COPPE/UFRJ

UMA ABORDAGEM PARA APOIAR ESPECIFICAÇÃO DE REQUISITOS PARA PROJETOS DE SOFTWARE UBÍQUO

Felipe Curty do Rego Pinto

Dissertação de Mestrado 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 Mestre em Engenharia de Sistemas e Computação.

Orientador: Guilherme Horta Travassos

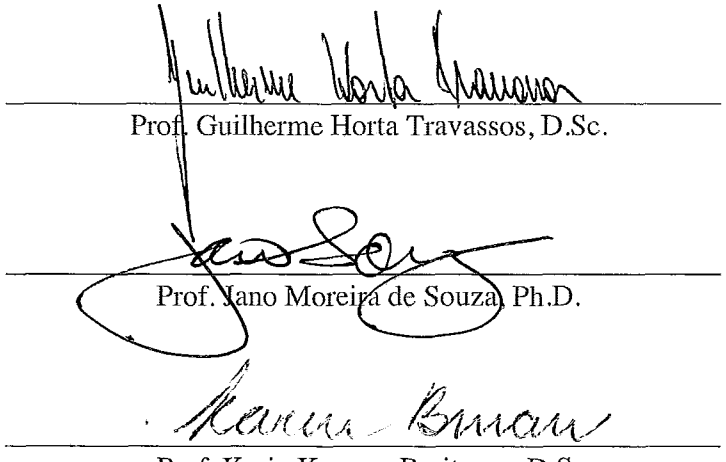
Rio de Janeiro
Setembro de 2009

UMA ABORDAGEM PARA APOIAR ESPECIFICAÇÃO DE REQUISITOS PARA PROJETOS DE SOFTWARE UBÍQUO

Felipe Curty do Rego Pinto

DISSERTAÇÃO 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 MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:



Prof. Guilherme Horta Travassos, D.Sc.

Prof. Jano Moreira de Souza, Ph.D.

Prof. Karin Koogan Breitman, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

SETEMBRO DE 2009

Pinto, Felipe Curty do Rego

Uma Abordagem para Apoiar Especificação de Requisitos para Projetos de Software Ubíquo / Felipe Curty do Rego Pinto – Rio de Janeiro: UFRJ/COPPE, 2009.

XI, 168 p.: il.; 29,7 cm.

Orientador: Guilherme Horta Travassos

Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2009.

Referencias Bibliográficas: p. 90-95.

1. Computação Ubíqua 2. Engenharia de Requisitos. 3. 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 amores da minha vida, Gabriela e Mariana.

Sem elas, nada seria possível.

AGRADECIMENTOS

À toda a minha família, que sempre acreditou e torceu por mim.

Ao meu orientador, professor Guilherme Travassos, por todo o seu conhecimento e sua dedicação a este trabalho em todos os momentos.

Aos professores Jano Souza e Karin Breitman, por suas contribuições à minha pesquisa e por terem aceitado participar desta banca.

Um agradecimento especial a Rodrigo Spínola, que, por diversas vezes, se dedicou a minha pesquisa, me co-orientando, ainda que extra-oficialmente, nas diversas "encruzilhadas" do caminho.

Meus mais sinceros agradecimentos aos vários amigos e companheiros que me incentivaram nesta longa jornada que foi o mestrado. Aos chefes, Leonardo Cardoso, Herilmar Freire, João Castellani e Carlos Octávio que me permitiram realizá-lo, dando mais um passo no meu desenvolvimento profissional.

Ao Grupo de Engenharia de Software, pela cooperação e compartilhamento de conhecimento. Em especial aos colegas Tayana Conte, Reinaldo Cabral, Jobson Massolar, Paulo Sérgio Medeiros e Arilo Neto.

Ao PESC, pelo apoio financeiro para apresentação de artigos em conferências.

E por último, mas não menos importante, a Deus, sem o qual nada seria possível.

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

UMA ABORDAGEM PARA APOIAR ESPECIFICAÇÃO DE REQUISITOS PARA PROJETOS DE SOFTWARE UBÍQUO

Felipe Curty do Rego Pinto

Setembro/2009

Orientador: Guilherme Horta Travassos

Programa: Engenharia de Sistemas e Computação

A ubiqüidade computacional é caracterizada como um novo paradigma onde computadores estão disponíveis de forma onipresente e imperceptível no ambiente do usuário. Explorar este paradigma em projetos de software permite tratar soluções de software para problemas até então inviáveis devido a suas características gerais de utilização e acesso. Contudo, existem muitos desafios no que diz respeito ao desenvolvimento deste tipo de software, principalmente considerando que os métodos, técnicas e instrumentos existentes na Engenharia de Software não foram construídos para tratar os aspectos de ubiqüidade inerente a este novo domínio de solução para o software. Nesse sentido, com vistas a facilitar e melhorar a qualidade de projetos de software ubíquos esta dissertação apresenta *UbiCheck*: uma abordagem para apoiar a definição de requisitos de ubiqüidade, que fornece um guia que conduz o desenvolvedor e direciona a sua atenção para as informações que devem ser capturadas nos requisitos de software. O processo de elaboração de *UbiCheck*, uma prova de conceito e instruções de aplicação da técnica são também descritos neste trabalho.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

AN APPROACH TO HELP REQUIREMENT DEFINITION IN UBIQUITOUS SOFTWARE PROJECTS.

Felipe Curty do Rego Pinto

September/2009

Advisor: Guilherme Horta Travassos

Department: System and Computer Engineering

The ubiquitous computing can be characterized as a new paradigm where computers are available and at the same time invisible in the user environment. Exploring this paradigm in software projects can allow software solutions for previously unfeasible problems due to their general use and access characteristics. However, there are many challenges concerned with the development of ubiquitous software. In general, the current methods, techniques and instruments available in software engineering were not built to address the features of ubiquity inherent in this new software domain. Aiming at to propose support to improve the quality of ubiquitous software projects, this dissertation presents *UbiCheck*: an approach to support ubiquitous requirements definition, which provides a guide to lead the developer and direct his attention to the information that must be captured in the software requirements. The development process of *UbiCheck*, a proof of concept and instructions to its application in ubiquitous software projects also described in this work.

Sumário

Índice de Figuras

Índice de Tabelas

Capítulo 1 - Introdução

Este capítulo apresenta a computação ubíqua, bem como as questões que levaram a realização deste trabalho. Em seguida é apresentada também a proposta e a organização dessa dissertação.

1.1. Apresentação

A computação ubíqua segundo WEISER (1991) é a inserção onipresente de computadores no ambiente para apoiar as atividades cotidianas do usuário. A essência dessa visão é tornar o computador disponível e ao mesmo tempo imperceptível para o usuário (NIEMELA e LATVAKOSKI, 2004).

De acordo com ABOWD (1999), para que essa visão seja possível, é necessário mudar a forma como os usuários utilizam seus computadores. Nesse sentido, é preciso fazer uso de dispositivos com softwares embarcados para habilitar maneiras mais simples para interação com o usuário.

Alguns autores (WEISER, 1991; KRIKKE, 2005) consideram a computação ubíqua a evolução para um novo paradigma. Conforme ilustra a figura 1, essa evolução pode ser descrita considerando três momentos:

- *mainframe*: caracterizado pelo compartilhamento de um computador de grande porte por diversos usuários;
- computador pessoal: caracterizado pelo uso individual do computador, sem

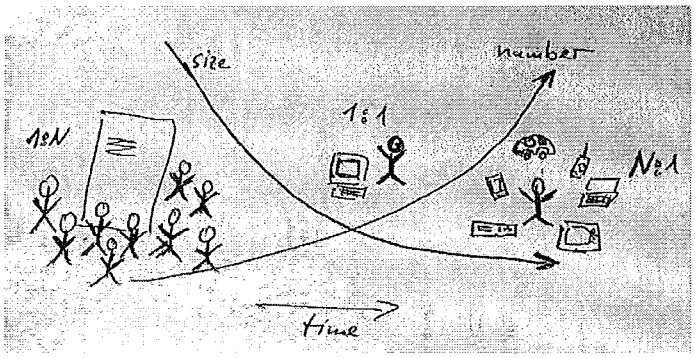


Figura 1: Paradigmas da computação (WEISER, 1991)

haver necessidade de compartilhar recursos com outros usuários; e

- computação ubíqua: caracterizado pelo uso individual de diversos dispositivos interconectados e com processamento limitado.

Em 1991, quando WEISER começou a vislumbrar esse novo paradigma, a tecnologia necessária simplesmente não existia. Segundo SATYANARAYANAN (2001), com o avanço da ciência, muitos dos elementos críticos previstos para a computação ubíqua já estão disponíveis. Rede sem fio, RFID[1] e PDAs[2] são alguns exemplos.

A inserção desses e de outros dispositivos na computação permitem que aplicações ubíquas percebam o contexto em que o usuário está inserido e se adaptem para prover serviços adequados para essas circunstâncias (COUTAZ *et al.*, 2005). Nesse sentido, a computação ubíqua pode ser aplicada em cenários em que a computação tradicional não era adequada.

Contudo, ao mesmo tempo em que a computação ubíqua pode ser mais abrangente que a tradicional, ela pode envolver diversas frentes de pesquisa, como, por exemplo, redes de computadores, inteligência artificial e processamento de sinais (RUSSEL *et al,* 2005).

Essa multidisciplinaridade pode dificultar o desenvolvimento de aplicações ubíquas, pois características antes não consideradas passam a ter papel importante na computação ubíqua, como é o caso, por exemplo, da sensibilidade ao contexto e da onipresença do serviço (SPINOLA, 2008).

Embora a computação ubíqua tenha produzido resultados interessantes no campo de dispositivos, infra-estrutura e aplicações, existem desafios tecnológicos, sociais, legais e econômicos que aumentam a complexidade do desenvolvimento do software ubíquo (DAVIES e GELLERSEN, 2002). Adicionalmente, de acordo com SAKAMURA (2006), existe pouco apoio para o desenvolvimento deste tipo de software, o que pode tornar a sua construção um problema ainda mais sério.

[1] Radio-frequency identification (RDID) é o uso de um cartão para prover identificação e rastreamento através do uso de ondas de radio.
[2] Personal Digital Assistant (PDA) são computadores de mão.

1.2. Computação Ubíqua e Engenharia de Software

Segundo DUCATEL *et al.* (2003), o software ubíquo apresenta características que normalmente não são consideradas em abordagens tradicionais de apoio ao desenvolvimento de software. Possivelmente, o uso dessas abordagens em softwares ubíquos deve reduzir a sua eficiência e/ou eficácia.

No contexto da engenharia de software, é interessante entender como essas características podem influenciar no desenvolvimento de software e, conseqüentemente, como elas podem ser consideradas em abordagens de apoio. Para compreender essas questões, é necessário entender como é o ciclo de vida de um software e qual o tipo de apoio necessário em cada etapa do desenvolvimento.

Segundo PFLEEGER (2004), o desenvolvimento de software envolve os seguintes estágios: análise e definição de requisitos, projeto do sistema, projeto do programa, programação, teste de unidades, teste de integração, teste do sistema, entrega do sistema e manutenção.

Em cada um desses estágios, as preocupações e o tipo de apoio necessário são bem diferentes, por exemplo, na definição de requisitos o analista está preocupado em identificar e documentar as necessidades do usuário, já na programação, ele precisa definir padrões, algoritmos e procedimentos (PFLEEGER, 2004). Dessa forma, analisar a influência das características de software ubíquo em um ciclo de vida completo seria um trabalho muito extenso, portanto, inviável no contexto dessa dissertação. Sendo assim, optou-se por focar na definição de requisitos de software. Parte dessa escolha foi motivada pelos resultados dos estudos descritos a seguir.

BOEHM e BASILI (2001) realizaram um estudo e verificaram que o custo para encontrar e corrigir defeitos é menor nas fases iniciais do desenvolvimento de software. Esse custo aumenta exponencialmente conforme o processo avança. O retrabalho aumenta de forma similar – defeitos inseridos ao longo do desenvolvimento são responsáveis por até 50% do esforço gasto por uma equipe de projeto em retrabalho.

Em outro estudo com 20 empresas de desenvolvimento de software, HALL *et al.* (2002) constataram que 48% dos problemas ocorriam na fase de análise dos requisitos. 37% decorrentes de problemas em processos, principalmente no que tange a correta identificação e definição de requisitos.

Embora esses estudos tenham sido realizados com softwares tradicionais, é razoável esperar que os resultados sejam semelhantes com softwares ubíquos. Adicionalmente, os requisitos de software desempenham um papel importante no desenvolvimento de software e representam os interesses dos diferentes *stakeholders*: usuários, gerentes, engenheiros de software etc. (MAFRA, 2006).

No que se refere à computação ubíqua, este trabalho está interessado em identificar e definir requisitos relacionados às características anteriormente definidas. Neste trabalho, eles foram chamados de requisitos de ubiqüidade. Logo, cabe identificar o que muda em técnicas que apóiam a definição deste tipo de requisito.

De acordo com o modelo proposto por JIANG e EBERLEIN (2007), a escolha da abordagem correta para apoiar a definição de requisitos considera diversos aspectos de software. Dentre estes aspectos estão a complexidade e o tipo do projeto, o que reforça a idéia de que pode ser mais adequado utilizar metodologias que compreendam as necessidades provenientes das características de ubiqüidade.

De fato, OLIVEIRA *et al.* (2000) já haviam relatado que abordagens de apoio que consideram um domínio específico podem mostrar conceitos, descrições e relações que podem facilitar a identificação e descrição de requisitos de software. Sendo assim, é esperado que abordagens de apoio a definição de requisitos que considerem as características de ubiqüidade contribuam com esse tipo de informação para melhorar a qualidade do software.

1.3. Motivação

Em 1991, WEISER (1991) vislumbrou um cenário onde usuários utilizariam a computação para apoiar suas atividades mais básicas. Nesse sentido, os dispositivos computacionais estariam imersos no ambiente e seu uso seria tão simples que o usuário se quer os perceberia. Para ajudar no entendimento do que é um software ubíquo, a seguir é fornecido um cenário de uso:

"Bom dia! O café está esquentando e seu jornal eletrônico já está disponível – uma voz agradável vinda do refrigerador o cumprimenta enquanto você entra na cozinha. Quando você senta no carro, o assento, os espelhos e o cinto de segurança são automaticamente ajustados - seu filho usou o carro na noite anterior. Na hora do almoço, enquanto você caminha no shopping, mensagens de restaurantes, localizados a

menos de 200 metros e que servem sua comida favorita, são enviadas para o seu celular" (LOKE, 2006, FERNANDES, 2009).

Contudo, um cenário como este envolve um complexo ambiente de software distribuído e o desenvolvimento deste tipo de software ainda é um problema que precisa ser melhor endereçado pela comunidade de Engenharia de Software (SAKAMURA, 2006).

Sendo assim, surgem dúvidas sobre como garantir a qualidade de um software ubíquo, o que remete a questões que motivaram a elaboração deste trabalho:

- Como reduzir os riscos associados ao desenvolvimento de software ubíquo e como garantir a qualidade deste software?
- Como capturar os requisitos em projetos de desenvolvimento de software ubíquos e como garantir que eles atendem as necessidades do seu usuário?
- Como melhorar a eficiência e a eficácia do desenvolvimento de software ubíquo no que diz respeito à qualidade do produto?

1.4. Objetivo

Tendo em vista os problemas relatados nas seções anteriores e a importância dos requisitos em um projeto de software, este trabalho tem como principal objetivo prover apoio adequado para a definição de requisitos de ubiqüidade em projetos de software.

Nesse sentido, foi elaborada uma abordagem, denominada *UbiCheck* (PINTO *et al.*, 2008; SPINOLA *et al.,* 2009), que procura orientar o engenheiro de software em relação as informações importantes que devem ser capturadas na etapa de definição de requisitos. Para que esse apoio seja fornecido, *UbiCheck* permite:

- a organização e a formalização das características próprias da computação ubíqua em modelos, para que possam prover informações explícitas sobre os conceitos e relações presentes neste domínio;
- a elaboração de um *Guia para Definição de Requisitos de Ubiqüidade* para orientar o desenvolvedor durante a definição de requisitos de ubiqüidade, apontando as informações que devem ser capturadas e registradas no documento de requisitos do software; e

- a configuração do guia para permitir que as informações capturadas nos requisitos possam ser adequadas às necessidades específicas de um projeto de software ubíquo.

Com vistas a facilitar o uso de *UbiCheck,* também foi elaborada uma infra-estrutura computacional para automatizar algumas das atividades da abordagem. Essa infra-estrutura é composta por duas aplicações: (a) uma para preparar o *Guia para Definição de Requisitos de Ubiqüidade*; e (b) outra para apoiar a sua aplicação em projetos de software ubíquo.

Na próxima seção, são apresentados os artigos publicados durante a elaboração deste trabalho e em seguida é descrito como esse trabalho foi organizado.

1.5. Publicações

Ao longo da elaboração deste trabalho foram publicados três artigos, os quais são listados a seguir:

- PINTO, F. C. D. R.; SPINOLA, R. & TRAVASSOS, G. H. **Abordagem para Apoiar a Definição de Requisitos de Software Ubíquo**, II Workshop on Pervasive and Ubiquitous Computing (WPUC), 2008
- SPÍNOLA, R. O.; PINTO, F. C. R. & TRAVASSOS, G. H. **Supporting Requirements Definition and Quality Assurance in Ubiquitous Software Project**, Leveraging Applications of Formal Methods, Verification and Validation Third International Symposium (ISOLA), Springer, 2008, 17, pp. 587-603
- SPÍNOLA, R. O.; PINTO, F. C. R & TRAVASSOS, G. H. **Apoio às Atividades de Especificação e Verificação de Requisitos Funcionais de Ubiqüidade em Projetos de Software**, 7th International Information and Telecommunication Technologies Symposium, 2008
- SPÍNOLA, R. O.; PINTO, F. C. R. & TRAVASSOS, G. H. **UbiCheck: An Approach to Support Requirements Definition in the Ubicomp Domain**, 25th Symposium On Applied Computing (SAC), Track on Requirement Engineering, 2009

1.6. Organização da Dissertação

Este trabalho está organizado em oito capítulos. No **segundo capítulo** são apresentadas características que podem estar presentes em software ubíquo. Elas são analisadas com o intuito de entender como elas podem influenciar o desenvolvimento de software, especificamente no que diz respeito à definição de requisitos. Nesse sentido foi realizada uma revisão da literatura para identificar como essas características são consideradas nas abordagens para apoiar a definição de requisitos de ubiqüidade.

No **terceiro capítulo** é apresentado como essas características de ubiqüidade podem ser formalizadas em modelos de ubiqüidade. Sendo assim, elas foram organizadas em modelos que apresentam os seus principais conceitos e relações.

Finalmente, **no quarto capítulo,** *UbiCheck* é apresentada como uma abordagem para apoiar a definição de requisitos de ubiqüidade. Ela explora as características de ubiqüidade apresentadas e os modelos elaborados para construir um *Guia para Definição de Requisitos de Ubiqüidade* para orientar o desenvolvedor nessa etapa do desenvolvimento.

No **quinto capítulo** é apresentado um estudo com o intuito de caracterizar o uso de *UbiCheck* no que diz respeito a aplicabilidade do *Guia para Definição de Requisitos de Ubiqüidade* em projetos de software ubíquo. Neste estudo foi verificado que *UbiCheck* pode ser aplicada em projetos de software ubíquo.

Contudo, algumas limitações foram encontradas. Sendo assim, no **sexto capítulo** são apresentadas melhorias em *UbiCheck* para compensar as limitações identificadas durante a execução do estudo. A abordagem com essas melhorias foi chamada *UbiCheck* 2.0.

No **sétimo capítulo** é descrito um protótipo de infra-estrutura computacional desenvolvido para automatizar e facilitar o uso de *UbiCheck* 2.0. Dois aplicativos são apresentados: um para apoiar a configuração da abordagem e outro para auxiliar a sua aplicação em projetos de software ubíquo.

No **oitavo e último capítulo**, as contribuições e limitações deste trabalho são apresentadas, bem como são apontados os trabalhos futuros que podem ser derivados deste trabalho.

Capítulo 2 - Computação Ubíqua

Este capítulo apresenta características presentes no software ubíquo e como elas são consideradas pelas abordagens que fornecem apoio a definição de requisitos de ubiqüidade em projetos de software.

2.1. Caracterização da Computação Ubíqua

Conforme apresentado no capítulo anterior, a computação ubíqua envolve diversas áreas de pesquisa. Ao mesmo tempo em que essa multidisciplinaridade a torna bastante abrangente, permitindo que possa ser aplicada em cenários bem diferentes, é difícil definir com precisão o que é computação ubíqua.

Segundo WEISER (1991), a computação ubíqua é a inserção onipresente de computadores no ambiente para apoiar as atividades cotidianas do usuário. Como essa definição é muito genérica e não foi encontrada nenhuma definição mais concreta, uma alternativa interessante seria definir a computação ubíqua a partir de características comuns encontradas em softwares ubíquos.

Nesse sentido, SPINOLA *et al.* (2006) realizaram uma revisão sistemática da literatura com o objetivo de descobrir a presença de características comuns em software ubíquo. Essas características foram chamadas de características de ubiqüidade e representam propriedades da computação ubíqua que podem ser notadas em softwares ubíquos. Ao todo, foram identificadas dez características:

- **Onipresença do Serviço:** o sistema se desloca junto com o usuário, permitindo seu acesso independente da localização. Essa característica gera a impressão de que o sistema está em toda parte e normalmente envolve a interconexão e redução do tamanho dos dispositivos (GEER, 2006).
    - **Exemplo:** o usuário utiliza um sistema de navegação para encontrar um shopping e, ao sair do carro, o sistema continua guiando-o até a loja desejada.
- **Invisibilidade:** o usuário usa o sistema sem perceber (NIEMELA e LATVAKOSKI, 2004). Essa característica tem como objetivo habilitar formas naturais de interação com o sistema, por exemplo, gestos e vozes.

- **Exemplo:** o usuário fala o nome de um canal e a televisão automaticamente o sintoniza.

- **Sensibilidade ao Contexto:** o sistema captura informações do ambiente em que está inserido e de acordo com elas, altera o seu comportamento. (SAKAMURA, 2006).
  - **Exemplo:** o usuário acende o forno para preparar uma pizza e o sistema, ao perceber que a pizza está pronta, desliga o forno.

- **Comportamento Adaptável:** para prover uma funcionalidade, o sistema se adapta ao ambiente em que está inserido (SAKAMURA, 2006).
  - **Exemplo:** durante uma conferência, ao identificar que a largura de banda da conexão diminuiu, o sistema reduz a qualidade da transmissão para não perder a comunicação.

- **Captura da Experiência:** o sistema acompanha e registra atividades de seus usuários e, em momento oportuno, utiliza esse conhecimento para apoiar atividades do usuário (KOGURE *et al.* 2003).
  - **Exemplo:** o usuário acorda todos os dias e prepara seu café, o sistema, ao perceber essa rotina, passa a esquentar a água pela manhã.

- **Descoberta de Serviços:** o sistema é capaz de descobrir e utilizar serviços disponíveis no ambiente (FRIDAY *et al.* 2001).
  - **Exemplo:** o usuário entra em uma loja e seu celular mostra o catalogo de promoções.

- **Composição de Funcionalidades:** o sistema fornece funcionalidades complexas a partir de funções elementares (SAKAMURA, 2006).
  - **Exemplo:** o sistema faz uma cópia de segurança utilizando as funcionalidades de compactação, criptografia e cópia remota.

- **Interoperabilidade Espontânea:** o sistema permite que dois dispositivos se comuniquem sem o intermédio do usuário (NIEMELA e LATVAKOSKI, 2004).
    - **Exemplo:** o funcionário da empresa de energia caminha pela rua enquanto seu PDA se comunica com o relógio de luz das casas próximas para emitir a conta de luz.
- **Heterogeneidade de Dispositivos:** o sistema pode ser utilizado em diferentes dispositivos (NIEMELA e LATVAKOSKI, 2004).
    - **Exemplo:** para fazer um *check-in* e embarcar em um vôo, o usuário pode utilizar um terminal de apoio, um celular ou um computador.
- **Tolerância a Falhas** – o sistema se adapta diante de falhas no ambiente e continua a prover funcionalidades essenciais (SATYANARAYANAN, 2001).
    - **Exemplo:** o sistema utiliza sensores de presença para contar pessoas em cômodos de uma casa. Ao perceber que os sensores de um quarto estão queimados, ele mantém a contagem a partir dos sensores disponíveis nos cômodos vizinhos.

Embora a identificação dessas características contribua para definir melhor o que é computação ubíqua, ainda há pouca informação sobre como elas são encontradas em um software ubíquo. Com vistas a esclarecer esse assunto, SPINOLA *et al.* (2007a) realizaram uma segunda revisão sistemática da literatura para identificar comportamentos que poderiam ocorrer em softwares ubíquos devido à presença de uma determinada característica de ubiqüidade.

Os comportamentos encontrados foram denominados fatores de ubiqüidade. Ao todo, foram encontrados 119 fatores. Cada um foi associado a uma característica de ubiqüidade e, devido ao número elevado de fatores, aqueles considerados complementares foram agrupados. A tabela 1 mostra a quantidade de fatores associados a cada característica de ubiqüidade, bem como os grupos de fatores que foram definidos para cada uma. No anexo B está disponível a lista completa de fatores de ubiqüidade.

Tabela 1: Quantidade de fatores por característica de ubiqüidade

Característica de Ubiqüidade	Quantidade de Fatores
Captura da Experiência • Grupos de Fatores: Captura de Informação, Interpretação de Informação e Gerência de Informações	7
Comportamento Adaptável • Grupos de Fatores: Gerência de Adaptações e Adaptação	23
Composição de Funcionalidade • Grupos de Fatores: Composição, Gerência de Serviços e Integração de Serviços	17
Descoberta de Serviços • Grupos de Fatores: Descoberta de Serviços, Conectar-se a Serviços e Seleção de Serviços	12
Heterogeneidade de Dispositivos • Grupos de Fatores: Busca por Dispositivos, Migração de Aplicação, e Compartilhamento de Recursos	9
Interoperabilidade Espontânea • Grupos de Fatores: Gerência de Informações e Interoperabilidade	10
Invisibilidade • Grupos de Fatores: Gerência de Entidades, Interface com Usuário e Redução do Nível de Interatividade com o Usuário	8
Onipresença de Serviço • Grupos de Fatores: Mobilidade, Gerência de Serviços, e Divulgação de Serviços	9
Sensibilidade ao Contexto • Grupos de Fatores: Captura de Informações, Controle de Sensores, Gerência de Informações de Contexto, Interpretação da Informação e Compartilhamento da Informação	21
Tolerância a Falhas • Grupos de Fatores: Falhas de Software e Falhas de Sistema	3
Total	**119**

Para auxiliar no entendimento, a seguir são apresentados exemplos de fatores da característica *Sensibilidade ao Contexto* que pertencem respectivamente aos grupos de

fatores *Captura de Informações*, *Gerência de Informações de Contexto* e *Interpretação da Informação*:

- "Considerar informações de contexto físico. Neste caso, buscam-se informações que dizem respeito à interação entre dispositivos e ambientes. Por exemplo, informações de mobilidade, localização, tempo, condições de iluminação e barulho, temperatura."
- "Armazenar as informações consideradas úteis"
- "Contextualizar e personalizar as informações capturadas de acordo com preferências do usuário"

A partir dos fatores identificados foi possível elaborar critérios mais objetivos para verificar se uma característica de ubiqüidade está presente em um software. Nesse sentido, é necessário observar se o software manifesta um comportamento semelhante ao descrito em um dos fatores dessa característica.

Uma vez que foi possível identificar a presença de características de ubiqüidade em software, foi possível atualizar a definição de software ubíquo (SPINOLA *et al.*, 2007b):

> *"A 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, para que a computação ubíqua se faça presente, é preciso que os sistemas que compõem o ambiente contemplem características de ubiqüidade, tais como: 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".*

Diante dessa definição, SPINOLA *et al.* (2008a) analisaram oito projetos de software ubíquo para verificar como se dava a presença dos fatores de ubiqüidade. Nesse sentido, para cada projeto analisado foram contabilizados quantos fatores de cada característica de ubiqüidade estavam presentes – se um projeto manifestava um comportamento descrito em um determinado fator, era considerado que esse fator estava presente no projeto. A quantidade de fatores foi utilizada para encontrar o nível de aderência de cada característica ao projeto, conforme a fórmula a seguir:

$$nível\ de\ aderência = \frac{Fc}{Ft}$$

Onde:

- *Fc* é o número de fatores da característica de ubiqüidade que estão presentes no projeto; e
- *Ft* é número de fatores da característica de ubiqüidade cujo nível de aderência está sendo calculado.

A figura 2 apresenta um gráfico com os resultados da análise. Os resultados de cada projeto foram consolidados, portanto, ela apresenta a média do nível de aderência (porcentagem de fatores contemplados) de cada característica nos projetos analisados.

É importante observar que os resultados foram pouco uniformes: enquanto as características *Sensibilidade ao Contexto* e *Invisibilidade* tiveram respectivamente a média de 36% e 28% de fatores contemplados, a característica *Composição de Funcionalidade* não foi manifestada nos projeto.

Por esse motivo, SPINOLA *et al.* (2008b) realizaram um estudo para avaliá-los quanto a sua aplicabilidade em projetos de software ubíquo e seu escopo. O objetivo deste estudo, segundo o paradigma *Goal/Question/Metric* (GQM) (BASILI e ROMBACH, 1988), foi:

Analisar	as características de ubiqüidade, seus fatores e grupos de fatores extraídos da literatura técnica
Com o propósito de	caracterizar
Em relação à	sua aplicabilidade e escopo

Do ponto de vista	de pesquisadores em engenharia de software que estejam trabalhando com pesquisa e desenvolvimento de projetos de software ubíquos
No contexto da	de projetos de software ubíquo

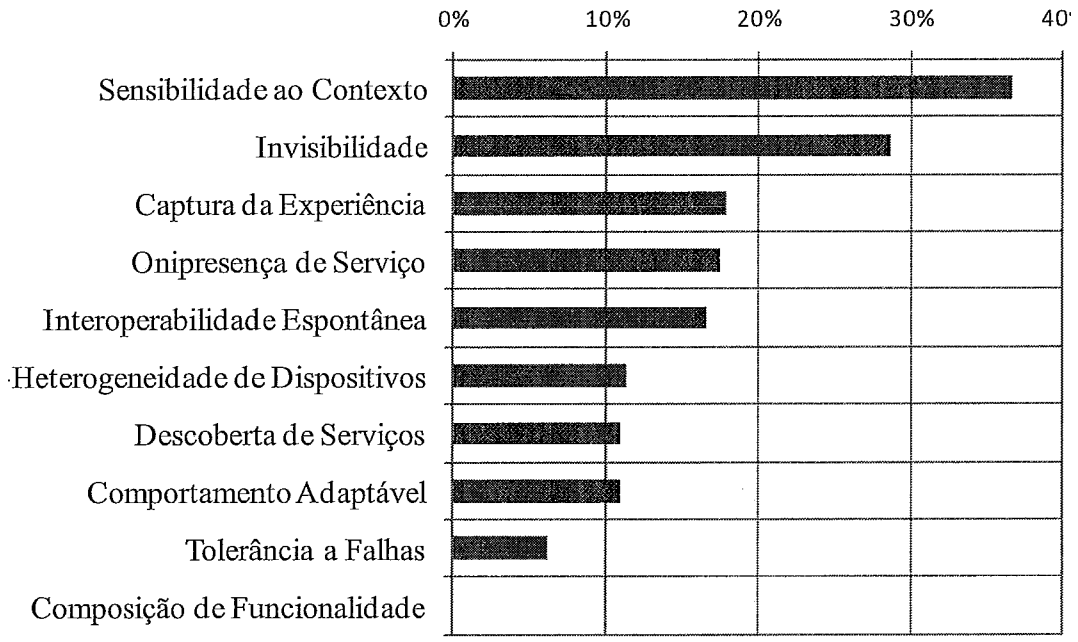


Figura 2: Nível de aderência médio das características de ubiqüidade na análise de oito projetos de software ubíquo

Esse estudo foi realizado através de um *survey*[3] com dez pesquisadores dos Grupos de Pesquisa do CNPq. Os resultados apontaram para a inclusão de três novas características de ubiqüidade:

- **Escalabilidade:** o sistema pode aumentar sua capacidade com a inserção de novos dispositivos no ambiente.
    - **Exemplo:** um sistema de consulta de preços pode permitir que mais pessoas o utilizem a partir da inserção de novos terminais de consulta.

[3] Segundo MAFRA (2006), um *survey* é um tipo de estudo experimental utilizado para capturar informações sobre uma situação. Geralmente, o *survey* é aplicado através de questionários ou entrevistas.

- **Qualidade de Serviço:** o sistema consegue manter um nível de serviço pré-acordado.
    - **Exemplo:** em uma vídeo conferência, para manter a qualidade da voz, o sistema reduz a qualidade do vídeo transmitido.
- **Privacidade e Confiança:** o sistema preserva a integridade e confidencialidade das suas informações.
    - **Exemplo:** apenas o usuário consegue alterar o seu próprio perfil.

Neste estudo, para cada característica de software ubíquo, havia pelo menos um especialista no assunto. As contribuições individuais permitiram ajustes nos fatores e a classificação das características de ubiqüidade de acordo com o aspecto funcional, conforme mostra a figura 3.

Com o resultado das revisões da literatura e do estudo, foi possível alcançar um conjunto consistente de fatores e características que podem ser utilizadas para identificar e caracterizar softwares ubíquos. Na seção seguinte, será discutido como esses fatores e características podem influenciar no desenvolvimento de software.

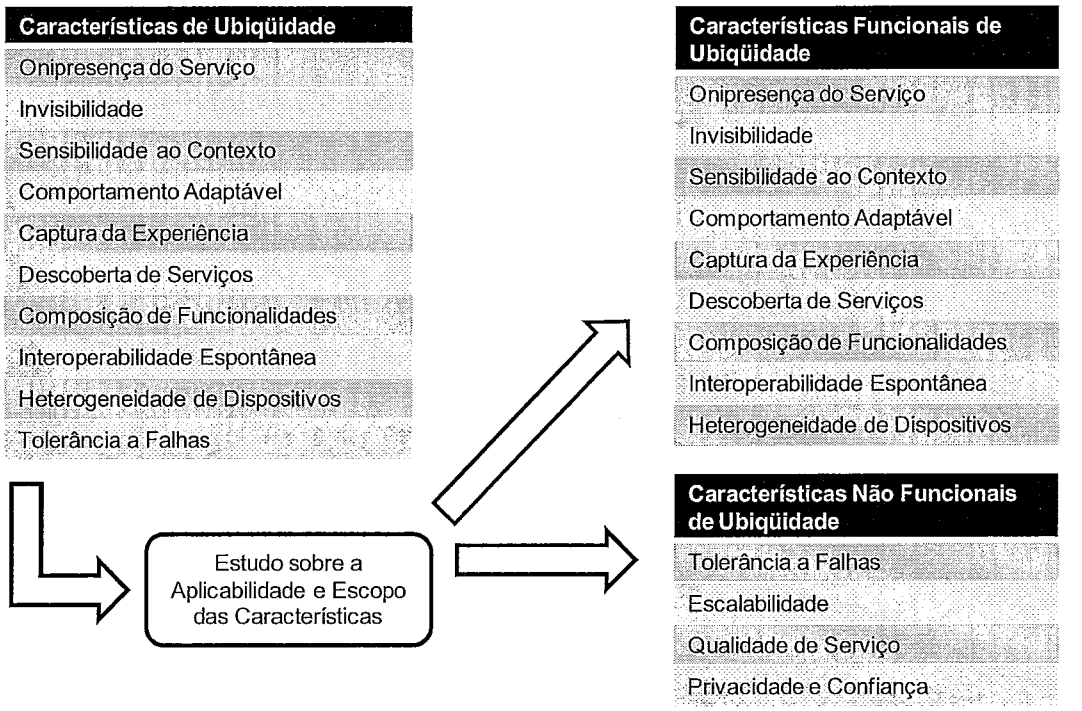


Figura 3: Características de ubiqüidade classificadas quanto ao aspecto funcional (Adaptado de SPINOLA, 2008)

2.2. Engenharia de Software aplicada a Computação Ubíqua

Na seção anterior foram identificadas características que podem estar presentes em software ubíquos. No contexto da engenharia de software, é interessante entender como essas características podem influenciar no desenvolvimento de software e, conseqüentemente, como elas podem ser consideradas em abordagens de apoio.

Nesse sentido, o primeiro passo realizado foi entender como as características de ubiqüidade eram consideradas em abordagens de apoio a definição de requisitos. Para isso, SPINOLA *et al.* (2008b) realizaram uma revisão tradicional da literatura para identificar quais eram as técnicas, metodologias, métodos e processos disponíveis sobre o assunto.

Foram encontradas seis abordagens através da pesquisa de trabalhos publicados na ACM e no IEEE:

1. os **Casos de Uso Executáveis** que exploraram o detalhamento dos requisitos a partir da simulação de casos de uso (JORGENSEN e BOSSEN, 2003);
2. a **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, 2006);
3. a **Metodologia para Definir Requisitos de Sistemas Adaptáveis e Sensíveis ao Contexto** que prove um conjunto de passos para apoiar a definição de requisitos de sistemas sensíveis ao contexto e adaptáveis (HONG *et al.*, 2005 e CHIU *et al.*, 2006 e 2007);
4. a **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 ubiqüidade (BO *et al.*, 2007);
5. o **Mecanismo para 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); e
6. o **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).

Essa revisão da literatura foi realizada no início de 2008, portanto, surgiu a desconfiança de que novos trabalhos poderiam ter surgido desde então. Como ela foi realizada com base em princípios de revisão sistemática[4], havia sido elaborado um protocolo de busca, o que permite que ela seja minimamente auditada e repetida (MAFRA e TRAVASSOS, 2005). Sendo assim, no contexto deste trabalho, ela foi repetida no início de 2009 e um novo trabalho foi encontrado (ver Anexo A para mais informações):

7. o **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).

As sete abordagens são apresentadas nas subseções a seguir. Sempre que 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. Em seguida é apresentada uma análise sobre a presença das características de ubiqüidade nestas abordagens.

2.2.1. Casos de Uso Executáveis

JORGENSEN e BOSSEN (2003) propuseram uma técnica baseada na simulação de casos de uso, denominada Casos de Uso Executáveis. Ela segue um fluxo interativo baseado em três atividades:

1. descrever os casos de uso a partir da análise do domínio;
2. modelar os casos de uso utilizando uma linguagem formal; e
3. simular os casos de uso com base nos modelos elaborados.

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.

Como as atividades são interativas, a comunicação entre o desenvolvedor e o usuário deve ser constante para que essa técnica possa ser utilizada. Conseqüentemente,

[4] Revisão sistemática é uma forma de identificar, avaliar e interpretar toda pesquisa relevante para uma questão de pesquisa (KITCHENHAN (2004).

também são reduzidos os mal entendidos entre os usuários e desenvolvedores (JORGENSEN e BOSSEN, 2003).

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 ubiqüidade e aparentemente poderia ser utilizada para apoiar outros tipos de softwares que demandassem atenção na definição dos requisitos.

2.2.2. Modelagem de Requisitos com a Linguagem KAOS

GOLDSBY e CHENG (2006) propuseram uma abordagem para definição de requisitos de sistemas dinamicamente adaptáveis (SDA). Ela é baseada nos quatro níveis de requisitos definidor por BERRY *et al.* (2005):

- nível 1: corresponde às atividades tradicionais de identificação de requisitos. Neste caso, são identificadas informações do domínio da aplicação que será construída e realizada a identificação de todas as possíveis configurações que o SDA pode atingir;

- nível 2: corresponde ao trabalho efetuado na definição do comportamento do SDA em tempo de execução para identificar quais variáveis do ambiente serão utilizadas para o sistema 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 de adaptação utilizar; e

- nível 4: corresponde ao trabalho de pesquisa com o intuito de criar novos mecanismos de adaptação.

Para apoiar em sua definição, os requisitos são mapeados nos níveis descritos acima e depois formalizados através da modelagem na linguagem KAOS (DARDENNE *et al.*, 1993).

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.

2.2.3. Metodologia para Definir Requisitos de Sistemas Adaptáveis e Sensíveis ao Contexto

HONG *et al.* (2005) e CHIU *et al.* (2006 e 2007) propuseram uma metodologia para definir requisitos de sistemas adaptáveis e sensíveis ao contexto. Ela fornece o seguinte fluxo de atividades que deve ser seguido pelo desenvolvedor:

1. determinar os grupos de usuários do sistema;
2. identificar os contextos em que estes grupos de usuário podem estar presentes;
3. para cada contexto, listar requisitos e serviços relacionados;
4. para cada serviço, determinar comportamentos sensíveis ao contexto requeridos pelo sistema;
5. para cada comportamento, detalhar as funcionalidades necessárias;
6. verificar se o conjunto de funcionalidades atende aos requisitos definidos na atividade 3;
7. projetar visões de processo que capturem os comportamentos sensíveis ao contexto;
8. projetar visões de dados com base nas visões de processos e nas funcionalidades identificadas;
9. projetar as visões das interfaces com o usuário com base na plataforma para a qual será desenvolvido o software; e
10. verificar a consistência entre as três visões projetadas anteriormente.

A metodologia proposta considera as características *Sensibilidade ao Contexto* e *Comportamento Adaptável.* No entanto, embora apóie a identificação de comportamentos (fatores) associados a essas características, não fornece os

comportamentos que são esperados em um software com essas características. Complementarmente, não foi relatado o uso dessa abordagem em nenhum projeto.

2.2.4. Definição de Requisitos do Usuário com base em uma Ontologia

BO *et al.* (2007) propuseram uma abordagem para guiar os usuários na definição de requisitos através de uma ontologia de requisitos composta por:

- uma ontologia do domínio com conceitos e relacionamentos sobre o domínio cujos requisitos serão especificados;
- um conjunto de requisitos funcionais que correspondem a comportamentos que devem ser realizados; e
- um conjunto de requisitos não funcionais que correspondem a restrições que devem ser aplicadas aos requisitos funcionais.

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.

Dessa forma, o desenvolvedor tem o mapeamento dos requisitos definidos anteriormente com um conjunto de requisitos mais simples, conseqüentemente mais fáceis de serem capturados.

Essa abordagem não compreende nenhuma característica de ubiqüidade e aparentemente poderia ser utilizada para apoiar outros tipos de softwares que demandassem atenção na definição dos requisitos.

2.2.5. Mecanismo para Definição de Requisitos de Serviços

XIANG *et al.* (2007) propuseram a abordagem denominada Mecanismo para Definição de Requisitos de Serviços, que é limitada a definição de requisitos de serviços. Através do preenchimento de um questionário, é elaborada uma ontologia de

requisitos que ajuda a capturar e acumular o conhecimento necessário para a definição dos requisitos do serviço.

Foi realizado uma prova de conceito, onde esta abordagem foi utilizada para definir requisitos de um serviço de gestão de pedidos na web. Essa abordagem não compreende nenhuma característica de ubiqüidade e aparentemente poderia ser utilizada para apoiar outros tipos de softwares que demandassem atenção na definição dos requisitos.

2.2.6. 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 dos 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.

Nesse processo são consideradas as características *Tolerância a Falhas* e *Privacidade e Confiança.* Adicionalmente, o apoio é restrito a requisitos não funcionais relacionados com segurança da informação.

2.2.7. 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 freqüência de uso do sistema;
2. observar relatos do usuário, por exemplo, *bugs* 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 ubiqüidade, o trabalho relata o uso de softwares sensíveis ao contexto para auxiliar na observação dos padrões de comportamento dos usuários.

2.2.8. Análise das Abordagens Identificadas na Literatura

As sete abordagens identificadas foram analisadas para verificar a presença e a influência das características de ubiqüidade. A tabela 2 mostra as características presentes em cada uma. Por limitação de espaço na tabela, as abordagens foram numeradas conforme descrito a seguir:

1. **Casos de Uso Executáveis** (JORGENSEN e BOSSEN, 2003);
2. **Modelagem de Requisitos com a Linguagem KAOS** (GOLDSBY e CHENG, 2006);
3. **Metodologia para Definir Requisitos de Sistemas Adaptáveis e Sensíveis ao Contexto** (HONG *et al.*, 2005 e CHIU *et al.*, 2006 e 2007);
4. **Definição de Requisitos do Usuário com Base em uma Ontologia** (BO *et al.*, 2007);
5. o **Mecanismo para Definição de Requisitos de Serviços** (XIANG *et al.*, 2007);
6. **Processo Sistemático para Análise de Requisitos de Segurança em Sistemas Embarcados** (MARKOSE *et al.*,2008);e

7. **Processo para Definição de Requisitos com Base nas Interações do Usuário** (OYAMA *et al.*, 2008 e MING *et al.*,2008).

Tabela 2: Características presentes em cada abordagem identificada na literatura

	Abordagens						
Características de Ubiqüidade	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Captura da Experiência							
Comportamento Adaptável		X	X				
Composição de Funcionalidade							
Descoberta de Serviços							
Heterogeneidade de Dispositivos							
Interoperabilidade Espontânea							
Invisibilidade							
Onipresença de Serviço							
Sensibilidade ao Contexto		X	X				X
Escalabilidade							
Privacidade e Confiança:						X	
Qualidade de Serviço							
Tolerância a Falhas						X	

Como é possível reparar, três das abordagens não contemplaram nenhuma característica de ubiqüidade. Em uma análise mais detalhada, percebeu-se que o auxílio provido por elas estava associado à simplificação dos requisitos, o que basicamente era feito através de simulações e decomposições. Dessa forma, o apoio tinha o objetivo de melhorar o entendimento dos requisitos e não de fornecer conceitos e relações que poderiam ajudar na captura dos requisitos. Aparentemente, essas abordagens poderiam ser utilizadas para apoiar a definição de requisitos de outros tipos de softwares.

Nas demais abordagens foi possível reparar que poucas características de ubiqüidade foram consideradas. Coincidentemente, a característica *Sensibilidade ao Contexto,* que havia sido mais popular na análise de projetos de software ubíquo, apresentada na seção 2.1, estava presente em três dessas quatro abordagens. Essas informações reforçam a idéia de que falta apoio para o desenvolvimento de software ubíquo e sugere que *Sensibilidade ao Contexto* pode ser é a característica mais estudada do conjunto.

2.3. Conclusão

Este capítulo apresentou 13 características de ubiqüidade que representam propriedades que podem ser notadas em softwares ubíquos. Como havia pouca informação sobre como identificá-las em softwares, foram apresentados comportamentos, denominados fatores de ubiqüidade, que poderiam ocorrer devido à presença dessas características no software. Esses fatores e características contribuíram para uma definição mais precisa do que é um software ubíquo.

Em seguida, foi analisado como essas características poderiam ser consideradas no desenvolvimento de software ubíquo. Nesse sentido foi realizada uma revisão tradicional da literatura para identificar trabalhos que apoiassem à definição de requisitos de software ubíquo e, a partir deles, investigar como as características de ubiqüidade eram tratadas.

Como nenhum dos trabalhos identificados considerava todas as características de ubiqüidade, seria interessante elaborar uma abordagem que considerasse os conceitos e relações presentes nos fatores das características de ubiqüidade. No entanto, como identificar essas informações não é simples, no capítulo seguinte será apresentada uma estratégia para formalizar essas características em modelos, de forma a permitir que suas informações sejam explicitadas.

Capítulo 3 - Modelos de Ubiqüidade

Este capítulo apresenta como as características de ubiqüidade podem ser organizadas e estruturadas em modelos que compreendem seus principais conceitos e relações.

3.1. Introdução

No capítulo anterior foram apresentadas as características de ubiqüidade consideradas neste trabalho. Para cada uma foram descritos fatores, ou seja, comportamentos que podem estar presente neste tipo de software, por exemplo:

- Categorizar contexto com base na fonte de dados de suas informações (informações de rede, dispositivo e interação do usuário);
- Compartilhar informações de contexto com usuários e outros dispositivos; e
- Considerar semântica na interpretação de informação de contexto.

Seria interessante identificar os conceitos e relações importantes dos fatores das características de ubiqüidade. Uma vez identificadas, essas informações poderiam ser explicitadas através de modelos. Essa estratégia também poderia tornar mais fácil o entendimento das características, pois suas principais informações estariam representadas graficamente.

Contudo, como os fatores apresentados no capítulo anterior foram redigidos de forma muito diferente e com informações em diferentes níveis de abstração, para modelá-los, seria interessante reescrevê-los na forma de requisitos gerais que, além de homogeneizá-los, destacassem os conceitos e relações que deveriam estar contidos nos modelos.

Sendo assim, a seção 3.2 apresenta como os fatores foram reescritos e como os conceitos e relações de cada característica de ubiqüidade foram destacadas, a seção 3.3 apresenta como eles foram modelados e, finalmente, a seção 3.4 conclui este capítulo.

3.2. Organização das Características de Ubiqüidade

Com vistas a reescrever os fatores na forma de requisitos para a elaboração dos modelos foi necessário analisar os fatores disponíveis no Anexo B para identificar os conceitos vinculados ao domínio de ubiqüidade, bem como para identificar relações entre esses conceitos.

Nesse sentido, as listagens a seguir exemplificam conceitos e relações que podem ser encontrados em seis fatores da característica *Sensibilidade ao Contexto*. Os conceitos e relações importantes foram identificados a partir da interpretação de cada fator por um especialista em ubiqüidade.

Fator 1: Tratar alto acoplamento: existe relacionamento entre informações contextuais em diferentes níveis de abstração. É preciso gerenciar estes relacionamentos

Conceitos importantes:

i. informação de contexto;

ii. relacionamento de informação de contexto; e

iii. nível de abstração do relacionamento da informação de contexto.

Relações importantes:

a. os relacionamentos das informações de contexto podem ser gerenciados.

Fator 2: Categorizar contexto com base na fonte de dados de suas informações (informações de rede, dispositivo e interação do usuário)

Conceitos importantes:

i. informação de contexto; e

ii. fonte de dados da informação de contexto.

Relações importantes:

a. as informações de contextos podem ser categorizadas, mas para isso, dependem de sua fonte de dados.

Fator 3: 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.

Conceitos importantes:

1. informação de contexto

Relações importantes:

a. a informação de contexto pode ser compartilhada

Fator 4: Considerar semântica na interpretação de informação de contexto.

Conceitos importantes:

i. informação de contexto; e

ii. semântica da informação de contexto.

Relações importantes:

a. a informação de contexto pode ser interpretada, mas para isso, depende da sua semântica.

Fator 5: Controlar os sensores (ativo/em espera).

Conceitos importantes:

i. sensores.

Relações importantes:

a. sensores podem ser controlados.

Fator 6: 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 nos dispositivos acessíveis) e outras aplicações que estão relacionadas para compor o sistema

Conceitos importantes:

i. informação de contexto

ii. informação de sistema

Relações importantes:

a. informação de sistema é um tipo de informação de contexto.

Seria interessante que essas informações pudessem ser explicitadas de alguma forma no momento em que um fator fosse redigido, pois neste momento, os conceitos importantes poderiam ser destacados pelo próprio especialista no domínio que os elaborou. Adicionalmente, a identificação prévia desses conceitos poderia facilitar a leitura dos fatores por outros pesquisadores, pois os conceitos seriam fornecidos de forma mais direta aos leitores.

Nesse sentido, foi idealizado um formato padronizado para descrever os fatores de ubiqüidade na forma de requisitos que pudessem ser utilizados para apoiar a sua modelagem. Para tal, foi investigada a possibilidade de agrupar os conceitos e relações importantes. O resultado foi a classificação dos conceitos em duas classes e das relações em quatro classes, conforme descrito a seguir:

Classes de Conceitos:

- **Entidade:** é possível perceber que alguns conceitos representam os elementos principais necessários para descrever cada fator, por exemplo, como é o caso de *informação de contexto*. A esse tipo de conceito foi dado o nome Entidade.
- **Serviço:** é possível perceber também que são descritas funcionalidades que envolvem entidades, como é o caso do *fator 5,* onde *controlar* está associado a *Sensor*. A esse tipo de conceito foi dado o nome Serviço.

Classes de Relações:

- **Funcionalidade:** um serviço associado a uma entidade fornece a idéia de que uma funcionalidade pode ser realizada, por exemplo, *controlar sensores*.
- **Dependência:** em alguns casos funcionalidades são descritas, mas há dependência de uma terceira entidade, como é o caso do *fator 4*, onde para a informação de contexto ser interpretada (funcionalidade) é necessário conhecer a sua semântica.
- **Tipo:** algumas entidades são descritas como se fossem um tipo de outra, por exemplo, no *fator 6,* informação de sistema é um tipo de informação de contexto.
- **Atributo:** algumas entidades são descritas como se fossem atributos de outras, como é o caso do *fator 1*, onde *informação de contexto* possui atributo *relacionamento* que possui atributo *nível de abstração*.

Além dessas classes, o formato proposto também deveria contemplar as informações complementares utilizadas em alguns dos fatores para fornecer exemplos e explicações mais detalhadas sobre os seus conceitos. Por exemplo, no *fator 6* há uma definição sobre *informação de sistema* e exemplos desse tipo de informação.

Como essas informações complementares apareciam em apenas alguns fatores, foi decidido que um fator poderia ser composto por duas partes. A primeira teria o objetivo de descrever os conceitos e relações importantes agrupados nas classes descritas anteriormente. A segunda parte seria opcional, teria o objetivo de descrever essas informações complementares e não seria estruturada dentro do formato padrão, portanto, sua descrição seria livre. Parte dessa decisão foi motivada pelo fato de que

essas informações complementares não serão consideradas na formalização das características de ubiqüidade, descrita na seção seguinte.

Sendo assim, foi elaborado um formato para a primeira parte do fator, feito com base em uma frase estruturada por uma regra na notação ABNF (CROCKER e OVERELL, 2008), conforme descrito a seguir:

Regras:

primeira parte do fator=	"O sistema deve" ("considerar" *atributo *entidade* / "considerar" entidade tipo / *funcionalidade* [*dependência*]).
funcionalidade =	*serviço* *atributo *entidade*
dependência =	"de acordo com" *atributo *entidade* / "a partir do uso de" *atributo *entidade*
tipo =	"do tipo" entidade
atributo =	entidade
entidade	<nome da entidade no singular>
serviço	<nome do serviço no infinitivo>

Cabe destacar que no contexto deste trabalho, os elementos gramaticais não foram considerados no formato proposto, portanto, as frases geradas devem ser corrigidas após a sua criação. Por exemplo, nas frases a seguir os trechos sublinhados foram inseridos para que elas fizessem sentido: *O sistema deve categorizar informações de contexto de acordo com sua fonte de dados; e: O sistema deve gerenciar os relacionamentos das informações de contexto.*

Na notação utilizada para descrever o formato proposto, uma regra mais complexa pode ser formada a partir de outras mais simples. Para facilitar o entendimento dessas regras, elas são explicadas a seguir:

- A **primeira parte do fator** sempre começa com "*O sistema deve*" e continua com uma das seguintes opções: (1) *"considerar"* zero ou mais **atributos** e uma **entidade**; (2) "considerar" uma **entidade** e um **tipo** para essa entidade; ou (3) uma **funcionalidade** que pode opcionalmente ter uma **dependência**.

Cada regra, por sua vez, pode ser expandida:

- Um **atributo** também é uma entidade, mas ao usá-lo, fica caracterizado que essa entidade possui uma relação de atributo com outra entidade. Por exemplo, no fator "*O sistema deve considerar o nível de abstração dos relacionamentos das informações de contexto*", *nível de abstração* é atributo de *relacionamento* que é atributo de *informação de contexto.*
- Um **tipo** caracteriza que a entidade seguinte é um tipo da entidade anterior, por exemplo, no fator *"O sistema deve considerar informações de contexto do tipo informação de sistema.", informação de sistema* é um tipo de *informação de contexto.*
- Uma **funcionalidade** caracteriza que um **serviço** está associado a um **atributo** ou a uma **entidade**, por exemplo, no fator "*O sistema deve controlar sensores*", o **serviço** *controlar* está associado à entidade *sensores.*
- Uma **dependência** caracteriza que um **serviço** de uma **funcionalidade** tem uma relação de dependência com outro **atributo** ou **entidade**, por exemplo, o fator "*O sistema deve categorizar informações de contexto de acordo com sua fonte de dados.*", o **serviço** *categorizar* depende do **atributo** *fonte de dados* da **entidade** *informação de contexto.*

Uma vez definido o formato, todos os fatores de ubiqüidade foram revisados e adequados a ele. Durante essa atividade percebeu-se que alguns fatores precisavam ser particionados em dois ou três fatores para se adequarem ao novo formato, portanto, após a revisão, a lista de fatores aumentou em quantidade, mas manteve os mesmos conceitos e relações.

Em seguida, todos os fatores revisados foram avaliados pelo pesquisador que havia feito a sua descrição original. Essa avaliação permitiu a identificação de algumas correções ainda necessárias e que foram todas realizadas. O resultado está disponível no anexo B.

Adicionalmente, cada fator recebeu um identificador único composto pela sigla da característica de ubiqüidade e um número de dois algarismos, por exemplo, SC01 representa o primeiro fator da característica *Sensibilidade ao Contexto.* Neste trabalho,

esse identificador tem um papel muito importante, pois permite que seja feita a rastreabilidade dos artefatos derivados de cada fator.

Com vista a facilitar a compreensão do novo padrão, os fatores apresentados anteriormente foram reescritos no novo formato padrão:

Fator 1: Tratar alto acoplamento: existe relacionamento entre informações contextuais em diferentes níveis de abstração. É preciso gerenciar estes relacionamentos

Reescrito novo formato:

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

Fator SC02: O sistema deve gerenciar os relacionamentos das informações de contexto.

Considerações:

- Esse fator foi particionado em dois.

Fator 2: Categorizar contexto com base na fonte de dados de suas informações (informações de rede, dispositivo e interação do usuário)

Reescrito novo formato:

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

Fator 3: 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.

Reescrito novo formato:

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

Considerações:

- Esse fator utiliza informações complementares para explicar como as informações de contexto podem ser compartilhadas.

Fator 4: Considerar semântica na interpretação de informação de contexto.

Reescrito novo formato:

Fator SC05: O sistema deve interpretar informações de contexto de acordo com a sua semântica.

Fator 5: Controlar os sensores (ativo/em espera).

Reescrito novo formato:

Fator SC06: O sistema deve controlar os sensores.

Fator 06: Considerar informações de contexto de sistema. Aquelas relacionadas aos dispositivos, infra-estrutura do ambiente, como, por exemplo, informações de hardware utilizado: CPU, memória, tamanho da tela, energia, largura de banda disponível nos dispositivos acessíveis, bem como, outras aplicações que estão relacionadas para compor o sistema

Reescrito novo formato:

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

Considerações:

- Esse fator utiliza informações complementares para exemplificar e explicar o que é uma informação de sistema.

A partir desses fatores e da explicação fornecida anteriormente sobre o formato padrão, podem ser identificadas os seguintes conceitos e relações:

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 nos dispositivos acessíveis) e outras aplicações que estão relacionadas para compor o sistema.

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

Na seção seguinte, os fatores descritos neste novo formato são explorados para elaborar modelos sobre as características de ubiqüidade.

3.3. Construção dos Modelos de Ubiqüidade

Os modelos de ubiqüidade foram elaborados com o propósito de apresentar os conceitos e relações de cada característica de ubiqüidade 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 ubiqüidade apresentados no capítulo anterior.

A primeira etapa para construir esses modelos foi escolher a linguagem que seria utilizada. No início deste trabalho foi avaliada a possibilidade de representar as informações dos fatores em modelos de *features* utilizando alguma notação de engenharia de domínio.

Nesse sentido, foram analisadas algumas notações descritas em uma revisão da literatura realizada por OLIVEIRA (2006). Como resultado, foi percebido que as informações disponíveis nos fatores não eram suficientes para gerar esses modelos de forma adequada. Faltariam informações fundamentais, como: variantes, elementos opcionais e obrigatórios etc. Sendo assim, optou-se por estender a notação *Unified Modeling Language* (UML) (OMG, 2009) para elaborar os modelos de ubiqüidade.

A UML estabelece um meta-modelo que define a semântica para cada um dos seus diagramas (OMG, 2009). Este meta-modelo é especificado através de um diagrama de classes e define o significado de cada elemento e relacionamento que pode ser utilizado.

Nesse trabalho foi desenvolvido um meta-modelo para que o diagrama de classes da UML pudesse contemplar os conceitos e relações importantes dos fatores de cada característica de ubiqüidade.

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. Um meta-modelo pode ser usado também para permitir a integração entre diferentes ambientes de modelagem; melhorar a comunicação entre desenvolvedores; e validar a consistência de modelos (OLIVEIRA, 2006).

Para elaborar o meta-modelo, foi verificado como os fatores poderiam ser modelados. Sendo assim, as classes descritas na seção anterior foram mapeadas em elementos e relacionamentos do modelo, o que foi feito da seguinte forma:

- **Entidade**: foi mapeada para uma classe com o estereótipo <<entidade>>.
- **Serviço:** foi mapeado para uma classe com o estereótipo <<serviço>>.
- **Funcionalidade:** foi mapeado para um relacionamento de associação com origem em uma entidade e destino em um serviço.
- **Dependência:** foi mapeado para um relacionamento de dependência com origem em um serviço e destino em uma entidade.
- **Tipo:** foi mapeada para um relacionamento de generalização com origem e destino em entidades distintas.
- **Atributo:** foi mapeado para um relacionamento de agregação com origem e destino em entidades distintas.

A partir desse mapeamento, o meta-modelo, ilustrado na figura 4 foi elaborado. A partir dele 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.

Adicionalmente, algumas restrições não foram representadas no meta-modelo devido a dificuldade em representá-las, portanto, foram descritas a seguir:

7. entidades e serviços não podem ter relacionamentos recursivos, ou seja, aqueles em que a origem é a mesma do destino; e

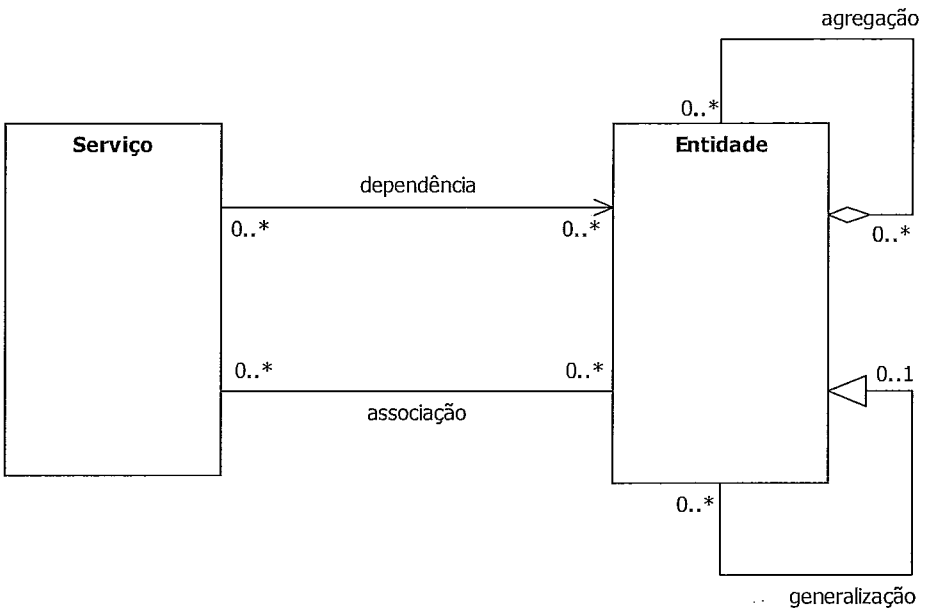


Figura 4: Meta-modelo das características de software ubíquo

8. entidades e serviços não podem ter relacionamentos circulares, ou seja, aqueles em que dois elementos possuem dois relacionamentos entre si, do mesmo tipo, mas com sentidos opostos.

Segundo CLEMENTS *et al.* (2004), para evitar erros de interpretação de modelos, é importante explicar o significado da notação utilizada, portanto ela foi novamente descrita, dessa vez, do ponto de vista de quem está lendo o modelo. Os elementos podem ser:

- **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.

E os relacionamentos podem ser:

- **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. A generalização é representada conforme a figura 5.

Figura 5: Exemplo de relacionamento de generalização

- **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. A agregação é representada conforme a figura 6;

Figura 6: Exemplo de relacionamento de agregação

- **Associação**: é um relacionamento entre um serviço e uma entidade que representa que um atua sobre o outro. A associação entre um serviço e uma entidade é ilustrada na figura 7.

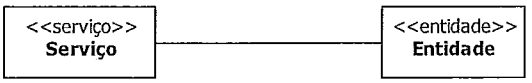


Figura 7: Exemplo de relacionamento de associação

- **Dependência**: é um relacionamento que representa que um serviço precisa consumir uma entidade para ser executado. A dependência é representada conforme a figura 8.

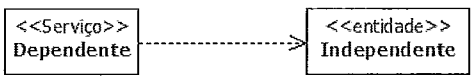


Figura 8: Exemplo de relacionamento de dependência

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

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, entre parêntesis, 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 ubiqüidade que estão relacionados (rastreabilidade).

Para mostrar como deve ser elaborado um modelo de ubiqüidade, os fatores apresentados na seção anterior são reapresentados a seguir:

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 9 ilustra o modelo construído. É possível reparar que cada conceito destacado nos fatores foi modelado em uma *entidade* ou *serviço*. Adicionalmente, os relacionamentos entre esses elementos se tornam muito mais claros 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 associada a *informação de contexto* e depende da *semântica da informação de contexto* porque essas relações foram explicitadas no *fatos SC05*

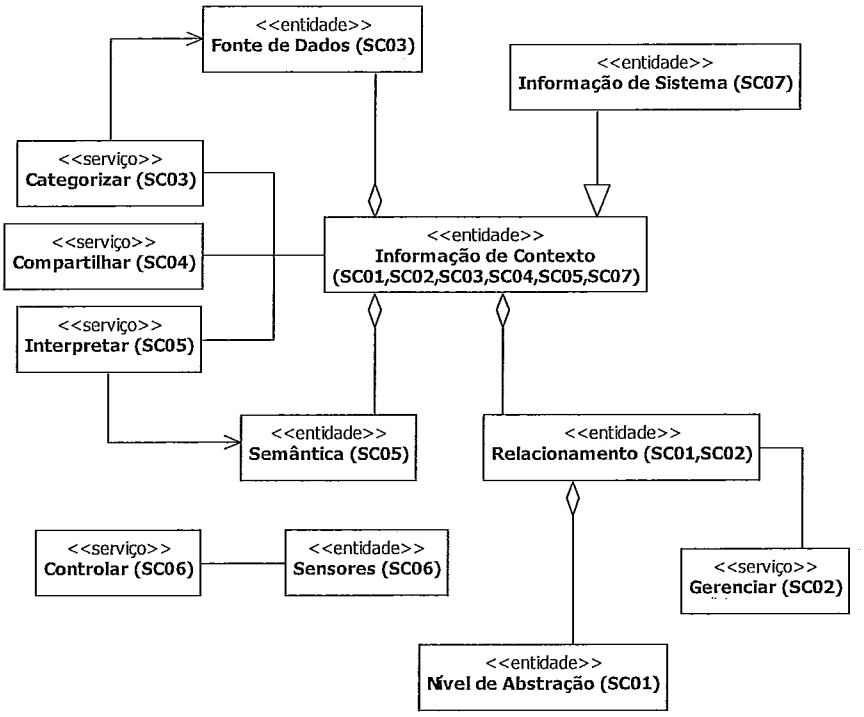


Figura 9: Exemplo de modelo de ubiqüidade de parte da característica *Sensibilidade ao Contexto*

Os modelos completos de cada uma das características estão disponíveis no anexo C. Cabe destacar que, no escopo deste trabalho, foram modeladas apenas as características funcionais: Captura da Experiência, Comportamento Adaptável, Composição de Funcionalidade, Descoberta de Serviços, Heterogeneidade de Dispositivos, Interoperabilidade Espontânea, Invisibilidade, Onipresença do Serviço e Sensibilidade ao Contexto.

3.4. Conclusão

Este capítulo apresentou como as informações importantes de cada característica de ubiqüidade podem ser formalizadas em modelos. Nesse sentido, foram modelados os conceitos e relações importantes descritos em cada fator.

Para tal foi necessário preparar um novo formato para a descrição dos fatores e adequar a ele os 119 fatores citados na tabela 1 (página 11) e disponíveis no Anexo B. Neste processo, alguns fatores foram particionados, o que resultou em uma lista maior de fatores. Entretanto, independente do tamanho da lista, os conceitos permaneceram os mesmos.

Os fatores adequados ao novo formato são apresentados no anexo B. A partir deles foi possível elaborar um modelo para cada característica de ubiqüidade, conforme apresentado no anexo C.

No próximo capítulo, esses modelos serão utilizados como base de uma abordagem para apoiar o desenvolvedor na definição de requisitos de ubiqüidade. Nesse sentido, será visto como as informações desses modelos podem ser transformadas em perguntas que guiem o desenvolvedor na captura dos requisitos associados às características de ubiqüidade.

Capítulo 4 - *UbiCheck*: Uma Abordagem para Apoiar a Definição de Requisitos de Ubiqüidade

Este capítulo apresenta UbiCheck, uma abordagem proposta para auxiliar o desenvolvedor na definição de requisitos de ubiqüidade em projetos de software. Ela explicita o conhecimento formalizado nos modelos de ubiqüidade em um guia que pode ser utilizado para indicar para o desenvolvedor as informações que devem ser capturadas pelos requisitos de ubiqüidade.

4.1. Introdução

Nos capítulos anteriores foram apresentados e formalizados fatores e características que podem estar presentes em softwares ubíquos. Neste capítulo, essas informações serão organizadas com o objetivo de propor uma abordagem para auxiliar o desenvolvedor a definir requisitos de softwares ubíquos.

Segundo OLIVEIRA *et al.* (2000) 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 de requisitos em particular, esse conhecimento poderia ser utilizado para destacar informações que deveriam ser capturadas nos requisitos de ubiqüidade do software.

Nesse sentido, os modelos elaborados anteriormente poderiam ser utilizados para fornecer conceitos e relações que poderiam ajudar na definição de requisitos de ubiqüidade. Entretanto, seria interessante que essas informações fossem trabalhadas para tornar o seu uso mais simples durante o desenvolvimento do software.

Sendo assim, surgiu a idéia de elaborar uma abordagem com objetivo de apresentar informações que permitam guiar o engenheiro de software na captura dos requisitos de ubiqüidade, a qual foi chamada *UbiCheck* (PINTO *et al.* 2008, SPINOLA *et al.* 2009). Esse Guia para a Definição de Requisitos de Ubiqüidade é representado por um *checklist* com perguntas sobre os assuntos importantes que devem ser capturados nos requisitos de ubiqüidade.

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

nestas perguntas. Dessa forma, é esperado que com o uso do Guia para Definição de Requisitos de Ubiqüidade seja possível conseguir reduzir a quantidade de defeitos de omissão[5] no documento de requisitos.

Na seção 4.2, a abordagem proposta será apresentada e em seguida, na seção 4.3, as suas atividades serão explicadas. Por fim, a seção 4.4 concluirá este capítulo.

4.2. Visão Geral de *UbiCheck*

UbiCheck é uma abordagem para apoiar a definição de requisitos de ubiqüidade. Ela está inserida no contexto de uma pesquisa mais ampla com objetivo de apoiar o desenvolvimento de software ubíquo (SPINOLA, 2008), que contempla também a inspeção de requisitos. A figura 10 representa uma visão geral dessa pesquisa, indicando: (1) os responsáveis pela execução de cada etapa; (2) as etapas consideradas; (3) as atividades executadas em cada etapa; e (4) as atividades realizadas com *UbiCheck* (marcadas em cinza).

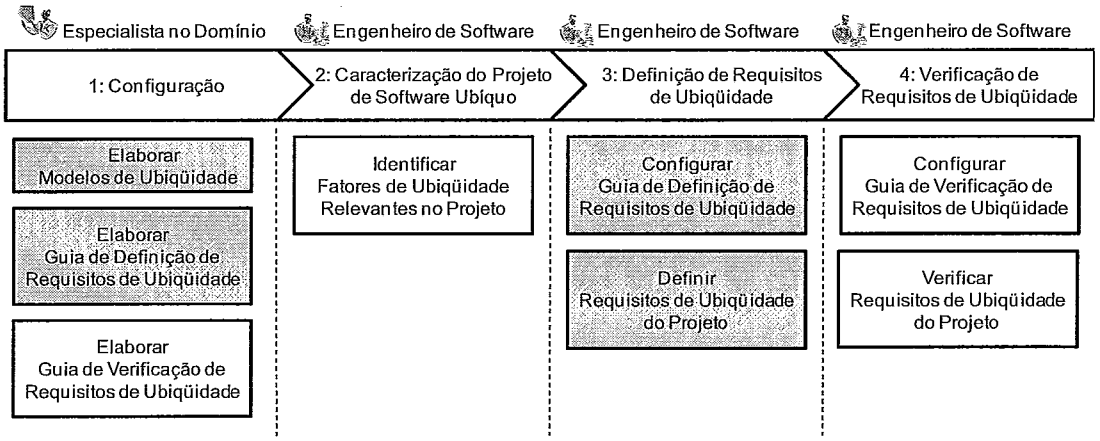


Figura 10: Visão geral do arcabouço para apoiar a definição e a verificação de requisitos de software ubíquo (SPINOLA, et al. 2008b)

A abordagem completa tem início com a sua **Configuração.** Nessa etapa as características de ubiqüidade são formalizadas em modelos e depois transformadas em um guia com perguntas para auxiliar na definição e na verificação de requisitos de ubiqüidade.

[5] Omissão é um tipo de defeito que representa que informações que deveriam ser capturadas não foram por algum motivo (BASILI *et al.*, 1996).

É importante destacar que esta atividade é responsável por preparar o guia (checklist) que será utilizado posteriormente para apoiar a definição de requisitos de ubiqüidade a partir de *Ubicheck*. 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 foi descrito no capítulo 3;
- **Elaborar Guia para Definição de Requisitos de Ubiqüidade**: nesta atividade os modelos de ubiqüidade são processados e perguntas sobre informações que precisam ser capturadas durante a definição de requisitos são elaboradas; e
- **Elaborar Guia para Verificação de Requisitos de Ubiqüidade**: nesta atividade os modelos de ubiqüidade 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.

Estas atividades, embora pertencentes a abordagem como um todo, estão no escopo deste trabalho e, portanto, todos os artefatos necessários para aplicação de *Ubicheck* para definição de requisitos já foram construídos. Desta forma, essas atividades só deveriam ser executadas novamente se houver alguma mudança nas características de ubiqüidade. Os modelos de ubiqüidade estão disponíveis no anexo C e o *Guia para Definição de Requisitos de Ubiqüidade* está disponível no anexo D.

A próxima etapa a ser realizada é a **Caracterização do Projeto de Software Ubíquo**, onde as necessidades do projeto são identificadas através da seguinte atividade:

- **Identificar Fatores de Ubiqüidade Relevantes no Projeto**: nesta atividade as necessidades do projeto são identificadas através de questionários que mapeiam os interesses do desenvolvedor com os fatores e características de ubiqüidade que podem ser relevantes para o projeto.

Na etapa de **Definição de Requisitos de Ubiqüidade** o desenvolvedor aplica o guia para definição de requisitos elaborado anteriormente e preenche o documento de requisitos de software com as informações capturadas. As seguintes atividades são realizadas:

- **Configurar o Guia para Definição de Requisitos de Ubiqüidade**: nesta atividade o desenvolvedor especializa o guia para definição de requisitos de ubiqüidade para que as necessidades do projeto sejam contempladas; e
- **Definir Requisitos de Ubiqüidade do Projeto**: nesta atividade os requisitos do projeto relacionados às características de ubiqüidade são capturados e o documento de requisitos de software é preenchido com essas informações.

Na etapa seguinte é realizada a **Verificação dos Requisitos de Ubiqüidade**. Nela, o documento de requisitos de software é inspecionado no intuito de assegurar que as informações que deveriam ser capturadas foram documentadas adequadamente. As seguintes atividades são realizadas:

- **Configurar Guia para Verificação de Requisitos de Ubiqüidade:** nesta atividade o *Guia para Verificação de Requisitos de Ubiqüidade* é especializado para contemplar as necessidades do projeto; e
- **Verificar Requisitos de Ubiqüidade do Projeto:** nesta atividade o documento de requisitos de software elaborado na etapa anterior é verificado e os erros encontrados são encaminhados para correção.

Conforme foi explicado anteriormente, essa abordagem completa faz parte de uma pesquisa mais ampla. No contexto dessa dissertação, *UbiCheck* contempla apenas as etapas referentes a definição de requisitos. Sendo assim, como não há apoio para a identificação dos fatores relevantes para o projeto e a verificação dos requisitos definidos pela abordagem, essas atividades ficam a cargo do desenvolvedor.

Nesse sentido, a figura 11, apresenta o encadeamento das atividades de

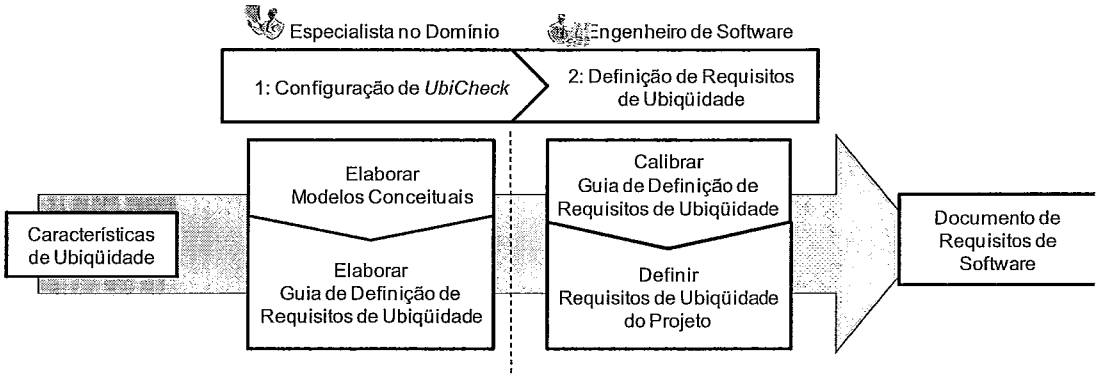


Figura 11: Visão geral de *UbiCheck*

UbiCheck, indicando: (1) os responsáveis pela execução de cada etapa; (2) as etapas consideradas; e (3) as atividades executadas em cada etapa; e (4) os artefatos consumidos e gerados no processo. As atividades realizadas na abordagem são explicadas na seção a seguir.

4.3. Atividades de *UbiCheck*

Conforme apresentado na seção anterior, *UbiCheck* possui duas etapas, cada uma com duas atividades. A primeira está associada à **Configuração de *UbiCheck***, ou seja, a preparação dos artefatos necessários para que ela possa ser utilizada. Cabe ressaltar que essa etapa já foi realizada nesta dissertação e o *Guia para Definição de Requisitos de Ubiqüidade* elaborado está disponível no anexo D.

A primeira atividade da etapa de **Configuração de *UbiCheck*** é a formalização das características de ubiqüidade em modelos. Como ela foi apresentada no capítulo anterior, não será abordada novamente.

Já a segunda etapa, está associada à **Definição de Requisitos de Ubiqüidade**, onde, para cada projeto, o desenvolvedor utiliza os artefatos gerados anteriormente para auxiliá-lo.

As próximas subseções explicam as atividades seguintes, respectivamente: **Elaborar Guia para Definição de Requisitos de Ubiqüidade**; **Configurar Guia para Definição de Requisitos de Ubiqüidade**; e **Definir Requisitos de Ubiqüidade**.

4.3.1. Elaborar Guia para Definição de Requisitos de Ubiqüidade

Esta atividade tem o objetivo de elaborar o *Guia para Definição de Requisitos de Ubiqüidade*. Ele é um conjunto de perguntas sobre informações presentes nos fatores de ubiqüidade que seriam interessantes de serem capturadas durante a definição de requisitos de ubiqüidade. Ele pode ser considerado uma forma mais direta de utilizar o conhecimento atrelado as características de ubiqüidade para apoiar a definição de requisitos de ubiqüidade.

Conforme ilustra a figura 12, o *Guia para Definição de Requisitos de Ubiqüidade* é elaborado a partir dos modelos das características de ubiqüidade. 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 13.

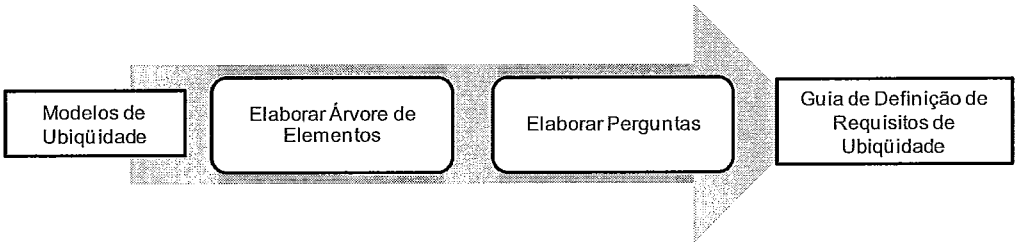


Figura 12: Sub-atividades para Elaborar o Guia para Definição de Requisitos de Ubiqüidade

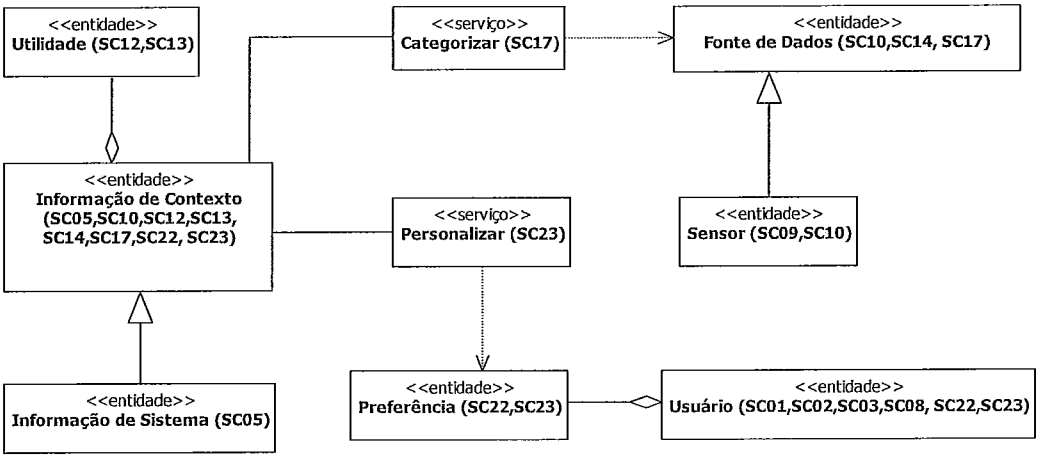


Figura 13: Trecho do Modelo de Sensibilidade ao Contexto

4.3.1.1. Elaborar Árvore de Elementos

Um modelo de ubiqüidade retrata uma visão onde são apresentados elementos (entidades e serviços) e relacionamentos (agregação, associação, dependência e generalização) de uma característica de ubiqüidade. Seria interessante que esses modelos apresentassem também informações sobre a importância que determinado elemento tem para uma característica.

Figura 14: Sub-atividades para Elaborar a Árvore de Elementos

Nesse sentido, foi elaborada uma nova visão do modelo, onde os elementos foram representados em árvores n-árias [6]. 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**. A figura 14 mostra os três passos realizados para transformar um modelo de ubiqüidade em uma árvore de elementos.

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, foram elaborados critérios para indicar a precedência entre os elementos do modelo.

Os critérios de precedência foram adaptados do trabalho de LIMA (2005), que descreve um conjunto de heurísticas para se 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

[6] Uma árvore n-ária é uma extensão da árvore binária onde um nó pode ter *N filhos* ao invés de apenas 2.

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 é mais importante, portanto, precede o elemento por ele contido.

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 13, apresentada anteriormente na página 45, destaca um trecho do modelo da característica *Sensibilidade ao Contexto,* o qual será utilizado para exemplificar os passos apresentados. Sendo assim, 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**. Foi convencionado 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[7] 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 15.

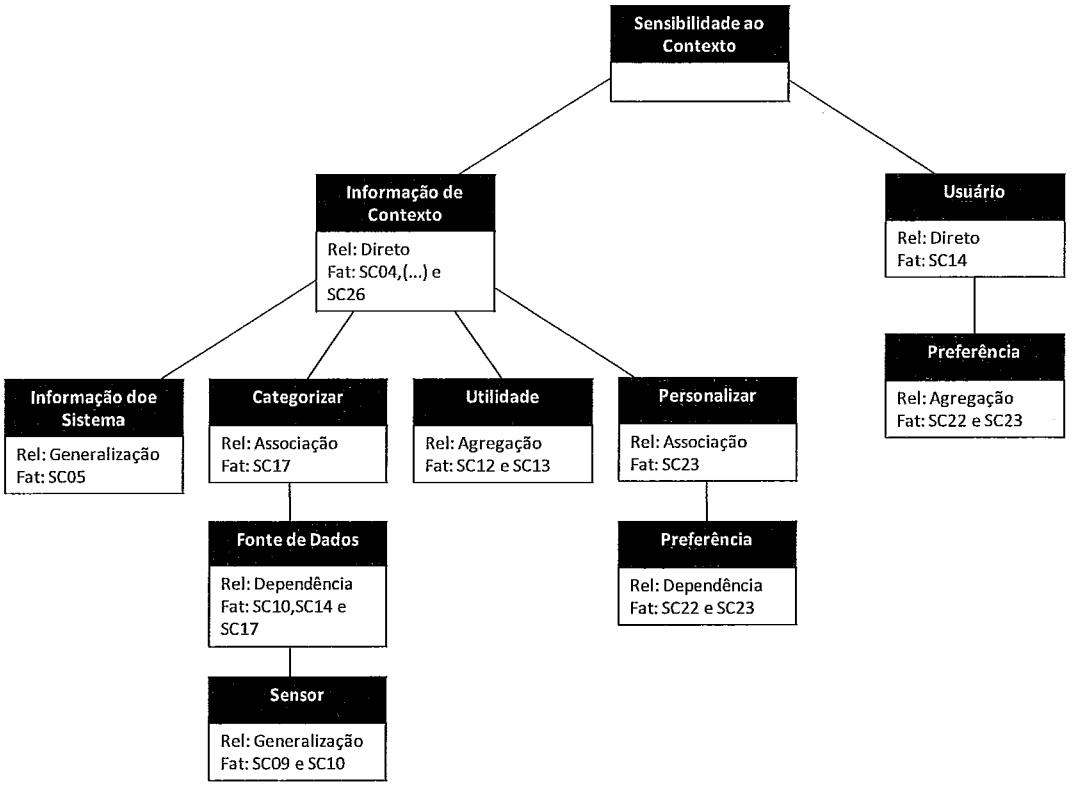


Figura 15: Árvore de Elementos de um trecho do modelo da característica *Sensibilidade ao Contexto* antes da Transformação.

[7] Um caminho é um encadeamento de nós distintos, de tal forma que exista sempre uma relação de "é filho de" ou "é pai de" entre nós consecutivos (SZWARCFILTER e MARKENZON, 1994).

Cabe destacar que, na construção, 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 15, 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.

Nesse ponto, a árvore de elementos está praticamente construída. Porém, como pode ser percebido na figura 15, ocorrem os seguintes problemas:

1. No critério de precedência 4, foi convencionado que uma *Entidade* é mais importante que um *Serviço*. Entretanto, as entidades *Fonte de Dados* e *Sensor* são filhos do serviço *Categorizar.*
2. O nó *Preferência*, filho de *Personalizar* ficou com o significado descontextualizado. Esse nó deveria ser alterado para *Preferência do Usuário.*

Para corrigir esses problemas é realizado o passo seguinte para **Transformar a Árvore de Elementos**, o que consiste na reclassificação de alguns nós, conforme descrito a seguir.

A primeira transformação é realizada quando um nó tem filhos e o seu atributo *Relacionamento* tem o valor *Dependência.* Esse nó é copiado e re-inserido a partir da raiz da árvore. O atributo relacionamento do novo nó é *Direto* e os filhos do nó anterior são movidos para o novo nó, conforme ilustra a figura 16a.

A segunda transformação é realizada quando um nó (A) cujo atributo *Relacionamento tem* o valor *Dependência* possui outro *nó similar* (B) com atributo *Relacionamento* com valor *Agregação.* Neste caso, o primeiro nó (A) deve mudar de

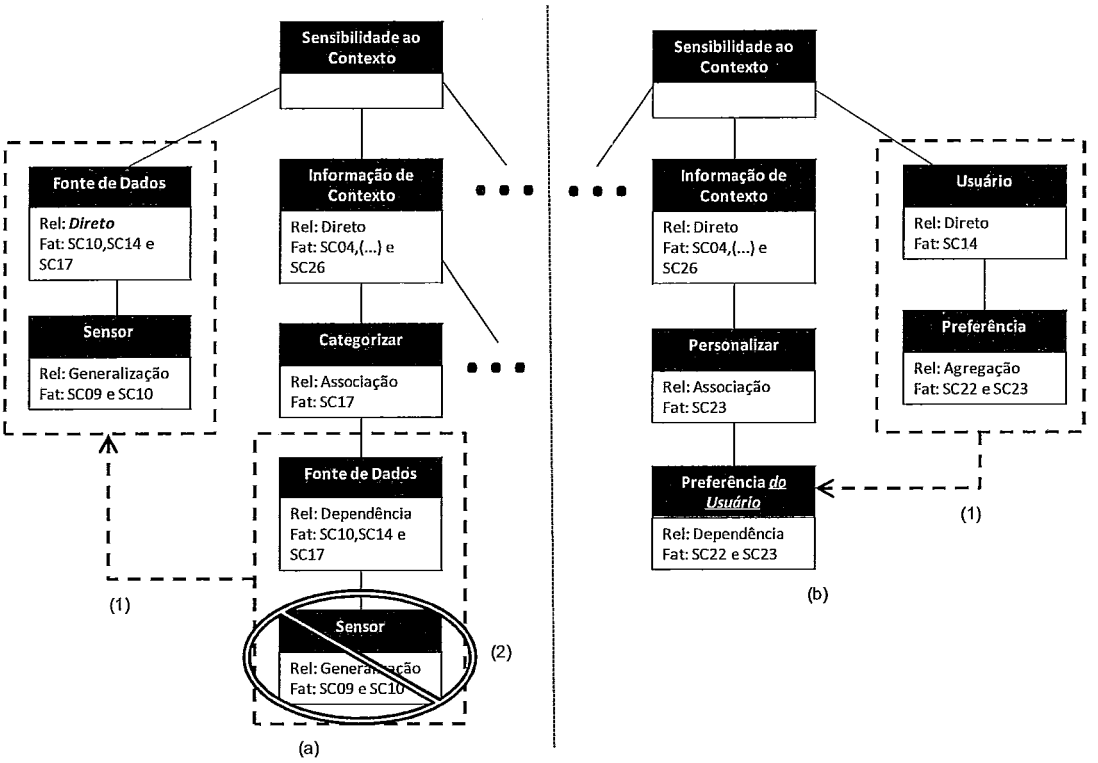


Figura 16: Transformações da Árvore de Elementos

nome para contemplar também o nome do pai do segundo nó (B). A figura 16b ilustra essa transformação. Nela, o nó *Preferência,* filho de *Personalizar* muda de nome para *Preferência do Usuário*.

Após as transformações serem realizadas, a árvore está pronta, conforme ilustra a figura 17.

Cabe destacar que o espaço consumido para representar a árvore de elementos cresce bastante conforme aumenta o número de nós da árvore. Sendo assim, como as árvores dos modelos de ubiqüidade completos possuem muitos nós, optou-se por representá-las em tabelas, fazendo uso da notação em barras e ocultando o nó que representa a raiz. A tabela 3 mostra a mesma árvore apresentada na figura 17 neste novo formato.

Na subseção seguinte a árvore de elementos gerada será utilizada para elaborar as perguntas do guia para definição de requisitos de ubiqüidade.

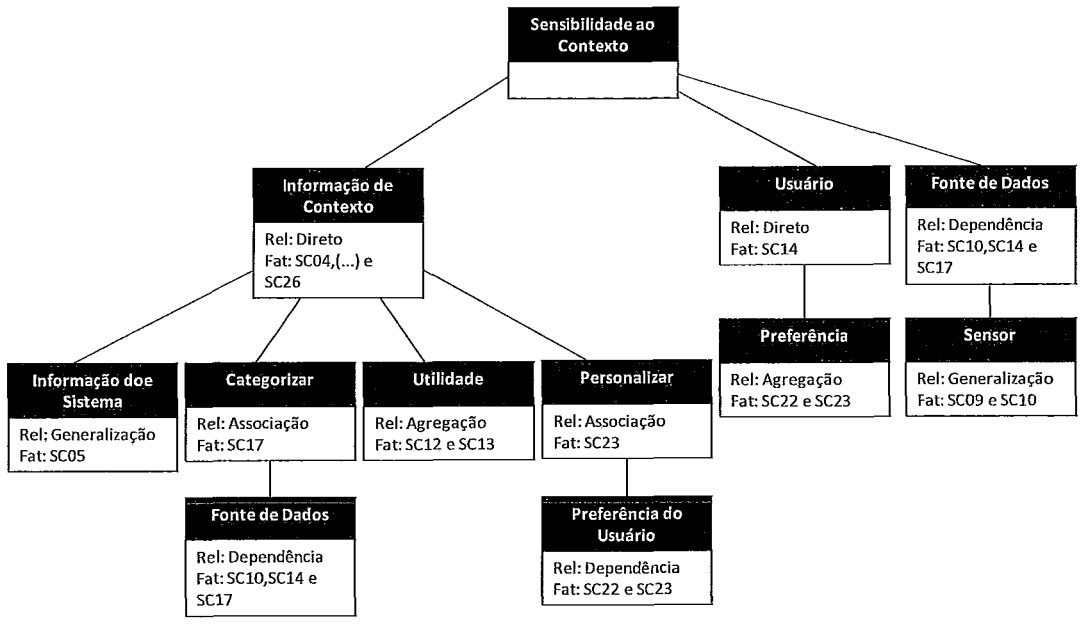


Figura 17: Exemplo de representação de uma árvore de elementos de um trecho do modelo da característica Sensibilidade ao Contexto

Tabela 3: 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

4.3.1.2. Elaborar Perguntas

As perguntas que compõem o Guia para Definição de Requisitos de Ubiqüidade são elaboradas a partir da árvore de elementos representadas em tabelas. A figura 18 mostra os dois passos realizados para elaborar perguntas a partir de uma árvore de

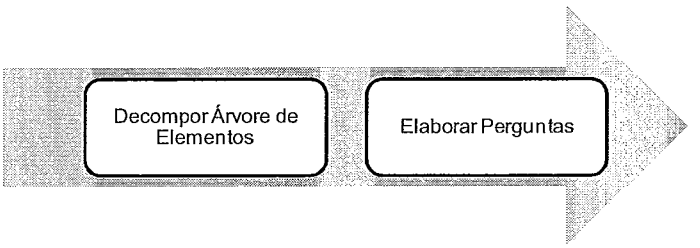


Figura 18: Sub-atividades para Elaborar Perguntas do Guia para Definição de Requisitos de Ubiqüidade

elementos. Para exemplificá-los será utilizada a mesma árvore de elementos apresentada anteriormente na Tabela 3.

O primeiro passo 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

O segundo passo é **Elaborar as Perguntas** a partir desses caminhos. Nesse sentido, 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 4 mostra o assunto, escopo e tipo extraído dos caminhos apresentados. Essa tríade é explicada em seguida.

Tabela 4: 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 pode se reparar, 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*.

Como se pode perceber, as perguntas para os nós mais próximos da raiz da árvore tendem a ser mais genéricas. Conforme vão sendo considerados 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 5 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 5: 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 mostra como as 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 *categorizar 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 ...?*"

Na sub-seção seguinte, será visto como essas perguntas podem ser especializadas para contemplar necessidades específicas de um projeto de software ubíquo.

Tabela 6: 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*?

4.3.2. Configurar Guia para Definição de Requisitos de Ubiqüidade

O *Guia para Definição de Requisitos de Ubiqüidade* elaborado até agora considera as informações de todos os fatores de ubiqüidade. Entretanto, não considera se todos esses fatores são importantes para o projeto em que ele será aplicado.

Como no capítulo 2 foi discutido que projetos de software ubíquo podem contemplar uma pequena parcela dos fatores, seria interessante aproveitar essa situação para configurar o *Guia para Definição de Requisitos de Ubiqüidade.* Nesse sentido, poderiam ser consideradas apenas as perguntas que tivessem relação com os fatores considerados relevantes para o projeto, o que poderia 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.

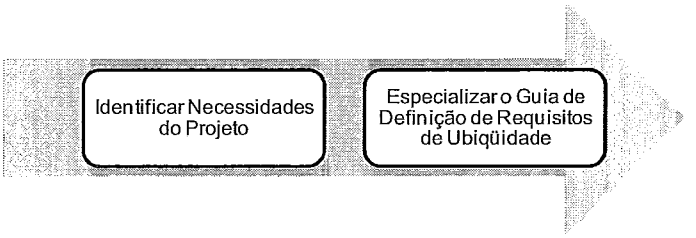


Figura 19: Passos da atividade Configurar Guia para Definição de Requisitos de Ubiqüidade

- **Redução do tamanho do guia:** ele conteria menos perguntas, portanto, exigiria menos esforço para ser aplicado em um projeto.

Sendo assim, foi preparada uma estratégia para especializar o *Guia para Definição de Requisitos de Ubiqüidade.* Conforme ilustra a figura 19, o primeiro passo é **Identificar as Necessidades do Projeto**, que consiste na seleção dos fatores de ubiqüidade relevantes.

Conforme visto no final da seção anterior, *UbiCheck* oferece apoio parcial para a seleção dos fatores: é fornecida uma lista de fatores e o desenvolvedor utilizar sua própria experiência e intuição para selecionar aqueles que são relevantes para o projeto.

Uma vez selecionados os fatores, o próximo passo é **Especializar o Guia para Definição de Requisitos de Ubiqüidade**, o que é feito selecionando apenas as perguntas que possuem relação com esses fatores, ou seja, aquelas em que foram originadas por pelo menos um fator considerado relevante para o projeto.

A tabela 7 mostra um trecho do *Guia para Definição de Requisitos de Ubiqüidade* da característica *Sensibilidade ao Contexto,* o qual será utilizado para exemplificar essa atividade.

Tabela 7: Trecho do Guia para Definição de Requisitos de Ubiqüidade da característica *Sensibilidade ao Contexto*

Fatores	Pergunta
SC05,SC10,SC12,SC13, 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*?

Fatores	Pergunta
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 *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 configurar o *Guia para Definição de Requisitos de Ubiqüidade* basta realizar a união das perguntas associadas a cada fator relevante para o projeto.

Supondo que seja interessante para o projeto definir requisitos associados apenas aos fatores SC12, SC13 e SC23, o guia configurado 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 configurado 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 era contemplada no guia anterior por fazer parte dos fatores SC12, SC13 e SC23.

Uma vez configurado, a próxima atividade é definir os requisitos de ubiqüidade do projeto, o que é descrito na próxima subseção.

4.3.3. Definir Requisitos de Ubiqüidade do Projeto

UbiCheck é uma abordagem baseada em perguntas (*Checklist based*), portanto, o apoio fornecido tem o intuito de 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 para Definição de Requisitos de Ubiqüidade* durante a definição de requisitos de ubiqüidade. Conforme o desenvolver as responde, a sua atenção é direcionada para os assuntos relacionados à ubiqüidade que devem ser capturados no documento de requisitos do projeto (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.

Dessa forma, é possível que o número de defeitos de omissão[8] no documento de requisitos reduza se comparado com as outras abordagens disponíveis na literatura (JORGENSEN e BOSSEN, 2003; HONG *et al.*, 2005; GOLDSBY e CHENG, 2006; CHIU *et al.*, 2006 e 2007; BO *et al.*, 2007; XIANG *et al.*, 2007; MARKOSE *et al.*,2008; OYAMA *et al.*, 2008; e MING *et al.*,2008). Esta é uma investigação que necessita ser realizada porém por restrição de tempo está fora do escopo deste trabalho.

Uma consideração importante sobre o uso de *UbiCheck* é que o apoio fornecido por ela se aplica apenas a definição de requisitos de ubiqüidade. Sendo assim, os demais requisitos (funcionais e não funcionais) ainda devem ser definidos. Com esse propósito, outras abordagens podem ser combinadas.

4.4. Conclusão

Este capítulo apresentou *UbiCheck:* uma abordagem para apoiar a definição de requisitos de ubiqüidade baseada em *checklists*. O apoio fornecido tem o intuito de

[8] Omissão é um tipo de defeito que representa que informações que deveriam ser capturadas não foram por algum motivo (BASILI *et al.*, 1996).

orientar o desenvolvedor sobre quais informações são importantes e devem ser capturadas nos requisitos do software.

A abordagem é composta por duas etapas. Na primeira ela é preparada para ser utilizada e na segunda ela é aplicada em projetos de software ubíquo. Nesta dissertação, a primeira etapa de *UbiCheck* foi realizada, portanto, ela está pronta para ser aplicada em projetos.

Contudo, é importante destacar que *UbiCheck* não pretende fornecer apoio para a definição de requisitos convencionais (não relacionados a computação ubíqua) e para a verificação da consistência dos requisitos de ubiqüidade definidos. Dessa forma, cabe ao desenvolvedor tratar essas limitações no processo de desenvolvimento do software.

No próximo capítulo, será apresentado um estudo sobre o uso de *UbiCheck* em um projeto de software ubíquo.

Capítulo 5 - Avaliação de *UbiCheck*

Este capítulo apresenta um estudo experimental para avaliar o apoio da abordagem UbiCheck na definição de requisitos de ubiqüidade em projetos de software.

5.1. Introdução

Segundo PFLEEGER (1999), nenhuma ciência avança sem experimentação e medição, o que reforça a idéia de que estudos experimentais devem ser realizados e repetidos para melhorar produtos, processos, tecnologias etc.

Sendo assim, foi realizado um estudo de observação para caracterizar *UbiCheck* no que diz respeito a sua viabilidade de aplicação na definição de requisitos em projetos de software ubíquo. Embora este tipo de estudo sirva para responder perguntas simples relacionadas ao que acontece quando, por exemplo, um processo é utilizado, ele é requisito para os demais estudos, portanto deve ser executado primeiro (TRAVASSOS *et al.* 2002).

Na seção 5.2 o plano desse estudo é apresentado, na seção 5.3 os resultados são discutidos e na seção 5.4 este capítulo é concluído.

5.2. Plano do Estudo

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. Nesse sentido, foi analisado o uso do *Guia para Definição de Requisitos de Ubiqüidade* no desenvolvimento do Sistema de Gestão de Inventário Patrimonial (SGI).

Segundo o paradigma GQM (BASILI e ROMBACH 1988), o objetivo deste estudo é:

Analisar a	a definição de requisitos de ubiqüidade com *UbiCheck.*
com o propósito de	caracterizar o *Guia para Definição de Requisitos de Ubiqüidade*
em relação à	sua aplicabilidade

do ponto de vista do	estudantes de engenharia de software
no contexto do	projeto do Sistema de Gestão de Inventário Patrimonial (SGP)

O intuito de caracterizar a aplicabilidade do *Guia para Definição de Requisitos de Ubiqüidade* é verificar se os desenvolvedores são capazes de entendê-lo e aplicá-lo em um projeto de software. Sendo assim, esse estudo pretende verificar se o o *Guia para a Definição de Requisitos de Ubiqüidade* **consegue** ser utilizado pelos desenvolvedores para apoiar o desenvolvimento de um projeto de software ubíquo.

Para executar esse estudo, foram convidados alunos de pós-graduação que estavam participando de uma disciplina de engenharia de software no segundo trimestre de 2009. Ao todo, foram considerados 8 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. Os modelos desses documentos estão disponíveis no anexo E.

Os participantes foram divididos em três equipes (A, B e C) – as equipes A e B foram formadas por 3 alunos e a equipe C por 2 alunos. A escolha dos participantes de cada equipe foi feita tomando como referencia 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 ubiqüidade, portanto, as equipes foram solicitadas a estender o documento para contemplar dois novos cenários (Controle de Bicicletas e Controle de Entrada e Saída de Patrimônio) que demandariam o uso da computação ubíqua. No Anexo E estão disponíveis informações sobre os cenários distribuídos.

A definição dos requisitos de cada novo cenário do SGP foi feita de forma seqüencial, a primeira sem o apoio e a segunda com o apoio de *UbiCheck*. Sendo assim, na primeira interação a equipe A definiu requisitos para o cenário *Controle de Bicicletas* e as equipes B e C definiram requisitos para o cenário *Controle de Entrada e Saída de Patrimônio*. Na segunda interação, a equipe A definiu requisitos para o cenário *Controle*

de Entrada e Saída de Patrimônio e as equipes B e C definiram requisitos para o cenário *Controle de Bicicletas*, dessa vez com o uso de *UbiCheck*. A tabela 8 resume essas interações.

Tabela 8: Equipes, Interações e Cenários do Estudo de Observação

	1ª Interação (sem apoio)	2ª Interação (*UbiCheck*)
Equipe A	Controle de Bicicletas	Controle de Entrada e Saída de Patrimônio
Equipe B	Controle de Entrada e Saída de Patrimônio	Controle de Bicicletas
Equipe C	Controle de Entrada e Saída de Patrimônio	Controle de Bicicletas

Na segunda interação, o *Guia para Definição de Requisitos de Ubiqüidade* foi entregue preparado e já 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 ubiqüidade. 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.

5.3. Resultado do Estudo

Conforme mencionado, o objetivo deste estudo é verificar se o *Guia para Definição de Requisitos de Ubiqüidade* conseguiria ser aplicado em um projeto de software ubíquo. Nesse sentido, foram analisadas as respostas dos questionários preenchidos por cada equipe. As respostas indicam 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.

Contudo, a análise desses questionários apontou que algumas das perguntas do *Guia para Definição de Requisitos de Ubiqüidade* 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 endereçados em etapas posteriores do desenvolvimento.

Mesmo com essa limitação, o resultado apóia que o *Guia para a Definição de Requisitos de Ubiqüidade* **consegue** ser utilizado pelos desenvolvedores para apoiar o desenvolvimento de um projeto de software ubíquo e, por conseqüência. Um estudo mais elaborado deveria ser realizado para realizar uma avaliação quantitativa deste comportamento.

Adicionalmente, nos questionários foram registrados os tempos que cada equipe gastou para definir os requisitos de cada cenário, conforme mostra a tabela 9.

Tabela 9: Tempo gasto por cada equipe para definir os requisitos de ubiqüidade

	Tempo Gasto em cada Interação	
	1ª Interação (sem apoio)	2ª Interação (*UbiCheck*)
Equipe A	300 minutos Controle de Bicicletas	240 minutos – Cenário 2 Controle de Entrada e Saída de Patrimônio
Equipe B	240 minutos Controle de Entrada e Saída de Patrimônio	120 minutos Controle de Bicicletas
Equipe C	180 minutos Controle de Entrada e Saída de Patrimônio	180 minutos Controle de Bicicletas

Os tempos registrados sugerem que aplicar *UbiCheck* pode não onerar o processo de desenvolvimento. Comparando o resultado das equipes A e B (possuíam o mesmo número de alunos), nota-se que na segunda interação, com o uso de *UbiCheck,,* houve redução de 60% para o cenário *Controle de Bicicletas* e o tempo para o cenário *Controle de Entrada e Saída de Patrimônio* se manteve.

Contudo, cabe observar que existem riscos a 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 deveria ser realizada visando fortalecer a observação deste comportamento. Algumas ameaças a validade são apresentadas na conclusão deste capítulo.

5.4. Conclusão

Esse capítulo apresentou um estudo simples sobre a aplicabilidade de *UbiCheck*. Destacamos algumas das ameaças a validade deste estudo:

- o estudo foi realizado com alunos de pós graduação em uma turma de engenharia de software, os quais não possuem o mesmo grau de experiência e conhecimento. Tentou-se mitigar esse risco definindo grupos equilibrados.
- o estudo foi realizado em apenas um projeto de sala de aula, cujos dois cenários eram parecidos, o que pode ter influenciado o tempo da segunda interação pela questão da aprendizagem.
- O tamanho da população não permite um tratamento estatístico adequado
- O estudo fez parte de um contexto de discussão amplo em verificação, validação e teste de software, cujos conceitos discutidos em sala de aula podem ter influenciado os resultados.

Contudo, a observação da utilização de *UbiCheck* por parte das equipes nos permitiu observar que, a principio, *UbiCheck* pôde ser utilizada e, ao mesmo tempo, identificar oportunidades de melhoria na abordagem proposta, que serão objeto de discussão do próximo capítulo.

Capítulo 6 - UbiCheck 2.0

Este capítulo apresenta melhorias implementadas em UbiCheck para suprir as limitações encontradas durante a execução do estudo sobre a abordagem.

6.1. Introdução

No capítulo anterior foram descritos resultados de um estudo de observação relacionado a utilização de *UbiCheck* num projeto de software ubíquo. Embora a abordagem tenha se mostrado viável para apoiar a definição de requisitos de ubiqüidade 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 guardá-las para fases posteriores do desenvolvimento do software;
- algumas perguntas não foram respondidas porque não se sabia como escrever um requisito para respondê-las; e
- algumas perguntas eram difíceis de entender, pois utilizavam termos muitos 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 20 representa 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 para Definição de Requisitos de Ubiqüidade: essa **atividade foi alterada** para que durante a elaboração das perguntas, o especialista em

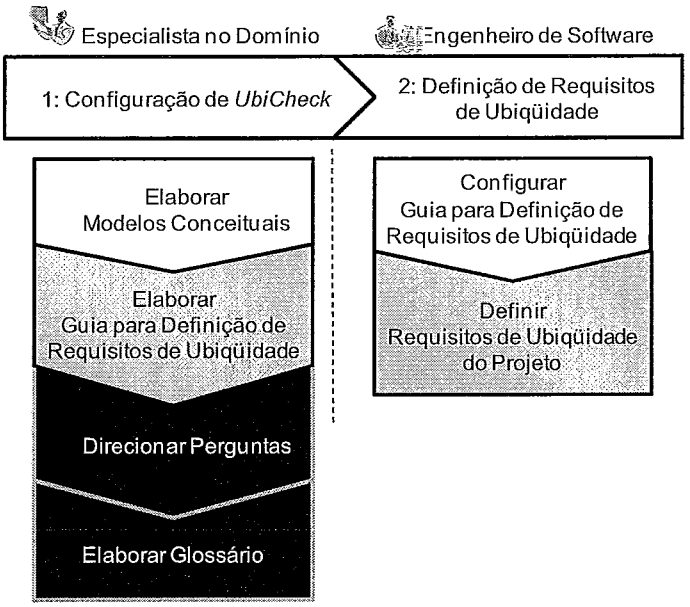


Figura 20: Visão geral de *UbiCheck* 2.0

ubiqüidade pudesse definir a etapa do projeto de desenvolvimento que cada pergunta deveria ser aplicada.

- Elaborar Direcionamento da Pergunta: 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 ubiqüidade.
- Elaborar Glossário: essa **atividade foi inserida** para que os termos utilizados no guia para definição de requisitos de ubiqüidade pudessem ser explicados, o que conseqüentemente deve melhorar o entendimento de suas perguntas.

Etapa 2: Definição de Requisitos de Ubiqüidade

- Configurar Guia para Definição de Requisitos de Ubiqüidade: essa **atividade não foi alterada** na nova versão de *UbiCheck*.
- Definir Requisitos de Ubiqüidade do Projeto: essa **atividade foi alterada** para que novas orientações fossem fornecidas para os desenvolvedores que vão aplicar a abordagem em um projeto de software ubíquo.

Na seção 6.2 as atividades que foram alteradas ou inseridas na nova versão da abordagem são explicadas, em seguida, na seção 6.3 são apresentados feedbacks, sobre a nova abordagem, de participantes do estudo realizado no capítulo anterior. Por fim, a seção 6.4 conclui este capítulo.

6.2. Atividades Alteradas e Inseridas em *UbiCheck* 2.0

6.2.1. Elaborar Guia para Definição de Requisitos de Ubiqüidade

Embora os fatores das características de ubiqüidade 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 ubiqüidade 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. Com vistas a 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 as informações do usuário são consideradas?

Classificação: aplicável em requisitos

Motivo: Essa é uma pergunta que discute como as informações de usuários devem ser tratadas, o que pode ser feito na 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 que 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 Anexo F já se encontram classificadas.

6.2.2. Elaborar Direcionamento da Pergunta

As perguntas que compõem o guia para definição de requisitos de ubiqüidade têm o objetivo de destacar as informações importantes que devem ser capturadas nos requisitos de ubiqüidade. Contudo, no estudo apresentado no capítulo anterior 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 especificar:

- **Item de Especificação:** o tipo de informação que deve ser definida para responder a pergunta, por exemplo, um requisito funcional, uma regra de negócio, uma diretriz, um caso de uso etc.
- **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 **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.

Novamente, é importante destacar que neste trabalho, o guia para definição de requisitos de ubiqüidade fornecido no Anexo F já contém um direcionamento para cada pergunta. Contudo, foram consideradas apenas as características de ubiqüidade - Captura de Experiência, Comportamento Adaptável, Heterogeneidade de Dispositivo, Onipresença de Serviços e Sensibilidade ao Contexto, devido à decisão de continuidade a pesquisa no contexto de desenvolvimento de software ubíquo.

6.2.3. Elaborar Glossário

As perguntas do guia para definição de requisitos de ubiqüidade foram elaboradas com base nos fatores das características de ubiqüidade. Como esses fatores utilizam termos específicos do domínio da computação ubíqua, as perguntas acabaram também contendo esses termos.

Como a idéia é que *UbiCheck* possa ser aplicada por desenvolvedores 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 ubiqüidade, pois elas definem os assuntos das perguntas do guia para definição de requisitos de ubiqüidade e representam os principais conceitos presentes em cada fator. Sendo assim, para cada entidade dos modelos de ubiqüidade, foi elaborada uma definição, disponível no Anexo G.

6.2.4. Definir Requisitos de Ubiqüidade do Projeto

O uso de *UbiCheck* para apoiar na definição de requisitos foi alterado devido a inserção dos direcionamentos nas perguntas do guia para definição de requisitos de

ubiqüidade 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 etc.

Dessa forma, além da abordagem mostrar para o desenvolvedor quais informações devem ser capturadas nos requisitos de ubiqüidade, ela o orienta sobre como elas devem ser capturadas. Adicionalmente, o desenvolvedor tem a possibilidade de consultar um glossário de termos importantes quando tiver dificuldade em entender o assunto de uma pergunta.

Sendo assim, é provável que as melhorias acrescidas em *UbiCheck* 2.0 possam simplificar e fornecer mais apoio para os desenvolvedores durante a fase de definição dos requisitos de ubiqüidade em um projeto de desenvolvimento de software.

6.3. *Avaliação de UbiCheck 2.0*

Devido a restrição de tempo, não foi possível repetir o estudo anterior com a nova versão de UBiCheck. Desta maneira, visando ter algum retorno sobre as melhorias realizadas, optou-se por apresentar estas melhorias para um dos participantes do estudo descrito no capítulo 5.

O participante escolhido era um daqueles que haviam feito críticas sobre o uso da abordagem. Ele foi apresentado às melhorias implementadas na abordagem e, ao final, concordou que elas possivelmente melhoram a compreensão e o uso da abordagem.

Ele acrescentou que, para algumas perguntas, seria interessante fornecer exemplos de requisitos. Contudo, foi explicado que essa alteração não será realizada devido ao aumento de tamanho que ela pode proporcionar ao *guia para definição de requisitos de ubiqüidade*

Contudo, esse resultado é limitado e não pode ser generalizado. É importante que o estudo apresentado no capítulo anterior seja repetido com a nova versão de *UbiCheck* para fortalecer a observação deste comportamento.

6.4. Conclusão

Este capítulo apresentou mudanças em *UbiCheck* com o intuito de suprir deficiências na abordagem. 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 perguntas.

Nesse sentido, na versão 2.0 da abordagem foram alteradas duas atividades e incluídas mais duas. Essas alterações foram reapresentadas para um participante do estudo realizado nessa dissertação (descrito no capítulo 5). O retorno foi positivo, o que sugere que *UbiCheck* 2.0 é mais adequada que sua versão anterior para apoiar a definição de requisitos de ubiqüidade em projetos de software. Contudo, é importante

No capítulo seguinte será apresentado um protótipo de infra-estrutura computacional para apoiar a configuração e a aplicação da abordagem. Dessa forma, é esperado que o seu uso seja simplificado.

Capítulo 7 - Protótipo de Infra-estrutura Computacional para Apoiar o Uso de *UbiCheck*

Este capítulo apresenta o protótipo de infra-estrutura computacional desenvolvida para automatizar parte das atividades realizadas em UbiCheck. A partir do seu uso é esperado que o esforço gasto e os defeitos inseridos durante a execução das atividades da abordagem sejam reduzidos.

7.1. Introdução

No capítulo anterior foi apresentada a versão 2.0 de *UbiCheck* que provê melhorias para suprir limitações da primeira versão da abordagem. Neste capítulo será apresentado o protótipo de uma infra-estrutura computacional para apoiar a configuração da abordagem e a sua aplicação para apoiar a definição de requisitos de ubiqüidade. Nesse sentido, foi observado o que poderia ser automatizado em cada atividade de *UbiCheck,* conforme descrito a seguir e ilustrado na figura 21.

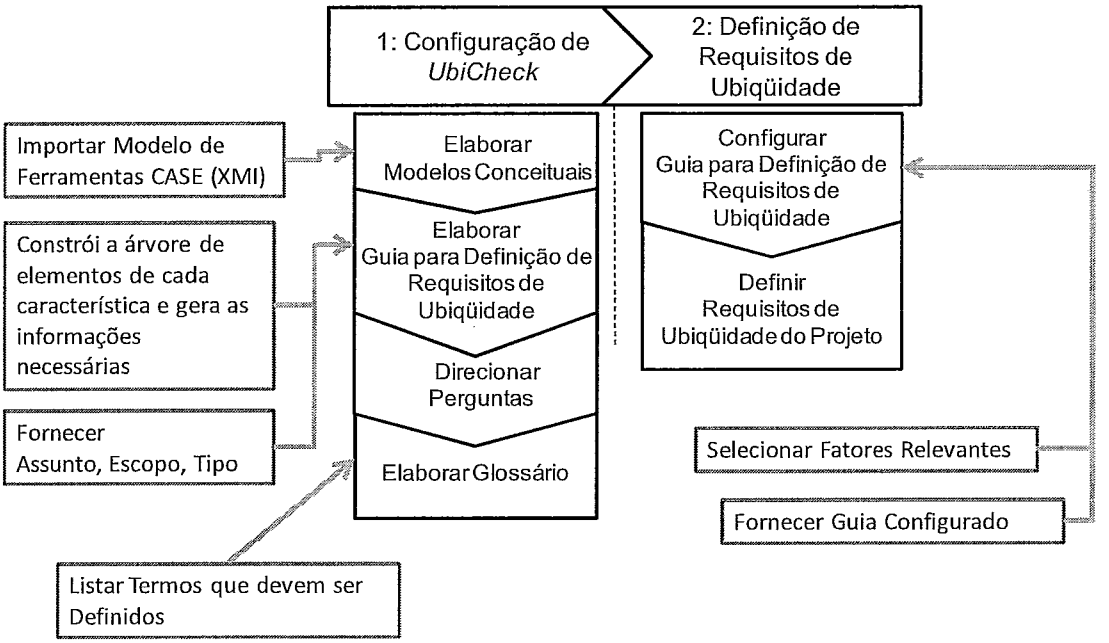


Figura 21: Funcionalidades do protótipo de infra-estrutura computacional

Na etapa de **Configuração de UbiCheck**, para auxiliar na atividade **Elaborar Modelos** foi elaborado um componente para permitir a importação de modelos para a

infra-estrutura computacional. Dessa forma, o especialista pode escolher a ferramenta de modelagem UML de sua preferência. Adicionalmente, durante a importação, a infra-estrutura valida se o modelo está de acordo com a sintaxe estabelecida no meta-modelo definido no capítulo 3.

Na etapa seguinte, **Elaborar Guia para Definição de Requisitos de Ubiqüidade**, os modelos importados são processados para que a árvore de elementos pudesse ser construída automaticamente. A partir dela, é extraído o assunto, escopo e tipo de cada pergunta que o especialista no domínio terá que elaborar. Essas informações são apresentadas através de uma planilha eletrônica, conforme ilustra a figura 22.

	A	B	C	D	E	F
1	Fatores	Tipo	Árvode de Elementos	Pergunta	Aplicabilidade	Direcionamento
2	CE01,CE02,CE03,CE	DIR	Usuário			
3	CE01,CE02,CE04	AGR	-- Interação			
4	CE01,CE02	AGR	--- Informação			
5	CE01,CE03	ASS	---- Capturar			
6	CE02	ASS	---- Armazenar			
7	CE04	AGR	--- Padrão			

Figura 22: Planilha Eletrônica Gerada pela Infra-estrutura Computacional

A atividade **Elaborar Direcionamento da Pergunta** não foi automatizada, pois o trabalho depende exclusivamente do conhecimento do especialista e, a atividade **Elaborar Glossário** foi parcialmente automatizada: a infra-estrutura fornece para o especialista a lista de termos que devem ser definidos.

Na etapa seguinte, **Definição de Requisitos de Ubiqüidade**, foi elaborada uma interface *Web* para apoiar na atividade **Configurar Guia para Definição de Requisitos de Ubiqüidade**. Ela apresenta os fatores na tela e, conforme o desenvolvedor seleciona aqueles fatores que considera relevantes para o seu projeto, as perguntas são apresentadas. A figura 23 ilustra essa interface: a esquerda estão os fatores selecionados e a direita as perguntas relacionadas.

A atividade **Definir Requisitos de Ubiqüidade do Projeto** foi parcialmente automatizada: a interface descrita anteriormente fornece a lista de perguntas que devem ser respondidas para capturar os requisitos de ubiqüidade do projeto.

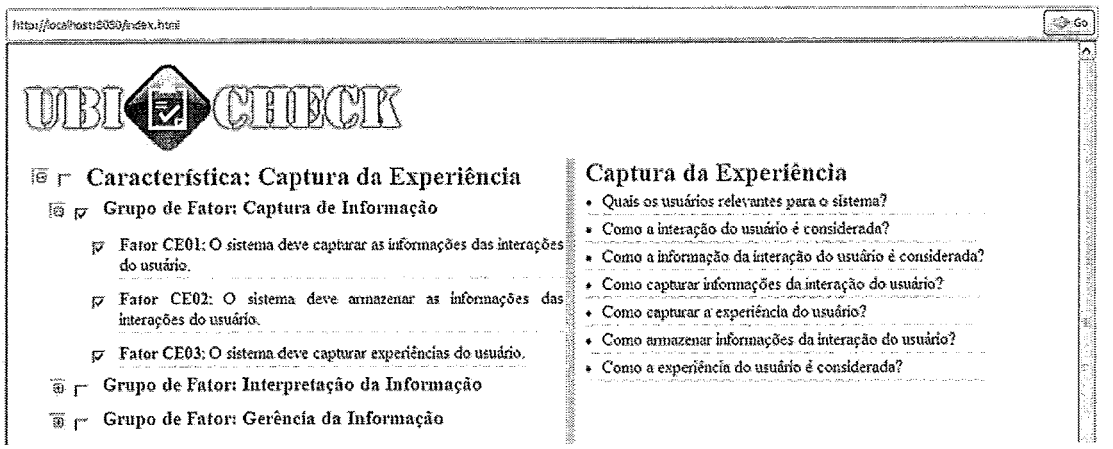


Figura 23: Infra-estrutura computacional mostrando o guia para definição de requisitos de ubiqüidade a partir da seleção de alguns fatores.

Com vistas a explicar como esse protótipo de infra-estrutura foi elaborado, na seção 7.2 são apresentados os requisitos e casos de uso da solução. Em seguida, na seção 0 é apresentada a infra-estrutura computacional de apoio. Por fim, na seção 7.4 este capítulo é concluído.

7.2. Requisitos e Casos de Uso

Essa seção descreve os requisitos e os casos de uso utilizados para especificar a infra-estrutura computacional de apoio a configuração e aplicação de *UbiCheck* 2.0.

7.2.1. Requisitos

Os requisitos da infra-estrutura computacional proposta foram definidos visando apoiar as duas etapas de *UbiCheck* 2.0*: a* configuração da abordagem e a aplicação em projetos de software ubíquo. Adicionalmente, eles foram identificados com base nas atividades apresentadas nos capítulos 4 e 6.

Configuração de *UbiCheck*

REQ 01. A infra-estrutura computacional deve prover apoio para a modelagem das características de ubiqüidade.

REQ 02. A infra-estrutura computacional deve verificar a consistência dos modelos elaborados para cada característica de ubiqüidade através das informações descritas no meta-modelo de ubiqüidade.

REQ 03. A infra-estrutura computacional deve construir a árvore de elementos de uma característica de ubiqüidade a partir do seu respectivo modelo. Nesse sentido, ela deve analisar os relacionamentos do modelo e aplicar os critérios de precedência definidos no capítulo anterior; preencher a árvore de elementos e aplicar as transformações consideradas necessárias.

REQ 04. A infra-estrutura computacional deve identificar a partir de cada nó da árvore de elementos a tríade assunto, escopo e tipo que será utilizada na elaboração das perguntas do Guia para Definição de Requisitos de Ubiqüidade.

REQ 05. A infra-estrutura computacional deve permitir a elaboração e o armazenamento das perguntas do Guia para Definição de Requisitos de Ubiqüidade.

REQ 06. A infra-estrutura computacional deve permitir que cada pergunta seja classificada de acordo com a sua aplicabilidade na etapa dos requisitos ou em etapas futuras do desenvolvimento do software.

REQ 07. A infra-estrutura computacional deve permitir a elaboração de direcionamentos para as respostas das perguntas.

REQ 08. A infra-estrutura computacional deve permitir a visualização do Guia para Definição de Requisitos de Ubiqüidade.

REQ 09. A infra-estrutura computacional deve apoiar a geração do glossário de termos utilizados nas perguntas.

Aplicação de *UbiCheck*

REQ 10. A infra-estrutura computacional deve permitir a seleção de fatores relevantes para um projeto.

REQ 11. A infra-estrutura computacional deve permitir que o Guia para Definição de Requisitos de Ubiqüidade seja configurado de acordo com as necessidades de um projeto.

REQ 12. A infra-estrutura computacional deve permitir a visualização do Guia para Definição de Requisitos de Ubiqüidade configurado para um projeto.

7.2.2. Casos de Uso e Formas de Utilizar a Infra-estrutura Computacional

Essa subseção apresenta os casos de uso da infra-estrutura computacional. Esses casos de uso foram escritos propositalmente com o nível de detalhe necessário para que o usuário possa utilizar a aplicação. Dessa forma, ele compõe também um manual para o usuário. Foram elaborados os seguintes casos de uso, os quais são descritos em seguida.

- Elaborar Perguntas do Guia para Definição de Requisitos de Ubiqüidade: neste caso de uso o usuário importa e valida os modelos de ubiqüidade, gera a tríade assunto, escopo e tipo de cada pergunta, elabora as perguntas, classifica cada uma de acordo com a sua aplicabilidade na etapa de definição de requisitos e elabora direcionamentos para as respostas.
- Elaborar Glossário de Termos do Guia para Definição de Requisitos de Ubiqüidade: neste caso de uso o usuário prepara o glossário de termos do guia para definição de requisitos de ubiqüidade.
- Aplicar Guia para Definição de Requisitos de Ubiqüidade: neste caso de uso o usuário configura e aplica o guia para definição de requisitos de ubiqüidade em um projeto.

7.2.2.1. Elaborar Perguntas do Guia para Definição de Requisitos de Ubiqüidade

Fluxo Principal:

Esse caso de uso tem início quando o especialista em computação ubíqua termina de elaborar os modelos de ubiqüidade e continua a partir do seguinte fluxo:

1. O especialista exporta o modelo para o formato XMI (OMG, 2001) e utiliza a infra-estrutura computacional para validar, importar e processar o modelo. Esse trabalho é feito através do comando a seguir:

```
java -jar ubicheck.jar
          -m <caminho para o arquivo xmi>
          -s <diretório de trabalho>
```

2. A infra-estrutura computacional valida o modelo, importa o modelo e para cada característica modelada, gera uma planilha eletrônica no diretório de trabalho, a qual possui as seguintes colunas:
    - Fatores: os fatores relacionados aquela pergunta (essa informação é importante para manter a rastreabilidade entre os fatores e as perguntas)
    - Tipo: o tipo da pergunta, conforme descrito no capítulo 4; e
    - Árvore de Elementos: de onde se extraí o escopo e o assunto da pergunta, conforme descrito no capítulo 4.

    E as seguintes colunas para serem preenchidas pelo especialista:
    - Pergunta: a pergunta que irá compor o guia para definição de requisitos de ubiqüidade, descrita conforme o padrão descrito no capítulo 4;
    - Aplicabilidade em Requisitos: indicador que representa se a pergunta é aplicável (S) ou não (N) na etapa de requisitos. Apenas aquelas marcadas com (S) serão incluídas no guia para definição de requisitos de ubiqüidade; e
    - Direcionamento: texto para explicitar o que é esperado como resposta desta pergunta, conforme descrito no capítulo 6.
3. O especialista abre cada uma das planilhas geradas e preenche as colunas: *Pergunta, Aplicabilidade em Requisitos* e *Direcionamento*. Depois, ele utiliza o comando a seguir para gerar e armazenar o guia para definição de requisitos de ubiqüidade a partir das planilhas editadas:

```
java -jar ubicheck.jar
          -i <planilha 1>
          -i <planilha 2>
          -i ...
          -i <planilha N>
          -o <caminho para armazenar o guia para
          definição de requisitos de ubiqüidade>
```

4. A infra-estrutura computacional armazena o guia para definição de requisitos de ubiqüidade e este caso de uso é finalizado.

7.2.2.2. Elaborar Glossário de Termos do Guia para Definição de Requisitos de Ubiqüidade

Fluxo Principal:

Esse caso de uso tem início quando o especialista em computação ubíqua termina de elaborar os modelos de ubiqüidade e continua a partir do seguinte fluxo:

1. O especialista exporta o modelo para o formato XMI (OMG, 2001) e utiliza a infra-estrutura computacional para gerar a lista de termos que devem constar no glossário, o que é feito através do comando a seguir:

```
java -jar ubicheck.jar
          -m <caminho para o arquivo xmi>
          -g <caminho para o glossário>
```

2. A infra-estrutura computacional processa o modelo e gera uma planilha com todos os termos que devem ser definidos.
3. O especialista elabora uma definição para cada termo e este caso de uso é finalizado.

7.2.2.3. Aplicar Guia para Definição de Requisitos de Ubiqüidade

Fluxo Principal:

Esse caso de uso tem início quando o desenvolvedor pretende utilizar a *UbiCheck* 2.0 para apoiar na definição de requisitos de ubiqüidade. O fluxo continua a seguir:

1. O desenvolvedor acessa o endereço eletrônico da infra-estrutura computacional através de um browser.
2. A infra-estrutura computacional apresenta para o desenvolvedor a lista de fatores, agrupados por características e por grupo de fatores.
3. O desenvolvedor seleciona os fatores que considerados pertinentes para o projeto.
4. A infra-estrutura apresenta o guia para definição de requisitos de ubiqüidade configurado para as necessidades daquele projeto. Esse caso de uso é finalizado nesta atividade.

7.3. Infra-estrutura Computacional

A partir dos requisitos e dos casos de uso definidos na seção anterior, foi projetada a infra-estrutura computacional para apoiar a utilização de *UbiCheck* 2.0. Essa infra-estrutura foi elaborada com base em dois aplicativos desenvolvidos. Um aplicativo ficou responsável pela configuração de *UbiCheck* 2.0 e o outro ficou responsável pela aplicação da abordagem em projetos de software ubíquo. Cada um foi respectivamente chamado de ***UbiCheck*-Config** e ***UbiCheck-Aplique.***

Essa seção descreve esses aplicativos, sendo que na subseção 7.3.1 é apresentado o diagrama de classes da infra-estrutura computacional e na subseção 0 são apresentados os componentes que constituem a solução.

7.3.1. Diagrama de Classes

O modelo de classes conceitual da infra-estrutura computacional é apresentado na figura 24. A partir dele é possível interpretar que uma *característica* de ubiqüidade pode possuir vários *grupos de fatores* que por sua vez podem possuir vários *fatores*. Os *fatores* de uma característica estão associados a *elementos* de um *modelo,* que por sua

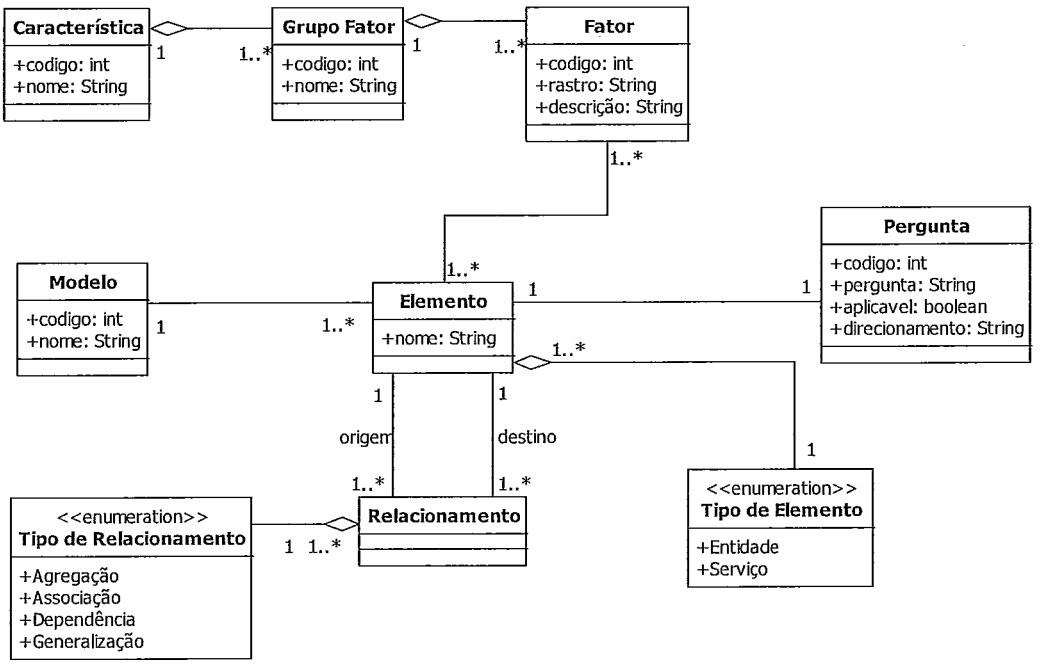


Figura 24: Modelo de classes da infra-estrutura computacional

vez, possuem relacionamentos com outros *elementos*. Cada *elemento* possuí uma *pergunta* associada. Através de navegação entre as entidades mapeadas é possível perceber que os *Fatores* estão associados com *Perguntas*, de forma que um fator pode ter várias perguntas associadas e uma pergunta pode representar vários fatores.

7.3.2. Componentes

Para desenvolver ***UbiCheck*-Config** e ***UbiCheck*-Aplique** foram elaborados alguns componentes. Conforme mostra a figura 25, ***UbiCheck*-Config** é composta pelos componentes de modelagem, elaboração do guia para definição de requisitos de ubiqüidade e persistência. Já ***UbiCheck*-Aplique** é composto pelo componente de configuração do guia para definição de requisitos de ubiqüidade, o qual se comunica com o componente de persistência de ***UbiCheck*-Config**. Esses componentes são explicados a seguir.

7.3.2.1. Componente de Modelagem

O componente de modelagem tem por objetivo importar modelos de características de ubiqüidade e processá-los. Para importar os modelos, foi utilizada a linguagem *XML Metadata Interchange* (XMI) (OMG, 2001) que é um padrão para trocar informações de modelos e está disponível na maior parte das ferramentas CASE

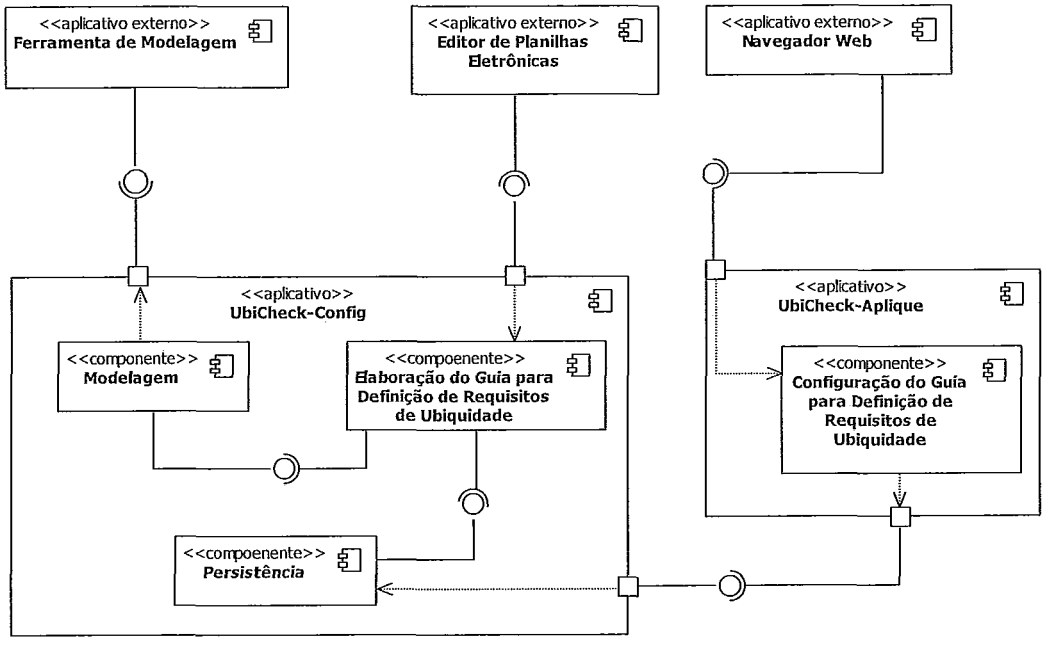


Figura 25: Arquitetura da infra-estrutura computacional de apoio a *UbiCheck*

para apoio a modelagem UML.

Utilizar XMI para importar os modelos foi uma boa estratégia, pois muitos editores de modelos UML trabalham com esse padrão e, ao mesmo tempo, permite que a infra-estrutura computacional seja independente de uma ferramenta específica. Dessa forma, na maior parte das vezes, o especialista poderá utilizar a ferramenta de sua preferência para construir os modelos.

Sendo assim, esse componente ficou com o papel de integrar as diferentes ferramentas de modelagem com a infra-estutura computacional proposta, o que foi feito através do uso da linguagem XMI. Esse componente também ficou responsável por verificar a consistência do modelo com o meta-modelo de ubiqüidade.

O XMI é um *Extensive Markup Language* (XML) com o propósito de armazenar informações sobre modelos UML. A seguir é apresentado, com a finalidade de exemplo, um trecho de um modelo representado nessa linguagem.

```
<UML:Model xmi.id="UMLModel.2" name="Captura da Experiência"
visibility="public" isSpecification="false" namespace="UMLProject.1"
isRoot="false" isLeaf="false" isAbstract="false">

   <UML:Namespace.ownedElement>

      <UML:Class xmi.id="UMLClass.3" name="Usuário
      (CE01,CE02,CE03,CE04,CE08,CE09,CE10)" visibility="public"
      isSpecification="false" namespace="UMLModel.2" isRoot="false"
      isLeaf="false" isAbstract="false" participant="UMLAssociationEnd.7
      UMLAssociationEnd.48 UMLAssociationEnd.51 UMLAssociationEnd.54
      UMLAssociationEnd.67" isActive="false" />

      <UML:Class xmi.id="UMLClass.4" name="Interação (CE01,CE02,CE04)"
      visibility="public" isSpecification="false" namespace="UMLModel.2"
      isRoot="false" isLeaf="false" isAbstract="false"
      participant="UMLAssociationEnd.6 UMLAssociationEnd.11
      UMLAssociationEnd.27" isActive="false" />

      <UML:Association xmi.id="UMLAssociation.5" name="" visibility="public"
      isSpecification="false" namespace="UMLModel.2">

         <UML:Association.connection>

            <UML:AssociationEnd xmi.id="UMLAssociationEnd.6" name=""
            visibility="public" isSpecification="false" isNavigable="true"
            ordering="unordered" aggregation="none" targetScope="instance"
            changeability="changeable" association="UMLAssociation.5"
            type="UMLClass.4" />

            <UML:AssociationEnd xmi.id="UMLAssociationEnd.7" name=""
            visibility="public" isSpecification="false" isNavigable="true"
            ordering="unordered" aggregation="aggregate" targetScope="instance"
            changeability="changeable" association="UMLAssociation.5"
            type="UMLClass.3"/>

         </UML:Association.connection>

   </UML:Association>

   ... ... ...

<UML:Model>
```

Sendo assim, foi elaborado um leitor XMI que interpreta o arquivo de um modelo e o carrega para memória, o que é feito através do mapeamento das *tags* do arquivo com os elementos do modelo de entidade de negócios.

Uma vez que o modelo foi mapeado, as entidades e relacionamentos são verificadas quanto a sua consistência com o meta-modelo e as informações pertinentes estão prontas para serem utilizadas pela infra-estrutura computacional.

7.3.2.2. Componente de Elaboração do Guia para Definição de Requisitos de Ubiqüidade

O componente de Elaboração do Guia para Definição de Requisitos de Ubiqüidade tem por objetivo gerar as informações necessárias para que o pesquisador possa construir o Guia para Definição de Requisitos de Ubiqüidade. Nesse sentido, ele gera uma planilha para cada característica de ubiqüidade com as seguintes informações:

- a árvore de elementos, que permite a seleção do assunto e do escopo de cada pergunta;
- os fatores que deram origem a cada nó da árvore de elementos; e
- o tipo da pergunta que deve ser elaborada.

A planilha gerada é semelhante a apresentada na tabela 10. A partir dela e das instruções fornecidas nos capítulos 4 e 6, o pesquisador pode gerar perguntas de forma semelhante às apresentadas na tabela 6, disponível na página 55.

Tabela 10: Exemplo de planilha gerada pela infra-estrutura computacional

Fatores	Tipo	Árvore de Elementos	Pergunta	Aplicabilidade	Direcionamento
SC04,(...) e SC26	DIR.	Informação de Contexto			
SC05	GEN	- - Informação do Sistema			
SC17	ASS	- - Categorizar			
SC10,SC14 e SC17	DEP	- - - - Fonte de Dados			

Uma vez que a planilha foi gerada, o pesquisador deve preencher as perguntas e ao terminar, ele a importa para que o Guia para Definição de Requisitos de Ubiqüidade possa ser armazenado.

Esse componente também é utilizado para gerar a lista de termos que devem compor o glossário.

As planilhas geradas utilizam o formato *comma-separated values* (CSV) (SHAFRANOVICH, 2005), que consiste em um arquivo de texto com as colunas delimitadas por vírgula. Esse formato é compatível com a maioria dos editores de planilha eletrônica disponíveis, o que permite que o pesquisador escolha o de sua preferência.

Adicionalmente, é um formato que é fácil de interpretar, o que facilita a elaboração de um programa para consumi-lo. Nesse sentido, a planilha preenchida com as perguntas do Guia para Definição de Requisitos de Ubiqüidade é processada e suas informações são mapeadas para os elementos do modelo de entidades de negócio.

7.3.2.3. Componente de Persistência

O componente de persistência tem por objetivo armazenar o Guia para Definição de Requisitos de Ubiqüidade e restaurá-lo quando solicitado, por exemplo, quando houver a necessidade de configurá-lo para um projeto. Sendo assim, esse componente guarda em um arquivo texto as informações mapeadas nos elementos do modelo de entidades de negócio e, quando necessário, as recupera a partir do mesmo arquivo.

7.3.2.4. Componente de Configuração do Guia para Definição de Requisitos de Ubiqüidade

O componente de Configuração do Guia para Definição de Requisitos de Ubiqüidade tem o propósito de preparar um *checklist* para ser utilizado em um projeto de software ubíquo. Ele é um componente WEB, portanto, precisa ser visualizado através de um navegador *WEB*.

Ele oferece uma interface simples, onde na esquerda da tela do navegador, o pesquisador tem a lista de fatores agrupados por característica de ubiqüidade e por grupo de fator. O pesquisador pode selecionar fatores individualmente, um grupo de fator inteiro ou até uma característica de ubiqüidade inteira.

Enquanto o pesquisador seleciona os fatores, no lado direito da tela aparecem as perguntas do Guia para Definição de Requisitos de Ubiqüidade configurado para aquela seleção de fatores.

7.4. Conclusão

Este capítulo apresentou a infra-estrutura computacional desenvolvida para apoiar o uso da abordagem *UbiCheck* 2.0*:* Essa infra-estrutura é composta por dois aplicativos, ***UbiCheck-Config*** e ***UbiCheck-Aplique***. O primeiro tem o propósito de apoiar na configuração da abordagem e o segunda tem o propósito de auxiliar no uso da abordagem em projetos de software ubíquo.

O uso dessa infra-estrutura computacional deve reduzir o esforço necessário para utilizar a abordagem *UbiCheck* 2.0*,* pois grande parte das atividades são automatizadas. Adicionalmente, a automação proporcionada pode evitar erros no processo, visto que o número de interações manuais é reduzido.

Uma vez que a abordagem *UbiCheck* 2.0 foi definida e suas tarefas mais complicadas foram automatizadas, esse trabalho atingiu o seu objetivo. Sendo assim, no próximo capítulo é apresentada a conclusão dessa dissertação.

Capítulo 8 - Conclusão

Este capítulo apresenta as conclusões dessa dissertação, resumindo sua proposta e apresentando as suas contribuições. Também são discutidas as limitações e as perspectivas futuras de continuação desse trabalho.

8.1. Contribuições

Este trabalho apresentou como características presentes em softwares ubíquos podem ser consideradas no seu desenvolvimento. Na definição de requisitos em particular, verificou-se que as abordagens identificadas a partir de uma revisão da literatura não consideravam todas as características presentes neste tipo de software.

Esse fato sugeriu a falta de apoio adequado na etapa de análise de requisitos de projetos de software ubíquo. Por esse motivo, este trabalho propôs uma abordagem para apoiar a definição de requisitos de ubiqüidade, a qual foi chamada de *UbiCheck*.

Essa abordagem foi desenvolvida com base nas características de ubiqüidade apresentadas no capítulo 2. Nesse sentido, elas foram organizadas em modelos, o que permitiu que seus conceitos e relações pudessem ser destacados. Essas informações foram utilizadas para gerar um *Guia para Definição de Requisitos de Ubiqüidade,* o qual pode ser utilizado para orientar o desenvolvedor nesta etapa do desenvolvimento do software.

Ao final da pesquisa, as principais contribuições obtidas neste trabalho foram:

1. **A definição da abordagem *UbiCheck,* para apoiar a definição de requisitos de ubiqüidade em projetos de software.** Nesse sentido foi elaborado um processo para formalizar as características de ubiqüidade em um guia para definição de requisitos de ubiqüidade, o qual contém perguntas que explicitam as informações que devem ser capturadas durante a definição de requisitos de ubiqüidade.
2. **A configuração dessa abordagem,** o que permite que ela possa ser aplicada de imediato, sem a necessidade de um especialista no domínio da computação ubíqua para definir os artefatos necessários para utilizar *UbiCheck.*

Além dessas, podem ser notadas outras contribuições que também merecem destaque:

3. **A formalização das características de ubiqüidade em modelos.** Nesse sentido os conceitos e relações descritos nos fatores das características de ubiqüidade foram identificados e a partir deles, modelos foram elaborados. Esses modelos permitem que as informações contidas nestas características possam ser utilizadas de forma mais direta, assim como foi feito em *UbiCheck*.
4. **A realização de um estudo para caracterizar *UbiCheck*** no que diz respeito a sua aplicabilidade em projetos de software ubíquo.
5. **O desenvolvimento de uma infra-estrutura computacional para apoiar a aplicação de *UbiCheck*.** Nesse sentido, foram implementados softwares com o objetivo de automatizar e simplificar a aplicação da abordagem.
6. **A extensão da revisão da literatura realizada por SPINOLA *et al.* (2008b)** para que essa pudesse contemplar o período em que essa dissertação foi elaborada.

8.2. Limitações e Trabalhos Futuros

No decorrer deste trabalho algumas limitações foram identificadas. A primeira diz respeito à conciliação entre os requisitos de ubiqüidade e os demais requisitos do software. Como *UbiCheck* só apóia a definição dos requisitos de ubiqüidade, não há como garantir a consistência desses requisitos com os demais requisitos do software (funcionais e não funcionais).

Nesse sentido, seria interessante que *UbiCheck* fosse combinada com outra abordagem que provesse apoio a definição de requisitos de software tradicionais. Nesta combinação, seria importante que a consistência entre os requisitos definidos pelas duas abordagens fosse verificada.

Outra limitação deste trabalho foi que não houve oportunidade para realizar um estudo para caracterizar a eficiência e eficácia do apoio fornecido por *UbiCheck* durante a definição de requisitos. No entanto, foi realizado um estudo de observação (Capítulo 5) que sugere que a abordagem pode ser viável para apoiar a definição de requisitos em projetos de software ubíquo.

Espera-se, contudo, que os resultados deste trabalho possam representar um passo a mais no desenvolvimento mais eficiente e efetivo de software ubíquo. Dessa forma, possibilitando que alguns dos benefícios almejados pela comunidade de Engenharia de Software possam ser alcançados.

Referências Bibliográficas

ABOWD, G. D., 1999, "Software engineering issues for ubiquitous computing". In: *Proceedings of the 21st international conference on Software engineering*, pp. 75-84, May.

BASILI, V. R., S., G.; LAITENBERGER, O., LANUBILE, 1996, "The Empirical Investigation of Perspective-Based Reading" *Empirical Software Engineering Journal*, n. 1, pp. 133-164

BASILI, V. R., ROMBACH, H. D., 1988, "The TAME project: towards improvement-oriented software environments" *IEEE Transactions on Software Engineering*, v. 14, n.6, pp. 758-773.

BERRY, D. M., CHENG, B. H., ZHANG, J., 2005, "The four levels of requirements engineering for and in dynamic adaptive systems". *11th International Workshop on Requirements Engineering: Foundation for Software Quality, Porto, Portugal.*

BO, C., XIANG-WU, M., JUN-LIANG, C., 2007, "An Adaptive User Requirements Elicitation Framework". *1st Annual International Computer Software and Applications Conference (COMPSAC)*, v.2, pp. 501-502, Beijing, July.

BOEHM, B., BASILI, V. R., 2001, "Software Defect Reduction Top 10 List". *IEEE Computer Society Press*, v. 34, n. 1, pp. 135-137

BOOCH, G., RUMBAUGH, J., JACOBSON, I., 2000, *UML – Guia do Usuário*, Editora Campus.

CHIU, D. K. W., HONG, D., CHEUNG, S. C., KAFEZA, E., 2006, "Adapting Ubiquitous Enterprise Services with Context and Views". *Enterprise Distributed Object Computing Conference (EDOC)*, pp. 391-394, October.

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". In: *Proceedings of the 40th Annual Hawaii International Conference on System Sciences,* pp. 94, Washington, DC, USA.

CLEMENTS, P., BACHMANN, F., BASS, L., GARIAN, D., IVERS, J., LITTLE, R.; NORD, R., STAFFORD, J., 2004, *Documenting Software Architecture.* Addison-Wesley.

CROCKER, E., OVERELL., P. 2008, "Augmented BNF for Syntax Specifications: ABNF". *The Internet Engineering Task Force (IETC), January.*

DARDENNE, A., VAN LAMSWEERDE, A., FICKAS, S, 1993, "Goal-directed requirements acquisition". *Sixth International Workshop on Software Specification and Design (IWSSD)*, pp. 3-50, Amsterdam, Netherlands.

DAVIES, N., GELLERSEN, H. W., 2002, "Beyond prototypes: challenges in deploying ubiquitous systems". *IEEE Pervasive Computing*, v. 1, pp. 26-35, January.

DUCATEL, K., BOGDANOWICZ, M., SCAPOLO, F., LEIJTEN, J., BURGELMAN, J. C., 2003, "Ambient Intelligence: From Vision to Reality". *IST Advisory Group Draft Report*, pp. 45-48.

FERNANDES, P. C. C., 2009, *Ubifex: uma abordagem para modelagem de características de linha de produtos de software sensíveis ao contexto.* COPPE/UFRJ, Rio de Janeiro, RJ, Brasil

FRIDAY, A., DAVIES, N., CATTERALL, E., 2001, "Supporting service discovery, querying and interaction in ubiquitous computing environments" In: *Proceedings of the 2nd ACM international workshop on Data engineering for wireless and mobile access*, pp. 7-13, New York, NY, USA.

GEER, D., 2006, "Nanotechnology: the growing impact of shrinking computers". *IEEE Pervasive Computing*, v. 5, pp. 7-11, January.

GOLDSBY, H., CHENG, B. H. C., 2006, "Goal-Oriented Modeling of Requirements Engineering for Dynamically Adaptive System Requirements Engineering". *14th IEEE International Conference*, pp. 345-346, Minneapolis/St. Paul, MN, September.

HALL, T., BEECHAM, S., RAINER, A., 2002, "Requirements problems in twelve software companies: an empirical analysis". In: *Proceedings of IEE Software*, v. 149, pp. 153-160, October.

HARMON, P. , WATSON, M., 1997, *Understanding UML: the Developer's Guide: with a Webbased Application in Java.* Morgan Kaufmann Publishers.

HONG, D., CHIU, D. K. W., SHEN, V. Y., 2005, "Requirements elicitation for the design of context-aware applications in a ubiquitous environment". In: *Proceedings of the 7th international conference on Electronic commerce (ICEC)*, pp. 590-596, New York, USA.

JIANG, L., EBERLEIN, A., 2007, "Selecting Requirements Engineering Techniques Based on Project Attributes: A Case Study Engineering of Computer-Based Systems". In: *Proceedings of 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, pp. 269-278, March.

JORGENSEN, J. B., BOSSEN, C., 2003, "Requirements engineering for a pervasive health care system". In: *Proceedings of. 11th IEEE International Requirements Engineering Conference*, pp. 55-64, September.

KITCHENHAM, B., 2004, *Procedures for Performing Systematic Reviews*. In: Joint Technical Report Software Engineering Group, Department of Computer Science Keele University, United King and Empirical Software Engineering, National ICT Australia Ltd.

KOGURE, K., HAGITA, N., SUMI, Y., KUWAHARA, N., ISHIGURO, H., "Toward ubiquitous intelligent robotics". In: *Proceedings of International Conference on Intelligent Robots and Systems*, v. 2, pp. 1826-1831, October.

KRIKKE, J., 2005, "T-Engine: Japan's Ubiquitous Computing Architecture Is Ready for Prime Time". *IEEE Pervasive Computing*, v. 4, pp. 4-9.

LAITENBERGER, O., ATKINSON, C., SCHLICH, M., EMAM, K. E., 2000, "An experimental comparison of reading techniques for defect detection in UML design documents". *Journal of Systems and Software*, v.53, n. 2, pp. 183-204.

LAITENBERGER, O., EL EMAM, K., HARBICH, T. G., 2001, "An Internally Replicated Quasi-Experimental Comparison of Checklist and Perspective-Based Reading of Code Documents". IEEE Transactions in Software Engineering, IEEE Press, v. 27, n. 5, pp. 387-421.

LIMA, G. M. P. S., 2005, *Heurísticas para Identificação da Ordem de Integração de Classes em Testes Aplicados a Software Orientado a Objetos*. COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

LIU, X. F., DONG, L., LIN, H., 2001, "High order object-oriented modeling technique for structured object-oriented analysis". *International Journal of Computer and Information Science*, v. 2, pp. 74-96, June.

LOKE, S., 2006, "Context-Aware Pervasive Systems: Architectures for a New Breed of Applications". *Auerbach Publications.*

MAFRA, S. N., 2006, *Definição de uma técnica de leitura baseada em perspectiva (OO-PBR) apoiada por estudos experimentais*. COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

MAFRA, S. N., TRAVASSOS, G., 2005, "Técnicas de Leitura de Software: Uma Revisão Sistemática". *XIX Simpósio Brasileiro de Engenharia de Software*, Uberlândia, Brasil.

MARKOSE, S., LIU, X. F., MCMILLIN, B., 2008, "A Systematic Framework for Structured Object-Oriented Security Requirements Analysis in Embedded Systems". *International Conference on Embedded and Ubiquitous Computing, v.* 1, pp. 75-81, December.

MING, H., OYAMA, K., CHANG, C. K., 2008, "Human-Intention Driven Self Adaptive Software Evolvability in Distributed Service Environments". *12th IEEE International Workshop on* Future Trends of Distributed Computing Systems, pp. 51-57, October.

NIEMELÄ, E., LATVAKOSKI, J., 2004, "Survey of requirements and solutions for ubiquitous software". In: *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, pp. 71-78, New York, NY, USA.

OLIVEIRA, K. M., TRAVASSOS, G. H., MENEZES, C. E. A., 2000, "Ambientes de Desenvolvimento de Software Orientados a Domínio". *XIV Simpósio Brasileiro de Engenharia de Software (SBES)*, Sessão de Ferramentas, Outubro, João Pessoa, Brasil.

OLIVEIRA, R. F., 2006, *Formalização e Verificação de Consistência na Representação de Variabilidades*. COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

OMG., 2009, *Unified Modeling Language*. Object Management Group.

OYAMA, K., JAYGARL, H., XIA, J., CHANG, C. K., TAKEUCHI, A., FUJIMOTO, H., 2008, "Requirements Analysis Using Feedback from Context Awareness

Systems". *32nd Annual IEEE International Computer Software and Applications*, pp. 625-630, July.

PFLEEGER, S. L., 1999, "Albert Einstein and Empirical Software Engineering". *IEEE Computer*, v. 32, n. 10, pp. 32-38.

PFLEEGER, S. L., 2004. *Engenharia de Software: Teoria e Prática.* Pearson Education do Brasil.

PINTO, F. C. D. R., SPINOLA, R., TRAVASSOS, G. H., 2008, "Abordagem para Apoiar a Definição de Requisitos de Software Ubíquo". *II Workshop on Pervasive and Ubiquitous Computing (WPUC)*, Campo Grande, MS, Outubro.

SAKAMURA, K., 2006, "Challenges in the age of ubiquitous computing: a case study of T-Engine, an open development platform for embedded systems". In: *Proceedings of the 28th international conference on Software engineering*, pp. 713-720, New York, NY, USA.

SATYANARAYANAN, M., 2001, "Pervasive computing: vision and challenges". *IEEE Personal Communications*, v. 8, n.4, pp. 10-17, August.

SÖDERSTRÖM, E., ANDERSSON, B., JOHANNESSON, P., PERJONS, E., WANGLER, B., 2002, "Towards a Framework for Comparing Process Modelling Languages". *14th International Conference on Advanced Information Systems Engineering (CAiSE)*, v. 2348, pp. 600-611.

SPÍNOLA, R., 2008, *Arcabouço para Especificação e Garantia da Qualidade de Requisitos de Ubiqüidade em Projetos de Software Ubíquo*, Exame de Qualificação, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

SPÍNOLA, R., SILVA, J., TRAVASSOS, G., 2006, "Towards a Conceptual Framework to Classify Ubiquitous Software Projects". In: *Proceedings of the 8th International Conference on Software Engineering and Knowledge Engineering (SEKE),* San Francisco, USA.

SPÍNOLA, R., SILVA, J., TRAVASSOS, G.H., 2007a, "Characterizing Ubicomp Software Projects through a Checklist". In: *Proceedings of the I Workshop on Pervasive and Ubiquitous Computing.*

SPÍNOLA, R., SILVA, J. & TRAVASSOS, G., 2007b, "Checklist to Characterize Ubiquitous Software Projects". In: *Proceedings of the XXI Brazilian Symposium on Software Engineering*.

SPÍNOLA, R. O., DO R. PINTO, F. C., TRAVASSOS, G. H., 2008a, "Apoio às Atividades de Especificação e Verificação de Requisitos Funcionais de Ubiqüidade em Projetos de Software". *7th International Information and Telecommunication Technologies Symposium*, Foz do Iguaçu, Brasil.

SPÍNOLA, R. O., PINTO, F. C. R., TRAVASSOS, G. H., 2008b, "Supporting Requirements Definition and Quality Assurance in Ubiquitous Software Project". *Leveraging Applications of Formal Methods, Verification and Validation Third International Symposium (ISOLA)*, v. 17, pp. 587-603.

SPÍNOLA, R. O.; PINTO, F. C. R., TRAVASSOS, G. H., 2009, "UbiCheck: An Approach to Support Requirements Definition in the Ubicomp Domain", *25th Symposium On Applied Computing (SAC)*.

SZWARCFILTER, J. L., MARKENZON, L., 1994, *Estrutura de Dados e seus Algoritmos,* LTC Editora.

RUSSEL, D., STREITZ, N., WINOGRAD, T., 2005, "Building disappearing computers". *Communications of the ACM*, pp. 42 – 48.

TRAVASSOS, G. H., GUROV, D., AMARAL, E. A. G. G., 2002, *Introdução à Engenharia de Software Experimental,* COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

XIANG, J., LIU, L., QIAO, W., YANG, J., 2007, "SREM: A Service Requirements Elicitation Mechanism based on Ontology". *31st Annual International Computer Software and Applications Conference (COMPSAC)*, v. 1, pp. 196-203, July.

YANGFAN, H., KEQING, H., CHONG, W., 2005, "Research on semantic Web service-oriented MMFI for complex information registration". *IEEE International Workshop of* Service-Oriented System Engineering *(SOSE)*,pp. 229-234, October.

WEISER, M., 1991, "The Computer for the 21st Century", Scientific American, pp. 94-104.

Anexo A – Protocolo da Revisão de Literatura

1. Introdução

Este anexo tem o propósito de apresentar o protocolo utilizado por SPINOLA *et al.* (2008b) para identificar as abordagens de apoio a definição e verificação de software ubíquo. Neste trabalho essa revisão da literatura foi executada novamente para verificar se novas abordagens haviam surgido no período em que essa dissertação foi elaborada. O resultado foi disponibilizado no final deste anexo.

2. Protocolo

Data de Execução: 01 de Março de 2009

Objetivo: identificar abordagens de apoio a definição e a verificação de requisitos de software ubíquo.

Critérios de seleção de fontes: disponibilidade de consulta através da web; presença de mecanismo de busca através de palavras chaves e que aceite o uso dos operadores booleanos "E" e "OU"; garantia de resultados únicos mediante a busca realizada através do mesmo conjunto de palavras-chave;

Métodos de busca: as fontes serão acessadas através da Internet, portanto, não será realizada busca manual.

Listagem de Fontes: IEEEXplore e Portal ACM.

String de Busca:

1. (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)
2. (ubicomp or "ubiquitous computing" or "pervasive computing" or "ambient intelligence" or ubiquitous or pervasive) and ("requirement verification" or "requirement inspection" or "requirement review") and (approach or methodology or method or guideline or process or systematic)

3. (invisibility) and ("requirement definition" or "requirement elicitation" or "requirement identification" or "requirement analysis")
4. (invisibility) and ("requirement verification" or "requirement inspection" or "requirement review")
5. ("context sensitivity" or "context aware" or "context awareness") and ("requirement definition" or "requirement elicitation" or "requirement identification" or "requirement analysis")
6. ("context sensitivity" or "context aware" or "context awareness") and ("requirement verification" or "requirement inspection" or "requirement review")
7. (adaptability or "adaptable behavior") and ("requirement definition" or "requirement elicitation" or "requirement identification" or "requirement analysis")
8. (adaptability or "adaptable behavior") and ("requirement verification" or "requirement inspection" or "requirement review")
9. ("automated capture" or "experience capture") and ("requirement definition" or "requirement elicitation" or "requirement identification" or "requirement analysis")
10. ("automated capture" or "experience capture") and ("requirement verification" or "requirement inspection" or "requirement review")
11. ("service discovery") and ("requirement definition" or "requirement elicitation" or "requirement identification" or "requirement analysis")
12. ("service discovery") and ("requirement verification" or "requirement inspection" or "requirement review")
13. ("service composition" or "functionality composition" or "function composition") and ("requirement definition" or "requirement elicitation" or "requirement identification" or "requirement analysis")
14. ("service composition" or "functionality composition" or "function composition") and ("requirement verification" or "requirement inspection" or "requirement review")
15. ("spontaneous interoperability" or interoperability) and ("requirement definition" or "requirement elicitation" or "requirement identification" or "requirement analysis")
16. ("spontaneous interoperability" or interoperability) and ("requirement verification" or "requirement inspection" or "requirement review")

17. ("device heterogeneity" or "heterogeneity of devices") and ("requirement definition" or "requirement elicitation" or "requirement identification" or "requirement analysis")

18. ("device heterogeneity" or "heterogeneity of devices") and ("requirement verification" or "requirement inspection" or "requirement review")

19. ("service omnipresence" or "computer everywhere") and ("requirement definition" or "requirement elicitation" or "requirement identification" or "requirement analysis")

20. ("service omnipresence" or "computer everywhere") and ("requirement verification" or "requirement inspection" or "requirement review")

3. Resultados

Trabalhos Selecionados:

Oyama, K.; Jaygarl, H.; Xia, J.; Chang, C. K.; Takeuchi, A. & Fujimoto, H. Requirements Analysis Using Feedback from Context Awareness Systems Computer Software and Applications, 2008. COMPSAC '08. 32nd Annual IEEE International, 2008, 625-630

Trabalhos não selecionados:

BALADRON, C.; AGUIAR, J.; CARRO, B. & SANCHEZ-ESGUEVILLAS, A. Integrating User-Generated Content and Pervasive Communications IEEE Pervasive Computing, 2008, 7, 58-61

MING, H.; OYAMA, K. & CHANG, C. K. Human-Intention Driven Self Adaptive Software Evolvability in Distributed Service Environments Future Trends of Distributed Computing Systems, 2008. FTDCS '08. 12th IEEE International Workshop on, 2008, 51-57

MARKOSE, S.; LIU, X. F. & MCMILLIN, B. A Systematic Framework for Structured Object-Oriented Security Requirements Analysis in Embedded Systems Embedded and Ubiquitous Computing, 2008. EUC '08. IEEE/IFIP International Conference on, 2008, 1, 75-81

RONG, H.; ZHOU, N.; CHEN, H. & CHENG, H. Research on Strategy of Web Service Composition Based on Software Life Cycle Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on, 2008, 1-4

ZENG, P.; HAO, Y.; SHAO, W. & LIU, Y. Towards a Software Integration Framework in Product Collaborative Design Environment Computer Science and Software Engineering, 2008 International Conference on, 2008, 2, 527-530

ZENG, P.; LIU, Y. & HAO, Y. An Adaptive Multidisciplinary Integration Framework for Mechatronic Systems Collaborative Design Mechtronic and Embedded Systems and Applications, 2008. MESA 2008. IEEE/ASME International Conference on, 2008, 341-346

Anexo B – Características de Ubiqüidade

1. Introdução

Este anexo apresenta os fatores das nove características funcionais da computação ubíqua (SPINOLA *et al.* 2007a): Captura de Experiência, Comportamento Adaptável, Composição de Funcionalidades, Descoberta de Serviços, Heterogeneidade de Dispositivos, Interoperabilidade Espontânea, Invisibilidade, Onipresença do Serviço e Sensibilidade ao Contexto. Conforme apresentado no capítulo 3, os fatores foram reescritos na forma de requisitos para que fossem modelados e cada fator recebeu um identificador único. Nas seções a seguir são apresentados os fatores de cada característica.

2. Fatores de Captura da Experiência

Código	Descrição Original	Descrição na forma de Requisitos
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.

Código	Descrição Original	Descrição na forma de Requisitos
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.
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.

3. Fatores de Comportamento Adaptável

Código	Descrição Original	Descrição na forma de Requisitos
Grupo de Fator: Gerência de Adaptações		
CA01	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 sistema deve considerar informações sobre adaptações
CA02		O sistema de 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	O sistema deve prever alterações no ambiente.

Código	Descrição Original	Descrição na forma de Requisitos
CA07	adaptações antes que as alterações ocorram no ambiente	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 informações do ambiente	O sistema deve configurar uma aplicação de acordo com as necessidades do usuário.
CA11		O sistema deve configurar uma aplicação de acordo com as informações do ambiente.
CA12	Antes de executar um serviços, checar: • 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	O sistema deve executar serviços. Para isso deve verificar: 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; e o tempo requerido para uso do serviço em relação ao tempo disponível da próxima instancia 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.

Código	Descrição Original	Descrição na forma de Requisitos
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 conseqüência 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.
CA25	Efetuar ajustes de conteúdo e formato nos dados gerados pelos serviços	O sistema deve adaptar serviços.

4. Fatores de 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.

Código	Descrição Original	Descrição no Novo Formato
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		
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 o estados dos componentes sejam mantidos durante e após a "manutenção" do software	O sistema deve considerar o estado dos componentes.

Código	Descrição Original	Descrição no Novo Formato
CF16	Monitorar a execução dos serviços criados para checar 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.

5. Fatores de 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 um 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 ambigüidade na definição de serviços	O sistema deve considerar ambigüidade na definição dos serviços.
DS04	Lidar com a liberação de um serviço/componente que tem sido utilizado	O sistema deve liberar serviços.

Código	Descrição Original	Descrição no Novo Formato
DS05	Interagir com serviços de endereçamento de serviço	O sistema deve interagir com serviços de endereçamento de serviços.
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.

6. Fatores de Heterogeneidade de Dispositivos

Código	Descrição Original	Descrição no Novo Formato

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 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.

7. Fatores de Interoperabilidade Espontânea

Código	Descrição Original	Descrição no Novo Formato

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.
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.

8. Fatores de 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 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 tempo de execução o menos intrusiva possível	O sistema deve adaptar as aplicações de acordo com o dispositivo.
IN08	Reduzir a interatividade com o usuário	O sistema deve minimizar a interatividade do usuário.

9. Fatores de 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 fazer cachê dos serviços.

10. Fatores de 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, 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	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.

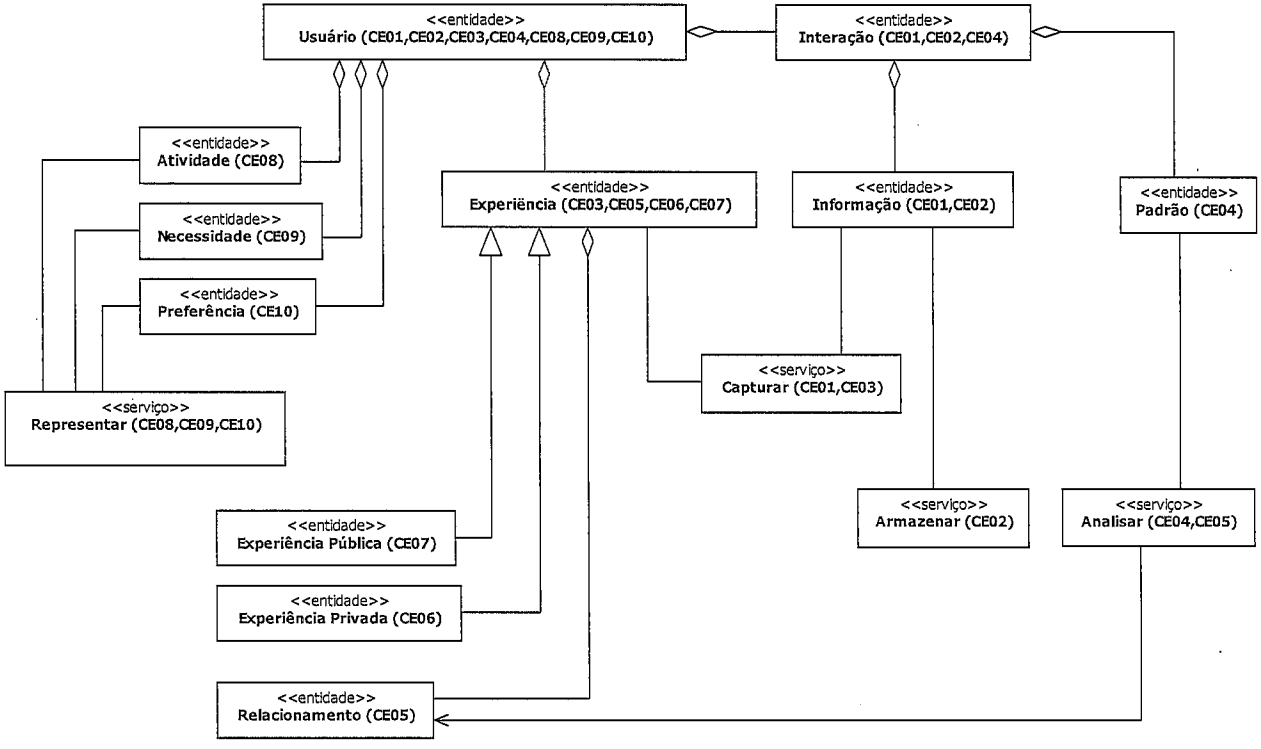
Código	Descrição Original	Descrição no Novo Formato
SC08	Considerar informações sobre usuário: perfil de usuário (preferências, objetivo, calendário do usuário, informações pessoais)	O sistema deve considerar informações de contexto do usuário. Por exemplo: perfil do usuário,preferências, objetivo, calendário 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.
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.

Código	Descrição Original	Descrição no Novo Formato
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.
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.

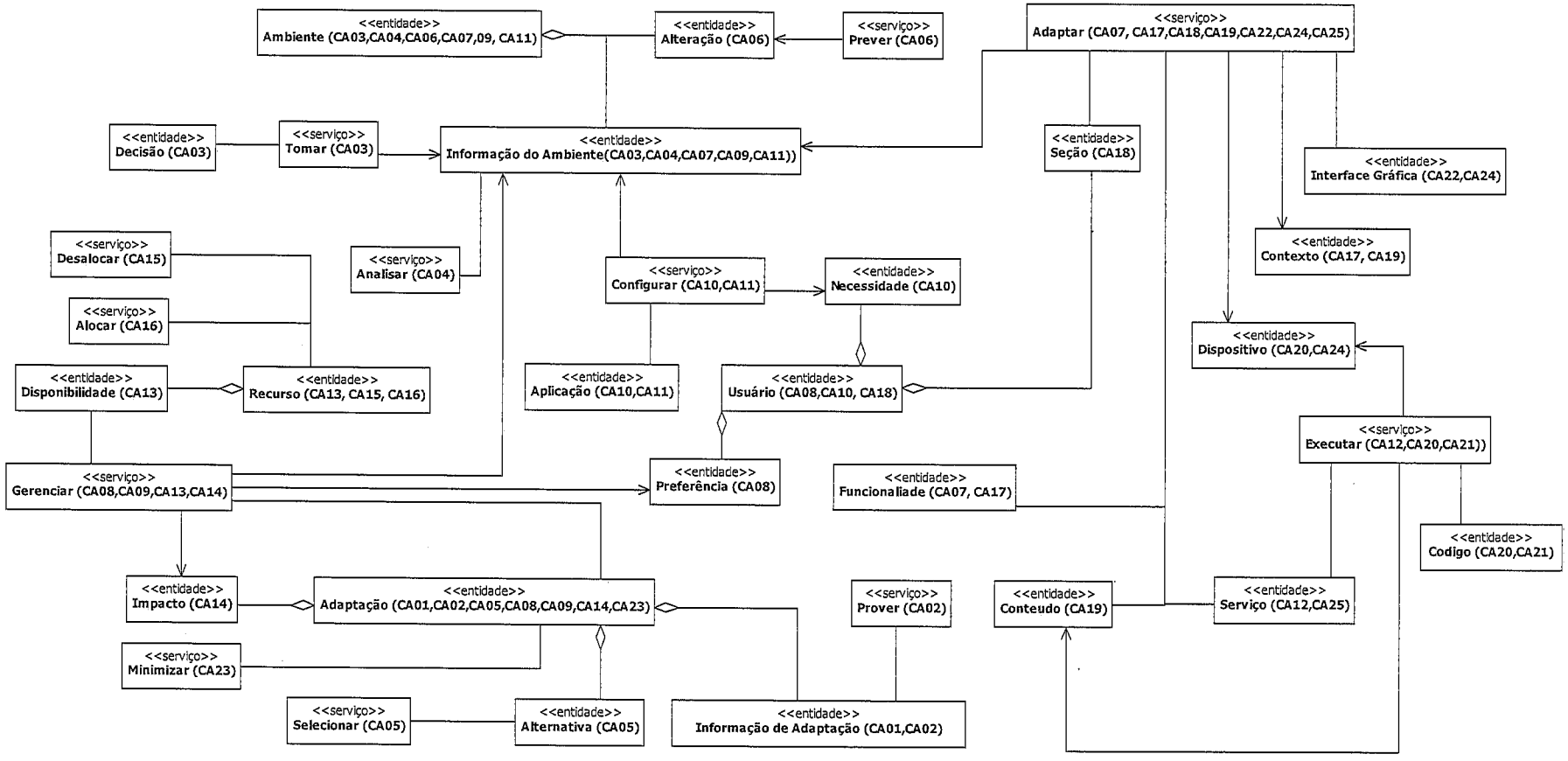
Código	Descrição Original	Descrição no Novo Formato
Grupo de Fator: Compartilhamento da Informação		
SC26	Compartilhar informações de contexto com usuários e outros dispositivos. Disponibilizar para o usuário as informações de contexto como um recurso a ser utilizado para, por exemplo, efetuar configurações no software	O sistema deve compartilhar informações de contexto.

Anexo C – Modelos de Ubiqüidade

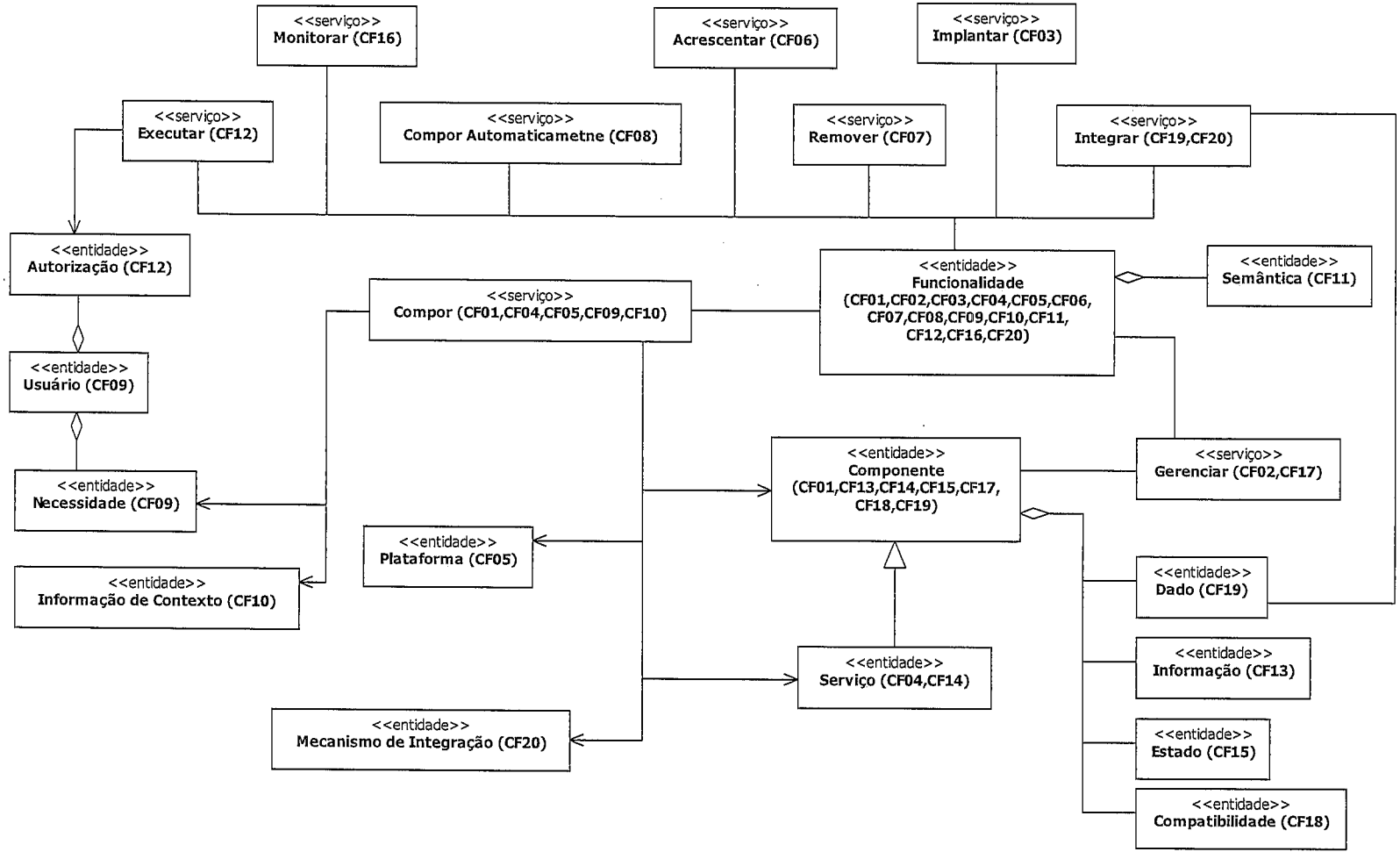
1. Modelo de Captura da Experiência



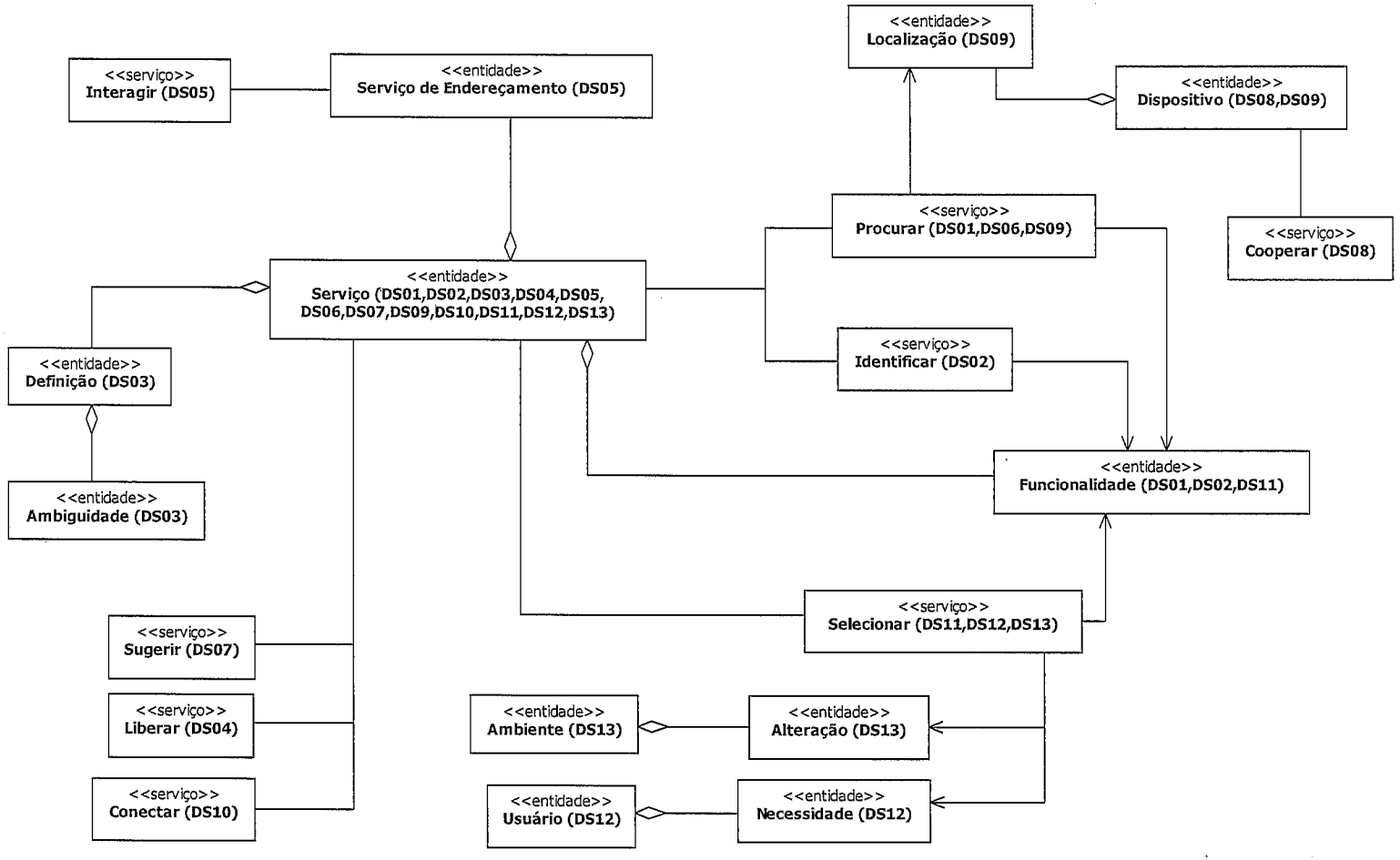
2. Comportamento Adaptável



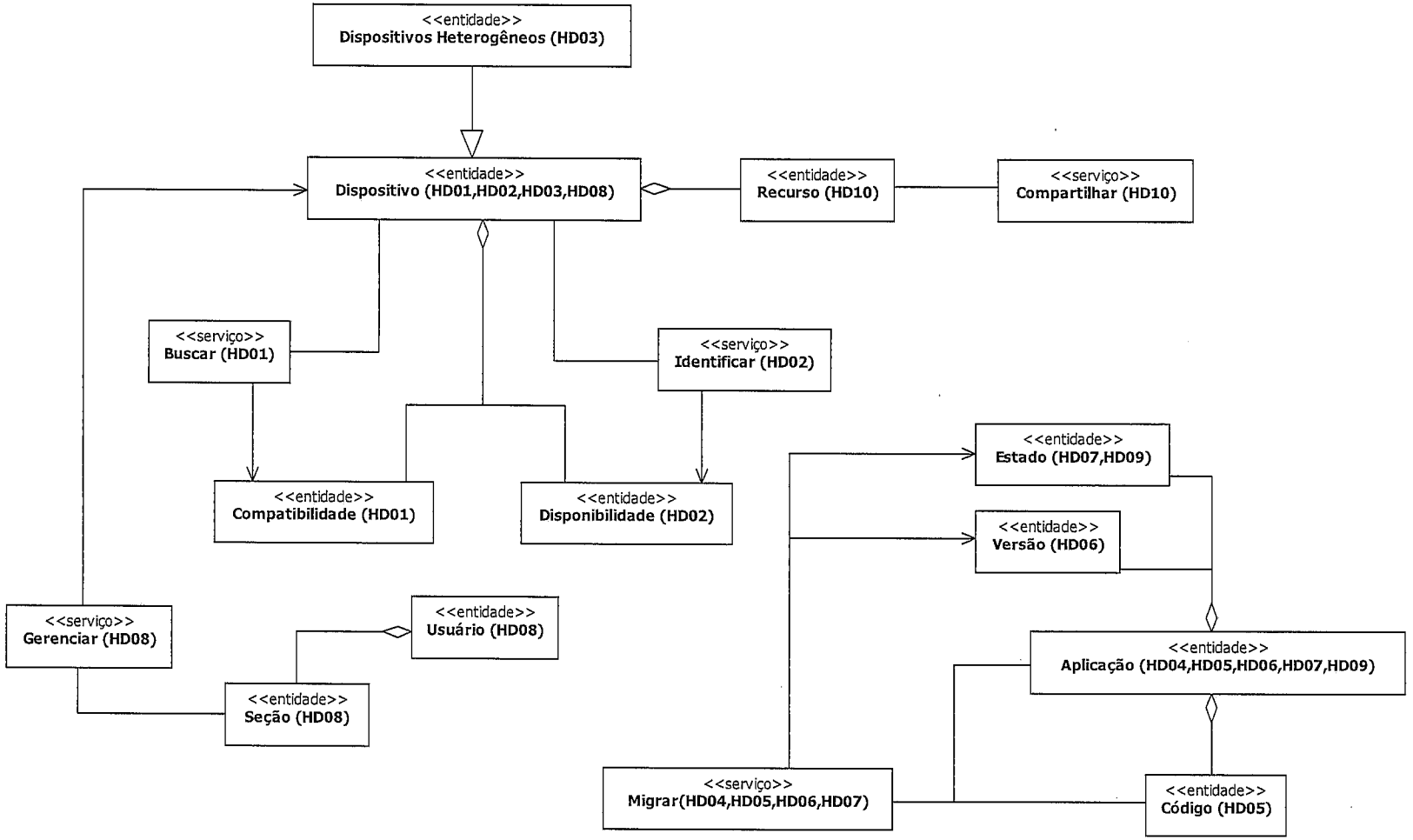
3. Modelo de Composição de Funcionalidades



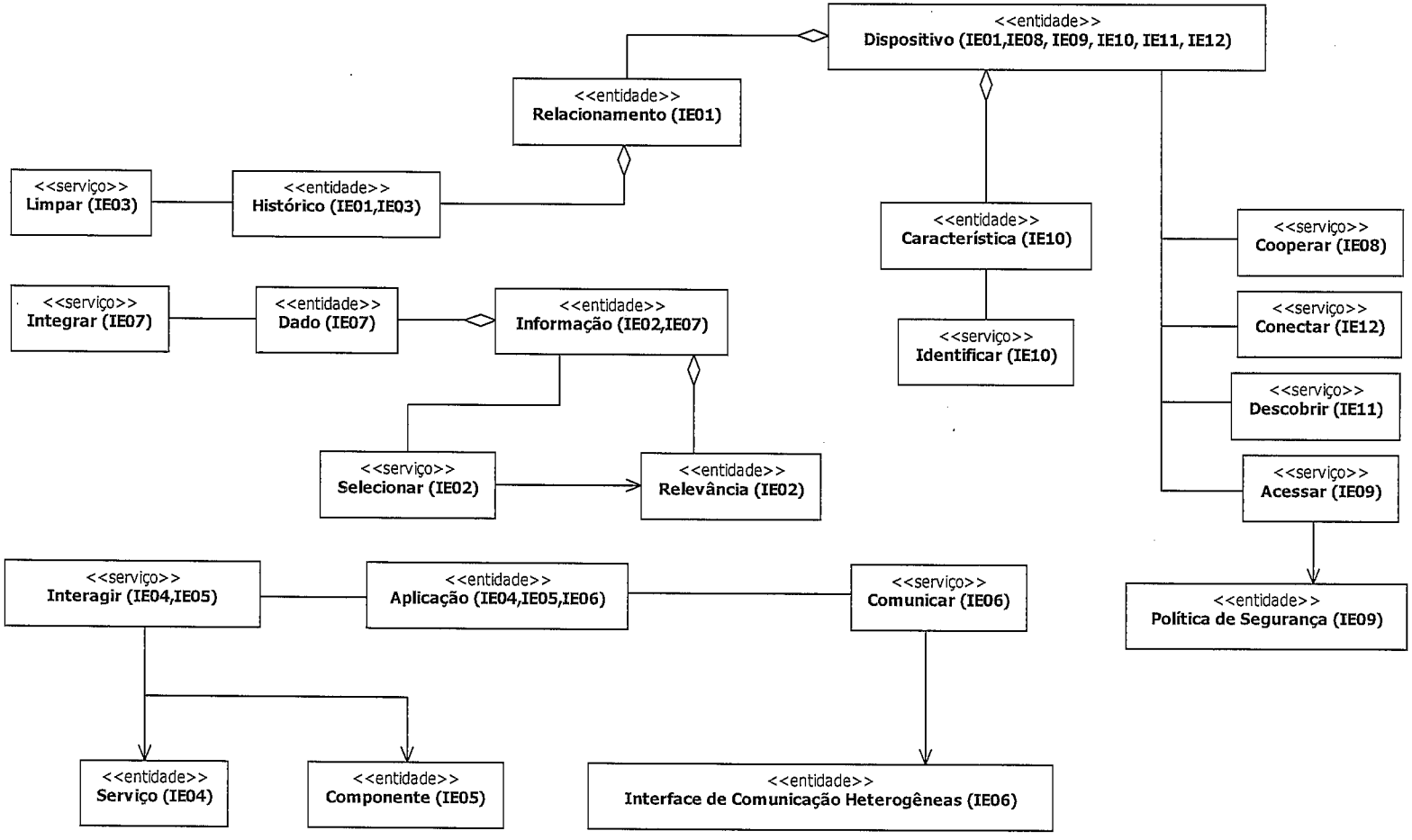
4. Modelo de Descoberta de Serviço



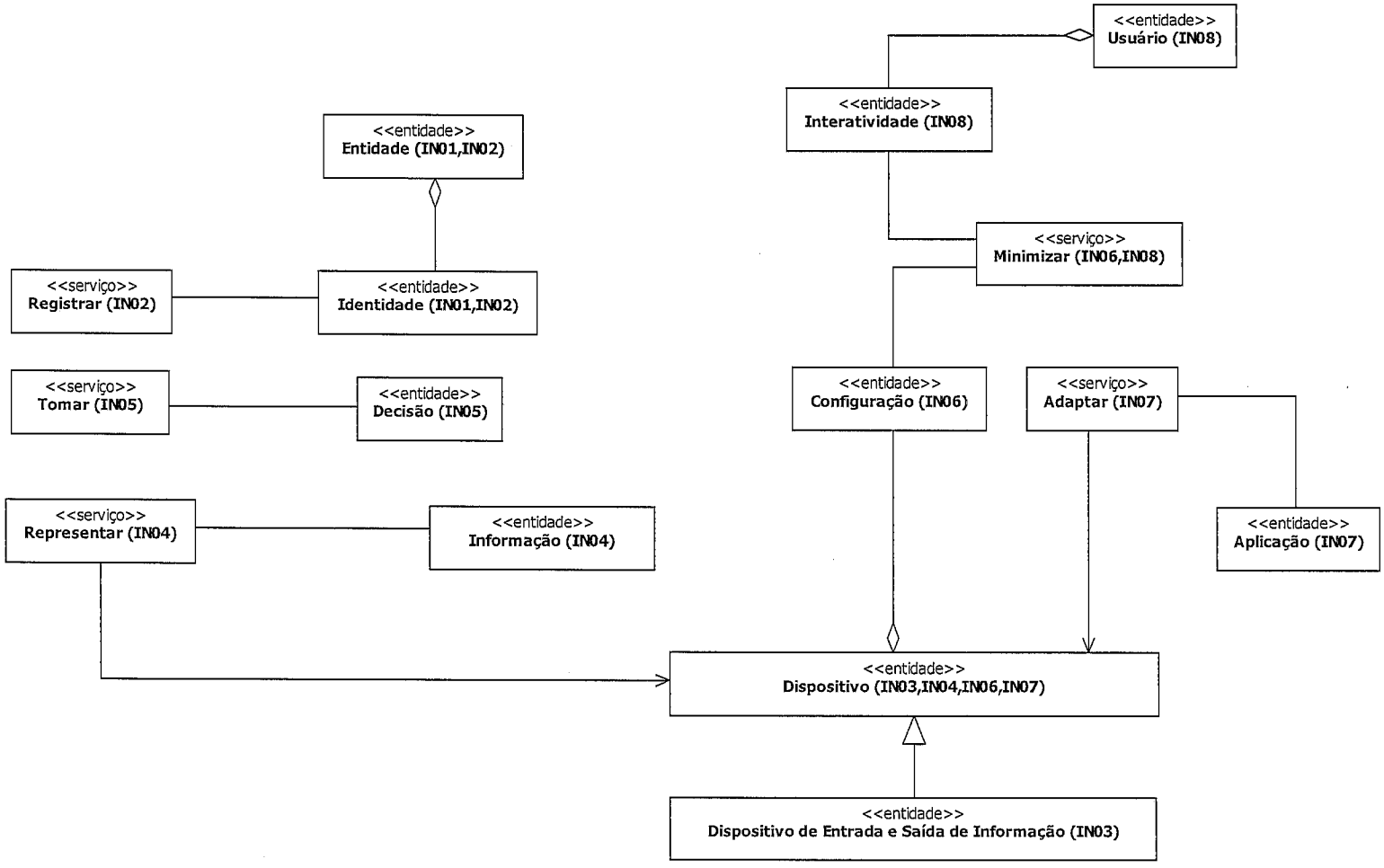
5. Modelo de Heterogeneidade de Dispositivos



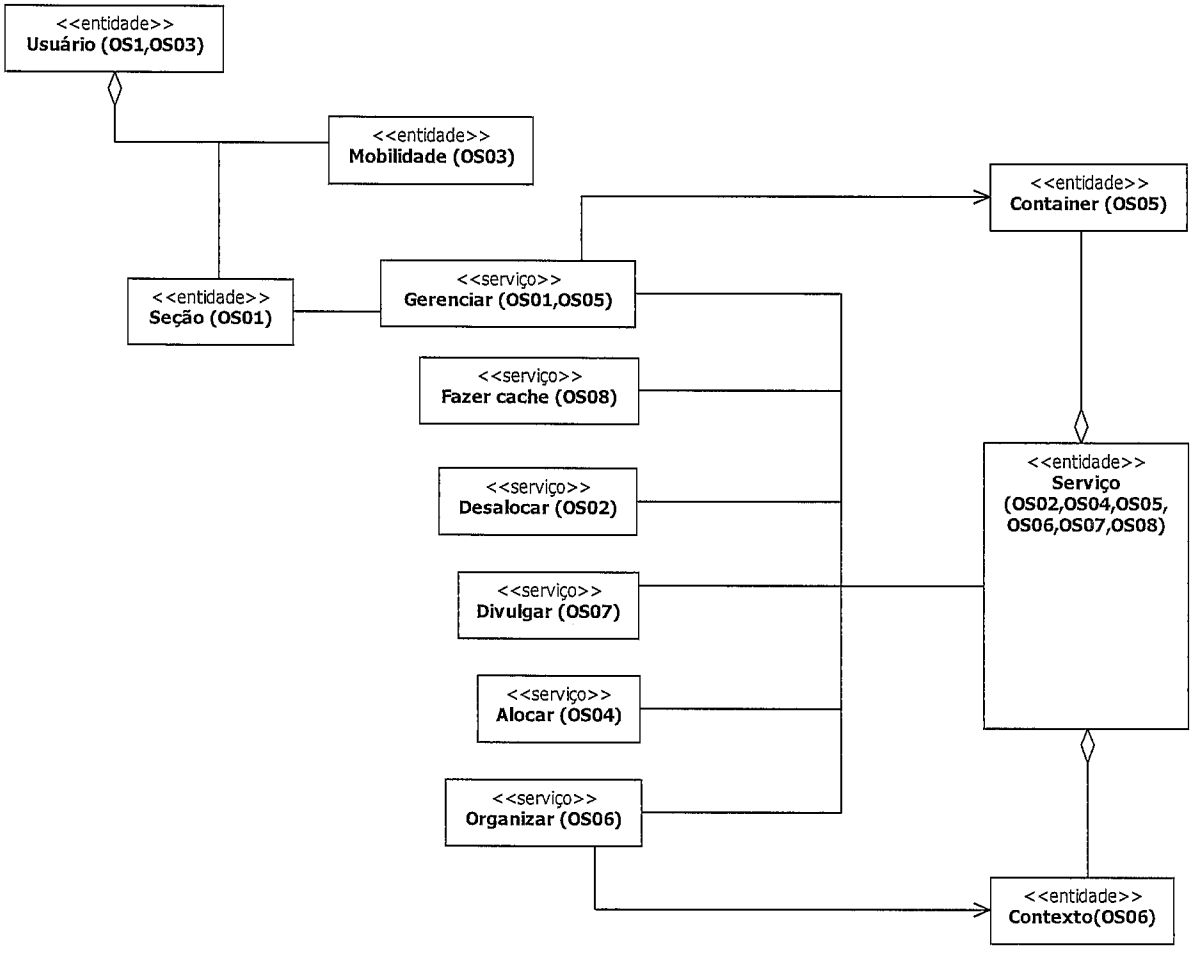
6. Modelo de Interoperabilidade Espontânea



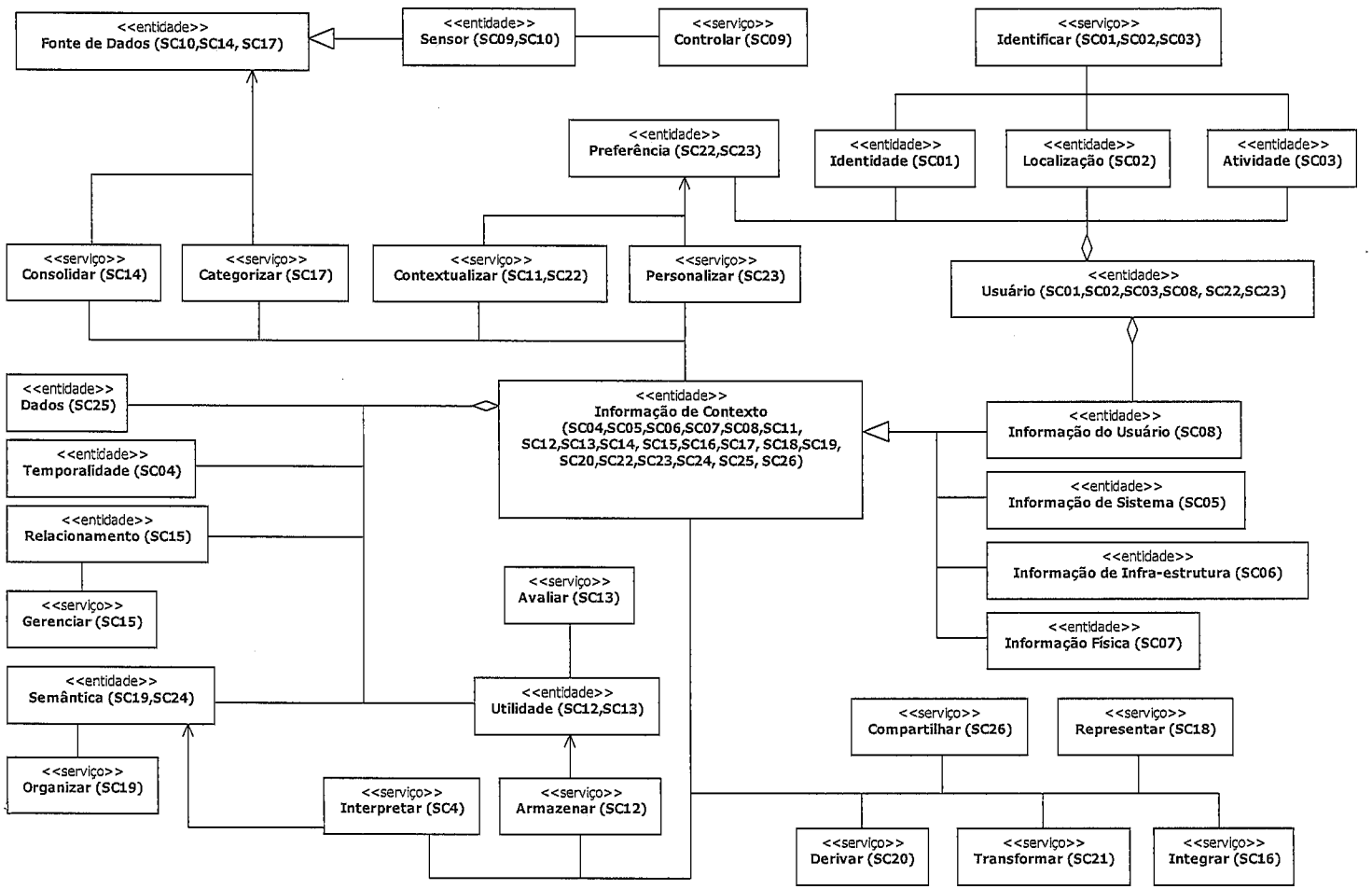
7. Modelo de Invisibilidade



8. Modelo de Onipresença de Serviços



9. Modelo de Sensibilidade ao Contexto



Anexo D – Guia para Definição de Requisitos de Ubiqüidade de *UbiCheck*

1. Questionário de Captura da 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?

Fatores	Tipo	Árvore de Elementos	Pergunta
CE03,CE05,CE06, CE07	AGR	-- Experiëncia	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?

2. Questionário de Comportamento Adaptável

Fatores	Tipo	Árvore de Elementos	Pergunta
CA07,CA17	DIR	Funcionaliade	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?

Fatores	Tipo	Árvore de Elementos	Pergunta
CA03,CA04,CA07, CA09,CA11	DEP	-- -- Informação do Ambiente	Como a informação do ambiente influencia na configuração das aplicações?
CA19	DIR	Conteudo	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	Codigo-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	-- -- Conteudo	Como o conteúdo influencia na execução do código fonte?
CA03,CA04,CA07, CA09,CA11	DIR	Informação do Ambiente	Quais as informações do ambient relevantes para o sistema?
CA04	ASS	-- Analisar	Como analisar as informações do ambiente?
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, CA07,09,CA11	DIR	Ambiente	Quais os ambientes relevantes para o sistema?

Fatores	Tipo	Árvore de Elementos	Pergunta
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	-- -- Conteudo	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?
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?

Fatores	Tipo	Árvore de Elementos	Pergunta
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 Usuáiro	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?

3. Questionário de Composição de Funcionalidades

Fatores	Tipo	Árvore de Elementos	Pergunta
CF01,CF02,CF03, CF04,CF05,CF06, CF07,CF08,CF09, CF10,CF11,CF12, CF16,CF20		Funcionalidade	Quais as funcionalidades relevantes para o sistema?
CF01,CF04,CF05, CF09,CF10	ASS	-- Compor	Como compor funcionalidades?
CF09	DEP	-- -- Necessidade do Usuário	Como a necessidade do usuário influencia na composição de funcionalidades?
CF10	DEP	-- -- Informação de Contexto	Como a informação de contexto influencia na composição de funcionalidade?
CF05	DEP	-- -- Plataforma	Como a plataforma influencia na composição de funcionalidade?
CF01,CF13,CF14, CF15,CF17,CF18, CF19	DEP	-- -- Componente	Como os componentes influenciam na composição de funcionalidade?
CF04,CF14	DEP	-- -- Serviço	Como os serviços influenciam na composição de funcionalidade?
CF20	DEP	-- -- Mecanismo de Integração	Como o mecanismo de integração influencia na composição de funcionalidade?
CF02,CF17	ASS	-- Gerenciar	Como gerenciar as funcionalidades?
CF03	ASS	-- Implantar	Como implantar funcionalidades?
CF06	ASS	-- Acrescentar	Como acrescentar funcionalidades?
CF07	ASS	-- Remover	Como remover funcionalidades?
CF08	ASS	-- Compor Automaticametne	Como compor automaticamente funcionalidades?
CF11	AGR	-- Semântica	Como a semântica da funcionalidade é considerada?

Fatores	Tipo	Árvore de Elementos	Pergunta
CF12	ASS	-- Executar	Como executar funcionalidades?
CF12	DEP	-- -- Autorização do Usuário	Como a autorização do usuário influencia a execução de funcionalidades?
CF16	ASS	-- Monitorar	Como monitorar funcionalidades?
CF19,CF20	ASS	-- Integrar	Como integrar funcionalidades?
CF01,CF13,CF14, CF15,CF17,CF18, CF19		Componente	Quais os componentes relevantes para o sistema?
CF13	AGR	-- Informação	Como a informação dos componentes é considerada?
CF04,CF14	GEN	-- Serviço	Quais componentes são considerados serviços?
CF15	AGR	-- Estado	Como o estado do componente é considerado?
CF02,CF17	ASS	-- Gerenciar	Como gerenciar componentes?
CF19	AGR	-- Dado	Como os dados dos componentes são considerados?
CF19,CF20	ASS	-- -- Integrar	Como integrar dados de componentes?
CF18	AGR	-- Compatibilidade	Como a compatibilidade do componente é considerada?
CF09		Usuário	Quais os usuários relevantes para o sistema?
CF09	AGR	-- Necessidade	Como a necessidade do usuário é considerada?
CF12	AGR	-- Autorização	Como a autorização do usuário é considerada?

4. Questionário de Descoberta de Serviço

Fatores	Tipo	Árvore de Elementos	Pergunta
DS01,DS02,DS03, DS04,DS05,DS06, DS07,DS09,DS10, DS11,DS12,DS13	DIR	Serviço	Quais os serviços relevantes para o sistema?
DS01,DS06,DS09	ASS	-- Procurar	Como procurar por serviços?
DS01,DS02,DS11	DEP	-- -- Funcionalidade	Como a funcionalidade influência na procura por serviços?
DS09	DEP	-- -- Localização do Usuário	Como a localização do usuário influência na procura por serviços?
DS01,DS02,DS11	AGR	-- Funcionalidade	Como a funcionalidade do serviço é considerada?
DS02	ASS	-- Identificar	Como identificar os serviços?
DS01,DS02,DS11	DEP	-- -- Funcionalidade	Como a funcionalidade influencia a identificação do serviço?
DS03	AGR	-- Definição	Como a definição do serviço é considerada?
DS03	AGR	-- -- Ambiguidade	Como a ambiguidade na definição do serviço é considerada?
DS04	ASS	-- Liberar	Como liberar os serviços?
DS05	AGR	-- Serviço de Endereçamento	Como o serviço de endereçamento de serviços é considerado?
DS05	ASS	-- -- Interagir	Como interagir com serviços de endereçamento de serviços?
DS07	ASS	-- Sugerir	Como sugerir serviços?
DS10	ASS	-- Conectar	Como conectar a serviços?
DS11,DS12,DS13	ASS	-- Selecionar	Como selecionar serviços?
DS01,DS02,DS11	DEP	-- -- Funcionalidade	Como a funcionalidade influencia a seleção de serviços?
DS12	DEP	-- -- Necessidade do Usuário	Como a necessidade do usuário influencia na seleção de serviços?

Fatores	Tipo	Árvore de Elementos	Pergunta
DS13	DEP	-- -- Alteração do Ambiente	Como uma alteração do ambiente influencia na seleção de serviços?
DS13	DIR	Ambiente	Quais os ambientes relevantes para o sistema?
DS13	AGR	-- Alteração	Como alterações no ambiente são consideradas?
DS12	DIR	Usuário	Quais os usuários relevantes para o sistema?
DS12	AGR	-- Necessidade	Como as necessidades do usuário são consideradas?
DS08,DS09	DIR	Dispositivo	Quais dispositivos são relevantes para o sistema?
DS08	ASS	-- Cooperar	Como cooperar com dispositivos?
DS09	AGR	-- Localização	Como a localização do dispositivo é considerada?

5. Questionário de Heterogeneidade de Dispositivos

Fatores	Tipo	Árvore de Elementos	Pergunta
HD08	DIR	Usuário	Quais os usuarios relevantes para o sistema?
HD08	AGR	-- Seção	Como a seção dos usuários é considerada?
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?

Fatores	Tipo	Árvore de Elementos	Pergunta
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 dispostivos?
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?

Fatores	Tipo	Árvore de Elementos	Pergunta
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?

6. Questionário de 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?
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?

Fatores	Tipo	Árvore de Elementos	Pergunta
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?

7. Questionário de Invisibilidade

Fatores	Tipo	Árvore de Elementos	Pergunta
IN08	DIR	Usuário	Quais os usuários relevantes para o sistema?
IN08	AGR	-- Interatividade	Como a interatividade do usuário é considerada?
IN06,IN08	ASS	-- -- Minimizar	Como minimizar a interatividade do usuário?
IN04	DIR	Informação	Quais as informações relevantes para o sistema?
IN04	ASS	-- Representar	Como representar as informações do sistema?
IN03,IN04,IN06, IN07	DEP	-- -- Dispositivo	Como dispositivo influencia a representação da informação?

Fatores	Tipo	Árvore de Elementos	Pergunta
IN03,IN04,IN06, IN07	Dir	Dispositivo	Quais dispositivos são relevantes para o sistema?
IN06	AGR	-- Configuração	Como a configuração dos dispositivos é considerada?
IN06,IN08	ASS	-- -- Minimizar	Como minimizar a configuração dos dispositivos?
IN03	GEN	-- Dispositivo de Entrada e Saída de Informação	Quais dispositivos são considerados dispositivos de entrada e saída de informação?
IN07	DIR	Aplicação	Quais aplicações são relevantes para o sistema?
IN07	ASS	-- Adaptar	Como adaptar as aplicações do sistema?
IN03,IN04,IN06, IN07	DEP	-- -- Dispositivo	Como o dispositivo influencia as aplicações do sistema?
IN05	DIR	Decisão	Quais as decisões relevantes para o sistema?
IN05	ASS	-- Tomar	Como o sistema toma as decisões?
IN01,IN02	DIR	Entidade	Quais as entidades relevantes para o sistema?
IN01,IN02	AGR	-- Identidade	Como a identidade das entidades é considerada?
IN02	ASS	-- -- Registrar	Como registrar a identidade de cada entidade?

8. Questionário de 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?

Fatores	Tipo	Árvore de Elementos	Pergunta
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?
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?

9. Questionário de 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?

Fatores	Tipo	Árvore de Elementos	Pergunta
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 relavantes 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?
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?

Fatores	Tipo	Árvore de Elementos	Pergunta
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?

Fatores	Tipo	Árvore de Elementos	Pergunta
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?
SC09	ASS	-- -- Controlar	Como controlar os sensores?

Anexo E – Estudo sobre *UbiCheck*

• Introdução

Este anexo apresenta os artefatos que foram entregues para os participantes do estudo realizado para observar a aplicabilidade de *UbiCheck:*

- *Termo de Consentimento:* documento assinado pelos participantes do estudo autorizando o uso das informações coletadas.
- *Formulário de Caracterização:* documento preenchido pelos participantes para que o nível de conhecimento e experiência de cada um fosse conhecido.
- *Descrição do cenário Controle de Bicicletas:* documento entregue para as equipes para explicar o cenário *Controle de Bicicletas.*
- *Descrição do cenário Controle de Entrada e Saída de Patrimônio*: documento entregue para as equipes para explicar o cenário *Controle de Entrada e Saída de Patrimônio.*

1. Termo de Consentimento

Eu declaro ter mais de 18 anos de idade e que concordo em participar em estudos conduzidos pelo Prof. Guilherme Horta Travassos e pesquisador Felipe Curty do Rego Pinto. Estes estudos visam caracterizar a eficiência e eficácia de uma abordagem para apoiar a definição de requisitos de ubiqüidade, chamada *UbiCheck.*

CONFIDENCIALIDADE

Toda informação coletada neste estudo é confidencial, e meu nome não será identificado em momento algum. Da mesma forma, me comprometo a não comunicar os meus resultados enquanto não terminar o estudo, bem como manter sigilo das técnicas e documentos apresentados e que fazem parte do experimento.

BENEFÍCIOS, LIBERDADE DE DESISTÊNCIA

Eu entendo que os benefícios que receberei deste estudo são limitados ao aprendizado do material que será distribuído e ensinado. Compreendo também que os pesquisadores esperam aprender mais sobre quão eficiente e eficaz é a abordagem que está sendo avaliada, bem como, quais os benefícios trazidos por este estudo para o contexto da Engenharia de Software.

Eu entendo que sou livre para realizar perguntas a qualquer momento ou solicitar que qualquer informação relacionada à minha pessoa não seja incluída no estudo. Eu entendo que minha participação no estudo não afetará minha nota final de qualquer forma, e que participo de livre e espontânea vontade com o único intuito de contribuir para o avanço e desenvolvimento de técnicas e processos para a Engenharia de Software.

PESQUISADORES RESPONSÁVEIS

Prof. Guilherme Horta Travassos
Felipe Curty do Rego Pinto
Programa de Engenharia de Sistemas e Computação
COPPE/UFRJ

Nome (em letra de forma): _____

Assinatura: _____ Data:_____

2. Formulário de Caracterização

Caracterização do Desenvolvedor

Nome _____

Nível de Escolaridade: _____

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

Experiência em Contextos Diferentes

Nós usaremos esta seção para compreender quão familiar você está com vários sistemas que poderão ser utilizados como exemplos ou para exercícios durante o estudo.

Por favor, indique o grau de experiência nesta seção seguindo a escala de 3 pontos abaixo:

1 = Eu não tenho familiaridade com a área. Eu nunca fiz isto.

3 = Eu utilizo isto algumas vezes, mas não sou um especialista.
5 = Eu sou muito familiar com esta área. Eu me sentiria confortável fazendo isto.

Quanto você sabe sobre...

- Elaborar uma lista de compras de supermercado 1 3 5
- Dirigir em um local que não conhece bem 1 3 5

3. Cenário Controle de Bicicletas

Este documento objetiva descrever um novo cenário de uso para o Sistema de Gestão de Inventário Patrimonial. Este cenário trará um novo conjunto de funcionalidades que demandam soluções inseridas no contexto da ubiqüidade computacional.

Antes de apresentar o novo cenário, é importante ter em mente a situação atual do projeto assim como suas limitações do ponto de vista sistêmico. A situação atual pode ser observada na tabela abaixo:

Contexto	Sistema de Gestão de Inventário Patrimonial – SGP
Solução Atual	O sistema objetiva, a partir da web: • permitir que as informações relacionadas aos itens adquiridos por uma organização possam ser registradas e tenham seu histórico de movimentação também armazenado; • permitir que os administradores possam saber, a qualquer instante, qual a localização de um item, bem como onde e quando foi adquirido; • permitir que o valor patrimonial dos itens possa ser calculado, considerando-se a sua depreciação anual.
Documento de Referência	Especificação dos Requisitos do Software definida no documento: Inventario_Patrimonial_Requisitos.pdf

A solução definida inicialmente para o SGP, e encontrada no documento de referência, possibilita o controle do patrimônio através de um sistema web. Entretanto, embora importante, esta solução não contempla uma nova demanda da organização: disponibilizar bicicletas para facilitar o deslocamento dentro da organização.

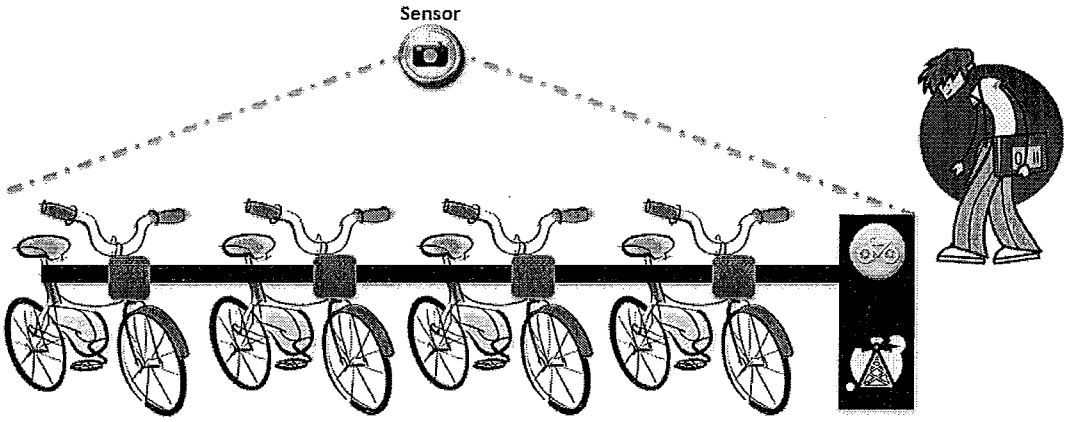
Para lidar com esta nova demanda, as seguintes necessidades adicionais foram definidas e devem ser consideradas no projeto do SGP:

- As bicicletas deverão ser classificadas como item rastreavel. Uma bicicleta rastreavel é aquela que, além do número de patrimônio, possui uma etiqueta eletrônica de identificação;
- Cada indivíduo da organização deve possuir um crachá de identificação, que o identifica como membro da organização. Indivíduos que desejam fazer uso das

bicicletas devem habilitar seu crachá para esta operação. Cada individuo da organização tem direito a utilizar apenas 1 bicicleta de cada vez.

- As bicicletas se localizam em bicicletários. Cada bicicleta se encontra presa por meio de uma trava de segurança que pode ser desbloqueada utilizando o crachá de identificação da organização que tenha sido previamente habilitado. Desta forma, para utilizar uma bicicleta, o indivíduo deverá passar o *crachá* pelo dispositivo de leitura da trava de segurança para que o sistema identifique o indivíduo e verifique se tem permissão para usar uma bicicleta. Em caso positivo, o sistema deve liberar a trava de segurança, registrar o numero da bicicleta, data e hora e o individuo da organização que ficará responsável pela bicicleta até que esta seja devolvida a um dos bicicletários. Em caso negativo, o sistema informa ao individuo da organização o motivo da recusa.
- Ao fazer a devolução de uma bicicleta, o indivíduo deverá apenas conectar a bicicleta à trava de segurança. Ao fazer isso, o sistema identificará a bicicleta devolvida, liberará a responsabilidade do individuo, registrando a data e hora do evento de devolução.

A figura a seguir ilustra um cenário de integração do sistema ao ambiente da organização:



Com base na figura acima e na descrição das necessidades adicionais apresentadas anteriormente, deve-se ter a seguinte descrição em mente:

1. Imaginemos um ponto de bicicleta. Devem existir sensores instalados para identificação das bicicletas e também um dispositivo de leitura 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 *crachá* no dispositivo de leitura, 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 marcará sua saída no 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.

4. Cenário Controle de Entrada e Saída de Patrimônio

Este documento objetiva descrever um novo cenário de uso para o Sistema de Gestão de Inventário Patrimonial. Este cenário trará um novo conjunto de funcionalidades que demandam soluções inseridas no contexto da ubiqüidade computacional.

Antes de apresentar o novo cenário, é importante ter em mente a situação atual do projeto assim como suas limitações do ponto de vista sistêmico. A situação atual pode ser observada na tabela abaixo:

Contexto	Sistema de Gestão de Inventário Patrimonial – SGP
Solução Atual	O sistema objetiva, a partir da web: • permitir que as informações relacionadas aos itens adquiridos por uma organização possam ser registradas e tenham seu histórico de movimentação também armazenado; • permitir que os administradores possam saber, a qualquer instante, qual a localização de um item, bem como onde e quando foi adquirido; • permitir que o valor patrimonial dos itens possa ser calculado, considerando-se a sua depreciação anual.
Documento de Referência	Especificação dos Requisitos do Software definida no documento: Inventario_Patrimonial_Requisitos.pdf

A solução definida inicialmente para o SGP, e encontrada no documento de referência, possibilita o controle do patrimônio através de um sistema web. Entretanto, embora importante, esta solução traz um conjunto de limitações do ponto de vista do controle dos itens de patrimônio gerenciados uma vez que não há um controle efetivo sobre seu transporte e localização. Estas limitações são:

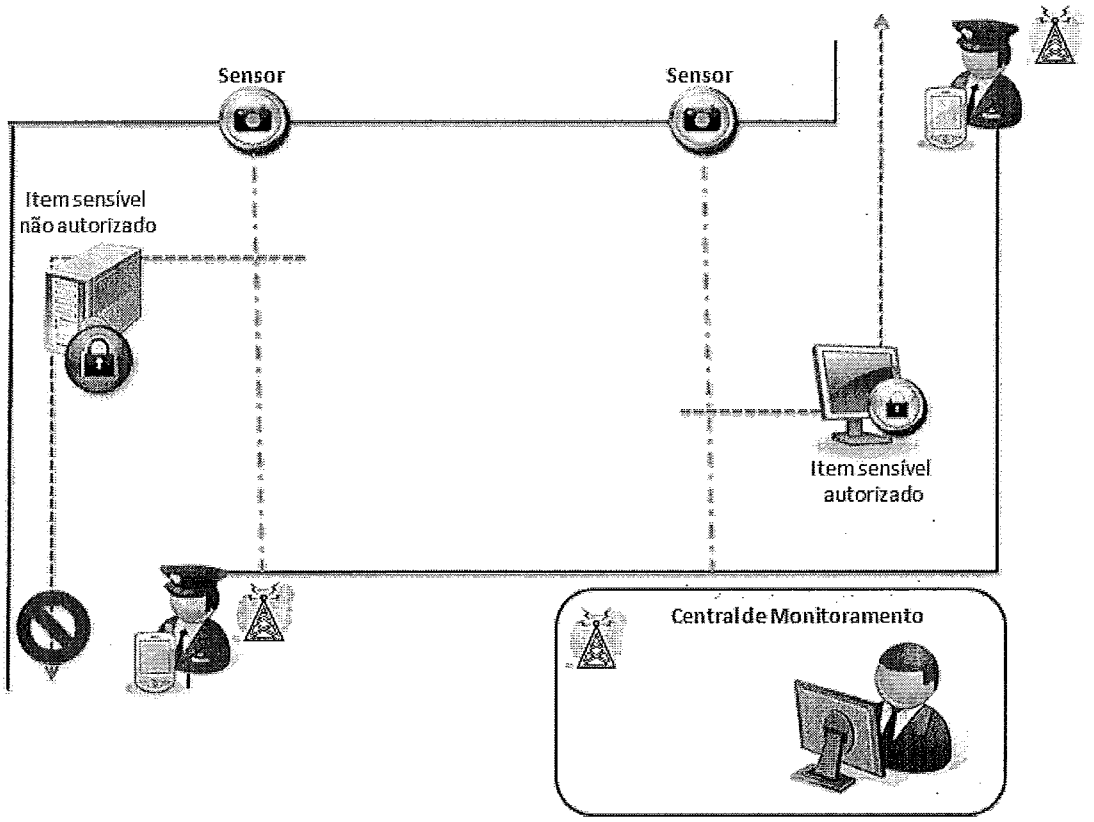
- Não existe um controle sistêmico considerando a real saída do item de patrimônio de um determinado local. Ou seja, embora possa ter havido um cadastro de transferência, nada garante que o item foi de fato retirado.
- Não existe um controle sistêmico que impeça a saída de um item de patrimônio sem prévia autorização.
- É muitas vezes difícil encontrar um item de patrimônio que tenha sido deslocado dentro da organização.

- A solução atual prevê um conjunto de impressões de formulários de retirada de item de patrimônio que poderiam ser minimizadas com um controle sistêmico mais abrangente.

Para lidar com estas limitações, as seguintes necessidades adicionais foram definidas e devem ser consideradas no projeto do SGP:

- Os bens de capital passam a ser classificados também como item rastreavel. Um item rastreavel é aquele que, além do número de patrimônio, possui uma etiqueta eletrônica de identificação;
- Para ser classificado como rastreavel, um item deve ter seus dados incluídos no cadastro de itens rastreáveis;
- As entradas e saídas dos prédios terão sensores para estas etiquetas. Ao passar com um item rastreavel, 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 que um item rastreavel está saindo/entrando na instalação;
- A saída de um item rastreavel 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 rastreavel deve sempre ser notificada para a central de monitoramento que acompanhará em tempo real as entradas e saídas de itens rastreáveis.

A Figura abaixo ilustra um cenário de integração do sistema ao ambiente da organização:



Com base na figura acima e na descrição das necessidades adicionais apresentadas anteriormente, deve-se ter a seguinte descrição em mente:

6. Imaginemos o corredor do Bloco H com suas duas saídas. Em ambos os lados, devem existir sensores instalados há uma certa distância da saída do corredor de forma que o setor de segurança possa ser notificado da saída de um item rastreavel antes de sua saída;
7. Para que um item rastreavel possa ser retirado, deve existir uma autorização prévia indicando também quem poderá retirar o item;
8. 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;
9. 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;
10. O resultado desta ação é enviado para a central de monitoramento.

Anexo F: Guia para Definição de Requisitos de Ubiqüidade com Direcionamento

1. 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?

2. 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?

3. 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?

4. 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 dispostivos?

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?

5. 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?

Anexo G – Glossário de *UbiCheck* 2.0

- **adaptação:** mudança no comportamento do sistema para continuar oferecendo conteúdo, funcionalidades e serviços.
- **alternativa de adaptação:** conjunto de adaptações que podem ser utilizadas em uma determinada situação.
- **alteração no ambiente:** mudanças que podem ocorrer em um ambiente.
- **ambiente:** espaço em que o usuário utiliza o sistema.
- **aplicações:** softwares utilizados pelos usuários do sistema.
- **atividade do usuário:** algo que o usuário pode desempenhar enquanto utiliza o sistema.
- **compatibilidade do dispositivo:** capacidade do dispositivo se comunicar com o sistema.
- **conteúdo:** informações que o usuário recebe do sistema.
- **contexto:** conjuntura em que o usuário que utiliza o sistema se encontra.
- **contexto do serviço:** informações que definem o estado de um serviço.
- **dados das informações de contexto:** dados que compõem uma informação.
- **decisão:** escolha do sistema entre mais de uma alternativa de adaptação.
- **disponibilidade do dispositivo:** o dispositivo poder ser acionado pelo sistema quando preciso.
- **dispositivo:** hardware que pode ser utilizado para acessar o sistema.
- **dispositivo heterogêneo:** dispositivo que pode utilizar o sistema como se fosse outro.
- **estado da aplicação:** condição de uma aplicação.
- **experiência do usuário:** conhecimento adquirido pelo usuário.
- **experiência privadas:** experiência do usuário que não podem ser compartilhadas.
- **experiência públicas:** experiências do usuário que podem ser compartilhadas.
- **fonte de dados:** software ou hardware responsável por capturar informações de contexto.
- **funcionalidade:** tarefa desempenhada pelo sistema.
- **identificação do dispositivo:** capacidade do sistema reconhecer um dispositivo.
- **impacto da adaptação:** problemas previstos para depois que uma adaptação ocorre.

- **informação da interação do usuário:** informação que pode ser capturada durante uma interação do usuário.
- **informação de contexto:** informação sobre o contexto.
- **informação física:** informação de contexto proveniente de sensores.
- **informação de infra-estrutura:** informação de contexto sobre a validade das informações capturadas e utilizadas pelo sistema.
- **informação de sistema:** informação de contexto proveniente de um sistema.
- **informação do ambiente:** informações sobre o espaço em que o usuário do sistema se encontra.
- **informação do usuário:** informação de contexto adquirida do usuário.
- **interação do usuário:** a forma com que o usuário interage com o sistema.
- **localização do usuário:** local geográfico que o usuário se encontra.
- **mobilidade do usuário:** a capacidade do usuário mudar de posição geográfica.
- **necessidade do usuário:** uma informação imprescindível para o usuário utilizar o sistema.
- **padrão de interação do usuário:** uma interação comum que costuma se repetir.
- **preferência do usuário:** uma informação que o sistema utiliza para saber o que agrada o usuário.
- **recurso dos dispositivos:** propriedades do dispositivo que podem ser utilizadas pelo sistema, por exemplo, processador, memória, disco etc.
- **relacionamento de experiências do usuário:** informações de experiências de usuário que podem afetar outras experiências.
- **seção do usuário:** informações do usuário que estão disponíveis quando ele utiliza o sistema.
- **serviço:** uma funcionalidade fornecida pelo sistema que pode ser utilizada no contexto de outro sistema.
- **temporalidade da informação de contexto:** quando a informação foi adquirida.
- **usuário:** aquele que utiliza o sistema, podendo ser uma pessoa ou até mesmo outro sistemas.
- **utilidade da informação de contexto:** serventia e relevância da informação de contexto.
- **versão da aplicação:** identificador que permite referenciar as funcionalidades disponíveis em uma aplicação.