidade associadas à característica de ubiquidade captura de experiência? Se sim, como se caracterizam?
  - ((automated capture <or> experience capture) <and> (requirement definition <or> requirement elicitation <or> requirement identification <or> requirement analysis))
  - ((automated capture <or> experience capture) <and> (requirement verification <or> requirement inspection <or> requirement review))
- 5: Existem abordagens que apóiam a definição e a verificação de requisitos de ubiquidade associadas à característica de ubiquidade interoperabilidade espontânea? Se sim, como se caracterizam?

        - ((spontaneous interoperability <or> interoperability) <and> (requirement definition <or> requirement elicitation <or> requirement identification <or> requirement analysis))
        - ((spontaneous interoperability <or> interoperability) <and> (requirement verification <or> requirement inspection <or> requirement review))
    - 6: Existem abordagens que apóiam a definição e a verificação de requisitos de ubiquidade associadas à característica de ubiquidade heterogeneidade de dispositivos? Se sim, como se caracterizam?
        - ((device heterogeneity <or> heterogeneity of devices) <and> (requirement definition <or> requirement elicitation <or> requirement identification <or> requirement analysis))
        - ((device heterogeneity <or> heterogeneity of devices) <and> (requirement verification <or> requirement inspection <or> requirement review))
    - 7: Existem abordagens que apóiam a definição e a verificação de requisitos de ubiquidade associadas à característica de ubiquidade onipresença de serviços? Se sim, como se caracterizam?
        - ((service omnipresence <or> computer everywhere) <and> (requirement definition <or> requirement elicitation <or> requirement identification <or> requirement analysis))
        - ((service omnipresence <or> computer everywhere) <and> (requirement verification <or> requirement inspection <or> requirement review))

- **Fontes de busca**:
    - Biblioteca digital IEEEXplorer;
    - Biblioteca digital da ACM Portal;
    - *Proceedings* do UBICOMP;
    - International Requirements Engineering Conference;
    - Possíveis referências identificadas nos artigos selecionados para revisão.

A quantidade de bibliotecas digitais também foi reduzida nesta revisão devido à grande quantidade de *strings* de busca utilizada. Isto dificultaria a adaptação destas diferentes *strings* para cada máquina de busca e tornaria a análise dos resultados excessivamente demorada.

- **Critérios de Inclusão e Exclusão de artigos**:
    - Os artigos devem estar disponíveis na web;
    - Os artigos devem estar em inglês;
    - Os artigos devem contemplar abordagens para definição ou verificação de requisitos de ubiquidade em projetos de software ubíquo com embasamento científico ou que foram experimentalmente caracterizadas.

A partir deste protocolo, a revisão da literatura foi executada. Somados os resultados, obteve-se um total de 130 artigos retornados. Um pesquisador deu início ao processo de filtragem (todos os artigos foram analisados). Durante o processo de filtragem bastante material foi desconsiderado. Aplicada a filtragem considerando os critérios de inclusão e exclusão, chegou-se a uma quantidade de 9 artigos para leitura e extração dos resultados. Abaixo temos uma listagem das propostas identificadas:

- Casos de Uso Executáveis que exploraram o detalhamento dos requisitos a partir da simulação de casos de uso (JORGENSEN e BOSSEN, 2003);
- Modelagem de Requisitos com a Linguagem KAOS que define quatro níveis de requisitos e utiliza essa linguagem para formalizar os requisitos associados a cada nível (GOLDSBY e CHENG, 2007) (CHENG *et al.*, 2005);
- Metodologia para Definir Requisitos de Sistemas Adaptáveis e Sensíveis ao Contexto (HONG *et al*., 2005) (CHIU *et al*., 2007);
- Definição de Requisitos do Usuário com Base em uma Ontologia que compreende atividades para construir uma ontologia do domínio e a partir dela definir os requisitos de ubiquidade (BO *et al*., 2007);
- Definição de Requisitos de Serviços que elabora modelos com base em uma ontologia e a partir do preenchimento de um questionário de caracterização (XIANG *et al.*, 2007);
- Definição de um conjunto de exemplos de requisitos de ubiquidade no contexto do projeto ENQUETE-BAISE (CHENG *et al.*, 2007);
- Definição de um conjunto de exemplo de requisitos de ubiquidade no contexto de um sistema Multimodal Multimedia tolerante a falhas e ubíquo (CHERIF *et al*., 2007).

A Tabela 5.1 sumariza os trabalhos identificados, a abordagem proposta no contexto da engenharia de requisitos, a característica de ubiquidade considerada e a questão de pesquisa associada.

Tabela 5.1. Abordagens identificadas.

Questão de Pesquisa	Trabalhos identificados	Abordagem Proposta	Características de Ubiquidade
2	(BO *et al.,* 2007)	Identificação	Sensibilidade ao contexto
2	(CHIU *et al.,* 2007)	Identificação e especificação	Sensibilidade ao contexto
1	(CHENG *et al.,* 2007)	Especificação	Não especificado[8]
1	(CHERIF *et al.,* 2007)	Especificação	Não especificado
7	(XIANG *et al.,* 2007)	Identificação	Onipresença de Serviços
2	(HONG *et al.,* 2005)	Identificação e especificação	Sensibilidade ao contexto
4	(GOLDSBY e CHENG, 2007) (CHENG *et al.,* 2005)	Identificação e especificação	Sensibilidade ao contexto e Comportamento adaptável
1	(JORGENSEN e BOSSEN, 2003)	Especificação	Não especificado

Essa revisão da literatura foi realizada no início de 2008 e re-executada no ano de 2009 por (PINTO, 2009). Durante esta re-execução, dois novos trabalhos relacionados foram identificados:

- Processo Sistemático para Análise de Requisitos de Segurança em Sistemas Embarcados que auxilia na identificação e tratamento dos casos de uso que podem trazer problemas de segurança (MARKOSE *et al.*,2008);
- Processo para Definição de Requisitos com Base nas Interações do Usuário que observa os comportamentos dos usuários para sugerir melhorias para o sistema (OYAMA *et al.*, 2008 e MING *et al.,*2008).

Em 2010 uma nova re-execução foi efetuada. Entretanto, não foram identificados outros trabalhos considerando o protocolo de execução.

5.2.2 Trabalhos Relacionados Identificados

As nove abordagens identificadas serão apresentadas nas subseções a seguir. Quando possível, tentou-se identificar: passos realizados pelo desenvolvedor para utilizar a abordagem; apoio fornecido para melhorar a qualidade dos requisitos; e relatos de estudos ou aplicações em projetos de software ubíquo.

[8] *Este trabalho não menciona nenhuma das características de ubiquidade.*

5.2.2.1 Casos de Uso Executáveis

Jorgensen e Bossen (2003) apresentam uma abordagem para definição de requisitos considerando o contexto de desenvolvimento de um sistema pervasivo de acompanhamento médico (PHCS – *Pervasive Health Care System*).

Esta proposta é fundamentada no conceito de casos de uso e se chama casos de uso executáveis (EUC – *Executables Use Cases*). A abordagem apresentada segue um fluxo interativo composto por três camadas:

- Inicialmente, o domínio do problema deve ser conhecido. Ele será base para as atividades relacionadas à primeira camada referente à definição textual dos requisitos. Esta definição é feita na descrição dos casos de uso.
- Para a segunda camada, o conjunto de atividades que representam as interações entre homem-máquina são modeladas utilizando uma notação com semântica bem definida. São exemplos deste tipo de notação: diagrama de estados, diagrama de atividades.
- Tendo modelado o comportamento do sistema, é elaborado um conjunto de protótipos que apoiarão as atividades de verificação e validação dos requisitos, e dessa forma, apoiar na obtenção de requisitos mais completos.

A técnica apóia a definição de requisitos através da possibilidade de mostrar para o usuário como os requisitos que estão sendo definidos serão implementados no sistema. Dessa forma, ele pode verificar com maior facilidade se o fluxo de trabalho que está sendo descrito atende às suas necessidades.

Os casos de uso executáveis foram utilizados na definição de requisitos do *Pervasive Health Care System* (PHCS), um software ubíquo de prontuário eletrônico utilizado no hospital de Aarhus Country (Dinamarca).

Embora essa abordagem tenha sido utilizada para apoiar a definição de um software ubíquo, ela não compreende nenhuma característica de ubiquidade e aparentemente poderia ser utilizada para apoiar outros tipos de software.

5.2.2.2 Modelagem de Requisitos com a Linguagem KAOS

Goldsby e Cheng (2007) argumentam que um fator chave no desenvolvimento de sistemas dinamicamente adaptáveis (DAS) é ter sólido entendimento dos requisitos do sistema. Para isto, os autores partem do trabalho definido em (BERRY *et al*., 2005) e (CHENG *et al.*, 2005) que propõem o uso de quatro níveis que devem ser considerados em atividades relacionadas à engenharia de requisitos no contexto dos DAS:

- Nível 1: corresponde às atividades tradicionais de identificação de requisitos. Neste caso, são identificadas informações do domínio da aplicação que será construída e é realizada a identificação de todas as possíveis configurações que o DAS pode atingir;
- Nível 2: corresponde ao trabalho efetuado na definição do comportamento do DAS em tempo de execução para identificar quais variáveis do ambiente serão utilizadas para o DAS determinar como efetuar as adaptações;
- Nível 3: corresponde ao trabalho executado para definir a partir do domínio da aplicação e das informações específicas da aplicação qual técnica (baseada nas técnicas definidas em 4) de adaptação utilizar;
- Nível 4: corresponde ao trabalho de pesquisa com intuito de criar novos mecanismos de adaptação.

Baseado neste conjunto de níveis, é proposta uma abordagem apoiada pela linguagem de especificação KAOS (DARDENNE *et al.*, 1993) para apoiar a modelagem dos requisitos. Esta abordagem basicamente efetua um mapeamento entre os níveis descritos acima e como estes podem ser formalizados utilizando a linguagem KAOS.

A abordagem considera as características Sensibilidade ao Contexto e Comportamento Adaptável e fornece através dos níveis dos requisitos, conhecimento sobre as informações que precisam ser capturadas. Adicionalmente, a formalização dos requisitos permite a identificação de inconsistências.

5.2.2.3 Metodologia para Definir Requisitos de Sistemas Adaptáveis e Sensíveis ao Contexto

Chiu *et al*. (2007) apresentam uma abordagem (baseada na proposta inicial de Hong *et al.* (2005)) para auxiliar a identificação de requisitos considerando projetos de software ubíquo cujas características principais são: estar disponível na web e ser sensível ao contexto. Para isto, eles utilizam como base os seguintes objetivos:

- Usabilidade: um sistema deve prover uso efetivo e eficiente de seus serviços assim como facilitar para o usuário aprender como usar o sistema e lembrar-se do que aprendeu;
- Experiência do usuário: o software deve ser usável, efetivo e agradável de usar.

Estes elementos estão associados diretamente a algumas questões discutidas corriqueiramente na área de ubiquidade computacional: facilidade computacional, interação com usuário e ambiente.

Partindo destes conceitos, os autores estendem o conceito de contexto para três principais fatores: computacional, usuário e físico. Aliado a isso, eles consideram também que o desenvolvimento de software para web tem sido feito utilizando o conceito de três camadas: interface com usuário, lógica e de dados. Unindo estes conceitos às perspectivas de contexto consideradas, eles chegam ao modelo conceitual apresentado na Figura 5.1. Baseado neste modelo conceitual, os autores propõem um conjunto de passos a serem seguidos para auxiliar na identificação de requisitos e projeto do software:

1. Determinar os diferentes grupos alvo de usuários que participarão da definição do software;
2. Identificar os contextos[9] em que estes grupos de usuário podem estar presentes, especialmente os contextos que influenciam nas atividades do sistema;
3. Para cada contexto definido, listar seus requisitos específicos;
4. Para cada serviço, investigar como diferentes contextos podem impactar nas suas interações e assim determinar as características de sensibilidade ao contexto requeridas para o sistema;
5. Para cada característica requerida, detalhar as funcionalidades;
6. Verificar se as funcionalidades definidas atendem às necessidades dos requisitos definidos no passo 3. Caso negativo, retornar ao passo 4;
7. Projetar visões do processo que capture conjuntos típicos de características de sensibilidade ao contexto para adaptação;
8. Projetar visões de dados para cada fonte de dados baseada nos requisitos da visão de processos e funcionalidades sensíveis ao contexto;
9. Projetar as visões de interface com o usuário baseado na plataforma para a qual será desenvolvido o software;
10. Executar algum mecanismo de validação para checar a consistência entre as visões.

[9] *Considera-se contexto neste trabalho como sendo as situações de uso em ambientes diferenciados que levem à necessidade de funcionalidades específicas para cada ambiente.*

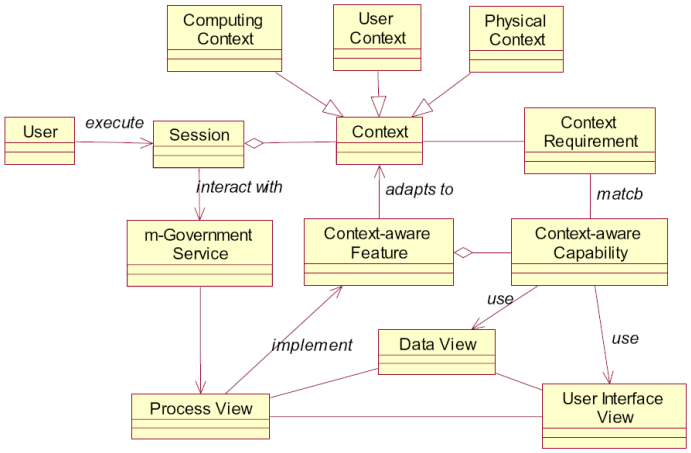


Figura 5.1. Abordagem de apoio à identificação de requisitos (CHIU *et al.,* 2007).

5.2.2.4 Definição de Requisitos do Usuário com base em uma Ontologia

Bo *et al*. (2007) relatam inicialmente que a identificação de requisitos pode ser considerada uma das mais importantes áreas da engenharia de requisitos e possivelmente, do processo de software como um todo. Entretanto, segundo os autores, requisitos de software orientados a serviços diferem de requisitos identificados para software convencionais. Neste contexto, os autores apresentam uma abordagem para identificação de requisitos baseada em ontologia. Esta é a base semântica para a estruturação dos requisitos de forma que se possa identificar e definir o contexto no qual cada requisito estará associado (os serviços são disponibilizados de acordo com o contexto em que o usuário se encontra).

A abordagem é composta por duas etapas. Primeiro o desenvolvedor constrói a ontologia de requisitos, depois realiza as seguintes atividades para construir uma árvore de requisitos:

1. decompor a ontologia de requisitos em sub-requisitos;
2. repetir a atividade 1 para os sub-requisitos até que eles não possam mais ser decompostos;
3. construir uma árvore de requisitos para conectar os requisitos definidos na ontologia com todos os seus sub-requisitos.

Para isto, a ontologia utilizada contém a definição do conhecimento do domínio para a área na qual o software será desenvolvido de forma a apoiar a definição de requisitos (ver Figura 5.2).

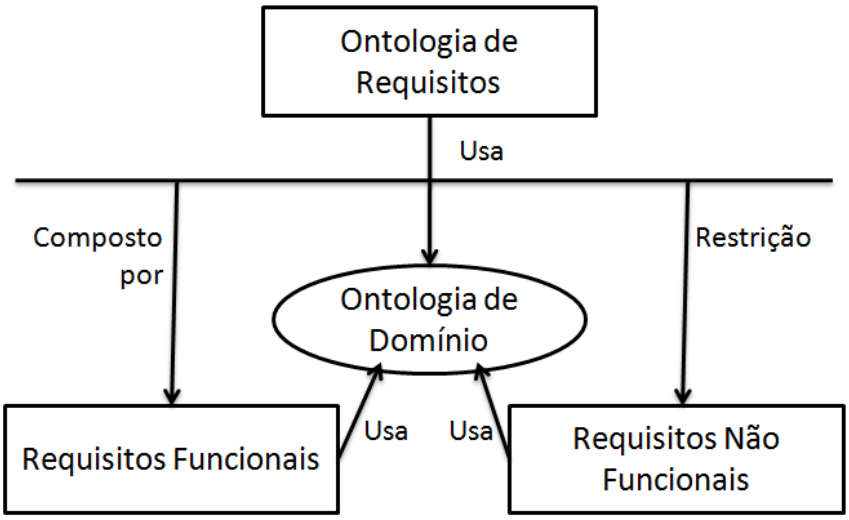


Figura 5.2. Abordagem de apoio à identificação de requisitos. Adaptada de (BO *et al.*, 2007).

A ontologia de requisitos define em alto nível de abstração um conjunto de conhecimentos para auxiliar na representação da formulação de um problema. A ontologia de domínio contém o conjunto de conceitos e seus relacionamentos semânticos para o domínio em questão. Os requisitos funcionais representam as necessidades comportamentais a serem capturadas para serem disponibilizadas através de serviços. Requisitos não funcionais representam limitadores dos requisitos funcionais identificados.

Definida esta abordagem inicial, os autores argumentam que mudanças no contexto dos usuários podem implicar em mudanças de suas necessidades frente a serviços disponibilizados por software. Então, é proposto o uso de sensibilidade ao contexto para apoiar a evolução de requisitos do usuário. Esta abordagem é interessante uma vez que associa os diferentes contextos de uso do sistema a seus respectivos requisitos, facilitando o projeto de aplicações sensíveis ao contexto e onipresentes.

5.2.2.5 Mecanismo para Definição de Requisitos de Serviços

Em seu trabalho, Xiang *et al*. (2007) argumentam que a atividade de identificação de requisitos para serviços impõe diferentes desafios daqueles encontrados nos processos de identificação de requisitos convencionais. Em abordagens convencionais, tem-se procedimentos voltados para levantamentos face a face com o usuário. Empresas que pretendam disponibilizar serviços geralmente não possuem um público alvo de usuários tão bem definido. Ainda assim, é importante ter mecanismos eficazes de definição de requisitos, uma vez que o sucesso dos serviços disponibilizados depende de quão bem os fornecedores e usuários dos serviços estão de acordo com os requisitos definidos para os serviços.

Os autores argumentam que a engenharia de requisitos para serviços representa um importante papel durante o ciclo de vida de desenvolvimento dos serviços. Para se ter serviços eficientemente projetados, publicados, descobertos, ocultados e evoluídos, é preciso facilidades computacionais associadas às atividades de definição de requisitos para serviços.

Neste contexto, os autores apresentam o SREM (*Service Requirement Elicitation Mechanism*). Este é formado por três componentes principais:

- SRMO (*Service Requirement Modeling Ontology*): define a ontologia para a modelagem de requisitos;
- SREP (*Service Requirement Elicitation Process*): direciona o processo para elicitar detalhes de requisitos e informações estruturais;
- SRRP (*Service Requirement Reconciliation Process*): utilizado para integrar as instâncias dos requisitos identificados a partir dos diferentes usuários. Esse conhecimento é utilizado para construir um repositório de conhecimento de requisitos o qual conterá heurísticas para publicação de serviços.

De forma simplificada, a abordagem proposta considera a seguinte linha de raciocínio: (1) é definida uma ontologia contendo os conceitos que serão utilizados para capturar as necessidades do usuário, ou seja, define-se a ontologia para a modelagem dos requisitos. (2) De posse desta ontologia, é proposto um conjunto de questionamentos a serem realizados para instanciar os conceitos definidos na ontologia. Este conjunto de questionamentos e a ordem e relacionamentos entre eles é definida no SREP. (3) Por fim, os requisitos identificados são mapeados para a ontologia e passa-se por um processo de conciliação.

5.2.2.6 Exemplos de Requisitos de Ubiquidade do Projeto ENQUETE-BAISE

Em (CHENG *et al*., 2007), os autores apresentam o desenvolvimento do ENQUETE-BAISE, um servidor de questionários para funcionar em ambientes computacionais ubíquos. Os autores não apresentam uma técnica para definição ou verificação de requisitos, apenas listam alguns requisitos definidos para o projeto. Entretanto, a forma como os requisitos foram definidos podem servir de exemplo sobre como definir requisitos de projetos de software ubíquo.

Basicamente, os autores consideraram dois níveis de abstração ao definir os requisitos funcionais para o sistema. Inicialmente eles listam uma série de características genéricas que o software deverá possuir. Em seguida, eles definem o

conjunto de requisitos funcionais (chamados no artigo de requisitos concretos) associados a estas características. Por exemplo:

Requisito genérico:

"ER1(anytime and anywhere availability, maintainability, and reconfigurability): As ubiquitous questionnaire required, the E-questionnaire server should be able to provide services for its users anytime and anywhere, and should be able to be maintained and reconfigured anytime and anywhere without stopping its services." (CHENG *et al*., 2007)

Requisito concreto:

"CR1: The E-questionnaire server must run continuously and persistently to provide its services without stopping even if it needs to be maintained, upgraded, or reconfigured, it has some trouble, or it is attacked." (CHENG *et al.*, 2007)

5.2.2.7 Exemplos de Requisitos de Ubiquidade de um Sistema Multimodal Multimedia

Cherif *et al*. (2007) apresentam o desenvolvimento de um sistema Multimodal Multimedia tolerante a falhas e ubíquo. Assim como no trabalho analisado anteriormente, os autores não apresentam uma técnica para definição ou verificação de requisitos, mas a forma como os requisitos foram definidos pode servir de exemplo sobre como definir requisitos de projetos de software ubíquo.

Neste caso, os autores simplesmente listam os possíveis requisitos para o sistema. Não há um detalhamento adicional e nem uma preocupação diferenciada pelo fato de se tratar de um sistema ubíquo. Um exemplo de requisito definido pode ser visto abaixo:

"The safety factor – this refers to the detection of how safe (or risky) the user's workplace is based on (1) the presence or absence of other people in the vicinity of user's workplace, and (2) who is sitting in the user's chair facing the computer. To accomplish this, a software agent has to read samples from two sensors, namely: (1) an infrared detector that detects the presence of other people within the vicinity of user's workplace, and (2) a camera with retinal recognition that detects if it is the legitimate user who is sitting in the user's seat. The results of these two sensors are combined together to determine the safety factor in the user's workplace. The agent yields a final assessment that indicates if the safety factor is either (i) good or ideal, (ii) acceptable, (iii) sensitive, or (iv) bad, worse or worst. The calculation for the final assessment will be provided in the next section." (CHERIF *et al*., 2007)

5.2.2.8 Processo Sistemático para Análise de Requisitos de Segurança em Sistemas Embarcados

MARKOSE *et al*. (2008) propuseram um processo sistemático para analisar requisitos de segurança em sistemas embarcados. Ele é baseado nas seguintes atividades:

1. desenvolver um modelo de objetos de contexto para mostrar as relações do sistema com componentes externos. Esses modelos são desenvolvidos através da técnica *High-order Object-oriented Modeling Technique* (HOOMT) (LIU *et al*. 2001);
2. representar os sub-componentes de cada objeto;
3. especificar através de casos de uso as principais funcionalidades de cada objeto; e
4. identificar e tratar os casos de uso que podem trazer problemas de segurança.

O processo foi utilizado para apoiar a definição de requisitos do *FACTS Power System*. Nele, dispositivos foram incorporados em uma rede de distribuição de energia para prover um sistema de controle distribuído, tolerante a falhas e em tempo real.

5.2.2.9 Processo para Definição de Requisitos com base nas Intenções dos Usuários

OYAMA *et al.* (2008) e MING *et al*. (2008) propuseram um processo para definição de requisitos com base nas intenções dos usuários. Ele tem como objetivo descobrir os desejos do usuário e sugerir alterações no sistema. Para isso, são propostas as seguintes atividades:

1. observar padrões de comportamento dos usuários, por exemplo, a frequência de uso do sistema;
2. observar relatos do usuário, por exemplo, *defeitos* encontrados e comentários sobre o sistema; e
3. associar os padrões de comportamento e relatos do usuário aos requisitos do sistema e alterá-los de acordo com a necessidade.

Esse processo demanda que o sistema já esteja funcionando para ser realizado, portanto, se enquadra como uma abordagem incremental para a definição de requisitos.

Embora a abordagem não compreenda nenhuma característica de ubiquidade, o trabalho relata o uso de softwares sensíveis ao contexto para auxiliar na observação dos padrões de comportamento dos usuários.

5.2.3 Análise das Abordagens Identificadas na Literatura

Além da própria descrição de cada abordagem identificada nos 11 artigos analisados, alguns pontos interessantes que puderam ser observados foram:

- Utilização de “projeto interativo” nas atividades. Neste contexto, projeto interativo significa quebrar a atividade de identificação e definição de requisitos em um conjunto de passos, e tornar a execução destes passos baseada em uma interação mais forte com o cliente;
- Uso de ontologias em algumas abordagens para apoiar as atividades de identificação dos requisitos;
- As características de ubiquidade consideradas foram comportamento adaptável e sensibilidade ao contexto;
- As abordagens apresentadas em (CHIU *et al.*, 2007) e (HONG *et al.*, 2005) definem um conjunto de passos descrevendo as atividades a serem executadas para apoiar a definição dos requisitos.

Por outro lado, notou-se também:

- A ausência de abordagens para lidar com a verificação dos requisitos de ubiquidade;
- A ausência de uma preocupação mais detalhada em indicar como executar cada uma das atividades definidas nas abordagens com base em conhecimentos sobre computação ubíqua ou a característica de ubiquidade específica considerada nos trabalhos;
- Foram identificados no máximo alguns passos que deveriam ser executados pelo analista de requisitos para identificar e especificar os requisitos de ubiquidade. Ou seja, não há a definição de um processo nem um detalhamento sobre técnicas que podem ser utilizadas para apoiar a execução de suas atividades.

Este cenário reforça a importância em se definir uma abordagem de apoio às atividades de definição e verificação de requisitos no contexto de desenvolvimento de projetos de software ubíquo.

5.3 – Considerações Finais

Este capítulo apresentou um conjunto de abordagens que apóiam as atividades de identificação e especificação de requisitos de ubiquidade. Para isto, foi elaborado um protocolo de busca baseado nos princípios da revisão sistemática. Nenhuma abordagem de apoio à garantia de qualidade de requisitos de ubiquidade foi identificada.

A partir da elaboração do corpo de conhecimento sobre computação ubíqua apresentada até o Capítulo 4 e dos resultados alcançados a partir da investigação sobre trabalhos relacionados, este trabalho apresenta uma abordagem de apoio à definição e verificação dos requisitos de ubiquidade em projetos de software (SPÍNOLA *et al*., 2008). O capítulo seguinte descreve a abordagem proposta e os elementos que a compõem.

Capítulo 6 – *u*-RDV: Abordagem de Apoio à Definição e Verificação de Requisitos Funcionais de Ubiquidade em Projetos de Software

Este capítulo apresenta u-RDV, uma abordagem para apoiar a definição e a verificação de requisitos funcionais de ubiquidade em projetos de software. A abordagem é fundamentada em três elementos: o corpo de conhecimento em computação ubíqua, uma abordagem de apoio às atividades de definição de requisitos e uma abordagem de apoio às atividades de verificação de requisitos.

6.1 - Introdução

O desenvolvimento de projetos de software ubíquo traz um conjunto de desafios muitas vezes ainda não tratados por abordagens voltadas para o desenvolvimento de projetos de software convencionais (SPÍNOLA *et al.*, 2007).

Segundo OLIVEIRA *et al.* (2004), o conhecimento do domínio pode revelar conceitos, descrições e relações que poderiam ser organizados para evidenciar em cada etapa do desenvolvimento o que precisa ser investigado. Na definição e na verificação de requisitos em particular, esse conhecimento poderia ser utilizado para destacar informações que deveriam ser capturadas nos requisitos do software.

Nesse sentido, o corpo de conhecimento estruturado anteriormente (Capítulo 4) pode ser utilizado para fornecer conceitos e relações que podem ajudar na definição e verificação[10] dos requisitos de ubiquidade de um projeto de software. Entretanto, é importante que essas informações sejam trabalhadas para tornar o seu uso mais efetivo durante o desenvolvimento do software.

Sendo assim, surgiu a idéia de elaborar uma abordagem baseada no uso de *checklist*[11] com objetivo de apresentar informações que permitam direcionar o engenheiro de software na captura e na verificação dos requisitos de ubiquidade

[10] *A verificação de requisitos examina a especificação de requisitos do software de forma a minimizar problemas inseridos na especificação relacionados a questões como ambiguidade, inconsistência ou omissão, detectando-os e corrigindo-os ainda durante a fase de definição dos requisitos.*

[11] *Segundo LAITENBERGER et al. (2001), responder perguntas de um checklist permite direcionar o foco do engenheiro de software para as informações contidas nestas perguntas.*

utilizando o corpo de conhecimento definido em (SPINOLA *et al*. 2009). É esperado que com o uso dos *checklists* organizados em Guias para Definição e Verificação de Requisitos de Ubiquidade seja possível aprimorar a qualidade do documento de requisitos através da redução de defeitos de omissão, ambiguidade, inconsistência, fato incorreto e informação estranha associados aos requisitos de ubiquidade do projeto ao mesmo tempo em que é aumentada a satisfação do engenheiro de software ao realizar estas atividades.

Neste contexto, este capítulo apresentará u-RDV (***u****biquity-****R****equirement* ***D****efinition and* ***V****erification approach*), abordagem de apoio à definição e verificação de requisitos de ubiquidade em projetos de software. Esta tem como escopo as seis características funcionais da computação ubíqua: sensibilidade ao contexto, comportamento adaptável, onipresença de serviço, heterogeneidade de dispositivos, captura de experiência e interoperabilidade espontânea.

Esta definição de escopo, conforme descrito no Capítulo 1, se deu pelo fato de Spínola *et al.* (2007) terem identificado que esta decisão reduziria os riscos da pesquisa uma vez que pesquisas no campo da computação ubíqua têm sido mais focadas nos requisitos funcionais deste tipo de software. Além disso, existem diferentes técnicas tradicionais da engenharia de software de apoio às atividades de verificação de requisitos funcionais já avaliadas experimentalmente que podem ser utilizadas como base para apoiar a definição de abordagens de verificação de requisitos de ubiquidade (SILVA *et. al.*, 2004) (MAFRA e TRAVASSOS, 2006).

Além desta introdução, este capítulo está organizado em mais 6 seções. Na seção 6.2 é apresentada uma visão geral da abordagem e o processo definido para seu uso. Na sequência, seção 6.3, é apresentada a primeira etapa de uso de *u*-RDV: Configuração da Abordagem. Em seguida, na seção 6.4, é apresentado o procedimento realizado para permitir a caracterização de projetos de software ubíquo frente às características e fatores de ubiquidade. Já a seção 6.5 apresenta *UbiCheck*, que é responsável por apoiar a definição dos requisitos de ubiquidade. A seção 6.6 apresenta *UbiVeri*, responsável por apoiar a verificação dos requisitos de ubiquidade. Por fim, a seção 6.7 concluirá este capítulo.

6.2 – Visão Geral da Abordagem

A abordagem proposta nesta pesquisa define um arcabouço composto de um conjunto de facilidades associadas às atividades de definição e verificação de requisitos de ubiquidade em projetos de software. A Figura 6.1 apresenta uma visão geral do arcabouço que contempla:

- as facilidades apoiadas (caixas em negrito da Figura 6.1) pela abordagem, que são:
    - **Caracterização do projeto:** permite definir quais características de ubiquidade e fatores funcionais de ubiquidade (considerando suas respectivas características) deverão ser contempladas no projeto;
    - **Especificação dos Requisitos Funcionais de Ubiquidade:** permite especificar os requisitos de ubiquidade do software com base nas características e fatores funcionais definidos na etapa de caracterização do projeto;
    - **Verificação dos Requisitos de Ubiquidade:** permite avaliar a qualidade dos requisitos especificados frente ao corpo de conhecimento em computação ubíqua e a taxonomia de defeitos definida em (SHULL *et al.,* 2000).
- o conhecimento utilizado para apoiar a realização das facilidades (caixas destacadas na parte superior da figura):
    - **Descrição das Características de Ubiquidade:** fundamentação teórica da computação ubíqua considerando as características de ubiquidade descritas nos Capítulo 2 e 4;
    - **Descrição dos Fatores Funcionais de Ubiquidade:** fundamentação teórica da computação ubíqua considerando os fatores funcionais das características de ubiquidade descritas nos Capítulos 2 e 4;
    - ***UbiCheck*:** abordagem de apoio à definição de requisitos funcionais de ubiquidade apresentada na seção 6.5;
    - ***UbiVeri*:** abordagem de apoio à verificação de requisitos funcionais de ubiquidade apresentada na seção 6.6;
- os principais artefatos gerados (caixas em cinza claro):
    - **Características de Ubiquidade do Projeto:** define o conjunto de características de ubiquidade que deverão estar contempladas no projeto;
    - **Fatores Funcionais de Ubiquidade do Projeto:** define o conjunto de fatores funcionais de ubiquidade que deverão estar contemplados no projeto;
    - **Especificação dos Requisitos de Ubiquidade do Projeto:** define de forma detalhada os requisitos de ubiquidade do software de maneira que possam ser utilizados posteriormente pelas equipes de projeto e desenvolvimento;

- **Lista de Defeitos:** define o conjunto de defeitos que foram identificados durante a revisão da especificação de requisitos de ubiquidade do software.

A Figura 6.1 também destaca o fato de que todas as facilidades da abordagem são baseadas no corpo de conhecimento em computação ubíqua previamente discutido nos Capítulos 2 e 4.

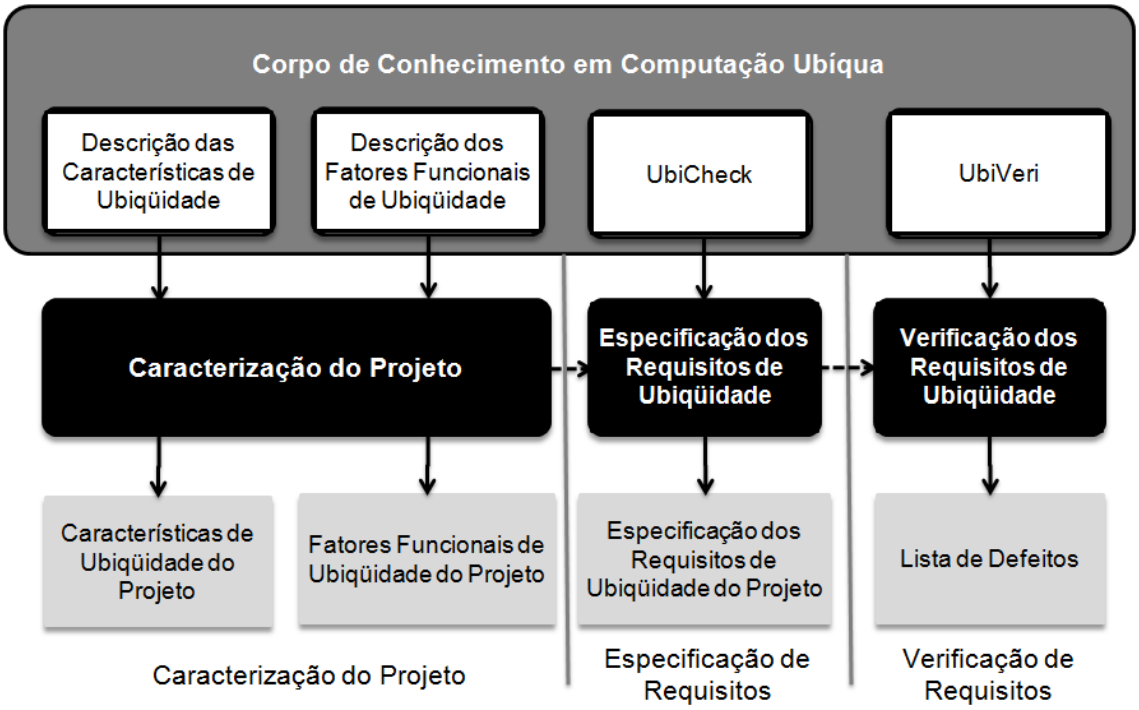


Figura 6.1. Visão geral da abordagem de apoio à definição e verificação de requisitos de ubiquidade em projetos de software.

A partir desta visão inicial da abordagem, foi definido um processo contendo um conjunto de atividades que permitisse definir mais precisamente como estas facilidades (caixas em negrito da Figura 6.1) poderiam ser apoiadas em termos de técnicas ou metodologias. Desta forma, este processo define cada uma das atividades que devem ser executadas assim como os responsáveis pela sua execução.

O processo visto na Figura 6.2 apresenta as atividades definidas no contexto desta proposta e as macro-atividades que as envolvem (configuração da abordagem, caracterização do projeto de software ubíquo, definição de requisitos de ubiquidade e verificação dos requisitos de ubiquidade).

Algumas das atividades definidas são comuns no desenvolvimento de projetos de software tradicional e a engenharia de requisitos já disponibiliza um amplo conhecimento se considerarmos este tipo de projeto. Entretanto, neste trabalho estamos lidando com projetos de software ubíquo, e estes possuem características

diferenciadas que trazem novas necessidades no contexto da engenharia de requisitos. Assim, a contribuição desta proposta está na adaptação de atividades já conhecidas para um determinado domínio de aplicação a partir de um corpo de conhecimento definido através de estudos experimentais.

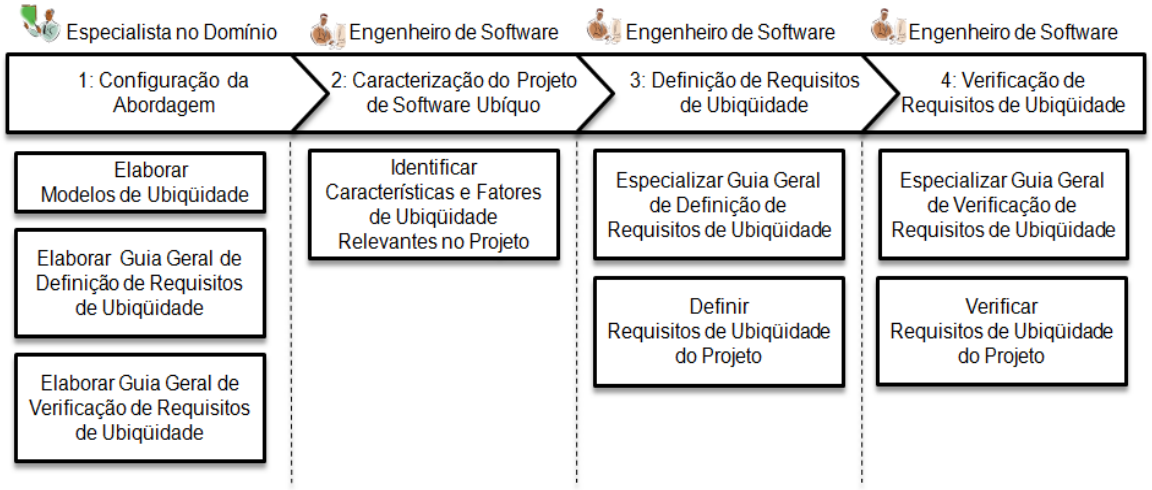


Figura 6.2. Processo para definição e verificação de requisitos de ubiquidade.

Para utilizar a abordagem, é preciso inicialmente ter o corpo de conhecimento organizado. A abordagem proposta já vem previamente configurada considerando o corpo de conhecimento apresentado no Capítulo 4 desta Tese. Assim, caso não seja necessário fazer qualquer ajuste, a abordagem já se encontra pronta para uso e serão utilizadas as etapas 2, 3 e 4 da Figura 6.2.

Entretanto, caso seja necessário fazer algum ajuste, foi definida a etapa 1: Configuração da Abordagem. Sendo assim, uma característica importante das atividades inseridas nesta etapa é que o resultado de sua execução já está disponível na abordagem proposta. Isto significa que elas só precisam ser executadas novamente se houver alguma mudança nas características de ubiquidade (evolução do corpo de conhecimento) ou se houver necessidade de ajustes nos guias gerais de definição e verificação identificados por parte da equipe envolvida no projeto.

Na etapa Configuração da Abordagem as características de ubiquidade são formalizadas em modelos e depois transformadas em guias com perguntas que posteriormente serão utilizadas para auxiliar na definição e na verificação de requisitos de ubiquidade. Nesse sentido, um especialista no domínio da computação ubíqua realiza as seguintes atividades:

- **Elaborar Modelos Conceituais:** nesta atividade as características são formalizadas em modelos, conforme será descrito na seção 6.3;
- **Elaborar Guia Geral de Definição de Requisitos de Ubiquidade:** nesta atividade os modelos de ubiquidade são processados e perguntas sobre

informações que precisam ser capturadas durante a definição de requisitos são elaboradas (descrito na seção 6.5);

- **Elaborar Guia Geral de Verificação de Requisitos de Ubiquidade:** nesta atividade os modelos de ubiquidade são processados e transformados em perguntas que serão utilizadas para verificar se as informações que precisavam ser capturadas durante a definição de requisitos foram descritas no documento de requisitos do software (descrito na seção 6.6).

A próxima etapa a ser realizada é a **Caracterização do Projeto de Software Ubíquo**, onde se tem a seguinte atividade:

- **Identificar Características e Fatores de Ubiquidade Relevantes no Projeto:** nesta atividade, o projeto em desenvolvimento é caracterizado através de questionários que mapeiam os interesses do desenvolvedor com os fatores e características de ubiquidade que podem ser relevantes (descrito na seção 6.4).

Na etapa de **Definição de Requisitos de Ubiquidade** o desenvolvedor utiliza o guia para definição de requisitos e preenche o documento de requisitos de software com as informações capturadas. As seguintes atividades são realizadas (descritas na seção 6.5):

- **Especializar Guia Geral de Definição de Requisitos de Ubiquidade:** nesta atividade o engenheiro de software especializa o guia geral, a partir da caracterização do projeto, para definição de requisitos de ubiquidade de forma que as necessidades do projeto sejam contempladas;
- **Definir Requisitos de Ubiquidade do Projeto:** nesta atividade os requisitos do projeto relacionados às características de ubiquidade são capturados e o documento de requisitos de software é preenchido com essas informações.

Na etapa seguinte é realizada a **Verificação dos Requisitos de Ubiquidade**. Nela, o documento de requisitos de software é inspecionado com intuito de assegurar que as informações que deveriam ser capturadas foram documentadas adequadamente. As seguintes atividades são realizadas (descritas na seção 6.6):

- **Especializar Guia Geral de Verificação de Requisitos de Ubiquidade:** nesta atividade o Guia Geral é especializado, a partir da caracterização do projeto, para contemplar as necessidades do projeto;

- **Verificar Requisitos de Ubiquidade do Projeto:** nesta atividade o documento de requisitos de software elaborado na etapa anterior é verificado e os defeitos encontrados são encaminhados para ajuste.

Considerando o processo apresentado na Figura 6.2, temos a descrição apresentada na Tabela 6.1.

Tabela 6.1. Descrição do processo.

Configuração da Abordagem	
Atividade:	**Elaborar Modelos de Ubiquidade**
Pré-atividade:	--
Responsável:	Especialista no domínio
Participante:	--
Artefato Requerido:	Corpo de conhecimento em computação ubíqua
Artefato Produzido:	Modelos de ubiquidade
Pós-atividade:	Elaborar Guia Geral de Definição de Requisitos de Ubiquidade, Elaborar Guia Geral de Verificação de Requisitos de Ubiquidade
Ferramenta:	
Atividade:	**Elaborar Guia Geral de Definição de Requisitos de Ubiquidade**
Pré-atividade:	Elaborar Modelos de Ubiquidade
Responsável:	Especialista no domínio
Participante:	--
Artefato Requerido:	Corpo de conhecimento em computação ubíqua e modelo de ubiquidade
Artefato Produzido:	Guia Geral de Definição de Requisitos de Ubiquidade
Pós-atividade:	Identificar Características e Fatores de Ubiquidade Relevantes no Projeto
Ferramenta:	Editor de texto, visualizador dos modelos
Atividade:	**Elaborar Guia Geral de Verificação de Requisitos de Ubiquidade**
Pré-atividade:	Elaborar Modelos de Ubiquidade
Responsável:	Especialista no domínio
Participante:	--
Artefato Requerido:	Corpo de conhecimento em computação ubíqua e modelo de ubiquidade
Artefato Produzido:	Guia Geral de Verificação de Requisitos de Ubiquidade
Pós-atividade:	Identificar Características e Fatores de Ubiquidade Relevantes no Projeto
Ferramenta:	Editor de texto, visualizador dos modelos
Caracterização do Projeto de Software	
Atividade:	**Identificar Características e Fatores de Ubiquidade Relevantes no Projeto**
Pré-atividade:	--
Responsável:	Engenheiro de Software
Participante:	Usuário
Artefato Requerido:	Formulário de Caracterização
Artefato Produzido:	Formulário de Caracterização atualizado
Pós-atividade:	Especializar Guia Geral de Definição de Requisitos de Ubiquidade, Especializar Guia Geral de Verificação de Requisitos de Ubiquidade
Ferramenta:	Editor de planilhas
Definição de Requisitos de Ubiquidade	
Atividade:	**Especializar Guia Geral de Definição de Requisitos de Ubiquidade**
Pré-atividade:	Identificar Características e Fatores de Ubiquidade Relevantes no Projeto
Responsável:	Engenheiro de Software

Participante:	--
Artefato Requerido:	Formulário de Caracterização atualizado, Guia Geral de Definição de Requisitos de Ubiquidade
Artefato Produzido:	Guia Especializado de Definição de Requisitos de Ubiquidade
Pós-atividade:	Definir Requisitos de Ubiquidade do Projeto
Ferramenta:	Editor de texto e planilha
Atividade:	**Definir Requisitos de Ubiquidade do Projeto**
Pré-atividade:	Especializar Guia Geral de Definição de Requisitos de Ubiquidade
Responsável:	Engenheiro de Software
Participante:	Usuário
Artefato Requerido:	Guia Especializado de Definição de Requisitos de Ubiquidade
Artefato Produzido:	Especificação de Requisitos do Projeto
Pós-atividade:	--
Ferramenta:	Editor de texto
Verificação dos Requisitos de Ubiquidade	
Atividade:	**Especializar Guia Geral de Verificação de Requisitos de Ubiquidade**
Pré-atividade:	Identificar Características e Fatores de Ubiquidade Relevantes no Projeto
Responsável:	Engenheiro de Software
Participante:	--
Artefato Requerido:	Formulário de Caracterização atualizado, Guia Geral de Verificação de Requisitos de Ubiquidade
Artefato Produzido:	Guia Especializado de Verificação de Requisitos de Ubiquidade
Pós-atividade:	Verificar Requisitos de Ubiquidade do Projeto
Ferramenta:	Editor de texto e planilha
Atividade:	**Verificar Requisitos de Ubiquidade do Projeto**
Pré-atividade:	Especializar Guia Geral de Verifica de Requisitos de Ubiquidade
Responsável:	Engenheiro de Software
Participante:	--
Artefato Requerido:	Guia Especializado de Verificação de Requisitos de Ubiquidade
Artefato Produzido:	Formulário de Discrepâncias
Pós-atividade:	--
Ferramenta:	Editor de texto e planilha

As próximas quatro seções apresentam cada uma das etapas que compõem a abordagem proposta e como ela irá prover um conjunto de facilidades que não estão presentes em técnicas tradicionais de engenharia de software para apoiar a construção de projetos de software ubíquo no contexto de atividades da engenharia de requisitos.

6.3 – Configuração da Abordagem

Conforme visto na seção anterior, a etapa Configuração da Abordagem é composta de três atividades: Elaborar Modelos de Ubiquidade, Elaborar Guia Geral de Definição de Requisitos de Ubiquidade, Elaborar Guia Geral de Verificação de Requisitos de Ubiquidade. Nesta seção será apresentada a elaboração dos modelos de ubiquidade. As outras duas atividades serão descritas, respectivamente, nas seções 6.5 e 6.6.

6.3.1. Elaborar Modelos de Ubiquidade

Os modelos de ubiquidade foram elaborados com o propósito de apresentar os conceitos e relações de cada característica de ubiquidade em um formato mais estruturado e gráfico. Dessa forma, acredita-se que fique mais fácil para um pesquisador entender as informações relacionadas ao conjunto de características de ubiquidade apresentados nos capítulos 2 e 4.

A elaboração dos modelos ocorreu no contexto de uma dissertação de mestrado. Nesta seção será descrito brevemente como esta atividade é realizada. Maiores detalhes podem ser encontrados em (PINTO, 2009).

A primeira etapa para construir esses modelos foi escolher a linguagem que seria utilizada. PINTO (2009) optou por estender a notação UML (OMG, 2009) para elaborar os modelos de ubiquidade. Para isso, foi desenvolvido um meta-modelo[12] de forma que o diagrama de classes da UML pudesse contemplar os conceitos e relações importantes dos fatores de cada característica de ubiquidade.

Para elaborar o meta-modelo, visualizado na Figura 6.3, PINTO (2009) considerou o seguinte significado para a notação utilizada nos modelos de ubiquidade:

- **Classe com estereótipo *Entidade*:** representa um conceito relevante para uma característica de software ubíquo.
- **Classe com estereótipo *Serviço*:** representa uma funcionalidade associada a uma entidade.
- **Generalização**: é um relacionamento entre duas entidades distintas que representa a herança entre elas (BOOCH, 1994). Neste tipo de relacionamento, há uma entidade denominada pai e outra denominada filha. A filha herda as características do pai.
- **Agregação**: é um relacionamento entre duas entidades distintas que representa que uma é parte da outra (HARMON e WATSON, 1997). Ela também pode representar que uma entidade pertence a outra, como se fosse um atributo.
- **Associação**: é um relacionamento entre um serviço e uma entidade que representa que um atua sobre o outro.
- **Dependência**: é um relacionamento que representa que um serviço precisa consumir uma entidade para ser executado.

[12] *Segundo SODERSTROM (2002), por prover uma ilustração dos conceitos básicos e de seus relacionamentos, um meta-modelo pode facilitar o entendimento de um modelo, mesmo para leitores com pouca experiência. YANGFAN et al. (2005) acrescenta que um meta-modelo tem o intuito de nortear a construção de modelos, especificando o que pode e o que não pode ser modelado.*

A partir do meta-modelo é possível verificar as seguintes restrições:

1. um serviço pode depender de zero ou mais (0..*) entidades;
2. uma entidade pode ser dependência de zero ou mais (0..*) serviços;
3. um serviço deve estar associado a uma ou mais entidades (1..*);
4. uma entidade pode ser associada com zero ou mais (0..*) serviços;
5. uma entidade pode opcionalmente (0..1) ter um pai e pode ter zero ou mais filhos (0..*), nestes casos, tanto o pai quanto os filhos devem ser também entidades; e
6. uma entidade pode ter zero ou mais (0..*) agregações com outras entidades.

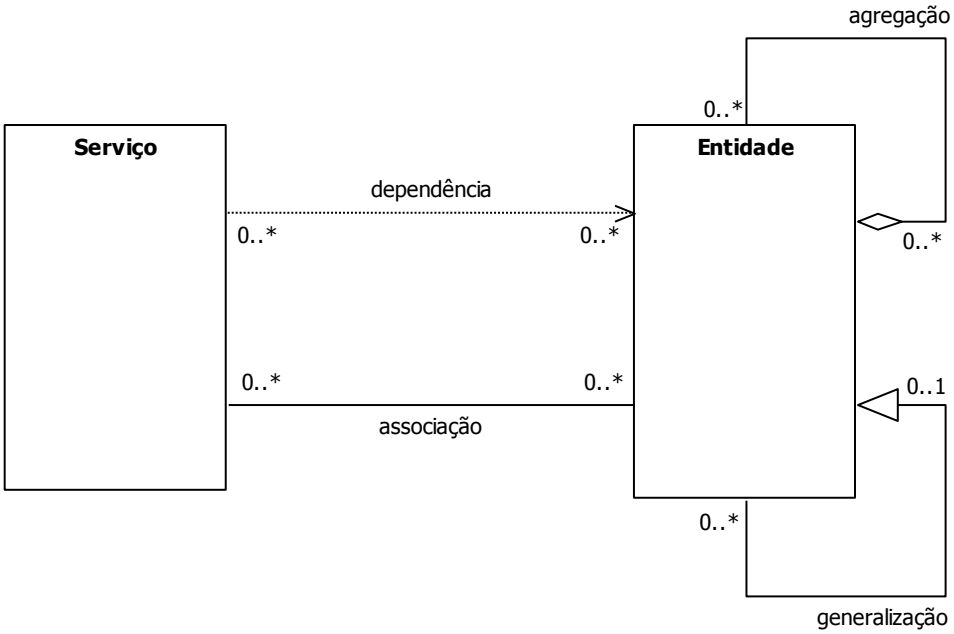


Figura 6.3. Meta-modelo das características de software ubíquo

Com o meta-modelo elaborado, o próximo passo foi construir os modelos de ubiquidade, o que foi feito com base nas informações dos fatores sobre os conceitos e relações importantes de cada característica de ubiquidade.

Na modelagem também foi convencionado que todo elemento deveria fazer referência aos fatores que o originaram. Nesse sentido, após o nome do elemento foram inseridos os códigos desses fatores. Essa é uma informação importante para que os artefatos, que venham a ser construídos a partir destes modelos, possam apontar para os fatores de ubiquidade que estão relacionados (rastreabilidade).

Para exemplificar como deve ser elaborado um modelo de ubiquidade, tomaremos os seguintes fatores de ubiquidade:

Fator SC01: O sistema deve considerar o nível de abstração (atributo) dos relacionamentos (atributo) das informações de contexto (entidade).

Fator SC02: O sistema deve gerenciar (serviço) os relacionamentos (atributo) das informações de contexto (entidade).

Fator SC03: O sistema deve categorizar (serviço) informações de contexto (entidade) de acordo com sua fonte de dados (entidade).

- *categorizar* depende de *fonte de dados*.

Fator SC04: O sistema deve compartilhar (serviço) informações de contexto (entidade). O compartilhamento pode ser entre usuários ou entre dispositivos, podendo ser utilizado, por exemplo, para efetuar configurações no software.

Fator SC05: O sistema deve interpretar (serviço) informações de contexto (entidade) de acordo com a sua (entidade) semântica (atributo).

- *interpretar* depende de *semântica*.
- "sua semântica" é uma forma de escrever "semântica (atributo) da informação de contexto (entidade)"

Fator SC06: O sistema deve controlar (serviço) sensores (entidade).

Fator SC07: O sistema deve considerar informações de contexto (entidade) do tipo informação de sistema (entidade). Essas informações são aquelas relacionadas aos dispositivos, infra-estrutura do ambiente (informações de hardware utilizado (CPU, memória, tamanho da tela, energia), largura de banda disponível e dispositivos acessíveis) e outras aplicações que estão relacionadas para compor o sistema.

- *informação de sistema* é um tipo de *informação de contexto*

A Figura 6.4 ilustra o modelo construído. É possível perceber que cada conceito destacado nos fatores foi modelado em uma *entidade* ou *serviço*. Adicionalmente, os relacionamentos entre esses elementos se tornam mais explícitos quando observados do ponto de vista do modelo.

Por exemplo, no modelo apresentado, *informação de contexto* possuí uma agregação com *relacionamento* porque no *fator SC01*, *relacionamento* é atributo de *informação de contexto*. De forma semelhante, *interpretar* está associado a *informação de contexto* e depende da *semântica da informação de contexto* porque essas relações foram explicitadas no *Fator SC05.*

Os modelos completos de cada uma das características estão disponíveis no Apêndice D. Cabe destacar que, no escopo do trabalho (PINTO, 2009), foram modeladas apenas as características funcionais apresentadas no Capítulo 4: Captura

da Experiência, Comportamento Adaptável, Heterogeneidade de Dispositivos, Interoperabilidade Espontânea, Onipresença do Serviço e Sensibilidade ao Contexto.

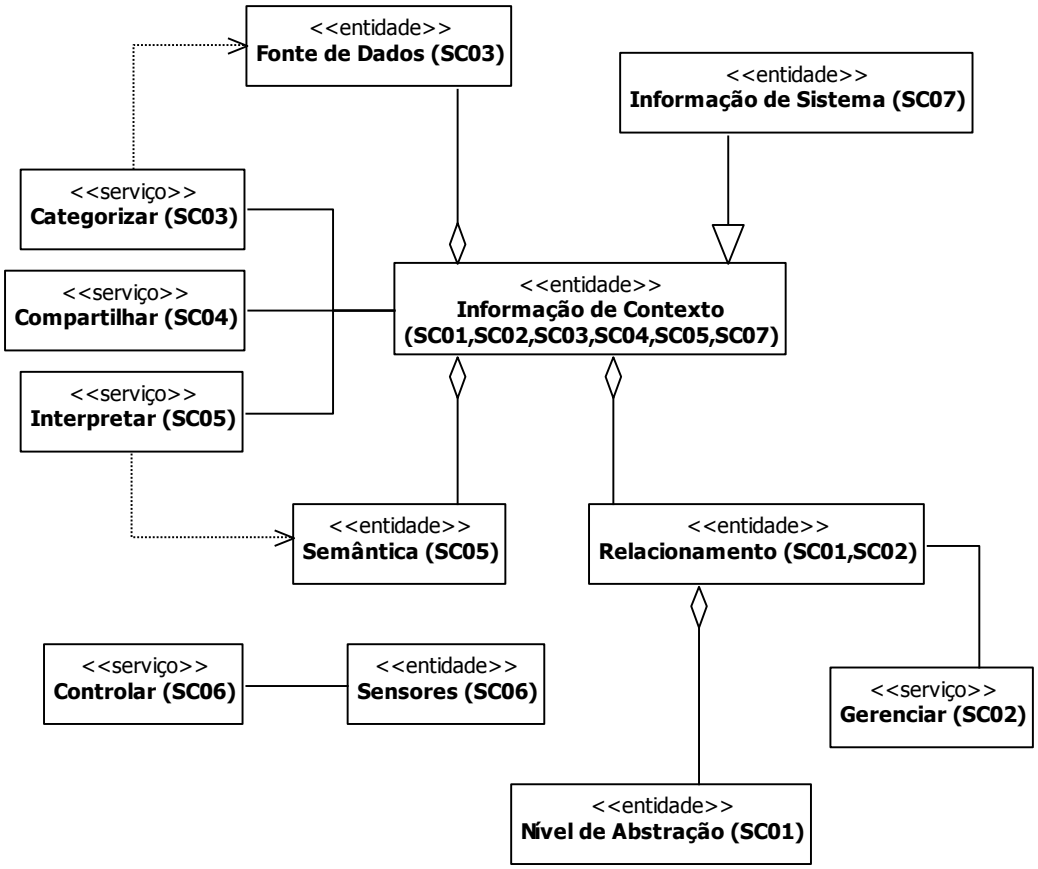


Figura 6.4. Exemplo de modelo de ubiquidade de parte da característica *Sensibilidade ao Contexto.*

6.4 – Caracterização do Projeto de Software Ubíquo

Foi visto no capítulo 4 que para a computação ubíqua se fazer presente, é preciso que os sistemas que compõem o ambiente contemplem (total ou parcialmente) características funcionais como sensibilidade ao contexto, comportamento adaptável, onipresença de serviço, heterogeneidade de dispositivos, captura de experiência e interoperabilidade espontânea. É importante notar que nem todas as características e seus respectivos fatores necessitam estar presentes. Desta forma, utilizar **todo** o corpo de conhecimento estruturado para apoiar o desenvolvimento de projetos de software ubíquo sem levar em consideração as restrições de cada projeto pode **reduzir** a efetividade do uso deste conhecimento. Assim, é importante caracterizar o projeto frente às características e fatores de ubiquidade para "filtrar" o corpo de conhecimento às necessidades do projeto.

Para lidar com isto, a abordagem proposta nesta tese define um formulário de caracterização baseado no formulário apresentado no Capítulo 3.

6.4.1. Identificar Características e Fatores de Ubiquidade Relevantes ao Projeto

Para apoiar a caracterização de projetos de software ubíquo, a abordagem proposta permite a identificação:

- das características de ubiquidade que deverão estar presentes no projeto de software;
- dos fatores funcionais de ubiquidade que deverão estar presentes no projeto de software.

Para isto, foi definido um formulário de caracterização dividido em duas etapas. Na primeira etapa, Identificar características de ubiquidade, o formulário auxilia o engenheiro de software a identificar as características de ubiquidade que devem ser consideradas no projeto de software. Para isto, foi definido um conjunto de perguntas para cada característica de ubiquidade.

A Figura 6.5 apresenta um fragmento do formulário para guiar a execução da entrevista para identificação das características de ubiquidade do projeto considerando as características *sensibilidade ao contexto* e *comportamento adaptável* (a versão completa deste questionário pode ser vista no Apêndice E). À medida que o analista de requisitos obtém as respostas dos usuários, ele registra se uma dada característica é ou não necessária. É importante notar que o formulário contém, para cada característica de ubiquidade, uma explicação inicial sobre a característica em questão e um conjunto de perguntas que deverão ser feitas aos usuários.

Uma vez identificadas as características de ubiquidade que serão contempladas no projeto, deve-se descrever em termos funcionais como elas estarão presentes no software. Esta pode ser considerada uma tarefa complexa uma vez que estas funcionalidades podem não ser facilmente percebidas por possuírem um domínio de conhecimento muito específico, o que dificulta sua identificação ao se construir projetos de software ubíquos.

Caracterização do Projeto - Etapa 1

1	O objetivo desta etapa é Identificar junto ao cliente quais características de ubiqüidade estarão presentes na aplicação. Para isto, o conjunto de questionamentos listados na seqüência junto com suas definições pode ser utilizado. A sugestão é que a definição de cada característica e seu cenário de exemplo sejam utilizados apenas pelo analista para um melhor entendimento sobre as características. Ao cliente devem ser efetuados apenas os questionamentos.

Característica: Sensibilidade ao Contexto		
Descrição:	Capacidade de coletar informações sobre o ambiente onde o software está sendo utilizado.	
Exemplo:	Um sistema para controle de temperatura de um frigorífico deve estar constantemente monitorando a temperatura para manter o ambiente no estado ideal para manutenção dos produtos.	
Questionário:		
SC1:	O software deve ser capaz de monitorar alguma entidade externa ao sistema?	
SC2:	O software deve ser capaz de capturar informações do ambiente (por exemplo: temperatura, taxa de transferência) para seu funcionamento?	
Característica: Comportamento Adaptável		
Descrição:	Capacidade de, dinamicamente, adaptar os serviços disponíveis ao ambiente onde está sendo utilizado dentro de suas limitações.	
Exemplo:	Imaginemos um software para gerência de vídeo conferência. Ao identificar a redução de largura de banda a ponto de prejudicar a transmissão do áudio e vídeo na qualidade atual, o software deve reduzir a qualidade do áudio e do vídeo de forma a permitir que a comunicação entre os participantes continue fluindo normalmente sem interrupções.	
Questionário:		
CA1:	O software deve ser capaz de prover alguma modificação (por exemplo: reduzir qualidade do vídeo, reduzir a temperatura do ambiente interno com base na temperatura externa) em seu funcionamento com base nas informações capturadas do ambiente?	
CA2:	O software deve ser capaz de controlar variáveis externas ao sistema e executar ações com base nos valores destas variáveis (por exemplo: temperatura externa > 30º diminuir temperatura interna para 24º)?	

Figura 6.5. Fragmento do formulário para apoiar a identificação das características de ubiquidade que estarão contempladas no projeto de software.

Para lidar com esta questão, foi definida a segunda etapa do questionário de caracterização, identificar requisitos funcionais de ubiquidade. Nesta etapa é feita a definição de quais fatores funcionais serão utilizados para apoiar a definição dos requisitos funcionais. Isto é realizado em dois momentos. O primeiro é resultado da identificação de quais características de ubiquidade estarão presentes no projeto. Ao se identificar quais características serão contempladas, têm-se também o conjunto de fatores funcionais que poderão estar presentes no software (chamaremos esta lista de fatores funcionais candidatos).

A partir da lista de fatores funcionais candidatos, é preciso definir quais fatores estarão contemplados no software para só então se poder explicitar os requisitos do projeto. Assim, na segunda etapa de preenchimento do questionário de caracterização, cada fator funcional candidato deve estar associado a pelo menos uma pergunta. Caso a resposta seja positiva, isto indica que aquele fator candidato estará contemplado no projeto e servirá de base para a definição dos requisitos funcionais.

A Figura 6.6 apresenta um fragmento do questionário para a característica de ubiquidade *sensibilidade ao contexto* (uma versão completa deste questionário pode ser vista no Apêndice E).

Ao término da execução desta atividade, tem-se o conjunto de características e fatores funcionais de ubiquidade que estarão presentes no projeto de software. Este corpo de conhecimento especializado para as necessidades do projeto será utilizado nas demais atividades da abordagem para apoiar a definição e a revisão dos requisitos de ubiquidade do projeto.

6.5 – Definição de Requisitos de Ubiquidade - *UbiCheck*

Para apoiar as atividades associadas à definição dos requisitos de ubiquidade foi definido *UbiCheck*[13] (PINTO *et al.*, 2009). A Figura 6.7[14] apresenta o encadeamento das atividades de *UbiCheck*, indicando: (1) os responsáveis pela execução de cada etapa; (2) as etapas consideradas; (3) as atividades executadas em cada etapa; e (4) os artefatos consumidos e gerados no processo.

Como pode ser observado, *UbiCheck* possui duas etapas. A primeira está associada à **Configuração de *UbiCheck***, ou seja, a preparação dos artefatos necessários para que ela possa ser utilizada. Já a segunda etapa está associada à **Definição de Requisitos de Ubiquidade** onde, para cada projeto, o desenvolvedor utiliza os artefatos gerados anteriormente para auxiliá-lo.

As próximas subseções explicam as atividades de *UbiCheck*: Elaborar Guia Geral de Definição de Requisitos de Ubiquidade; Especializar Guia Geral de Definição de Requisitos de Ubiquidade; e Definir Requisitos de Ubiquidade.

---

[13] O desenvolvimento de UbiCheck ocorreu no contexto de uma dissertação de mestrado. Nesta seção será descrita a abordagem considerando seus principais elementos. Maiores detalhes podem ser encontrados em (PINTO, 2009).

[14] Esta figura representa um recorte da Figura 6.2 com foco nas atividades que compõem UbiCheck.

Caracterização do Projeto - Etapa 2 - SC

2	O objetivo desta etapa é identificar junto ao cliente como a característica sensibilidade ao contexto estará presente no projeto. Para isto, identifique junto ao cliente se os itens de 1 a 21 estarão ou não presentes no projeto.

Característica:	**Sensibilidade ao Contexto**	
Questionário:		
SC1	O software deve ser capaz de identificar o usuário que o está manipulando?	
SC2	O software deve ser capaz de identificar o local onde está sendo utilizado?	
SC3	O software deve ser capaz de identificar a atividade que está sendo realizada pelo usuário?	
SC4	O software deve ser capaz de organizar as informações coletadas considerando o momento (horário) em que elas foram coletadas?	
SC5	O software deve ser capaz de capturar informações de contexto de sistema (uso de memório, cpu, tamanho de tela, energia, largura de banda disponível)?	
SC6	O software deve ser capaz de capturar informações de contexto de infra-estrutura (estado de funcionamento dos sensores)?	
SC7	O software deve ser capaz de capturar informações de contexto físico (temperatura, condições de iluminação, barulho)?	
SC8	O software deve ser capaz de capturar informações de contexto de usuário (perfil do usuário, preferências, ação)?	
SC9	O software deve ser capaz de fazer uso de sensores para capturar as informações de contexto?	
SC10	O software deve ser capaz de manter as informações de contexto capturadas?	
SC11	O software deve ser capaz de filtrar as informações de contexto capturadas para armazenar apenas aquelas consideradas úteis?	
SC12	O software deve ser capaz de organizar as informações capturadas considerando a fonte de dados?	
SC13	O software deve ser capaz de manter as relações existentes entre as diferentes informações de contexto?	
SC14	O software deve ser capaz de integrar as informações capturadas mesmo que elas tenham sido geradas em formatos diferentes?	
SC15	O software deve ser capaz de considerar as informações identidade, tempo e localização ao armazenar cada informação de contexto?	
SC16	O software deve ser capaz de considerar a semântica da informação de contexto ao armazená-la?	
SC17	O software deve ser capaz de derivar novas informações a partir das informações capturadas do contexto?	
SC18	O software deve ser capaz de manter um rastro das transformações efetudas nas informações?	
SC19	O software deve ser capaz de organizar as informações de contexto considerando as preferências do usuário?	
SC20	O software deve ser capaz de considerar a semântica ao interpretar as informações?	
SC21	O software deve ser capaz de compartilhar as informações de contexto capturadas com outros sistemas?	

2	Etapa de caracterização finalizada para a característica sensibilidade ao contexto.

Figura 6.6. Fragmento do *checklist* para apoiar a identificação dos fatores funcionais que deverão estar contempladas no projeto de software.

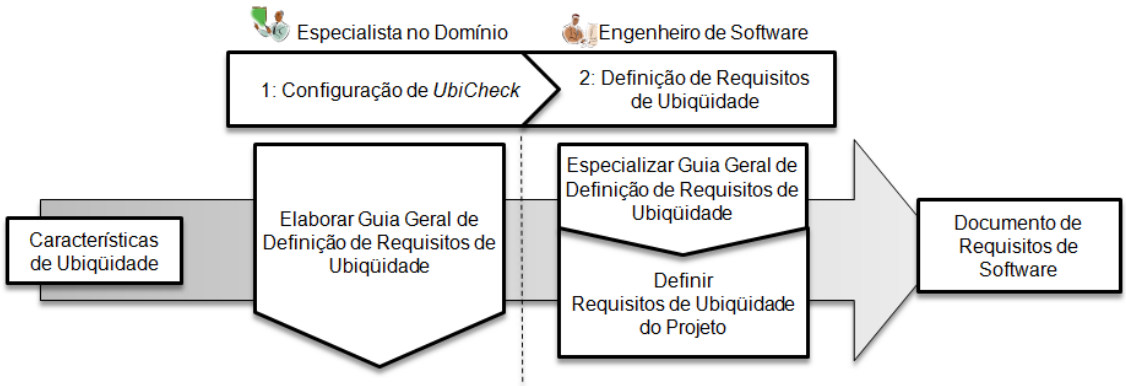


Figura 6.7. Visão geral de *UbiCheck*

6.5.1. Elaborar Guia Geral de Definição de Requisitos de Ubiquidade

Esta atividade tem o objetivo de elaborar o *Guia Geral de Definição de Requisitos de Ubiquidade.* Ele é um conjunto de perguntas sobre informações presentes nos fatores de ubiquidade que seriam interessantes de serem capturadas durante a definição de requisitos de ubiquidade. Dessa forma, trata-se de uma forma mais direta de utilizar o conhecimento atrelado às características de ubiquidade para apoiar a definição de requisitos de ubiquidade.

Conforme ilustra a Figura 6.8, o *Guia Geral de Definição de Requisitos de Ubiquidade* é elaborado a partir dos modelos das características de ubiquidade. Nesse sentido, duas sub-atividades são realizadas e descritas a seguir (durante a explicação, serão utilizados exemplos baseados no trecho do modelo da característica *Sensibilidade ao Contexto* apresentado na Figura 6.9).

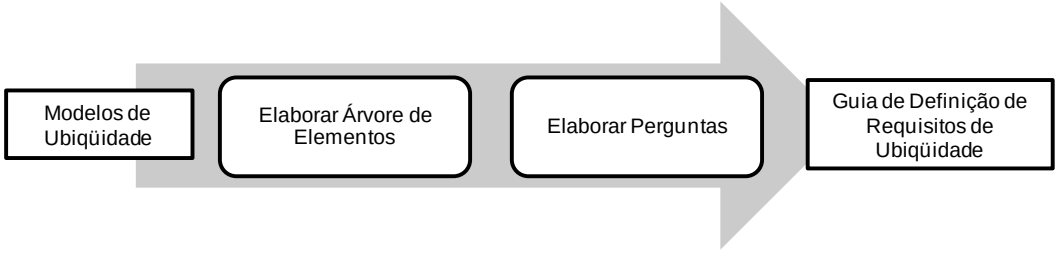


Figura 6.8. Sub-atividades para Elaborar o Guia para Definição de Requisitos de Ubiquidade.

6.5.1.1 Elaborar Árvore de Elementos

Um modelo de ubiquidade retrata uma visão onde são apresentados elementos (entidades e serviços) e relacionamentos (agregação, associação, dependência e generalização) de uma característica de ubiquidade. Seria interessante que esses

modelos apresentassem também informações sobre a importância que determinado elemento tem para uma característica.

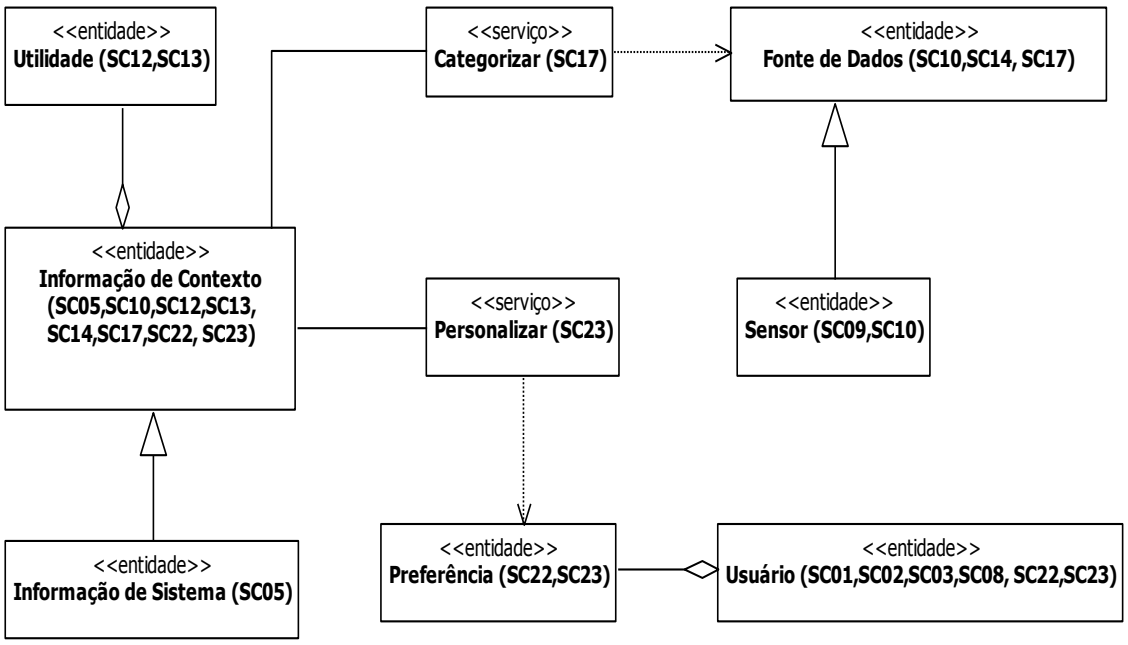


Figura 6.9. Trecho do Modelo de Sensibilidade ao Contexto.

Nesse sentido, Pinto *et al.* (2009) elaboraram uma nova visão do modelo, onde os elementos foram representados em árvores. Nestas árvores, convencionou-se que os elementos inseridos mais próximos da raiz seriam considerados mais importantes que os inseridos mais próximos das folhas. Essa forma de organizar os elementos foi chamada **árvore de elementos**.

Para criar a árvore de elementos são necessários três passos. O primeiro passo é **Analisar os Relacionamentos do Modelo de Ubiqüidade**, que tem o objetivo de estabelecer entre cada par de elementos conectados qual é o mais importante, ou seja, o que precede o outro e será incluído primeiro na árvore. Nesse sentido, Pinto *et al.* (2009) definiram critérios para indicar a precedência entre os elementos do modelo adaptados do trabalho de LIMA (2005). Este descreve um conjunto de heurísticas para identificar a ordem de integração das classes em um projeto de software visando reduzir o esforço dos testes de integração. Para tal, os critérios identificam primeiro as classes menos acopladas no modelo. Como nesse trabalho estamos interessados em encontrar as classes mais importantes, os critérios foram invertidos e convencionou-se que os elementos mais importantes eram aqueles mais acoplados no modelo.

Esses critérios avaliam o relacionamento que conecta cada par de elementos, conforme descrito a seguir:

1. **Critério para relacionamentos de generalização:** um relacionamento de generalização representa uma relação de herança entre dois elementos. Dessa forma, um elemento filho herda características de um elemento pai (adaptado de BOOCH, 1994). Neste caso, convencionou-se que o elemento pai é mais importante, portanto, precede o elemento filho.
2. **Critério para relacionamentos de agregação**: um relacionamento de agregação representa que um elemento contém outro (adaptado de HARMON e WATSON, 1997). Neste caso, convencionou-se que o elemento que contém (todo) é mais importante, portanto, precede o elemento por ele contido (parte).
3. **Critério para relacionamentos de dependência**: um relacionamento de dependência representa que um elemento (consumidor) depende de outro para prover um serviço (fornecedor). Neste caso, convencionou-se que o elemento fornecedor é mais importante, portanto, precede o elemento consumidor.
4. **Critério para relacionamentos do tipo associação**: um relacionamento de associação representa a conexão entre dois elementos e no contexto deste trabalho, representa também que um serviço age sobre uma entidade. Neste caso, convencionou-se que a entidade é mais importante que o serviço, portando, precede o serviço.

A Figura 6.9 destaca um trecho do modelo da característica *Sensibilidade ao Contexto,* o qual será utilizado para exemplificar os passos apresentados. O resultado da aplicação dos critérios de precedência neste modelo é:

De acordo com o critério 1, *Informação de Contexto* precede *Informação do Sistema*.
De acordo com o critério 1, *Fonte de Dados* precede *Sensor*.
De acordo com o critério 2, *Informação de Contexto* precede *Utilidade*
De acordo com o critério 2, *Usuário* precede *Preferência*
De acordo com o critério 3, *Categorizar* precede *Fonte de Dados*
De acordo com o critério 3, *Personalizar* precede *Preferência*
De acordo com o critério 4, *Informação de Contexto* precede *Categorizar*
De acordo com o critério 4, *Informação de Contexto* precede *Personalizar*

Com a precedência entre os elementos definida é possível listá-los em seqüência, conforme exemplificado a seguir:

Seqüência 1: Informação de Contexto -> Informação do Sistema.
Seqüência 2: Informação de Contexto -> Utilidade
Seqüência 3: Informação de Contexto -> Categorizar -> Fonte de Dados -> Sensor
Seqüência 4: informação de Contexto -> Personalizar -> Preferência
Seqüência 5: Usuário -> Preferência

O passo seguinte é **Preencher a Árvore de Elementos**. Pinto *et al.* (2009) definiram que a raiz da árvore é a característica de ubiqüidade que está sendo tratada, no caso do exemplo, *Sensibilidade ao Contexto*. Em seguida, cada uma das seqüências elaboradas anteriormente é utilizada para descrever um caminho da raiz até uma folha da árvore. Como todos os caminhos da raiz até as folhas foram descritos nestas seqüências, é possível construir a árvore de elementos, conforme mostra a Figura 6.10.

Cabe destacar que, durante a construção da árvore, alguns elementos são mapeados para mais de um nó da árvore, como foi o caso do elemento *Preferência*, que aparece como filho do elemento *Personalizar* (seqüência 4) e do elemento Usuário (seqüência 5). Neste caso, eles são chamados *nós similares.*

Adicionalmente, conforme mostrou a Figura 6.10, para cada nó diferente da raiz, dois atributos foram acrescentados:

- **Relacionamento (Rel.):** o tipo de relacionamento entre esse nó e o seu pai. Essa informação é extraída do modelo e pode ser: *Agregação, Associação, Dependência ou Generalização.* Adicionalmente, para os nós inseridos no primeiro nível da árvore (elementos ligados direto a raiz) foi convencionado que o relacionamento deles seria: *Direto*.
- **Fatores (Fat):** o código dos fatores que originaram o respectivo nó, para que possa ser realizada a rastreabilidade entre os nós e os fatores de ubiqüidade.

Assim, ao final desta atividade, tem-se a árvore de elementos elaborada conforme pode ser visto na **Figura 6.10.** Como o espaço consumido para representar a árvore de elementos cresce bastante conforme se aumenta o número de nós, optou-se por representar a árvore de elementos em tabelas, fazendo uso da notação em

barras e ocultando o nó que representa a raiz (PINTO *et al.* (2009)). A Tabela 6.2 mostra a mesma árvore apresentada na Figura 6.10 neste novo formato.

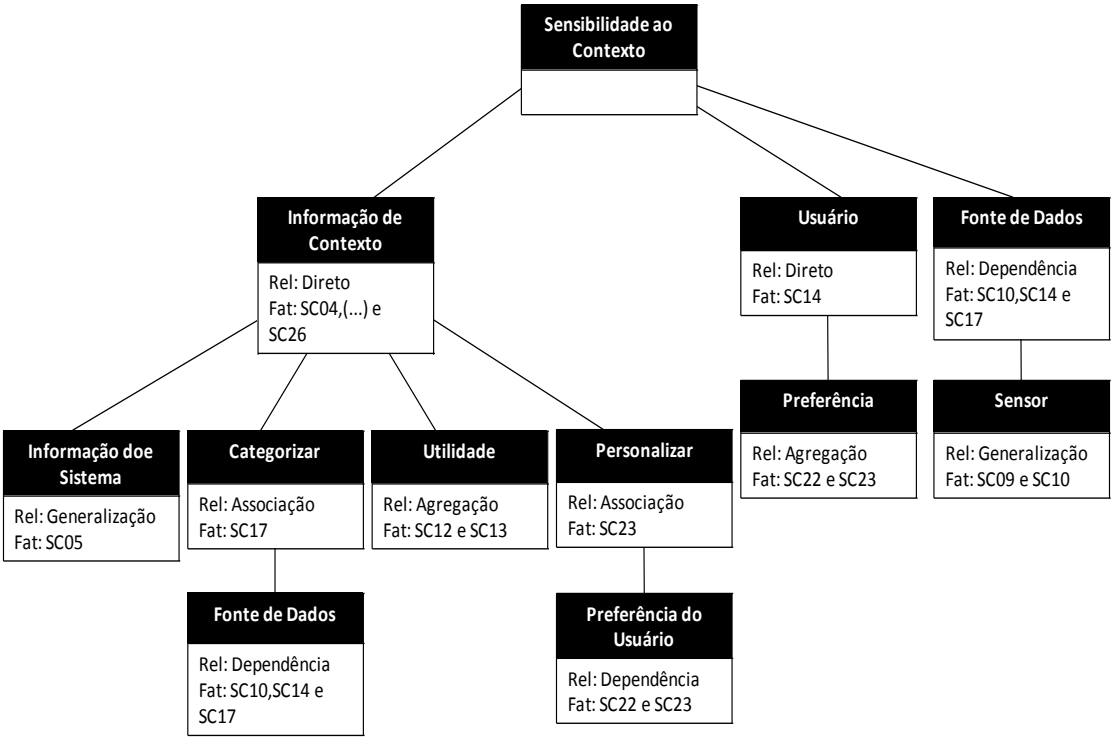


Figura 6.10. Exemplo de representação de uma árvore de elementos de um trecho do modelo da característica Sensibilidade ao Contexto.

Tabela 6.2. Exemplo de representação EM TABELA de uma árvore de elementos de um trecho do modelo da característica Sensibilidade ao Contexto

Fatores	Relacionamento	Nó
SC04,(...) e SC26	Direto	Informação de Contexto
SC05	Generalização	- - Informação do Sistema
SC17	Associação	- - Categorizar
SC10,SC14 e SC17	Dependência	- - - - Fonte de Dados
SC12 e SC13	Agregação	- - Utilidade
SC23	Associação	- - Personalizar
SC22 e SC23	Dependência	- - - - Preferência do Usuário
SC14	Direto	Usuário
SC22 e SC23	Agregação	- - Preferência
SC10,SC14 e SC17	Direto	Fonte de Dados
SC09 e SC10	Generalização	- - Sensor

Na seção seguinte a árvore de elementos gerada será utilizada para elaborar as perguntas do guia geral de definição de requisitos de ubiquidade.

6.5.1.2 Elaborar Perguntas

As perguntas que compõem o Guia Geral de Definição de Requisitos de Ubiquidade são elaboradas a partir da árvore de elementos representadas em tabelas. Para isto são realizadas duas tarefas. Para descrevê-las, será utilizada a mesma árvore de elementos apresentada na Tabela 6.2.

A primeira tarefa é **Decompor a Árvore de Elementos**. Nesse sentido, são listados os caminhos de cada nó da árvore até a raiz, sem incluí-la, conforme mostra o exemplo a seguir:

Informação de Contexto
Informação do Sistema > Informação de Contexto
Categorizar > Informação de Contexto
Fonte de Dados > Categorizar > Informação de Contexto
Utilidade > Informação de Contexto
Personalizar > Informação de Contexto
Preferência do Usuário > Personalizar > Informação de Contexto
Usuário
Preferência > Usuário
Fonte de Dados
Sensor > Fonte de Dados

A segunda tarefa é **Elaborar as Perguntas** a partir desses caminhos. Para isso, cada caminho será utilizado para gerar uma pergunta. O primeiro nó do caminho é utilizado para especificar o **assunto da pergunta**; o seu atributo de relacionamento (*direto, associação, generalização, dependência ou agregação*) é utilizado para especificar o **tipo da pergunta**; e os demais nós são utilizados para especificar o **escopo da pergunta** (contexto).

Para auxiliar no entendimento, a Tabela 6.3 mostra o assunto, escopo e tipo extraído dos caminhos apresentados.

Tabela 6.3. Assunto, Escopo e Tipo da Pergunta

Assunto	Escopo	Tipo
Informação de Contexto		Direto
Informação do Sistema	Informação de Contexto	Generalização
Categorizar	Informação de Contexto	Associação
Fonte de Dados	Categorizar Informação de Contexto	Dependência
Utilidade	Informação de Contexto	Agregação
Personalizar	Informação de Contexto	Associação
Preferência do Usuário	Personalizar Informação de Contexto	Dependência
Usuário		Direto

Preferência	Usuário	Agregação
Fonte de Dados		Direto
Sensor	Fonte de Dados	Generalização

A partir do assunto se define o que a pergunta deve capturar, por exemplo, para o assunto referente ao nó *Utilidade*, pretende-se capturar informações sobre como a *Utilidade* é considerada no sistema. Como se pode perceber, considerando apenas o assunto, a pergunta pode ficar muito vaga.

Com o escopo da pergunta, se define o contexto em que ela será considerada. Nesse sentido, o escopo tem o papel de restringir o assunto da pergunta, tornando-a mais objetiva. Por exemplo, o escopo *Informação de Contexto* junto ao assunto *Utilidade* pretende capturar informações sobre a *Utilidade* da *Informação de Contexto*.

As perguntas para os nós mais próximos da raiz da árvore tendem a ser mais genéricas. Conforme vão sendo considerados os nós mais distantes, as perguntas tendem a ser mais objetivas, porque vão existir mais nós para definir o seu escopo.

Adicionalmente, o tipo da pergunta define como será o seu formato. A Tabela 6.4 mostra como cada pergunta deve ser elaborada de acordo com o seu tipo. Nela também são apresentadas as abreviaturas que serão utilizadas para representar cada tipo de pergunta.

Tabela 6.4. Tipos de Perguntas

Tipo	Abrev.	Formato da Pergunta
Direto	DIR	Quais ... relevantes para o sistema? Ex: Quais as *informações de contexto* relevantes para o sistema?
Agregação	AGR	Como ... são consideradas? Ex: Como a *utilidade da informação de contexto* é considerada?
Associação	ASS	Como ...? Exemplo: Como *categorizar informações de contexto*?
Generalização	GEN	Quais ... são do tipo ...? Exemplo: Quais *fontes de dados* são do tipo *sensor*?
Dependência	DEP	Como ... influencia ...? Exemplo: Como a *fonte de dados* influencia na *categorização de informações de contexto*?

A Tabela 6.5 mostra alguns exemplos de perguntas elaboradas com base na tríade assunto, escopo e tipo. Por exemplo, a pergunta 4 *"Como a fonte de dados influencia na categorização das informações de contexto?"* foi redigida dessa forma por que:

1. seu assunto é o nó *fonte de dados,* portanto, pretende-se investigar informações sobre fontes de dados;
2. seu escopo é c*ategorizar informação de contexto*, portanto, pretende-se investigar informações sobre fontes de dados relacionadas a categorização de informações de contexto; e
3. seu tipo é *DEP* (dependência), portanto, deve ser escrita no formato "*Como ... influencia ...?*"

Tabela 6.5. Exemplos de Perguntas

Fatores	Tipo	Nó	Pergunta
SC04,(...) e SC26	DIR.	Informação de Contexto	1. Quais as *informações de Contexto* relevantes para o sistema?
SC05	GEN	- - Informação do Sistema	2. Quais *informações de contexto* são do tipo *informação do sistema*?
SC17	ASS	- - Categorizar	3. Como *categorizar informações de contexto*?
SC10,SC14 e SC17	DEP	- - - - Fonte de Dados	4. Como a *fonte de dados* influencia na categorização das *informações de contexto*?
SC12 e SC13	AGR	- - Utilidade	5. Como a *utilidade* da *informação de contexto* é considerada?
SC23	ASS	- - Personalizar	6. Como *personalizar informações de contextos*?
SC22 e SC23	DEP	- - - - Preferência do Usuário	7. Como a *preferência do usuário* influencia na *personalização* de *informações de contexto*?
SC14	DIR	Usuário	8. Quais os usuários relevantes para o sistema?
SC22 e SC23	AGR	- - Preferência	9. Como a *preferência* do *usuário* é considerada?
SC10,SC14 e SC17	DIR	Fonte de Dados	10. Quais *fontes de dados* são relevantes para o sistema?
SC09 e SC10	GEN	- - Sensor	11. Quais *fontes de dados* são do tipo *sensor*?

Ao final desta etapa, tem-se o conjunto de perguntas geradas que formarão o Guia Geral de Definição de Requisitos de Ubiquidade considerando todas as características e seus respectivos fatores funcionais presentes no corpo de conhecimento organizado neste trabalho. A versão completa deste guia pode ser encontrada no Apêndice F.

6.5.2. Especializar Guia Geral de Definição de Requisitos de Ubiquidade

O *Guia Geral de Definição de Requisitos de Ubiquidade* elaborado até este ponto considera as informações de todos os fatores de ubiquidade. Entretanto, não considera se todos esses fatores devem ou não estar presentes no projeto.

Conforme discutido na seção 6.4, projetos de software ubíquo podem contemplar uma pequena parcela dos fatores de ubiquidade. Dessa forma, é importante aproveitar essa situação para especializar o *Guia Geral de Definição de Requisitos de Ubiquidade* de forma que seja possível considerar apenas as perguntas que tenham relação com os fatores de ubiquidade considerados relevantes para o projeto, o que pode trazer os seguintes benefícios:

- **Foco no que é importante**: o desenvolvedor não perderia tempo respondendo perguntas que não dizem respeito ao seu projeto, o que poderia reduzir a quantidade de informações estranhas capturadas nos requisitos.
- **Redução do tamanho do guia:** ele passará a conter menos perguntas, portanto, exigirá menos esforço para ser aplicado em um projeto.

Para lidar com esta necessidade de configuração, foi preparada uma estratégia para especializar o *Guia Geral de Definição de Requisitos de Ubiquidade* a partir do questionário de caracterização de projetos de software ubíquo discutido na seção 6.4.

Uma vez selecionados os fatores que farão parte do projeto através do uso do questionário de caracterização, o próximo passo é **Especializar o Guia Geral de Definição de Requisitos de Ubiquidade**. A especialização é realizada através da seleção das perguntas que possuem relação com esses fatores, ou seja, aquelas perguntas que foram originadas por pelo menos um fator considerado relevante para o projeto durante sua caracterização.

A Tabela 6.6 mostra um trecho do *Guia Geral de Definição de Requisitos de Ubiquidade* da característica *Sensibilidade ao Contexto* que será utilizado para exemplificar como esta atividade é realizada (a versão completa pode ser encontrada no Apêndice F).

Tabela 6.6. Trecho do Guia Geral de Definição de Requisitos de Ubiquidade da característica Sensibilidade ao Contexto

Fatores	Pergunta
SC05,SC10,SC12,SC 13, SC14,SC17, SC22 e SC23	1. Quais as *informações de Contexto* relevantes para o sistema?
SC05	2. Quais *informações de contexto* são do tipo *informação do sistema*?
SC17	3. Como *categorizar informações de contexto*?
SC10,SC14 e SC17	4. Como a *fonte de dados* influencia na categorização das *informações de contexto*?
SC12 e SC13	5. Como a *utilidade* da *informação de contexto* é considerada?
SC23	6. Como *personalizar informações de contextos*?
SC22 e SC23	7. Como a *preferência do usuário* influencia na *personalização* de

Fatores	Pergunta
	informações de contexto?
SC14	8. Quais os usuários relevantes para o sistema?
SC22 e SC23	9. Como a *preferência* do *usuário* é considerada?
SC10,SC14 e SC17	10. Quais *fontes de dados* são relevantes para o sistema?
SC09 e SC10	11. Quais *fontes de dados* são do tipo *sensor*?

A partir dela, as perguntas foram agrupadas de acordo com os fatores que as originaram, conforme mostram as listagens a seguir:

SC05
1. Quais as *informações de Contexto* relevantes para o sistema?
2. Quais *informações de contexto* são do tipo *informação do sistema*?

SC09
11. Quais *fontes de dados* são do tipo *sensor*?

SC10
1. Quais as *informações de Contexto* relevantes para o sistema?
4. Como a *fonte de dados* influencia na categorização das *informações de contexto*?
10. Quais *fontes de dados* são relevantes para o sistema?
11. Quais *fontes de dados* são do tipo *sensor*?

SC12
1. Quais as *informações de Contexto* relevantes para o sistema?
5. Como a *utilidade* da *informação de contexto* é considerada?

SC13
1. Quais as *informações de Contexto* relevantes para o sistema?
5. Como a *utilidade* da *informação de contexto* é considerada?

SC14
1. Quais as *informações de Contexto* relevantes para o sistema?
4. Como a *fonte de dados* influencia na categorização das *informações de contexto*?
8. Quais os usuários relevantes para o sistema?
10. Quais *fontes de dados* são relevantes para o sistema?

SC17
1. Quais as *informações de Contexto* relevantes para o sistema?
3. Como *categorizar informações de contexto*?
4. Como a *fonte de dados* influencia na categorização das *informações de contexto*?
10. Quais *fontes de dados* são relevantes para o sistema?

SC22

1. Quais as *informações de Contexto* relevantes para o sistema?

7. Como a *preferência do usuário* influencia na *personalização* de *informações de contexto*?

9. Como a *preferência* do *usuário* é considerada?

SC23

1. Quais as informações de Contexto relevantes para o sistema?

6. Como *personalizar informações de contextos*?

7. Como a *preferência do usuário* influencia na *personalização* de *informações de contexto*?

9. Como a *preferência* do *usuário* é considerada?

Continuando o exemplo, para especializar o *Guia Geral de Definição de Requisitos de Ubiquidade* realiza-se a união das perguntas associadas a cada fator relevante para o projeto.

Supondo que tenha sido identificado como pertinente para o projeto definir requisitos associados apenas aos fatores SC12, SC13 e SC23, o guia especializado teria todas as perguntas associadas a esses fatores. Como o fator SC12 tem as perguntas 1 e 5 associadas; o fator SC13 também tem as mesmas perguntas (1 e 5) e o fator SC23 tem as perguntas 1, 6, 7 e 9; o guia especializado para esses três fatores seria o conjunto de perguntas 1, 5, 6, 7 e 9, conforme listado a seguir:

SC12, SC13 e SC23

1. Quais as *informações de Contexto* relevantes para o sistema?

5. Como a *utilidade* da *informação de contexto* é considerada?

6. Como *personalizar informações de contextos*?

7. Como a *preferência do usuário* influencia na *personalização* de *informações de contexto*?

9. Como a *preferência* do *usuário* é considerada?

Caso o fator SC05 também fosse interessante para o projeto, as suas perguntas (1 e 2) também seriam consideradas no guia especializado, conforme listado a seguir:

SC05, SC12, SC13 e SC23
1. Quais as *informações de Contexto* relevantes para o sistema?
2. Quais *informações de contexto* são do tipo *informação do sistema*?
5. Como a *utilidade* da *informação de contexto* é considerada?
6. Como *personalizar informações de contextos*?
7. Como a *preferência do usuário* influencia na *personalização* de *informações de contexto*?
9. Como a *preferência* do *usuário* é considerada?

É importante destacar que apenas a pergunta 2 é realmente nova neste novo guia, pois a pergunta 1 já estava contemplada no guia anterior por fazer parte dos fatores SC12, SC13 e SC23.

Uma vez finalizada esta atividade, tem-se como resultado o *Guia Especializado de Definição de Requisitos de Ubiquidade*. A próxima atividade é definir os requisitos de ubiquidade do projeto, o que é descrito na próxima seção.

6.5.3. Definir Requisitos de Ubiquidade do Projeto

UbiCheck é um *checklist* que objetiva indicar para o desenvolvedor as informações importantes que devem ser observadas e capturadas durante a definição de requisitos.

Neste tipo de abordagem, o desenvolvedor não recebe apoio sobre como deve realizar a captura das informações. Dessa forma, aplicar *UbiCheck* consiste em responder as perguntas do *Guia Especializado de Definição de Requisitos de Ubiquidade* durante a definição dos requisitos de ubiquidade do projeto. Conforme o desenvolver responde as perguntas, a sua atenção é direcionada para os assuntos relacionados à ubiquidade que devem ser capturados no documento de requisitos (LAITENBERGER *et al.*, 2001).

Mesmo não havendo indicações de como proceder para capturar os requisitos, o simples fato de poder utilizar o conhecimento de domínio durante essa atividade já pode ser considerado uma vantagem, pois pode permitir que o desenvolvedor saiba o que precisa capturar nos requisitos do projeto.

Por fim, uma consideração importante sobre o uso de *UbiCheck* é que o apoio fornecido por ela se aplica apenas à definição dos requisitos de ubiquidade. Os demais requisitos (funcionais e não funcionais) associados ao domínio do problema ainda devem ser definidos e para isso outras abordagens podem ser combinadas ao uso de *UbiCheck*.

6.6 – Verificação de Requisitos de Ubiquidade - *UbiVeri*

Para apoiar as atividades associadas à verificação dos requisitos de ubiquidade foi definido *UbiVeri*. A Figura 6.11[15] apresenta o encadeamento das atividades de *UbiVeri*, indicando: (1) os responsáveis pela execução de cada etapa; (2) as etapas consideradas; (3) as atividades executadas em cada etapa; e (4) os artefatos consumidos e gerados no processo.

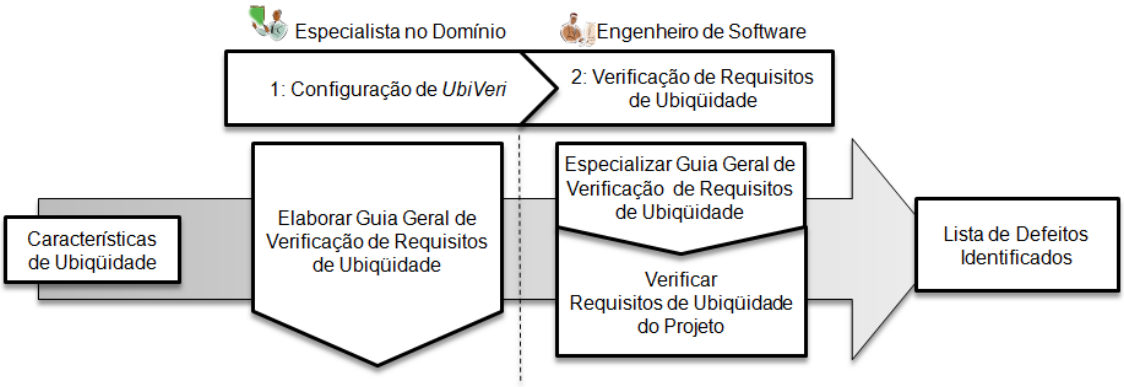


Figura 6.11. Visão geral de *UbiVeri*

Como pode ser observado, *UbiVeri* é organizada para ser aplicada em duas etapas. A primeira está associada à sua configuração e é composta pela atividade **Elaborar Guia Geral de Verificação de Requisitos de Ubiquidade**. Já a segunda etapa está associada à **Verificação de Requisitos de Ubiquidade** e possui duas atividades: Especializar Guia Geral de Verificação de Requisitos de Ubiquidade; e Verificar Requisitos de Ubiquidade do Projeto. O resultado final do uso da abordagem é a lista de defeitos identificados na especificação dos requisitos do projeto. Esta lista deverá ser enviada ao autor do documento para que os ajustes possam ser efetuados.

As próximas seções apresentarão cada uma das atividades que compõem *UbiVeri*.

6.6.1. Elaborar Guia Geral de Verificação de Requisitos de Ubiquidade

Esta atividade tem o objetivo de elaborar o *Guia Geral de Verificação de Requisitos de Ubiquidade.* Ele é composto por um conjunto de instruções e perguntas sobre informações presentes nos fatores de ubiquidade que seriam interessantes de serem verificados durante a revisão de requisitos de ubiquidade do projeto.

A elaboração das perguntas de apoio à identificação de defeitos considerou:

[15] *Esta figura representa um recorte da Figura 6.2 com foco nas atividades que compõem UbiVeri.*

- os fatores das características funcionais de ubiquidade;
- os modelos conceituais definidos para cada uma das características de ubiquidade (desta forma, o conhecimento está fundamentado nos conceitos definidos nos Capítulos 2 e 4 deste texto);
- a taxonomia de defeitos descrita em (SHULL *et al.,* 2000) contendo os seguintes tipos de defeito: ambiguidade, inconsistência, fato incorreto, omissão e informação estranha.

Para isto, as perguntas foram classificadas em dois tipos:

- **Específicas:** estão associadas a defeitos de omissão. Estas perguntas estão definidas de forma padronizada e seguem a estrutura:

  Está definido *+ complemento da funcionalidade + ?*

- **Genéricas:** estão associadas aos defeitos de ambiguidade, inconsistência, fato incorreto e informação estranha. As perguntas genéricas devem ser formuladas para cada uma das perguntas específicas durante o processo de revisão. Por exemplo, considere a pergunta específica:

  Está definido *em quais dispositivos o software deverá executar?*

  As perguntas genéricas complementam esta pergunta e seriam:

  Ambiguidade: *A definição sobre os dispositivos utilizados* ***está clara?***

  Inconsistência: *A definição sobre os dispositivos utilizados* ***é feita de forma diferente em locais diferentes no documento de requisitos****?*

  Fato Incorreto: *A definição sobre os dispositivos utilizados* ***está correta****?*

  Informação Estranha: *Existe algo descrito no documento de requisitos que está claramente fora do escopo do projeto?*

Assim, o processo de geração das perguntas específicas e genéricas está fundamentado nos questionamentos apresentados na Tabela 6.7.

Tabela 6.7. Questionamentos associados às perguntas específicas e genéricas.

Tipo de Defeito	Questionamento
Omissão	**Está definido** + *funcionalidade* + **?**
Ambiguidade	**A definição sobre a** + *Funcionalidade* + **está clara?**
Inconsistência	**A definição sobre a** + *Funcionalidade* + **é feita de forma diferente em locais diferentes no documento de requisitos?**
Fato Incorreto	**A definição sobre a** + *Funcionalidade* + **está correta?**
Informação estranha	Existe algo descrito no documento de requisitos que está claramente fora do escopo do projeto?

Estes questionamentos são feitos em duas etapas:

- Etapa 1: identifica se há uma omissão. Se houver uma omissão associada a um fator de ubiquidade, isso indica que ele não está definido e um defeito deve ser reportado. Caso o requisito associado ao fator esteja definido, parte-se para a etapa 2;
- Etapa 2: identifica se há um problema de ambiguidade, inconsistência, fato incorreto ou informação estranha associado ao requisito definido.

Com isto, a formulação das perguntas genéricas é apenas realizada em um segundo momento (caso não haja problema de omissão associado ao requisito).

Definidos estes conceitos, partiu-se então para a elaboração dos questionamentos presentes na guia geral de verificação. De forma semelhante à atividade Elaborar Guia Geral de Definição de Requisitos de Ubiquidade, para esta atividade utilizou-se a árvore de elementos descrita na seção 6.5.1.1 para se definir as perguntas que apóiam a identificação de defeitos nos requisitos de ubiquidade do projeto.

O processo de definição das perguntas é bastante semelhante como um todo e não será descrito novamente aqui. Entretanto, uma adaptação precisou ser efetuada. Observe que definir cinco perguntas (1 específica e 4 genéricas) para cada nó folha da árvore de elementos poderia gerar uma quantidade de questionamentos muito grande e isso certamente impactaria negativamente o uso do guia de verificação. Para ilustrar isso, basta imaginar que um conjunto de 10 elementos da árvore (um número pequeno) geraria 50 perguntas de verificação.

Para lidar com isto e melhorar a usabilidade da guia, optou-se por definir apenas as perguntas específicas no guia (associadas aos defeitos de omissão). A elaboração das perguntas genéricas passou a ser de responsabilidade do engenheiro de software responsável por efetuar a revisão dos requisitos.

Este processo de elaboração de perguntas genéricas é simples e sua explicação foi acrescentada no início do guia de forma que o engenheiro entendesse seu funcionamento.

Além da taxonomia de defeitos mencionada anteriormente, a abordagem também considera a escala de severidade do defeito:

- 0 → Problema leve - baixa prioridade para consertá-lo;
- 1 → Problema grave - alta prioridade para consertá-lo.

É importante capturar esta informação uma vez que ela formará junto com a lista de defeitos, uma base de dados que permitirá a evolução das abordagens de definição e verificação de requisitos apresentadas neste texto.

Assim, ao final da definição das perguntas, acrescentou-se um conjunto de instruções à guia geral de verificação de forma a facilitar seu entendimento por parte do engenheiro de software. O resultado da execução desta atividade foi o Guia Geral de Verificação de Requisitos de Ubiquidade elaborado. A Figura 6.12 apresenta um fragmento do guia para a característica sensibilidade ao contexto.

Alguns outros elementos definidos no guia e que apóiam seu uso são:

- **Item de Especificação:** indica em quais trechos de uma especificação de requisitos aquele tipo de questionamento pode ser feito para a identificação de defeitos;
- **Item de Glossário:** os termos sublinhados no guia são específicos do domínio da computação ubíqua e estão definidos em um glossário de termos que compõe a guia de verificação (este glossário pode ser encontrado no Apêndice G.2).

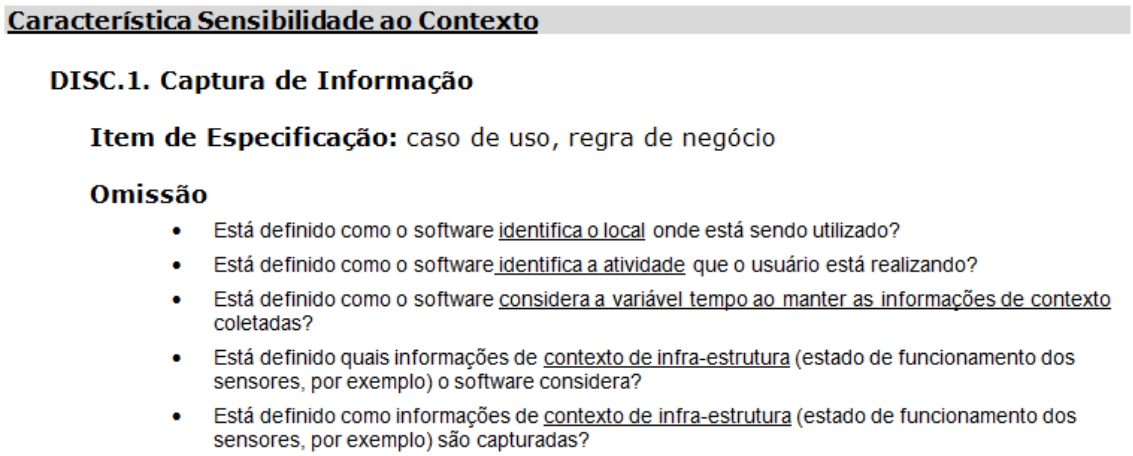


Figura 6.12. Fragmento da Guia Geral de Verificação de Requisitos de Ubiquidade.

Por fim, foi definido um formulário[16] para relato das discrepâncias identificadas durante a revisão do documento de requisitos considerando os requisitos de ubiquidade (ver Figura 6.13).

Relatório de Discrepâncias

ID	Localização	Característica de Ubiquidade	Tipo de Defeito	Severidade	Descrição

Figura 6.13. Fragmento do Formulário de Relato de Discrepâncias.

Dessa forma, a Guia Geral de Verificação de Requisitos de Ubiquidade é composta por:

- Instruções de uso;
- Um conjunto de questionamentos para identificação de defeitos;
- Um glossário de termos em computação ubíqua;
- Um formulário para relato das discrepâncias identificadas durante a revisão dos requisitos.

Uma versão completa da Guia Geral de Verificação pode ser encontrada no Apêndice G.

6.6.2. Especializar Guia Geral de Verificação de Requisitos de Ubiquidade

O *Guia Geral de Verificação de Requisitos de Ubiquidade* elaborado até este ponto considera as informações de todos os fatores funcionais de ubiquidade. Entretanto, assim como no Guia Geral de Definição, ele não considera se todos esses fatores são importantes para o projeto em que ele será aplicado.

Conforme discutido na seção 6.4, projetos de software ubíquo podem contemplar uma pequena parcela dos fatores de ubiquidade. Dessa forma, é importante especializar o *Guia Geral de Verificação de Requisitos de Ubiquidade* de forma que seja possível considerar apenas as perguntas que tenham relação com os fatores considerados relevantes para o projeto. Dessa forma, o guia terá um foco mais preciso e exigirá menos esforço para ser aplicado uma vez que a quantidade de perguntas presente no guia será reduzida.

[16] *Este formulário foi inspirado nos relatórios de discrepância utilizados por Kalinowski et. al. (2004) e Silva et al. (2004).*

Para realizar esta especialização, foi utilizado o questionário de caracterização de projetos de software ubíquo discutido na seção 6.4. Uma vez selecionados os fatores que farão parte do projeto através do uso do questionário de caracterização, o Guia Geral de Verificação é especializado através da seleção das perguntas que possuem relação com esses fatores.

Este processo de especialização é semelhante ao discutido na seção 6.5.2 e por isso não será novamente detalhado aqui. Ao final da especialização tem-se como resultado o *Guia Especializado de Verificação de Requisitos de Ubiquidade*. A próxima atividade, descrita na próxima seção, é verificar os requisitos de ubiquidade do projeto.

6.6.3. Verificar Requisitos de Ubiquidade

UbiVeri é uma abordagem baseada em perguntas, portanto, o apoio fornecido objetiva indicar para o desenvolvedor que tipo de informação (baseado em um corpo de conhecimento em computação ubíqua) ele deve procurar na especificação dos requisitos com objetivo de identificar defeitos.

Na abordagem proposta o desenvolvedor não recebe apoio sobre como deve realizar a leitura dos requisitos em busca de defeitos. Dessa forma, aplicar *UbiVeri* consiste em responder as perguntas do *Guia Especializado de Verificação de Requisitos de Ubiquidade* durante a revisão dos requisitos de ubiquidade do projeto. Conforme o engenheiro de software responde as perguntas, a sua atenção é direcionada para os assuntos relacionados à ubiquidade que deveriam ter sido capturados no documento de requisitos, o que permitirá a ele reportar os defeitos identificados durante a revisão.

Por fim, uma consideração importante sobre o uso de *UbiVeri* é que o apoio fornecido por ela se aplica apenas à verificação dos requisitos de ubiquidade. Os demais requisitos (funcionais e não funcionais) associados ao domínio do problema ainda devem ser revisados e para isso outras abordagens podem ser combinadas ao uso de *UbiVeri*.

6.7 – Considerações Finais

Este capítulo apresentou a abordagem de apoio à definição e verificação de requisitos de ubiquidade em projetos de software. Dentre as principais características presentes nesta abordagem que a tornam original no contexto do desenvolvimento de projetos de software ubíquo, podem ser citadas:

- Apoiar as atividades considerando um corpo de conhecimento em computação ubíqua construído através de estudos experimentais;
- Direcionar os trabalhos do engenheiro de software dentro de um domínio muito específico (computação ubíqua) em atividades que vão desde a identificação inicial de quais características de ubiquidade estarão presentes no software até o detalhamento dos requisitos de ubiquidade que estarão contemplados;
- Unir o corpo de conhecimento em computação ubíqua à taxonomia de classificação de defeitos apresentada em (SHULL *et al.,* 2000), através de um conjunto de diretrizes para auxiliar o revisor a identificar defeitos na especificação de requisitos do software considerando os requisitos de ubiquidade.

Por fim, considerando as limitações das abordagens identificadas na literatura técnica e relatadas no Capítulo 5, *u*-RDV traz as seguintes contribuições:

- Uso de um processo interativo para identificação e definição de requisitos direcionado por um corpo de conhecimento em computação ubíqua. Nos trabalhos identificados na literatura, havia o uso de um processo interativo, o conhecimento do domínio em computação ubíqua não era utilizado no apoio à realização das atividades;
- A abordagem proposta apresenta um processo e as técnicas utilizadas para apoiar a execução de cada atividade do processo. Nos trabalhos identificados na literatura, foram identificados no máximo alguns passos que deveriam ser executados pelo analista de requisitos para identificar e especificar os requisitos de ubiquidade. Ou seja, não havia a definição de um processo nem um detalhamento sobre técnicas que poderiam ser utilizadas para apoiar a execução de suas atividades.
- A abordagem proposta neste trabalho apóia a verificação dos requisitos especificados. Não foi identificado na literatura nenhum trabalho com este propósito no contexto da computação ubíqua.

Definida a abordagem, o próximo passo é sua avaliação experimental. Esta atividade será dividida em duas etapas. Na primeira será avaliada *UbiCheck* (abordagem de apoio à definição de requisitos). Em seguida, será efetuada a avaliação da abordagem de verificação *UbiVeri*.

Sendo assim, o próximo Capítulo apresentará o planejamento e os resultados do estudo experimental que teve como objetivo avaliar a abordagem de apoio à definição de requisitos, *UbiCheck*.

Capítulo 7 - Avaliação de *UbiCheck*

Este capítulo descreve o plano e o resultado do estudo experimental para avaliar os benefícios que o uso de UbiCheck - abordagem de apoio à definição de requisitos de ubiquidade em projetos de software ubíquo - traz. Ao final, é apresentada a evolução da abordagem a partir dos resultados obtidos no estudo.

7.1 - Introdução

Nesta pesquisa, foi realizado um estudo de observação para caracterizar *UbiCheck* no que diz respeito a sua viabilidade de aplicação na definição de requisitos de ubiquidade em projetos de software. O plano do estudo e os resultados de sua execução serão apresentados neste capítulo.

7.2 - Avaliação do uso de *UbiCheck*

A apresentação do estudo experimental seguirá o padrão sugerido por Wohlin *et al.* (2000) para descrever um plano de um estudo experimental. Dessa forma, o experimento executado será descrito em quatro seções: definição do estudo experimental, planejamento do estudo experimental, operação, análise e interpretação dos dados. Os instrumentos do estudo podem ser encontrados no Apêndice H e I.

7.2.1 Definição do Estudo Experimental

O propósito deste estudo é observar a aplicação de *UbiCheck* em um projeto de desenvolvimento de software ubíquo para compreender se a abordagem pode ser utilizada para apoiar a definição de requisitos de ubiquidade. Nesse sentido, foi analisado o uso do Guia para Definição de Requisitos de Ubiquidade no desenvolvimento de um Sistema de Gestão de Inventário Patrimonial (SGP).

Utilizando o GQM (BASILI *et al.*, 1994), o objetivo deste estudo é:

- **Analisar a** definição de requisitos de ubiquidade com *UbiCheck*
- **Com o propósito de** caracterizar o Guia para Definição de Requisitos de Ubiquidade
- **Com respeito a** sua aplicabilidade
- **Do ponto de vista de** estudantes de engenharia de software

- **No contexto do** projeto de um Sistema de Gestão de Inventário Patrimonial (SGP).

O intuito de caracterizar a aplicabilidade do Guia para Definição de Requisitos de Ubiquidade é verificar se os desenvolvedores são capazes de entendê-lo e aplicá-lo em um projeto de software. Assim, pretende-se avaliar, através da coleta e análise de dados quantitativos e qualitativos durante a operação do experimento, as seguintes questões gerais:

- Q1: O uso da abordagem proposta reduz o esforço em tempo gasto durante a atividade de definição de requisitos de ubiquidade do projeto quando comparado à definição de requisitos de ubiquidade de forma *ad hoc*?
- Q2: O uso da abordagem proposta aumenta o grau de satisfação dos engenheiros de software na definição dos requisitos de ubiquidade quando comparada à definição dos requisitos de ubiquidade de forma *ad hoc*?

7.2.2 Planejamento do Estudo Experimental

Seleção do Contexto

De acordo com (WOHLIN *et al.*, 2000), o contexto pode ser caracterizado em quatro dimensões:

- **processo:** on-line / **off-line**;
- **participantes: alunos** / profissionais;
- **realidade:** problema real / **modelado**;
- **generalidade: específico** / geral.

Este estudo experimental se propõe a ser executado de forma *off-line*, uma vez que as atividades serão distribuídas aos participantes e será definido um prazo para sua conclusão. Os participantes utilizados no estudo serão alunos do curso de engenharia de software atuando em um problema modelado. A generalidade do estudo é específica.

Formulação das Hipóteses

A seguir são apresentadas as hipóteses nula e alternativas deste estudo experimental:

- **Hipótese nula (H_0):** Não existe diferença na atividade de definição de requisitos de ubiquidade usando *UbiCheck* e uma abordagem *ad hoc*.

$$H_0 : (\iota_a = \iota_u) \wedge (\mu_a = \mu_u)$$, onde:

  - ι_a é o tempo gasto na atividade de definição de requisitos de ubiquidade de forma *ad hoc*.
  - ι_u é o tempo gasto na atividade de definição de requisitos de ubiquidade utilizando *UbiCheck*.
  - μ_a é o grau de satisfação dos participantes na atividade de definição de requisitos de ubiquidade de forma *ad hoc*.
  - μ_u é o grau de satisfação dos participantes na atividade de definição de requisitos de ubiquidade utilizando *UbiCheck*.

- **Hipótese alternativa (H_1):** O uso de *UbiCheck* reduz o esforço de realização da atividade de definição dos requisitos de ubiquidade se comparada ao uso de uma abordagem *ad hoc*.

$$H_1 : \iota_a > \iota_u$$

- **Hipótese alternativa (H_2):** O uso de *UbiCheck* leva a um grau de satisfação maior dos participantes na definição de requisitos de ubiquidade se comparada à definição de requisitos de ubiquidade de forma *ad hoc*.

$$H_2 : \mu_a < \mu_u$$

Seleção de Variáveis

O estudo experimental utiliza as seguintes variáveis:

- Variáveis independentes:
  - Abordagem de Apoio à Definição de Requisitos de Ubiquidade - *UbiCheck*.
  - Domínio da Aplicação (Sistema de Gestão de Patrimônio).
- Variáveis dependentes:
  - **O tempo gasto na definição dos requisitos:** os participantes preencherão os Formulários UB_1 e UB_2 (apresentados no Apêndice H.3.1 e H.3.2) informando o tempo gasto nas diferentes atividades (análise do material entregue e definição dos requisitos).

- **O grau de satisfação dos participantes na verificação dos requisitos:** os participantes preencherão o Formulário UB_3 (apresentado no Apêndice H.4), informando o nível de satisfação na aplicação da abordagem.

Seleção de Indivíduos

Os participantes selecionados devem representar alunos do curso de engenharia de software. Sua caracterização será realizada através do questionário de caracterização que conterá perguntas a respeito de sua experiência prática com definição de requisitos e formação acadêmica.

Os participantes devem então ser divididos em três grupos utilizando o questionário de caracterização para formar uma divisão de grupos equilibrada em termos de experiência dos participantes. Cada grupo deverá conter no máximo três participantes.

Projeto do Estudo Experimental

O projeto deste estudo experimental envolve dois fatores: (1) definição de requisitos de ubiquidade de um projeto de software; (2) domínio de aplicação.

O fator definição de requisitos de ubiquidade de um projeto de software terá dois tratamentos (no Apêndice I estão disponíveis informações sobre os cenários de ubiquidade distribuídos):

- **Definição de Requisitos de Ubiquidade de forma *ad hoc*:** os participantes receberão, para efetuar a definição dos requisitos de ubiquidade do projeto, a especificação dos requisitos a ser evoluída considerando os cenários de ubiquidade.
- **Definição de Requisitos de Ubiquidade usando *UbiCheck*:** os participantes receberão, para efetuar a definição dos requisitos de ubiquidade do projeto, a especificação dos requisitos a ser evoluída considerando os cenários de ubiquidade e o guia de apoio à definição de requisitos de ubiquidade.

O fator domínio de aplicação terá dois cenários:

- **Sistema de Gestão Patrimonial + Item de Patrimônio Rastreável (Cenário 1):** Extensão ao SGP com objetivo de controlar de forma efetiva a entrada e saída de itens rastreáveis na organização.

- **Sistema de Gestão Patrimonial + Bicicleta Rastreável (Cenário 2):** Extensão ao SGP com objetivo de controlar de forma efetiva o uso de bicicletas na organização.

Embora façam parte do mesmo domínio (gestão patrimonial), os dois cenários estão sendo tratados neste trabalho como dois tratamentos. Isto por que eles implicam nas mesmas preocupações no projeto do estudo experimental que deveriam ser consideradas caso tivéssemos dois tratamentos distintos para o fator domínio da aplicação.

O estudo experimental faz uso da seguinte divisão de grupos:

- **Grupo 1:** fará inicialmente uma definição *ad hoc* dos requisitos para o cenário 2 (bicicletário). Em seguida, fará a definição dos requisitos considerando o cenário 1 (item rastreável) de ubiquidade utilizando *UbiCheck*.
- **Grupo 2:** fará inicialmente uma definição *ad hoc* dos requisitos para o cenário 1 (item rastreável). Em seguida, fará a definição dos requisitos considerando o cenário 2 (bicicletário) de ubiquidade utilizando *UbiCheck*.
- **Grupo 3:** fará inicialmente uma definição *ad hoc* dos requisitos para o cenário 1 (item rastreável). Em seguida, fará a definição dos requisitos considerando o cenário 2 (bicicletário) de ubiquidade utilizando *UbiCheck*.

A Tabela 7.1 apresenta a distribuição das atividades considerando estes três grupos.

Tabela 7.1. Alocação das abordagens de verificação por grupos e dos cenários de ubiquidade

	Grupo 1		**Grupo 2**		**Grupo 3**	
Rodada 1	*Ad Hoc*	C_2	*Ad Hoc*	C_1	*Ad Hoc*	C_1
Rodada 2	*UbiCheck*	C_1	*UbiCheck*	C_2	*UbiCheck*	C_2

Procedimento Experimental

No estudo experimental, os participantes dos dois tratamentos recebem um problema modelado, que é a definição dos requisitos de ubiquidade para o Sistema de Gestão Patrimonial. Os participantes receberão a especificação de requisitos do sistema e realizarão a definição dos requisitos com objetivo de acrescentar os requisitos de ubiquidade ao projeto. A atividade de definição será feita de forma

assíncrona e contará com a participação de um moderador que acompanhará o experimento *off-line*.

Além da especificação dos requisitos do software, na segunda rodada do estudo os grupos também receberão o guia de apoio à definição de requisitos de ubiquidade.

Ao final da definição dos requisitos, os participantes devem preencher um questionário de acompanhamento para que possam fornecer informações qualitativas a respeito da atividade realizada, incluindo esforço envolvido e grau de satisfação na realização da tarefa.

O experimento foi organizado em 2 rodadas, como mostrado na Tabela 7.2. Inicialmente, para todos os grupos, será apresentado o trabalho e será realizado um treinamento sobre os principais conceitos associados à ubiquidade computacional. O objetivo deste treinamento é apresentar o domínio da computação ubíqua para os envolvidos no estudo.

Feito isso, os Grupos 1, 2 e 3 receberão o pacote do estudo para realização da definição dos requisitos de ubiquidade de forma *ad hoc*. Os grupos terão o prazo de uma semana para realização desta atividade e ao seu término deverão entregar o documento de especificação de requisitos do software atualizado com os requisitos de ubiquidade do projeto assim como os demais formulários disponibilizados no pacote do estudo.

Encerrada a primeira rodada do estudo, será dado início à segunda. Para esta rodada, os Grupos 1, 2 e 3 receberão um novo cenário de ubiquidade junto ao pacote do estudo para realização da definição de requisitos de ubiquidade utilizando *UbiCheck*. Neste momento, estes grupos também serão treinados no uso de *UbiCheck*. A distribuição dos diferentes cenários de ubiquidade utilizados está definida na Tabela 7.1.

Maiores detalhes sobre as atividades realizadas em cada sessão podem ser encontrados na Tabela 7.3.

Tabela 7.2. Procedimento Experimental

	Sessão 1		**Sessão 2**	
Grupo 1	Explicação sobre o Trabalho + Treinamento em Computação Ubíqua	Definição de Requisitos *Ad Hoc*	Treinamento sobre *UbiCheck*	Definição de Requisitos com *UbiCheck*
Grupo 2	Explicação sobre o Trabalho + Treinamento em Computação	Definição de Requisitos *Ad Hoc*	Treinamento sobre *UbiCheck*	Definição de Requisitos com *UbiCheck*

	Ubíqua			
Grupo 3	Explicação sobre o Trabalho + Treinamento em Computação Ubíqua	Definição de Requisitos *Ad Hoc*	Treinamento sobre *UbiCheck*	Definição de Requisitos com *UbiCheck*

Cronograma para Realização do Treinamento e Explicação sobre o Estudo Experimental

A execução do estudo experimental foi realizada entre o período de 06/2009 e 07/2009, como mostrado na Tabela 7.3.

Tabela 7.3. Cronograma do Estudo Experimental

Sessão	Dia	Atividades
1	25/06/2009	1. Explicação sobre o trabalho e apresentação do projeto. 2. Entrega dos cenários a serem considerados para as equipes. 3. Entrega dos formulários a serem preenchidos.
	25/06/2009 a 02/07/2009	1. Execução da definição dos requisitos de ubiquidade para o projeto.
2	02/07/2009	1. Recolhimento dos formulários referentes à primeira parte do estudo. 2. Explicação sobre o trabalho e apresentação do projeto. 3. Treinamento em *UbiCheck.* 4. Entrega dos formulários a serem preenchidos.
	02/07/2009 a 09/07/2009	1. Execução da definição dos requisitos de ubiquidade para o projeto.
	09/07/2009	1. Recolhimento dos formulários referentes à segunda parte do estudo.

Instrumentação

Para a realização do estudo experimental, a instrumentação conta com a especificação de requisitos do Sistema de Gestão Patrimonial acrescido de dois cenários de ubiquidade: item de patrimônio rastreável e bicicletário.

Assim, além da especificação de requisitos, também será disponibilizado a descrição dos cenários de ubiquidade envolvidos. Em específico, o tratamento que fará uso de *UbiCheck* também possui o guia de apoio à definição de requisitos de ubiquidade.

Além destes itens, o experimento também conta com os seguintes instrumentos:

- Um formulário de consentimento (Apêndice H.1).
- Um questionário de caracterização (Apêndice H.2).
- A descrição da tarefa a ser desempenhada pelos participantes (Apêndice H.3).

- Um questionário de acompanhamento para possibilitar a análise qualitativa (Apêndice H.4).

Avaliação da Validade

A validade interna, externa, de constructo e de conclusão do experimento se encontra discutida a seguir:

- **Validade interna.** Como se trata de um estudo envolvendo mais de um grupo, onde cada grupo é submetido a um tratamento distinto, a maior ameaça de validade interna é a relação dos resultados com a seleção dos participantes do estudo. Para evitar esta ameaça e aumentar a validade interna, o estudo deverá utilizar os dados do questionário de caracterização para a divisão dos participantes em grupos de acordo com suas características individuais.
- **Validade externa.** O estudo envolve alunos de um curso de engenharia de software e utiliza um problema real, a definição de requisitos de ubiquidade do Sistema de Gestão Patrimonial - SGP. Não é possível generalizar os resultados para outros contextos empresariais ou problemas reais, sendo necessário para isto elaborar novos estudos para ampliar a validade externa. Entretanto, os resultados do estudo podem servir como possíveis indícios de viabilidade de utilizar *UbiCheck* para apoiar a atividade de definição de requisitos de ubiquidade em projetos de software ubíquo.
- **Validade de constructo.** Para obter validade de constructo precisamos verificar que os tratamentos representem bem a construção da causa e que a saída do experimento represente bem a construção do efeito. Este estudo possui dois fatores e dois tratamentos e a causa pode ser mapeada no uso ou não de *UbiChek*. A saída, por sua vez, está ligada diretamente ao efeito, uma vez que ambos podem ser extraídos a partir dos valores das variáveis dependentes.
- **Validade de conclusão.** O propósito do estudo é a análise de viabilidade. Como se trata de apenas um grupo em cada tratamento nenhum teste estatístico (como *t-test*) poderá ser aplicado e os resultados deverão ser tratados apenas como indícios preliminares de viabilidade. Assim, a validade de conclusão está atrelada à replicação do estudo em outros contextos para viabilizar a aplicação de testes estatísticos e aumentar a validade externa do estudo.

7.2.3 Operação

Para executar esse estudo, foram convidados alunos de pós-graduação que estavam participando da disciplina Engenharia de Software Orientada a Objetos no segundo trimestre de 2009 na Universidade Federal do Rio de Janeiro. Ao todo, foram considerados 8 alunos no estudo. Tais estudantes relataram possuir experiência em Engenharia de Software, incluindo as atividades de especificação de requisitos, conforme pode ser observado na Tabela 7.4 que descreve a caracterização dos participantes referente às atividades de especificação de requisitos.

Tabela 7.4. Caracterização dos Participantes

Grupo	Participante	Experiência escrevendo requisitos	Experiência escrevendo casos de uso	Experiência revisando requisitos	Experiência revisando casos de uso	Experiência modificando requisitos para manutenção
1	1	3	3	3	3	3
	2	4	4	3	3	4
	3	5	5	4	4	5
2	4	4	4	4	4	4
	5	3	3	3	3	3
	6	4	4	4	4	4
3	7	5	5	5	5	5
	8	3	3	3	3	3

Legenda:

1 = nenhum

2 = estudei em aula ou em livro

3 = pratiquei em 1 projeto em sala de aula

4 = usei em 1 projeto na indústria

5 = usei em vários projetos na indústria

Cada participante assinou um termo de consentimento autorizando que o material produzido fosse utilizado nesta pesquisa. Os participantes também preencheram um formulário de caracterização para avaliar o grau de conhecimento e a experiência de cada um com projetos de software.

Os participantes foram divididos em três grupos (1, 2 e 3) – os grupos 1 e 2 foram formados por três alunos e o grupo 3 por dois alunos. A escolha dos participantes de cada equipe foi feita tomando como referência as respostas dos formulários de caracterização e teve o objetivo de manter as equipes semelhantes quanto ao conhecimento e a experiência de cada participante.

As equipes receberam uma versão correta (inspeção prévia) do documento de requisitos do Sistema de Gestão de Inventário Patrimonial (SGP). Neste documento não havia requisitos de ubiquidade, portanto, as equipes foram solicitadas a estender o documento para contemplar dois novos cenários (Bicicletário e Item Rastreável) que demandariam o uso da computação ubíqua.

A definição dos requisitos de cada novo cenário do SGP foi feita de forma sequencial, a primeira sem o apoio e a segunda com o apoio de *UbiCheck*. Sendo assim, na primeira interação o grupo 1 definiu requisitos para o cenário Bicicletário e os grupos 2 e 3 definiram requisitos para o cenário Controle Item Rastreável. Na segunda interação, o grupo 1 definiu requisitos para o cenário Item Rastreável e os grupos 2 e 3 definiram requisitos para o cenário Bicicletário, dessa vez com o uso de *UbiCheck*. A Tabela 7.1 resume essas interações.

Na segunda interação, o Guia para Definição de Requisitos de Ubiquidade foi entregue preparado e configurado para as necessidades de cada cenário. Dessa forma, bastava o desenvolvedor responder as perguntas e preencher o documento de requisitos de software.

Durante a execução de cada interação, as equipes responderam a questionários que tinham o propósito de capturar as dificuldades encontradas para definir os requisitos de ubiquidade. Os resultados deste estudo foram computados a partir das informações obtidas destes questionários e do documento de requisitos elaborado por cada equipe, conforme descrito na seção seguinte.

7.2.4 Análise e Interpretação dos Dados

Conforme mencionado, o objetivo deste estudo é verificar se o Guia para Definição de Requisitos de Ubiquidade conseguiria ser aplicado em um projeto de software ubíquo. Nesse sentido, foram analisadas as respostas dos questionários preenchidos por cada grupo. Ao total, foram preenchidos 8 formulários de avaliação. As respostas indicaram que o uso da abordagem facilitou a definição dos requisitos, pois as perguntas fizeram com que os desenvolvedores refletissem sobre assuntos que eles normalmente não capturariam no documento de requisitos. Ainda segundo os participantes, isso se deve ao fato de que a computação ubíqua se trata de um domínio bastante específico.

Adicionalmente, nos questionários foram registrados os tempos que cada grupo gastou para definir os requisitos de cada cenário, conforme mostra a Tabela 7.5 e gráfico da Figura 7.1.

Tabela 7.5. Tempo gasto por cada grupo para definir os requisitos de ubiquidade

	Tempo Gasto em cada Interação	
	1ª Rodada (sem apoio)	2ª Rodada (*UbiCheck*)
Grupo 1	300 minutos Bicicletário	240 minutos Item Rastreável
Grupo 2	240 minutos Item Rastreável	120 minutos Bicicletário
Grupo 3	180 minutos Item Rastreável	180 minutos Bicicletário

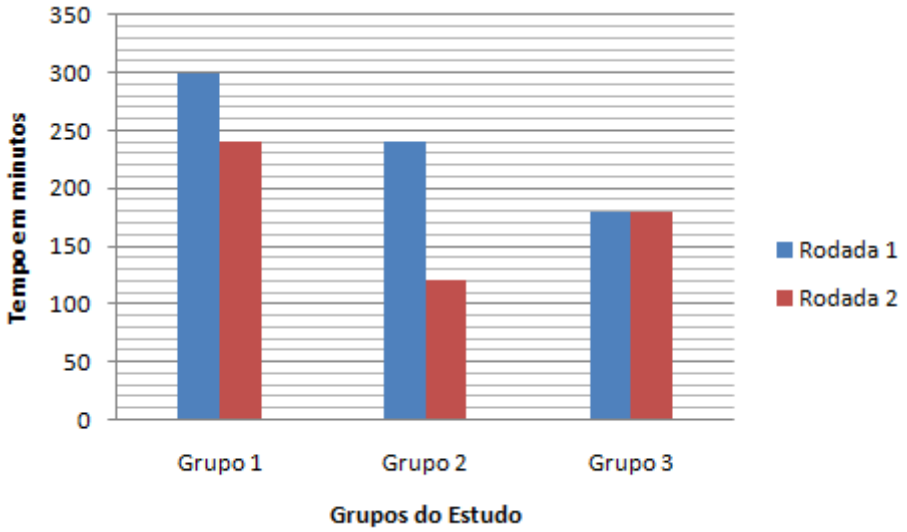


Figura 7.1. Tempo gasto por cada grupo para definir os requisitos de ubiqüidade

Os tempos registrados sugerem que aplicar *UbiCheck* pode não onerar o processo de desenvolvimento. Comparando o resultado dos grupos 1 e 2 (possuíam o mesmo número de alunos), nota-se que na segunda rodada, com o uso de *UbiCheck*, houve redução de 60% para o cenário Bicicletário e o tempo para o cenário Item Rastreável se manteve constante.

Contudo, a análise desses questionários apontou que algumas das perguntas do Guia para Definição de Requisitos de Ubiquidade foram difíceis de responder. Na maior parte das vezes, a dificuldade estava relacionada ao não entendimento de um termo contido na pergunta ou a falta de indicação de como registrar um requisito que respondesse aquela pergunta. Em alguns casos, algumas perguntas não se aplicavam adequadamente no momento da definição de requisitos, pois tratavam de assuntos que seriam melhor tratados em etapas posteriores do desenvolvimento.

Mesmo com essa limitação, o resultado indica que o Guia para a Definição de Requisitos de Ubiquidade conseguiu ser utilizado pelos desenvolvedores para apoiar o desenvolvimento de um projeto de software ubíquo.

Cabe observar que existem riscos à validade deste estudo e os resultados aqui apresentados são limitados, portanto, não podem ser generalizados. A repetição do estudo com uma população mais abrangente e maior rigor de execução deve ser realizada visando fortalecer a observação deste comportamento. Algumas ameaças a validade são:

- o estudo foi realizado com alunos de pós graduação em uma turma de Engenharia de Software Orientada a Objetos, os quais não possuem o mesmo grau de experiência e conhecimento. Tentou-se mitigar esse risco definindo grupos equilibrados;
- o número de participantes para este estudo foi considerado baixo, o que impossibilita a generalização dos resultados obtidos, e;
- sob a perspectiva da validade de conclusão, identificou-se a necessidade de se repetir o estudo com objetivo de se obter mais pontos para análise, o que poderia aumentar a confiança nos resultados alcançados.

Por fim, a observação da utilização de *UbiCheck* por parte das equipes nos permitiu observar que, a principio, *UbiCheck* pode ser utilizada e, ao mesmo tempo, identificar oportunidades de melhoria na abordagem proposta, que serão objeto de discussão da próxima seção.

7.3 – *UbiCheck* - Versão 2.0

Na seção anterior foram descritos os resultados do estudo de observação relacionado à utilização de *UbiCheck* em um projeto de software ubíquo. Embora a abordagem tenha se mostrado viável para apoiar a definição de requisitos de ubiquidade em projetos de software, os participantes do estudo relataram as seguintes limitações:

- algumas perguntas não se encaixavam bem na etapa da definição de requisitos, sendo mais adequado aplicá-las para fases posteriores do desenvolvimento do software;
- algumas perguntas não foram respondidas porque não se sabia como escrever um requisito que atendesse à demanda explicitada pelo questionamento; e
- algumas perguntas eram difíceis de entender, pois utilizavam termos muito específicos do domínio da computação ubíqua.

Essas limitações motivaram o aprimoramento de *UbiCheck*. Nesse sentido, foi desenvolvida a versão 2.0 da abordagem. A Figura 7.2 apresenta uma visão geral de *UbiCheck* 2.0, indicando: (1) os responsáveis pela execução de cada etapa; (2) as etapas consideradas; (3) as atividades executadas em cada etapa; e (4) as atividades alteradas (marcadas em cinza) e inseridas (marcadas em preto) na nova versão da abordagem. A seguir cada atividade é resumida:

Etapa 1: Configuração de *UbiCheck*

- **Elaborar Modelos:** essa atividade não foi alterada na nova versão de *UbiCheck*.
- **Elaborar Guia Geral de Definição de Requisitos de Ubiquidade:** essa atividade foi alterada para que durante a elaboração das perguntas, o especialista em ubiquidade pudesse definir a etapa do projeto de desenvolvimento em que cada pergunta deveria ser aplicada;
- **Elaborar Direcionamento da Resposta:** essa atividade foi inserida para que o especialista pudesse explicar o que é esperado como resposta para cada pergunta do guia para definição de requisitos de ubiquidade;
- **Elaborar Glossário:** essa atividade foi inserida para que os termos utilizados no guia para definição de requisitos de ubiquidade pudessem ser explicados, o que consequentemente deve melhorar o entendimento de suas perguntas.

Etapa 2: Definição de Requisitos de Ubiquidade

- **Especializar Guia Geral de Definição de Requisitos de Ubiquidade:** essa atividade não foi alterada na nova versão de *UbiCheck*;
- **Definir Requisitos de Ubiquidade do Projeto:** essa atividade foi alterada para que novas orientações fossem fornecidas para os desenvolvedores que vão aplicar *UbiCheck* em um projeto de software ubíquo.

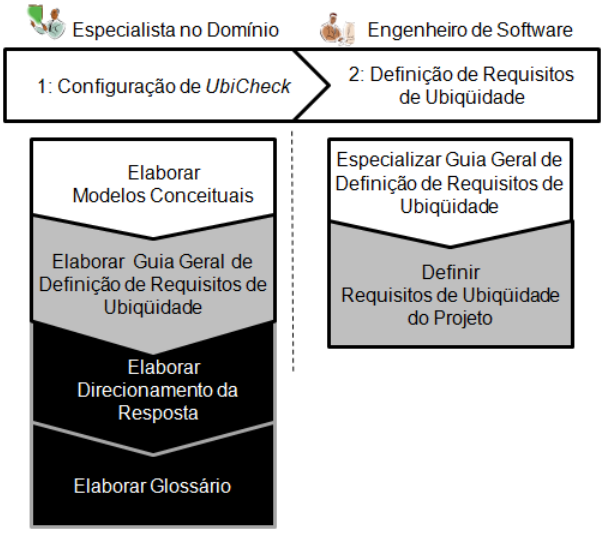


Figura 7.2. Visão geral de UbiCheck 2.0

Na seção 7.3.1 as atividades que foram alteradas ou inseridas na nova versão de *UbiChek* são explicadas.

7.3.1. Atividades Alteradas e Inseridas em *UbiCheck* 2.0

7.3.1.1 Elaborar Guia Geral de Definição de Requisitos de Ubiquidade

Embora os fatores das características de ubiquidade capturem comportamentos que podem ocorrer em softwares ubíquos, alguns deles refletem conceitos tecnológicos que são mais importantes nas etapas de arquitetura e de projeto do que na definição de requisitos.

Por esse motivo, após as perguntas do guia para definição de requisitos de ubiquidade serem elaboradas, é importante classificá-las de acordo com a etapa do desenvolvimento em que cada uma melhor se aplica. Como *UbiCheck* é uma abordagem para apoiar a definição de requisitos, optou-se por classificar as perguntas como aplicáveis em requisitos e aplicáveis em outras etapas do desenvolvimento.

Nesse sentido, o especialista deve avaliar cada pergunta elaborada e definir em qual dessas duas categorias ela se enquadra. Para simplificar o entendimento, são fornecidos alguns exemplos de perguntas e suas respectivas classificações, bem como o motivo pelo qual aquela classificação foi atribuída:

- **Pergunta:** Quais os serviços relevantes para o sistema?
- **Classificação:** aplicável em requisitos
- **Motivo:** Essa é uma pergunta sobre funcionalidades de um sistema que devem ser providas como serviços, portanto, adequada a definição de requisitos.

- **Pergunta:** Como o *container* do serviço é considerado?
- **Classificação:** aplicável em outras etapas.
- **Motivo:** A escolha de um *container* é uma atividade de arquitetura, portanto, na definição de requisitos não é necessário especificar esse tipo de informação, pois a forma como cada serviço é considerado em um *container* pode variar de acordo com a escolha da tecnologia.

- **Pergunta:** Como fazer *cache* dos serviços?
- **Classificação:** aplicável em outras etapas.
- **Motivo:** Cache é tipicamente uma atividade de projeto para melhorar o desempenho de uma aplicação, portanto, não é necessário definir como fazer cache de serviços na definição de requisitos.

É importante destacar que as perguntas consideradas não aplicáveis em requisitos serão descartadas do guia para definição de requisitos. Adicionalmente, é importante ressaltar que as perguntas que compõem o guia para definição de requisitos disponível no Apêndice F já se encontram classificadas.

7.3.1.2 Elaborar Direcionamento da Resposta

As perguntas que compõem o guia para definição de requisitos de ubiquidade têm o objetivo de destacar as informações importantes que devem ser capturadas nos requisitos de ubiquidade. Contudo, no estudo apresentado na seção 7.2, percebeu-se que para apoiar a definição de requisitos é importante que também seja fornecido um direcionamento sobre como essas informações devem ser registradas. Nesse sentido, é importante definir os seguintes direcionamentos:

- **Item de Especificação:** o tipo de informação que deve ser definido para responder a pergunta, por exemplo, um requisito funcional, uma regra de negócio, um caso de uso, dentre outros.
- **Orientação:** detalhamento de como o item de especificação deve ser capturado.

Para facilitar o entendimento, a seguir são fornecidos alguns exemplos de direcionamentos:

- **Pergunta:** Quais os usuários relevantes para o sistema?
- **Item de Especificação:** Mapa de Atores

- **Orientação:** Definir os atores que interagem com o sistema.

- **Pergunta:** Como as informações do usuário são consideradas?
- **Item de Especificação:** Requisito Funcional
- **Orientação:** Definir as informações de interação do usuário que devem ser consideradas pelo sistema.

- **Pergunta:** Como avaliar a utilidade das informações de contexto?
- **Item de Especificação:** Passo de Caso de Uso, Regra de Negócio
- **Orientação:** Definir na captura da informação de contexto, como ela deve ser avaliada no que diz respeito a sua utilidade.

- **Pergunta:** Como a fonte de dados influencia a consolidação de informações de contexto?
- **Item de Especificação:** Regra de Negócio
- **Orientação:** Definir um critério para consolidar informações de acordo com a fonte de dados.

Esta nova versão do guia geral para definição de requisitos de ubiquidade está no Apêndice K.

7.3.1.3 Elaborar Glossário

As perguntas do guia para definição de requisitos de ubiquidade foram elaboradas com base nos fatores das características de ubiquidade. Como esses fatores utilizam termos específicos do domínio da computação ubíqua, as perguntas também contêm esses termos.

Como a idéia é que *UbiCheck* possa ser aplicada por engenheiros de software e não por especialistas em computação ubíqua, foi considerado necessário definir um glossário com os termos utilizados para facilitar o entendimento das perguntas.

Nesse sentido, foram consideradas as entidades que compõem os modelos das características de ubiquidade, pois elas definem os assuntos das perguntas do guia para definição de requisitos de ubiquidade e representam os principais conceitos presentes em cada fator. Sendo assim, para cada entidade dos modelos de ubiquidade disponível no Apêndice D foi elaborada uma definição. O glossário de termos utilizado em *UbiCheck* é o mesmo utilizado em *UbiVeri* e pode ser encontrado no Apêndice G.2.

7.3.1.4 Definir Requisitos de Ubiquidade do Projeto

O uso de *UbiCheck* para apoiar a definição de requisitos foi alterado devido à inserção dos direcionamentos nas perguntas do guia para definição de requisitos de ubiquidade e pela disponibilidade de um glossário com os principais termos abordados nessas perguntas.

Enquanto na primeira versão de *UbiCheck* não haviam indicações de como proceder para registrar os requisitos, na versão 2.0, a inclusão dos direcionamentos nas perguntas permite que seja sugerido como e onde esses requisitos podem ser definidos, por exemplo, através de um caso de uso, de uma funcionalidade, de um requisito funcional, dentre outros.

Dessa forma, além da abordagem mostrar para o engenheiro de software quais informações devem ser capturadas nos requisitos de ubiquidade, ela o orienta sobre como elas devem ser capturadas. Adicionalmente, o engenheiro de software tem a possibilidade de consultar um glossário de termos quando tiver dificuldade em entender o assunto de uma pergunta.

Sendo assim, é possível que as melhorias acrescidas em *UbiCheck 2.0* possam simplificar e fornecer mais apoio para os engenheiros de software durante a fase de definição dos requisitos de ubiquidade em um projeto de desenvolvimento de software ubíquo.

7.4 – Considerações Finais

Este capítulo apresentou inicialmente o plano e o resultado do estudo experimental para avaliar os benefícios que o uso de *UbiCheck* traz. A observação da utilização de *UbiCheck* nos permitiu observar que:

- ela pode ser utilizada pelos engenheiros de software para apoiar o desenvolvimento de um projeto de software ubíquo;
- os tempos registrados sugerem que aplicar *UbiCheck* pode não onerar o processo de desenvolvimento.

Em seguida, foram apresentadas as mudanças realizadas em *UbiCheck* com o intuito de suprir deficiências identificadas na abordagem durante o estudo. Os objetivos dessas mudanças foram: (1) remover do guia aquelas perguntas que são mais adequadas em fases de arquitetura e projeto do software; (2) tornar mais claro o que é esperado como resposta para cada pergunta; e (3) melhorar o entendimento sobre o que é discutido em cada pergunta.

O próximo passo é a avaliação experimental da abordagem de apoio à verificação de requisitos de ubiquidade, tema do próximo capítulo.

Capítulo 8 - Avaliação de *UbiVeri*

Este capítulo descreve o plano e o resultado do estudo experimental para avaliar os benefícios que o uso de UbiVeri, abordagem de apoio à verificação de requisitos de ubiquidade, traz para a identificação de defeitos em requisitos de projetos de software ubíquo.

8.1 - Introdução

Nesta pesquisa foi realizado um estudo experimental para caracterizar *UbiVeri* no que diz respeito a não oneração da atividade de revisão (em tempo), aumento no número de defeitos identificados e aumento da satisfação do usuário ao revisar requisitos de ubiquidade em projetos de software ubíquo. O plano do estudo e os resultados de sua execução serão apresentados neste capítulo.

8.2 – Avaliação do uso de *UbiVeri*

Neste capítulo, a apresentação do estudo experimental também seguirá o padrão sugerido por Wohlin *et al.* (2000) para descrever um plano de um estudo experimental. Dessa forma, teremos quatro seções: definição do experimento, planejamento do estudo experimental, operação do estudo experimental, análise e interpretação dos dados. O pacote do estudo experimental está disponível na COPPE/UFRJ e seus instrumentos podem ser encontrados no Apêndice J.

8.2.1 Definição do Estudo Experimental

No contexto desta pesquisa, o objetivo deste estudo é avaliar aspectos de interesse para engenheiros de software considerando a abordagem de apoio à verificação de requisitos de ubiquidade descrita no Capítulo 6. Estes aspectos são: número de discrepâncias identificadas ao revisar a especificação de requisitos, número de falsos positivos reportados, número de defeitos identificados na revisão da especificação dos requisitos do projeto, esforço envolvido no uso da abordagem e satisfação do usuário no uso da abordagem.

Utilizando o GQM (BASILI *et al*., 1994), o objetivo do estudo experimental pode ser definido da seguinte forma:

Analisar a verificação dos requisitos de ubiquidade com e sem o uso de *UbiVeri*

Com o propósito de caracterizar

Com respeito ao esforço (em tempo), número de discrepâncias identificadas, número de falsos positivos reportados, número de defeitos identificados e satisfação do usuário

No ponto de vista de engenheiros de software

No contexto do projeto de um Sistema de Gestão de Inventário Patrimonial (SGP).

Assim, pretende-se avaliar, através da coleta e análise de dados quantitativos e qualitativos durante a operação do estudo experimental, as seguintes questões gerais:

- Q1: O uso de *UbiVeri* impacta o número de defeitos identificados durante a verificação de requisitos de ubiquidade do projeto quando comparado à identificação de defeitos de forma *ad hoc*?
    - Q1.1: O uso de *UbiVeri* impacta o número de discrepâncias identificadas durante a verificação de requisitos de ubiquidade do projeto quando comparado à verificação de requisitos de forma *ad hoc*?
    - Q1.2: O uso de *UbiVeri* impacta o número de falsos positivos durante a verificação de requisitos de ubiquidade do projeto quando comparado ao número de falsos positivos reportados na revisão de forma *ad hoc*?
- Q2: O uso de *UbiVeri* onera o esforço em tempo gasto durante a atividade verificação de requisitos de ubiquidade do projeto quando comparado à verificação de requisitos de forma *ad hoc*?
- Q3: O uso de *UbiVeri* impacta o grau de satisfação dos inspetores na verificação dos requisitos de ubiquidade quando comparada à verificação dos requisitos de forma *ad hoc*?

8.2.2 Planejamento do Estudo Experimental

Seleção do Contexto

De acordo com (WOHLIN *et al.*, 2000), o contexto pode ser caracterizado em quatro dimensões:

- **processo:** on-line / **off-line**;

- **participantes: <u>alunos</u>** / profissionais;
- **realidade:** problema real / **<u>modelado</u>**;
- **generalidade: <u>específico</u>** / geral.

Este estudo se propõe a ser executado de forma *off-line*, uma vez que as atividades serão distribuídas aos participantes e será definido um prazo para sua conclusão. Os participantes utilizados no estudo serão alunos do sétimo período da disciplina Engenharia de Software Orientada a Objetos do curso de Engenharia da Computação da Universidade Federal do Rio de Janeiro atuando em um problema modelado. A generalidade do estudo é específica.

Formulação das Hipóteses

A seguir são apresentadas as hipóteses nula e alternativas deste experimento:

- **Hipótese nula (H_0):** O uso de *UbiVeri* não impacta o número de defeitos identificados, discrepâncias identificadas e falso positivos reportados ao mesmo tempo em que onera a atividade de revisão dos requisitos quando comparado ao uso de uma abordagem *ad hoc* na atividade de verificação dos requisitos de ubiquidade de um projeto de software. Isto significa que:
    - **(H_{01}):** Não existe diferença na atividade de verificação de requisitos de ubiquidade usando *UbiVeri* e uma abordagem *ad hoc* ao se considerar as variáveis número de discrepâncias identificadas, falsos positivos reportados ou defeitos identificados.
    - **(H_{02}):** Não existe diferença na atividade de verificação de requisitos de ubiquidade usando *UbiVeri* e uma abordagem *ad hoc* ao se considerar a variável satisfação do usuário.
    - **(H_{03}):** O esforço em tempo para realizar a atividade de verificação de requisitos de ubiquidade com *UbiVeri* é maior quando comparado com uma abordagem *ad hoc*.

Sendo assim, tem-se que:

$\mathbf{H_0: (\Omega_a = \Omega_u) \vee (D_a = D_u) \vee (F_a = F_u) \vee (\mu_a = \mu_u) \vee (\iota_a < \iota_u)}$, onde:

- $\mathbf{\Omega_a}$ é a quantidade de defeitos identificados nos requisitos de ubiquidade utilizando uma abordagem de verificação *ad hoc*.

- Ω_u é a quantidade de defeitos identificados nos requisitos de ubiquidade utilizando *UbiVeri*.
- D_a é a quantidade de discrepâncias identificadas nos requisitos de ubiquidade utilizando uma abordagem de verificação *ad hoc*.
- D_u é a quantidade de discrepâncias identificadas nos requisitos de ubiquidade utilizando *UbiVeri*.
- F_a é a quantidade de falsos positivos reportados na revisão dos requisitos de ubiquidade utilizando uma abordagem de verificação *ad hoc*.
- F_u é a quantidade de falsos positivos reportados na revisão dos requisitos de ubiquidade utilizando *UbiVeri*.
- μ_a é o grau de satisfação dos participantes na atividade de verificação de requisitos de ubiquidade de forma *ad hoc*.
- μ_u é o grau de satisfação dos participantes na atividade de verificação de requisitos de ubiquidade utilizando *UbiVeri*.
- ι_a é o tempo gasto na atividade de verificação de requisitos de ubiquidade de forma *ad hoc*.
- ι_u é o tempo gasto na atividade de verificação de requisitos de ubiquidade utilizando *UbiVeri*.

- **Hipótese alternativa (H_1):** O uso de *UbiVeri* aumenta a quantidade de defeitos identificados no documento de requisitos de ubiquidade se comparada ao uso de uma abordagem *ad hoc*.

$$H_1 : \Omega_a < \Omega_u$$

- **Hipótese alternativa (H_2):** O uso de *UbiVeri* aumenta a quantidade de discrepâncias identificadas no documento de requisitos de ubiquidade se comparada ao uso de uma abordagem *ad hoc*.

$$H_2 : D_a < D_u$$

- **Hipótese alternativa (H_3):** O uso de *UbiVeri* reduz a quantidade de falsos positivos reportados na revisão do documento de requisitos de ubiquidade se comparada ao uso de uma abordagem *ad hoc*.

$$H_3 : F_a \geq F_u$$

- **Hipótese alternativa (H_4):** O uso de *UbiVeri* não onera o esforço de realização da atividade de verificação dos requisitos de ubiquidade se comparada ao uso de uma abordagem *ad hoc*.

$$H_4 : \iota_a \geq \iota_u$$

- **Hipótese alternativa (H_5):** O uso de *UbiVeri* leva a um grau de satisfação maior dos participantes na verificação de requisitos de ubiquidade se comparada à verificação de requisitos de ubiquidade de forma *ad hoc*.

$$H_5 : \mu_a < \mu_u$$

Seleção de Variáveis

O estudo experimental utiliza as seguintes variáveis:

- Variáveis independentes:
  - Abordagem de Apoio à Verificação de Requisitos - *UbiVeri*.
  - Domínio da Aplicação (Sistema de Gestão de Patrimônio).
- Variáveis dependentes:
  - **Número de discrepâncias identificadas durante a verificação dos requisitos:** a partir da especificação dos requisitos será executada uma verificação considerando a caracterização do projeto ubíquo. As discrepâncias identificadas durante a verificação serão relatadas através de um formulário para relato de discrepância.
  - **Número de falsos positivos reportados:** os falsos positivos serão identificados a partir da revisão da lista de discrepâncias reportada por cada participante do estudo.
  - **Número de defeitos identificados durante a verificação dos requisitos:** os defeitos serão identificados a partir da revisão da lista de discrepâncias reportada por cada participante do estudo.
  - **O tempo gasto na verificação dos requisitos:** os participantes preencherão os Formulários UB_1 e UB_2 (apresentados no Apêndice J.3.1 e J.3.2) informando o tempo gasto nas diferentes atividades.
  - **O grau de satisfação dos participantes na verificação dos requisitos:** os participantes preencherão o Formulário UB_3 (apresentado no Apêndice J.4), informando o nível de satisfação na aplicação da abordagem.

Seleção de Participantes

Os participantes selecionados serão representados por alunos do sétimo período do curso de Engenharia de Computação da Universidade Federal do Rio de Janeiro matriculados na disciplina Engenharia de Software Orientada a Objetos. Sua caracterização será realizada através do questionário de caracterização que conterá perguntas a respeito de sua experiência prática com verificação de requisitos e formação acadêmica.

Os participantes devem então ser divididos em quatro grupos utilizando o questionário de caracterização para formar uma divisão de grupos equilibrada em termos de experiência dos participantes. Cada grupo deverá conter dois participantes.

Esta divisão equilibrada facilita o uso de agrupamento (*blocking*) (WOHLIN *et al.*, 2000) para eliminar fatores de confusão na análise dos resultados.

Projeto do Estudo Experimental

O projeto deste estudo experimental envolve dois fatores: (1) verificação de requisitos de ubiquidade de um projeto de software ubíquo; (2) domínio de aplicação (que no contexto deste estudo tratará de um sistema de gestão de patrimônio).

O fator verificação de requisitos de ubiquidade de um projeto de software ubíquo terá dois tratamentos:

- **Verificação de Requisitos de Ubiquidade de forma *ad hoc*:** os participantes receberão, para efetuar a verificação dos requisitos de ubiquidade do projeto, a especificação dos requisitos acompanhada de um glossário de termos em computação ubíqua.
- **Verificação de Requisitos de Ubiquidade usando *UbiVeri*:** os participantes receberão, para efetuar a verificação dos requisitos de ubiquidade do projeto, a especificação dos requisitos, o guia de apoio à verificação de requisitos de ubiquidade e o glossário de termos em computação ubíqua.

O fator domínio de aplicação irá considerar dois cenários (no Apêndice I estão disponíveis informações sobre os cenários de ubiquidade distribuídos):

- **Sistema de Gestão Patrimonial (SGP) + Item de Patrimônio Rastreável (Cenário 1):** Extensão ao SGP com objetivo de controlar de forma efetiva a entrada e saída de itens rastreáveis na organização.

- **Sistema de Gestão Patrimonial + Bicicleta Rastreável (Cenário 2):** Extensão ao SGP com objetivo de controlar de forma efetiva o uso de bicicletas na organização.

Embora façam parte do mesmo domínio (gestão patrimonial), os dois cenários estão sendo tratados neste trabalho como dois tratamentos. Isto por que eles implicam nas mesmas preocupações no projeto do estudo experimental que deveriam ser consideradas caso tivéssemos dois tratamentos distintos para o fator domínio da aplicação.

Os cenários de ubiquidade, item patrimonial e bicicleta rastreável, escolhidos para o estudo fazem parte de uma série de requisitos do sistema de gestão patrimonial da COPPE/UFRJ. Sendo assim, ambos tratam de um problema real no contexto da referida Universidade. Esta demanda interna da Universidade acabou sendo utilizada como critério para escolha dos cenários de ubiquidade do estudo.

O experimento considera a seguinte divisão de grupos:

- **Grupo 1:** fará a verificação dos requisitos de forma *ad hoc* para os dois cenários de ubiquidade. Inicialmente, será revisada a especificação dos requisitos para o cenário 2 (bicicletário). Na sequência, será considerado o cenário 1 (item rastreável).
- **Grupo 2:** fará, inicialmente, uma verificação *ad hoc* dos requisitos para o cenário 1 (item rastreável). Em seguida, fará a revisão do documento de requisitos considerando o cenário 2 (bicicletário) de ubiquidade utilizando *UbiVeri*.
- **Grupo 3:** fará, inicialmente, uma verificação *ad hoc* dos requisitos para o cenário 2 (bicicletário). Em seguida, fará a revisão do documento de requisitos considerando o cenário 1 (item rastreável) de ubiquidade utilizando *UbiVeri*.
- **Grupo 4:** fará a verificação dos requisitos utilizando *UbiVeri* para os dois cenários de ubiquidade. Inicialmente será revisada a especificação dos requisitos para o cenário 2 (bicicletário). Na sequência, será considerado o cenário 1 (item rastreável).

A Tabela 8.1 apresenta a distribuição das atividades considerando estes quatro grupos.

Tabela 8.1. Alocação das abordagens de verificação por grupos e dos cenários de ubiquidade

	Grupo 1		**Grupo 2**		**Grupo 3**		**Grupo 4**	
Rodada 1	*Ad Hoc*	C_2	*Ad Hoc*	C_1	*Ad Hoc*	C_2	*UbiVeri*	C_2
Rodada 2	*Ad Hoc*	C_1	*UbiVeri*	C_2	*UbiVeri*	C_1	*UbiVeri*	C_1

Procedimento do Estudo Experimental

No estudo experimental, os participantes dos dois tratamentos recebem um problema modelado, que é a identificação de defeitos em requisitos de ubiquidade para o Sistema de Gestão Patrimonial. Os participantes receberão a especificação de requisitos do sistema e realizarão a verificação dos requisitos com objetivo de identificar defeitos associados às características de ubiquidade do projeto. A atividade de verificação será feita de forma assíncrona e contará com a participação de um moderador que acompanhará o experimento *off-line*.

Além da especificação dos requisitos do software e do glossário de termos em ubiquidade (que serão disponibilizados para os dois grupos), o grupo que utilizará *UbiVeri* receberá também o guia de apoio à verificação de requisitos de ubiquidade em projetos de software ubíquo.

Ao final da verificação dos requisitos, os participantes devem preencher um questionário de acompanhamento para que possam fornecer informações qualitativas a respeito da verificação dos requisitos de ubiquidade, incluindo esforço envolvido e grau de satisfação na realização da tarefa.

O experimento foi organizado em duas sessões, como mostrado na Tabela 8.2. Inicialmente, para todos os grupos, será apresentado o trabalho e será realizado um treinamento sobre os principais conceitos associados à ubiquidade computacional. O objetivo deste treinamento é apresentar o domínio da computação ubíqua para os envolvidos no estudo.

Feito isto, os Grupos 1, 2 e 3 receberão o pacote do estudo para realização da verificação de requisitos de ubiquidade de forma *ad hoc*. Em paralelo, o Grupo 4 será treinado no uso de *UbiVeri* e receberá o pacote do estudo para realização da verificação de requisitos de ubiquidade utilizando *UbiVeri*.

Os grupos terão o prazo de uma semana para realização desta atividade e ao seu término deverão entregar o formulário dos defeitos identificados na especificação dos requisitos, assim como os demais formulários disponibilizados no pacote do estudo.

Encerrada a primeira rodada do estudo, será dado início à segunda sessão. Para esta sessão, o Grupo 1 receberá um novo cenário de ubiquidade junto ao pacote do estudo para realização da verificação de requisitos de ubiquidade de forma *ad hoc.* Já os Grupos 2 e 3 serão treinados no uso de *UbiVeri* e, na sequência, receberão junto ao Grupo 4 um novo cenário e o pacote do estudo para realização da verificação de requisitos de ubiquidade utilizando *UbiVeri*. A distribuição dos diferentes cenários de ubiquidade utilizados está definida na Tabela 8.1.

Maiores detalhes sobre as atividades realizadas em cada sessão podem ser encontrados na Tabela 8.3.

Tabela 8.2. Procedimento do Estudo Experimental

	Sessão 1		**Sessão 2**	
Grupo 1	Explicação sobre o Trabalho + Treinamento em Computação Ubíqua	Verificação de Requisitos *Ad Hoc*	-	Verificação de Requisitos *Ad Hoc*
Grupo 2	Explicação sobre o Trabalho + Treinamento em Computação Ubíqua	Verificação de Requisitos *Ad Hoc*	Treinamento sobre *UbiVeri*	Verificação de Requisitos com *UbiVeri*
Grupo 3	Explicação sobre o Trabalho + Treinamento em Computação Ubíqua	Verificação de Requisitos *Ad Hoc*	Treinamento sobre *UbiVeri*	Verificação de Requisitos com *UbiVeri*
Grupo 4	Explicação sobre o Trabalho + Treinamento em Computação Ubíqua + Treinamento sobre *UbiVeri*	Verificação de Requisitos com *UbiVeri*		Verificação de Requisitos com *UbiVeri*

Cronograma para Realização do Treinamento e Explicação sobre o Estudo Experimental

O estudo experimental foi realizado entre 04/2010 e 05/2010, como mostrado na Tabela 8.3.

Tabela 8.3. Cronograma do Estudo Experimental

Sessão	Mês	Atividades
1	04/2010	1. Explicação sobre o trabalho e apresentação do projeto. 2. Entrega dos cenários a serem considerados para as equipes. 3. Entrega dos formulários a serem preenchidos.
	04/2010	1. Execução da verificação dos requisitos de ubiquidade para o projeto.
2	04/2010	1. Recolhimento dos formulários referentes à primeira parte do estudo. 2. Explicação sobre o trabalho e apresentação do projeto. 3. Entrega dos formulários a serem preenchidos.
	04/2010 a 05/2010	1. Execução da verificação dos requisitos de ubiquidade para o projeto.
	05/2010	1. Recolhimento dos formulários referentes à segunda parte do estudo.

Instrumentação

Para a realização do estudo experimental, a instrumentação conta com a especificação de requisitos do Sistema de Gestão Patrimonial acrescido de dois cenários de ubiquidade: item de patrimônio rastreável e bicicletário.

Assim, além da especificação de requisitos, também será disponibilizada a descrição dos cenários de ubiquidade envolvidos e um glossário de termos em ubiquidade. Em específico, o tratamento que fará uso de *UbiVeri* também possui o guia de apoio à verificação de requisitos de ubiquidade.

Além destes itens, o experimento também conta com os seguintes instrumentos:

- Um formulário de consentimento (Apêndice J.1).
- Um questionário de caracterização (Apêndice J.2).
- A descrição da tarefa a ser desempenhada pelos participantes (Apêndice J.3).
- Um questionário de acompanhamento para possibilitar a análise qualitativa (veja Apêndice J.4).

Análise dos Dados

Inicialmente, para cada variável analisada, será feita a análise de *outlier* para eliminar dos dados possíveis participantes do estudo que distorçam os resultados. Feito isto, será verificado se os dados possuem uma distribuição normal. Caso

positivo, será utilizado um método paramétrico para análise de variância (ANOVA). Em caso negativo, será utilizado um método não paramétrico (Wilcoxon).

O objetivo destes métodos será determinar se a diferença entre as médias das variáveis de resposta nos grupos estabelecidas pela combinação de níveis de fatores é estatisticamente significante. Devido à limitação na quantidade de dados para análise, definiu-se nesta pesquisa que o nível de significância seria de 10%.

Para apoiar na execução de ANOVA e Wilcoxon, neste estudo, foi utilizada a ferramenta JMP 4.

Avaliação da Validade

A validade interna, externa e de constructo do experimento se encontra discutida a seguir:

- **Validade interna.** Como se trata de um estudo envolvendo mais de um grupo, onde cada grupo é submetido a um tratamento distinto, a maior ameaça de validade interna é a relação dos resultados com a seleção dos participantes do estudo. Para evitar esta ameaça e aumentar a validade interna, o estudo deverá utilizar os dados do questionário de caracterização para a divisão dos participantes em grupos de acordo com suas características individuais.
- **Validade externa.** O estudo envolve alunos do sétimo período do curso de Engenharia de Computação da Universidade Federal do Rio de Janeiro matriculados na disciplina Engenharia de Software Orientada a Objetos no primeiro semestre de 2010 e utiliza um problema modelado, a identificação de defeitos no Sistema de Gestão Patrimonial - SGP. Sendo assim, os resultados não poderão ser generalizados para outros contextos, sendo necessário para isto elaborar novos estudos para ampliar a validade externa. Entretanto, os resultados do estudo podem servir como possíveis indícios de viabilidade de utilizar *UbiVeri* para apoiar a atividade de verificação de requisitos de ubiquidade em projetos de software ubíquo.
- **Validade de constructo.** Para obter validade de constructo, precisamos verificar que os tratamentos representem bem a construção da causa e que a saída do experimento represente bem a construção do efeito. Este estudo possui dois fatores e dois tratamentos e a causa pode ser mapeada no uso ou não de *UbiVeri*. A saída, por sua vez, está ligada diretamente ao efeito, uma vez que ambos podem ser extraídos a partir dos valores das variáveis dependentes.

8.2.3 Operação

Para executar esse estudo, foram convidados alunos que estavam participando da disciplina Engenharia de Software Orientada a Objetos no primeiro semestre de 2010. Ao total, foram considerados 18 alunos no estudo. Cada um assinou um termo de consentimento que explicava sobre o objetivo geral do estudo e autorizava que o material produzido fosse utilizado nesta pesquisa. Os participantes também preencheram um formulário de caracterização para avaliar o grau de conhecimento e a experiência de cada um com projetos de software.

Tais estudantes relataram possuir experiência em Engenharia de Software, incluindo as atividades de especificação de requisitos, conforme pode ser observado na Tabela 8.4 que descreve a caracterização dos participantes referente às atividades de especificação e verificação de requisitos.

Tabela 8.4. Caracterização dos Participantes

Grupo	Participante	Experiência escrevendo requisitos	Experiência escrevendo casos de uso	Experiência revisando requisitos	Experiência revisando casos de uso	Experiência modificando requisitos para manutenção
1	1	5	5	5	5	5
	2	3	3	2	2	3
1	3	4	4	3	3	4
	4	3	3	2	2	3
2	5	3	3	2	2	2
	6	4	4	3	3	3
2	7	3	3	2	2	3
	8	3	3	2	2	3
2	9	4	4	3	3	3
	10	3	3	3	3	3
3	11	3	3	2	2	2
	12	3	3	3	3	3
3	13	3	3	3	3	3
	14	3	3	2	2	3
4	15	4	4	3	3	4
	16	3	3	3	3	3
4	17	4	4	3	3	3
	18	3	3	3	3	3

Legenda:
1 = nenhum

2 = estudei em aula ou em livro
3 = pratiquei em 1 projeto em sala de aula
4 = usei em 1 projeto na indústria
5 = usei em vários projetos na indústria

Os participantes foram divididos em nove equipes com dois participantes cada. Cada equipe foi associada a um dos quatro tipos de grupo definidos na seção 8.2.2. A escolha dos participantes de cada equipe foi feita tomando como referência as respostas dos formulários de caracterização e teve o objetivo de manter as equipes as mais equivalentes possíveis considerando os critérios de caracterização quanto ao conhecimento e a experiência de cada participante.

As equipes receberam uma versão do documento de requisitos do Sistema de Gestão de Inventário Patrimonial (SGP). Este documento possuía requisitos associados aos cenários de ubiquidade (Bicicletário e Item Rastreável) previamente definidos. Cada grupo trabalhou com um cenário específico em cada rodada do estudo. Dessa forma, as equipes foram solicitadas a revisar o documento com objetivo de identificar defeitos associados ao domínio de ubiquidade.

A revisão dos requisitos de cada novo cenário do SGP foi feita de forma sequencial considerando a distribuição apresentada na Tabela 8.5. Durante a execução de cada interação, as equipes responderam a questionários que tinham o propósito de capturar o esforço envolvido na execução da atividade assim como o nível de satisfação do usuário em executá-las. Os resultados deste estudo foram computados a partir das informações obtidas com estes questionários e do formulário para relato das discrepâncias identificadas em cada revisão.

Por fim, é importante observar que embora tenham sido definidas duplas para a distribuição das atividades, a sua execução foi feita de forma individual. Ou seja, para cada grupo em cada iteração havia ao final da execução dois formulários de discrepância e dois formulários de acompanhamento da atividade preenchidos.

Tabela 8.5. Operação do estudo

	Grupo 1 (2 grupos)	**Grupo 2 (3 grupos)**	**Grupo 3 (2 grupos)**	**Grupo 4 (2 grupos)**
Rodada 1	*Ad Hoc*	*Ad Hoc*	*Ad Hoc*	*UbiVeri*
Rodada 2	*Ad Hoc*	*UbiVeri*	*UbiVeri*	*UbiVeri*

Branco = cenário item rastreável
Preto = cenário bicicletário

8.2.4 Análise e Interpretação dos Dados

Após a execução do estudo, foi realizada a fase de análise dos dados seguindo os procedimentos definidos durante o planejamento do estudo. As variáveis analisadas durante este estudo foram:

- **Número de Discrepâncias:** Analisa se o uso de *UbiVeri* impacta a identificação de discrepâncias associados à questão da ubiquidade;
- **Número de Falsos Positivos:** Analisa se o uso de *UbiVeri* impacta o número de falsos positivos associados às discrepâncias reportadas;
- **Número de Defeitos:** Analisa se o uso de *UbiVeri* impacta o número defeitos identificados (Número de Defeitos = Número de Discrepâncias – Número de Falsos Positivos);
- **Duração da Revisão:** Analisa se o uso de *UbiVeri* onera a atividade revisão da especificação de requisitos;
- **Satisfação do Usuário:** Analisa se o uso de *UbiVeri* aumenta a satisfação do usuário na execução da atividade de revisão de requisitos de ubiquidade.

A análise destas cinco variáveis foi realizada em duas etapas: (1) etapa quantitativa envolvendo as variáveis número de defeitos, número de discrepâncias, número de falsos positivos e duração da revisão; (2) etapa qualitativa envolvendo a variável satisfação do usuário.

Para a etapa quantitativa, o primeiro passo realizado para todas as variáveis foi a análise de *outlier*. Em seguida, é verificado se os dados são normalmente distribuídos. Em caso positivo, serão utilizados métodos paramétricos na análise dos dados. Caso negativo, os não paramétricos serão utilizados. Somente após isso, foram realizadas as análises quantitativas para cada variável conforme pode ser observado nas próximas seções.

8.2.4.1 Análise da Variável Número de Defeitos

O objetivo desta análise é identificar se o uso de *UbiVeri* impacta no número de defeitos identificados na revisão dos requisitos de ubiquidade se comparada à execução da revisão de forma *ad hoc*.

Para realizar a análise do impacto da abordagem de revisão (*ad hoc* e *UbiVeri*) na execução da atividade, foi selecionado o seguinte agrupamento:

- **Grupos para o cenário Bicicletário:** Grupos 1, 3 e 4 da Rodada 1;
- **Grupos para o cenário Item Rastreável:** Grupos 1, 3 e 4 da Rodada 2;

Análise de *Outlier* – Cenário Bicicletário

Aplicando-se análise de *outlier*, observa-se que nenhum participante é avaliado como sendo um *outlier* na avaliação do número de defeitos identificados por abordagem de revisão (Figura 8.1). Sendo assim, nenhum participante foi excluído para a análise da variável defeito.

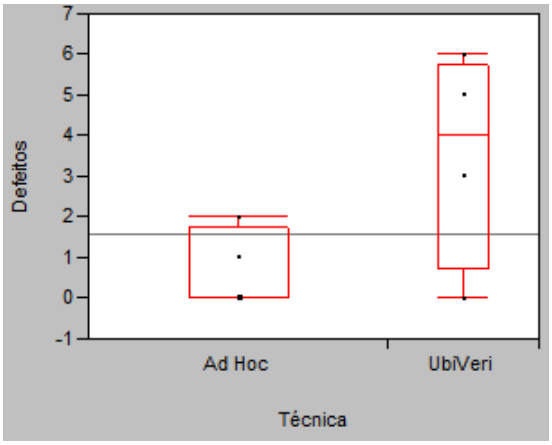


Figura 8.1. Análise de *outlier* para a variável Defeito.

Para este agrupamento, foi realizada a análise sobre a distribuição normal dos dados utilizando o método *levene*. Seu resultado indicou que a distribuição dos dados não era normal uma vez que o valor atingido, 0,0278 (Figura 8.2) é menor do que o valor indicado para distribuição normal deste método que é de 0,1. Desta forma, houve para esta análise a necessidade de usar um método não paramétrico para a análise de variância apresentada abaixo.

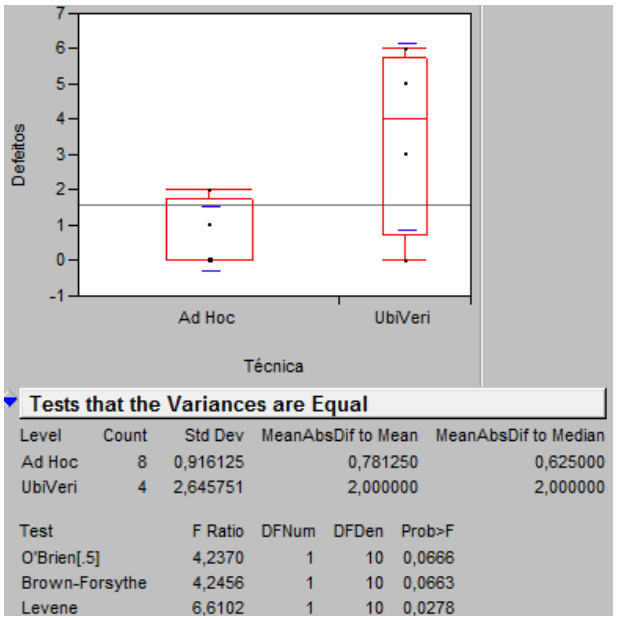


Figura 8.2. Análise de distribuição normal utilizando *levene*.

Análise de Variância – Wilcoxon – Cenário Bicicletário

Aplicando-se testes estatísticos (não paramétrico – Wilcoxon), percebe-se que existe diferença estatística no uso da abordagem *UbiVeri* se comparada à revisão *ad hoc* dos requisitos de ubiquidade, como pode ser observado pelo valor de Prob>|Z| = 0,0693 ser inferior ao valor 0,1 que representa o valor de significância estatística (taxa de erro) de 10% utilizada neste estudo (ver Figura 8.3).

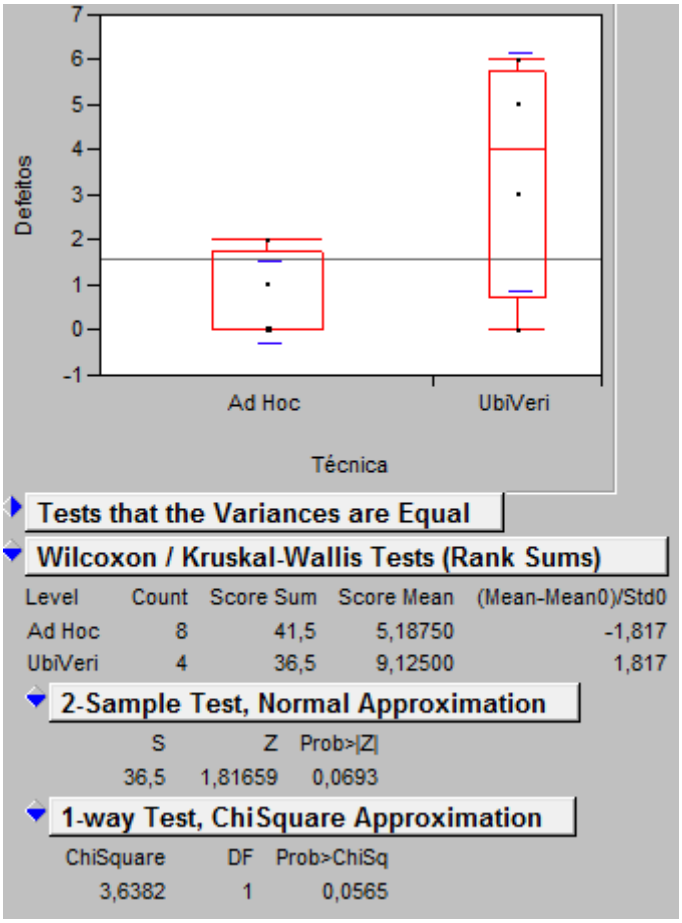


Figura 8.3. Resultado do uso de Wilcoxon.

Análise de *Outlier* – Cenário Item Rastreável

Aplicando-se análise de *outlier*, observa-se que nenhum participante é avaliado como sendo um *outlier* na avaliação do número de defeitos identificados por abordagem de revisão (Figura 8.4). Sendo assim, nenhum participante foi excluído para a análise da variável defeito.

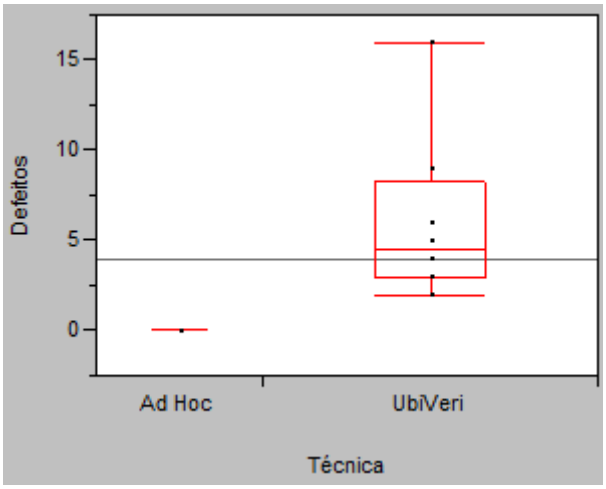


Figura 8.4. Análise de *outlier* para a variável Defeito.

Para este agrupamento, devido ao número limitado de dados para análise, não foi possível realizar a análise sobre a distribuição normal dos dados. Desta forma, optou-se por utilizar a análise paramétrica apresentada abaixo.

Análise de Variância – ANOVA – Item Rastreável

Aplicando-se testes estatísticos, percebe-se que existe diferença estatística no uso da abordagem *UbiVeri* se comparada à revisão *ad hoc* dos requisitos de ubiquidade, como pode ser observado pela não-interseção entre os círculos representando cada abordagem e pelo *p-value* que ficou em 0,0290, inferior ao valor 0,1 que representa o valor de significância estatística (taxa de erro) de 10% (ver Figura 8.5).

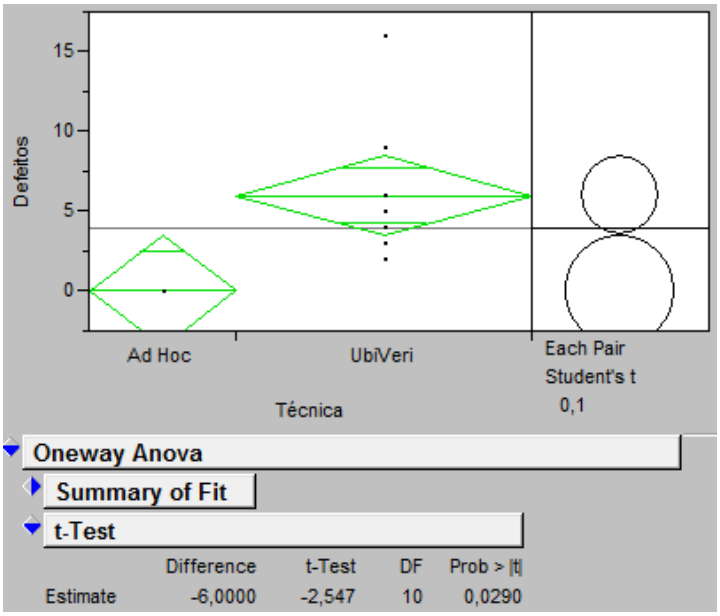


Figura 8.5. Resultado do uso de ANOVA.

Sendo assim, os resultados indicam que o fator avaliado (uso de *UbiVeri*) influencia no aumento do número de defeitos identificados na revisão de requisitos de ubiquidade em projetos de software ubíquo considerando os dois cenários de ubiquidade utilizados.

8.2.4.2 Análise da Variável Número de Discrepâncias

O objetivo desta análise é identificar se o uso de *UbiVeri* impacta no número de discrepâncias identificadas na revisão de requisitos de ubiquidade se comparada à execução da revisão de forma *ad hoc*.

Para realizar a análise do impacto da abordagem de revisão (*ad hoc* e *UbiVeri*) na execução da atividade, foi selecionado o seguinte agrupamento:

- **Grupos para o cenário Bicicletário:** Grupos 1, 3 e 4 da Rodada 1;
- **Grupos para o cenário Item Rastreável:** Grupos 1, 3 e 4 da Rodada 2;

Análise de *Outlier* – Cenário Bicicletário

Aplicando-se análise de *outlier*, observa-se que nenhum participante é avaliado como sendo um *outlier* na avaliação do número de discrepâncias identificadas por abordagem de revisão (Figura 8.6). Sendo assim, nenhum participante foi excluído para a análise da variável discrepância para o cenário Bicicletário.

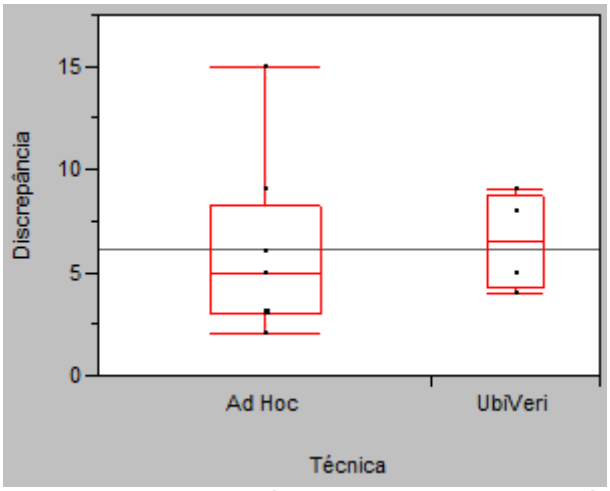


Figura 8.6. Análise de *outlier* para a variável Discrepância – Cenário Bicicletário.

Para este agrupamento, foi realizada a análise sobre a distribuição normal dos dados utilizando o método *levene*. Seu uso indicou que a distribuição dos dados era normal uma vez que o valor atingido, 0,5021, (Figura 8.7) é maior do que o valor

indicado para distribuição normal deste método que é de 0,1. Desta forma, para a análise de variância será utilizado um método paramétrico.

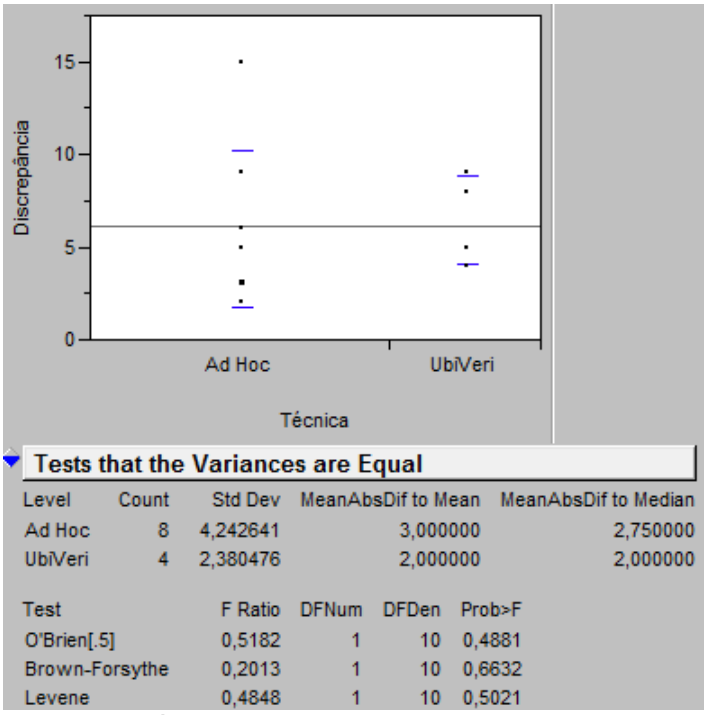


Figura 8.7. Análise de distribuição normal utilizando *levene*.

Análise de Variância – ANOVA – Cenário Bicicletário

Aplicando-se testes estatísticos, percebe-se que não existe diferença estatística no uso da abordagem *UbiVeri* se comparada à revisão *ad hoc* dos requisitos de ubiquidade, como pode ser observado pelo *p-value* que ficou em 0,8334, superior ao valor 0,1 que representa o valor de significância estatística (taxa de erro) de 10% (ver Figura 8.8).

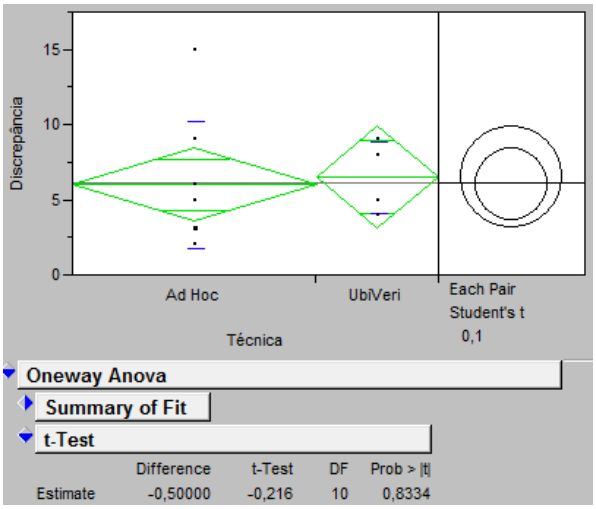


Figura 8.8. Resultado do uso de ANOVA.

Análise de *Outlier* – Cenário Item Rastreável

Aplicando-se análise de *outlier*, observa-se que um participante é avaliado como sendo um *outlier* na avaliação do número de discrepâncias identificadas por abordagem de revisão. Sendo assim, um participante foi excluído para a análise da variável discrepância para o cenário Item Rastreável (Figura 8.9).

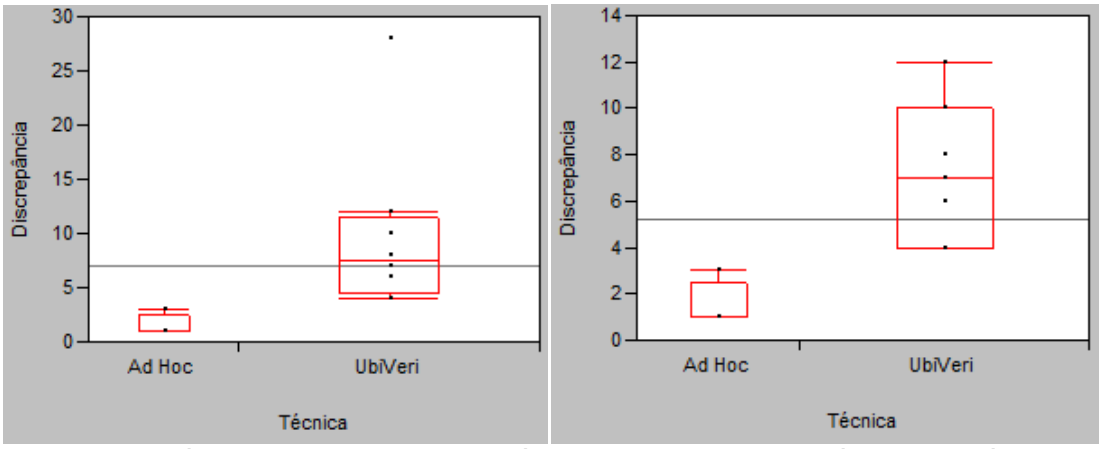


Figura 8.9. Análise de *outlier* para a variável Discrepância – Cenário Bicicletário antes e depois da análise de *outlier*.

Para este agrupamento foi realizada a análise sobre a distribuição normal dos dados utilizando o método *levene*. Seu uso indicou que a distribuição dos dados era normal uma vez que o valor atingido, 0,0944, (Figura 8.10) é maior do que o valor indicado para distribuição normal deste método que é de 0,1. Desta forma, para a análise de variância será utilizado um método paramétrico.

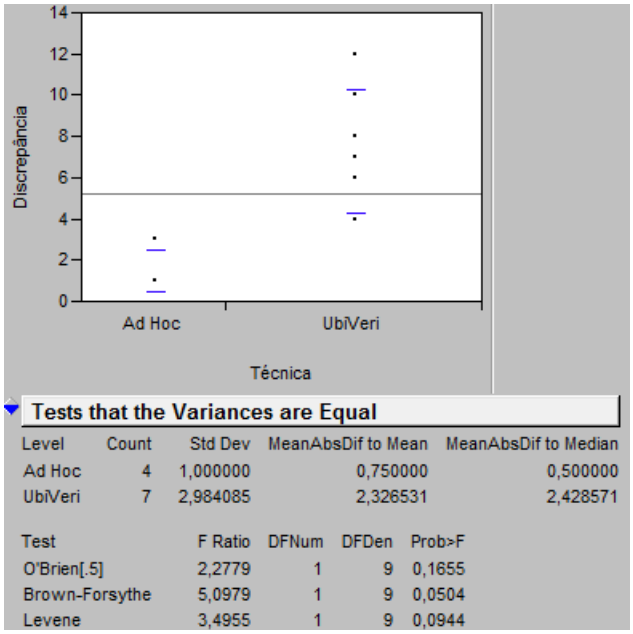


Figura 8.10. Análise de distribuição normal utilizando *levene*.

Análise de Variância – ANOVA – Cenário Item Rastreável

Aplicando-se testes estatísticos, percebe-se que existe diferença estatística no uso da abordagem *UbiVeri* se comparada à revisão *ad hoc* dos requisitos de ubiquidade para o cenário item rastreável, como pode ser observado pelo *p-value* que ficou em 0,0050, inferior ao valor 0,1 que representa o valor de significância estatística (taxa de erro) de 10% (ver Figura 8.11).

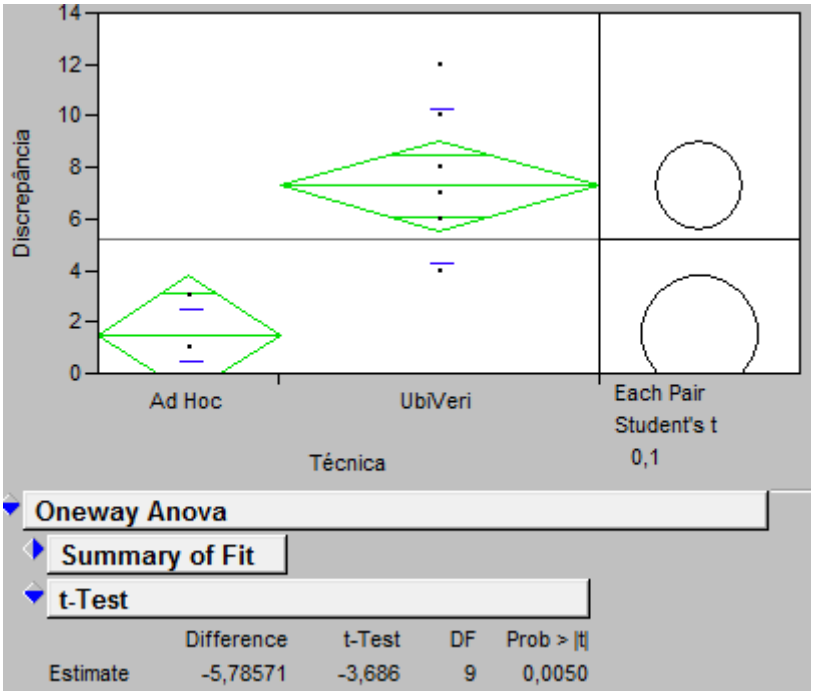


Figura 8.11. Resultado do uso de ANOVA.

Sendo assim, embora haja indícios do aumento do número de discrepâncias identificadas, nada se pode afirmar sobre a influencia do fator avaliado (uso de *UbiVeri*) no número de discrepâncias identificadas na revisão de requisitos de ubiquidade uma vez que embora haja uma diferença significativa para o cenário item rastreável, o mesmo não foi detectado para o cenário bicicletário.

8.2.4.3 Análise da Variável Número de Falso Positivos

O objetivo desta análise é identificar se o uso de *UbiVeri* impacta no número de falso positivos ocorridos na revisão de requisitos de ubiquidade se comparado à execução da revisão de forma *ad hoc*.

Para realizar a análise do impacto da abordagem de revisão (*ad hoc* e *UbiVeri*) na execução da atividade, foi selecionado o seguinte agrupamento:

- **Grupos para o cenário Bicicletário:** Grupos 1, 3 e 4 da Rodada 1;
- **Grupos para o cenário Item Rastreável:** Grupos 1, 3 e 4 da Rodada 2;

Análise de *Outlier* – Cenário Bicicletário

Aplicando-se análise de *outlier*, observa-se que nenhum participante é avaliado como sendo um *outlier* na avaliação do número de falso positivos por abordagem de revisão (Figura 8.12). Sendo assim, nenhum participante foi excluído para a análise da variável falso positivo para o cenário Bicicletário.

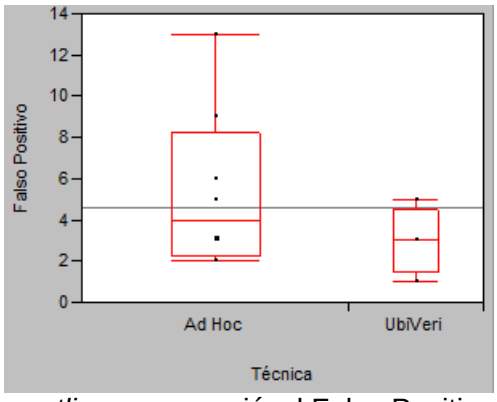
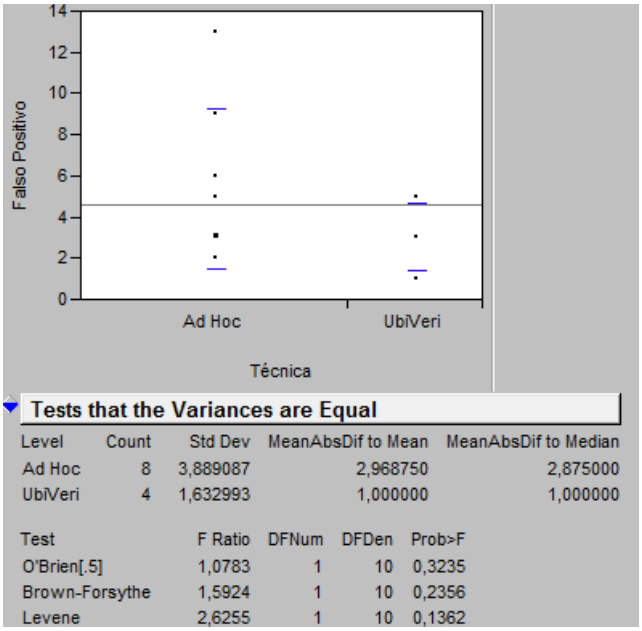


Figura 8.12. Análise de *outlier* para a variável Falso Positivo – Cenário Bicicletário.

Para este agrupamento, foi realizada a análise sobre a distribuição normal dos dados utilizando o método *levene*. Seu resultado indicou que a distribuição dos dados era normal uma vez que o valor atingido, 0,1362, (Figura 8.13) é maior do que o valor indicado para distribuição normal deste método que é de 0,1. Desta forma, para a análise de variância será utilizado um método paramétrico.



Tests that the Variances are Equal

Level	Count	Std Dev	MeanAbsDif to Mean	MeanAbsDif to Median
Ad Hoc	8	3,889087	2,968750	2,875000
UbiVeri	4	1,632993	1,000000	1,000000

Test	F Ratio	DFNum	DFDen	Prob>F
O'Brien[.5]	1,0783	1	10	0,3235
Brown-Forsythe	1,5924	1	10	0,2356
Levene	2,6255	1	10	0,1362

Figura 8.13. Análise de distribuição normal utilizando *levene*.

Análise de Variância – ANOVA – Cenário Bicicletário

Aplicando-se testes estatísticos, percebe-se que não existe diferença estatística no uso da abordagem *UbiVeri* se comparada à revisão *ad hoc* dos requisitos de ubiquidade, como pode ser observado pelo *p-value* que ficou em 0,2772, superior ao valor 0,1 que representa o valor de significância estatística (taxa de erro) de 10% (ver Figura 8.14).

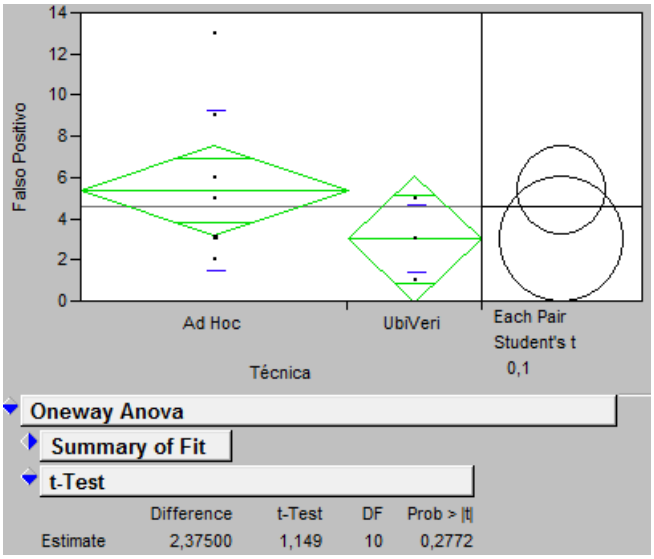


Figura 8.14. Resultado do uso de ANOVA.

Análise de *Outlier* – Cenário Item Rastreável

Aplicando-se análise de *outlier*, observa-se que um participante é avaliado como sendo um *outlier* na avaliação do número de falsos positivos identificados por abordagem de revisão. Sendo assim, um participante foi excluído para a análise da variável falso positivo para o cenário Item Rastreável (Figura 8.15).

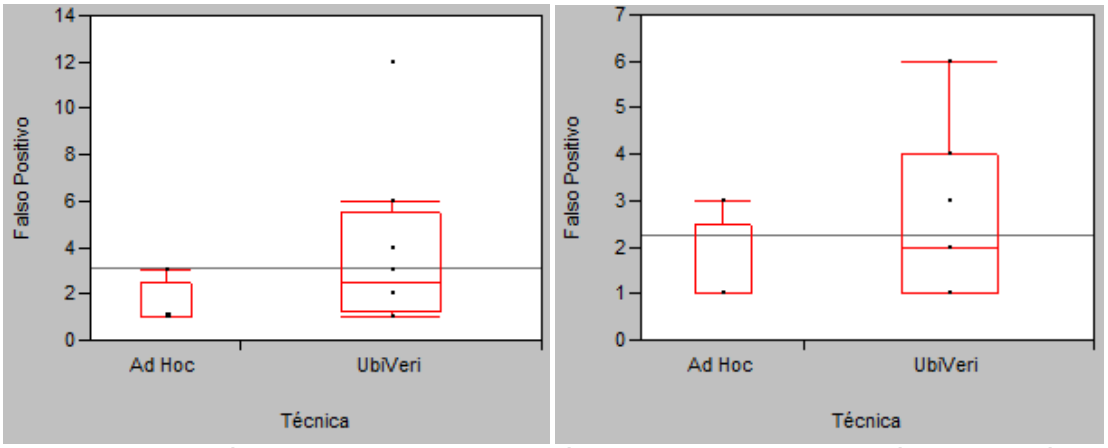


Figura 8.15. Análise de *outlier* para a variável Falso Positivo – Cenário Bicicletário antes e depois da análise de *outlier*.

Para este agrupamento foi realizada a análise sobre a distribuição normal dos dados utilizando o método *levene*. Seu resultado indicou que a distribuição dos dados era normal uma vez que o valor atingido, 0,2684, (Figura 8.16) é maior do que o valor indicado para distribuição normal deste método que é de 0,1. Desta forma, para a análise de variância será utilizado um método paramétrico.

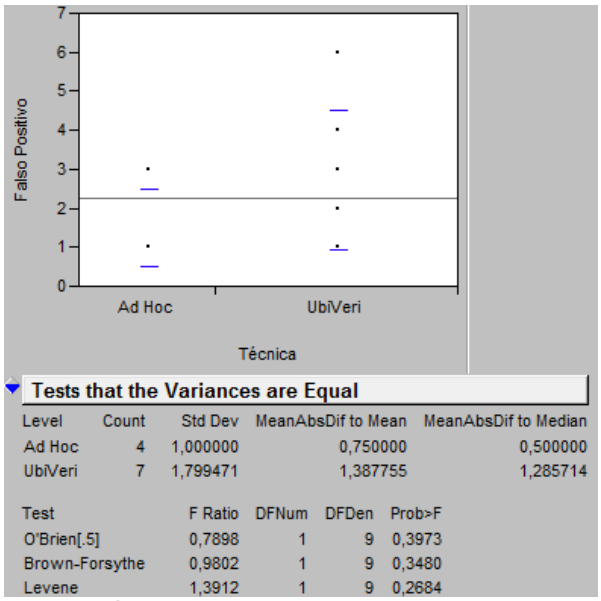


Figura 8.16. Análise de distribuição normal utilizando *levene*.

Análise de Variância – ANOVA – Cenário Item Rastreável

Aplicando-se testes estatísticos, percebe-se que não existe diferença estatística no uso da abordagem *UbiVeri* se comparada à revisão *ad hoc* dos requisitos de ubiquidade, como pode ser observado pelo *p-value* que ficou em 0,2509, superior ao valor 0,1 que representa o valor de significância estatística (taxa de erro) de 10% (ver Figura 8.17).

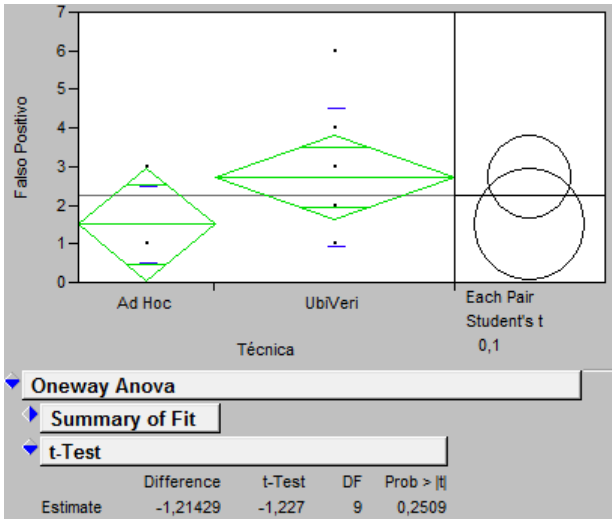


Figura 8.17. Resultado do uso de ANOVA.

Sendo assim, os resultados indicam que o uso de *UbiVeri* não influencia no número de falsos positivos reportados quando se comparado ao uso de uma abordagem *ad hoc* para apoiar a identificação de defeitos em requisitos de ubiquidade.

8.2.4.4 Análise da Variável Duração da Revisão

O objetivo desta análise é identificar se o uso de *UbiVeri* traz maior esforço em tempo na realização da atividade de revisão de requisitos de ubiquidade se comparado à execução da revisão de forma *ad hoc*.

Para realizar a análise do impacto da abordagem de revisão (*ad hoc* e *UbiVeri*) na execução da atividade, foi selecionado o seguinte agrupamento:

- **Grupos para o cenário Bicicletário:** Grupos 1, 3 e 4 da Rodada 1;
- **Grupos para o cenário Item Rastreável:** Grupos 1, 3 e 4 da Rodada 2;

Análise de *Outlier* – Cenário Bicicletário

Aplicando-se análise de *outlier*, observa-se que nenhum participante é avaliado como sendo um *outlier* na avaliação da duração da revisão por abordagem de revisão (Figura 8.18). Sendo assim, nenhum participante foi excluído para a análise da variável duração da revisão para o cenário Bicicletário.

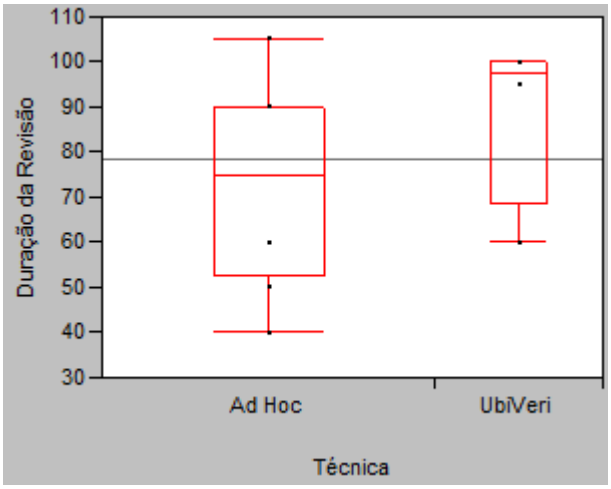


Figura 8.18. Análise de *outlier* para a variável Duração da Revisão – Cenário Bicicletário.

Para este agrupamento, foi realizada a análise sobre a distribuição normal dos dados utilizando o método *levene*. Seu resultado indicou que a distribuição dos dados era normal uma vez que o valor atingido, 0,2616, (Figura 8.19) é maior do que o valor indicado para distribuição normal deste método que é de 0,1. Desta forma, para a análise de variância será utilizado um método paramétrico.

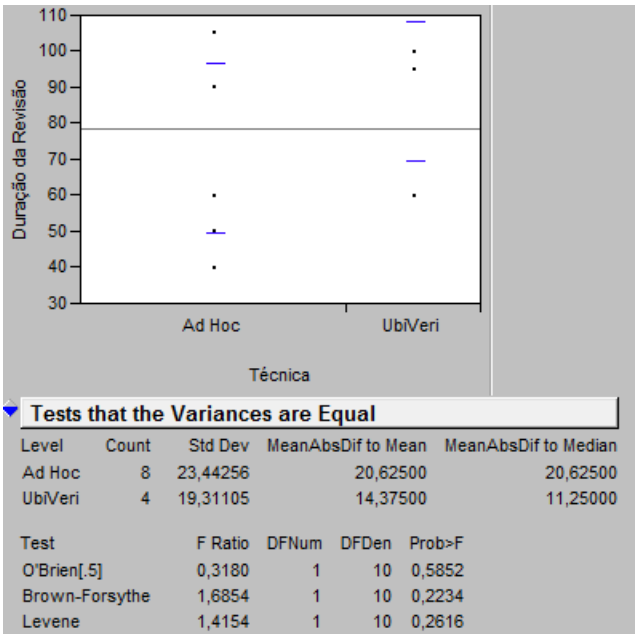


Figura 8.19. Análise de distribuição normal utilizando *levene*.

Análise de Variância – ANOVA – Cenário Bicicletário

Aplicando-se testes estatísticos, percebe-se que não existe diferença estatística na duração da atividade de verificação no uso da abordagem *UbiVeri* se comparada à revisão *ad hoc* dos requisitos de ubiquidade, como pode ser observado pelo *p-value* que ficou em 0,2789, superior ao valor 0,1 que representa o valor de significância estatística (taxa de erro) de 10% (ver Figura 8.20).

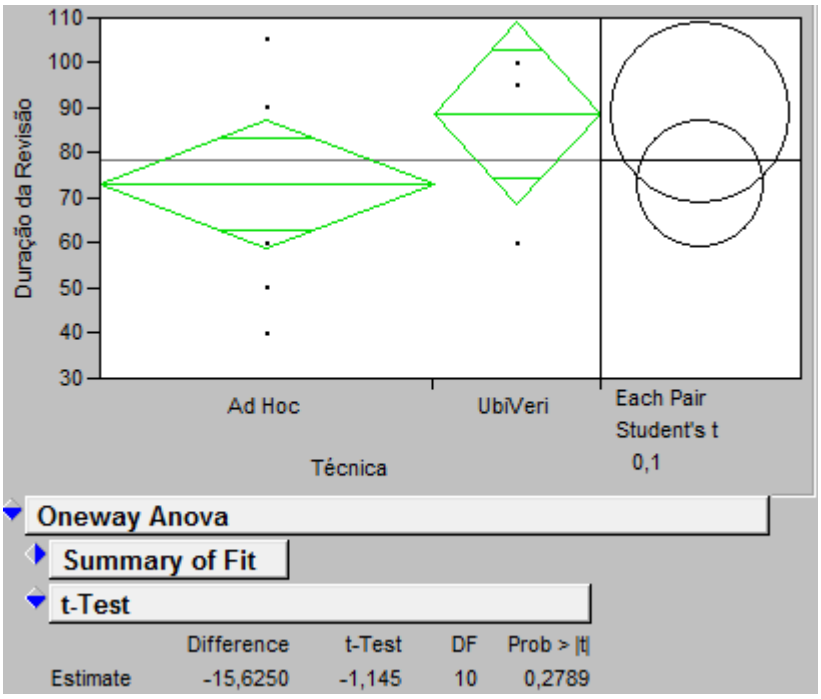


Figura 8.20. Resultado do uso de ANOVA.

Análise de *Outlier* – Cenário Item Rastreável

Aplicando-se análise de *outlier*, observa-se que nenhum participante é avaliado como sendo um *outlier* na avaliação da duração da revisão por abordagem de revisão. Sendo assim, nenhum participante foi excluído para a análise da variável duração da revisão o cenário Item Rastreável (Figura 8.21).

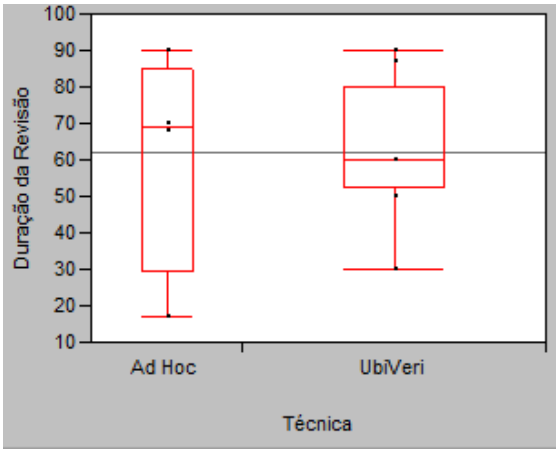


Figura 8.21. Análise de *outlier* para a variável Duração da Revisão – Cenário Bicicletário.

Para este agrupamento foi realizada a análise sobre a distribuição normal dos dados utilizando o método *levene*. Seu resultado indicou que a distribuição dos dados era normal uma vez que o valor atingido, 0,3431, (Figura 8.22) é maior do que valor indicado para distribuição normal deste método que é de 0,1. Desta forma, para a análise de variância será utilizado um método paramétrico.

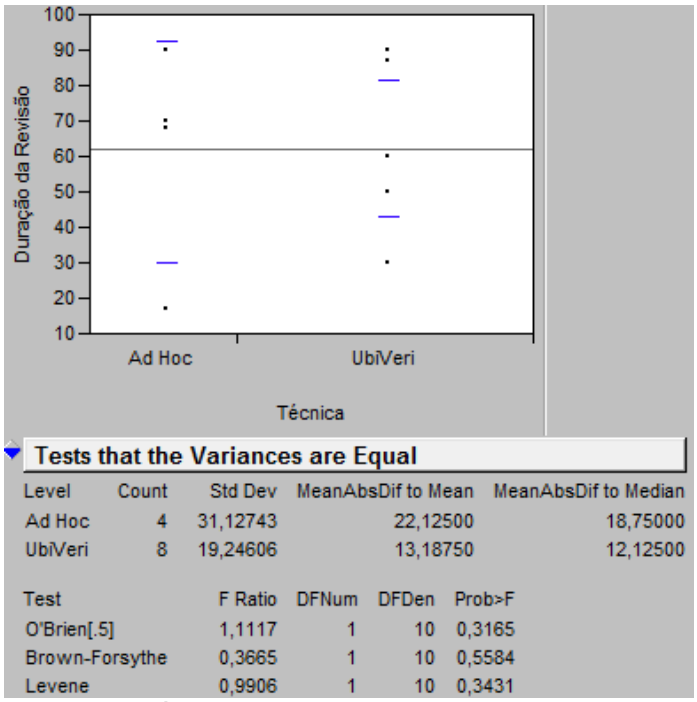


Figura 8.22. Análise de distribuição normal utilizando *levene*.

Análise de Variância – ANOVA – Cenário Item Rastreável

Aplicando-se testes estatísticos, percebe-se que não existe diferença estatística no uso da abordagem *UbiVeri* se comparada à revisão *ad hoc* dos requisitos de ubiquidade, como pode ser observado pelo *p-value* que ficou em 0,9526, superior ao valor 0,1 que representa o valor de significância estatística (taxa de erro) de 10% (ver Figura 8.17).

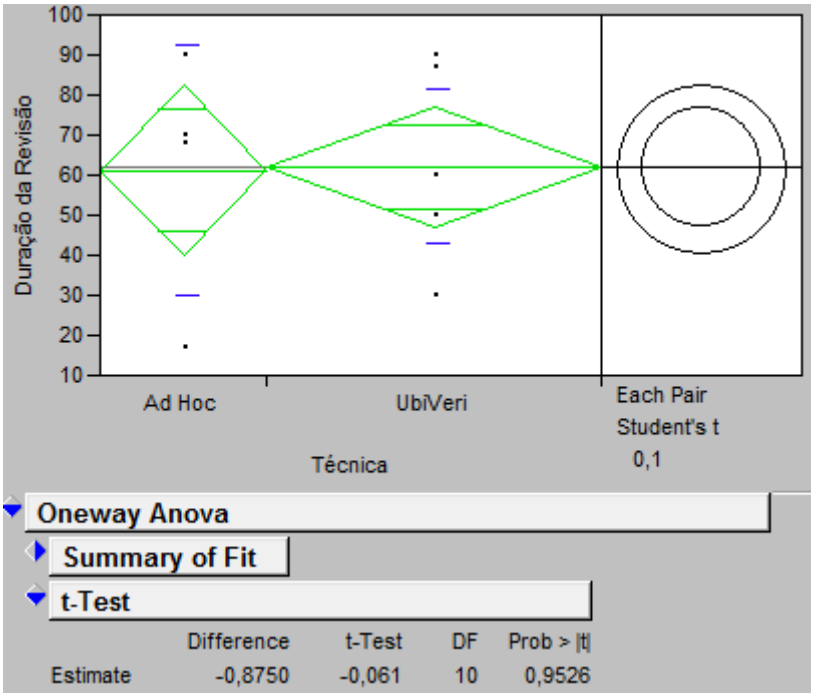


Figura 8.23. Resultado do uso de ANOVA.

Sendo assim, os resultados indicam que o uso de *UbiVeri* não onera a duração da revisão quando comparado ao uso de uma abordagem *ad hoc* para apoiar a identificação de defeitos em requisitos de ubiquidade.

8.2.4.5 Análise da Variável Satisfação do Usuário

O objetivo desta análise é identificar se o uso de *UbiVeri* aumenta a satisfação do usuário ao executar a atividade de revisão de requisitos de ubiquidade.

Para realizar esta análise, foram consideradas as respostas dos participantes do estudo ao formulário de acompanhamento entregue ao final da execução da segunda rodada do experimento. Ao total, foram entregues 18 formulários preenchidos com a avaliação do uso da abordagem. A partir de agora, serão apresentadas as perguntas presentes no formulário associadas à variável satisfação do usuário e as

respostas que retratam a percepção dos envolvidos no estudo quanto ao uso de *UbiVeri*.

- **Nível de satisfação na realização da atividade de revisão:**

Quando questionados sobre a satisfação na realização da atividade de revisão de requisitos, 67% dos participantes indicaram que ficaram parcialmente satisfeitos; 25% reportaram que ficaram largamente satisfeitos e 8% reportou totalmente satisfeito. Nenhum dos participantes do estudo reportou insatisfação na realização da revisão dos requisitos.

Alguns dos participantes que indicaram que ficaram parcialmente satisfeitos reportaram como justificativa o fato de terem pouca experiência em atividades de revisão de requisitos e que este fato havia dificultado o entendimento do objetivo da atividade.

- **Complexidade na realização da atividade:**

Quando questionados sobre a complexidade na relação da atividade de revisão utilizando *UbiVeri,* 58% dos participantes consideraram a atividade simples enquanto que 42% considerou a atividade complexa. Nenhum participante reportou que a atividade era muito simples ou muito complexa.

Aqueles que consideraram a atividade complexa indicaram que a maior parte da complexidade estava no fato do domínio da ubiquidade ser muito específico e não no uso da técnica especificamente.

- **Quais as principais dificuldades encontradas no uso da abordagem para apoiar a verificação dos requisitos de ubiquidade?**

Para este item, os participantes reportaram que não houve dificuldade no uso de *UbiVeri.*

Além disso, também foi acrescentado que algumas definições de ubiquidade deveriam ser adicionadas ao glossário para facilitar o entendimento de algumas questões presentes no Guia de Apoio à Verificação dos requisitos.

- **Você sentiu falta de algum apoio adicional no uso da abordagem?**

Os participantes do estudo indicaram que não há a necessidade de outros artefatos para apoiar a atividade de revisão dos requisitos além daqueles já fornecidos por *UbiVeri*.

Desta forma, os instrumentos fornecidos por *UbiVeri* foram considerados satisfatórios.

- **Você acredita que a abordagem proposta possa ser melhorada para tornar seu uso mais simples? Como?**

No geral, os participantes reportaram não saber o que poderia ser feito para aprimorar a técnica uma vez que ficaram satisfeitos com seu uso.

Apenas dois dos participantes sugeriram a adição de alguns exemplos associados às perguntas do Guia de Verificação de Requisitos para facilitar o entendimento sobre como aplicá-lo.

8.2.5 Resumo dos Resultados

Será apresentado agora um resumo dos resultados obtidos ao longo da análise dos dados deste estudo (Tabela 8.6). Como pode ser observado, os resultados estatísticos indicam qual tratamento influencia na atividade de revisão dos requisitos de ubiquidade.

Os resultados sugerem que o fator abordagem de verificação (*ad hoc* ou *UbiVeri*) possui influência direta no número de defeitos reportados. Segundo as análises realizadas, *UbiVeri* provê melhorias para o número de defeitos reportados ao mesmo tempo em que o esforço em tempo não é influenciado pela abordagem de verificação.

Tabela 8.6. Resumo dos Resultados

Variável	Cenário	p-value	Influência
Número de Discrepâncias	Bicicletário	0,8334	-
	Item Rastreável	0,0050	*UbiVeri*
Falso Positivo	Bicicletário	0,2772	-
	Item Rastreável	0,2509	-
Número de Defeitos	Bicicletário	0,0693	*UbiVeri*
	Item Rastreável	0,0290	*UbiVeri*
Duração da Revisão	Bicicletário	0,2789	-
	Item Rastreável	0,9526	-

Por fim, para a variável satisfação do usuário, foi identificado que *UbiVeri* aumenta a satisfação do usuário na execução da revisão de requisitos de ubiquidade se comparada à revisão *ad hoc*. Além disso, não foram identificadas necessidades de melhoria que impactassem no formato atual da abordagem. Apenas dois pequenos ajustes foram sugeridos: aprimorar o glossário e acrescentar alguns exemplos de uso da técnica. Sendo assim, é possível refutar a hipótese nula formulada para este estudo.

8.3 – Considerações Finais

Este capítulo apresentou um estudo experimental conduzido para avaliar a abordagem de apoio à verificação de requisitos de ubiquidade proposta neste trabalho (*UbiVeri*) em relação à execução da atividade de verificação de forma *ad hoc*.

O estudo foi realizado com alunos de graduação, e seus resultados indicam que, dada a população do estudo e os cenários de ubiquidade considerados, *UbiVeri* poderia contribuir para a melhoria na atividade de revisão dos requisitos de ubiquidade de um projeto de software ubíquo sem onerar, em tempo, a execução desta atividade através do aumento do número de defeitos efetivamente identificados. No entanto, os resultados não podem ser considerados conclusivos em virtude do pouco volume de dados obtidos.

Além disso, *UbiVeri* também foi avaliado de forma subjetiva pelos participantes do estudo a respeito da sua satisfação durante o uso da abordagem, e seus resultados também foram bastante positivos dentro da população do estudo.

O próximo capítulo apresenta as considerações finais deste trabalho, descrevendo suas conclusões, resultados obtidos, limitações e futuras linhas de pesquisas a serem seguidas para continuidade desta pesquisa.

Capítulo 9 – Considerações Finais

Neste capítulo são apresentadas as considerações finais sobre o trabalho realizado, incluindo os resultados obtidos, as limitações da pesquisa e os próximos passos a serem realizados com a continuidade desta pesquisa.

9.1 – Considerações Finais

As tecnologias mais avançadas são aquelas que desaparecem sob a perspectiva de seus usuários. Na computação ubíqua, os computadores estão embutidos no ambiente que nos cerca, criando um novo paradigma de acesso e manipulação de informação. Dessa forma, a computação ubíqua pressupõe uma forte integração com o mundo real, mantendo alta transparência e o foco no usuário.

A computação ubíqua tornou-se possível devido, em parte, aos avanços tecnológicos nas áreas de dispositivos móveis e redes de comunicação. Este avanço permite criar artefatos e ambientes que disponibilizam para o usuário recursos de comunicação e de processamento.

A computação ubíqua é uma área multidisciplinar envolvendo desafios de pesquisa em diversas áreas como redes de computadores, otimização, processamento de sinais e inteligência artificial, dentre outros. Sob a perspectiva da engenharia de software, estes desafios podem ser observados na necessidade de investigação de técnicas ou métodos de engenharia de software para apoiar a construção deste tipo de aplicação como: definição dos requisitos, desenvolvimento de metodologias, processos de software, abordagens de teste e técnicas de garantia da qualidade.

Neste contexto, este trabalho propôs uma abordagem que auxilia o engenheiro de software durante as atividades de definição e verificação dos requisitos de ubiquidade do projeto através de um conjunto de guias baseadas em conhecimento do domínio da computação ubíqua. Para isto, a pesquisa seguiu duas linhas de trabalho: (1) organização de um corpo de conhecimento em computação ubíqua baseado em resultados de estudos experimentais, e; (2) definição de guias de apoio às atividades de definição e verificação dos requisitos de ubiquidade em projetos de software de forma a atender às necessidades específicas da engenharia de requisitos no contexto da computação ubíqua.

9.2 – Resultados Obtidos

Analisando os sete objetivos definidos para este trabalho e descritos na Seção 1.3, os seguintes resultados foram obtidos:

1. **Organizar um corpo de conhecimento em computação ubíqua:** o corpo de conhecimento foi organizado através da execução de revisões sistemáticas que permitiram identificar características e fatores associados à computação ubíqua;
2. **Estruturar o corpo de conhecimento em computação ubíqua organizado:** o corpo de conhecimento foi estruturado através da execução de um *survey* que permitiu definir a pertinencia e a relevância das características que compõem o corpo de conhecimento;
3. **Definir o processo de apoio à definição e verificação de requisitos de ubiquidade:** foram estabelecidas atividades de apoio à definição e verificação de requisitos assim como seus artefatos consumidos e produzidos;
4. **Elaborar modelos conceituais representando os fatores associados a cada uma das características de ubiquidade:** foram elaborados modelos conceituais contemplando os principais conceitos contidos no corpo de conhecimento sobre computação ubíqua e como eles estão relacionados;
5. **Apoiar a especificação de requisitos de ubiquidade:** para este objetivo foi definido *UbiCheck*, abordagem de apoio à definição de requisitos de ubiquidade baseado em *checklist* e corpo de conhecimento em computação ubíqua;
6. **Apoiar a verificação de requisitos de ubiquidade:** para este objetivo foi definido *UbiVeri*, abordagem de apoio à verificação de requisitos de ubiquidade baseado em *checklist* e corpo de conhecimento em computação ubíqua;
7. **Apoiar à evolução do corpo de conhecimento em computação ubíqua:** foi definido um pacote de laboratório que permite evoluir o corpo de conhecimento em computação ubíqua baseado na re-execução simplificada da revisão sistemática.

9.3 – Contribuições da Pesquisa

As principais contribuições desta pesquisa estão classificadas em 5 categorias:

9.3.1 Corpo de Conhecimento em Computação Ubíqua

Um corpo de conhecimento pôde ser organizado contendo: (1) definição atualizada da computação ubíqua; (2) identificação das características da computação ubíqua; (3) identificação de fatores funcionais e restritivos associados às características; (4) definição da pertinência e do nível de relevância das características identificadas. Este corpo de conhecimento foi estruturado a partir da revisão sistemática da literatura e da consulta a especialistas em computação ubíqua por meio de um *survey*, onde os especialistas indicaram qual o conjunto de características de ubicomp possibilita a caracterização de projetos de software ubíquo e qual a relevância de cada uma delas nesta atividade.

9.3.2 Abordagem de Apoio à Definição de Requisitos de Ubiquidade: *UbiCheck*

Para apoiar as atividades associadas à definição dos requisitos de ubiquidade foi definido *UbiCheck*. *UbiCheck* é uma abordagem baseada em *checklist* cujo objetivo é apoiar a especificação dos requisitos de ubiquidade do software com base nas características e fatores funcionais de ubiquidade que compõem o corpo de conhecimento organizado neste trabalho.

Esta abordagem foi avaliada experimentalmente. Para isso, um estudo de observação foi planejado e executado. O propósito deste estudo foi observar a aplicação de *UbiCheck* em um projeto de desenvolvimento de software ubíquo para compreender se a abordagem pode ser utilizada para apoiar a definição de requisitos de ubiquidade. A observação da utilização de *UbiCheck* nos permitiu observar que: (1) ela pode ser utilizada pelos engenheiros de software para apoiar o desenvolvimento de um projeto de software ubíquo; (2) os tempos registrados sugerem que aplicar *UbiCheck* pode não onerar o processo de desenvolvimento. Além disso, os resultados do estudo permitiram evoluir a versão inicial de *UbiCheck* a partir de um conjunto de deficiências identificadas durante o estudo.

9.3.3 Abordagem de Apoio à Verificação de Requisitos de Ubiquidade: *UbiVeri*

Para apoiar as atividades associadas à verificação dos requisitos de ubiquidade foi definido *UbiVeri*. *UbiVeri* é uma abordagem baseada em *checklist*, portanto, o apoio fornecido objetiva indicar para o desenvolvedor que tipo de informação (baseado em um corpo de conhecimento em computação ubíqua) ele deve procurar na especificação dos requisitos com objetivo de identificar defeitos.

Esta abordagem foi avaliada experimentalmente. Para isso, um estudo experimental com o propósito de avaliar aspectos de interesse para engenheiros de software (aumento do número de defeitos identificados na especificação dos requisitos

do projeto, esforço envolvido no uso da abordagem e satisfação no uso da abordagem) foi planejado e executado. Este estudo permitiu observar que *UbiVeri* aumenta o número de defeitos identificados ao mesmo tempo em que não onera a atividade de revisão dos requisitos de ubiquidade e aumenta a satisfação do usuário na identificação de defeitos associados ao domínio da ubiquidade.

9.3.4 Guias de Apoio às atividades de Definição e Verificação de Requisitos Funcionais de Ubiquidade

UbiVeri e *UbiCheck* possuem como componente central os guias de apoio à definição e verificação de requisitos funcionais de ubiquidade em projetos de software. Estes guias podem ser considerados uma contribuição a parte das abordagens *UbiCheck* e *UbiVeri* uma vez que podem ser utilizados independente do uso da abordagem completa. Eles são uma importante fonte de conhecimento que podem ser utilizados no apoio às atividades de desenvolvimento em projetos de software ubíquos.

9.3.5 Metodologia Científica que apóia a Concepção de Tecnologias de Software a partir da condução de Estudos Secundários e Primários

Durante o desenvolvimento desta pesquisa foi adotada uma metodologia científica baseada na condução de estudos secundários e primários para apoiar na concepção de novas tecnologias de software, ou seja, na fase onde uma tecnologia é idealizada e construída, antes de ser avaliada através de novos estudos. Esta metodologia busca agrupar resultados providos por revisões sistemáticas da literatura e a opiniões de especialistas em um tópico de pesquisa para a construção de um corpo de conhecimento sobre diferentes áreas de domínio.

Esta metodologia está descrita no Capítulo 1 deste trabalho e tem sido seguida por outras pesquisas desenvolvidas por membros do Grupo de Engenharia de Software Experimental da COPPE/UFRJ.

9.4 – Limitações

As limitações deste trabalho estão relacionadas, principalmente, a três itens:

(1) Escopo das Abordagens de Definição e Verificação Propostas

Esta primeira limitação diz respeito à conciliação entre os requisitos de ubiquidade e os demais requisitos do software. Como *UbiCheck* e *UbiVeri* apóiam a definição e a verificação dos requisitos de ubiquidade, não há como garantir a consistência desses requisitos com os demais requisitos do software (funcionais e não funcionais).

Nesse sentido, seria interessante que *UbiCheck* e *UbiVeri* fossem combinadas com outras abordagens que provessem apoio à definição e verificação de requisitos de software tradicionais. Nesta combinação, seria importante que a consistência entre os requisitos definidos pelas duas abordagens fosse considerada.

(2) Avaliação Experimental de *UbiCheck*

Neste trabalho, foi realizada uma avaliação inicial de *UbiCheck* com intuito de observar sua aplicação em um projeto de desenvolvimento de software ubíquo para compreender se a abordagem pode ser utilizada para apoiar a definição de requisitos de ubiquidade. Os resultados deste estudo sugerem que a abordagem pode ser viável para apoiar a definição de requisitos em projetos de software ubíquo. Entretanto, não houve oportunidade para realizar um estudo mais abrangente que permitisse caracterizar a eficiência e eficácia do apoio fornecido por *UbiCheck* durante a definição de requisitos.

Nesse sentido, novos estudos podem ser realizados a fim de se obter mais resultados que auxiliem no amadurecimento desta nova tecnologia que está sendo proposta neste trabalho.

(3) Avaliação Experimental *UbiVeri*

Neste trabalho, também foi realizada uma avaliação inicial de *UbiVeri* com intuito de avaliar aspectos de interesse para engenheiros de software para compreender se a abordagem pode ser utilizada para apoiar a verificação de requisitos de ubiquidade.

Os resultados deste estudo sugerem que a abordagem é viável para apoiar a verificação de requisitos de ubiquidade em projetos de software ubíquo. Entretanto, novos estudos também podem ser realizados envolvendo novos cenários de ubiquidade de forma que se tenha mais dados indicando que *UbiVeri* aumenta de fato o número de defeitos identificados durante a revisão dos requisitos de ubiquidade do projeto ao mesmo tempo em que não onera a execução desta atividade e aumenta a satisfação do usuário em sua realização.

9.5 – Trabalhos Futuros

Como trabalhos futuros, há a intenção de continuar a pesquisa apresentada neste trabalho considerando os seguintes direcionamentos:

- Existe uma demanda por entender quais técnicas da engenharia de software associadas às atividades do desenvolvimento (por exemplo: projeto,

codificação, teste) podem ser utilizadas ou aprimoradas no sentido de apoiar o desenvolvimento de projetos de software ubíquo considerando cada uma das características da computação ubíqua. Assim, um possível trabalho futuro seria efetuar um mapeamento entre as características da computação ubíqua e técnicas de apoio às atividades do desenvolvimento. Como resultado desta pesquisa, pode-se chegar a um mapa de novas demandas para a área da engenharia de software no apoio a projetos de software ubíquo.

- Nesta pesquisa, optou-se por apoiar inicialmente as características funcionais da computação ubíqua. Entretanto, como foi destacado ao longo deste trabalho, as características restritivas também são muito importantes nesta área. Dessa forma, uma evolução natural deste trabalho é criar mecanismos de apoio à definição e verificação dos requisitos não funcionais de ubiquidade.
- Durante o desenvolvimento desta pesquisa, observou-se que as características de ubiquidade podem possuir relacionamentos de dependência e complemento entre elas. Esta questão não foi tratada inicialmente, mas é um ponto a ser observado uma vez que a presença de determinados fatores para uma dada característica podem trazer a necessidade de presença de uma outra característica da ubiquidade para o projeto.
- O corpo de conhecimento organizado neste trabalho está estruturado hoje através de descrições textuais e modelos conceituais em UML. Embora funcional, essa estrutura dificulta a atualização e a recuperação de informações do corpo de conhecimento. Neste sentido, outra questão a ser tratada seria pesquisar mecanismos que permita estruturar melhor o corpo de conhecimento em computação ubíqua.
- Atualmente, a geração dos artefatos utilizados na abordagem assim como sua customização para atender a demandas especializadas de projetos é feita manualmente através de planilhas e documentos texto. Assim, há uma necessidade de aprimorar a realização destas tarefas através da implementação de uma infraestrutura de apoio semi-automatizada.
- Por fim, ao longo desta pesquisa, notou-se também que a computação ubíqua possui um conjunto de termos técnicos associados a seu domínio que precisam ser organizados para aprimorar as discussões nesta área. Uma das possíveis formas de tratar este ponto seria estruturar uma ontologia de termos em computação ubíqua. Esta ontologia teria como ponto de partida as características identificadas neste trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

ABOWD, G. (1999). “Software engineering issues for ubiquitous computing”. Proceedings of the 21st International Conference on Software Engineering. Pages: 75 – 84, Los Angeles, USA.

ABOWD, G.D., MYNATT, E.D. (2000). “Charting Past, Present and Future Research in Ubiquitous Computing”. ACM Transactions on Computer-Human Interaction, Special issue on HCI in the new Millenium, 7(1):29-58, March.

ADOWD, G.D., MYNATT, E.D., e RODDEN, T. (2002). “The Human Experience”, IEEE Pervasive Computing, 1, 1, 48-57.

AHAMED, S.I., HAQUE, M.M., STAMM, K. (2007). “Wellness Assistant: A Virtual Wellness Assistant using Pervasive Computing”. Proceedings of the 2007 ACM Symposium on Applied Computing, March 11-15. Pages: 782 - 787, Seoul, Korea.

AIPPERSPACH, R., HOOKER, B., WOODRUFF, A. (2008). “The Heterogeneous Home”. Proceedings of the 10th International Conference on Ubiquitous Computing, September 21–24, 2008, Pages: 222-231, Seoul, Korea.

ALI, J.A., WON-SIK, Y., JAI-HOON, K., WE-DUKE, C. (2004). “U-kitchen: application scenario”. Proceedings of the Second IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, Pages: 169 – 171.

ALSOS, O.A., SVANAESM D. (2006). “Interaction Techniques for Using Handhelds and PCs Together in a Clinical Setting”. Proceedings of the 4th Nordic conference on Human-computer interaction: changing roles, Pages: 14-18.

ARAÚJO, G.M., SIQUEIRA, F. (2009). “The Device Service Bus: A Solution for Embedded Device Integration through Web Services”. Proceedings of the 2009 ACM symposium on Applied Computing, March 8-12, 2009, Pages: 185-189.

ARAÚJO, M.A., TRAVASSOS, G.H. (2008). “A System Dynamics Model based on Cause and Effect Diagram to Observe Object-Oriented Software Decay”. Technical Report ES. COPPE/UFRJ.

BANAVAR, G., BERNSTEIN, A. (2002). “Software infrastructure and design challenges for ubiquitous computing applications”. Communications of the ACM, Vol. 45, No 12. Pages: 92 – 96.

BANERJEE, N., SORBER, J., CORNER, M.D., ROLLINS, S., GANESAN, D. (2007). “Triage: Balancing Energy and Quality of Service in a Microserver”. Proceedings of the 5th International Conference on Mobile Systems, Applications and Services, June 11–14, San Juan, Puerto Rico, USA, Pages: 152-164.

BENFORD, S., MAGERKURTH C., LJUNGSTRAND P. (2005). “Bridging the physical and digital in pervasive gaming”. Communications of the ACM, Pages: 54 – 57.

BESSHO, M., KOBAYASHI, S., KOSHIZUKA, N., SAKAMURA, K. (2008). “Assisting Mobility of the Disabled using Space-Identifying Ubiquitous Infrastructure”. Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility. October 13-15, 2008, Halifax, Nova Scotia, Canada, Pages: 283-284.

BIOLCHINI, J., MIAN, P.G., NATALI, A.C.C., TRAVASSOS, G.H. (2005). “Systematic Review in Software Engineering”. Technical Report ES 679/05. COPPE/UFRJ.

BO, C., XIANG-WU, M., JUN-LIANG, C. (2007). “An Adaptive User Requirements Elicitation Framework”. Proceedings of the 31st Annual International Computer Software and Applications Conference, Vol. 2, Pages: 501-502, Beijing, China.

BOEHM, B. W., BASILI, V.R. (2001). “Software Defect Reduction Top 10 List.”, IEEE Computer 34 (1): 135-137.

BOEHM, B.W. (1981). “Software Engineering Economics”, Prentice Hall.

BOLCHINI, C., CURINO, C.C., QUINTARELLI, E., SCHREIBER, F.A., TANCA, L. (2007). “A Data-oriented Survey of Context Models”. SIGMOD Record, December 2007, Vol. 36, No. 4, Pages: 19-26.

BOSSEN, C., JORGENSEN, J.B. (2004). “Context-descriptive prototypes and their application to medicine administration”. Proceedings of the Conference on Designing interactive systems: processes, practices, methods, and techniques. Pages: 297 – 306.

BUNNINGEN, A. H. V., FENG, L., APERS, P.M.G. (2005). “Context for Ubiquitous Data Management”. International Workshop on Ubiquitous Data Management, Pages: 17-24, Enschede, Netherlands.

CAMPO, C., MUNOZ, M., PEREA, J.C., MARIN, A., GARCIA-RUBIO, C. (2005). “PDP and GSDL: A New Service Discovery Middleware to Support Spontaneous Interactions in Pervasive Systems”. Third IEEE International Conference on Pervasive Computing and Communications, Pages: 178-182, Kauai, Hawaii.

CAPPIELLO, C., COMUZZI, M., MUSSI, E., PERNICI, B. (2006). “Context Management for Adaptive Information Systems”. Electronic Notes in Theoretical Computer Science 146, Pages: 69 - 84.

CERVANTES, H., HALL, R.S. (2004). "Autonomous Adaptation to Dynamic Availability Using a Service-Oriented Component Model”. In Proceedings of 26th International Conference on Software Engineering, Pages: 614-623, Scotland, UK.

CHEN, L., BABAR, M.A., ZHANG, H. (2010). “Towards an Evidence-Based Understanding of Search Engines in Systematic Literature Reviews”. In Proceedings of 14th International Conference on Evaluation and Assessment in Software Engineering, Keele University, UK.

CHENG, B.H.C., BERRY, D.M., ZHANG, J. (2005). “The four levels of requirements engineering for and in dynamic adaptive systems”. In 11th International Workshop on Requirements Engineering Foundations for Software Quality.

CHENG, J., GOTO, Y., KOIDE, M., NAGAHAMA, K., SOMEYA, M., UTSUMI, Y., SHIONOIRI, A. (2007). “ENQUETE-BAISE: A General-Purpose E-Questionnaire Server for Ubiquitous Questionnaire”. In the Proceedings of the The 2nd IEEE Asia-Pacific Service Computing Conference, Tsukuba Science City, Japan.

CHIU, D.K.W., HONG, D., CHEUNG, S.C., KAFEZA, E. (2007). “Towards Ubiquitous Government Services through Adaptations with Context and Views in a Three-Tier Architecture”. Proceedings of the 40th Hawaii International Conference on System Sciences, Pages: 94-98.

CONCLAN, O., POWER, R., HIGEL, S., O’SULLIVAN, D., BARRETT, K. (2003). “Next Generation Context Aware Adaptive Services”. Proceedings of the 1st International Symposium on Information and Communication Technologies, Dublin, Ireland, September 24 – 26, Pages: 205 – 212.

CORRADI, A., MONTANARI, R., TIBALDI, D. (2004). "Context-Based Access Control for Ubiquitous Service Provisioning". In Proceedings of the 28th Annual International Computer Software and Applications Conference, Pages: 444-451, Hong Kong.

CORRADI, A., MONTANARI, R., TIBALDI, D., TONINELLI, A. (2005). “A Context-centric Security Middleware for Service Provisioning in Pervasive Computing”. In the Proceedings of the 2005 Symposium on Applications and the Internet, Pages: 421-429.

COUDERC, P., BANATRE, M. (2003). “Ambient computing applications: an experience with the SPREAD approach”. Proceedings of the 36th Annual Hawaii International Conference on System Sciences.

COUTAZ, J., CROWLEY, J.L., DOBSON S., GARLAN D. (2005). “Context is key”. Communications of the ACM, Pages: 49 – 53.

DARDENNE, A., LAMSWEERDE, A.V., FICKAS, S. (1993). “Goal directed requirements acquisition”. In Proceedings of the Sixth International Workshop on Software Specification and Design, Pages 3–50.

DEBATY, P. (2004). “A Toolkit To Design Adaptable User Interfaces in Ubiquitous Computing Environments”. Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, Pages: 171-175, Orlando, USA.

DEY, A. K., ABOWD, G.D. (2000). “Towards a better understanding of context and context-awareness”. In Proceedings of the 2000 Conference on Human Factors in Computing Systems, The Hague, The Netherlands, April.

DEY, A.K., ABOWD, G.D., WOOD, A. (1998). “CyberDesk: A Framework for Providing Self-integrating Context-aware Services”. Proceedings of the 3rd international conference on Intelligent user interfaces, San Francisco, California, United States, Pages: 47-54.

DIAS-NETO A.C., TRAVASSOS, G.H. (2008); “Surveying on Model Based Testing Approaches Characterization Attributes”, Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, Pages: 324-326, Alemanha.

DISZ, T., PAPKA, M.E., STEVENS, R. (1997). “UbiWorld: an environment integrating virtual reality, supercomputing, and design”. Proceedings of the Sixth Heterogeneous Computing Workshop, Pages: 46 – 57, Genebra, Suiça.

DUCATEL, K., BOGDANOWICZ, M., SCAPOLO, F., LEIJTEN, J., AND. BURGELMAN, J.-C. (2003). “Ambient Intelligence: From Vision to Reality”. IST Advisory Group Draft Rep., Eur. Comm.

FARIAS, L. L., ROCHA, A.R, TRAVASSOS, G.H. (2003). Managing Organizational Risk Knowledge. Journal of Universal Computer Science, Suiça, v. 9, n. 7, 2003.

FRANK, K., KALATZIS, N., ROUSSAKI, I., LIAMPOTIS, N. (2009). “Challenges for Context Management Systems Imposed by Context Inference”. Proceedings of

the 6th International Workshop on Managing Ubiquitous Communications and Services, June 15, Barcelona, Spain, Pages: 27-34.

FUJII, K., SUDA, T. (2004). “Dynamic Service Composition Using Semantic Information”. Proceedings of the 2nd International Conference on Service Oriented Computing, November 15–19, New York, USA, Pages: 39 - 48.

FUJINAMI, K., YAMABE, T., NAKAJIMA, T. (2004). "Take me with you!: a case study of context-aware application integrating cyber and physical spaces”. Proceedings of the 2004 ACM Symposium on Applied Computing, Pages: 1607-1614, Nicosia, Cyprus.

GARLAN D., SIEWIOREZ, D., SMAILAGIC, A., STEENKISTE, P. (2002). “Project Aura: Towards distraction-free pervasive computing”. IEEEPervasive Computing, April-June, Pages: 22 - 31.

GARLAN, D., POLADIAN, V., SCHMERL, B., e SOUSA, J.P. (2004). “Task-based Self-adaptation”. Proceedings of the ACM SIGSOFT 2004 Workshop on Self-Managing Systems, Newport Beach, CA, Oct/Nov, Pages: 54 - 57.

GLESNER, M., HOLLSTEIN, T., INDRUSIAK, L.S., ZIPF, P., PIONTECK, T., PETROV, M., ZIMMER, H., MURGAN, T. (2004). “Reconfigurable platforms for ubiquitous computing”. Conf. Computing Frontiers 2004, Pages: 377-389.

GOLDSBY, H., CHENG, B.H.C. (2007). “Goal-Oriented Modeling of Requirements Engineering for Dynamically Adaptive Systems”. 14th IEEE International Requirements Engineering Conference, Pages: 345 – 346, Minneapolis, USA.

GRASSI, V., SINDICO, A. (2007). “Towards Model Driven Design of Service-Based Context-Aware Applications”. International workshop on Engineering of software services for pervasive environments: in conjunction with the 6th ESEC/FSE joint meeting, September 4, Dubrovnik, Croatia, Pages: 69 - 74.

HALLSTEINSEN, S., STAV, E., FLOCH, J. (2004). “Self-Adaptation for Everyday Systems”. Proceedings of the 1st ACM SIGSOFT workshop on Self-managed systems, Oct 31-Nov 1, Newport Beach, CA, USA, Pages: 69 - 74.

HARRINGTON, A, CAHILL, V.. (2004). “Route profiling: putting context to work”. Proceedings of the 2004 ACM Symposium on Applied Computing, Pages: 1567 – 1573, Nicosia, Cyprus.

HATALA, M., WAKKARY, R., KALANTARI, L., 2005. “Rules and Ontologies in Support of Real-time Ubiquitous Application”. Journal of Web Semantics, Pages 5-22.

HODES, T.D., KATZ, R.H., SERVAN-SCHREIBER, E., ROWE, L.A. (1997). “Composable ad-hoc Mobile Services for Universal Interaction”. Proceedings of the 3rd Annual ACM/IEEE International Conference on Mobile Computing and Networking, Pages: 1 – 12, Budapest, Hungary.

HONG, D., CHIU, D.K.W., SHEN, V.Y. (2005). “Requirements Elicitation for the Design of Context-aware Applications in a Ubiquitous Environment”. Proceedings of the 7th international conference on Electronic commerce. China, Pages: 590 - 596.

HONG, D., CHIU, D.K.W., SHEN, V.Y. (2005). “Requirements Elicitation for the Design of Context-Aware Applications in a Ubiquitous Environment”. Proceedings of the 7th International Conference on Electronic Commerce, August 15–17, Xi’an, China, Pages: 590 - 596.

HUANG, A.C., BENJAMIN, C.L., BARTON, J.J., FOX, A. (2001). "Making Computers Disappear: Appliance Data Services". ACM SIGMOBILE Seventh Annual International Conference on Mobile Computing and Networking (Mobicom), Rome Italy, Pages: 108 - 121.

IEEE Standard 610-1990: IEEE Standard Glossary of Software Engineering Terminology, IEEE Press.

IEEE, (1998). “IEEE Recommended Practice for Software Requirements Specifications - Description, Standard 830”, IEEE Press.

JAENA, J., MOCHOLIA, J.A., CATALA, A., NAVARRO, E. (2009). “Digital ants as the best cicerones for museum visitors”. Applied Soft Computing. Volume 11, Issue 1, Pages: 111-119.

JEA, S., YAP, I., SRIVASTAVA, M.B. (2007). “Context-aware Access to Public Shared Devices”. HealthNet’07, June 11, 2007, San Juan, Puerto Rico, USA, Pages: 13 - 18.

JOEL, S., ARNOTT, J.L., HINE, N.A., INGVARSSON, H., RENTOUL, R., SCHOFIELD, S. (2004). “A framework for analyzing interactivity in a remote access field exploration system”. IEEE International Conference on Systems, Man and Cybernetics, Pages: 2669 – 2674.

JOHANSEN, D., JOHANSEN, H.D., RENESSE, R. (2004). “Environment mobility: moving the desktop around”. Proceedings of the 2nd Workshop on Middleware for Pervasive and ad-hoc Computing, Toronto, Ontario, Canada, Pages: 150-154.

JOHNSON, V.M., VERNIK, R. (2004). “e-Ghosts: Leaving Virtual Footprints in Ubiquitous Workspaces”. In Proceedings of the Fifth Australasian User Interface

Conference (AUIC2004), Dunedin, New Zealand. CRPIT, 28. Cockburn, A., Ed. ACS. Pages: 111-116.

JORGENSEN, J.B., BOSSEN, C. (2003). “Requirements Engineering for a Pervasive Health Case System”. In the Proceedings of the 11th IEEE International Requirements Engineering Conference, Monterey, CA, USA.

JOSE, R., MOREIRA, A., RODRIGUES, H., DAVIES, N. (2003). “The AROUND architecture for dynamic location-based services”, Mobile Networks and Applications, v.8 n.4, Pages: 377-387.

KAGAL, L., KOROLEV, V., AVANCHA, S., JOSHI, A., FININ, T., e YESHA, Y. (2002). “Centaurus: An Infrastructure for Service Management in Ubiquitous Computing Environments”, Wireless Networks, 8:6, Kluwer, Pages: 619 - 635.

KALAPRIYA, K., NANDY, S.K., SRINIVASAN, D., MAHESHWARI, R., SATISH, V. (2004). “A framework for resource discovery in pervasive computing for mobile aware task execution”, Proceedings of the 1st Conference on Computing Frontiers, Pages: 70-77, Ischia, Italy.

KALINOWSKI, M., SPÍNOLA, R.O., TRAVASSOS, G.H. (2004). “Infra-Estrutura Computacional para Apoio ao Processo de Inspeção de Software”. No: Simpósio Brasileiro de Qualidade de Software, Brasília, Brasil.

KEIDL, M., KEMPER, A. (2004). “Towards context-aware adaptable web services”, Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters, New York, NY, USA, Pages: 55-65.

KEMP, A.E., THOMPSON, A.J., JOHNSON, T.S. (2008). “Interface Evaluation for Invisibility and Ubiquity – an example from E-learning”. Proceedings of the 9th ACM SIGCHI New Zealand Chapter's International Conference on Human-Computer Interaction: Design Centered HCI, July 2, Wellington, New Zealand, Pages: 31-38.

KHEDR, M., KARMOUCH, A. (2003). “Exploiting SIP and agents for smart context level agreements”. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, Victoria, BC, Canada, Pages: 522 - 525.

KIENTZ, J. A., BORING, S., ABOWD, G. D., HAYES, G. R. (2005) “Abaris: Evaluating Automated Capture Applied to Structured Autism Interventions”. In the Proceedings of the International Conference on Ubiquitous Computing, September 11-14, Tokyo, Japan.

KIENTZ, J.A., HAYES, G.R., WESTEYN, T.L., STARNER, T., and ABOWD, G.D. (2007). “Pervasive Computing and Autism: Assisting Caregivers of Children with Special Needs”. IEEE Pervasive Computing. Special Issue on Pervasive Healthcare, January, Pages 28-35.

KIM, K.I., CHOI, W.G., LEE, E.J., KIM, U.M. (2009). “RBAC-Based Access Control for Privacy Protection in Pervasive Environments”. In: Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication, Suwon, Korea, Pages: 255-259.

KIMMO, E.E., Christensen, H.B., NAKAJIMA, T. (2002). “Application requirements for middleware for mobile and pervasive systems”. Mobile Computing and Communications Review 6(4), Pages: 16-24.

KINDBERG, T., BARTON, J., BECKER, G., CASWELL, D., DEBATY, P., GOPAL, G., FRIG, M., KRISHNAN, V., MORRIS, H., SCHETTINO, J., SERRA, B., SPASOJEVIC, M. (2000). “People, places, things: Web presence for the real world”, Third IEEE Workshop on Mobile Computing Systems and Applications, Los Alamitos, CA , USA, Pages: 19 – 28.

KINDBERG, T., e FOX, A. (2002). “System software for ubiquitous computing”. IEEE Pervasive Computing, 1(1):70-81, January-March 2002.

KITCHENHAM, B. (2004). Procedures for Performing Systematic Reviews. Technical report, Keele University, Australia.

KWAN, V., LAU, F.C.M., WANG, C. (2003). “Functionality Adaptation: A Context-Aware Service Code Adaptation for Pervasive Computing Environments”. IEEE/WIC International Conference on Web Intelligence, Halifax, Canada, Pages: 358-364.

LEE, S.H., CHUNG, T.C. (2004). “System Architecture for Context-Aware Home Application”. Second IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, Vienna, Austria.

LI, Y., LANDAY, J.A. (2008). “Activity-Based Prototyping of Ubicomp Applications for Long-Lived, Everyday Human Activities”. Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems, April 5–10, Florence, Italy, Pages: 1303-1312.

LIU, D., PENG, J., LAW, K.H., WIEDERHOLD, G. (2004). “Efficient Integration of Web Services with Distributed Data Flow and Active Mediation”. Sixth International Conference on Electronic Commerce, Delft, Netherlands, Pages: 11 - 20.

MAAMAR, Z., MOSTEFAOUI, S.K., YAHYAOUI, H. (2004). “A Web services composition approach based on software agents and context”. Proceedings of the 2004 ACM symposium on Applied computing, Nicosia, Cyprus, Pages: 1619-1623.

MAFRA, S. N., BARCELOS, R.F., TRAVASSOS, G.H. (2006). “Aplicando uma Metodologia Baseada em Evidencia na Definição de Novas Tecnologias de Software”. In: Simposio Brasileiro de Engenharia de Software, Florianopolis, SC, Brasil, v. 1, Pages: 239-254.

MAFRA, S.N., TRAVASSOS, G. H. (2006). “Leitura Baseada em Perspectiva: A Visão do Projetista Orientada a Objetos”. In: SBQS - Simpósio Brasileiro de Qualidade de Software, 2006, Vila Velha, ES, Brasil.

MANTORO, T., JOHNSON, C. (2003). “Location History in a Low-cost Context Awareness Environment”. Proceedings of the Australasian information security workshop conference on ACSW frontiers, Adelaide, Australia, Pages: 153-158.

MARTINS, D.S., BIAJIZ, M., PRADO, A.F., SOUZA, W.L. (2009). “Implicit Relevance Feedback for Context Aware Information Retrieval in UbiLearning Environments”. Proceedings of the 2009 ACM symposium on Applied Computing, March, Honolulu, Hawaii, U.S.A, Pages: 659-663.

MCLEAN, P.G. (2003). “A secure pervasive environment. Australasian Information Security Workshop”. Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003, Adelaide, Australia, Pages: 67 - 75.

MOTAHARI, S., MANIKOPOULOS, C., HILTZ, R., JONES, Q. (2007). “Seven Privacy Worries in Ubiquitous Social Computing”. Symposium On Usable Privacy and Security (SOUPS), July 18-20, Pittsburgh, PA, USA, Pages: 171 - 172.

NAWYN, J., INTILLE, S., AND LARSON, K. (2006). "Embedding Behavior Modification Strategies into Consumer Electronic Devices: A Case Study". In Proceedings of the 8th International Conference on Ubiquitous Computing (UBICOMP), Orange County, CA, USA.

NIEMELA, E., LATVAKOSKI, J. (2004). “Survey of requirements and solutions for ubiquitous software”, Proceedings of the 3rd International Conference on Mobile and Ubiquitous Multimedia, Pages: 71-78, October 27-29, College Park, Maryland.

NORBISRATH, U., ARMAC, I., RETKOWITZ, D., SALUMAA, P. (2006). “Modeling eHome Systems”. Proceedings of the 4th International Workshop on Middleware for Pervasive and Ad-Hoc Computing, November 27-December 1, 2006 Melbourne, Australia.

NUSEIBEH, B. A., e EASTERBROOK, S. M. (2000). "Requirements Engineering: A Roadmap", In A. C. W. Finkelstein (ed) "The Future of Software Engineering ". (Companion volume to the proceedings of the 22nd International Conference on Software Engineering, ICSE'00). IEEE Computer Society Press.

OLIVEIRA, K.M., ZLOT, F., ROCHA, A.R.C., TRAVASSOS, G.H., SILVA, C.G.M., MENEZES, C.S., (2004). Domain Oriented Software Development Environment. Journal Of Systems And Software, v. 72, n. 2, p. 145-161.

O'NEILL, E., KINDBERG, T., SCHIECK, A. F. GEN, JONES, T., PENN, A., AND FRASER, D. S. (2006). “Instrumenting the city: developing methods for observing and understanding the digital cityscape”. In Proceedings of the 8th International Conference on Ubiquitous Computing (UBICOMP), Orange County, CA, USA.

PARK, P., LEE, H.E. (2009). “u-healthcare Aide System for Ubiquitous Wellbeing Lifecare Smart Space”. International Conference on New Trends in Information and Service Science, Beijing, China, Pages: 781-783.

PFLEEGER, S. (2007). “Software Engineering: Theory and Practice”. Second Edition. Prentice Hall.

PINTO, F.C.R. (2009). “Uma Abordagem para Apoiar Especificação de Requisitos para Projetos de Software Ubíquo”. Dissertação de Mestrado. COPPE/UFRJ. Rio de Janeiro.

POLADIAN, V., SOUSA, J.P., GARLAN, D., SHAW, M. (2004). “Dynamic Configuration of Resource-Aware Services”. Proceedings of the 26th International Conference on Software Engineering, Pages: 604-613, Scotland, UK.

POPESCU-ZELETIN, R., STEGLICH, S., ARBANOWSKI, S. (2004). “Pervasive Communication - A Human-Centered Service Architecture”. Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems, Pages:140-146, Suzhou, China.

POUSMAN, Z., STASKO, J. (2006). “A Taxonomy of Ambient Information Systems: Four Patterns of Design”. Proceedings of the Working Conference on Advanced Visual Interfaces, Venezia, Italy, Pages: 67 - 74.

RAENTO, M., OULASVIRTA, A., PETIT, R. e TOIVPNEN, H. (2005). “ContextPhone: A Prototyping Platform for Context-Aware Mobile Applications.” IEEE Pervasive Computing, 4(2), Pages: 51-59.

RAMDANE-CHERIF, A., HINA, M.D., TADJ, C., LÉVY, N. (2007). “Analysis of a New Ubiquitous Multimodal Multimedia Computing System”. Ninth IEEE International Symposium on Multimedia, Pages: 161 – 168.

RANGANATHAN, A., CHETAN, S., CAMBELL, R. (2004). “Mobile polymorphic applications in ubiquitous computing environments,” The First Annual International Conference on Mobile and Ubiquitous Systems, Pages: 402 – 411, Boston, Massachusetts, USA.

RAPTIS, D., TSELIOS, N., AVOURIS, N. (2005). “Context-based design of mobile applications for museums: a survey of existing practices”. In Proceedings of the 7th international Conference on Human Computer interaction with Mobile Devices &Amp; Services (Salzburg, Austria, September 19 - 22, 2005). MobileHCI '05, vol. 111. ACM Press, New York, NY, Pages: 153-160.

RATSIMOR, O., CHAKRABORTY, D., JOSHI, A., FININ, T. (2002). “Allia: alliance-based service discovery for ad-hoc environments”, Proceedings of the second International Workshop on Mobile Commerce, Atlanta, Georgia, USA, Pages: 1-9.

REPO, P. (2004). “Facilitating user interface adaptation to mobile devices”. Proceedings of the third Nordic Conference on Human-Computer Interaction, October 23-27, 2004, Tampere, Finland, Pages: 433-436.

REPO, P., RIEKKI, J. (2004). “Middleware support for implementing context-aware multimodal user interfaces”. Proceedings of the 3rd International Conference on Mobile and Ubiquitous Multimedia, October 27-29, 2004, College Park, Maryland, Pages: 221-227.

ROMAN, M., ISLAM, N. (2004). “Dynamically programmable and reconfigurable middleware services”, Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware, October 18-22, 2004, Toronto, Canada, Pages: 372 - 396.

ROMAN, M., ZIEBART, B., CAMPBELL, R.H. (2003). “Dynamic Application Composition: Customizing the Behavior of an Active Space”. Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, Pages: 169 – 176, Texas, USA.

ROY, N., PALLAPA, G., DAS, S.K. (2008). “An Ontology-Driven Ambiguous Contexts Mediation Framework for Smart Healthcare Applications”. Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments, July 15-19, 2008, Athens, Greece.

RUSSEL D.M., STREITZ N.A., WINOGRAD T. (2005). “Building disappearing computers”. Communications of the ACM, Volume 48 , Issue 3, Pages: 42 – 48.

RUTA, M., NOIA, T., SCIASCIO, E., PAOLUCCI, M., SCIOSCIA, F., TINELLI, E. (2008). “A Semantic-based Registry enabling Discovery, Composition and Substitution of Pervasive Services”. In: Proceedings of the Seventh ACM International Workshop on Data Engineering for Wireless and Mobile Access. Vancouver, Canada, Pages: 63-70.

SADJADI, S.M., MCKINLEY, P.K., CHENG, B.H.C. (2005). “Transparent Shaping of Existing Software to Support Pervasive and Autonomic Computing”. Proceedings of the 2005 workshop on Design and evolution of autonomic application software, May 21, 2005, St. Louis, Missouri, USA, Pages: 1 - 7.

SAKAMURA, K. (2006) “Challenges in the Age of Ubiquitous Computing: A Case Study of T-Engine, An Open Development Platform for Embedded Systems”. Proceeding of the 28th International Conference on Software Engineering (ICSE), Shanghai, China. Pages: 713 - 720.

SALOVAARA, A., OULASVIRTA, A. (2004). “Six Modes of Proactive Resource Management: A User-Centric Typology for Proactive Behaviors”. Proceedings of the third Nordic conference on Human-computer interaction, October 23-27, 2004 Tampere, Finland, Pages: 57 - 60.

SAMESHIMA, S., KAWANO, K., TAKASHIO, K., MORIKAWA, H., MINAM, M. (2003). “Opportunities and issues relating to middleware technologies for context-aware service”. IEEE, SICE Annual Conference in Fukui, August 4-6, Pages: 2704 - 2708.

SANTOS, L.O.S., WIJNEN, R.P., VINK, P. (2007). “A Service-Oriented Middleware for Context-Aware Applications”. Proceedings of the 5th international workshop on Middleware for pervasive and ad-hoc computing: held at the ACM/IFIP/USENIX 8th International Middleware Conference, November 26–30, 2007, Newport Beach, CA, USA, Pages: 37-42.

SATOH, I. (2004). "A Mobile Agent-based Framework for Location-Based Services", Proceedings of IEEE International Conference on Communications (ICC'2004), IEEE Communication Society, June, Pages: 1355 - 1359.

SHARMIN, M., AHMED, S., AHAMED, S.I. (2005). “SAFE-RD (Secure, Adaptive, Fault Tolerant, and Efficient Resource Discovery) in a Pervasive Computing Environments”. Proceedings of the International Conference on Information Technology: Coding and Computing, Pages: 271 – 276, Wuhan, China.

SHIN, J.H., CHUNG, G.S., KIM, K.K., KIM, J.S., HWANG, B.S., PARK, K.S. (2007). “Ubiquitous House and Unconstrained Monitoring Devices for Home Healthcare System”. 6th International Special Topic Conference on Information Technology Applications in Biomedicine, Pages: 201 – 204.

SHULL, F., CARVER, J., TRAVASSOS, G.H. (2001). “An Empirical Methodology for Introducing Software Processes”. In: Proceedings of the Joint 8th European Software Engineering Conference (ESEC) and 9th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-9), Pages: 288-296.

SHULL, F., RUS, I., BASILI, V. (2000). “How Perspective-Based Reading Can Improve Requirements Inspections”, July, IEEE Software, Pages: 73-79.

SILVA, L.F.S., CHAPETTA, W.A., TRAVASSOS, G.H. (2004). “Inspeções de Requisitos de Software Utilizando PBR e Apoio Ferramental”. In: III Simposio Brasileiro de Qualidade de Software, 2004, Brasília. Anais do SBQS. Porto Alegre: Sociedade Brasileira de Computação, 2004. v. 1. p. 42-57.

SMANCHAT, S., LING, S., INDRAWAN, M. (2008). “A survey of context-aware workflow adaptations”. Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia, November 24–26, 2008, Linz, Austria, Pages: 414-417.

SPÍNOLA, R.O., and TRAVASSOS, G.H. (2010). “Characteristics of Ubiquitous Software Projects: Pertinence, Relevance, and Use”. In Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering (SEKE 2010), San Francisco Bay, CA, USA.

SPÍNOLA, R.O., PINTO, F.C.R., TRAVASSOS, G.H. (2008). “Apoio às Atividades de Especificação e Verificação de Requisitos Funcionais de Ubiquidade em Projetos de Software”. In: International Information and Telecommunication Technologies Symposium - I2TS’2008, Foz do Iguaçu.

SPÍNOLA, R.O., PINTO, F.C.R., TRAVASSOS, G.H. (2008). “Supporting Requirements Definition and Quality Assurance in Ubiquitous Software Project”. Communications in Computer and Information Science, 2009, Volume 17, Part 10, 587-603, DOI: 10.1007/978-3-540-88479-8_42.

SPÍNOLA, R.O., PINTO, F.C.R., TRAVASSOS, G.H. (2010). “UbiCheck: An Approach to Support Requirements Definition in the Ubicomp Domain”. In: 25th Symposium On Applied Computing, 2010, Sierre, Suiça. 25th Symposium On Applied Computing, 2010. Pages: 306-310.

SPÍNOLA, R.O., SILVA, J.L.M., TRAVASSOS, G.H. (2006) “Towards a Conceptual Framework to Classify Ubiquitous Software Projects”. Proceedings of the 8th International Conference on Software Engineering and Knowledge Engineering (SEKE), San Francisco, USA.

SPÍNOLA, R.O., SILVA, J.L.M., TRAVASSOS, G.H. (2007a). “Checklist to Characterize Ubiquitous Software Projects”. Proceedings of the XXI Brazilian Symposium on Software Engineering, João Pessoa, Brasil.

SPÍNOLA, R.O., SILVA, J.L.M., TRAVASSOS, G.H. (2007b). “Characterizing Ubicomp Software Projects through a Checklist”. In Proceedings of the I Workshop on Pervasive and Ubiquitous Computing, Gramado, Brasil.

SPÍNOLA, R.O., TRAVASSOS, G.H. (2008). “Arcabouço para Apoiar a Definição e a Garantia de Qualidade de Requisitos de Ubiquidade em Projetos de Software”. In: II Workshop on Pervasive and Ubiquitous Computing, 2008, Campo Grande, Brasil.

STOVALL, D., JULIEN, C. (2007). “Resource Discovery with Evolving Tuples”. International workshop on Engineering of software services for pervasive environments: in conjunction with the 6th ESEC/FSE joint meeting, September 4, 2007, Dubrovnik, Croatia, Pages: 1 - 10.

STREEFKERK, J.W., ESCH-BUSSEMAKERS, M.P., NEERINCX, M.A. (2006). “Designing personal attentive user interfaces in the mobile public safety domain”. Computers in Human Behavior 22 (2006), Pages: 749–770.

SU, M., ZHANG, H., LIN, Y., SU, Y., CHEN, S., CHEN, H. (2009). “Pilot Study on a Community-Based Ubiquitous Healthcare System for Current and Retired University Employees”. IEEE International Conference on Communications Workshops, Pages: 1-5, Kauai Island, HI, USA.

SULTANA, S., AKBAR, M.M., KARIM, R., AHAMED, S. (2008). “Ubicomp Secretary: A Web Service based Ubiquitous Computing Application”. Proceedings of the 2008 ACM symposium on Applied computing, March 16-20, 2008, Fortaleza, Ceará, Brazil, Pages: 1935-1939.

SUN, J., WU, Z. (2005). “A Comprehensive Context Model for Next Generation Ubiquitous Computing Applications”. Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, 2005., Pages: 447-450.

TAHTI, M., RAUTIO, V., ARHIPPAINEN, L. (2004). “Utilizing context-awareness in office-type working life”. Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia, College Park, Maryland, Pages: 79 - 84.

THAYER, S.M., STEENKISTE, P. (2003). “An Architecture for the Integration of Physical and Informational Spaces”. Personal and Ubiquitous Computing 7(2), Pages: 82-90.

TRUONG, K.N., ABOWD, G.D., BROTHERTON, J.A. (1999). “Personalizing the Capture of Public Experiences”. ACM Symposium on User Interface Software and Technology 1999, Asheville, North Carolina, United States, Pages: 121-130.

VAINIO, A.M., VALTONEN, M., VANHALA, J. (2006) “Learning and adaptive fuzzy control system for smart home”. Proceedings of the AmI.d, September 20-22, Pages: 28-47.

VALE, S., HAMMOUDI, S. (2008). “Towards Context Independence in Distributed Context-aware Applications by the Model Driven Approach”. Proceedings of the 3rd international workshop on Services integration in pervasive environments, July 7, 2008, Sorrento, Italy, Pages: 31-36.

VEIGA, L., FERREIRA, P. (2004). “PoliPer: policies for mobile and pervasive environments”. Proceedings of the 3rd workshop on Adaptive and reflective middleware, Toronto, Ontario, Canada, Pages: 238-243.

VENKATASUBRAMANIAN, N. (2002). “Safe 'composability' of middleware services”. Communications of the ACM, 2002. 45(6): p. 49-52.

WEIS, T., STERNUS, M., KNOLL, M., BRANDLE, A., COMBETTO, M. (2006). “Towards a General Purpose User Interface for Service-oriented Context-aware Applications”. Proceedings of the international workshop in conjunction with AVI 2006 on Context in advanced interfaces, May 23, 2006, Venice, Italy, Pages: 53 - 55.

WEISER M. (1991). “The Computer for the 21st Century”. Scientific American 1991, Pages: 94-104.

WHEELER, D.A., BRYKEZYNSKI, B., MEESON, R.N. (1996). “Software Inspections: An Industry Best Practice”, IEEE Computer Society.

XIANG, J., LIU, L., QIAO, W., YANG, J. (2007). “SREM: A Service Requirements Elicitation Mechanism based on Ontology”. In the Proceedings of the 31st Annual International Computer Software and Applications Conference, Beijing, China.

YAN, Z. (2008). “A Comprehensive Trust Model for Component Software”. Proceedings of the 4th international workshop on Security, privacy and trust in pervasive and ubiquitous computing, July 7, 2008, Sorrento, Italy, Pages: 1-6.

YANG, M.-J., CHEN, J.-H., WANG, T.-H., CHAO, L.R., SHIH, T. K. (2009). “To Construct the Outdoor Experience Game-Based”. Proceedings of the 2009 workshop on Ambient media computing, Beijing, China, Pages: 77-82.

YAU, S.S., KARIM, F. (2001). “Context-Sensitive Middleware for Real-Time Software in Ubiquitous Computing Environments”. Proceedings of the Fourth International Symposium on Object-Oriented Real-Time Distributed Computing, Pages: 163-170, Hawaii.

YAU, S.S., KARIM, F. (2003). “An Energy-Efficient Object Discovery Protocol for Context-Sensitive Middleware for Ubiquitous Computing”. IEEE Trans. Parallel Distrib. Syst. 14(11): 1074-1085.

YAU, S.S., YU, W., KARIM, F. (2002). “Development of situation-aware application software for ubiquitous computing environments”. Proceedings of the 26th Annual International Computer Software and Applications Conference. COMPSAC 2002. Pages: 233 – 238, Oxford, England.

ZHOU, P., NADEEM, T., KANG, P., BORCEA, C., IFTODE, L. (2005). “EZCab: A Cab Booking Application Using Short-Range Wireless Communication”. Third IEEE International Conference on Pervasive Computing and Communications (PerCom 2005), Kauai Island, HI, Page(s):27 - 38.

ZHOU, Y., RAYCHOUDHURY, V., SIEBERT, J., LU, J. (2007). “A Middleware Support for Agent-Based Application Mobility in Pervasive Environments”. 27th International Conference on Distributed Computing Systems Workshops (ICDCSW'07), Page: 9, Toronto, Ontario, Canada.

APÊNDICE A – ARTIGOS UTILIZADOS NAS REVISÕES SISTEMÁTICAS

Este apêndice apresenta a lista de artigos utilizados nas revisões sistemáticas executadas neste trabalho descritas no Capítulo 2.

A.1 – Revisão Sistemática: Características da Computação Ubíqua

A tabela abaixo lista os artigos selecionados para extração de informação da revisão sistemática sobre características da computação ubíqua.

Título do Artigo	Autores	Fonte	Ano
Questioning Ubiquitous Computing	Agustin A. Araya	ACM 23rd annual conference on Computer science	1995
Context-descriptive Prototypes and Their Application to Medicine Administration	Claus Bossen, Jens Bæk Jørgensen	Designing Interactive Systems: Across the Spectrum	2004
Project Aura: Toward Distraction-Free Pervasive Computing	David Garlan, *et al*.	Pervasive Computing	2002
Semantically Driven Service Interoperability for Pervasive Computing	Declan O'Sullivan, David Lewis	MobiDE	2003
Interaction in Pervasive Computing Settings using Bluetooth-Enabled Active Tags and Passive RFID Technology together with Mobile Phones	Frank Siegemund, Christian Florkemeier	PerCom	2003
Fuzzy control interoperability and scalability for adaptive domotic framework	Giovanni Acampora, Vincenzo Loia	Transactions on Industrial Informatics	2005
Software Engineering Issues for Ubiquitous Computing	Gregory D. Abowd	ICSE	1999
The Human Experience	Gregory D. Abowd, Elizabeth D. Mynatt, Tom Rodden	Pervasive Computing	2002
Software Infrastructure and Design Challenges for Ubiquitous Computing Applications	Guruduth Banavar, Abraham Bernstein	CACM	2002
Software Infrastructure and Design Challenges for Ubiquitous Computing Applications	Guruduth Banavar,Abraham Bernstein	CACM	2002
Utilizing Context History to Provide Dynamic Adaptations	Hee Eon Byun, Keith Cheverst	Applied Artificial Intelligence	2004
A Service Selection Method Based on	Hiroaki Kawamichi,	International	2004

Context Types for a Ubiquitous Service System in a Public Space	et al.	Symposium on Applications and the Internet	
Mobile Applications in Ubiquitous Computing Environments	Ichiro SATOH	IEICE Transactions on Communications	2005
U-Kitchen: Application Scenario	Junaid Ahsen Ali, Won-Sik Yoon, Jai-Hoon Kim, We-Duke Cho	Workshop on Software Technologies for Future Embedded and Ubiquitous Systems	2004
Issues and Challenges in Ubiquitous Computing	Kalle Lyytinen, Youngjin Yoo	CACM	2002
Issues and Challenges in Ubiquitous Computing	Kalle Lyytinen,Youngjin Yoo	CACM	2002
Application Requirements for Middleware for Mobile and Pervasive Systems	Kimmo Raatikainen, Henrik Baerbak Christensen, Tatsuo Nakajima	ACM SIGMOBILE	2002
Ubiquitous Access to Reconfigurable Hardware: Application Scenarios and Implementation Issues	Leandro Soare Indrusiak, Florian Lubitz, Ricardo Reis, Manfred Glesner	ATECE	2003
NIST Smart Space: Pervasive Computing Initiative	Lynne Rosenthal, Vincent Stanford	WET ICE	2000
Rules and ontologies in support of real-time ubiquitous application	Marek Hatala, Ron Wakkary, Leila Kalantari	Web Semantics	2005
Utilizing Context-Awareness in Office-Type Working Life	Marika Tahti, Ville-Mikko Rautio, Leena Arhuppainen	3rd international conference on Mobile and ubiquitous multimedia	2004
Some Computer Science Issues in Ubiquitous Computing	Mark Weiser	CACM	1993
The Computer for the 21st Century	Mark Weiser	ACM SIGMOBILE	1999
Some Computer Science Issues in Ubiquitous Computing	Mark Weiser	CACM	1993
The Disappearing Computer	Norbert Streitz,Paddy Nixon	CACM	2005
Ambient computing applications: an experience with the SPREAD approach.	Paul Couder, Michel Banâtre	ICSS	2003
EZCab: A Cab Booking Application Using Short-RangeWireless Communication	Peng Zhou, Tamer Nadeem, Porlin Kang, Cristian Borcea, and Liviu Iftode	PerCom	2005
Constraining Event Flow for Regulation in Pervasive Systems	Prashant S. Kumar, Qiang Zeng and Gurdip Singh	PerCom	2005
A Framework for Analysing Interactivity in a Remote Access Field Exploration System	S. Joel, J.L. Amott, N. Hine, S. Ingvarsson, R. Rentoul and S. Schofield	IEEE International Conference on Systems, Man and Cybernetics	2004

System Architecture for Context-Aware Home Application	Sang-Hak Lee, Tae-Choong Chung	Workshop on Software Technologies for Future Embedded and Ubiquitous Systems	2004
Development and Runtime Support for Situation-Aware Application Software in Ubiquitous Computing Environments	Stephen S. Yau, Dazhi Huang, Haishan Gong, Siddharth Seth	COMPSAC	2004
Development of Situation-Aware Application Software for Ubiquitous Computing Environments	Stephen S. Yau, Yu Wang, Fariaz Karim	COMPSAC	2002
UbiWorld: An Environment Integrating Virtual Reality, Supercomputing, and Design	Terrence Disx, Michael E. Papka, Rick Stevens	ISHPDC	1996
UbiWorld: An Environment Integrating Virtual Reality, Supercomputing, and Design	Terrence Disz, Michael E. Papka, and Rick Stevens	HCW	1997
System Software for Ubiquitous Computing	Tim Kindberg, Armando Fox	Pervasive Computing	2002
People, Places, Things: Web presence for Real World	Tim Kindberg, *et al.*	Workshop on Mobile Computing Systems and Applications	2000
Dynamic Configuration of Resource-Aware Services	Vahe Poladian, João Pedro Sousa, David Garlan, Mary Shaw	ICSE	2004
Pervasive Computing Puts Food on the Table	Vince Stanford	IEEE Pervasive Computing	2003
AmbientDB: P2P Data Management Middleware for Ambient Intelligence	Willem Fontijn, Peter Boncz	PerCom	2004
User interfaces for telepresentations—Input devices, interaction concepts and design issues	Wolfgang Hurst	Journal of Network and Computer Applications	2000
An Approach to Enhancing Mobile Applications with Situation-Awareness Capabilities	Yu Wang	International Conference on Parallel Processing	2004

A.2 – Revisão Sistemática: Fatores Funcionais e Restritivos

A tabela abaixo lista os artigos selecionados para extração de informação da revisão sistemática sobre fatores funcionais e restritivos associados às características da computação ubíqua.

Título do Artigo	Autores	Fonte	Ano
Mobile Polymorphic Applications in Ubiquitous Computing Environments	Anand Ranganathan, Shiva Chetan, Roy Campbell	MobiQuitous	2004
Making Computers Disappear: Appliance Data Services	Andrew C. Huang, Benjamin C. Ling, John Barton, Armando Fox	MobiCom	2001

CyberDesk: A Framework for Providing Self-Integrating Context-Aware Services	Anind K. Dey, Gregory D. Abowd, Andrew Wood	Intelligent User Interfaces	1998
Route Profiling – Putting Context to Work	Anthony Harrington, Vinny Cahill	SAC	2004
A context-centric security middleware for service provisioning in pervasive computing	Antonio Corradi *et al.*	Symposium on Applications and the Internet	2005
Context-based access control for ubiquitous service provisioning	Antonio Corradi, Rebecca Montanari, Daniela Tibaldi	COMPSAC	2004
Six Modes of Proactive Resource Management: A User-Centric Typology for Proactive Behaviors	Antti Salovaara, Antti Oulasvirta	NordiCHI '04	2004
Context for Ubiquitous Data Management	Arthur H. van Bunningen, Ling Feng e Peter M. G. Apers	International Workshop on Ubiquitous Data Management	2005
PDP and GSDL: a new service discovery middleware to support spontaneous interactions in pervasive systems	Celeste Campo *et al.*	PerCom	2005
Environment Mobility – Moving the Desktop Around	Dag Johansen, Hävard Johansen, Robbert van Renesse	Workshop on Middleware for Pervasive and Ad-Hoc Computing	2004
Requirements Elicitation for the Design of Context-Aware Applications in a Ubiquitous Environment	Dan Hong, Dickson K.W. Chiu, Vincent Y. Shen	ICEC	2005
Task-based Self-adaptation	David Garlan *et al.*	WOSS'04	2004
Project Aura: Toward Distraction-Free Pervasive Computing	David Garlan, Daniel P. Siewiorek, Asim Smailagic, Peter Steenkiste	IEEE Pervasive Computing	2002
Efficient Integration of Web Services with Distributed Data Flow and Active Mediation	David Liu, Jun Peng, Kincho H. Law, Gio Wiederhold	ICEC'04	2004
Survey of Requirements and Solutions for Ubiquitous Software	Eila Niemelä , Juhani Latvakoski	International Conference on Mobile and Ubiquitous Multimedia	2004
Towards Distributed Awareness – An Artifact based Approach	Florian Michahelles, et al.	WMCSA	2004
The Human Experience	Gregory D. Abowd, Elizabeth D. Mynatt, Tom Rodden	IEEE Pervasive Computing	2002
Autonomous Adaptation to Dynamic Availability Using a Service-Oriented Component Model	Humberto Cervantes, Richard S. Hall	ICSE	2004
A mobile Agent-based framework for location-based services	Ichiro Satoh	PerCom	2005
A comprehensive context model for next generation ubiquitous computing applications	Jie Sun, ZhaoHui Wu	RTCSA	2005
A framework for resource discovery in pervasive computing for mobile aware task execution	K.Kalapriya *et al.*	CF'04	2004
"Take me with you!": A case study of	Kaori Fujinami,	SAC	2004

context-aware application integration cyber and physical spaces	Tetsuo Yamabe, Tatsuo Nakajima		
Dynamic Service Composition Using Semantic Information	Keita Fujii, Tatsuya Suda	ICSOC'04	2004
Personalizing the Capture of Public Experiences	Khai N. Truong, Gregory D. Abowd, Jason A. Brotherton	UIST ’99	1999
Application Requirements for Middleware for Mobile and Pervasive Systems	Kimmo Raatikainen, Henrik Bærbak Christensen, Tatsuo Nakajima	Mobile Computing and Communications Review	2002
Centaurus: An Infastructure for service management in ubiquitous computing environments	Lalana Kagal *et al.*	Wireless Networks	2002
PoliPer: Policies for Mobile and Pervasive Environments	Lu´ýs Veiga, Paulo Ferreira	Workshop on Adaptive and Reflective Middleware	2004
Exploiting co-location history for efficient service selection in ubiquitous computing systems	M. Khedr, A. Karmouch	PerCom	2005
Exploiting Agents and SIP for Smart Context Level Agreements	M. khedr, A. Karmouch	Pacific Rim Conference on Communications, Computers and Signal Processing	2003
Reconfigurable Platforms for Ubiquitous Computing	Manfred Glesner *et al.*	CF’04	2004
Dynamic Application Composition: Customizing the Behavior of an Active Space	Manuel Román, Brian Ziebart, Roy H. Campbell	PerCom	2003
Dynamically Programmable and Reconfigurable Middleware Services	Manuel Roman, Nayeem Islam	Middleware, LNCS	2004
Towards Context-Aware Adaptable Web Services	Markus Keidl, Alfons Kemper	WWW	2004
e-Ghosts: leaving virtual footprints in ubiquitous workspaces	Michael Vernik, Steven Johnson, Rudi Vernik	AUIC2004	2004
ContextPhone: A Prototyping Platform for Context-Aware Mobile Applications	Mika Raento, Antti Oulasvirta, Renaud Petit, and Hannu Toivonen	Pervasive Computing	2005
SAFE-RD (Secure, Adaptive, Fault Tolerant, and Efficient Resource Discovery) in a Pervasive Computing Environments	Moushumi Sharmin, Shameem Ahmed, and Sheikh I. Ahamed	ITCC	2005
Safe “composability” of middleware services	Nalini Venkatasubramanian	CACM	2002
Context-Aware Middleware for Mobile Multimedia Applications	Oleg Davidyuk *et al.*	Int. Conference on Mobile and ubiquitous multimedia	2004
Alia: Alliance-based Service Discovery for Ad-hoc Environments	Olga Ratsimor *et al.*	ACM Mobile Commerce Workshop	2002
Next Generation Context Aware Adaptive Services	Owen Conlan *et al.*	International Symposium on Information and Communication	2003

		Technologies	
A secure pervasive environment	Patrick G. McLean	AISW2003	2003
Facilitating User Interface Adaptation to Mobile Devices	Pertti Repo	Conference on Human-computer Interaction	2004
Middleware Support for Implementing Context-Aware Multimodal User Interfaces	Pertti Repo, Jukka Riekki	MUM	2004
A Toolkit to design adaptable user interfaces in ubiquitous computing environments	Philippe Debaty	PerCom	2004
Context-based Design of Mobile Application for Museums: A Survey of Existing Practices	Raptis, D., Tselios, N., and Avouris, N.	MobileHCI	2005
The Around Architecture for Dynamic Location-Based Services	Rui Jose *et al.*	Mobile Networks and Applications	2003
Transparent Shaping of Existing SOFTWARE TO Support Pervasive and Autonomic Computing	S. Masoud Sadjadi, Philip K. McKinley, Betty H.C. Cheng	DEAS	2005
An Architecture for the Integration of Physical and Informational Spaces	Scott M. Thayer, Peter Steenkiste	Personal Ubiquitous Computing	2003
Opportunities and issues relating to middleware technologies for context-aware service	Shigetoshi Sameshima *et al.*	SICE Annual Conference	2003
Context-sensitive middleware for real-time software in ubiquitous computing environments	Stephen S . Yau and Fariaz Karim		2004
An Energy-efficient object discovery protocol for context-sensitive middleware for ubiquitous computing	Stephen S. Yau, Fariaz Karim	IEEE Transactions on Parallel and Distributed Systems	2003
Self-Adaptation for Everyday Systems	Svein Hallsteinsen, Erlend Stav, Jacqueline Floch	WOSS	2004
Location History in a Low-cost context awareness environment	Teddy Mantoro, Chris Johnson	ACSW Frontiers	2003
System Software for Ubiquitous Computing	Tim Kindberg, Armando Fox	IEEE Pervasive Computing	2002
Composable Ad-hoc Mobile Services for Universal Interaction	Todd D. Hodes *et al.*	MOBICOA4	1997
Dynamic Configuration of Resource-aware services	Vahe Poladian *et al.*	ICSE	2004
Functionality Adaptation: A context-aware service code adaptation for pervasive computing environments	VivienWai-Man Kwan, Francis Chi-Moon Lau, Cho-Li Wang	International Conference on Web Intelligence	2003
A Web Services Composition Approach based on Software Agents and Context	Zakaria Maamar, Soraya Kouadri M., Hamdi Yahyaoui	SAC'04	2004

APÊNDICE B – *CHECKLIST* DE CARACTERIZAÇÃO

Este apêndice apresenta o checklist elaborado para caracterizar projetos de software de acordo com a sua aderência às características e fatores de ubiquidade.

B.1 – *Checklist* de Caracterização

Característica	Nível de Aderência à Característica	Grupo de Fatores	Nível de Aderência ao Grupo de Fator	Fatores	Status
Onipresença de Serviços	0%	Mobilidade	0%	Gerenciar seções do usuário	✔
				Lidar com a mobilidade do usuário tornando o serviço utilizado pelo usuário disponível onde ele esteja	✔
				Ao deslocar os serviços, estes devem continuar operando a partir do ponto em que seu processamento foi interrompido para a migração da funcionalidade	✔
				Suportar a mobilidade entre domínios e dentro de um mesmo domínio	✔
		Gerência de Serviços	0%	Cada dispositivo deve conter um container para alocar serviços	✔
				Cada dispositivo deve gerenciar os serviços alocados em seu container	✔
				Organizar os serviços segundo o contexto	✔
		Divulgação de Serviços	0%	Divulgar a existência do serviço para outros dispositivos/aplicações	✔
				Manter o registro de serviços divulgados em cache para aumentar o desempenho em uma nova divulgação de serviços	✔
		Fator Restritivo	0%	Cada serviço deve ser genérico o suficiente para continuar operando enquanto ocorrem alterações no ambiente	✔
Invisibilidade	0%	Gerência de Entidades	0%	As entidades que compõem o sistema devem possuir um identificador	✔
				O registro da existência de cada identificador deve ser mantido	✔
		Interface com Usuário	0%	Prover mecanismos diferenciados para que o usuário entre com informações (comando de voz, aproximação do usuário)	✔
				Prover mecanismos diferenciados para apresentar alguma saída ao usuário	✔
		Redução do Nível de Interatividade com o Usuário	0%	Pró atividade: baseado nas informações coletadas, tomar uma decisão sem que haja necessidade de interação com usuário	✔
				Minimizar necessidade de configuração de dispositivos por parte do usuário	✔
				Tornar a adaptação das aplicações em tempo de execução o menos intrusiva possível	✔
				Reduzir a interatividade com o usuário	✔

Sensibilidade ao Contexto	**0%**	**Captura de Informações**	0,00%	Permitir identificar a identidade, localização, efeito ou atividade de um usuário quando este estiver no contexto de funcionamento do sistema	✔
				Temporalidade: considerar a variável Tempo para a informação de contexto capturada	✔
				Considerar informações de contexto de sistema. Aquelas relacionadas aos dispositivos, infra-estrutra do ambiente (informações de hardware utilizado (CPU, memória, tamanho da tela, energia), largura de banda disponível e dispositivos acessíveis) e outras aplicações que estão relacionadas para compor o sistema	✔
				Considerar informações de contexto de infra-estrutura. Neste caso, as informações dizem respeito à validade das informações dos outros tipos de contexto considerados. A importância deste tipo de informação de contexto está na diversidade de mecanismos de captura de informações	✔
				Considerar informações de contexto físico. Neste caso, busca-se informações que dizem respeito à interação entre dispositivos e ambiente. Por exemplo, informações de mobilidade, localização, tempo, condições de iluminação e barulho, temperatura	✔
				Considerar informações sobre usuário: perfil de usuário (preferências, objetivo, calendário do usuário, informações pessoais)	✔
		Controle de Sensores	0,00%	Controlar os sensores (ativo/em espera)	✔
		Gerência de Informações de Contexto	0,00%	Contextualizar as informações obtidas	✔
				Armazenar as informações consideradas úteis	✔
				Consolidar as informações originadas de diferentes sensores	✔
				Tratar alto acoplamento: existe relacionamento entre informações contextuais em diferentes níveis de abstração. É preciso gerenciar estes relacionamentos	✔
				Integração: é preciso criar mecanismos que possibilitem a integração de informações contextuais originadas em diferentes dispositivos com formatos distintos de representação	✔
				Categorizar contexto com base em sua fonte de dados (informações de rede, dispositivo e interação do usuário)	✔
				Possuir mecanismos de representar informações de contexto considerando duas variáveis: assunto e estado. O assunto, por sua vez, é dividido em três outras variáveis: identidade, tempo e localização	✔
				Considerar semântica na organização e captura das informações de contexto	✔
		Interpretação da Informação	0,00%	Learning e Reasoning: a partir das informações contextuais capturadas e de seus relacionamentos, derivar outras informações contextuais	✔
				Transformação da informação: os dados capturados podem ser transformados para gerar uma informação para o software. Deve existir um rastro indicando como a transformação foi efetuada e de qual dado uma determinada informação foi gerada	✔
				Contextualizar e personalizar as informações capturadas de acordo com preferências do usuário	✔
				Considerar semântica na interpretação de informação de contexto	✔
				Associar dados do sistema com informações do contexto	✔
		Compartilhamento da Informação	0,00%	Compartilhar informações de contexto com usuários e outros dispositivos. Disponibilizar para o usuário as informações de contexto como um recurso a ser utilizado para, por exemplo, efetuar configurações no software	✔

Comportamento Adaptavel	**0%**	**Gerência de Adaptações**	0,00%	Tractability: manter informações sobre as adaptações efetuadas para prover ao usuário informações que permitam a ele saber por que determinada decisão foi tomada pelo sistema	✔
				O software deve agir pró-ativamente, ou seja, analisar as variáveis e a partir dela tomar decisões que antecipem uma determinada situação	✔
				Selecionar a alternativa correta de adaptação	✔
				Prever alterações no ambiente e se adaptar ou preparar-se para adaptações antes que as alterações ocorram no ambiente	✔
				Gerenciar adaptações considerando preferências do usuário e condições do ambiente	✔
				O gerente de adaptações deve decidir quando alterações no ambiente são altas o suficiente para provocar adaptações	✔
				Calcular a configuração ótima de uma aplicação e seus requisitos de qualidade dada as necessidades do usuário e informações do ambiente	✔
				• Se o número de instancias de serviços sendo executadas no momento é menor que o número máximo permitido • O status da execução e localização de cada serviço • O tempo requerido para uso do serviço em relação ao tempo disponível da próxima instancia do serviço	✔
				Gerenciar a disponibilidade de recursos para execução dos serviços	✔
				O gerente de adaptações ao decidir sobre adaptações a serem realizadas deve considerar o impacto que estas adaptações terão no sistema como um todo, ou seja, verificar se outras aplicações relacionadas não sofrerão um impacto relevante depois da adaptação.	✔
				Finalizar o uso de recursos, liberando-os para futuro uso	✔
				Decidir onde alocar os processamentos necessários	✔
		Adaptação	0,00%	Adaptar automaticamente as funcionalidades baseado no contexto atual e passado	✔
				Adaptar automaticamente a transferência de informações à largura de banda atual	✔
				Adaptação automática de seção de usuário	✔
				Adaptação automática de conteúdo com base no contexto	✔
				Selecionar parte do código a ser executado com base no dispositivo e contexto em que a aplicação se encontra	✔
				Interface gráfica adaptável	✔
				Efetuar adaptações na arquitetura do software para este lida com alterações no ambiente que o cerca	✔
				Minimizar alterações em resultados de operações providos aos usuários como conseqüência de alterações no ambiente	✔
				Adaptar a interface com usuário a diferentes dispositivos em tempo de projeto ou execução. A adaptação é feita com base em informações do dispositivo e preferências do usuário	✔
				Efetuar ajustes de conteúdo e formato nos dados gerados pelos serviços	✔
				Lidar com reconfiguração de hardware	✔
Captura de Experiência	**0%**	**Captura de Informação**	0,00%	Captura de informações sobre a interação do usuário	✔
				Captura automática de experiências	✔
		Interpretação de Informação	0,00%	Análise de padrões que ocorrem nas interações de usuário capturadas	✔
				Criar mecanismos de análise de relacionamentos entre experiências privadas e públicas	✔
		Gerência de Informações	0,00%	Armazenar informações capturadas	✔
				Possuir um mecanismo de representação de atividades desempenhadas pelo usuário	✔
				Prover uma representação das necessidades do usuário e suas preferências	✔

Descoberta de Serviços	**0%**	**Descoberta de Serviços**	0,00%	Procurar serviços que desempenhem a mesma funcionalidade de um outro já utilizado	✔
				Identificar serviços semelhantes projetados para dispositivos diferentes	✔
				Lidar com ambigüidade na definição de serviços	✔
				Lidar com a liberação de um serviço/componente que tem sido utilizado	✔
				Interagir com serviços de endereçamento de serviço	✔
				Ao descobrir serviços, apresentar como sugestões de uso para o usuário	✔
				Cooperar com outros dispositivos que estejam em busca de um mesmo serviço	✔
				Serviços devem ser descobertos baseados na localização do dispositivo cliente seguindo duas abordagens: baseada em distância entre o cliente e o servidor; baseada no escopo de atuação do serviço	✔
		Conectar-se a Serviços	0,00%	Conectar-se dinamicamente a serviços	✔
		Seleção de Serviços	0,00%	Possuir mecanismos baseados em critérios para selecionar um serviço dentre um conjunto de serviços semelhantes disponíveis	✔
				Selecionar conjunto de serviços para atenderem a necessidades do usuário.	✔
				Selecionar conjunto de serviços em resposta a alterações no ambiente	✔
Composição de Funcionalidades	**0%**	**Composição**	0,00%	Compor componentes menores para formar a aplicação	✔
				Continuar executando as atividades durante a composição	✔
				Gerenciar a composição de funcionalidades	✔
				Efetuar deploy das funcionalidades geradas (instalação, ativação, atualização, remoção).	✔
				Compor funcionalidades a partir de serviços disponibilizados	✔
				Compor funcionalidades específicas para uma plataforma	✔
				Permitir que funcionalidades sejam retiradas ou acrescentadas em tempo de execução (manutenção)	✔
				Composição é executada sem interação do usuário	✔
				Compor funcionalidades baseadas em necessidades do usuário e informações do contexto	✔
				Considerar a semântica no processo de composição de funcionalidades	✔
				Ao final do processo de composição, solicitar permissão de execução do novo serviço para o usuário	✔
		Gerência de Serviços	0,00%	Prover informações sobre os componentes utilizados em tempo de execução.	✔
				Permitir que o estados dos componentes sejam mantidos durante e após a "manutenção" do software	✔
				Monitorar a execução dos serviços criados para checar se eles estão desempenhando suas atividades de acordo com suas especificações	✔
				Gerenciar componentes remotos (criar, destruir, carregar, descarregar e transferir).	✔
		Integração de Serviços	0,00%	Lidar com incompatibilidade entre interfaces de serviços	✔
				Integrar dados trocados entre os serviços	✔
				Fazer uso de mediadores no mecanismo de integração. Os mediadores devem ser móveis e acopláveis a diferentes serviços. Assim, não fazem parte da estrutura de um serviço, mas sim, utilizados por eles	✔

Interoperabilidade Espontanea	**0%**	**Gerência de Informações**	0,00%	Manter histórico de relacionamentos estabelecidos com dispositivos	✔
				Manter mecanismos de seleção e limpeza de informações históricas possivelmente não mais relevantes	✔
		Interoperabilidade	0,00%	Interagir com aplicações existentes para uso de serviços	✔
				Componentes devem interagir de forma espontânea através da disponibilização de informações que descrevam o componente e funcionalidades	✔
				Possuir mecanismos para possibilitar a comunicação entre aplicações que possuam interfaces de comunicação heterogêneas	✔
				Possuir mecanismos para integração de dados	✔
				Cooperar com outros dispositivos	✔
				Acessar outros dispositivos desde que o acesso seja permitido pela política de segurança	✔
				Identificar características dos dispositivos descobertos (capacidade de processamento e armazenamento, por exemplo)	✔
				Conexão dinâmica: conexões com dispositivos imersos no ambiente podem ser perdidas temporariamente, é necessário possibilitar seu restabelecimento	✔
Heterogeneidade de Dispositivos	**0%**	**Busca por Dispositivos**	0,00%	Buscar dispositivo compatível com um determinado componente	✔
				Identificar dispositivos disponíveis no ambiente	✔
		Migração de Aplicação	0,00%	Lidar com migração de código entre dispositivos heterogêneos	✔
				Possibilitar a migração de uma aplicação inteira entre dispositivos ou parte de seu código	✔
				Aplicações são migradas em tempo de execução, não havendo interrupção do serviço	✔
				Controlar a migração de aplicações através do controle de versionamento e estado	✔
				Prover o gerenciamento de sessões quando o usuário troca de dispositivos	✔
				Manter o estado do conjunto de aplicações para estas poderem ser migradas sem perdas para o usuário	✔
		Compartilhamento de Recursos	0,00%	Permitir o compartilhamento de recursos	✔
Tolerancia a Falhas	**0%**	**Falhas de Software**	0,00%	Lidar com falha de rotinas	✔
				Criar mecanismos para armazenar informações de estado da aplicação para as falhas consideradas mais comuns de acontecer	✔
		Falhas de Sistema	0,00%	Lidar com falhas no ambiente (conexão perdida, por exemplo)	✔

APÊNDICE C – PROTOCOLO DE ATUALIZAÇÃO DO CORPO DE CONHECIMENTO

Este apêndice apresenta o protocolo de atualização do corpo de conhecimento em computação ubíqua.

C.1 – Protocolo de Atualização

Objetivo

Executar uma revisão sistemática cujo objetivo é atualizar o corpo de conhecimento definido em (SPÍNOLA *et al.*, 2008) considerando dois questionamentos:

1. Que características definem aplicações para computação ubíqua?
2. Quais os fatores funcionais associados a cada característica de ubiquidade?

Formulação da Pergunta
P0: Quais são as características básicas que definem sistemas ubíquos?

- **Intervenção:** características básicas utilizadas para definir sistemas ubíquos.
- **Efeito:** definição de um conjunto de características básicas que definem sistemas ubíquos.
- **Medida de desfecho:** quantidade de características encontradas.
- **População:** artigos de computação ubíqua.
- **Problema:** identificar as características básicas que definem sistemas ubíquos.
- **Aplicação:** projeto de desenvolvimento de sistemas ubíquos.

Além disso, buscamos repostas para uma **segunda pergunta**:

P1: Quais requisitos funcionais caracterizam as características de ubiquidade (onipresença de serviços, sensibilidade ao contexto, comportamento adaptável, captura de experiência, heterogeneidade de dispositivos)?

- **Intervenção:** requisitos funcionais presentes nas características de ubiquidade.
- **Efeito:** requisitos funcionais que caracterizam as características de ubiquidade.
- **Medida de desfecho:** quantidade de requisitos funcionais.
- **População:** artigos de computação ubíqua.

- **Problema:** encontrar requisitos funcionais que caracterizam as características de ubiquidade.
- **Aplicação:** fundamento para ponto de partida em pesquisas envolvendo computação ubíqua.

Seleção de Fontes

Será fundamentada na base eletrônica SCOPUS. O motivo de sua seleção se deve ao fato dela indexar as principais bases eletrônicas e ter sido utilizada na última execução deste protocolo. Serão considerados somente os artigos publicados a partir de 2005, sendo este o ano de realização das revisões sistemáticas que originaram o corpo de conhecimento base que será atualizado através da execução deste protocolo.

Palavras-chave

Ubiquitous computing, pervasive computing
Ubiquitous application, Ubiquitous system, ubiquitous software
Feature, requirement, characteristic (exclusivo para P1)
Functional requirement, functionality, feature
Computer everywhere, Service omnipresence
Context awareness, context aware, context sensitivity
Adaptability, Adaptable Behavior
Experience capture
Heterogeneity of Devices

Idioma dos Estudos
Inglês.

Identificação de Fontes

Métodos de busca de fontes:
As fontes serão acessadas via *web*. No contexto dessa revisão não será considerada a busca manual.

Fonte de Artigos:
SCOPUS

Critérios de Inclusão e Exclusão de Artigos
- Os artigos devem estar disponíveis na *web*;

- Os artigos devem estar descritos em inglês;
- Os artigos devem apresentar características que definem sistemas ubíquos. Assim, não serão considerados artigos que trazem somente características ligadas ao domínio da aplicação (exclusivo para P0).
- Os artigos devem contemplar requisitos funcionais pertinentes a cada uma das características de ubiquidade para a qual a string de busca foi formulada (exclusivo para P1).

Processo de Seleção dos Estudos Preliminares

Um pesquisador aplicará a estratégia de busca para a identificação de potenciais artigos. Os artigos identificados serão selecionados pelo mesmo pesquisador através da leitura e verificação dos critérios de inclusão e exclusão estabelecidos.

Avaliação da Qualidade dos Estudos Primários

Não temos definido um checklist para a avaliação da qualidade dos artigos. A abordagem para definição da qualidade está fundamentada na fonte para extração do material e na aplicação dos critérios de inclusão / exclusão dos estudos.

Estratégia de Extração de Informação

Para cada estudo selecionado após a execução do processo de seleção, o pesquisador irá extrair os seguintes dados:

- Título do artigo
- Autores
- Fonte
- Característica identificada (exclusivo P0)
- Descrição das características (exclusivo P0 – deve ser preenchido caso a característica seja nova)
- Requisitos funcionais (exclusivo P1)

Sumarização de Resultados

Os resultados serão tabulados. Serão realizadas análises para:

- definir se houveram novas características de ubiquidade a serem acrescentadas ao conjunto inicial.

- definir os novos requisitos funcionais para cada uma das características de ubiquidade.

Buscas Utilizadas

Para a **primeira pergunta (P0)**:

Busca P0: (ubiquitous computing <or pervasive computing>) <and> (feature <or> requirement <or> characteristic)

Periódico	Escopo da busca
SCOPUS	Texto completo.

Para a **segunda pergunta (P1)**:

Periódico	Escopo da busca
SCOPUS	Texto completo.

Esta pergunta foi quebrada em 6 questões específicas para cada uma das características de ubiquidade:

P1.1: (“ubiquitous computing” OR “pervasive computing”) AND (“functional requirement” OR “functionality” OR “feature” OR “characteristic”) AND (“service omnipresence” OR “computer everywhere”) AND PUBYEAR AFT 2005

P1.2: (“ubiquitous computing” OR “pervasive computing”) AND ("functional requirement" OR "functionality" OR "feature" OR "characteristic") AND ("context awareness" OR "context sensitivity") AND PUBYEAR AFT 2005

P1.3: (“ubiquitous computing” OR “pervasive computing”) AND ("functional requirement" OR "functionality" OR "feature" OR "characteristic") AND (“adaptability” OR “adaptable behavior”) AND PUBYEAR AFT 2005

P1.4: (“ubiquitous computing” OR “pervasive computing”) AND ("functional requirement" OR "functionality" OR "feature" OR "characteristic") AND (“experience capture”) AND PUBYEAR AFT 2005

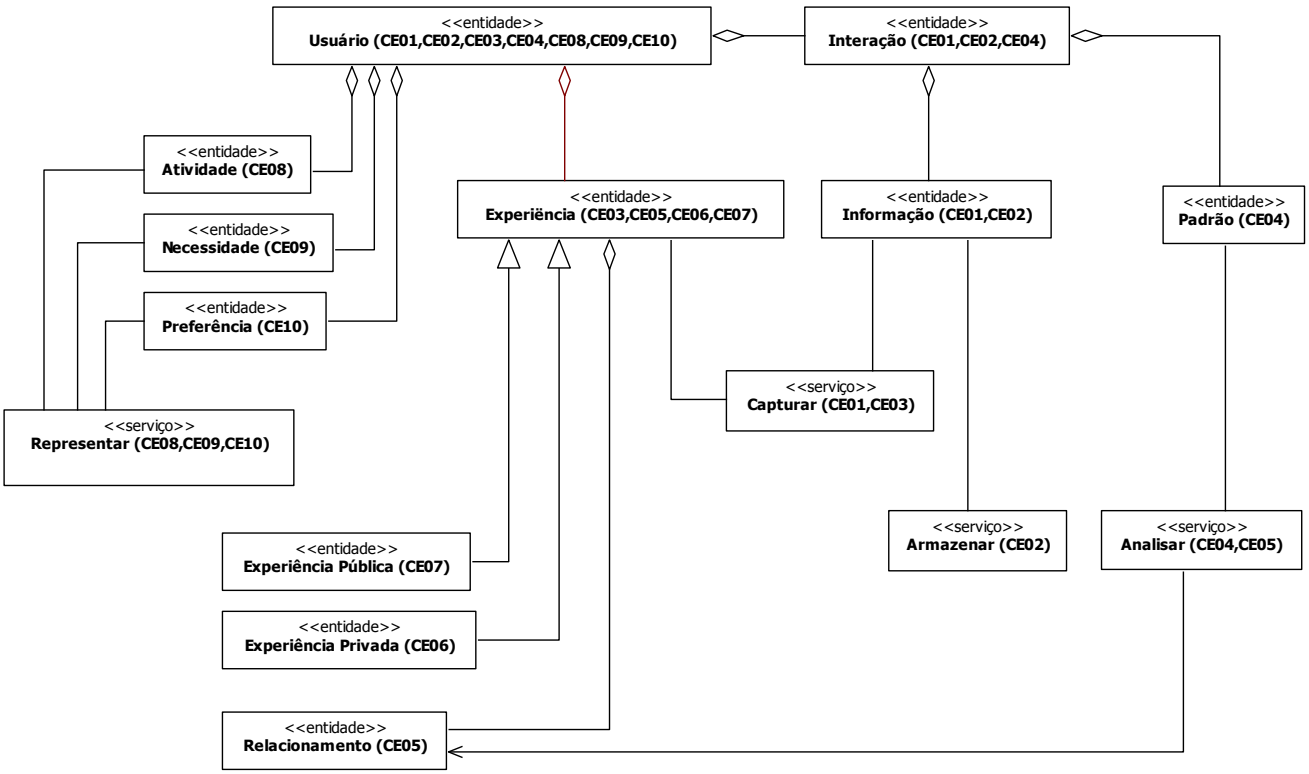
P1.5: (“ubiquitous computing” OR “pervasive computing”) AND ("functional requirement" OR "functionality" OR "feature" OR "characteristic") AND (“device heterogeneity”) AND PUBYEAR AFT 2005

P1.6: (“ubiquitous computing” OR “pervasive computing”) AND ("functional requirement" OR "functionality" OR "feature" OR "characteristic") AND (“spontaneous interoperability”) AND PUBYEAR AFT 2005

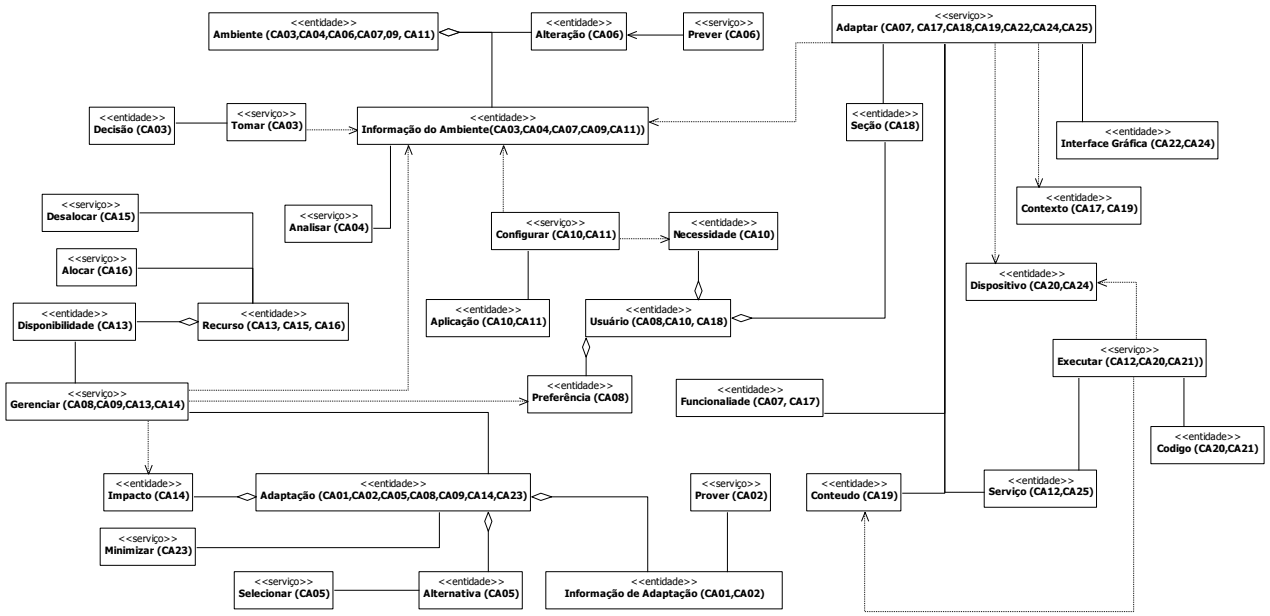
APÊNDICE D – MODELOS CONCEITUAIS

Este apêndice apresenta os modelos conceituais elaborados para cada uma das características funcionais de ubiquidade.

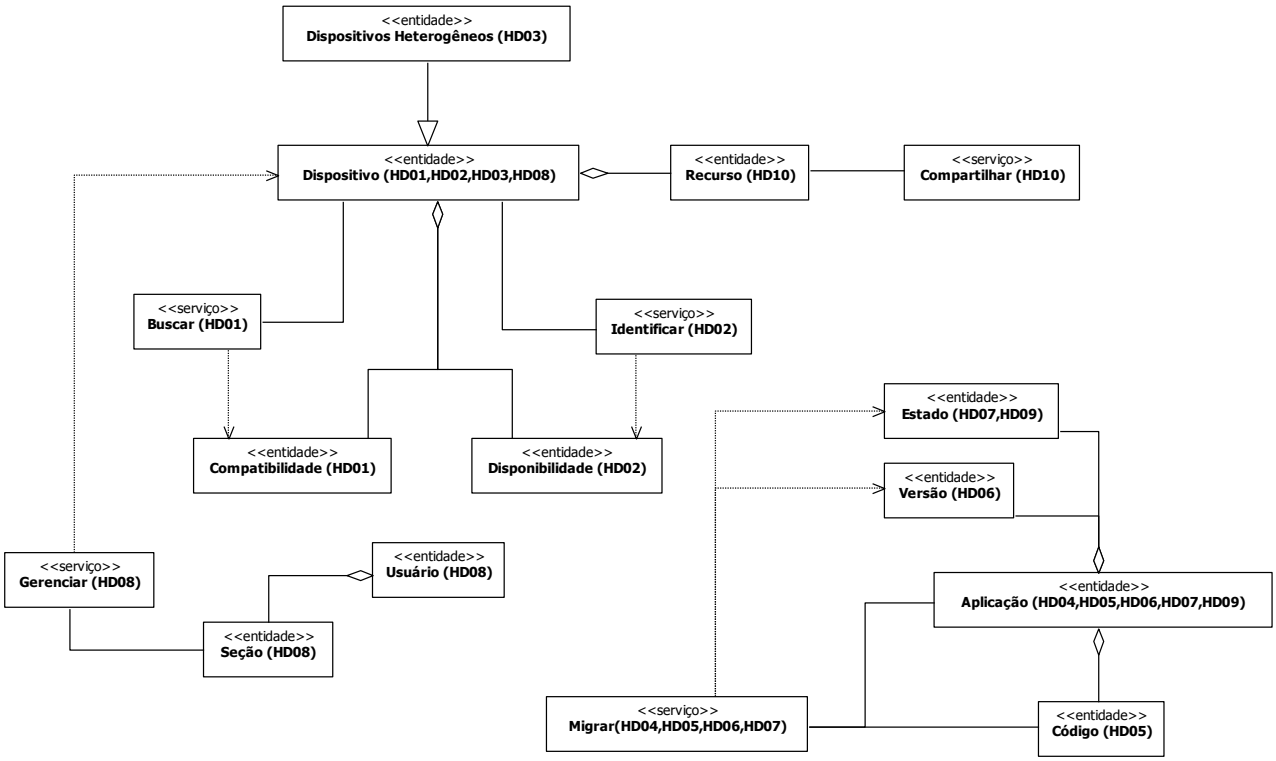
D.1 – Captura de Experiência



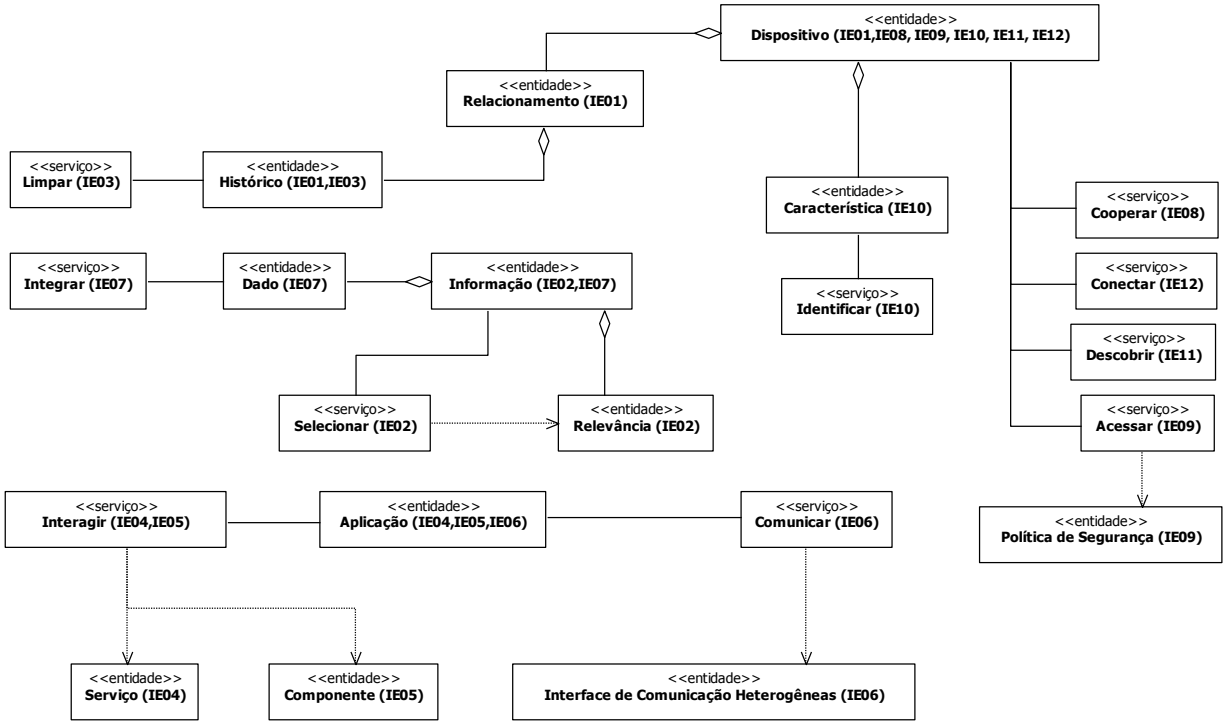
D.2 – Comportamento Adaptável



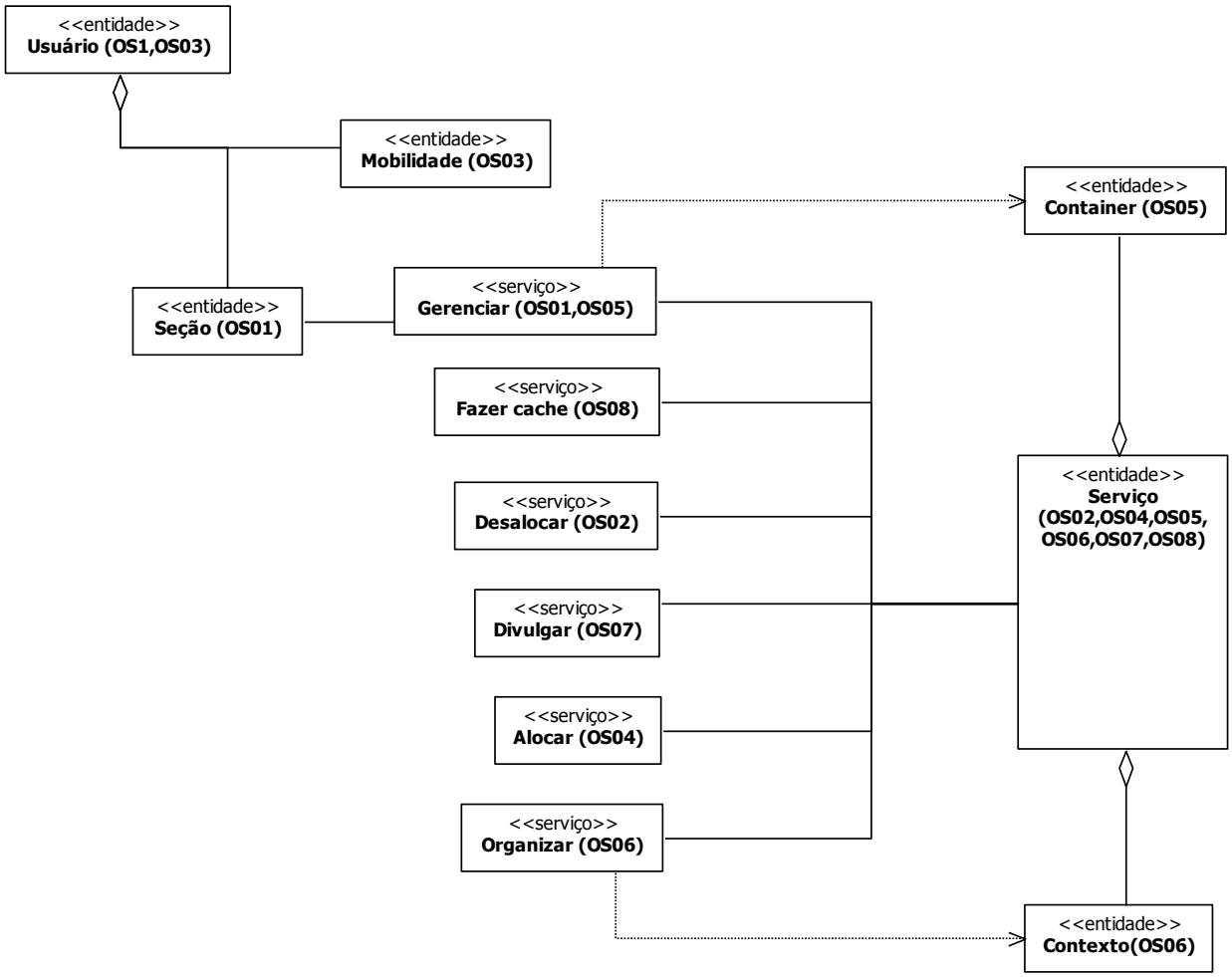
D.3 – Heterogeneidade de Dispositivos



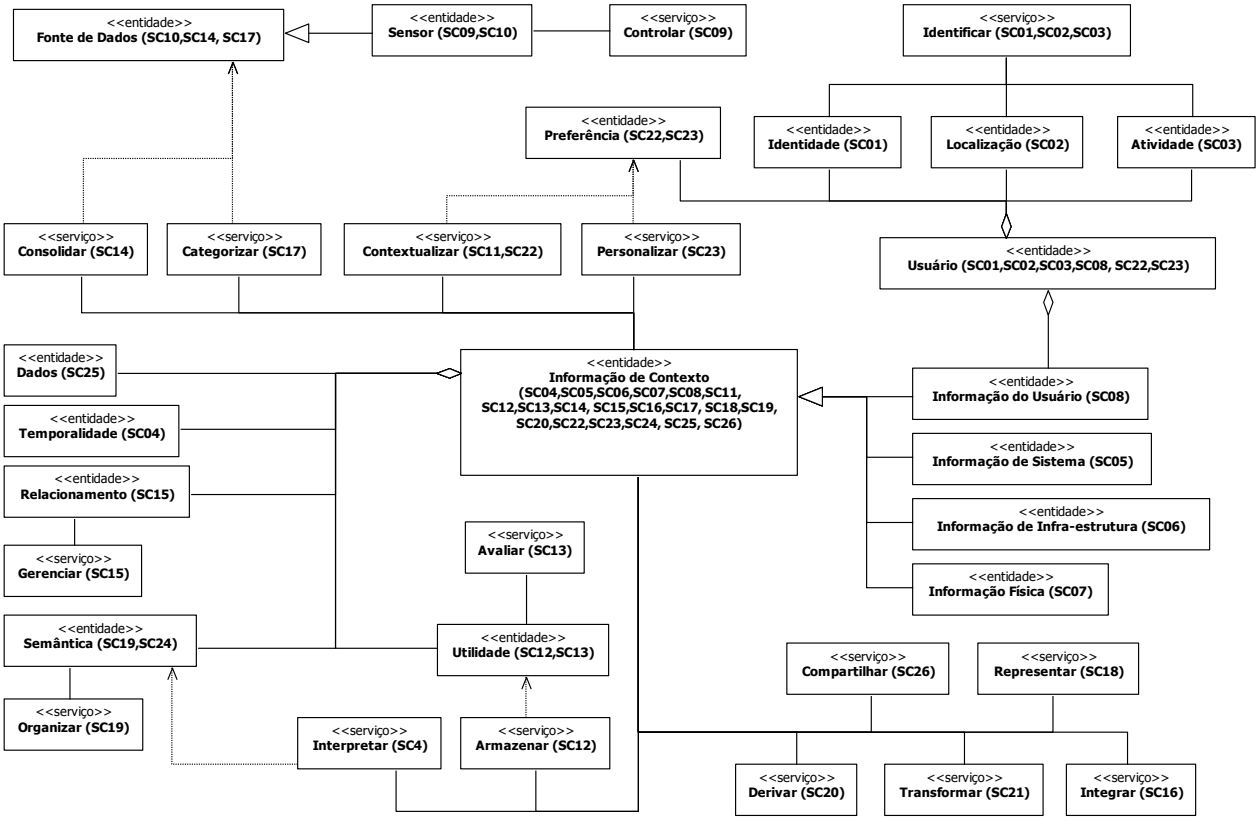
D.4 – Interoperabilidade Espontânea



D.5 – Onipresença de Serviços



D.6 – Sensibilidade ao Contexto



APÊNDICE E – QUESTIONÁRIO DE CARACTERIZAÇÃO

Este apêndice apresenta o questionário de caracterização de projetos de software ubíquo.

E.1 – Questionário de Caracterização – Etapa 1

Para este questionário, as seguintes abreviações serão consideradas: Onipresença de Serviços **(OS)**, Comportamento Adaptável **(CA)**, Captura de Experiência **(CE)**, Heterogeneidade de Dispositivos **(HD)**, Interoperabilidade Espontânea **(IE)**, Sensibilidade ao Contexto **(SC)**.

Caracterização do Projeto - Etapa 1			
1	O objetivo desta etapa é Identificar junto ao cliente quais características de ubiqüidade estarão presentes na aplicação. Para isto, o conjunto de questionamentos listados na seqüência junto com suas definições pode ser utilizado. A sugestão é que a definição de cada característica e seu cenário de exemplo sejam utilizados apenas pelo analista para um melhor entendimento sobre as características. Ao cliente devem ser efetuados apenas os questionamentos.		
Característica: Onipresença de Serviços			
Descrição:	Permitir a movimentação física do usuário, dando a ele a percepção de estar levando consigo os serviços computacionais. Além disso, refere-se à capacidade de estar presente nos objetos de uso do dia-a-dia, descaracterizando, do ponto de vista do usuário, a "utilização" de um computador e acentuando a percepção de objetos ou dispositivos que provêem serviços ou algum tipo de "inteligência".		
Exemplo:	Ao passar em frente a um cartaz de um filme, o indivíduo recebe no celular maiores informações sobre o filme em questão como: trilha sonora, diretor, duração, elenco e o resumo.		
Questionário:		**Glossário**	
OS1:	O software deve ser capaz de permitir que suas funcionalidades estejam acessíveis ao usuário em diferentes localidades (casa, rua, trabalho) ou diferentes dispositivos (celular, PC, notebook)?		
OS2:	O software, ou parte dele, será desenvolvido para algum dispositivo móvel (celular, smartphone, palm)?		
OS3:	O software deve ser capaz de permitir seu uso através de comandos disparados por voz, gesto, movimento ou algo diferente de uma interação através de teclado?		
OS4:	O software deve ser capaz de adaptar seu comportamento ao ambiente em que se encontra funcionando para disponibilizar ao usuário uma melhor experiência de uso? Por exemplo, ao entrar em uma reunião alterar o modo de chamada para silencioso.		
Característica: Sensibilidade ao Contexto			
Descrição:	Capacidade de coletar informações sobre o ambiente onde o software está sendo utilizado.		
Exemplo:	Um sistema para controle de temperatura de um frigorífico deve monitorar constantemente a temperatura para manter o ambiente no estado ideal para manutenção dos produtos.		
Questionário:			
SC1:	O software deve ser capaz de capturar informações sobre alguma entidade externa ao sistema?	GSC1; GSC2;	
SC2:	O software deve ser capaz de capturar informações do ambiente (por exemplo: temperatura, taxa de transferência) para seu funcionamento?		

Característica:	**Comportamento Adaptável**		
Descrição:	Capacidade de adaptar os serviços/funcionalidades de acordo com as restrições do ambiente.		
Exemplo:	Imaginemos um software para gerência de vídeo conferência. Ao identificar a redução de largura de banda a ponto de prejudicar a transmissão do áudio e vídeo na qualidade atual, o software deve reduzir a qualidade do áudio e do vídeo de forma a permitir que a comunicação entre os participantes continue fluindo normalmente sem interrupções.		
Questionário:			
CA1:	O software deve ser capaz de prover alguma modificação (por exemplo: reduzir qualidade do vídeo, reduzir a temperatura do ambiente interno com base na temperatura externa) em seu funcionamento com base nas informações capturadas do ambiente?		
CA2:	O software deve ser capaz de controlar variáveis externas ao sistema e executar ações com base nos valores destas variáveis (por exemplo: temperatura externa > 30º -> diminuir temperatura interna para 24º)?		
Característica:	**Captura de Experiência**		
Descrição:	Capacidade de capturar e registrar experiências (ações do usuário, preferências do usuário) para uso posterior.		
Exemplo:	Imaginemos um software para gerenciamento da casa. Ele pode identificar comportamentos comuns do morador, por exemplo: o morador ao chegar liga a lâmpada da sala, esquenta a água para o café, liga a banheira e define a temperatura da água para 28º C. Ao perceber que este comportamento se repete (uma experiência), o software poderá gerenciar estas atividades quando o morador chegar em casa sem a necessidade deste desempenhar diretamente estas atividades.		
Questionário:			
CE1:	O software deve ser capaz de manter as preferências do usuário e utilizá-lo como base para ajuste de suas funcionalidades?	GSC19;	
CE2:	O software deve ser capaz de capturar informações do usuário (por exemplo: casas inteligentes) para adequar seu funcionamento às preferências do usuário?		
Característica:	**Heterogeneidade de Dispositivos**		
Descrição:	Capacidade de prover mobilidade da aplicação através de dispositivos heterogêneos.		
Exemplo:	Imagine um software para monitoramento do mercado de ações. Ele possui suas funcionalidades totais para acesso via desktop. Entretanto, no meio de suas atividades você tem que se deslocar para outra unidade da organização tendo disponível neste meio tempo um PDA. Nesta situação, o software deveria migrar parte de suas funcionalidades para serem acessadas pelo PDA e você continuar tendo acesso às informações necessárias.		
Questionário:			
HD1:	O software deve ser capaz de funcionar em dispositivos diferentes (por exemplo: palm, smartphone, PC e notebook)?		
HD2:	O software deve ser capaz de identificar outros dispositivos no ambiente compatíveis de forma a possibilitar a migração de serviços?	GHD1	
Característica:	**Interoperabilidade Espontânea**		
Descrição:	Capacidade de interagir com outros dispositivos durante a sua operação conforme a sua movimentação.		
Exemplo:	Imagine que você esteja se deslocando e o software, executando no PDA, está desempenhando uma tarefa intensiva em processamento. Durante o percurso, o software poderá interagir com outros dispositivos para alocação temporária de processamento.		
Questionário:			
IE1:	O software deve ser capaz de compartilhar a execução de determinadas funcionalidades com outros sistemas?		
IE2:	O software deve ser capaz de identificar outros dispositivos no ambiente compatíveis de forma a possibilitar a execução de serviços?		
1	No final desta etapa, tem-se o conjunto de características de ubiquidade para o projeto definido. É necessário agora identificar como cada característica que estará contemplada no projeto irá se comportar. **Para isso, para cada característica que obteve pelo menos um "Sim" como resposta neste questionário, responda o checklist correspondente.**		

E.2 – Questionário de Caracterização – Etapa 2

Caracterização do Projeto - Etapa 2 - OS	
2	O objetivo desta etapa é identificar junto ao cliente como a característica onipresença de serviços estará presente no projeto. Para isto, identifique junto ao cliente se os itens de 1 a 6 estarão ou não presentes no projeto.

Característica:	**Onipresença de Serviços**		
Questionário:		**Glossário**	
OS1:	O software deve gerenciar a sessão do usuário?	GOS1; GOS2	
OS2:	Caso haja necessidade de interromper seus serviços por algum motivo, o software deve ser capaz de retornar ao ponto onde deixou de executar ao ser interrompido?		
OS3:	O software deve ser desenvolvido considerando algum dispositivo que permita que o usuário tenha acesso ao sistema em qualquer lugar (por exemplo: celular, smartphone, palm)?		
OS4:	O software deve ser capaz de gerenciar os serviços por ele fornecidos?	GOS3; GOS4	
OS5:	O software deve ser capaz de disponibilizar para uso apenas os serviços apropriados ao cenário em que ele se encontra?	GOS5	
OS6:	O software deve ser capaz de divulgar os serviços por ele fornecidos?	GOS6	

2	Etapa de caracterização finalizada para a característica onipresença de serviços.

Caracterização do Projeto - Etapa 2 - SC

2

O objetivo desta etapa é identificar junto ao cliente como a característica sensibilidade ao contexto estará presente no projeto.

Para isto, identifique junto ao cliente se os itens de 1 a 21 estarão ou não presentes no projeto.

Característica: Sensibilidade ao Contexto			
Questionário:		**Glossário**	
SC1	O software deve ser capaz de identificar o usuário que o está manipulando?	GSC3;	
SC2	O software deve ser capaz de identificar o local onde está sendo utilizado?	GSC4;	
SC3	O software deve ser capaz de identificar a atividade que está sendo realizada pelo usuário?	GSC5	
SC4	O software deve ser capaz de organizar as informações coletadas considerando o momento (horário) em que elas foram coletadas?	GSC6;	
SC5	O software deve ser capaz de capturar informações de contexto de sistema (uso de memória, cpu, tamanho de tela, energia, largura de banda disponível)?	GSC7; GSC8;	
SC6	O software deve ser capaz de capturar informações de contexto de infra-estrutura (estado de funcionamento dos sensores)?	GSC9;	
SC7	O software deve ser capaz de capturar informações de contexto físico (temperatura, condições de iluminação, ruído)?	GSC10;	
SC8	O software deve ser capaz de capturar informações de contexto de usuário (preferências, ação)?	GSC11;	
SC9	O software deve ser capaz de fazer uso de sensores para capturar as informações de contexto?	GSC7; GSC12;	
SC10	O software deve ser capaz de manter as informações de contexto capturadas?		
SC11	O software deve ser capaz de filtrar as informações de contexto capturadas para armazenar apenas aquelas consideradas úteis?	GSC13;	
SC12	O software deve ser capaz de organizar as informações capturadas considerando o sensor que as originou?	GSC14;	
SC13	O software deve ser capaz de manter as relações existentes entre as diferentes informações de contexto?	GSC15;	
SC14	O software deve ser capaz de integrar as informações capturadas mesmo que elas tenham sido geradas em formatos diferentes?	GSC16;	
SC15	O software deve ser capaz de considerar as informações identidade, tempo e localização ao armazenar cada informação de contexto?		
SC16	O software deve ser capaz de considerar a semântica da informação de contexto ao armazená-la?	GSC17;	
SC17	O software deve ser capaz de derivar novas informações a partir das informações capturadas do contexto?	GSC18;	
SC18	O software deve ser capaz de manter um rastro das transformações efetuadas nas informações?		
SC19	O software deve ser capaz de organizar as informações de contexto considerando as preferências do usuário?	GSC19;	
SC20	O software deve ser capaz de considerar a semântica ao interpretar as informações?		
SC21	O software deve ser capaz de compartilhar as informações de contexto capturadas com outros sistemas?	GSC20;	

2

Etapa de caracterização finalizada para a característica sensibilidade ao contexto.

Caracterização do Projeto - Etapa 2 - CA

2

O objetivo desta etapa é identificar junto ao cliente como a característica comportamento adaptável estará presente no projeto.

Para isto, identifique junto ao cliente se os itens de 1 a 14 estarão ou não presentes no projeto.

Característica:	**Comportamento Adaptável**		
Questionário:		**Glossário**	
CA1	O software deve ser capaz de armazenar informações que indicam qual adaptação foi efetuada?	GCA1;	
CA2	O software deve ser capaz de efetuar adaptações em seu funcionamento?	GCA1;	
CA3	O software deve ser capaz de selecionar a alternativa de adaptação mais apropriada caso haja mais de uma possibilidade?	GCA2;	
CA4	O software deve ser capaz de adaptar as funcionalidades de acordo com alterações no ambiente?	GCA1;	
CA5	O software deve ser capaz de adaptar as funcionalidades de acordo com as preferências do usuário?	GCA1; GSC19;	
CA6	O software deve ser capaz de identificar se o serviço está disponível para uso?	GOS7;	
CA7	O software deve ser capaz de gerenciar a disponibilidade de recursos de forma que seja possível considerá-los na análise sobre a realização de uma adaptação?	GCA3;	
CA8	O software deve ser capaz de analisar impactos provenientes das adaptações a serem realizadas?	GCA4;	
CA9	O software deve ser capaz de adaptar as funcionalidades de acordo com o cenário em que está inserido?	GOS5;	
CA10	O software deve ser capaz de adaptar o conteúdo de acordo com o cenário em que está inserido?	GCA5; GCA6;	
CA11	O software deve ser capaz de adaptar o código da funcionalidade de acordo com o cenário em que está inserido?	GCA7;	
CA12	O software deve ser capaz de adaptar a interface com o usuário de acordo com o cenário em que está inserido?		
CA13	O software deve ser capaz de adaptar a interface com o usuário de acordo com o dispositivo?		
CA14	O software deve ser capaz de adaptar a interface com o usuário de acordo com as preferências do usuário?	GSC19;	

2

Etapa de caracterização finalizada para a característica comportamento adaptável.

Caracterização do Projeto - Etapa 2 - CE

2

O objetivo desta etapa é identificar junto ao cliente como a característica captura de experiência estará presente no projeto.

Para isto, identifique junto ao cliente se os itens de 1 a 5 estarão ou não presentes no projeto.

Característica: Captura de Experiência			
Questionário:		**Glossário**	
CE1	O software deve ser capaz de capturar informações sobre a interação do usuário com o sistema ao longo do tempo?	GCE1;	
CE2	O software deve ser capaz de analisar as informações sobre a interação do usuário com o sistema para identificar padrões de uso?	GCE2;	
CE3	O software deve ser capaz de armazenar as informações sobre a interação do usuário capturadas?		
CE4	O software deve possuir algum mecanismo que possibilite representar as atividades desempenhadas pelo usuário? Ou seja, deve conseguir atribuir semântica à interação?		
CE5	O software deve ser capaz de, a partir das informações capturadas, identificar as preferências de uso do sistema?	GCE1; GCE2;	

2

Etapa de caracterização finalizada para a característica captura de experiência.

Caracterização do Projeto - Etapa 2 - HD

2

O objetivo desta etapa é identificar junto ao cliente como a característica heterogeneidade de dispositivo estará presente no projeto.

Para isto, identifique junto ao cliente se os itens de 1 a 7 estarão ou não presentes no projeto.

Característica: Heterogeneidade de Dispositivo			
Questionário:		**Glossário**	
HD1	O software deve ser capaz de identificar dispositivos compatíveis para migração de serviços?	GHD1; GHD2; GHD3;	
HD2	O software deve ser capaz de identificar dentre os dispositivos compatíveis identificados aqueles que estão disponíveis para uso?	GHD4;	
HD3	O software deve estar preparado para executar parte de suas funcionalidades em outros dispositivos?		
HD4	O software deve ser capaz de gerenciar a sessão do usuário considerando que suas funcionalidades estão sendo executadas em dispositivos diferentes?	GOS2; GHD5;	
HD5	O software deve ser capaz de gerenciar o estado da aplicação considerando que suas funcionalidades estão sendo executadas em diferentes dispositivos?	GHD6; GHD7;	
HD6	O software deve ser capaz de compartilhar os recursos de hardware que está fazendo uso?	GHD8;	
HD7	O software de ser capaz de migrar suas funcionalidades/serviços para serem executadas em outros dispositivos?	GHD1;	

2

Etapa de caracterização finalizada para a característica heterogeneidade de dispositivo.

Caracterização do Projeto - Etapa 2 - IE

2	O objetivo desta etapa é identificar junto ao cliente como a característica interoperabilidade espontânea estará presente no projeto. Para isto, identifique junto ao cliente se os itens de 1 a 4 estarão ou não presentes no projeto.

Característica:	**Interoperabilidade Espontânea**		
Questionário:		**Glossário**	
IE1	O software deve ser capaz de manter um histórico de sistemas com os quais interagiu para solicitar a execução de suas funcionalidades?		
IE2	O software deve ser capaz de interagir com outros sistemas para executar parte de suas funcionalidades?		
IE3	O software deve ser capaz de integrar informações heterogêneas de diferentes aplicações?		
IE4	O software deve ser capaz de identificar dentre os dispositivos compatíveis identificados aqueles que estão disponíveis para uso?		

2	Etapa de caracterização finalizada para a característica interoperabilidade espontânea.

APÊNDICE F – GUIA GERAL DE DEFINIÇÃO DE REQUISITOS DE UBIQUIDADE

Este apêndice apresenta o Guia Geral de Definição de Requisitos de Ubiquidade.

F.1 – Característica Captura de Experiência

Fatores	Tipo	Árvore de Elementos	Pergunta
CE01,CE02, CE03, CE04,CE08, CE09, CE10	DIR	Usuário	Quais os usuários relevantes para o sistema?
CE01,CE02, CE04	AGR	-- Interação	Como a interação do usuário é considerada?
CE01,CE02	AGR	-- -- Informação	Como a informação da interação do usuário é considerada?
CE01,CE03	ASS	-- -- --Capturar	Como capturar informações da interação do usuário?
CE02	ASS	-- -- -- Armazenar	Como armazenar informações da interação do usuário?
CE04	AGR	-- -- Padrão	Como o padrão de interação do usuário é considerado?
CE04,CE05	ASS	-- -- -- Analisar	Como analisar o padrão de interação do usuário?
CE08	AGR	-- Atividade	Como as atividades do usuário são consideradas?
CE08,CE09, CE10	ASS	-- -- Representar	Como representar as atividades do usuário?
CE09	AGR	-- Necessidade	Como a necessidade do usuário é considerada?
CE08,CE09, CE10	ASS	-- -- Representar	Como representar as necessidades do usuário?
CE10	AGR	-- Preferência	Como a preferência do usuário é considerada?
CE08,CE09, CE10	ASS	-- -- Representar	Como representar as preferências do usuário?
CE03,CE05, CE06, CE07	AGR	-- Experiência	Como a experiência do usuário é considerada?
CE01,CE03	ASS	-- -- Capturar	Como capturar a experiência do usuário?
CE05	AGR	-- -- Relacionamento	Como o relacionamento de experiências do usuário é considerado?
CE04,CE05	ASS	-- -- -- Analisar	Como analisar os relacionamentos de experiências do usuário?
CE06	GEN	-- Experiência Privada	Quais experiências do usuário são consideradas experiências privadas?
CE07	GEN	-- Experiência Pública	Quais experiências do usuário são consideradas experiências públicas?

F.2 – Característica Comportamento Adaptável

Fatores	Tipo	Árvore de Elementos	Pergunta
CA07,CA17	DIR	Funcionalidade	Quais as funcionalidades relevantes para o sistema?
CA07,CA17, CA18, CA19,CA22, CA24, CA25	ASS	-- Adaptar	Como adaptar as funcionalidades?
CA03,CA04, CA07, CA09, CA11	DEP	-- -- Informação do Ambiente	Como a informação do ambiente influencia na adaptação das funcionalidades?
CA17,CA19	DEP	-- -- Contexto	Como o contexto influencia na adaptação das funcionalidades?
CA20,CA24	DEP	-- -- Dispositivo	Como o dispositivo influencia na adaptação das funcionalidades?
CA03	DIR	Decisão	Quais as decisões relevantes para o sistema?
CA03	ASS	-- Tomar	Como tomar as decisões?
CA03,CA04, CA07, CA09, CA11	DEP	-- -- Informação do Ambiente	Como a informação do ambiente influencia na tomada de decisão?
CA10,CA11	DIR	Aplicação	Quais as aplicações relevantes para o sistema?
CA10,CA11	ASS	-- Configurar	Como configurar aplicações?
CA10	DEP	-- -- Necessidade do Usuário	Como a necessidade do usuário influencia na configuração das aplicações?
CA03,CA04, CA07, CA09, CA11	DEP	-- -- Informação do Ambiente	Como a informação do ambiente influencia na configuração das aplicações?
CA19	DIR	Conteúdo	Quais os conteúdos relevantes para o sistema?
CA07,CA17, CA18, CA19,CA22, CA24, CA25	ASS	-- Adaptar	Como adaptar o conteúdo?
CA03,CA04, CA07, CA09, CA11	DEP	-- -- Informação do Ambiente	Como a informação do ambiente influencia na adaptação de conteúdo?
CA17,CA19	DEP	-- -- Contexto	Como o contexto influencia na adaptação do conteúdo?
CA20,CA24	DEP	-- -- Dispositivo	Como o dispositivo influencia na adaptação do conteúdo?
CA20,CA21	DIR	código-fonte	Quais os códigos fontes relevantes para o sistema?
CA12,CA20, CA21	ASS	-- Executar	Como executar códigos-fontes?
CA20,CA24	DEP	-- -- Dispositivo	Como o dispositivo influencia na execução do código-fonte?
CA19	DEP	-- -- Conteúdo	Como o conteúdo influencia na execução do código fonte?
CA03,CA04, CA07, CA09, CA11	DIR	Informação do Ambiente	Quais as informações do ambiente são relevantes para o sistema?
CA04	ASS	-- Analisar	Como analisar as informações do ambiente?

Fatores	Tipo	Árvore de Elementos	Pergunta
CA22,CA24	DIR	Interface Gráfica	Quais as interfaces gráficas relevantes para o sistema?
CA07,CA17, CA18, CA19,CA22, CA24,CA25	ASS	-- Adaptar	Como adaptar a interface gráfica?
CA03,CA04, CA07, CA09, CA11	DEP	-- -- Informação do Ambiente	Como as informações do ambiente influenciam na adaptação da interface gráfica?
CA17,CA19	DEP	-- -- Contexto	Como o contexto influencia na adaptação da interface gráfica?
CA20,CA24	DEP	-- -- Dispositivo	Como o dispositivo influencia na adaptação da interface gráfica?
CA03,CA04, CA06, CA07,CA09, CA11	DIR	Ambiente	Quais os ambientes relevantes para o sistema?
CA03,CA04, CA07, CA09, CA11	AGR	-- Informação do Ambiente	Como a informação do ambiente é considerada?
CA04	ASS	-- -- Analisar	Como analisar as informações do ambiente?
CA06	AGR	-- Alteração	Como a alteração no ambiente é considerada?
CA06	ASS	-- -- Prever	Como prever alterações no ambiente?
CA12,CA25	DIR	Serviço	Quais os serviços relevantes para o sistema?
CA12,CA20, CA21	ASS	-- Executar	Como executar serviços?
CA20,CA24	DEP	-- -- Dispositivo	Como o dispositivo influencia a execução de serviços?
CA19	DEP	-- -- Conteúdo	Como o conteúdo influencia a execução do serviços?
CA07,CA17, CA18, CA19,CA22, CA24, CA25	ASS	-- Adaptar	Como adaptar serviços?
CA03,CA04, CA07, CA09, CA11	DEP	-- -- Informação do Ambiente	Como as informações do ambiente influenciam na adaptação do serviço?
CA17,CA19	DEP	-- -- Contexto	Como o contexto influencia a adaptação do serviço?
CA20,CA24	DEP	-- -- Dispositivo	Como o dispositivo influencia a adaptação do serviço?
CA08,CA10, CA18	DIR	Usuário	Quais os usuários relevantes para o sistema?
CA08	AGR	-- Preferência do Usuário	Como a preferência do usuário é considerada?
CA10	AGR	-- Necessidade do Usuário	Como a necessidade do usuário é considerada?
CA18	AGR	-- Seção	Como a seção do usuário é considerada?
CA07,CA17, CA18, CA19,CA22, CA24, CA25	ASS	-- -- Adaptar	Como adaptar a seção do usuário?

Fatores	Tipo	Árvore de Elementos	Pergunta
CA03,CA04, CA07, CA09, CA11	DEP	-- -- -- Informação do Ambiente	Como a informação do ambiente influencia na adaptação da seção do usuário?
CA17,CA19	DEP	-- -- -- Contexto	Como o contexto influencia na adaptação da seção do usuário?
CA20,CA24	DEP	-- -- -- Dispositivo	Como o dispositivo influencia na adaptação da seção do usuário?
CA01,CA02, CA05, CA08,CA09, CA14, CA23	DIR	Adaptação	Quais as adaptações relevantes para o sistema?
CA01,CA02	AGR	-- Informação de Adaptação	Como a informação da adaptação é considerada?
CA02	ASS	-- -- Prover	Como prover informações sobre adaptações?
CA05	AGR	-- Alternativa	Como a alternativa de adaptação é considerada?
CA05	ASS	-- -- Selecionar	Como selecionar a alternativa de adaptação?
CA08,CA09, CA13, CA14	ASS	-- Gerenciar	Como gerenciar a adaptação?
CA08	DEP	-- -- Preferência do Usuário	Como as preferências do usuário influenciam a gerência das adaptações?
CA03,CA04, CA07, CA09, CA11	DEP	-- -- Informação do Ambiente	Como as informações do ambiente influenciam a gerência das adaptações?
CA14	DEP	-- -- Impacto da Adaptação	Como o impacto da adaptação influencia a gerência das adaptações?
CA14	AGR	-- Impacto	Como o impacto da adaptação é considerado?
CA23	ASS	-- Minimizar	Como minimizar as adaptações?
CA13,CA15, CA16	DIR	Recurso	Quais os recursos relevantes para o sistema?
CA13	AGR	-- Disponibilidade	Como a disponibilidade do recurso é considerada?
CA08,CA09, CA13, CA14	ASS	-- -- Gerenciar	Como gerenciar a disponibilidade dos recursos?
CA08	DEP	-- -- -- Preferência do Usuário	Como as preferências do usuário influenciam a gerência da disponibilidade de recursos?
CA03,CA04, CA07, CA09,CA11	DEP	-- -- -- Informação do Ambiente	Como as informações do ambiente influenciam a gerência da disponibilidade de recursos?
CA14	DEP	-- -- -- Impacto da Adaptação	Como o impacto da adaptação influencia a gerência da disponibilidade de recursos?
CA15	ASS	-- Desalocar	Como desalocar recursos?
CA16	ASS	-- Alocar	Como alocar recursos?

F.3 – Característica Heterogeneidade de Dispositivos

Fatores	Tipo	Árvore de Elementos	Pergunta
HD08	DIR	Usuário	Quais os usuários relevantes para o sistema?
HD08	AGR	-- Seção	Como a seção dos usuários é considerada?

Fatores	Tipo	Árvore de Elementos	Pergunta
HD08	ASS	-- -- Gerenciar	Como gerenciar a seção dos usuários?
HD01,HD02, HD03, HD08	DEP	-- -- -- Dispositivo	Como o dispositivo influencia na gerencia da seção dos usuários?
HD01,HD02, HD03, HD08	DIR	Dispositivo	Quais os dispositivos relevantes para o sistema?
HD01	AGR	-- Compatibilidade	Como a compatibilidade dos dispositivos é considerada?
HD01	ASS	-- Buscar	Como buscar dispositivos?
HD01	DEP	-- -- Compatibilidade do Dispositivo	Como a compatibilidade do dispositivo é considerada na busca por dispositivos?
HD02	ASS	-- Identificar	Como identificar dispositivos?
HD02	DEP	-- -- Disponibilidade do Dispositivo	Como a disponibilidade do dispositivo é considerada na identificação de dispositivos?
HD02	AGR	-- Disponibilidade	Como a disponibilidade do dispositivo é considerada?
HD03	GEN	-- Dispositivos Heterogêneos	Quais os dispositivos são considerados dispositivos heterogêneos?
HD10	AGR	-- Recurso	Como os recursos dos dispositivos são considerados?
HD10	ASS	-- -- Compartilhar	Como compartilhar recursos de dispositivos?
HD04,HD05, HD06, HD07, HD09	DIR	Aplicação	Quais as aplicações relevantes para o sistema?
HD04,HD05, HD06, HD07	ASS	-- Migrar	Como migrar aplicações?
HD06	DEP	-- -- Versão da Aplicação	Como a versão da aplicação é considerada na migração das aplicações?
HD07,HD09	DEP	-- -- Estado da Aplicação	Como o estado da aplicação é considerado na migração das aplicações?
HD05	AGR	-- Código	Como o código da aplicação é considerado?
HD04,HD05, HD06, HD07	ASS	-- -- Migrar	Como migrar código de aplicações?
HD06	DEP	-- -- -- Versão da Aplicação	Como a versão da aplicação é considerada na migração de código das aplicações?
HD07,HD09	DEP	-- -- -- Estado da Aplicação	Como o estado da aplicação é considerado na migração de código das aplicações?
HD06	AGR	-- Versão	Como a versão da aplicação é considerada?
HD07,HD09	AGR	-- Estado	Como o estado da aplicação é considerado?

F.4 – Característica Interoperabilidade Espontânea

Fatores	Tipo	Árvore de Elementos	Pergunta
IE02,IE07	DIR	Informação	Quais as informações relevantes para o sistema?
IE02	AGR	-- Relevância	Como a relevância da informação é considerada?
IE02	ASS	-- Selecionar	Como selecionar as informações?
IE02	DEP	-- -- Relevância da Informação	Como a relevância da informação influencia na seleção de informações?

Fatores	Tipo	Árvore de Elementos	Pergunta
IE07	AGR	-- Dado	Como os dados das informações são consideradas?
IE07	ASS	-- -- Integrar	Como integrar os dados das informações?
IE04,IE05, IE06	DIR	Aplicação	Quais as aplicações relevantes para o sistema?
IE04,IE05	ASS	-- Interagir	Como interagir com as aplicações?
IE04	DEP	-- -- Serviço	Como os serviços influenciam na interação com aplicações?
IE05	DEP	-- -- Componente	Como os componentes influenciam na interação com aplicações?
IE06	ASS	-- Comunicar	Como comunicar com a aplicação?
IE06	DEP	-- -- Interface de Comunicação Heterogêneas	Como a interface de comunicação heterogênea influencia na comunicação com a aplicação?
IE01,IE08, IE09, IE10,IE11, IE12	DIR	Dispositivo	Quais os dispositivos relevantes para o sistema?
IE08	ASS	-- Cooperar	Como cooperar com os dispositivos?
IE09	ASS	-- Acessar	Como acessar os dispositivos?
IE09	DEP	-- -- Política de Segurança	Como a Política de Segurança influencia no acesso aos dispositivos?
IE10	AGR	-- Característica	Como as características dos dispositivos são consideradas?
IE10	ASS	-- Identificar	Como identificar as características dos dispositivos?
IE11	ASS	-- Descobrir	Como descobrir dispositivos?
IE12	ASS	-- Conectar	Como conectar a dispositivos?
IE01	AGR	-- Relacionamento	Como os relacionamentos de dispositivos são considerados?
IE01,IE03	AGR	-- -- Histórico	Como o histórico dos relacionamentos de dispositivos é considerado?
IE03	ASS	-- -- -- Limpar	Como limpar o histórico dos relacionamentos de dispositivos?

F.5 – Característica Onipresença de Serviços

Fatores	Tipo	Árvore de Elementos	Pergunta
OS02,OS04, OS05, OS06,OS07, OS08	DIR	Serviço	Quais os serviços relevantes para o sistema?
OS02	ASS	-- Desalocar	Como desalocar os serviços?
OS04	ASS	-- Alocar	Como alocar os serviços?
OS01,OS05	ASS	-- Gerenciar	Como gerenciar os serviços?
OS05	DEP	-- -- Container do Serviço	Como o container do serviço influencia a gerência de serviços?
OS05	AGR	-- Container	Como o container do serviço é considerado?
OS06	ASS	-- Organizar	Como organizar os serviços?
OS06	DEP	-- -- Contexto do Serviço	Como o contexto do serviço influencia a organização dos serviços?

Fatores	Tipo	Árvore de Elementos	Pergunta
OS06	AGR	-- Contexto	Como o contexto do serviço é considerado?
OS07	ASS	-- Divulgar	Como divulgar os serviços?
OS08	ASS	-- Fazer cache	Como fazer cache dos serviços?
OS1,OS03	DIR	Usuário	Quais os usuários relevantes para o sistema?
OS01	AGR	-- Seção	Como a seção do usuário é considerada?
OS01,OS05	ASS	-- -- Gerenciar	Como gerenciar a seção do usuário?
OS05	DEP	-- -- -- Container do Serviço	Como o container do serviço influencia a gerência da seção do usuário?
OS03	AGR	-- Mobilidade	Como a mobilidade do usuário é considerada?

F.6 – Característica Sensibilidade ao Contexto

Fatores	Tipo	Árvore de Elementos	Pergunta
SC01,SC02, SC03, SC08,SC22, SC23	DIR	Usuário	Quais os usuários relevantes para o sistema?
SC02	AGR	-- Localização	Como a localização do usuário é considerada?
SC01,SC02, SC03	ASS	-- -- Identificar	Como identificar a localização do usuário?
SC01	AGR	-- Identidade	Como a identidade do usuário é considerada?
SC01,SC02, SC03	ASS	-- -- Identificar	Como identificar a identidade do usuário?
SC03	AGR	-- Atividade	Como a atividade do usuário é considerada?
SC01,SC02, SC03	ASS	-- -- Identificar	Como identificar a identidade do usuário?
SC08	AGR	-- Informação do Usuário	Como as informações do usuário são consideradas?
SC22,SC23	AGR	-- Preferência	Como as preferências do usuário são consideradas?
SC04,SC05, SC06,SC07, SC08,SC11, SC12,SC13, SC14,SC15, SC16,SC17, SC18,SC19, SC20,SC22, SC23,SC24, SC25,SC26	DIR	Informação de Contexto	Quais as informações de contexto relevantes para o sistema?
SC04	AGR	-- Temporalidade	Como a temporalidade das informações de contexto é considerada?
SC05	GEN	-- Informação de Sistema	Quais informações de contexto são consideradas informações de sistema?
SC06	GEN	-- Informação de Infra-estrutura	Quais informações de contexto são consideradas informações de infra-estrutura?
SC07	GEN	-- Informação Física	Quais informações de contexto são consideradas informações físicas?

Fatores	Tipo	Árvore de Elementos	Pergunta
SC08	GEN	-- Informação do Usuário	Quais informações de contexto são consideradas informações do usuário?
SC11,SC22	ASS	-- Contextualizar	Como contextualizar informações de contexto?
SC22,SC23	DEP	-- -- Preferência do Usuário	Como a preferência do usuário influencia na contextualização das informações de contexto?
SC12	ASS	-- Armazenar	Como armazenar informações de contexto?
SC12,SC13	DEP	-- -- Utilidade	Como a utilidade da informação influencia o armazenamento de informações de contexto?
SC12,SC13	AGR	-- Utilidade	Como a utilidade das informações de contexto é considerada?
SC13	ASS	-- -- Avaliar	Como avaliar a utilidade das informações de contexto?
SC14	ASS	-- Consolidar	Como consolidar as informações de contexto?
SC10,SC14, SC17	DEP	-- -- Fonte de Dados	Como a fonte de dados influencia a consolidação de informações de contexto?
SC15	AGR	-- Relacionamento	Como o relacionamento de informações de contexto é considerado?
SC15	ASS	-- -- Gerenciar	Como gerenciar os relacionamentos de informações de contexto?
SC16	ASS	-- Integrar	Como integrar as informações de contexto?
SC17	ASS	-- Categorizar	Como categorizar as informações de contexto?
SC10,SC14, SC17	DEP	-- -- Fonte de Dados	Como a fonte de dados é considerada na categorização das informações de contexto?
SC18	ASS	-- Representar	Como representar as informações de contexto?
SC19,SC24	AGR	-- Semântica	Como a semântica das informações de contexto é considerada?
SC19	ASS	-- -- Organizar	Como organizar a semântica das informações de contexto?
SC20	ASS	-- Derivar	Como derivar as informações de contexto?
SC21	ASS	-- Transformar	Como transformar as informações de contexto?
SC23	ASS	-- Personalizar	Como personalizar as informações de contexto?
SC22,SC23	DEP	-- -- Preferência do Usuário	Como a preferência do usuário influencia na personalização das informações de contexto?
SC4	ASS	-- Interpretar	Como interpretar as informações de contexto?
SC19,SC24	DEP	-- -- Semântica	Como a semântica influencia na interpretação das informações de contexto?
SC25	AGR	-- Dados	Como os dados das informações de contexto são considerados?
SC26	ASS	-- Compartilhar	Como compartilhar as informações de contexto?
SC10,SC14, SC17	DEP	Fonte de Dados	Quais as fontes de dados relevantes para o sistema?
SC09,SC10	GEN	-- Sensor	Quais fontes de dados são consideradas sensores?

Fatores	Tipo	Árvore de Elementos	Pergunta
SC09	ASS	-- -- Controlar	Como controlar os sensores?

APÊNDICE G – GUIA GERAL DE VERIFICAÇÃO DE REQUISITOS DE UBIQUIDADE

Este apêndice apresenta o Guia Geral de Verificação de Requisitos de Ubiquidade.

G.1 – Guia Geral de Verificação de Requisitos de Ubiquidade

Guia de Verificação de Requisitos de Ubiquidade

Este documento apresenta o conjunto de perguntas para apoiar a detecção de defeitos em requisitos de ubiquidade em projetos de software.

Sugere-se a sequência de passos listados abaixo para a verificação dos requisitos:

1. Para cada característica de ubiquidade do projeto, considerar o conjunto de perguntas para verificação de requisitos correspondente.
    a. Os termos sublinhados nas perguntas encontram-se definidos no glossário.
    b. Para cada agrupamento de perguntas, são informados possíveis locais da especificação de requisitos onde a definição do requisito correspondente pode ser encontrada (**item de especificação**).
    c. As perguntas estão organizadas em duas categorias:
        i. **Específicas:** estão associadas a defeitos de omissão. Estas perguntas estão definidas de forma padronizada e seguem a estrutura:

            Está definido + *complemento da funcionalidade* + ?

        ii. **Genéricas:** são perguntas definidas uma vez no início do *checklist* e estão associadas aos defeitos de ambiguidade, inconsistência, fato incorreto e informação estranha.

            As perguntas genéricas devem ser modificadas para cada uma das perguntas específicas durante o processo de revisão. Por exemplo, considere a pergunta específica:

            Está definido em quais dispositivos o software deverá executar?

            As perguntas genéricas complementam esta pergunta e seriam:

            Ambiguidade: *A definição sobre os dispositivos utilizados* ***está clara?***

            Inconsistência*: A associação entre funcionalidade e dispositivo* ***é definida de forma diferente em locais diferentes no documento de requisitos****?*

Fato Incorreto: *A associação entre funcionalidades e o dispositivo em que irá executar* ***está correta****?*

Informação Estranha: *Existe algo descrito no documento de requisitos que está claramente fora do escopo do projeto?*

2. Identificar se a partir da especificação dos requisitos as perguntas formuladas podem ser respondidas satisfatoriamente. Em caso negativo, ir ao passo 3.
3. Relatar as discrepâncias identificadas no relatório de discrepância.

Os tipos de defeitos considerados, a escala de severidade adotada e as perguntas genéricas são:

Tipos de Defeito

Omissão	Informação necessária não incluída
Ambiguidade	Informação passível de ter múltiplas interpretações
Inconsistência	Informações conflitantes
Fato Incorreto	Informação que não é verdadeira para as condições especificadas
Informação Estranha	Informação desnecessária

Escala de Severidade dos Problemas

0	Problema leve - baixa prioridade para consertá-lo
1	Problema grave - alta prioridade para consertá-lo

Perguntas Genéricas

Ambiguidade

A definição está clara? Ou seja, permite apenas um entendimento?

Inconsistência

O item é definido de forma diferente em locais diferentes no documento de requisitos?

Fato Incorreto

A definição está correta?

Informação Estranha

Existe algo descrito no documento de requisitos que está claramente fora do escopo do projeto?

Característica Onipresença de Serviços

OS1. Mobilidade

Item de Especificação: caso de uso, requisitos não funcionais

Omissão

- Está definido como a sessão do usuário será gerenciada?
- Estão definidas quais informações deverão ser consideradas para gerenciar a sessão do usuário?
- Está definido em quais dispositivos o software deverá executar?
- Está definido em quais dispositivos cada funcionalidade irá executar?

OS2. Gerência de Serviços

Item de Especificação: caso de uso

Omissão

- Está definido como o software gerencia os serviços por ele fornecidos?
- Está definido como o software deve fazer uso de serviços externos?
- Está definido o formato do dado a ser importado/exportado durante o uso dos serviços?
- Está definido como o software adapta os serviços por ele fornecidos ao contexto em que o software se encontra?

OS3. Divulgação de Serviços

Item de Especificação: caso de uso

Omissão

- Está definido como o software deve proceder para divulgar seus serviços para sistemas externos?

Característica Sensibilidade ao Contexto

DISC.1. Captura de Informação

Item de Especificação: caso de uso, regra de negócio

Omissão

- Está definido como o software identifica o usuário que o está manipulando?
- Está definido como o software identifica o local onde está sendo utilizado?
- Está definido como o software identifica a atividade que o usuário está realizando?
- Está definido como o software considera a variável tempo ao manter as informações de contexto coletadas?
- Está definido quais informações de contexto de sistema (uso de memória, cpu, tamanho de tela, energia, largura de banda disponível) o software considera?
- Está definido como informações de contexto de sistema (uso de memória, cpu, tamanho de tela, energia, largura de banda disponível) são capturadas?
- Está definido quais informações de contexto de infra-estrutura (estado de funcionamento dos sensores, por exemplo) o software considera?
- Está definido como informações de contexto de infra-estrutura (estado de funcionamento dos sensores, por exemplo) são capturadas?

- Está definido quais informações de contexto físico (temperatura, condições de iluminação, barulho) o software considera?
- Está definido como informações de contexto físico (temperatura, condições de iluminação, barulho) são capturadas?
- Está definido quais informações de usuário (perfil do usuário, preferência) o software considera?
- Está definido como informações de usuário (perfil do usuário, preferência) são capturadas?

DISC.2. Controle de Sensores

Item de Especificação: caso de uso, regra de negócio

Omissão

- Está definido como o software controla o estado (ativado, desativado, em espera...) dos sensores?
- Estão definidos quais são esses estados?
- Está definido como o software interage com os sensores?

DISC.3. Gerência de Informações de Contexto

Item de Especificação: caso de uso, regra de negócio

Omissão

- Está definido como o software mantém as informações de contexto coletadas?
- Está definido o critério que o software considera para definir se uma informação de contexto é útil?
- Está definido como o software consolida as informações de contexto?
- Está definido como o software mantém as relações existentes entre as diferentes informações de contexto?
- Está definido como o software deve efetuar a integração dos dados provenientes de diferentes origens e com diferentes formatos?
- Está definido como as informações identidade, tempo e localização são utilizadas no armazenamento das informações de contexto?
- Está definido como a semântica das informações influencia na manutenção das informações de contexto capturadas?

DISC.4. Interpretação da Informação

Item de Especificação: caso de uso, regra de negócio

Omissão

- Está definido como o software deve derivar novas informações a partir das informações capturadas do contexto?
- Está definido como o software mantém um rastro das transformações efetuadas nas informações?
- Está definido como o software organiza as informações de contexto considerando as preferências do usuário?
- Está definido como o software considera a semântica ao interpretar as informações?

DISC.5. Compartilhamento da Informação

Item de Especificação: caso de uso

Omissão

- Está definido como o compartilhamento das informações com outras aplicações é efetuado?

Característica Captura de Experiência

CE1. Captura de Informação

Item de Especificação: caso de uso

Omissão

- Está definido como as informações de interação do usuário são capturadas?
- Estão definidas quais informações deverão ser consideradas sobre a interação do usuário com o sistema?

CE2. Interpretação de Informação

Item de Especificação: caso de uso, regra de negócio

Omissão

- Está definido como o software identifica padrões de uso?

CE3. Gerência de Informações

Item de Especificação: caso de uso, regra de negócio

Omissão

- Está definido como o software deve proceder para armazenar as informações de interação do usuário com o sistema capturadas?
- Está definido como o software deve proceder para identificar as preferências de uso do sistema?

Característica Heterogeneidade de Dispositivos

HD1. Busca por Dispositivos

Item de Especificação: caso de uso, regra de negócio

Omissão

- Está definido como o software efetua a busca por outros dispositivos?
- Está definido como o software identifica se um dispositivo é ou não compatível?
- Está definido como o software identifica se o dispositivo está disponível para uso?

HD2. Migração de Aplicação

Item de Especificação: caso de uso, regra de negócio

Omissão

- Está definido como o software efetua a migração de parte de suas funcionalidades para serem executadas em outros dispositivos?
- Está definido como o software gerencia a sessão do usuário durante a migração da aplicação?
- Está definido como o software gerencia seu estado durante a migração das funcionalidades?

HD3. Compartilhamento de Recursos

Item de Especificação: requisitos não funcional

Omissão

- Está definido como o software deve compartilhar os recursos de hardware que está alocando?

Característica Interoperabilidade Espontânea

IE1. Gerência de Informações

Item de Especificação: caso de uso, regra de negócio

Omissão

- Está definido como o software mantém um histórico de sistemas com os quais interagiu para solicitar a execução de suas funcionalidades?
- Está definido como o software seleciona e descarta informações do histórico de acordo com a sua relevância?
- Está definido como o software interage com outros sistemas para executar parte de suas funcionalidades?
- Está definido como o software integra informações heterogêneas de diferentes aplicações?

IE2. Interoperabilidade

Item de Especificação: caso de uso, regra de negócio

Omissão

- Está definido como o software efetua a busca por outros dispositivos?
- Está definido como o software identifica se um dispositivo é ou não compatível?
- Está definido como o software identifica se o dispositivo está disponível para uso?

G.2 – Glossário de Termos em Computação Ubíqua

Onipresença de Serviços

GOS1. **Sessão do Usuário:** refere-se ao período em que há interações entre usuário e sistema;

GOS2. **Gerência da Sessão do Usuário:** refere-se a funcionalidades associadas à manutenção do estado da interação entre usuário e sistema enquanto o usuário faz uso

dele. Por exemplo, ao efetuar login, o sistema deve disponibilizar o conjunto de funcionalidades associadas ao usuário enquanto sua sessão for válida.

GOS3. **Serviço:** refere-se a qualquer funcionalidade disponibilizada pelo software ou, disponibilizada no "ambiente" e utilizada pelo software;

GOS4. **Gerenciar Serviço:** refere-se à capacidade de:

- **Ativar Serviço:** executar a funcionalidade do serviço, tornar o serviço disponível para uso;
- **Desativar:** interromper a funcionalidade do serviço, tornar o serviço indisponível para uso;
- **Coletar informações do serviço** (por exemplo: descrição dos serviços, formato de entrada de dados esperada, formato de saída de dados disponibilizada)

GOS5. **Cenário:** indica o conjunto de variáveis que podem influenciar no funcionamento do sistema (por exemplo: localização, contexto físico (GSC10), contexto de sistema (GSC8));

GOS6. **Divulgar Serviço:** disponibilizar informações do serviço (por exemplo: descrição dos serviços, formato de entrada de dados esperada, formato de saída de dados disponibilizada) para sistemas externos;

GOS7. **Disponibilidade de Serviço:** indica se um determinado serviço pode ou não ser utilizado naquele momento. Fatores como desativação do serviço ou número máximo de instâncias dos serviços sendo usadas são exemplos que podem interferir na disponibilidade de um serviço.

Sensibilidade ao Contexto

GSC1. **Capturar Informação:** refere-se à capacidade que o sistema possui de, a partir de sensores ou outro dispositivo que permita a captura de dados (por exemplo, webcam), capturar informações do ambiente;

GSC2. **Entidade Externa:** refere-se qualquer objeto (por exemplo: pessoa, etiqueta eletrônica) ou estado (por exemplo: temperatura, iluminação, ruído) que pode ter informações capturadas a seu respeito.

GSC3. **Identificar Usuário:** refere-se à capacidade que o sistema possui de identificar o usuário que o está manipulando. Esta identificação pode ser, por exemplo, através de logins, reconhecimento de face, digital;

GSC4. **Identificar Local de Uso:** refere-se à capacidade que o software possui de, a partir de variáveis do ambiente (por exemplo, nível de ruído, informações de posicionamento – GPS), identificar o local onde ele está sendo utilizado. Em geral, este tipo de funcionalidade é importante caso o software deva adaptar seu uso (por exemplo, habilitar um conjunto de funcionalidades ao usuário chegar com seu smartphone no escritório) ao ambiente em que se encontra;

GSC5. **Identificar Atividade:** refere-se à capacidade que o software possui de, a partir da captura de informações do usuário, identificar que atividade ele está realizando. Por

exemplo: um software de direção ativa para automóveis deve monitorar constantemente o motorista para identificar se ele está sonolento;

GSC6. **Organizar Informações de Contexto considerando a variável Tempo:** refere-se à capacidade que o software deve possuir de, ao armazenar as informações de contexto, armazenar também o momento em que elas foram capturadas;

GSC7. **Informação de Contexto:** refere-se a informações capturadas sobre qualquer entidade externa ao software que esteja sendo controlada;

GSC8. **Contexto de Sistema:** quando a variável do ambiente que está sendo monitorada refere-se a informações associadas às restrições da infra-estrutura (uso de memória, cpu, tamanho de tela, energia, largura de banda disponível), tem-se o conceito de contexto de sistema;

GSC9. **Contexto de Infra-estrutura:** quando a variável do ambiente que está sendo monitorada refere-se a informações associadas ao correto funcionamento da infra-estrutura (por exemplo: estado de funcionamento dos sensores), tem-se o conceito de contexto de infra-estrutura;

GSC10. **Contexto Físico:** quando a entidade externa que está sendo monitorada refere-se a informações associadas ao ambiente (por exemplo: temperatura, condições de iluminação, ruído), tem-se o conceito de contexto físico;

GSC11. **Contexto de Usuário:** quando a entidade externa que está sendo monitorada refere-se a informações associadas ao usuário (por exemplo: preferências, ação), tem-se o conceito de contexto usuário;

GSC12. **Sensor:** refere-se a qualquer dispositivo que permita a captura de informações de contexto (por exemplo: webcam, sensor Bluetooth, sensores em geral);

GSC13. **Filtrar Informações de Contexto Úteis:** a quantidade de informações capturadas pode ser muito grande, o que pode inviabilizar seu armazenamento pelo software. Este item indica que o software pode ter a necessidade de possuir alguma regra que defina quando uma informação de contexto é considerada útil para que só elas sejam armazenadas;

GSC14. **Organizar as Informações de acordo com o Sensor:** refere-se à capacidade que o software deve possuir de considerar a origem dos dados como uma informação a ser considerada no seu armazenamento;

GSC15. **Relações entre Informações de Contexto:** refere-se à existência de relacionamentos entre informações capturadas do ambiente e cuja manutenção é importante para uso dos dados coletados;

GSC16. **Integrar Informações:** refere-se à capacidade que o software pode ter para integrar dados provenientes de diferentes origens. Este integração pode envolver, por exemplo, conversões numéricas, estrutura dos dados.

GSC17. **Semântica da Informação:** refere-se ao significado do dado que está sendo manipulado. Este tipo de conceito pode ser útil ao estruturar as informações capturadas para armazenamento;

GSC18. **Derivar Informação:** refere-se à capacidade que o software poder ter de, a partir de um conjunto de dados coletados, criar novas informações. Por exemplo, a partir de informações de contexto de ambiente coletadas como umidade do ar e temperatura é possível indicar se há possibilidade de chuva;

GSC19. **Preferência do Usuário:** indica informações que definem as preferências do usuário para uso do sistema;

GSC20. **Compartilhar Informação de Contexto:** refere-se à capacidade que o software pode ter de disponibilizar para sistemas externos as informações de contexto que ele capturou.

Comportamento Adaptável

GCA1. **Adaptar:** refere-se à capacidade que o sistema tem alterar sua forma de funcionar ou como um serviço em particular está sendo provido de forma a se adequar às limitações do ambiente;

GCA2. **Selecionar Alternativa de Adaptação:** refere-se à capacidade que o software tem de, a partir de um conjunto de adaptações possíveis, selecionar a alternativa mais adequada. Isto pode implicar na necessidade de definição de um conjunto de regras de seleção;

GCA3. **Gerenciar Disponibilidade de Recurso:** refere-se à capacidade que o software tem de verificar constantemente a disponibilidade dos recursos (serviços, memória, cpu) de que faz uso e manter esta informação atualizada para que possam ser tomadas decisões sobre o uso de recursos pelo software;

GCA4. **Analisar Impacto de Adaptação:** refere-se à capacidade que o software tem de identificar quais possíveis impactos uma adaptação pode trazer para o funcionamento do sistema. Por exemplo, aumentar o volume do auto-falante irá aumentar o consumo de energia;

GCA5. **Conteúdo:** refere-se às informações (dados) e interface com usuário;

a. **Adaptar Conteúdo:** refere-se à capacidade que o software tem de, por exemplo, limitar ou expandir as informações que são apresentadas com base no cenário em que se encontra;

GCA6. **Adaptar Código:** refere-se à capacidade que o software tem de executar de forma diferenciada funcionalidades com base nas restrições (largura de banda, capacidade de processamento, dispositivo em que a funcionalidade está executando) impostas pelo ambiente;

Captura de Experiência

GCE1. **Capturar Interação do Usuário com o Sistema:** refere-se à capacidade que o sistema tem de capturar informações sobre interações do usuário com o sistema. Por

exemplo: imaginemos um software para gerenciamento da casa. Ele pode capturar informações da interação do usuário com o sistema como: o morador ao chegar liga a lâmpada da sala, esquenta a água para o café, liga a banheira e define a temperatura da água para 28º C

GCE2. **Identificar Padrões de Uso:** refere-se à capacidade que o software tem de, a partir das informações de interação do usuário com o sistema capturadas, identificar algum padrão de uso do sistema. Por exemplo: ao perceber que o comportamento descrito no exemplo de GCE1 se repete (uma experiência), o software poderá gerenciar estas atividades quando o morador chegar em casa sem a necessidade deste desempenhar diretamente estas atividades.

Heterogeneidade de Dispositivo

GHD1. **Migração de Serviços:** refere-se à capacidade que o sistema tem de executar parte de seus serviços em outro dispositivo (diferente do que ele está executando no momento);

GHD2. **Dispositivo Compatível:** refere-se a qualquer dispositivo que atenda às restrições de tecnologia (plataforma, capacidade de processamento, memória disponível para uso, tamanho de tela) do sistema;

GHD3. **Identificar Dispositivo Compatível:** refere-se à capacidade que o sistema tem de identificar no ambiente dispositivos compatíveis;

GHD4. **Identificar Disponível para Uso:** refere-se à capacidade que o sistema tem de identificar no ambiente dispositivos que estejam disponíveis para uso. É importante ressaltar que não necessariamente um dispositivo compatível (GHD3) identificado no ambiente está disponível para uso;

GHD5. **Dispositivo Diferente:** cada dispositivo possui uma identidade que o define unicamente no sistema. O sistema pode estar em execução em mais de um dispositivo ao mesmo tempo. Cada dispositivo que compõem este cenário é considerado um dispositivo diferente;

GHD6. **Estado da Aplicação:** refere-se ao conjunto de variáveis e funcionalidades (em execução, parada...) que caracterizam o sistema em um determinado momento do tempo;

GHD7. **Gerência do Estado da Aplicação:** refere-se a funcionalidades associadas à manutenção do estado da aplicação. Por exemplo, uma aplicação possui um conjunto de variáveis em memória e está executando um conjunto de funcionalidades quando é interrompida. Ao ser reiniciada, a aplicação deve possuir o mesmo conjunto de variáveis que possuía antes de ser interrompida e ser capaz de voltar a executar as funcionalidades do ponto onde parou;

GHD8. **Compartilhar Recursos de Hardware:** refere-se à capacidade que a aplicação possui de disponibilizar para sistemas externos o uso do hardware.

APÊNDICE H – INSTRUMENTOS DO ESTUDO EXPERIMENTAL PARA AVALIAÇÃO DE UBICHECK

Este apêndice descreve os instrumentos usados no estudo experimental realizado ao longo deste trabalho que avaliou a abordagem UbiCheck, de apoio à definição de requisitos de ubiquidade em projeto de software ubíquo.

H.1 – Consentimento de Participação

Eu declaro ter mais de 18 anos de idade e que concordo em participar de estudos conduzidos pelos pesquisadores Prof. Guilherme Horta Travassos, Jobson Massolar da Silva e Rodrigo Oliveira Spínola.

PROCEDIMENTO

Eu entendo que serei solicitado a participar de atividades de desenvolvimento considerando os requisitos de um Sistema de Gestão Patrimonial. Nestes exercícios alguns métodos experimentais serão aplicados visando avaliar aspectos associados ao desenvolvimento de projetos de software.

Os pesquisadores conduzirão os estudos consistindo da coleta, análise e relato dos dados do exercício. Eu entendo que não tenho obrigação alguma em contribuir com informação sobre meu desempenho nos exercícios, e que posso solicitar a retirada de meus resultados dos experimentos a qualquer momento. Eu entendo também que quando os dados forem coletados e analisados, meu nome será removido dos dados e que este não será utilizado em nenhum momento durante a análise ou quando os resultados forem apresentados.

CONFIDENCIALIDADE

Toda informação coletada nestes estudos é confidencial, e meu nome não será identificado em momento algum. Da mesma forma, me comprometo a manter sigilo das tarefas solicitadas e dos documentos apresentados e que fazem parte dos experimentos.

BENEFÍCIOS, LIBERDADE DE DESISTÊNCIA

Eu entendo que sou livre para realizar perguntas a qualquer momento ou solicitar que qualquer informação relacionada a minha pessoa não seja incluída no estudo. Eu entendo que participo

dos estudos experimentais de livre e espontânea vontade com o único intuito de contribuir para o avanço e desenvolvimento de técnicas, ferramentas e processos para a Engenharia de Software.

RESPONSÁVEIS
Prof. Guilherme Horta Travassos
Jobson Massolar da Silva
Rodrigo Oliveira Spínola
Programa de Engenharia de Sistemas e Computação
COPPE/UFRJ

Nome (em letra de forma): _____

Assinatura: _____ Data:_____

H.2 – Caracterização dos Participantes

Caracterização do Desenvolvedor

Nome _____ **Nível (Ms.c/D.Sc.):** _____

Formação Geral

Qual é sua experiência anterior com desenvolvimento de software na prática ? (marque aqueles itens que melhor se aplicam)

__ nunca desenvolvi software.

__ tenho desenvolvido software para uso próprio.

__ tenho desenvolvido software como parte de uma equipe, relacionado a um curso.

__ tenho desenvolvido software como parte de uma equipe, na indústria.

Por favor, explique sua resposta. Inclua o número de semestres ou número de anos de experiência relevante em desenvolvimento (E.g. “Eu trabalhei por 10 anos como programador na indústria”)

Experiência em Desenvolvimento de Software

Por favor, indique o grau de sua experiência nesta seção seguindo a escala de 5 pontos abaixo:

1 = nenhum

2 = estudei em aula ou em livro

3 = pratiquei em 1 projeto em sala de aula

4 = usei em 1 projeto na indústria

5 = usei em vários projetos na indústria

Experiência com Requisitos

•	Experiência escrevendo requisitos	1	2	3	4	5
•	Experiência escrevendo casos de uso	1	2	3	4	5
•	Experiência revisando requisitos	1	2	3	4	5
•	Experiência revisando casos de uso	1	2	3	4	5
•	Experiência modificando requisitos para manutenção	1	2	3	4	5

Experiência em Projeto

•	Experiência em projeto de sistemas	1	2	3	4	5
•	Experiência em desenvolver projetos a partir de requisitos e casos de uso	1	2	3	4	5
•	Experiência criando projetos Orientado a Objetos	1	2	3	4	5
•	Experiência lendo projetos Orientado a Objetos	1	2	3	4	5
•	Experiência com Unified Modeling Language (UML)	1	2	3	4	5
•	Experiência alterando projeto para manutenção	1	2	3	4	5

Outras Experiências

•	Experiência com gerenciamento de projeto de software?	1	2	3	4	5
•	Experiência com inspeções de software?	1	2	3	4	5
•	Experiência com planejamento de inspeções de software?	1	2	3	4	5
•	Experiência com testes de integração de software?	1	2	3	4	5
•	Experiência desenhando interfaces com o usuário?	1	2	3	4	5
•	Experiência com avaliação de usabilidade de software?	1	2	3	4	5

Conhecimento do Domínio

Nós usaremos esta seção para compreender quão familiar você está com diferentes domínios de aplicação.

Por favor, indique o grau de experiência nesta seção seguindo a escala de 3 pontos abaixo:

1 = Eu não tenho familiaridade com a área.
3 = Eu tenho alguma familiaridade com a área.
5 = Eu sou muito familiar com esta área.

Quanto você sabe sobre…

• Sistema de Gestão Patrimonial?	1	3	5
• Sistema para Bicicletário?	1	3	5
• Sistema para Monitoração de Itens de Patrimônio?	1	3	5

H.3 – Orientações para a Operação do Trabalho

H.3.1 – Versão Entregue aos Participantes da Abordagem *Ad hoc*

Formulário UB_1 – Execução do Exercício/*Abordagem Proposta*

Nome: _____

Grupo: _____ **Data:** _____

Objetivo: Medir os tempos e conclusões

Material

Verifique que você recebeu todos os instrumentos que você precisa para realizar este exercício. Você deve possuir:

- Um documento de requisitos do Sistema de Gestão de Inventário Patrimonial

- Um documento de descrevendo o cenário de ubiquidade a ser considerado

- Os Formulários UB_1, UB_3

Exercício

Os passos a serem seguidos são:

1. Para realizar este exercício relembre que você deve:

a. Ler a documentação do projeto.

b. Ler o cenário de ubiquidade a ser considerado.

c. Especificar os requisitos de ubiquidade do projeto.

2. Hora em que você começou a examinar a documentação do projeto: ___:___ (hora:minuto)
3. Hora em que você terminou de examinar a documentação do projeto: ___:___ (hora:minuto)
4. Tempo gasto para realizar a especificação dos requisitos de ubiquidade (sem considerar possíveis interrupções): _____ horas
5. Tempo que você levou para fazer o exercício (não considere possíveis interrupções): _____ minutos

Conclusões

O objetivo destas conclusões é estudar a evolução das opiniões dos participantes sobre o problema de definição de requisitos de ubiquidade enquanto estava fazendo este exercício.

1. O que você aprendeu com este exercício?

- Em relação à definição de requisitos;

- Em relação ao domínio de ubiquidade.

2. Se você tivesse que fazer este exercício novamente, você faria coisas diferentes? Quais? Por quê?

H.3.2 – Versão Entregue aos Participantes que utilizaram *UbiCheck*

Formulário UB_2 – Execução do Exercício/*Abordagem Proposta*

Nome: _____

Grupo: _____ **Data:** _____

Objetivo: Medir os tempos e conclusões

Material

Verifique que você recebeu todos os instrumentos que precisa para realizar este exercício:

- Um documento de requisitos do Sistema de Gestão de Inventário Patrimonial
- Um documento de descrevendo o cenário de ubiquidade a ser considerado
- Um Guia de Apoio à Definição de Requisitos de Ubiquidade
- Os Formulários UB_2, UB_3

Exercício

Os passos a serem seguidos são:

1. Para realizar este exercício relembre que você deve:

a. Ler a documentação do projeto.

b. Ler o cenário de ubiquidade a ser considerado.

c. Examinar a Caracterização do Projeto

d. Examinar o Checklist de apoio à definição de Requisitos de Ubiquidade

e. Especificar os requisitos de ubiquidade do projeto.

2. Hora em que você começou a examinar a documentação do projeto: ___:___ (hora:minuto)
3. Hora em que você terminou de examinar a documentação do projeto: ___:___ (hora:minuto)
4. Tempo gasto examinando a documentação do projeto (sem considerar interrupções): _____ minutos
5. Hora em que você começou a examinar caracterização do projeto: ___:___ (hora:minuto)
6. Hora em que você terminou de examinar a caracterização do projeto: ___:___ (hora:minuto)
7. Tempo gasto examinando a caracterização do projeto (sem considerar interrupções): _____ minutos
8. Hora em que você começou a examinar o Checklist de apoio à definição de Requisitos de Ubiquidade: ___:___ (hora:minuto)
9. Hora em que você terminou de examinar o Checklist de apoio à definição de Requisitos de Ubiquidade: ___:___ (hora:minuto)

10. Tempo gasto examinando o Checklist de apoio à definição de Requisitos de Ubiquidade (sem considerar interrupções): _____ minutos
11. Tempo gasto para realizar a especificação dos requisitos de ubiquidade (sem considerar possíveis interrupções): _____ horas
12. Tempo que você levou para fazer o exercício (não considere possíveis interrupções): _____ minutos

Conclusões

O objetivo destas conclusões é estudar a evolução das opiniões dos participantes sobre o problema de definição de requisitos de ubiquidade enquanto estava fazendo este exercício.

1. O que você aprendeu com este exercício?

- Em relação à definição de requisitos;

- Em relação ao domínio de ubiquidade.

2. Se você tivesse que fazer este exercício novamente, você faria coisas diferentes? Quais? Por quê?

H.4 - Questionário de Acompanhamento

Formulário UB_3 – Avaliação da Abordagem Proposta

Nome:_____

Objetivo: Verificar o potencial que os participantes observam sobre a abordagem proposta

O objetivo deste formulário é conhecer a opinião dos participantes sobre a Abordagem Proposta e estudar a opinião dos participantes sobre o problema da definição de requisitos de ubiquidade durante o exercício.

1. Você acha que a abordagem proposta facilita a definição de requisitos de ubiquidade? Por quê?

2. Você acha que a abordagem é simples de ser entendida?

3. Quais as principais dificuldades encontradas no uso da abordagem para apoiar a definição dos requisitos de ubiquidade?

4. Você sentiu falta de algum apoio adicional no uso da abordagem?

5. Você acredita que a abordagem proposta possa ser melhorada para tornar seu uso mais simples? Como?

6. O que você aprendeu sobre a abordagem proposta?

APÊNDICE I – CENÁRIOS DE UBÍQUIDADE

Este apêndice apresenta os cenários de ubiquidade utilizados nos estudos experimentais descritos nos Capítulos 7 e 8.

I.1 – Cenário Bicicletário

Cenários de Ubiquidade

Este documento descreve um novo cenário de uso para o Sistema de Gestão de Inventário Patrimonial. Este cenário traz um conjunto de funcionalidades que demandam soluções inseridas no contexto da ubiquidade computacional.

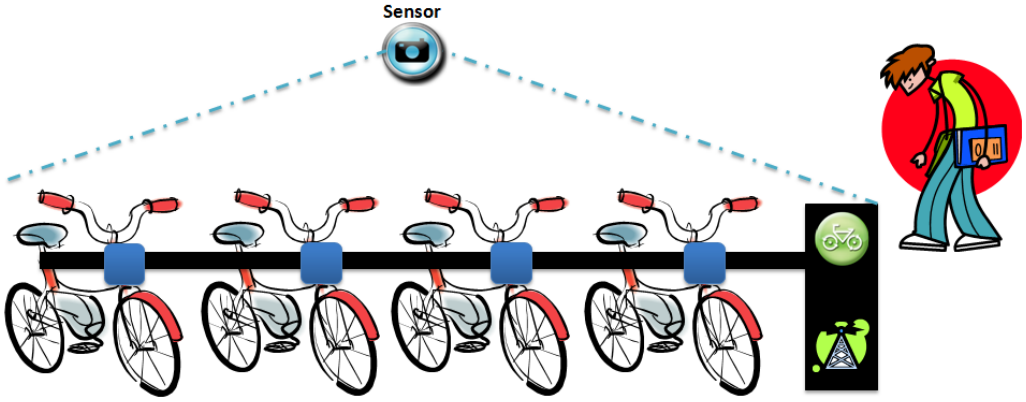
Cenário Bicicletário

Bicicletário: Disponibilizar bicicletas para facilitar o deslocamento dentro da organização.

Para lidar com esta demanda, as seguintes necessidades foram definidas:

- As bicicletas deverão ser classificadas como item rastreável. Uma bicicleta rastreável é aquela que, além do número de patrimônio, possui uma etiqueta eletrônica de identificação;
- Cada indivíduo da organização deve possuir um *smart card*. Um *smart card* é um cartão que indica quais são as permissões de uso da bicicleta;
- As bicicletas estarão presas a barras de controle que possuirão um leitor para *smart card;*
- Desta forma, para fazer uso de uma bicicleta, o indivíduo deverá passar o *smart card* pelo leitor. O sistema deverá fazer a liberação de uma bicicleta preza à barra de controle e o sistema deverá identificar qual bicicleta foi liberada através dos sensores.
- Ao fazer a devolução de uma bicicleta, o indivíduo deverá apenas conectar a bicicleta à barra de controle. Ao fazer isso, o sistema identificará a bicicleta devolvida e notificará a devolução ao sistema de controle de patrimônio.
- Caso o *smart card* não esteja habilitado a fazer retirada da bicicleta, o setor de controle de patrimônio deve ser notificado e uma mensagem deve ser enviada para o celular do indivíduo indicando que ele não pode fazer a retirada da bicicleta (nem todos os indivíduos podem fazer a retirada da bicicleta em todos os pontos em que bicicletas podem ser retiradas/devolvidas).
- Os sensores devem ter seus estados controlados pelo sistema.

A Figura abaixo ilustra um cenário de integração do sistema ao ambiente da organização:



Com base na figura e na descrição das necessidades adicionais apresentadas anteriormente, deve-se ter a seguinte descrição em mente:

1. Imaginemos um bicicletário. Devem existir sensores instalados para identificação das bicicletas e também um leitor de *smart card* para identificação do indivíduo que está retirando uma bicicleta;
2. Para que uma bicicleta possa ser retirada, deve existir uma autorização prévia. Esta autorização é recuperada pelo sistema através de um serviço web disponibilizado pelo setor de RH;
3. Ao passar o *smart card* no leitor, o sistema verifica junto ao serviço web se a bicicleta está autorizada e autoriza ou não sua saída com base nas informações recebidas para o sistema da barra de controle;
4. Ao fazer a autorização, o sensor identificará qual bicicleta foi retirada e notificará sua saída ao sistema de gestão de patrimônio.
5. Ao fazer a devolução de uma bicicleta, o indivíduo deverá apenas conectar a bicicleta à barra de controle. Ao fazer isso, o sistema identificará a bicicleta devolvida e notificará a devolução ao sistema de controle de patrimônio. Caso a bicicleta devolvida não seja a mesma que foi retirada, o indivíduo deve ser notificado via celular.

I.2 – Cenário Item Rastreável

Cenários de Ubiquidade

Este documento descreve um novo cenário de uso para o Sistema de Gestão de Inventário Patrimonial. Este cenário traz um conjunto de funcionalidades que demandam soluções inseridas no contexto da ubiquidade computacional.

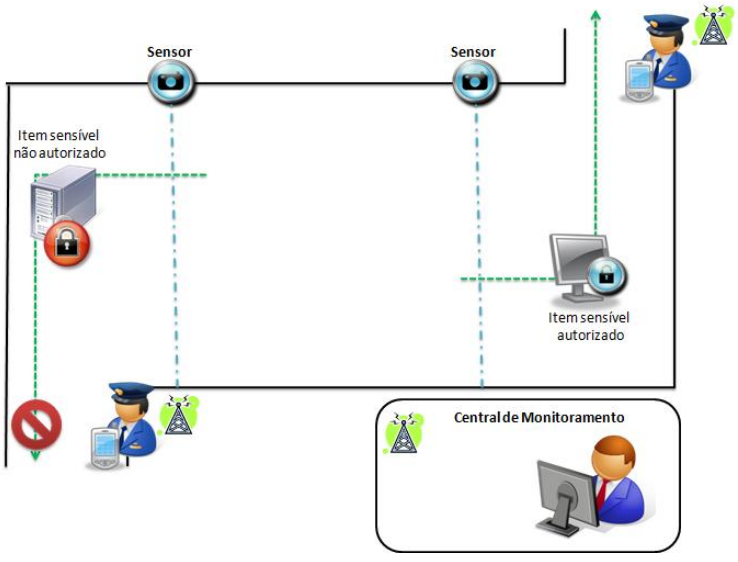
Cenário – Item Rastreável

Item Patrimonial Rastreável: controle efetivo dos itens de patrimônio considerando seu transporte e localização apoiado pelo uso de sensores e etiquetas eletrônicas nos itens patrimoniais.

Para lidar com esta demanda, as seguintes necessidades foram definidas:

- Os bens de capital passam a ser classificados também como item rastreável. Um item rastreável é aquele que, além do número de patrimônio, possui uma etiqueta eletrônica de identificação;
- Para ser classificado como rastreável, um item deve ter seus dados incluídos no cadastro de itens rastreáveis;
- As entradas e saídas dos prédios terão sensores para estas etiquetas eletrônicas de identificação. Ao passar com um item rastreável, deve ser registrado no sistema o evento, enviada uma mensagem para o setor de segurança e fazer com que o agente de portaria seja avisado (em seu dispositivo móvel) que um item rastreável está saindo/entrando na instalação;
- A saída de um item rastreável passa a ser feita em duas etapas: (1) deve haver um cadastro de transferência considerando a solução atual descrita no documento de referência; (2) deve haver uma confirmação da saída do item pelo setor de segurança – essa confirmação envolve tanto a verificação se o item que está sendo retirado foi autorizado e se o indivíduo que o está retirando está associado ao item como responsável pela sua retirada;
- A confirmação ou proibição de saída de um item rastreável deve sempre ser notificada para a central de monitoramento que acompanhará em tempo real as entradas e saídas de itens rastreáveis;
- Os sensores devem ter seus estados controlados pelo sistema.

A Figura abaixo ilustra um cenário de integração do sistema ao ambiente da organização:



Com base na figura e na descrição das necessidades adicionais apresentadas, deve-se ter a seguinte descrição em mente:

1. Imaginemos o corredor do Bloco H com suas duas saídas. Em ambos os lados, devem existir sensores instalados a uma certa distância da saída do corredor de forma que o setor de segurança possa ser notificado da saída de um item rastreável antes de sua saída;
2. Para que um item rastreável possa ser retirado, deve existir uma autorização prévia indicando também quem poderá retirar o item;
3. Ao passar por um sensor, uma mensagem é enviada ao setor de segurança indicando se o item está autorizado e qual indivíduo pode fazer sua retirada;
4. O setor de segurança recebe esta mensagem em um smartphone, faz a verificação da saída do item e autoriza ou não sua saída com base nas informações recebidas;
5. O resultado desta ação é enviado para a central de monitoramento.

APÊNDICE J – INSTRUMENTOS DO ESTUDO EXPERIMENTAL PARA AVALIAÇÃO DE UBIVERI

Este apêndice descreve os instrumentos usados no estudo experimental realizado ao longo deste trabalho que avaliou a abordagem UbiVeri, de apoio à verificação de requisitos de ubiquidade em projeto de software ubíquo.

J.1 – Consentimento de Participação

Eu declaro ter mais de 18 anos de idade e que concordo em participar de estudos conduzidos pelos pesquisadores Prof. Guilherme Horta Travassos, Jobson Massolar da Silva e Rodrigo Oliveira Spínola.

PROCEDIMENTO

Eu entendo que serei solicitado a participar de atividades de desenvolvimento considerando os requisitos de um Sistema de Gestão Patrimonial. Nestes exercícios alguns métodos experimentais serão aplicados visando avaliar aspectos associados ao desenvolvimento de projetos de software.

Os pesquisadores conduzirão os estudos consistindo da coleta, análise e relato dos dados do exercício. Eu entendo que não tenho obrigação alguma em contribuir com informação sobre meu desempenho nos exercícios, e que posso solicitar a retirada de meus resultados dos experimentos a qualquer momento. Eu entendo também que quando os dados forem coletados e analisados, meu nome será removido dos dados e que este não será utilizado em nenhum momento durante a análise ou quando os resultados forem apresentados.

CONFIDENCIALIDADE

Toda informação coletada nestes estudos é confidencial, e meu nome não será identificado em momento algum. Da mesma forma, me comprometo a manter sigilo das tarefas solicitadas e dos documentos apresentados e que fazem parte dos experimentos.

BENEFÍCIOS, LIBERDADE DE DESISTÊNCIA

Eu entendo que sou livre para realizar perguntas a qualquer momento ou solicitar que qualquer informação relacionada a minha pessoa não seja incluída no estudo. Eu entendo que participo dos estudos experimentais de livre e espontânea vontade com o único intuito de contribuir para o avanço e desenvolvimento de técnicas, ferramentas e processos para a Engenharia de Software.

RESPONSÁVEIS
Prof. Guilherme Horta Travassos
Jobson Massolar da Silva
Rodrigo Oliveira Spínola
Programa de Engenharia de Sistemas e Computação
COPPE/UFRJ

Nome (em letra de forma): _____

Assinatura: _____ Data:_____

J.2 – Caracterização dos Participantes

Nome _____ **Nível (Ms.c/D.Sc.):** _____

Experiência Prática Com Desenvolvimento De Software

Qual é sua experiência anterior com desenvolvimento de software na prática? (marque aqueles itens que melhor se aplicam)

__ nunca desenvolvi software.
__ tenho desenvolvido software para uso próprio.
__ tenho desenvolvido software como parte de uma equipe, relacionado a um curso.
__ tenho desenvolvido software como parte de uma equipe, na indústria.

Por favor, explique sua resposta. Inclua o número de semestres ou número de anos de experiência relevante em desenvolvimento (E.g. “Eu trabalhei por 10 anos como programador na indústria”)
_____
_____
_____

Experiência com Desenvolvimento de Software

Por favor, indique o grau de sua experiência nesta seção seguindo a escala de 5 pontos abaixo:

1 = nenhum
2 = estudei em aula ou em livro
3 = pratiquei em 1 projeto em sala de aula
4 = usei em 1 projeto na indústria
5 = usei em vários projetos na indústria

- Experiência com gerenciamento de projeto de software? 1 2 3 4 5
- Experiência com requisitos de software? 1 2 3 4 5
- Experiência inspecionando requisitos de software? 1 2 3 4 5

Conhecimento do Domínio

Nós usaremos esta seção para compreender quão familiar você está com diferentes domínios de aplicação.

Por favor, indique o grau de experiência nesta seção seguindo a escala de 3 pontos abaixo:

1 = Eu não tenho familiaridade com a área.
3 = Eu tenho alguma familiaridade com a área.
5 = Eu sou muito familiar com esta área.

Quanto você sabe sobre…

• Sistema de Gestão Patrimonial?	1	3	5
• Sistema para Bicicletário?	1	3	5
• Sistema para Monitoração de Itens de Patrimônio?	1	3	5

J.3 – Orientações para a Operação do Trabalho

J.3.1 – Versão Entregue aos Participantes da Abordagem Ad hoc

Formulário UB_1 – Execução do Exercício/*Ad hoc*

Nome: _____

Grupo: _____ **Data:** _____

Objetivo: Medir os tempos e conclusões

Material

Verifique que você recebeu todos os instrumentos que você precisa para realizar este exercício. Você deve possuir:

- Um documento de requisitos do Sistema de Gestão de Inventário Patrimonial

- Glossário de Termos de Ubiquidade

- Um documento de descrevendo o cenário de ubiquidade a ser considerado

- Um formulário para relato das discrepâncias identificadas

- O Formulário UB_1

Cenário Descritivo

Neste trabalho você assumirá o papel de um inspetor de software e realizará atividades associadas à garantia da qualidade de requisitos de software. De posse do material listado no item *Material*, você deverá efetuar a **revisão dos requisitos de ubiquidade** do Sistema de Gestão de Inventário Patrimonial considerando o cenário de ubiquidade associado. Os defeitos identificados deverão ser reportados no formulário de relato das discrepâncias.

Lembre-se de registrar o esforço em tempo investido nestas atividades e de preencher as informações que constam neste formulário e no formulário UB_1.

Exercício

Os passos a serem seguidos são:

6. Para realizar este exercício relembre que você deve:

a. Ler a documentação do projeto.

b. Ler o cenário de ubiquidade a ser considerado.

c. Revisar os requisitos de ubiquidade do projeto.

7. Hora em que você começou a examinar a documentação do projeto: ___:___ (hora:minuto)
8. Hora em que você terminou de examinar a documentação do projeto: ___:___ (hora:minuto)
9. Tempo gasto para realizar a revisão dos requisitos de ubiquidade (sem considerar possíveis interrupções): _____ horas
10. Tempo que você levou para fazer o exercício (não considere possíveis interrupções): _____ minutos

Conclusões

O objetivo destas conclusões é estudar a evolução das opiniões dos participantes sobre o problema de verificação de requisitos de ubiquidade enquanto estava fazendo este exercício.

3. O que você aprendeu com este exercício?

- Em relação à verificação de requisitos;

- Em relação ao domínio de ubiquidade.

_____
_____
_____

4. Se você tivesse que fazer este exercício novamente, você faria coisas diferentes? Quais? Por quê?

_____
_____
_____

J.3.2 – Versão Entregue aos Participantes da Abordagem Proposta

Formulário UB_2 – Execução do Exercício/*Abordagem Proposta*

Nome: _____

Grupo: _____ **Data:** _____

Objetivo: Medir os tempos e conclusões

Material

Verifique que você recebeu todos os instrumentos que precisa para realizar este exercício:

- Um documento de requisitos do Sistema de Gestão de Inventário Patrimonial

- Um documento descrevendo o cenário de ubiquidade a ser considerado

- Glossário de Termos de Ubiquidade

- Um Guia de Apoio à Verificação de Requisitos de Ubiquidade

- Um formulário para relato de discrepâncias

- Os Formulários UB_2, UB_3

Cenário Descritivo

Neste trabalho você assumirá o papel de um inspetor de software e realizará atividades associadas à garantia da qualidade de requisitos de software. De posse do material listado no item *Material*, você deverá efetuar a **revisão dos requisitos de ubiquidade** do Sistema de Gestão de Inventário Patrimonial considerando o cenário de ubiquidade associado. Para a realização desta atividade, você contará com o apoio de uma técnica de apoio à verificação de requisitos de ubiquidade. Ela **deverá** ser utilizada no apoio à identificação de defeitos associados aos requisitos de ubiquidade do projeto. Os defeitos identificados deverão ser reportados no formulário de relato das discrepâncias.

Lembre-se de registrar o esforço em tempo investido nestas atividades e de preencher as informações que constam neste formulário e nos formulários UB_2 e UB_3.

Exercício

Os passos a serem seguidos são:

13. Para realizar este exercício relembre que você deve:

f. Ler a documentação do projeto.

g. Ler o cenário de ubiquidade a ser considerado.

h. Examinar o Guia de apoio à Verificação de Requisitos de Ubiquidade

i. Revisar os requisitos de ubiquidade do projeto

j. Relatar os defeitos identificados no relatório de discrepâncias.

14. Hora em que você começou a examinar a documentação do projeto: ___:___ (hora:minuto)
15. Hora em que você terminou de examinar a documentação do projeto: ___:___ (hora:minuto)
16. Tempo gasto examinando a documentação do projeto (sem considerar interrupções): _____ minutos
17. Hora em que você começou a examinar o Guia de apoio à verificação de Requisitos de Ubiquidade: ___:___ (hora:minuto)
18. Hora em que você terminou de examinar o Guia de apoio à verificação de Requisitos de Ubiquidade: ___:___ (hora:minuto)
19. Tempo gasto examinando o Guia de apoio à Verificação de Requisitos de Ubiquidade (sem considerar interrupções): _____ minutos
20. Tempo gasto para realizar a revisão dos requisitos de ubiquidade (sem considerar possíveis interrupções): _____ horas
21. Tempo que você levou para fazer o exercício (não considere possíveis interrupções): _____ minutos

Conclusões

O objetivo destas conclusões é estudar a evolução das opiniões dos participantes sobre o problema de verificação de requisitos de ubiquidade enquanto estava fazendo este exercício.

3. O que você aprendeu com este exercício?

- Em relação à verificação de requisitos;

- Em relação ao domínio de ubiquidade.

_____
_____
_____

4. Se você tivesse que fazer este exercício novamente, você faria coisas diferentes? Quais? Por quê?

_____
_____
_____

J.4 - Questionário de Acompanhamento

Formulário UB_3 – Avaliação da Abordagem Proposta

Nome:_____

Estas são perguntas que precisamos fazer de forma a tornar mais eficazes os dados obtidos ao longo do estudo.

1. Em relação à complexidade da tarefa realizada:

__ Eu considerei a tarefa muito complexa.
__ Eu considerei a tarefa complexa.
__ Eu considerei a tarefa simples.
__ Eu considerei a tarefa muito simples.

Que dificuldades você encontrou para realizar esta tarefa?

_____
_____

2. Em relação à sua satisfação na realização da tarefa:

__ Eu fiquei totalmente satisfeito.
__ Eu fiquei largamente satisfeito.
__ Eu fiquei parcialmente satisfeito.
__ Eu fiquei insatisfeito.

3. Descreva os procedimentos que utilizou para verificar os requisitos de ubiquidade.

_____
_____
_____

4. Quais as principais dificuldades encontradas no uso da abordagem para apoiar a verificação dos requisitos de ubiquidade?

_____
_____

5. Você sentiu falta de algum apoio adicional no uso da abordagem?

_____
_____

6. Você acredita que a abordagem proposta possa ser melhorada para tornar seu uso mais simples? Como?

_____
_____

APÊNDICE K – GUIA GERAL DE DEFINIÇÃO DE REQUISITOS DE UBIQUIDADE COM DIRECIONAMENTO

Este apêndice apresenta a versão do Guia Geral de Definição de Requisitos de Ubiquidade presente em UbiCheck 2.0.

K.1 – Guia Geral com Direcionamento

Captura da Experiência

Quais os usuários relevantes para o sistema?

Item de Especificação: Mapa de Atores

Orientação: Descrever os atores que interagem com o sistema.

Fatores: CE01,CE02,CE03, CE04,CE08,CE09, CE10

Como a interação do usuário é considerada?

Item de Especificação: Requisito Funcional

Orientação: Descrever o conjunto de interações do usuário com o sistema que é considerado na captura de experiência.

Fatores: CE01,CE02,CE04

Como a informação da interação do usuário é considerada?

Item de Especificação: Requisito Funcional

Orientação: Definir as informações de interação do usuário que devem ser consideradas pelo sistema.

Fatores: CE01,CE02

Como capturar informações da interação do usuário?

Item de Especificação: Caso de Uso

Orientação: Descrever como o sistema deve proceder para capturar as informações de interação do usuário.

Fatores: CE01,CE03

Como armazenar informações da interação do usuário?

Item de Especificação: Passo de Caso de Uso

Orientação: Descrever na captura da interação do usuário como o sistema deve proceder para guardar as informações capturadas.

Fatores: CE02

Como o padrão de interação do usuário é considerado?

Item de Especificação: Regra de Negócio

Orientação: Descrever o que caracteriza um padrão de interação do usuário.

Fatores: CE04

Como analisar o padrão de interação do usuário?

Item de Especificação: Passo de Caso de Uso

Orientação: Descrever na captura da interação do usuário como o sistema deve proceder para analisar os padrões de interação do usuário.

Fatores: CE04,CE05

Como as atividades do usuário são consideradas?

Item de Especificação: Requisito Funcional

Orientação: Definir as atividades do usuário que o sistema deve compreender na captura da experiência.

Fatores: CE08

Como a necessidade do usuário é considerada?

Item de Especificação: Regra de Negócio

Orientação: Definir as necessidades do usuário que o sistema deve considerara na captura da experiência.

Fatores: CE09

Como a preferência do usuário é considerada?

Item de Especificação: Requisito Funcional

Orientação: Definir as informações que constituem as preferências do usuário.

Fatores: CE10

Como a experiência do usuário é considerada?

Item de Especificação: Requisito Funcional

Orientação: Definir as experiências do usuário que devem ser consideradas na captura da experiência.

Fatores: CE03,CE05,CE06, CE07

Como capturar a experiência do usuário?

Item de Especificação: Caso de Uso

Orientação: Definir como o sistema captura a experiência do usuário.

Fatores: CE01,CE03

Como o relacionamento de experiências do usuário é considerado?

Item de Especificação: Regra de Negócio

Orientação: Definir o que caracteriza o relacionamento entre experiências.

Fatores: CE05

Como analisar os relacionamentos de experiências do usuário?

Item de Especificação: Caso de Uso

Orientação: Definir como o sistema analisa os relacionamentos de experiências do usuário.

Fatores: CE04,CE05

Quais experiências do usuário são consideradas experiências privadas?

Item de Especificação: Regra de Negócio

Orientação: Definir o que caracteriza uma experiência do usuário com privada.

Fatores: CE06

Quais experiências do usuário são consideradas experiências públicas?

Item de Especificação: Regra de Negócio

Orientação: Definir o que caracteriza uma experiência do usuário com pública.

Fatores: CE07

Perguntas removidas do guia, mas que podem ser úteis em outras etapas do desenvolvimento:

Como representar as atividades do usuário?

Como representar as necessidades do usuário?

Como representar as preferências do usuário?

Comportamento Adaptável

Quais as funcionalidades relevantes para o sistema?

Item de Especificação: Requisito Funcional

Orientação: Definir as funcionalidades providas pelo sistema.

Fatores: CA07,CA17

Como adaptar as funcionalidades?

Item de Especificação: Requisito Funcional

Orientação: Definir as adaptações que cada funcionalidade permite.

Fatores: CA07,CA17,CA18, CA19,CA22,CA24, CA25

Como a informação do ambiente influencia na adaptação das funcionalidades?

Item de Especificação: Regra de Negócio

Orientação: ---

Fatores: CA03,CA04,CA07, CA09,CA11

Como o contexto influencia na adaptação das funcionalidades?

Item de Especificação: Regra de Negócio

Orientação: ---

Fatores: CA17,CA19

Como o dispositivo influencia na adaptação das funcionalidades?

Item de Especificação: Regra de Negócio

Orientação: ---

Fatores: CA20,CA24

Quais as decisões relevantes para o sistema?

Item de Especificação: Requisito Funcional

Orientação: Definir as decisões que o sistema pode tomar para adaptar o seu comportamento.

Fatores: CA03

Como tomar as decisões?

Item de Especificação: Regra de Negócio

Orientação: Definir critérios que devem ser considerados para que o sistema tome uma decisão sobre como adaptar o seu comportamento.

Fatores: CA03

Como a informação do ambiente influencia na tomada de decisão?

Item de Especificação: Regra de Negócio

Orientação: ---

Fatores: CA03,CA04,CA07, CA09,CA11

Quais as aplicações relevantes para o sistema?

Item de Especificação: Requisito Funcional

Orientação: Definir a lista de aplicações que devem ter o comportamento adaptável.

Fatores: CA10,CA11

Como configurar aplicações?

Item de Especificação: Requisito Funcional

Orientação: Definir os parâmetros que devem ser utilizados para configurar cada aplicação que pode ter comportamento adaptável.

Fatores: CA10,CA11

Como a necessidade do usuário influencia na configuração das aplicações?

Item de Especificação: Regra de Negócio

Orientação: ---

Fatores: CA10

Como a informação do ambiente influencia na configuração das aplicações?

Item de Especificação: Requisito de Negócio

Orientação: ---

Fatores: CA03,CA04,CA07, CA09,CA11

Quais os conteúdos relevantes para o sistema?

Item de Especificação: Requisito Funcional

Orientação: Definir os conteúdos que podem ser alterados em vista de uma adaptação.

Fatores: CA19

Como adaptar o conteúdo?

Item de Especificação: Passo de Caso de Uso

Orientação: Definir o que ocorre quando é necessário adaptar um conteúdo.

Fatores: CA07,CA17,CA18, CA19,CA22,CA24, CA25

Como a informação do ambiente influencia na adaptação de conteúdo?

Item de Especificação: Regra de Negócio

Orientação: ---

Fatores: CA03,CA04,CA07, CA09,CA11

Como o contexto influencia na adaptação do conteúdo?

Item de Especificação: Regra de Negócio

Orientação: ---

Fatores: CA17,CA19

Como o dispositivo influencia na adaptação do conteúdo?

Item de Especificação: Regra de Negócio

Orientação: ---

Fatores: CA20,CA24

Quais as informações do ambiente relevantes para o sistema?

Item de Especificação: Requisito Funcional

Orientação: Definir as informações do ambiente que podem influenciar uma adaptação.

Fatores: CA03,CA04,CA07, CA09,CA11

Como analisar as informações do ambiente?

Item de Especificação: Caso de Uso

Orientação: Definir como o sistema deve analisar as informações do ambiente.

Fatores: CA04

Quais os ambientes relevantes para o sistema?

Item de Especificação: Requisito Funcional

Orientação: Definir os ambientes em que o sistema opera.

Fatores: CA03,CA04,CA06, CA07,09,CA11

Como a informação do ambiente é considerada?

Item de Especificação: Requisito Funcional

Orientação: Definir as informações do ambiente que podem influenciar uma adaptação.

Fatores: CA03,CA04,CA07, CA09,CA11

Como analisar as informações do ambiente?

Item de Especificação: Passo de Caso de Uso

Orientação: Definir como o sistema deve analisar as informações do ambiente.

Fatores: CA04

Quais os serviços relevantes para o sistema?

Item de Especificação: Requisito Funcional

Orientação: Definir quais os serviços fornecidos e utilizados pelo sistema.

Fatores: CA12,CA25

Como adaptar serviços?

Item de Especificação: Requisito Funcional

Orientação: Definir para cada serviço, as adaptações que podem ser realizadas.

Fatores: CA07,CA17,CA18, CA19,CA22,CA24, CA25

Como as informações do ambiente influenciam na adaptação do serviço?

Item de Especificação: Regra de Negócio

Orientação: ---

Fatores: CA03,CA04,CA07, CA09,CA11

Como o contexto influencia a adaptação do serviço?

Item de Especificação: Regra de Negócio

Orientação: ---

Fatores: CA17,CA19

Como o dispositivo influencia a adaptação do serviço?

Item de Especificação: Regra de Negócio

Orientação: ---

Fatores: CA20,CA24

Quais os usuários relevantes para o sistema?

Item de Especificação: Mapa de Atores

Orientação: Definir os atores relevantes para o sistema.

Fatores: CA08,CA10,CA18

Como a preferência do usuário é considerada?

Item de Especificação: Requisito Funcional

Orientação: Definir o conjunto de informações que o usuário pode informar para o sistema.

Fatores: CA08

Como a necessidade do usuário é considerada?

Item de Especificação: Regra de Negócio

Orientação: Definir o que caracteriza uma necessidade do usuário.

Fatores: CA10

Como a seção do usuário é considerada?

Item de Especificação: Requisito Funcional

Orientação: Definir as informações que constituem a seção do usuário.

Fatores: CA18

Como adaptar a seção do usuário?

Item de Especificação: Passo de Caso de Uso

Orientação: Definir como o sistema adapta a seção do usuário.

Fatores: CA07,CA17,CA18, CA19,CA22,CA24, CA25

Como a informação do ambiente influencia na adaptação da seção do usuário?

Item de Especificação: Regra de Negócio

Orientação: ---

Fatores: CA03,CA04,CA07, CA09,CA11

Como o contexto influencia na adaptação da seção do usuário?

Item de Especificação: Regra de Negócio

Orientação: ---

Fatores: CA17,CA19

Como o dispositivo influencia na adaptação da seção do usuário?

Item de Especificação: Regra de Negócio

Orientação: ---

Fatores: CA20,CA24

Quais as adaptações relevantes para o sistema?

Item de Especificação: Requisito Funcional

Orientação: Definir as adaptações previstas para o sistema.

Fatores: CA01,CA02,CA05, CA08,CA09,CA14, CA23

Como a informação da adaptação é considerada?

Item de Especificação: Requisito Funcional

Orientação: Definir as informações que caracterizam uma adaptação.

Fatores: CA01,CA02

Como prover informações sobre adaptações?

Item de Especificação: Passo de Caso de Uso

Orientação: Definir como o sistema captura as informações sobre adaptações realizadas.

Fatores: CA02

Como a alternativa de adaptação é considerada?

Item de Especificação: Fluxo Alternativo de Caso de Uso

Orientação: Definir como o sistema realiza adaptações alternativas quando não consegue realizar uma adaptação.

Fatores: CA05

Como selecionar a alternativa de adaptação?

Item de Especificação: Regra de Negócio

Orientação: Definir os critérios para selecionar a adaptação que será realizada.

Fatores: CA05

Como gerenciar a adaptação?

Item de Especificação: Caso de Uso

Orientação: Definir como o sistema gerencia as adaptações.

Fatores: CA08,CA09,CA13, CA14

Como as preferências do usuário influenciam a gerência das adaptações?

Item de Especificação: Regra de Negócio

Orientação: ---

Fatores: CA08

Como as informações do ambiente influenciam a gerência das adaptações?

Item de Especificação: Regra de Negócio

Orientação: ---

Fatores: CA03,CA04,CA07, CA09,CA11

Como o impacto da adaptação influencia a gerência das adaptações?

Item de Especificação: Passo de Caso de Uso

Orientação: Definir como o sistema gerencia uma adaptação que pode gerar impacto.

Fatores: CA14

Como o impacto da adaptação é considerado?

Item de Especificação: Requisito Funcional

Orientação: Definir os impactos previstos para cada adaptação que o sistema pode realizar.

Fatores: CA14

Como minimizar as adaptações?

Item de Especificação: Regra de Negócio

Orientação: Definir limites aceitáveis para o sistema aguardar antes de necessitar realizar uma adaptação.

Fatores: CA23

Quais os recursos relevantes para o sistema?

Item de Especificação: Requisito Funcional

Orientação: Definir os recursos que devem ser monitorados pelo sistema com o intuito de disparar uma adaptação.

Fatores: CA13,CA15,CA16

Perguntas removidas do guia, mas que podem ser úteis em outras etapas do desenvolvimento:

Quais os códigos fontes relevantes para o sistema?

Como executar códigos-fontes?

Como o dispositivo influencia na execução do código-fonte?

Como o conteúdo influencia na execução do código fonte?

Quais as interfaces gráficas relevantes para o sistema?

Como adaptar a interface gráfica?

Como as informações do ambiente influenciam na adaptação da interface gráfica?

Como o contexto influencia na adaptação da interface gráfica?

Como o dispositivo influencia na adaptação da interface gráfica?

Como executar serviços?

Como o dispositivo influencia a execução de serviços?

Como o conteúdo influencia a execução do serviços?

Como a disponibilidade do recurso é considerada?

Como gerenciar a disponibilidade dos recursos?

Como as preferências do usuário influenciam a gerência da disponibilidade de recursos?

Como as informações do ambiente influenciam a gerência da disponibilidade de recursos?

Como o impacto da adaptação influencia a gerência da disponibilidade de recursos?

Como desalocar recursos?

Como alocar recursos?

Onipresença de Serviços

Quais os serviços relevantes para o sistema?

Item de Especificação: Requisito Funcional

Orientação: Definir uma lista de serviços providos pelo sistema e descrever suas principais funcionalidades.

Fatores: OS02,OS04,OS05, OS06,OS07,OS08

Como desalocar os serviços?

Item de Especificação: Passo de Caso de Uso

Orientação: Definir no gerenciamento do serviço o que deve ser considerado quando um novo serviço é desalocado.

Fatores: OS02

Como alocar os serviços?

Item de Especificação: Passo de Caso de Uso

Orientação: Definir no gerenciamento do serviço o que deve ser considerado quando um novo serviço é alocado.

Fatores: OS04

Como gerenciar os serviços?

Item de Especificação: Caso de Uso

Orientação: Definir como o sistema deve proceder para administrar um serviço.

Fatores: OS01, OS05

Como organizar os serviços?

Item de Especificação: Regra de Negócio

Orientação: Definir critérios que devem ser considerados na organização dos serviços.

Fatores: OS06

Como o contexto do serviço influencia a organização dos serviços?

Item de Especificação: Requisito Funcional

Orientação: Definir como a organização dos serviços pode ser afetada quando as informações do contexto do serviço são alteradas.

Fatores: OS06

Como o contexto do serviço é considerado?

Item de Especificação: Requisito Funcional

Orientação: Definir as informações que caracterizam o contexto de um serviço.

Fatores: OS06

Como divulgar os serviços?

Item de Especificação: Requisito Funcional

Orientação: Definir as informações que serão fornecidas quando o serviço for divulgado.

Fatores: OS07

Quais os usuários relevantes para o sistema?

Item de Especificação: Mapa de Atores

Orientação: Definir os atores que interagem com o sistema.

Fatores: OS1,OS03

Como a seção do usuário é considerada?

Item de Especificação: Requisito Funcional

Orientação: Definir as informações que constituem a seção do usuário.

Fatores: OS01

Como a mobilidade do usuário é considerada?

Item de Especificação: Regra de Negócio

Orientação: Definir o que ocorre com um serviço quando o usuário muda de posição geográfica.

Fatores: OS03

Perguntas removidas do guia, mas que podem ser úteis em outras etapas do desenvolvimento:

Como o container do serviço influencia a gerência de serviços?

Como o container do serviço é considerado?

Como fazer cache dos serviços?

Como gerenciar a seção do usuário?

Como o container do serviço influencia a gerência da seção do usuário?

Heterogeneidade de dispositivos.

Quais os usuários relevantes para o sistema?

Item de Especificação: Mapa de Atores

Orientação: Definir os atores que interagem com o sistema.

Fatores: HD08

Como a seção dos usuários é considerada?

Item de Especificação: Requisito Funcional

Orientação: Definir as informações que constituem a seção do usuário.

Fatores: HD08

Quais os dispositivos relevantes para o sistema?

Item de Especificação: Requisito Funcional

Orientação: Definir o conjunto de dispositivos que podem utilizar o sistema. É importante definir pelo menos os requisitos mínimos desses dispositivos.

Fatores: HD01,HD02,HD03, HD08

Como a compatibilidade dos dispositivos é considerada?

Item de Especificação: Regra de Negócio

Orientação: Definir o que é necessário para um dispositivo se comunicar com o sistema.

Fatores: HD01

Como buscar dispositivos?

Item de Especificação: Caso de Uso

Orientação: Definir como um dispositivo pode buscar outro.

Fatores: HD01

Como identificar dispositivos?

Item de Especificação: Caso de Uso

Orientação: Definir os como o sistema pode identificar um determinado dispositivo.

Fatores: HD02

Como a disponibilidade do dispositivo é considerada na identificação de dispositivos?

Item de Especificação: Fluxo de Exceção

Orientação: Definir o que sistema deve fazer caso um dispositivo fique indisponível durante a sua identificação.

Fatores: HD02

Como a disponibilidade do dispositivo é considerada?

Item de Especificação: Passo de Caso de Uso

Orientação: Definir como a disponibilidade do dispositivo é considerada durante a sua identificação.

Fatores: HD02

Quais os dispositivos são considerados dispositivos heterogêneos?

Item de Especificação: Requisito Funcional

Orientação: Definir os dispositivos heterogêneos que podem utilizar o sistema.

Fatores: HD03

Como os recursos dos dispositivos são considerados?

Item de Especificação: Requisito Funcional

Orientação: Descrever os recursos que o sistema pode acessar em cada dispositivo compatível.

Fatores: HD10

Como compartilhar recursos de dispositivos?

Item de Especificação: Caso de Uso

Orientação: Definir como o sistema pode compartilhar os recursos dos diferentes dispositivos que o acessam.

Fatores: HD10

Quais as aplicações relevantes para o sistema?

Item de Especificação: Requisito Funcional

Orientação: Definir as aplicações que devem estar disponíveis nos dispositivos para que eles possam utilizar o sistema.

Fatores: HD04,HD05,HD06, HD07,HD09

Como a versão da aplicação é considerada?

Item de Especificação: Requisito Funcional

Orientação: Definir as versões das aplicações necessárias para o dispositivo utilizar o sistema.

Fatores: HD06

Como o estado da aplicação é considerado?

Item de Especificação: Regra de Negócio

Orientação: Definir estados em que as aplicações devem se encontrar para utilizar o sistema, por exemplo, o usuário deve estar autenticado.

Fatores: HD07, HD09

Perguntas removidas do guia, mas que podem ser úteis em outras etapas do desenvolvimento:

Como gerenciar a seção dos usuários?

Como o dispositivo influencia na gerencia da seção dos usuários?

Como migrar aplicações?

Como a versão da aplicação é considerada na migração das aplicações?

Como o estado da aplicação é considerado na migração das aplicações?

Como o código da aplicação é considerado?

Como migrar código de aplicações?

Como a versão da aplicação é considerada na migração de código das aplicações?

Como o estado da aplicação é considerado na migração de código das aplicações?

Sensibilidade ao Contexto

Quais os usuários relevantes para o sistema?

Item de Especificação: Mapa de Atores

Orientação: Definir os atores que interagem com o sistema.

Fatores: SC01,SC02,SC03, SC08,SC22,SC23

Como a localização do usuário é considerada?

Item de Especificação: Regra de Negócio

Orientação: Definir o que caracteriza a localização do usuário, por exemplo, coordenada geográfica, cômodo da casa etc.

Fatores: SC02

Como identificar a localização do usuário?

Item de Especificação: Caso de Uso

Orientação: Definir como o sistema identifica a localização de um usuário.

Fatores: SC01,SC02,SC03

Como a identidade do usuário é considerada?

Item de Especificação: Regra de Negócio

Orientação: Definir os atributos que identificam um usuário.

Fatores: SC01

Como identificar a identidade do usuário?

Item de Especificação: Passo de Caso de Uso

Orientação: Definir na identificação da localização do usuário, como aquele usuário é identificado.

Fatores: SC01,SC02,SC03

Como a atividade do usuário é considerada?

Item de Especificação: Requisito Funcional

Orientação: Definir o conjunto de atividades do usuário que o sistema deve monitorar.

Fatores: SC03

Como as informações do usuário são consideradas?

Item de Especificação: Requisito Funcional

Orientação: Definir o conjunto de informações do usuário que o sistema deve observar.

Fatores: SC08

Como as preferências do usuário são consideradas?

Item de Especificação: Requisito Funcional

Orientação: Definir o conjunto de informações que o usuário pode informar para o sistema.

Fatores: SC22,SC23

Quais as informações de contexto relevantes para o sistema?

Item de Especificação: Requisito Funcional

Orientação: Definir o conjunto de informações de contexto que o sistema deve observar.

Fatores: SC04, SC05, SC06, SC07, SC08, SC11, SC12, SC13, SC14, SC15, SC16, SC17, SC18, SC19, SC20, SC22, SC23, SC24, SC25, SC26

Como a temporalidade das informações de contexto é considerada?

Item de Especificação: Requisito Funcional

Orientação: Definir as informações que devem considerar a temporalidade.

Fatores: SC04

Quais informações de contexto são consideradas informações de sistema?

Item de Especificação: Requisito Funcional

Orientação: Definir o conjunto de informações de contexto provenientes de sistemas.

Fatores: SC05

Quais informações de contexto são consideradas informações de infra-estrutura?

Item de Especificação: Requisito Funcional

Orientação: Definir o conjunto de informações de contexto provenientes da infra-estrutura.

Fatores: SC06

Quais informações de contexto são consideradas informações físicas?

Item de Especificação: Requisito Funcional

Orientação: Definir o conjunto de informações de contexto provenientes do ambiente.

Fatores: SC07

Quais informações de contexto são consideradas informações do usuário?

Item de Especificação: Requisito Funcional

Orientação: Definir o conjunto de informações de contexto provenientes do usuário.

Fatores: SC08

Como contextualizar informações de contexto?

Item de Especificação: Passo de Caso de Uso

Orientação: Definir como a informação de contexto pode ser contextualizada durante a sua captura.

Fatores: SC11,SC22

Como a preferência do usuário influencia na contextualização das informações de contexto?

Item de Especificação: Regra de Negócio

Orientação: Definir como a preferência de um usuário pode mudar a contextualização de uma informação de contexto.

Fatores: SC22, SC23

Como armazenar informações de contexto?

Item de Especificação: Passo de Caso de Uso

Orientação: Definir como a informação do contexto capturada pode ser armazenada.

Fatores: SC12

Como a utilidade da informação de contexto influencia o armazenamento de informações de contexto?

Item de Especificação: Regra de Negócio

Orientação: Definir um critério para as informações serem armazenadas de acordo com a sua utilidade.

Fatores: SC12,SC13

Como a utilidade das informações de contexto é considerada?

Item de Especificação: Regra de Negócio

Orientação: Definir o que caracteriza uma informação ser util.

Fatores: SC12,SC13

Como avaliar a utilidade das informações de contexto?

Item de Especificação: Passo de Caso de Uso

Orientação: Definir na captura da informação de contexto, como ela deve ser avaliada no que diz respeito a sua utilidade.

Fatores: SC13

Como consolidar as informações de contexto?

Item de Especificação: Passo de Caso de Uso

Orientação: Definir na captura da informação de contexto, como elas podem ser consolidadas para reduzir a heterogeneidade das informações.

Fatores: SC14

Como a fonte de dados influencia a consolidação de informações de contexto?

Item de Especificação: Regra de Negócio

Orientação: Definir um critério para consolidar informações de acordo com a fonte de dados.

Fatores: SC10,SC14,SC17

Como categorizar as informações de contexto?

Item de Especificação: Requisito Funcional

Orientação: Definir as categorias em que as informações de contexto podem ser agrupadas e os critérios para esse agrupamento.

Fatores: SC17

Como a fonte de dados é considerada na categorização das informações de contexto?

Item de Especificação: Regra de Negócio

Orientação: Definir critérios para categorizar informações de contexto que considerem a fonte de dados.

Fatores: SC10,SC14,SC17

Como transformar as informações de contexto?

Item de Especificação: Regra de Negócio

Orientação: Definir critérios para transformar informações de contexto em outras.

Fatores: SC21

Como personalizar as informações de contexto?

Item de Especificação: Requisito Funcional

Orientação: Definir as informações de contexto que podem ser personalizadas.

Fatores: SC23

Como a preferência do usuário influencia na personalização das informações de contexto?

Item de Especificação: Regra de Negócio

Orientação: ---

Fatores: SC22,SC23

Como os dados das informações de contexto são considerados?

Item de Especificação: Requisito Funcional

Orientação: Definir os dados que compõem cada informação de contexto trabalhada pelo sistema.

Fatores: SC25

Como compartilhar as informações de contexto?

Item de Especificação: Caso de Uso

Orientação: Definir um fluxo para compartilhar o sistema compartilhar informações de contexto.

Fatores: SC26

Quais as fontes de dados relevantes para o sistema?

Item de Especificação: Requisitos Funcional

Orientação: Definir as fontes de dados que serão utilizadas pelo sistema.

Fatores: SC10,SC14,SC17

Quais fontes de dados são consideradas sensores?

Item de Especificação: Requisito Funcional

Orientação: Definir os sensores que serão utilizados pelo sistema.

Fatores: SC09,SC10

Perguntas removidas do guia, mas que podem ser úteis em outras etapas do desenvolvimento:

Como integrar as informações de contexto?

Como representar as informações de contexto?

Como a semântica das informações de contexto é considerada?

Como organizar a semântica das informações de contexto?

Como derivar as informações de contexto?

Como interpretar as informações de contexto?

Como a semântica influencia na interpretação das informações de contexto?