UMA ABORDAGEM PARA ESPECIFICAÇÃO DE REQUISITOS FUNCIONAIS DE UBIQUIDADE EM PROJETOS DE SOFTWARE

Leonardo da Silva Mota

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 2013

UMA ABORDAGEM PARA ESPECIFICAÇÃO DE REQUISITOS FUNCIONAIS DE UBIQUIDADE EM PROJETOS DE SOFTWARE

Leonardo da Silva Mota

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.

Examinada por:

_____

Prof. Guilherme Horta Travassos, D.Sc.

_____

Prof[a].Claudia Maria Lima Werner, D.Sc.

_____

Prof[a]. Karin Koogan Breitman, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

SETEMBRO DE 2013

Mota, Leonardo da Silva

Uma Abordagem para Especificação de Requisitos Funcionais de Ubiquidade em Projetos de Software / Leonardo da Silva Mota – Rio de Janeiro: UFRJ/COPPE, 2013.

XI, 138 p.: il.; 29,7 cm.

Orientador: Guilherme Horta Travassos.

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2013.

Referências Bibliográficas: p. 101-103.

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

A minha família

Agradecimentos

A toda a minha família e amigos, por sempre torcerem e acreditarem em mim, mesmo quando tivemos boa parte do nosso tempo de convívio reduzido em favor de um sonho.

À Thamine, pelo companheirismo, dedicação e toda a ajuda dada em muitos momentos ao longo deste trabalho.

Ao amigo Marco Antônio, por me apresentar a pesquisa científica, pelo incentivo e por acreditar em minha capacidade de atuação nesta área.

Ao Jobson Massollar, grande regente deste trabalho, fundamental em cada etapa do desenvolvimento da dissertação, pela solicitude demonstrada ao atuar como meu coorientador, mesmo que extraoficialmente, por suas valiosas contribuições, pela paciência e por me conduzir de forma impecável ao longo deste trabalho. Todo agradecimento se torna insuficiente perante a dedicação empenhada.

Ao meu orientador, Guilherme Travassos, importante para o meu desenvolvimento, pela oportunidade, orientação, conselhos, motivação, paciência e por me guiar, desde o início do mestrado, pelos caminhos da vida acadêmica.

A todos os amigos e colegas do grupo de Engenharia de Software Experimental, pelas contribuições pessoais e profissionais, em especial Breno, Karen, Victor, Ciro, Paulo Sérgio, Rafael e Verônica.

Às professoras Claudia Werner e Karin Breitman por participarem de minha banca de defesa de mestrado.

A CAPES pelo apoio financeiro. À COPPE por prover a infraestrutura.

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 ESPECIFICAÇÃO DE REQUISITOS FUNCIONAIS DE UBIQUIDADE EM PROJETOS DE SOFTWARE

Leonardo da Silva Mota

Setembro/2013

Orientador: Guilherme Horta Travassos

Programa: Engenharia de Sistemas e Computação

A computação ubíqua se caracteriza como um paradigma onde o poder de processamento está presente de forma imperceptível e onipresente no ambiente do usuário. Poucas abordagens são encontradas na literatura técnica para apoiar a construção deste tipo de software e, neste sentido, sua qualidade pode ficar comprometida pela utilização de abordagens tradicionais da Engenharia de Software, que normalmente não tratam as características específicas deste domínio. Portanto, existe a necessidade de investigação sobre como apoiar o desenvolvimento de softwares ubíquos.

Esta dissertação propõe uma abordagem de apoio a especificação de requisitos funcionais de ubiquidade que possibilita tratar questões relacionadas à garantia da qualidade, baseada em um metamodelo elaborado, denominado *UbiModel*, que organiza as características e fatores de ubiquidade e suas interrelações. *UbiModel* foi integrado a uma abordagem para especificação de requisitos de aplicações web para tornar a especificação mais abrangente ao permitir a descrição detalhada do comportamento sistêmico, tendo em vista que esse tipo de aplicação possui características comuns que favorecem a integração.

Foi desenvolvida e avaliada uma infraestrutura computacional a partir do *UbiModel* para apoiar a especificação de requisitos de ubiquidade, onde os resultados permitiram observar que a abordagem é viável, a infraestrutura é útil e fácil de usar nesse contexto específico.

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 SPECIFICATION OF UBIQUITY FUNCTIONAL REQUIREMENTS IN SOFTWARE PROJECTS

Leonardo da Silva Mota

September/2013

Advisor: Guilherme Horta Travassos

Department: Computer Science and Systems Engineering

Ubiquitous computing is characterized as a paradigm where the processing power is present seamlessly and ubiquitously in the user environment. Few approaches are found in the technical literature to support the construction of this type of software and, thus, the software quality may be compromised by using traditional software engineering approaches that usually do not address specific domain characteristics. Therefore, there is a need for research on how to support the development of ubiquitous software.

This paper proposes an approach to support the specification of ubiquity functional requirements that allows to address issues related to quality assurance, based on *UbiModel*, a metamodel that organizes the features and factors and their ubiquity interrelationships. *UbiModel* has been integrated into a requirements specification approach for web applications to turn the specification more comprehensive allowing a detailed description of system behavior, given that these types of application share common characteristics that favor integration.

A computational infrastructure to support the specification of ubiquity functional requirements according to *UbiModel* has been developed and empirically evaluated. The evaluation results indicate the approach is feasible and the infrastructure is useful and easy to be used for this specific context.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

1 Introdução

Neste capítulo são apresentados os conceitos básicos de computação ubíqua, bem como as questões que motivaram a realização deste trabalho. Em seguida, é apresentada também a proposta e a organização dessa dissertação.

1.1 Introdução

Ao longo dos últimos anos, observou-se uma modificação no cenário de desenvolvimento de software, que passou a ser caracterizado pela proliferação de diferentes dispositivos com poder de processamento interligados a partir de mecanismos de comunicação sem fio. Estes avanços tecnológicos impulsionaram o crescimento da computação ubíqua que é caracterizada como a presença destes dispositivos disponíveis de forma onipresente e imperceptível no ambiente do usuário (ARK & SELKER, 1999; DRYER, EISBACH & ARK, 1999; MOBILEMAN, 2011).

Alguns autores (WEISER, 1991; KRIKKE, 2005) consideram a computação ubíqua como o terceiro paradigma da computação. O primeiro paradigma foi o dos *mainframes*, onde muitas pessoas compartilhavam um mesmo sistema. O segundo foi determinado pela introdução e ampla disseminação dos computadores pessoais, onde cada pessoa usava seu próprio computador. O terceiro paradigma caracteriza-se pela utilização individual de diversos dispositivos interconectados e com poder de processamento. Esta evolução pode ser observada na Figura 1-1.

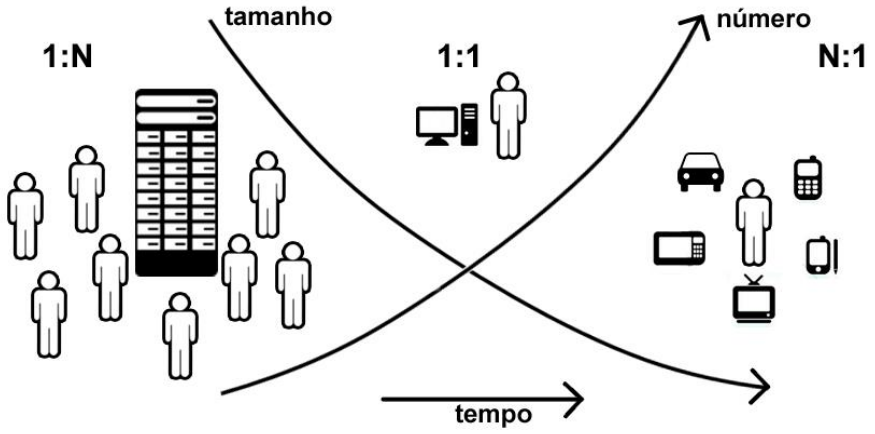


Figura 1-1 – Paradigmas da computação (WEISER, 1991)

Sendo assim, o objetivo da computação ubíqua é tornar os serviços computacionais disponíveis no ambiente que cerca o usuário, sem que a utilização dos serviços seja intrusiva para o mesmo (WEISER, 1991; ABOWD, 1999).

1.2 Computação Ubíqua e Engenharia de Requisitos

A busca pela qualidade dos produtos de software tem levado a indústria de software ao aprimoramento de seus processos e, dentre as diversas áreas passíveis de melhoria, uma das mais visadas é a Engenharia de Requisitos. Isto ocorre porque os requisitos possuem um papel central no processo de desenvolvimento de software, pois são a base para estimativas, modelagem, projeto, implementação e testes, estando presentes ao longo de todo o ciclo de desenvolvimento do sistema. Atividades de controle de qualidade devem ser realizadas para verificar, validar e garantir a qualidade dos requisitos, pois a partir deles outros artefatos ao longo do projeto serão gerados. Defeitos encontrados tardiamente oriundos dos requisitos podem aumentar o custo de correção, retrabalho e riscos de falha do software, sendo essencial tentar reduzir a inserção de defeitos nas fases iniciais do projeto (BOEHM, 2001) a fim de não permitir sua propagação para fases posteriores do desenvolvimento. Ou seja, quanto mais cedo os defeitos são corrigidos ou minimizados, menores serão os custos de sua correção.

Os softwares ubíquos possuem características específicas que os diferenciam dos softwares tradicionais. Estas características normalmente não são consideradas por abordagens de desenvolvimento de software convencionais disponíveis no corpo de conhecimento da engenharia de software (ABOWD, 1999; BANAVAR e BERNSTEIN, 2002; KINDBERG e FOX, 2002; DUCATEL *et. al.*, 2003; e NIEMELA e LATVAKOSKI, 2004). A utilização de abordagens tradicionais no desenvolvimento de software ubíquo possivelmente diminui sua eficiência e sua eficácia (DUCATEL *et. al.*, 2003) e por esta razão é importante entender como as características de ubiquidade podem influenciar o processo de desenvolvimento do software e como podem ser consideradas por abordagens de desenvolvimento. Entretanto, ainda existem poucas investigações sobre como a engenharia de software pode apoiar o desenvolvimento de software ubíquo, o que aumenta a dificuldade e os riscos associados a esta categoria de software (SPÍNOLA, 2010).

A partir desse cenário, SPÍNOLA (2010) apresenta uma abordagem para apoiar a definição e verificação de requisitos de ubiquidade com preocupações relacionadas à garantia da qualidade da especificação. Esta abordagem foi elaborada a partir de um conjunto de características e fatores da computação ubíqua (SPÍNOLA *et. al.*, 2006),

com a finalidade de se obter uma especificação de software mais objetiva ao considerar as especificidades do domínio do problema frente à ubiquidade computacional.

O conjunto de características, fatores de ubiquidade e suas respectivas definições SPÍNOLA (2010) foi obtido a partir de revisões sistemáticas da literatura (SPÍNOLA *et. al.*, 2006) e de pesquisas de opinião com especialistas no domínio da computação ubíqua. A partir desse corpo de conhecimento, PINTO (2009) propôs um guia, denominado *UbiCheck*, que tem como objetivo apoiar e guiar a captura de requisitos de ubiquidade através de um conjunto de perguntas. *UbiCheck* está estruturado como um conjunto de questões abertas que procuram direcionar o especificador a identificar e especificar requisitos relevantes no contexto de sistemas ubíquos, no entanto, sem informar como a captura das informações deve ser realizada, cabendo ao especificador interpretar o conjunto de questões de acordo com sua experiência.

Inicialmente, quando o foco é obter e aprimorar o entendimento do sistema, a utilização de *UbiCheck* para a especificação de requisitos de ubiquidade atende seus objetivos, pois direciona a atenção do especificador aos aspectos a serem considerados na descrição do que o sistema deve fazer. Quando o foco passa a ser o comportamento esperado do sistema para atender as necessidades do usuário, a utilização de *UbiCheck* não se mostra tão eficiente, pois não apoia a descrição dos requisitos sob essa ótica, onde os resultados dependem muito mais da experiência de quem está especificando os requisitos. A descrição detalhada do comportamento do sistema é de extrema importância, pois serve como base para a construção do software em fases posteriores no processo de desenvolvimento. As questões de *UbiCheck* levam a definição de requisitos sob o ponto de vista do usuário sem apresentar detalhes relacionados ao comportamento sistêmico para a realização desses requisitos.

1.3 Motivação

Apesar de *UbiCheck* auxiliar o especificador a direcionar sua atenção para as informações importantes na definição dos requisitos de ubiquidade, sua estrutura de questões abertas oferece oportunidades de melhoria no que diz respeito a:

- Organizar um apoio adequado à especificação de requisitos com algum nível de automatização;
- Minimizar a subjetividade da abordagem, que deixa a cargo do especificador a interpretação de algumas perguntas e direcionamentos.

É importante ressaltar que essa foi uma das questões centrais abordadas pelos participantes do estudo experimental realizado com *UbiCheck* (SPÍNOLA, 2010), e;

- Ampliar a abrangência da especificação dos requisitos, passando a considerar também a descrição do comportamento sistêmico que atende às demandas e necessidades do usuário.

Assim, tais oportunidades de melhoria remetem a questões que motivaram a elaboração deste trabalho:

- Como prover um direcionamento mais objetivo acerca da especificação de requisitos de ubiquidade?
- Como especificar requisitos ubíquos tanto sob a ótica do usuário quanto sob a do sistema?

1.4 Objetivo

Tendo em vista as oportunidades de melhoria relatadas nas seções anteriores, este trabalho se caracteriza por ser uma pesquisa aplicada ao produzir conhecimento visando solucionar problemas concretos e imediatos (BARROS & LEHFELD, 2000; APOLINÁRIO, 2004). O objetivo deste trabalho é estender a abordagem apresentada em SPÍNOLA (2010) através da elaboração de um metamodelo denominado *UbiModel* que procura organizar os conceitos e relações relacionados as características e fatores de ubiquidade de forma mais detalhada, com o objetivo de eliminar o caráter interpretativo característico da abordagem. A partir de *UbiModel* os requisitos definidos podem ser verificados sintaticamente evitando a introdução de alguns defeitos na especificação, principalmente aqueles relacionados a omissão. Outro aspecto explorado na pesquisa está relacionado às perspectivas nas quais os requisitos se apresentam. Os requisitos definidos em SPÍNOLA (2010) são descritos sob o ponto de vista do usuário, ou seja, representam suas demandas, necessidades e restrições. Sendo assim, observou-se a possibilidade de tornar a especificação mais abrangente ao descrever os requisitos também sob a perspectiva do sistema, ou seja, os desdobramentos das demandas, necessidades e restrições no comportamento efetivo do sistema. Para isso, o presente trabalho considera a integração entre a abordagem proposta por SPINOLA (2010) e a proposta por MASSOLLAR (2011). MASSOLLAR (2011) propõe a especificação estruturada de requisitos funcionais de aplicações Web através de um arcabouço para análise, classificação e especificação de requisitos alinhado aos conceitos explorados pelos métodos Web contemporâneos, que permite a especificação de casos de uso através de diagramas de atividades, em um nível de

detalhe que permite a definição do comportamento esperado do sistema para a realização dos requisitos definidos sob o ponto de vista do usuário. Desta forma, este trabalho pretende realizar a integração entre dois domínios: computação ubíqua e aplicações web.

No contexto do presente trabalho, foi adotada a definição de Aplicação Web definida em MASSOLLAR (2011), adaptada de KAPPEL *et. al.* (2006):

> "*Uma aplicação Web é um sistema de software baseado em tecnologias e padrões do World Wide Web Consortium (W3C) que provê recursos específicos da Web, como conteúdo e serviços, através de um* ***cliente*** *Web*".

O objetivo ao realizar esta integração é obter uma especificação que apresente as informações da forma como o sistema faz para realizar as demandas do usuário e consequentemente tornar possível a identificação dos requisitos que dão origem aos comportamentos especificados do sistema. A Figura 1-2 resume o histórico da pesquisa realizada neste trabalho. Inicialmente, SPÍNOLA *et. al.* (2006) realizou pesquisas para elaborar o corpo de conhecimento em computação ubíqua, ou seja, as características e fatores de ubiquidade. A partir dessas definições, PINTO (2009) construiu o *checklist UbiCheck*, que compõe uma abordagem mais abrangente para definição de requisitos proposta por SPÍNOLA (2010). A partir da análise de *UbiCheck*, observou-se oportunidades de melhoria que levaram a elaboração do *UbiModel*, que pode ser considerado como uma extensão de *UbiCheck*. Durante a definição de *UbiModel*, foi possível explorar outra oportunidade de melhoria no que diz respeito a abrangência da especificação dos requisitos. Nesse sentido, foi realizada a junção do *UbiModel* à abordagem apresentada por MASSOLLAR (2011), para a especificação de requisitos de aplicações web através de diagramas de atividades e casos de uso. Por fim, foi realizado um estudo para avaliar a utilidade e a facilidade de uso da infraestrutura computacional proposta na captura e especificação de requisitos de ubiquidade.

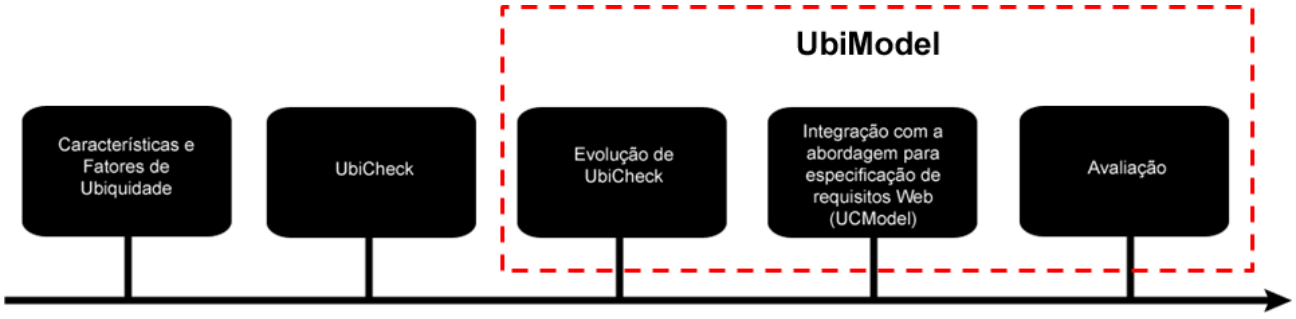


Figura 1-2 – Linha do tempo do histórico da pesquisa

1.5 Metodologia de Trabalho

A metodologia utilizada para a realização deste trabalho seguiu os seguintes passos (Figura 1-2):

- Realizar revisão inicial da literatura sobre a computação ubíqua, enfatizando as atividades de especificação de requisitos, seus conceitos, suas características e benefícios para uma organização de software. Com isso, obteve-se uma melhor compreensão dos conceitos relacionados ao tema;
- Revisar os artefatos produzidos pelas atividades descritas em PINTO (2009) e SPÍNOLA (2010) por constituírem o oráculo no desenvolvimento da abordagem proposta. Foram encontrados alguns defeitos que poderiam impactar a abordagem proposta;
- Elaborar o metamodelo *UbiModel* que permite a organização da especificação dos requisitos de acordo com um conjunto de critérios e restrições.
- Atualizar as descrições das características de ubiquidade segundo a interpretação dada na elaboração do metamodelo *UbiModel*;
- Elaborar glossário de termos sobre computação ubíqua para apoiar a especificação de requisitos;
- Elaborar conjunto de orientações voltadas para a redação das descrições dos requisitos de ubiquidade;
- Elaborar os modelos de apoio à especificação de requisitos de ubiquidade;
- Definir a infraestrutura computacional, resultando em um conjunto de funcionalidades a ser implementado;
- Construir a infraestrutura computacional;
- Realizar estudo de viabilidade com o objetivo de verificar a viabilidade do uso da infraestrutura quanto à percepção sobre utilidade e sobre facilidade de uso.

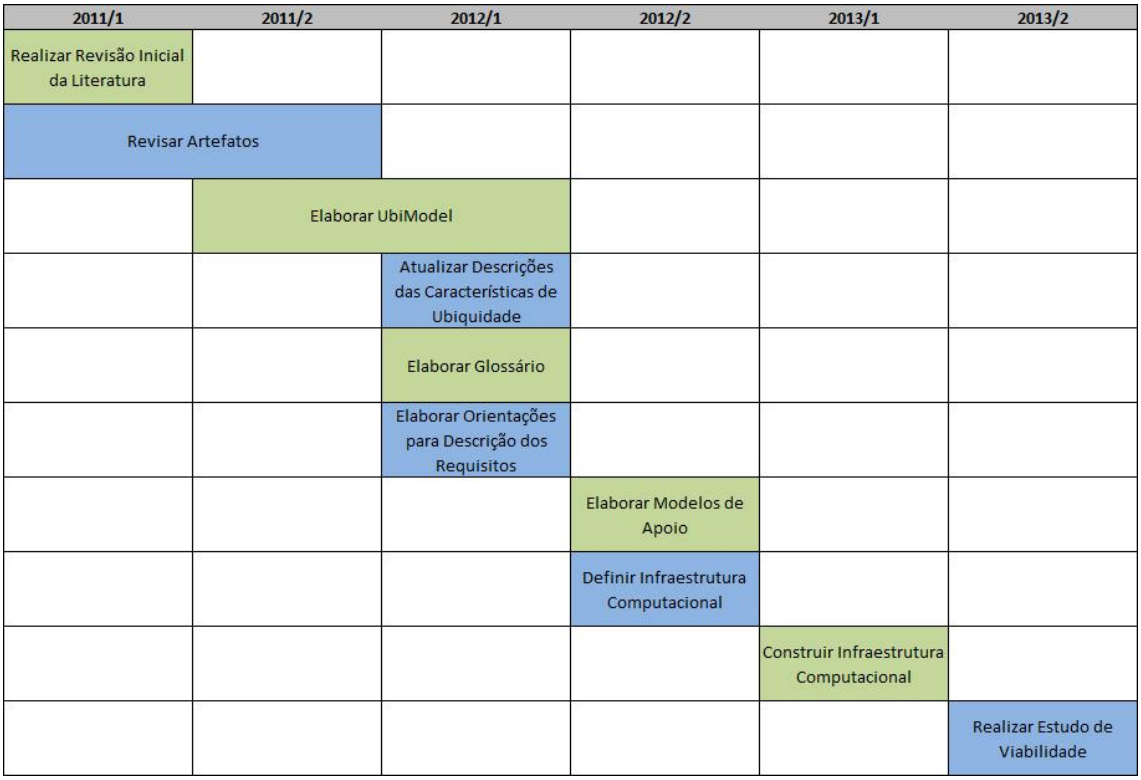


Figura 1-3 – Etapas da metodologia de trabalho aplicada no desenvolvimento da abordagem

1.6 Organização da Dissertação

Este trabalho está organizado em sete capítulos. Este capítulo apresentou o contexto e a motivação para realização dessa pesquisa, bem como as questões de pesquisa e os objetivos planejados.

O segundo capítulo apresenta os trabalhos que serviram de base para o desenvolvimento da abordagem proposta neste trabalho. Um dos trabalhos apresenta uma abordagem para apoiar a definição e verificação de requisitos de ubiquidade em projetos de software. O outro trabalho apresenta uma abordagem para apoiar a definição de requisitos de aplicações Web.

O terceiro capítulo apresenta o metamodelo proposto para apoiar a especificação de requisitos de ubiquidade em projetos de software bem como permitir a avaliação da qualidade dessa especificação.

O quarto capítulo apresenta como o metamodelo proposto foi elaborado para considerar a descrição dos requisitos de ubiquidade sob o ponto de vista do comportamento do sistema, de forma que a especificação possa ser utilizada em etapas posteriores à etapa de requisitos no processo de desenvolvimento do sistema.

O quinto capítulo apresenta a infraestrutura computacional para apoiar a abordagem de especificação de requisitos de ubiquidade proposta nesta pesquisa e

detalha as três ferramentas desenvolvidas no escopo deste trabalho que compõem essa infraestrutura.

O sexto capítulo apresenta uma avaliação da infraestrutura computacional para identificar indícios do seu apoio na definição de requisitos de ubiquidade em projetos de software.

O sétimo e último capítulo apresenta as contribuições e limitações deste trabalho, bem como os trabalhos futuros que podem ser derivados do mesmo.

2 Requisitos de Aplicações Ubíquas e Requisitos de Aplicações Web

Neste capítulo são apresentados os trabalhos que serviram de base para o desenvolvimento da abordagem proposta. Um dos trabalhos apresenta uma abordagem para apoiar a definição e verificação de requisitos de ubiquidade em projetos de software. O outro trabalho apresenta uma abordagem para apoiar a definição de requisitos de aplicações Web.

2.1 Introdução

De acordo com o cenário apresentado no capítulo 1 e relacionado à falta de abordagens apropriadas que apoiem o desenvolvimento de projetos de software ubíquos de forma efetiva, SPÍNOLA (2010) organizou um corpo de conhecimento em computação ubíqua que pode ser usado como ponto de partida para o uso dos métodos, técnicas e práticas da engenharia de software à luz desse novo paradigma, com o objetivo de mitigar os riscos relacionados ao desenvolvimento de projetos de software nesse domínio de aplicação.

De acordo com BOEHM (1981), WHEELER *et. al.* (1996) e BOEHM e BASILI (2001), uma das principais formas de minimizar impactos negativos em fases avançadas do desenvolvimento de software é através da redução da inserção de defeitos nas fases iniciais do projeto e criação de mecanismos que possam identificar os defeitos nas fases em que são inseridos, a fim de não encontrá-los em fases posteriores do desenvolvimento, o que aumentaria o custo de correção, além de retrabalho. Nesse sentido, SPÍNOLA (2010), baseado no corpo de conhecimento organizado, desenvolveu uma abordagem para apoiar a definição e verificação de requisitos funcionais de ubiquidade em projetos de software.

Em uma outra abordagem apresentada por MASSOLLAR (2011), também desenvolvida no contexto do grupo de Engenharia de Software Experimental da COPPE/UFRJ, tem como objetivo principal estruturar um arcabouço para análise, classificação e especificação de requisitos funcionais alinhados aos conceitos explorados por métodos Web contemporâneos, oferecendo, assim, uma forma consistente para tratamento dos requisitos desde a fase inicial do projeto (MASSOLLAR, 2011).

Segundo MASSOLLAR (2011), ao estruturar os requisitos funcionais, a qualidade do produto final aumenta, pois o mesmo passa pela estruturação adequada e pelo controle de qualidade dos requisitos funcionais, que representam o oráculo a partir do qual os modelos Web e o plano de testes funcionais são derivados.

Um software ubíquo pode ser representado por uma aplicação web. De acordo com KAPPEL *et. al.* (2001), projetos Web possuem características da computação ubíqua: disponibilidade a qualquer momento, em qualquer lugar e a partir de qualquer dispositivo. Assim, a pesquisa apresentada nessa dissertação se caracteriza por ser a integração da pesquisa iniciada em SPÍNOLA (2010) e a metodologia proposta por MASSOLAR (2011). Mais especificamente, foram observadas oportunidades de melhoria, exploradas no presente trabalho, relacionadas ao alto nível de abstração dos requisitos definidos pela abordagem apresentada por SPÍNOLA (2010). Outra oportunidade observada está relacionada ao fato dos requisitos definidos em SPÍNOLA (2010) serem descritos apenas sob o ponto de vista do usuário, ou seja, representarem suas demandas, necessidades e restrições. Observou-se que os requisitos também poderiam ser descritos sob a perspectiva do sistema, ou seja, apresentando os detalhes que descrevem o comportamento efetivo do sistema. Além disso, é possível realizar a junção da abordagem apresentada por SPÍNOLA (2010) àquela apresentada por MASSOLLAR (2011) para a definição de requisitos de aplicações Web, onde os requisitos descrevem exatamente o comportamento esperado do sistema. Com isso, espera-se obter requisitos mais completos, descritos tanto sob o ponto de vista do usuário quanto sob o ponto de vista do sistema.

De forma a permitir maior clareza no entendimento da proposta apresentada nesta dissertação, as seções 2.2 e 2.3 apresentam as descrições das abordagens envolvidas no presente trabalho para:

- especificação de requisitos funcionais de ubiquidade;
- especificação de requisitos funcionais de aplicações Web.

2.2 Apoio à Especificação e Verificação de Requisitos Funcionais de Ubiquidade em Projetos de Software

A abordagem apresentada por SPÍNOLA (2010) apoia a definição dos requisitos de ubiquidade considerando questões relacionadas à garantia da qualidade dos requisitos. A abordagem foi elaborada a partir de um conjunto de características e fatores da computação ubíqua identificadas por SPÍNOLA *et. al.* (2006) com a finalidade de se obter uma especificação mais alinhada às especificidades desse domínio.

Como as definições encontradas na literatura para computação ubíqua eram muito genéricas e não explicitavam quais características um software deveria contemplar para ser considerado ubíquo, SPÍNOLA *et. al.* (2006) realizaram uma revisão sistemática da literatura com o objetivo de elaborar uma definição de computação ubíqua e identificar suas características a partir do conhecimento disponível na literatura técnica. A partir do resultado desse estudo, foi adotada uma definição de computação ubíqua mais abrangente, explicitando as características de ubiquidade encontradas (SPÍNOLA *et. al.*, 2006):

> "*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. Nesse contexto, os sistemas que compõem o ambiente podem contemplar, total ou parcialmente, as seguintes características: onipresença de 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*"

As características representam propriedades da computação ubíqua que podem ser observadas em softwares ubíquos. Ao todo, foram identificadas dez características:

Onipresença de Serviço: permitir o deslocamento do usuário de um lugar para outro, dando a ele a impressão que os serviços computacionais o acompanham em seu deslocamento;

- **Cenário:** o usuário utiliza um sistema de navegação para encontrar uma loja em um centro comercial e, após estacionar e sair do veículo, o sistema continua fornecendo as coordenadas até a loja desejada.

Invisibilidade: presença em objetos de uso cotidiano do usuário, descaracterizando o uso de um computador tradicional e reforçando a percepção de dispositivos comuns que fornecem serviços. Com isto, procura-se viabilizar formas naturais de interação com o sistema, como gestos e vozes.

- **Cenário:** o usuário fala o nome de uma emissora de rádio e o aparelho de som automaticamente a sintoniza.

Sensibilidade ao Contexto: capturar informações do ambiente de utilização, podendo alterar seu comportamento em função dessas informações;

- **Cenário:** um sistema para controle de temperatura de uma metalúrgica deve estar constantemente monitorando a temperatura para evitar a ocorrência de acidentes.

Comportamento Adaptável: adaptar uma funcionalidade para se adequar ao ambiente em que está inserido dentro de suas limitações;

- **Cenário:** um sistema, ao perceber a queda de energia elétrica e acionamento dos geradores, se adapta de forma que o gasto da energia seja otimizado para minimizar desperdícios.

Captura de Experiências: monitorar e registrar as interações do usuário com o sistema para posterior utilização desse conhecimento em benefício do usuário;

- **Cenário:** o usuário todos os dias ao chegar a sua casa após o trabalho liga a TV no canal de notícias, o sistema, ao perceber essa rotina, passa a ligar e sintonizar a TV no canal de notícias ao perceber a chegada do usuário na casa, sem que ele precise desempenhar diretamente estas atividades.

Descoberta de Serviços: descobrir, selecionar e utilizar serviços disponíveis no ambiente;

- **Cenário:** o usuário entra em um supermercado e seu celular mostra o catálogo de produtos em promoção do dia, utilizando um serviço disponibilizado pelo supermercado.

Composição de Funcionalidades: fornecer ao usuário funcionalidades mais elaboradas a partir de serviços básicos;

- **Cenário:** o usuário deseja abrir um arquivo desconhecido. O sistema pode identificar e realizar o download de um visualizador compatível com o arquivo, identificar um serviço de antivírus e investigar o arquivo para garantir a ausência de vírus no arquivo.

Interoperabilidade Espontânea: permitir a comunicação entre dispositivos sem a intervenção do usuário;

- **Cenário:** o funcionário da empresa de água e esgoto caminha pela rua enquanto seu PDA se comunica com o hidrômetro digital das casas próximas para leitura do consumo de água.

Heterogeneidade de Dispositivos: permitir utilização em dispositivos com características distintas;

- **Cenário:** para pagar uma conta com cartão de crédito em uma loja, o usuário pode utilizar um terminal de apoio, um celular, um *tablet* ou um *notebook*.

Tolerância a Falhas: adaptar-se diante de falhas no ambiente para continuar a prover funcionalidades indispensáveis.

- **Cenário:** o sistema utiliza sensores para manter a temperatura de um ambiente refrigerado. Ao perceber que os sensores estão danificados, ele passa a utilizar um conjunto de sensores sobressalentes para manter a temperatura do ambiente adequada.

Embora a identificação dessas características contribuísse para melhor definir a computação ubíqua, elas ainda apresentavam um grau de abstração muito elevado, portanto, SPÍNOLA *et. al.* (2007a) realizaram uma segunda revisão sistemática da literatura com o objetivo de esclarecer como estas características eram encontradas em um software ubíquo. Esta revisão teve como objetivo identificar comportamentos que poderiam ocorrer em softwares ubíquos devido à presença de uma determinada característica de ubiquidade. Os comportamentos foram denominados fatores de ubiquidade. A seguir é apresentado um exemplo de fator de ubiquidade para a característica Sensibilidade ao Contexto:

- Permitir identificar a identidade, localização ou atividade de um usuário quando este estiver no contexto de funcionamento do sistema.

Como o processo de aquisição de informação para reduzir os riscos envolvidos com o desenvolvimento de projetos de software ubíquo deve começar com a identificação do impacto das características de ubiquidade sobre projetos de software, SPÍNOLA *et. al.* (2007) elaboraram uma abordagem para apoiar a caracterização de projetos de software ubíquo, baseada no conjunto de características e fatores de

ubiquidade encontrados nos estudos realizados. A abordagem foi aplicada em um conjunto de projetos identificados na literatura técnica e o resultado da aplicação indicou a necessidade de realização de ajustes no corpo de conhecimento organizado.

Nesse sentido, foi realizada uma avaliação do corpo de conhecimento para verificar se o conjunto de características de ubiquidade e seus fatores faziam sentido e se existiam ajustes que deveriam ser realizados tanto nas características quanto nos fatores. Para isso foram realizados *surveys* com especialistas em computação ubíqua. Como resultado, algumas características e fatores foram desconsiderados, segundo critérios adotados nos estudos, e algumas características foram adicionadas por sugestão dos especialistas. Além disso, as características foram reorganizadas considerando-se as perspectivas funcional e restritiva.

É importante destacar que como a abordagem elaborada por SPÍNOLA (2010) não apoia a definição e verificação de requisitos não funcionais, do conjunto original de características de ubiquidade identificadas, apenas as seguintes características funcionais permaneceram no escopo da abordagem: **Onipresença de Serviços**, **Sensibilidade ao Contexto**, **Comportamento Adaptável**, **Captura de Experiência**, **Heterogeneidade de Dispositivos** e **Interoperabilidade Espontânea**.

Foi identificado que a partir do conhecimento organizado seria possível apoiar atividades associadas a definição e verificação de requisitos para projetos de software ubíquos. Nesse sentido, SPÍNOLA (2010) definiu um arcabouço composto de um conjunto de facilidades associadas às atividades de definição e verificação de requisitos de ubiquidade em projetos de software. A seguir são descritas cada uma das facilidades previstas na abordagem.

2.2.1 Caracterização do Projeto

De acordo com a definição de computação ubíqua adotada por SPÍNOLA *et. al.* (2006), um sistema para ser considerado ubíquo não precisa contemplar necessariamente todas as características de ubiquidade. Sendo assim, lançar mão de todo o corpo de conhecimento organizado para apoiar o desenvolvimento de sistemas pode reduzir a efetividade de utilização deste conhecimento. Nesse sentido, foi definido um formulário de caracterização baseado nas características de ubiquidade e seus fatores, com o objetivo de configurar o corpo de conhecimento às necessidades do projeto.

2.2.2 Especificação de requisitos de ubiquidade

Para apoiar a especificação dos requisitos de ubiquidade foi definido *UbiCheck* (PINTO *et. al.*, 2009), um guia para definição de requisitos de ubiquidade baseado em *checklist* composto por perguntas sobre os assuntos relevantes de serem capturados nos requisitos de ubiquidade. Essa abordagem foi desenvolvida no contexto de uma dissertação de mestrado por um aluno do grupo de Engenharia de Software Experimental da COPPE/UFRJ.

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. Nesse sentido, PINTO *et. al.* (2008) construíram modelos conceituais para explicitar os conceitos e relações importantes identificados a partir dos fatores das características de ubiquidade, chegando-se a um corpo de conhecimento composto por um conjunto de modelos de características de ubiquidade com o objetivo de tornar mais fácil para um pesquisador entender as informações relacionadas às características de ubiquidade. A Figura 2-1 apresenta uma visão parcial do modelo de ubiquidade da característica Sensibilidade ao Contexto.

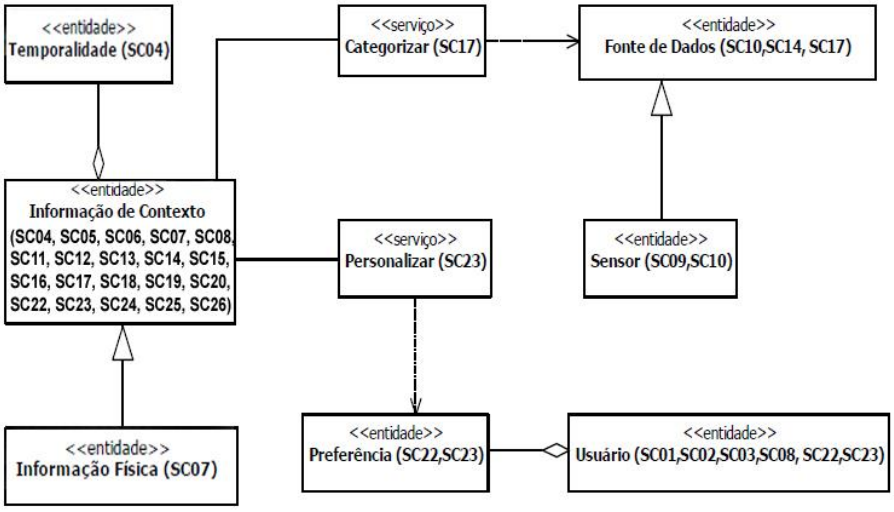


Figura 2-1 - Visão parcial do modelo conceitual da característica de ubiquidade Sensibilidade ao Contexto (PINTO, 2009)

Os modelos conceituais serviram de base à construção de *UbiCheck* onde suas informações foram transformadas em perguntas para ajudar o especificador na captura dos requisitos associados às características de ubiquidade. A seguir são apresentados três exemplos de perguntas obtidas através das transformações realizadas a partir do modelo da característica Sensibilidade ao Contexto:

- Quais as informações de Contexto relevantes para o sistema?

- Quais informações de contexto são do tipo informação física?
- Como a preferência do usuário é considerada?

2.2.3 Verificação dos requisitos de ubiquidade

Posteriormente, com o objetivo de avaliar a qualidade dos requisitos definidos através de *UbiCheck*, SPÍNOLA (2010) elaborou um conjunto de perguntas sobre informações presentes nos fatores de ubiquidade que seriam interessantes de serem verificados durante a revisão de requisitos de ubiquidade do projeto, denominado UbiVeri (SPÍNOLA, 2010). A elaboração das perguntas considerou a taxonomia de defeitos descrita em SHULL *et. al.* (2000) contendo os seguintes tipos de defeito: ambiguidade, inconsistência, fato incorreto, omissão e informação estranha. A seguir é apresentada uma pergunta para cada tipo de defeito considerado.

- **Omissão**: **Está definido** como o software identifica o usuário que o está manipulando?
- **Ambiguidade**: A definição sobre a identificação dos usuários **está clara**?
- **Inconsistência**: A definição sobre a identificação dos usuários **é feita de forma diferente em locais diferentes no documento de requisitos**?
- **Fato Incorreto**: A definição sobre a identificação dos usuários **está correta**?
- **Informação Estranha**: Existe algo descrito no documento de requisitos que está claramente fora do escopo do projeto?

A Figura 2-2 ilustra a abordagem completa desenvolvida em SPÍNOLA (2010).

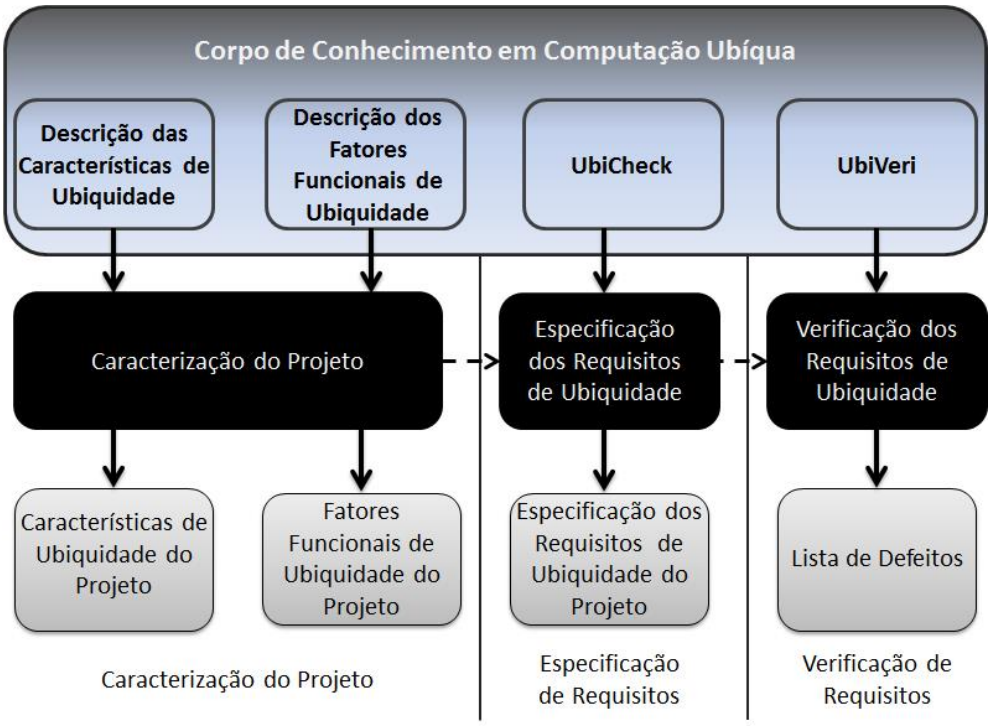


Figura 2-2 - Visão geral da abordagem de apoio à definição e verificação de requisitos de ubiquidade em projetos de software (SPÍNOLA, 2010)

A abordagem definida por SPÍNOLA (2010) é dividida em três etapas: a caracterização do projeto ubíquo, a especificação de requisitos de ubiquidade e a verificação de requisitos de ubiquidade.

Para apoiar a caracterização do projeto, foi desenvolvido um serviço web que permite a seleção dos fatores relacionados ao projeto, com a finalidade de especializar o guia *UbiCheck* para conter apenas as questões relevantes. Conforme o especificador seleciona os fatores que considera relevantes para o seu projeto, as perguntas são apresentadas. Desta forma, o especificador, ao responder as perguntas, é direcionado às questões relevantes na descrição dos requisitos de ubiquidade.

Finalmente, um guia contendo questões relacionadas à verificação de requisitos é gerado de acordo com a caracterização do projeto e, ao respondê-las, o especificador está avaliando a qualidade da especificação elaborada.

2.3 Uma Abordagem para Especificação de Requisitos Dirigida por Modelos Integrada ao Controle da Qualidade de Aplicações Web

Ainda sobre a questão de descrição de requisitos, outro trabalho realizado dentro do mesmo grupo de pesquisa, apresentou uma proposta para especificação de

requisitos dirigida por modelos integrada ao controle de qualidade de aplicações WEB (MASSOLAR, 2011), organizada de forma que seus elementos são descritos baseados nas perspectivas de projeto Web (conceituação, navegação e apresentação) e das visões de modelagem (estrutural e comportamental) permitindo, assim, que essas visões e perspectivas sejam exploradas de forma consistente desde a fase de especificação do projeto, já que estudos preliminares não identificaram na literatura abordagens com essas características. Para a definição dos requisitos, foi utilizada a representação em Casos de Uso (JACOBSON, 1992) para descrever a interação do sistema com o meio externo. A abordagem utiliza Diagrama de Atividades (OMG, 2010) para representação dos casos de uso, e adota de regras para a construção do Diagrama de Atividades baseadas em regras para a construção de casos de uso.

A Figura 2-3 apresenta um exemplo de caso de uso descrito através do diagrama de atividades, utilizando a abordagem proposta por MASSOLLAR (2011). A Tabela 2-1 apresenta os estereótipos específicos utilizados pela abordagem.

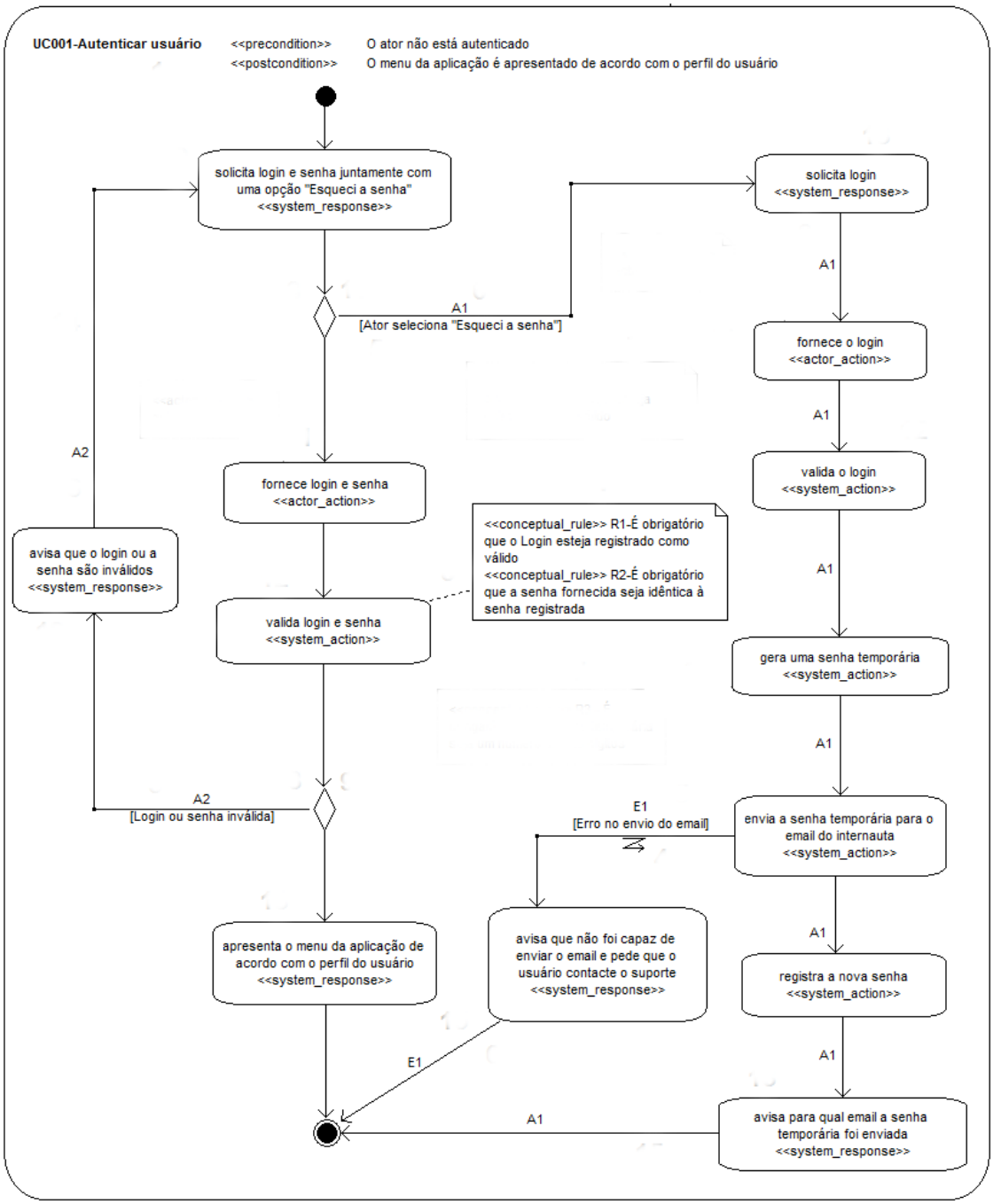


Figura 2-3 - Exemplo de caso de uso descrito através da abordagem proposta por MASSOLLAR (2011)

Tabela 2-1 – Estereótipos para descrição de casos de uso

Estereótipo	Objetivo
<<include>>	Representa o relacionamento de inclusão de caso de uso
<<extend>>	Representa o relacionamento de extensão de caso de uso
<<system action>>	Representa uma ação interna do sistema
<<actor action>>	Representa uma ação do ator
<<system response>>	Representa uma ação de resposta do sistema
<<business rule>>	Indica que uma regra de negócio é referenciada

A construção de diagramas de atividades é realizada a partir de um conjunto de elementos que estende o metamodelo da UML (OMG, 2010). Assim, MASSOLLAR (2011) elaborou um metamodelo que propõe um conjunto de elementos e restrições que visam mapear todos os conceitos do caso de uso explorados pela abordagem de especificação proposta em seu trabalho, de forma que o caso de uso possa ser completamente descrito usando o diagrama de atividades. A Figura 2-3 mostra como o metamodelo *UCModel* se relaciona com o metamodelo da UML.

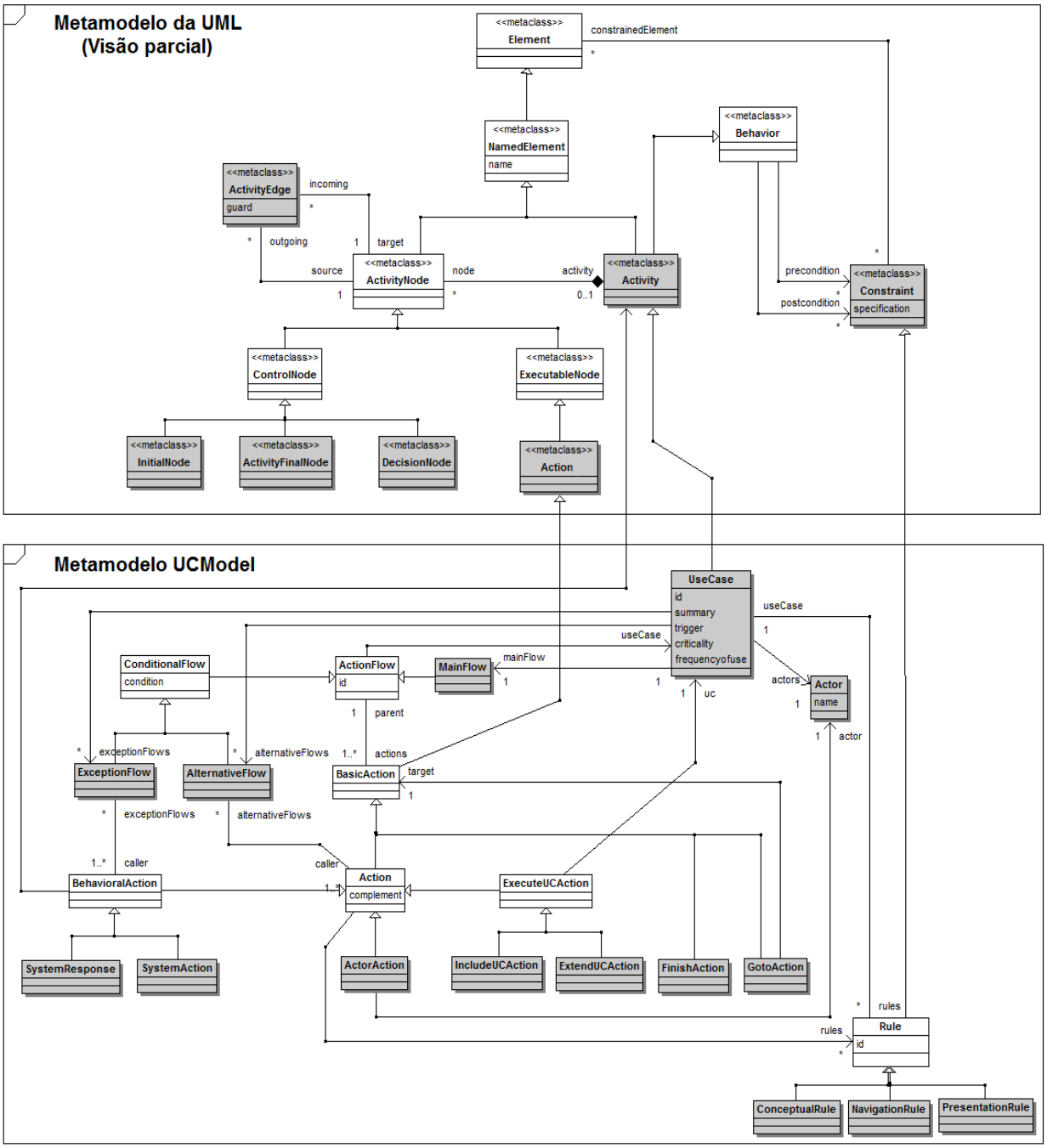


Figura 2-4 - Metamodelo *UCModel* e seu relacionamento com o metamodelo da UML (MASSOLLAR, 2011)

Para apoiar a construção da especificação dos casos de uso, MASSOLLAR (2011) desenvolveu uma ferramenta denominada *UseCaseAgent*, com o objetivo de garantir que as especificações estejam de acordo com as regras e restrições definidas em *UCModel*. Desta forma, a ferramenta recebe o resultado da análise das informações coletadas durante a elicitação dos requisitos como insumo e gera, a partir de *UCModel*, um conjunto de diagramas de atividades que representam o modelo de casos de uso.

A ferramenta permite ainda:

- verificação automática das restrições previstas no *UCModel*;
- geração automática do diagrama de atividades correspondente ao caso de uso especificado;
- edição da especificação do caso de uso por meio de uma abordagem gráfica através de um editor de diagrama de atividades de uma ferramenta case UML ou ainda por meio de uma abordagem textual, através da ferramenta UseCaseAgent, com a geração automática do diagrama de atividades e;
- geração da especificação do caso de uso em formato texto (padrão RTF - *Rich Text Format*).

2.4 Conclusão

Este capítulo teve como objetivo descrever os trabalhos que formam a base da abordagem proposta nesta dissertação. Foram utilizados os resultados de duas pesquisas realizadas dentro do contexto do grupo de Engenharia de Software Experimental da COPPE/UFRJ que tratam questões relacionadas a especificação e garantia da qualidade de requisitos de software em domínios de aplicação específicos. O trabalho realizado por SPÍNOLA (2010) trata requisitos associados ao domínio da computação ubíqua, onde é proposta uma abordagem para apoiar a definição de requisitos funcionais de ubiquidade. MASSOLLAR (2011) apresentou um arcabouço para a definição de requisitos funcionais em projetos de software web.

O objetivo geral desta dissertação é apresentar uma abordagem para especificação de requisitos de software que integre as questões associadas a definição de requisitos de ubiquidade com as questões associadas a requisitos de aplicações web, por serem requisitos relacionados a domínios com características comuns que favorecem essa integração, como disponibilidade a qualquer momento, em qualquer lugar e a partir de qualquer dispositivo.

Nesse sentido, o presente trabalho se caracteriza por ser a continuidade dos trabalhos apresentados neste capítulo, ao explorar oportunidades de melhoria

observadas, com objetivo de aprimorá-los e no final apresentar um resultado que é maior do que simplesmente o resultado isolado de cada um dos trabalhos.

SPÍNOLA (2010) elabora um guia de apoio à definição de requisitos de ubiquidade composto de um conjunto de perguntas que procuram direcionar o especificador acerca das informações importantes que devem ser capturadas na definição dos requisitos. Um aspecto que pode ser observado ao aplicar essa abordagem é o caráter interpretativo do guia. As perguntas do guia se apresentam em um nível de abstração tal que deixam a cargo de quem está efetivamente aplicando a abordagem o entendimento exato do que se espera na definição dos requisitos. Nesse contexto, para resolver possíveis ambiguidades o especificador lança mão de um glossário previsto pela abordagem para entender o significado dos termos, porém o glossário não apresenta todos os termos e nessa situação o especificador é obrigado a interpretar alguns termos por conta própria. Outro aspecto observado é que os requisitos definidos pela abordagem representam somente o ponto de vista do usuário, ou seja, representam suas demandas, necessidades e restrições.

A abordagem definida por MASSOLLAR (2010) trata a questão da descrição do comportamento do sistema sem considerar as necessidades do usuário, ou seja, ela não fornece o rastro que permite a identificação das necessidades que dão origem aos requisitos.

Em suma, a abordagem proposta por SPÍNOLA (2010) procura entender as necessidades do usuário, só que em um nível alto de abstração. Por outro lado, a abordagem proposta por MASSOLLAR (2011) tem o objetivo de descrever o comportamento do sistema sem a preocupação com as demandas que lhe dão origem. Um dos objetivos deste trabalho é descrever com maior nível de detalhes os conceitos relacionados a abordagem proposta por SPÍNOLA (2010) e definir estruturas necessárias para guiar a geração da especificação de requisitos eliminando a necessidade de interpretação do guia por parte do especificador. Dessa forma, é possível pensar em algum tipo de apoio automatizado na definição dos requisitos de ubiquidade. Outro objetivo é permitir a descrição de como essas demandas do usuário serão realizadas através do detalhamento do comportamento do sistema. Um metamodelo para especificação e verificação de requisitos de ubiquidade é proposto a partir dessas observações. Este assunto será explorado em detalhes no próximo capítulo.

3 *UbiModel*: Um Modelo para Apoiar a Especificação de Requisitos Funcionais de Ubiquidade

Neste capítulo é apresentado UbiModel, um metamodelo para apoiar a especificação de requisitos de ubiquidade em projetos de software bem como permitir a avaliação da qualidade dessa especificação.

3.1 Introdução

Apesar dos benefícios alcançados com a adoção de *UbiCheck*, sua estrutura não permite a exploração de cenários mais proeminentes, como a organização de um apoio à especificação com algum nível de automação, pois as perguntas que o compõem se apresentam em um nível de abstração elevado, fazendo com que o entendimento do que se espera na definição de um requisito muitas vezes dependa da interpretação do especificador. Nesse capítulo, é apresentado o processo de elaboração de *UbiModel*, um metamodelo proposto para apoiar a especificação e verificação de requisitos de ubiquidade, a partir da identificação dos conceitos e relações importantes, oriundos do corpo de conhecimento desenvolvido em (SPINOLA, 2010). Dessa forma, acredita-se que a especificação de requisitos de ubiquidade se torne mais completa e objetiva e que o número de defeitos na especificação seja reduzido, pois *UbiModel* apresenta estruturas mais detalhadas em um nível de abstração menor comparado a abordagem apresentada por SPINOLA (2010). *UbiModel* foi elaborado de forma a permitir sua integração ao *UCModel* (MASSOLLAR, 2010) com a finalidade de apoiar a descrição detalhada do comportamento do sistema associado aos requisitos definidos e que se apresentam a nível de demandas e necessidades do usuário. A utilização do *UCModel* se justifica por uma série de motivos: (1) casos de uso representam uma das abordagens mais utilizadas pela indústria de software; (2) ampla disseminação de conhecimento acerca dessa técnica; (3) seus elementos serem descritos à luz das perspectivas de projeto Web, ou seja, projeto de aplicações com características da computação ubíqua: disponibilidade a qualquer momento, em qualquer lugar e a partir de qualquer dispositivo (KAPPEL *et. al.*, 2001); e (4) principalmente pelas próprias definições

usadas em SPINOLA (2010) que indicam elementos de casos de uso na especificação dos requisitos.

A Seção 3.2 apresenta as atividades que foram realizadas para garantir a qualidade dos artefatos utilizados como base para elaboração do *UbiModel*.

3.2 Qualidade dos artefatos utilizados na elaboração do UbiModel

Como o ponto de partida para o desenvolvimento do *UbiModel* foram os artefatos produzidos por atividades descritas em PINTO (2009) e SPÍNOLA (2010), torna-se importante a revisão desses artefatos para garantir a consistência entre os mesmos, já que constituem a principal fonte de informação para a elaboração do *UbiModel*.

Os guias de apoio à definição de requisitos de ubiquidade foram elaborados a partir de um conjunto de modelos que procuravam representar as relações entre os fatores de ubiquidade sob um ponto de vista tal que permitisse a construção das perguntas para apoiar as atividades de definição dos requisitos de ubiquidade. Foram executadas algumas transformações nos modelos seguindo um determinado algoritmo para que fossem obtidas estruturas de dados intermediárias. Essas estruturas foram utilizadas para que fossem geradas as perguntas que compõem o guia de apoio à definição de requisitos de ubiquidade.

Como as transformações para todos os modelos não estavam explícitas nos trabalhos publicados, ou seja, tinha-se apenas os artefatos de entrada e os artefatos de saída, representados respectivamente pelos modelos conceituais e pelos guias, sentiu-se a necessidade de reexecutar o processo de transformação dos modelos para que fosse possível verificar a consistência dos guias com os artefatos intermediários produzidos ao longo do processo, já que não estava muito clara a relação entre o conjunto de perguntas dos guias e o conjunto de modelos.

Ao longo da reexecução do processo de transformação dos modelos, alguns defeitos que poderiam comprometer a qualidade do *UbiModel* foram encontrados. A descrição desses defeitos bem como seus impactos serão apresentados na subseção seguinte. É importante destacar que essa avaliação funcionou como uma releitura dos trabalhos relacionados para preencher lacunas que eram importantes e permitir a definição dos elementos de uma forma mais concreta em um nível de detalhe maior, compatível com os propósitos do *UbiModel*.

3.2.1 Artefatos revisados

Como a base para a construção dos guias são os modelos de ubiquidade que procuram estabelecer os relacionamentos entre os conceitos identificados nas revisões sistemáticas realizadas em SPÍNOLA *et. al.* (2006), os mesmos foram revisados com a finalidade de encontrar defeitos que pudessem ser propagados e comprometer a qualidade dos guias. As características de ubiquidade foram organizadas em modelos com base nas descrições dos fatores de ubiquidade pela identificação de seus principais conceitos e relações, de acordo com um metamodelo que foi elaborado para apoiar essa atividade (PINTO, 2009). Uma inspeção *ad-hoc* foi realizada nos modelos para todas as características de ubiquidade.

Após a inspeção dos modelos de características de ubiquidade, foram realizadas revisões dos artefatos gerados pela atividade de elaboração do guia, que correspondem às estruturas que são transformadas até a obtenção das perguntas do guia sobre informações que precisam ser capturadas durante a definição dos requisitos. Essa atividade foi reexecutada para recuperar os artefatos necessários à verificação da consistência do guia produzido.

Como resultado das revisões realizadas, foram encontrados alguns defeitos:

- Multiplicidade de relacionamentos entre elementos do metamodelo equivocada. Esse tipo de defeito poderia levar a elaboração de perguntas incompletas no guia.
- Em alguns modelos de ubiquidade, o tipo do relacionamento entre elementos inconsistente com as descrições dos fatores de ubiquidade. Esse defeito foi propagado no processo de elaboração das perguntas do guia, resultando na construção de perguntas incorretas.
- Perguntas duplicadas com orientações diferentes no guia de definição dos requisitos de ubiquidade. Esse tipo de defeito poderia levar a inserção de informações inconsistentes ou redundantes na especificação de requisitos.

O Apêndice A apresenta a lista completa com os defeitos encontrados nas revisões.

A Seção 3.3 descreve a estratégia adotada na elaboração do *UbiModel*.

3.3 Estratégia de Elaboração

Para a elaboração do *UbiModel* foram extraídos os elementos e relações a partir das perguntas de *UbiCheck* (PINTO, 2009). Inicialmente nenhuma crítica foi

realizada, pois a preocupação era obter uma versão inicial com todos os elementos previstos em *UbiCheck*, para a partir deste entender melhor esses conceitos e relações sob uma nova perspectiva com o objetivo de aprimorar o modelo, eliminando, inserindo e/ou alterando elementos para fornecer uma estrutura de apoio efetiva a especificação de requisitos.

Algumas perguntas do guia eram de difícil compreensão e levaram à busca do entendimento nos trabalhos de PINTO (2009) e SPÍNOLA (2010), observando-se os modelos de características de ubiquidade, glossários e direcionamento das perguntas. Foi observado nesses trabalhos a tentativa de definir os conceitos através de um glossário e, posteriormente, definir orientações para apoiar o entendimento de cada pergunta. Apesar disso, o guia manteve-se em um nível de abstração elevado, pois tanto o glossário quanto as orientações não foram completamente detalhados de forma que não ocorressem problemas de entendimento no guia. Além disso, foram encontrados alguns problemas com esses artefatos, como conceitos ambíguos ou ausentes, redundância, orientações que “repetem” a pergunta ou não detalham o que se espera na resposta.

Existem perguntas no guia *Ubicheck* que aparecem em mais de uma característica para capturar o mesmo requisito, pois existem áreas comuns em muitas dessas características. Por exemplo, Interoperabilidade Espontânea, envolve serviços fornecidos ou serviços requeridos pela aplicação? Nesse caso foi determinado que Interoperabilidade Espontânea considera os serviços requeridos e os serviços fornecidos são considerados na Onipresença de Serviços. Sendo assim, foi realizado um trabalho no sentido de eliminar essas sobreposições pela definição de limites de cobertura de cada característica, de forma que cada conceito seja tratado unicamente por cada característica.

Em razão dessas questões, todas as definições foram revistas e, quando necessário, redefinidas de uma forma mais precisa a fim de evitar a inserção de defeitos na especificação devido a interpretações diferentes do que se espera na definição do requisito. Foi realizado um esforço no sentido de interpretar as características, fatores e diversos conceitos com o objetivo de preencher as lacunas existentes para se obter um modelo com um maior nível de qualidade com elementos mais detalhados, de forma que todas as questões presentes em *UbiCheck*, de certa forma, continuem presentes no modelo evoluído, ou seja, até que todos os elementos fossem suficientes para responder todas as perguntas do guia.

Foi elaborado um glossário com os termos da computação ubíqua utilizados por *UbiModel* para que eles pudessem ser explicados, com a finalidade de melhorar o entendimento ao especificar os requisitos desse domínio.

A Seção 3.4 apresenta a versão final do metamodelo de acordo com a interpretação que foi dada para cada característica de ubiquidade na elaboração do *UbiModel*.

3.4 Modelo Evoluído

A seguir são apresentadas as definições das características de ubiquidade, as visões parciais do metamodelo referentes a essas características e as definições de cada elemento do metamodelo, segundo a interpretação dada a partir das decisões tomadas em função dos problemas encontrados nas versões preliminares do *UbiModel*. Além disso, são apresentados cenários onde cada característica pode ser observada. A Tabela 3-1 apresenta todos os elementos da versão final do *UbiModel*. Na primeira coluna encontram-se os elementos originais, ou seja, os elementos considerados explicitamente em *UbiCheck*, enquanto na segunda coluna encontram-se os elementos que surgiram a partir do detalhamento dos conceitos originais para aprimorar o modelo.

Tabela 3-1 – Elementos do *UbiModel*

Elementos originais	Elementos novos
activity	adaptation
actor	adaptationalternative
availability	behavior
contextinformation	condition
datasource	context
device	dataformat
devicegroup	dynamic
devicesearch	hardwarerequirement
event	interaction
hardware	interactionpattern
identification	migration
identity	own
infraestructureinformation	preference
inputdataformat	protocol
interoperability	requirement
location	restriction
outputdataformat	softwarerequirement
physicalinformation	static
service	systemaction
serviceprovided	systemadaptation
servicerequired	trigger
servicesearch	
software	
systeminformation	
transformation	
user	
userinformation	

Nas figuras 3-1 a 3-12 os elementos em cinza representam os conceitos que apareceram em função das soluções dadas para os problemas encontrados nas versões iniciais do *UbiModel*, ou seja, representam algumas das contribuições desse trabalho.

3.4.1 Sensibilidade ao Contexto

Comportamento esperado do sistema frente a um contexto que se apresenta. Um contexto é um cenário de utilização do sistema, em outras palavras, em sua utilização, sempre haverá algum usuário participando de alguma atividade em algum lugar, sob determinadas condições do ambiente. O sistema pode se comportar de maneira distinta para cada contexto que se configura.

Ex.: Ao detectar no laboratório de informática que a temperatura do ambiente ultrapassou os 50° C, o sistema dispara o alarme, aciona o sistema de refrigeração extra e envia email para o responsável.

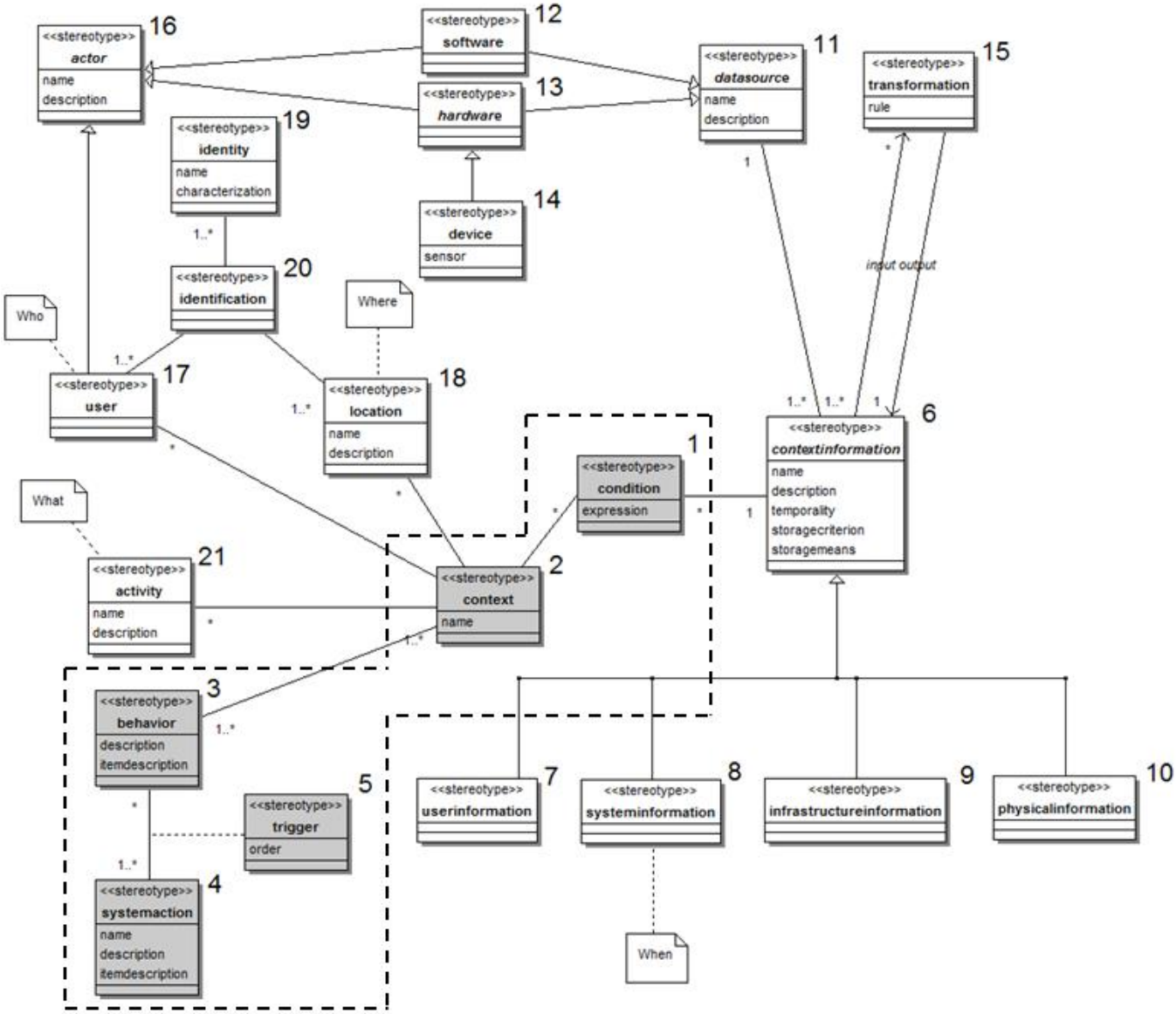


Figura 3-1–Visão do *UbiModel* para Sensibilidade ao Contexto

Interpretação do modelo:

É importante descrever as informações relacionadas aos elementos do ambiente relevantes ao sistema. As informações de contexto podem ser do tipo física quando estão relacionadas ao ambiente (temperatura, iluminação, ruído,...), podem ser informações de sistema quando estão relacionadas às restrições de infraestrutura (uso de memória, cpu, tamanho de tela,...), podem ser do tipo infraestrutura quando estão relacionadas ao correto funcionamento da infraestrutura (estado de funcionamento dos sensores,...) e finalmente podem ser do tipo usuário quando estão relacionadas aos usuários do sistema (temperatura corporal, pressão sanguínea, batimento cardíaco,...). Deve-se informar a fonte de dados ou a transformação na qual a informação de contexto tem origem. É importante também determinar a validade das informações em relação ao tempo, além do critério e do meio de armazenamento

das informações. Adicionalmente, deve-se descrever o que o sistema deve fazer ao perceber determinados contextos. Um contexto é um cenário de utilização do sistema. Em outras palavras, na utilização do sistema, sempre haverá algum usuário participando de alguma atividade em algum lugar, sob determinadas condições do ambiente. Quando relevantes, deverão ser especificados para cada contexto quais são os usuários, atividades, localizações e condições que o definem. A partir desses contextos o sistema pode se comportar de maneira distinta para cada um ou para cada conjunto de contextos determinado.

Alguns elementos surgiram em função da interpretação:

- O elemento ***condition*** (elemento 1 – Figura 3-1 surge pela necessidade de representar informações sobre o estado desejado do ambiente, em função de uma informação de contexto. Ex.: temperatura do ambiente > 50°C. A condição é um dos elementos que compõem o contexto.
- O elemento ***context*** (elemento 2 – Figura 3-1) surge pela necessidade de representar os cenários de utilização do sistema, o que não ocorre explicitamente em *UbiCheck*.
- O elemento ***behavior*** (elemento 3 – Figura 3-1) surge pela necessidade de representar, a partir dos contextos, o comportamento do sistema ao percebê-los. Um comportamento agrega um conjunto de ações do sistema que devem ocorrer em uma ordem determinada.
- O elemento ***systemaction*** (elemento 4 – Figura 3-1) surge pela necessidade de representar as ações que o sistema deve ser capaz de realizar.
- O elemento ***trigger*** (elemento 5 – Figura 3-1) para representar a ordem das ações que o sistema deve realizar, ao perceber um contexto.

A seguir, são apresentadas as definições dos elementos do modelo final a luz da interpretação dada para a característica:

contextinformation(elemento 6 – Figura 3-1): Informação que está relacionada aos elementos do ambiente controlado pelo sistema. Ela pode ser obtida através de fontes de dados ou por meio de transformações de outras informações de contexto. Deve ser considerado o momento que a informação é obtida, além do critério e do meio de armazenamento da informação.

userinformation(elemento 7 – Figura 3-1): Informações de contexto relacionadas aos usuários.

systeminformation(elemento 8 – Figura 3-1): Informações de contexto relacionadas às restrições de infraestrutura (uso de memória, cpu, tamanho de tela, energia, largura de banda disponível).

infrastructureinformation(elemento 9 – Figura 3-1): Informações de contexto relacionadas ao correto funcionamento da infraestrutura (estado de funcionamento dos sensores, por exemplo).

physicalinformation(elemento 10 – Figura 3-1): Informações de contexto relacionadas ao ambiente (temperatura, iluminação, ruído).

datasource(elemento 11 – Figura 3-1): Fonte de onde uma informação de contexto é obtida pelo sistema.

software(elemento 12 – Figura 3-1): Fonte de dados, do tipo software, que fornece informações para o sistema ou consome informações geradas pelo mesmo. Pode ser um ator do sistema.

hardware(elemento 13 – Figura 3-1): Fonte de dados, do tipo hardware, que fornece informações para o sistema ou consome informações geradas pelo mesmo. Pode ser um ator do sistema.

device(elemento 14 – Figura 3-1): Hardware, do tipo dispositivo, que fornece informações para o sistema ou consome informações geradas pelo mesmo. Pode ser um ator do sistema. Considera dispositivos do tipo sensor.

transformation(elemento 15 – Figura 3-1): define como informações de contexto obtidas via fonte de dados podem originar outra informação de contexto.

actor(elemento 16 – Figura 3-1): Usuário ou entidade que interage com o sistema.

user(elemento 17 – Figura 3-1): Indivíduo que interage com o ambiente controlado pelo sistema.

location(elemento 18 – Figura 3-1): Local controlado pelo sistema.

identity(elemento 19 – Figura 3-1): Forma de identificar os usuários nos locais controlados pelo sistema.

identification(elemento 20 – Figura 3-1): Localizações do ambiente, seus respectivos usuários e como esses usuários são identificados em cada localização.

activity(elemento 21 – Figura 3-1): Evento controlado pelo sistema (reunião, palestra, aula, etc.).

behavior(elemento 3 – Figura 3-1): Como o sistema deve se comportar ao perceber um determinado contexto.

trigger(elemento 5 – Figura 3-1): Ordem de execução da ação do sistema para um comportamento.

systemaction(elemento 4 – Figura 3-1): Algo que o sistema deve fazer (ligar alarme, enviar email, etc.).

condition(elemento 1 – Figura 3-1): Uma informação sobre o estado desejado do ambiente, em função de uma informação de contexto, definida por meio de uma expressão.

context(elemento 2 – Figura 3-1): Um cenário de utilização do sistema, composto por localizações do ambiente, usuários, atividades e condições determinada.

Exemplo de requisito

A seguir é apresentado um exemplo onde é demonstrada a correspondência dos termos do exemplo com os elementos do modelo para a característica Sensibilidade ao Contexto.

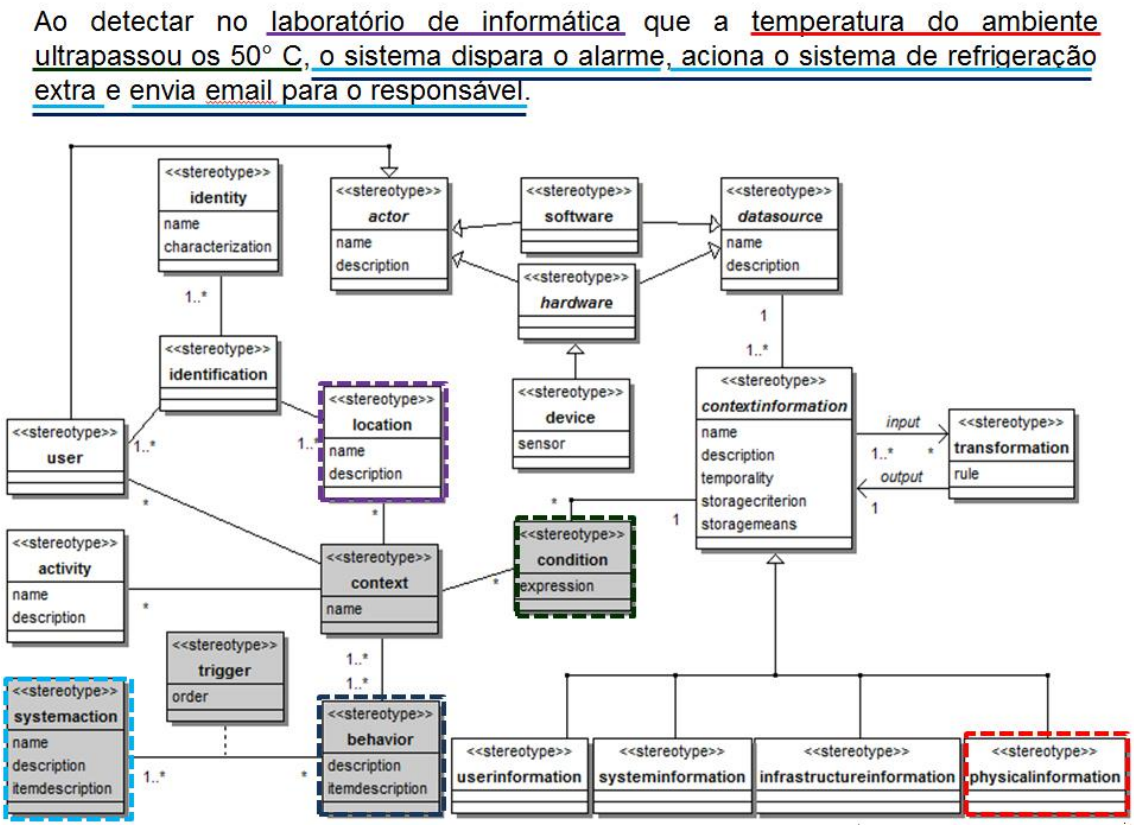


Figura 3-2–Exemplo de requisitos no modelo para Sensibilidade ao Contexto

3.4.2 Captura de Experiências

Comportamento que o sistema deve realizar em benefício do usuário a partir da identificação de um padrão de interação, que é a sequência de ações do usuário que ocorre de maneira recorrente, de acordo com um critério de análise.

Ex.: José chega em sua residência por volta das 18:00hs todos os dias. Entra pela porta principal, acende a luz da sala, liga o ar condicionado na temperatura de 18°C e liga a televisão no canal de esportes. O sistema ao perceber a repetição dessas ações passa a realizá-las automaticamente para o usuário. Portanto, quando José entrar em sua residência pela porta principal por volta das 18:00hs, o sistema irá acender a luz da sala, ligar o ar condicionado na temperatura de 18°C e ligar a televisão no canal de esportes.

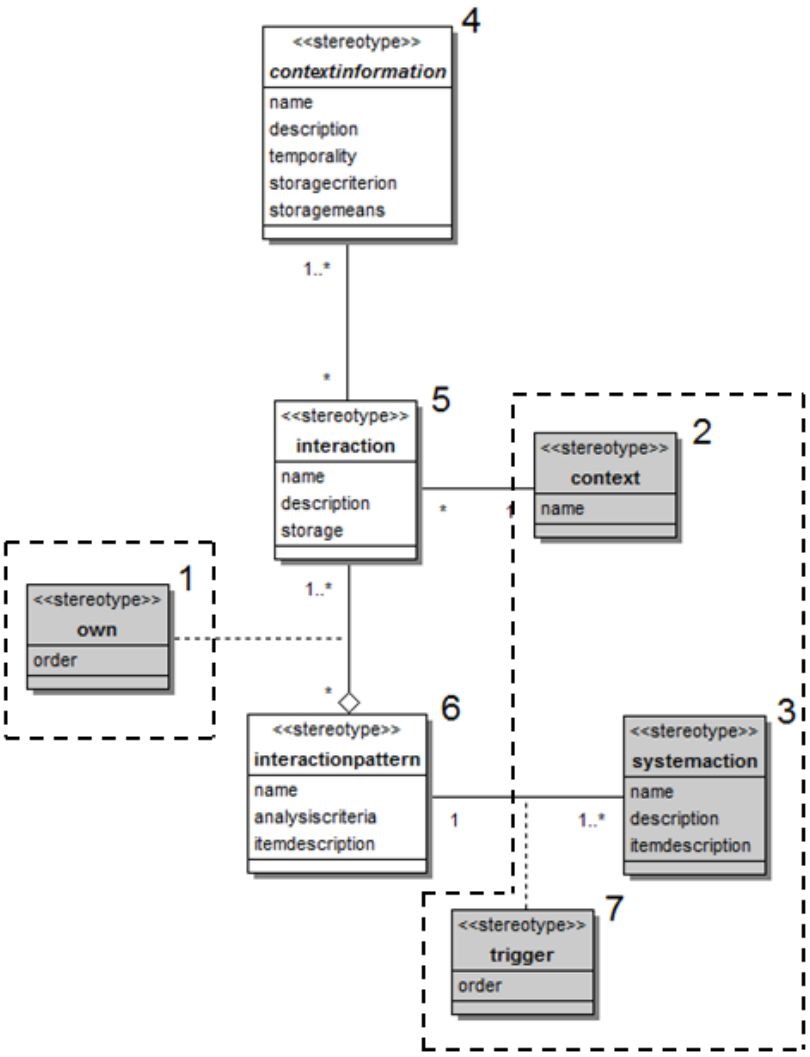


Figura 3-3–Visão do UbiModel para Captura de Experiências

Interpretação do modelo:

É importante descrever as sequências de interações que ocorrem de maneira recorrente, segundo critérios, que ao serem identificadas fazem com que o sistema realize ações em benefício do usuário. Interações são ações que o usuário realiza no ambiente controlado pelo sistema (acender a luz, ligar a TV, ligar o computador,...) e têm um conjunto de informações de contexto relacionadas. Além disso, elas ocorrem

em determinados contextos de utilização do sistema. Na definição dos padrões é importante informar a ordem em que as interações devem ocorrer e os critérios utilizados para determinar a validade de um padrão (tempo decorrido entre as interações, tempo decorrido entre as sequencias de interações,...). Na definição do que o sistema deve fazer na ocorrência de padrões, é importante informar a ordem das ações que devem ser realizadas.

Um novo elemento surgiu em função da interpretação:

- O elemento ***own***(elemento 1 – Figura 3-3) para representar a ordem em que as interações ocorrem dentro de um padrão e a ordem das ações que o sistema deve realizar a partir da identificação de um padrão. Esses aspectos não são considerados originalmente.

Os elementos ***context*** (elemento 2 – Figura 3-3), ***systemaction*** (elemento 3 – Figura 3-6) e ***contextinformation*** (elemento 4 – Figura 3-3) já foram descritos anteriormente por serem elementos comuns a outras características.

A seguir, são apresentadas as definições dos elementos do modelo final a luz da interpretação dada para a característica:

interaction(elemento 5 – Figura 3-3): Uma ação que o usuário realiza no ambiente controlado pelo sistema.

interactionpattern(elemento 6 – Figura 3-3): Sequência de interações que ocorre de maneira recorrente, de acordo com um critério, que ao ser identificada pelo sistema o mesmo deve realizar ações que beneficiem o usuário.

own(elemento 1 – Figura 3-3): Ordem da interação dentro de um padrão específico.

trigger(elemento 7 – Figura 3-3): Ordem de execução da ação do sistema para um padrão de interação.

Exemplo de requisito

A seguir é apresentado um exemplo onde é demonstrada a correspondência dos termos do exemplo com os elementos do modelo para a característica Captura de Experiências.

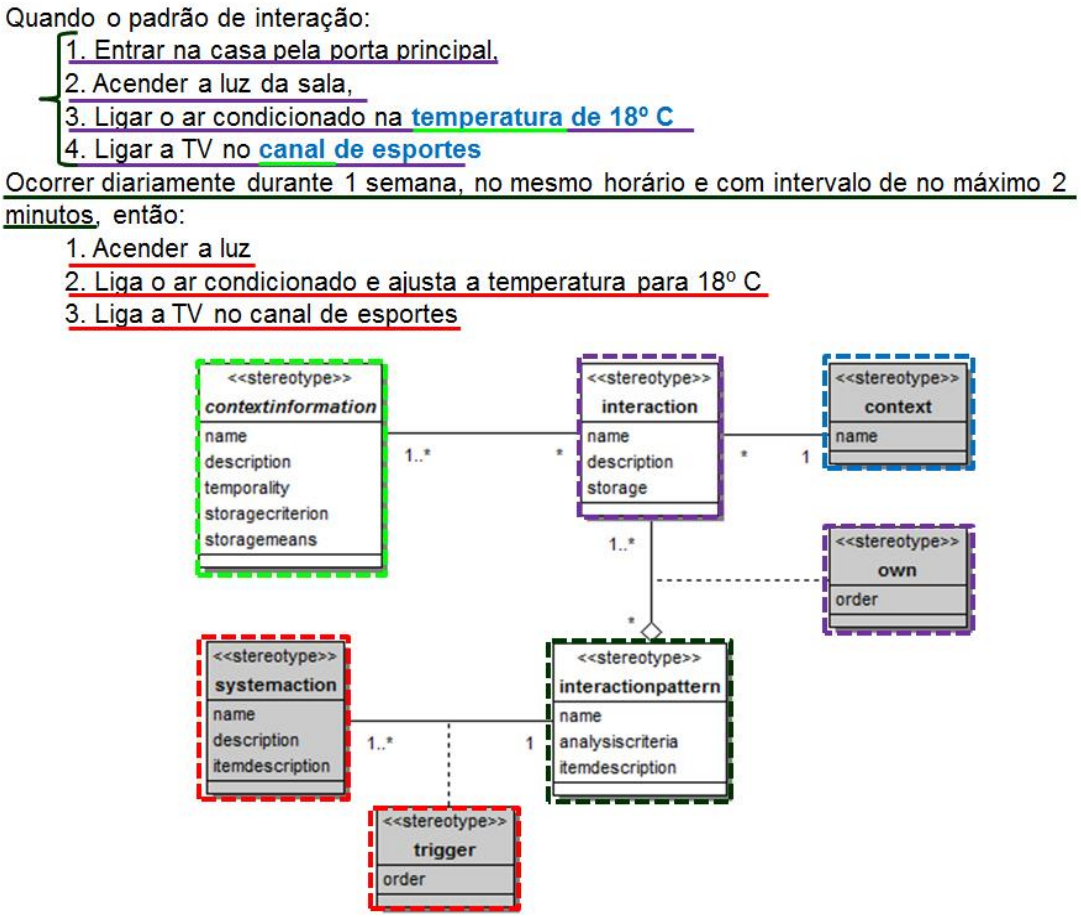


Figura 3-4–Exemplo de requisitos no modelo para Captura de Experiências

3.4.3 Comportamento Adaptável

Comportamento do sistema pode ser adaptado, dinamicamente, de acordo com preferências do usuário, que estabelecem alguns limites de trabalho em determinados contextos de execução do sistema. Em outras palavras, dentro de um determinado contexto o usuário pode preferir que o sistema apresente alguma característica específica, caso isso não ocorra ele pode se adaptar de alguma forma. Resumindo, a preocupação está em definir o que vai acontecer caso alguma restrição não seja satisfeita. Adaptações que envolvam a utilização de outros dispositivos estão fora do escopo dessa característica.

Ex.: Um usuário está em uma videoconferência, que tem como preferência não perder o áudio. Portanto, se a taxa de transmissão cair muito, uma adaptação seria interromper o vídeo e deixar a rede apenas com o áudio.

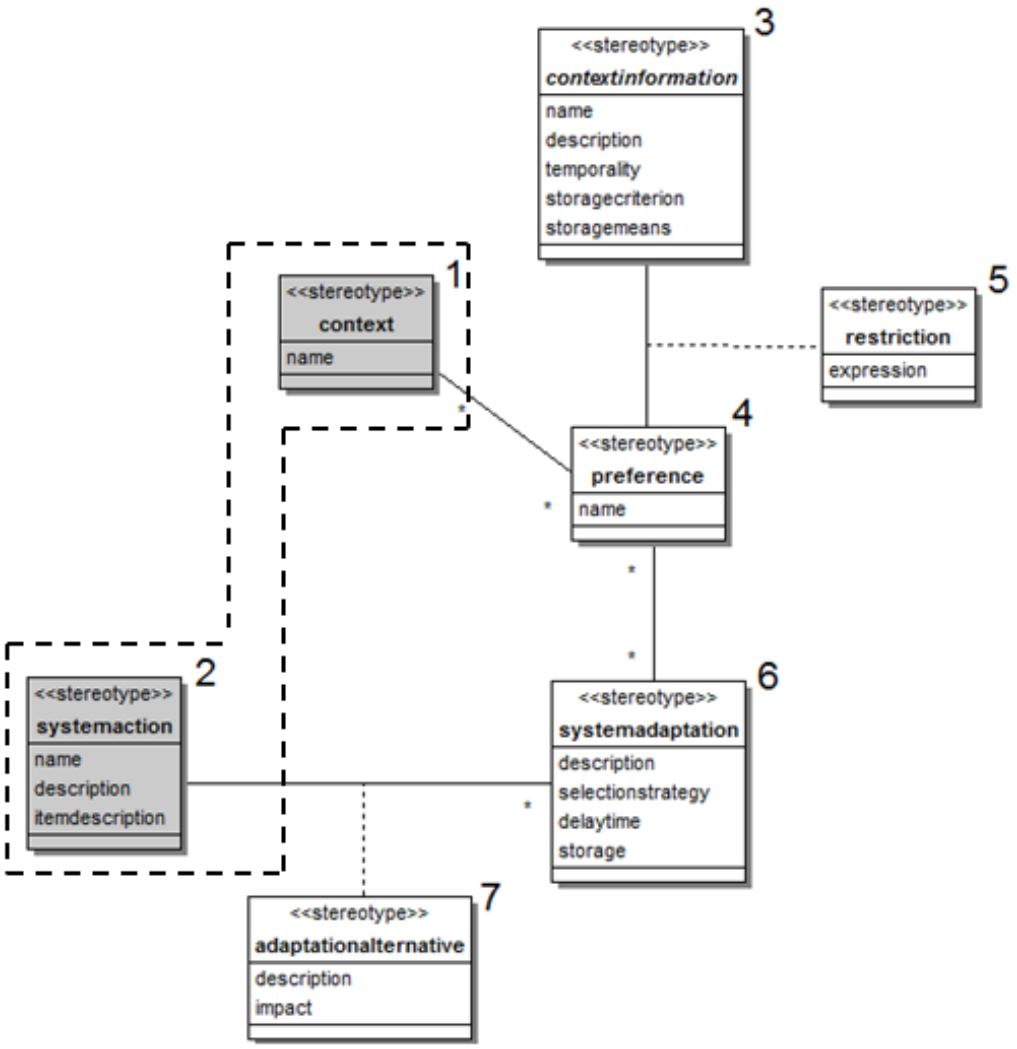


Figura 3-5–Visão do UbiModel para Comportamento Adaptável

Interpretação do modelo:

É importante descrever as adaptações que o sistema deve realizar caso as preferências sejam violadas. Uma preferência é uma condição do sistema em função de uma informação de contexto que deve ser obedecida para determinados contextos. As adaptações são compostas por alternativas de adaptação que o sistema deve selecionar considerando-se seus impactos no sistema. É importante informar o tempo que o sistema deve aguardar para a efetivação de uma adaptação, sendo que o contexto ao qual a preferência é violada pode se desfazer rapidamente, não havendo a necessidade da adaptação. Deve-se informar a estratégia de seleção utilizada pelo sistema para selecionar uma alternativa de adaptação e como as informações sobre as adaptações realizadas devem ser armazenadas.

Os elementos ***context***(elemento 1 – Figura 3-5), ***systemaction***(elemento 2 – Figura 3-5) e ***contextinformation*** (elemento 3 – Figura 3-5) já foram descritos anteriormente por serem elementos comuns a outras características.

A seguir, são apresentadas as definições dos elementos do modelo final a luz da interpretação dada para a característica:

preference(elemento 4 – Figura 3-5): Característica do ambiente, para um contexto específico, que ao ser violada faz com que o sistema tente se adaptar.

restriction(elemento 5 – Figura 3-5): Limita uma informação de contexto a valores determinados por meio de expressões que representam as preferências que o sistema deve garantir.

systemadaptation(elemento 6 – Figura 3-5): Adaptação do sistema para uma preferência violada, que considera a estratégia de seleção de uma alternativa de adaptação, o tempo de espera até que a adaptação seja realizada de fato e o armazenamento das informações sobre a adaptação.

adaptationalternative(elemento 7 – Figura 3-5): Forma como uma adaptação pode ser realizada, considerando seu impacto sobre o sistema.

Exemplo de requisito

A seguir é apresentado um exemplo onde é demonstrada a correspondência dos termos do exemplo com os elementos do modelo para a característica Comportamento Adaptável.

Ao realizar uma videoconferência, se a taxa de transmissão ficar abaixo de 10fps
então desligar o vídeo e manter somente o áudio.

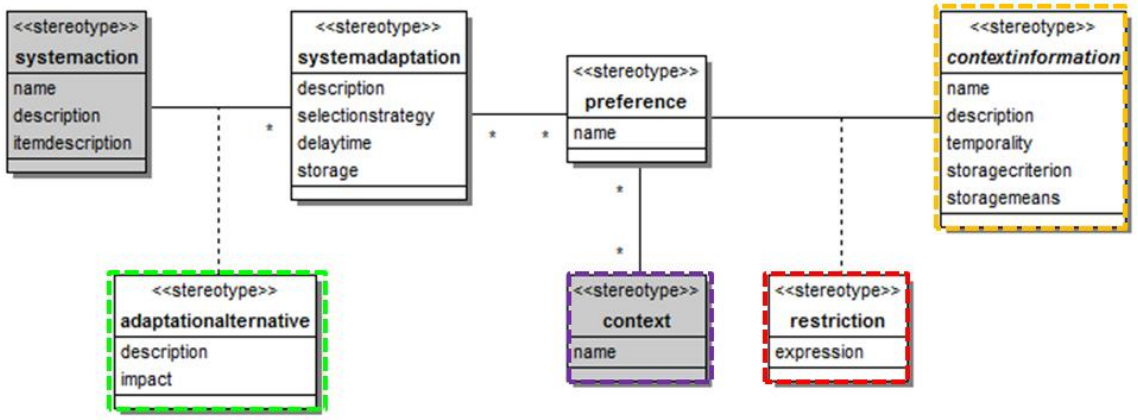


Figura 3-6–Exemplo de requisitos no modelo para Comportamento Adaptável

3.4.4 Onipresença de Serviços

Capacidade do sistema em disponibilizar algumas de suas funcionalidades como serviços a serem utilizados por outras aplicações.

Ex.: Um consumidor ao entrar em um shopping center recebe em seu smartphone informações de apoio a compra de produtos, por exemplo: informação sobre promoções e informação sobre a localização de lojas.

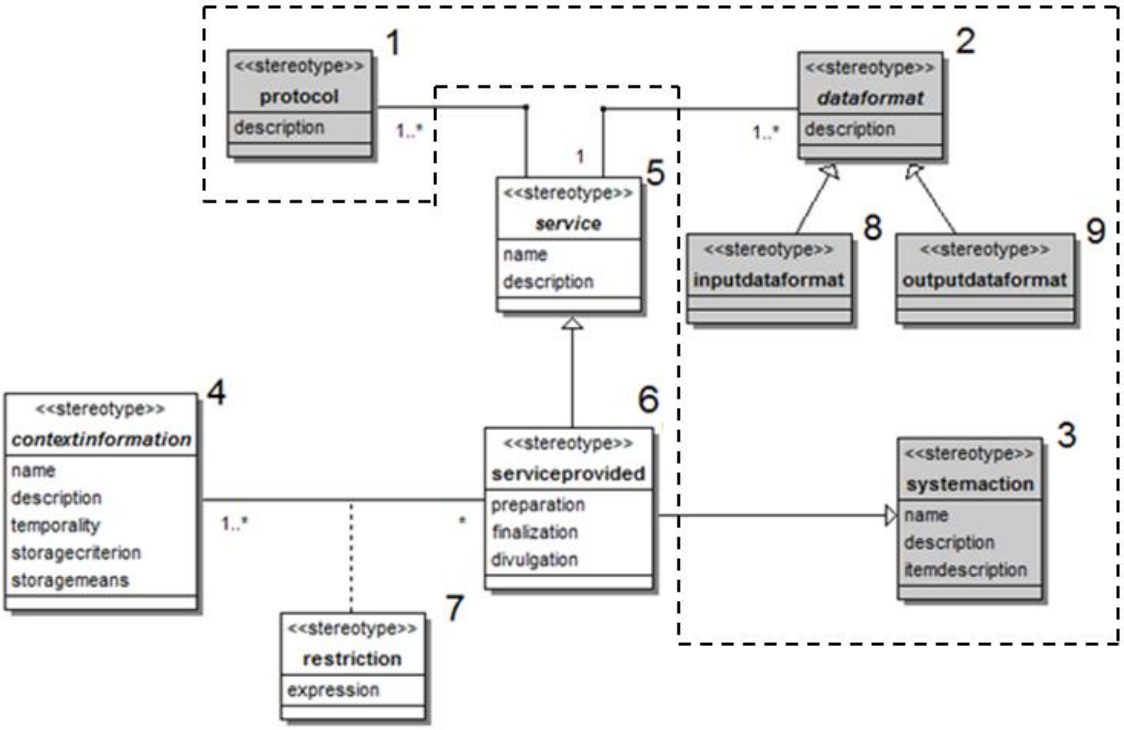


Figura 3-7–Visão do *UbiModel* para Onipresença de Serviços

Interpretação do modelo:

É importante descrever as ações do sistema que devem ser disponibilizadas como serviços a outras aplicações. Além de informações de nome e descrição dos serviços, é importante descrever o processo de inicialização dos mesmos, o que é feito antes dos serviços iniciarem, o que ocorre em suas finalizações, quais as condições mínimas para que estejam disponíveis e como são divulgados. Informações básicas sobre os serviços como protocolos e formato de dados também devem ser descritos.

Alguns elementos surgiram em função da interpretação:

- Os elementos ***protocol*** (elemento 1 – Figura 3-7) e ***dataformat*** (elemento 2 – Figura 3-8) para representar os protocolos e os formatos dos dados dos serviços fornecidos e utilizados para que a comunicação entre os sistemas seja possível. Essas informações não são consideradas originalmente.

Os elementos ***systemaction*** (elemento 3 – Figura 3-7) e ***contextinformation*** (elemento 4 – Figura 3-8) já foram descritos anteriormente por serem elementos comuns a outras características.

A seguir, são apresentadas as definições dos elementos do modelo final a luz da interpretação dada para a característica:

service(elemento 5 – Figura 3-7): Funcionalidade de um sistema disponível a outros sistemas fora dos seus limites.

serviceprovided(elemento 6 – Figura 3-7): Serviço do sistema disponível a outros sistemas.

restriction(elemento 7 – Figura 3-7): Limita uma informação de contexto a valores determinados por meio de expressões que representam as condições mínimas para que o serviço esteja disponível.

protocol(elemento 1 – Figura 3-7): Forma como os sistemas se comunicam. Comunicação se dá via serviços.

dataformat(elemento 2 – Figura 3-7): Tamanho e tipo do dado utilizado por um serviço.

inputdataformat(elemento 8 – Figura 3-7): Formato de dado do tipo entrada.

outputdataformat(elemento 9 – Figura 3-7): Formato de dado do tipo saída.

Exemplo de requisito

A seguir é apresentado um exemplo onde é demonstrada a correspondência dos termos do exemplo com os elementos do modelo para a característica Onipresença de Serviços.

Disponibilizar serviço de visualização de fotos no formato JPG e BMP usando o protocolo bluetooth quando:
Número de clientes usando o serviço for igual a zero

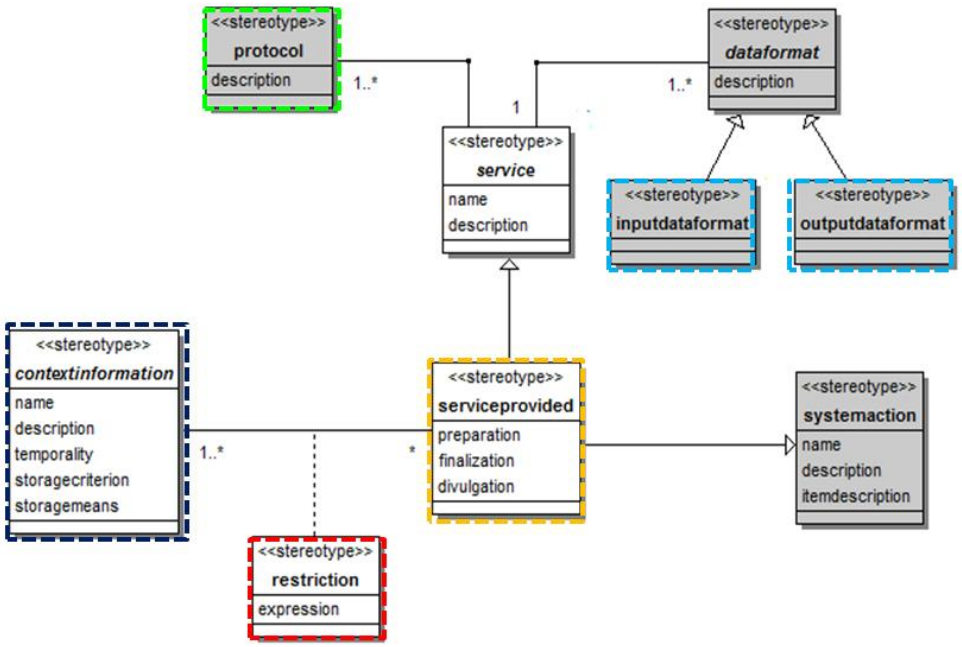


Figura 3-8–Exemplo de requisitos no modelo para Onipresença de Serviços

3.4.5 Heterogeneidade de Dispositivos

Capacidade do sistema em migrar todas ou parte de suas funcionalidades para dispositivos heterogêneos. Existem 2 situações que caracterizam uma migração. A primeira chamada de Migração Estática, o usuário é responsável por iniciar o sistema usando dispositivos heterogêneos e a partir de então o sistema se adapta de acordo com as características do dispositivo. A outra situação chamada de Migração Dinâmica caracteriza-se por ser iniciada pelo sistema, a partir da ocorrência de algum evento (usuário bloquear o computador, fechar a tampa do notebook, etc.). Após o evento ser percebido, o sistema busca e seleciona dispositivos no ambiente capazes de executar o sistema, permitindo ao usuário continuar acessando o sistema.

Ex.: Migração Estática – Um usuário de uma rede social está utilizando o sistema por meio de um desktop em uma *lan house* e no meio de uma conversa via chat com um amigo seu tempo de utilização do computador expira e o mesmo fica bloqueado. Ao sair, ele acessa o sistema por meio de um smartphone e continua a conversa com o amigo do ponto onde haviam parado, agora com a limitação de utilizarem apenas texto para se comunicarem.

Ex.: Migração dinâmica – Um sistema executando em um determinado dispositivo, ao perceber que a carga de sua bateria está terminando, busca no ambiente dispositivos aptos a continuarem sua execução evitando que algum processamento importante seja interrompido.

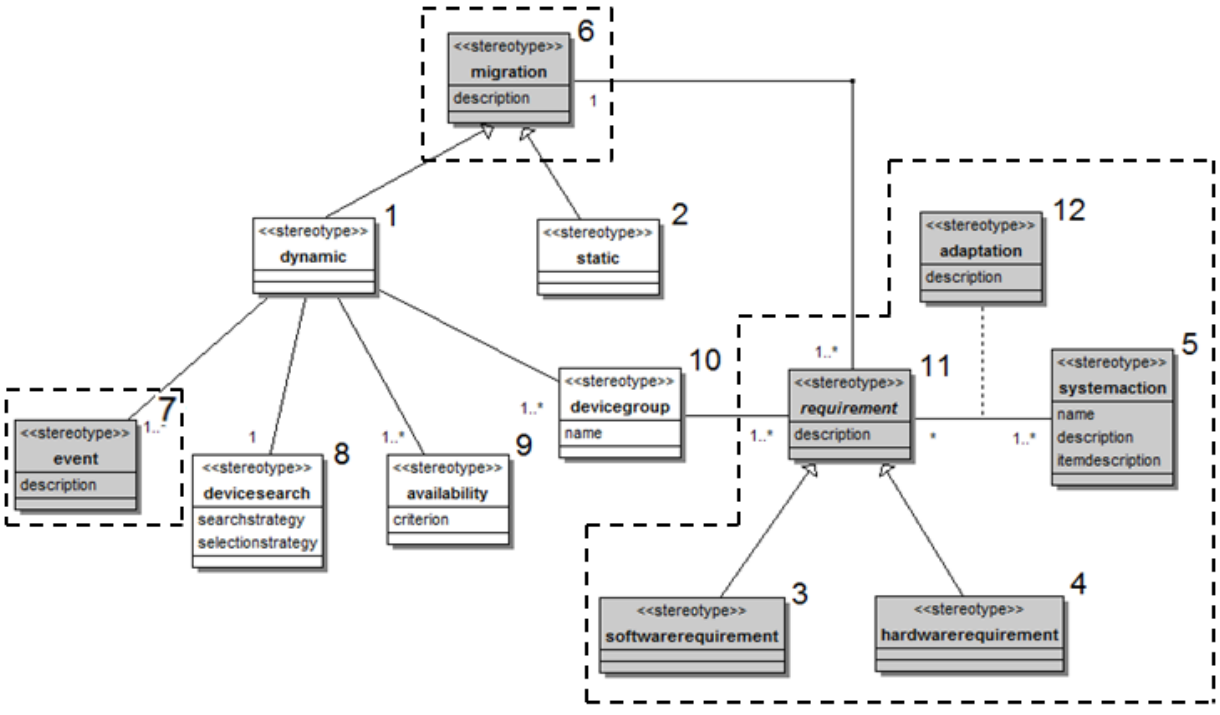


Figura 3-9–Visão do *UbiModel* para Heterogeneidade de Dispositivos

Interpretação do modelo:

É importante descrever as informações necessárias para que a aplicação possa migrar para outros dispositivos diferentes do utilizado originalmente pelo sistema. É importante descrever os requisitos que os dispositivos devem atender e as adaptações que devem ocorrer para que a migração seja possível. Existem 2 tipos de migração. Em um deles, o usuário é responsável por iniciar o sistema via dispositivo heterogêneo e a partir de então o sistema se adapta de acordo com as características do dispositivo. A esse tipo de migração demos o nome de Migração Estática. O outro tipo, denominado Dinâmica, caracteriza-se por ser iniciado pelo sistema, a partir da ocorrência de algum evento (usuário bloquear o computador, fechar a tampa do notebook,...). Após o evento ser percebido, o sistema busca e seleciona dispositivos no ambiente de acordo com estratégias e critérios de disponibilidade dos dispositivos que devem estar presentes na especificação. Na migração Dinâmica, diferentemente da migração Estática, os requisitos são organizados por grupos de dispositivos, ou seja, o sistema pode se adaptar de diferentes maneiras para cada grupo. Na migração Estática, o conjunto de requisitos é único, pela existência de apenas um conjunto de adaptações do sistema, portanto, é desnecessária a definição de grupos de dispositivos para esse tipo de migração.

Alguns elementos surgiram em função da interpretação:

- Os elementos ***dynamic***(elemento 1 – Figura 3-9) e ***static*** (elemento 2 – Figura 3-9) surgem pela necessidade de distinguir as 2 formas como uma migração pode ser realizada. Em *Ubicheck*, é dada ênfase a migração dinâmica, mesmo assim o conceito de evento que inicia a migração não é considerado, portanto foi adicionado a *UbiModel* o elemento ***event*** para capturar essa informação.
- Os elementos ***softwarerequirement*** (elemento 3 – Figura 3-9) e ***hardwarerequirement*** (elemento 4 – Figura 3-9) surgem pela necessidade de distinguir os 2 tipos de requisitos que um dispositivo deve atender para que a migração possa acontecer. Originalmente, não é feita essa distinção. Os requisitos determinam adaptações sobre o núcleo de funcionalidades que caracterizam a aplicação a ser migrada. Para que as adaptações possam ser capturadas, foi adicionado o elemento Adaptação ao modelo, para descrever uma adaptação para cada requisito associado a uma ação do sistema.

O elemento ***systemaction*** (elemento 5 – Figura 3-9) já foi descrito anteriormente por ser um elemento comum a outras características.

A seguir, são apresentadas as definições dos elementos do modelo final a luz da interpretação dada para a característica:

migration(elemento 6 – Figura 3-9): Capacidade do sistema em continuar sua execução em um outro dispositivo com características diferentes do dispositivo atual.

dynamic(elemento 1 – Figura 3-9): O sistema é responsável por perceber eventos que iniciam uma migração, a partir daí, o mesmo deve buscar e selecionar dispositivos no ambiente segundo estratégias e critérios de disponibilidade dos dispositivos. A migração é iniciada pela ocorrência de um evento.

static(elemento 2 – Figura 3-9): Sistema executado por dispositivos heterogêneos onde todas ou algumas funcionalidades são adaptadas de acordo com os requisitos de migração. O usuário é o responsável por iniciar o sistema no dispositivo heterogêneo.

event(elemento 7 – Figura 3-9): Algo que inicia o processo de migração dinâmica (deslocamento do usuário, etc.).

devicesearch(elemento 8 – Figura 3-9): Forma como o sistema busca os dispositivos, considerando-se uma estratégia de busca e seleção dos dispositivos disponíveis no ambiente.

availability(elemento 9 – Figura 3-9): Critério que define a disponibilidade de migração.

devicegroup(elemento 10 – Figura 3-9): Grupo de dispositivos compatível para a realização de uma migração caracterizado por um conjunto de requisitos.

requirement(elemento 11 – Figura 3-9): Requisitos que os dispositivos heterogêneos devem ter para que a migração seja possível.

softwarerequirement(elemento 3 – Figura 3-9): Requisito relacionado a software.

hardwarerequirement(elemento 4 – Figura 3-9): Requisito relacionado a hardware.

adaptation(elemento 12 – Figura 3-9): Adaptação de uma ação do sistema de acordo com os requisitos de migração.

Exemplo de requisito

A seguir é apresentado um exemplo onde é demonstrada a correspondência dos termos do exemplo com os elementos do modelo para a característica Heterogeneidade de Dispositivos.

Ao realizar uma videoconferência, se a taxa de transmissão ficar abaixo de 10fps então desligar o vídeo e manter somente o áudio.

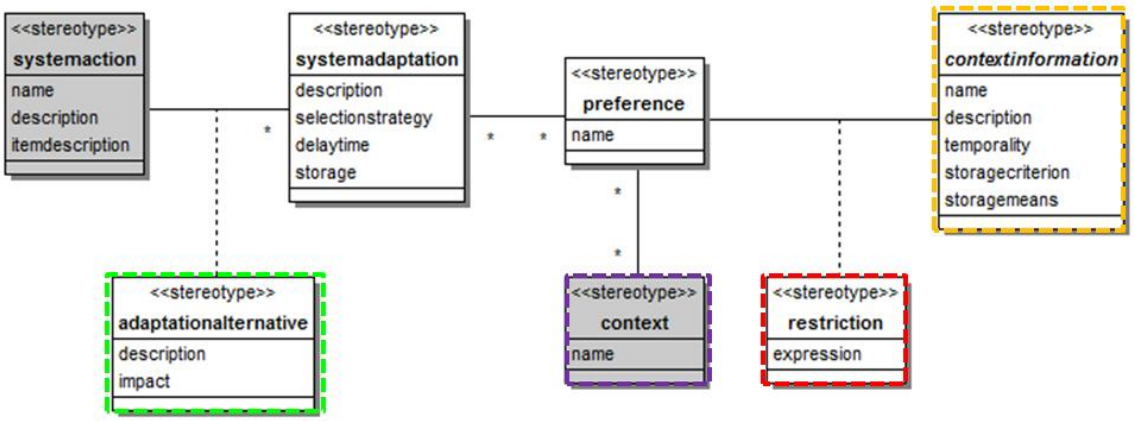


Figura 3-10–Exemplo de requisitos no modelo para Heterogeneidade de Dispositivos

3.4.6 Interoperabilidade Espontânea

Capacidade do sistema interoperar com outros sistemas ou dispositivos. Podem existir funcionalidades que devem interagir com elementos externos para fazer algo para o usuário. Nesse caso, o sistema deve ser capaz de buscar e selecionar serviços que executem uma funcionalidade que não possa ser executada no dispositivo atual ou que executem a funcionalidade de forma mais adequada, ou seja, que forneça alguma vantagem em relação a execução no dispositivo atual.

Ex.: Um usuário quer visualizar uma foto utilizando um dispositivo com tela de apenas 3'', porém a foto é grande e a tela não é capaz de exibi-la. O sistema percebe em sua proximidade a existência de uma tela com dimensões adequadas que oferece serviço de display. Com isso, o sistema utiliza-se desse serviço para exibição da foto.

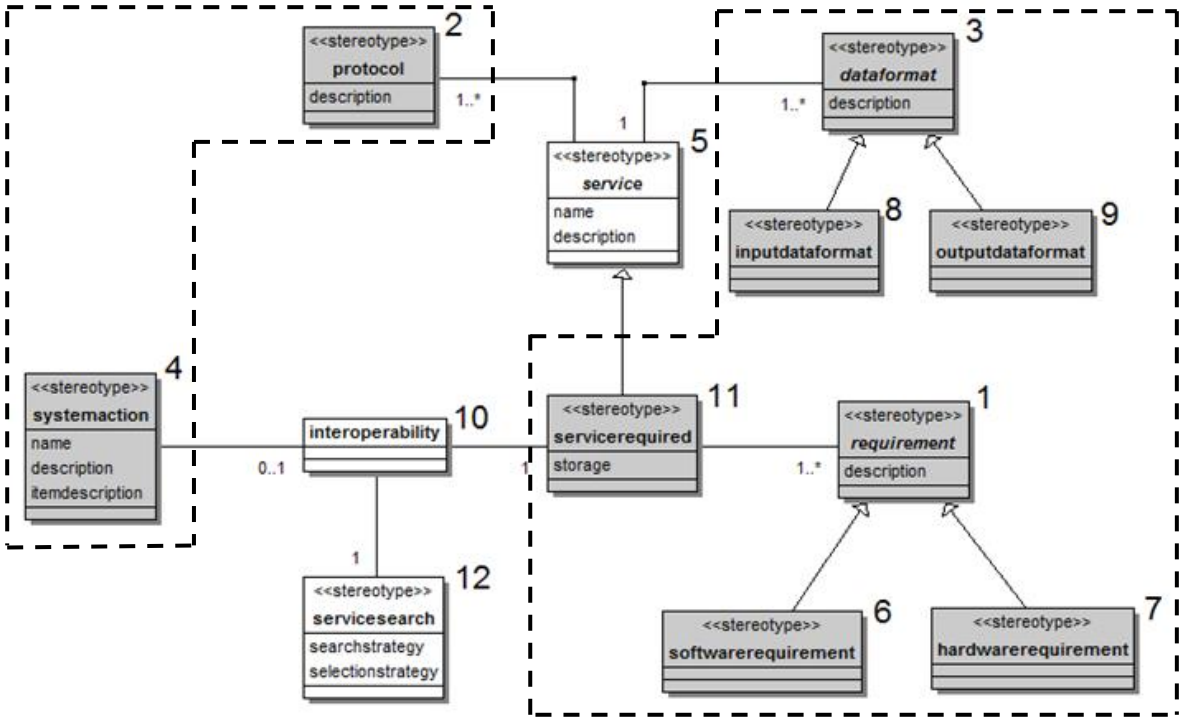


Figura 3-11–Visão do *UbiModel* para Interoperabilidade Espontânea

Interpretação do modelo:

É importante descrever os serviços externos que o sistema deve buscar e utilizar para atender as necessidades que ele não é capaz de atender, pela não realização das ações ou pela existência no ambiente de serviços que realizem as ações de maneira a trazer algum ganho para o sistema em relação a maneira como ele as realiza. Ambos os cenários são descritos por requisitos mínimos que os serviços devem apresentar para que sejam efetivamente utilizados. Além de informações de nome e descrição dos serviços, é importante descrever como as informações relevantes durante a utilização dos serviços serão armazenadas e as estratégias utilizadas pelo sistema para buscar e selecionar os serviços. Devem ser informadas quais ações do sistema os serviços a serem utilizados devem atender. Informações básicas sobre os serviços como protocolos e formato de dados também devem ser descritos.

Alguns elementos surgiram em função da interpretação:

- O elemento ***requirement*** (elemento 1 – Figura 3-11) surge pela necessidade de descrição dos requisitos que o serviço a ser utilizado deve atender tanto de hardware quanto de software quando o sistema não implementa a funcionalidade ou quando ela existe porém existem

outros serviços disponíveis no ambiente que forneçam alguma vantagem em sua utilização.

Os elementos ***protocol***(elemento 2 – Figura 3-11), ***dataformat*** (elemento 3 – Figura 3-11), ***systemaction*** (elemento 4 – Figura 3-11) ***service*** (elemento 5 – Figura 3-11), ***softwarerequirement*** (elemento 6 – Figura 3-11), ***hardwarerequirement*** (elemento 7 – Figura 3-11), ***inputdataformat*** (elemento 8 – Figura 3-11) e ***outputdataformat*** (elemento 9 – Figura 3-11) já foram descritos anteriormente por serem elementos comuns a outras características.

A seguir, são apresentadas as definições dos elementos do modelo final à luz da interpretação dada para a característica:

interoperability(elemento 10 – Figura 3-11): Capacidade do sistema em utilizar serviços de outros sistemas.

servicerequired(elemento 11 – Figura 3-11): Serviço disponível por outros sistemas a ser utilizado para atender a uma demanda do sistema principal, considerando-se os requisitos mínimos para sua utilização.

servicesearch(elemento 12 – Figura 3-11): Forma como o sistema procura por serviços no ambiente, considerando-se uma estratégia para busca e para seleção dos serviços disponíveis.

requirement(elemento 1 – Figura 3-11): Requisitos mínimos para que um serviço externo seja utilizado pelo sistema.

Exemplo de requisito

A seguir é apresentado um exemplo onde é demonstrada a correspondência dos termos do exemplo com os elementos do modelo para a característica Interoperabilidade Espontânea.

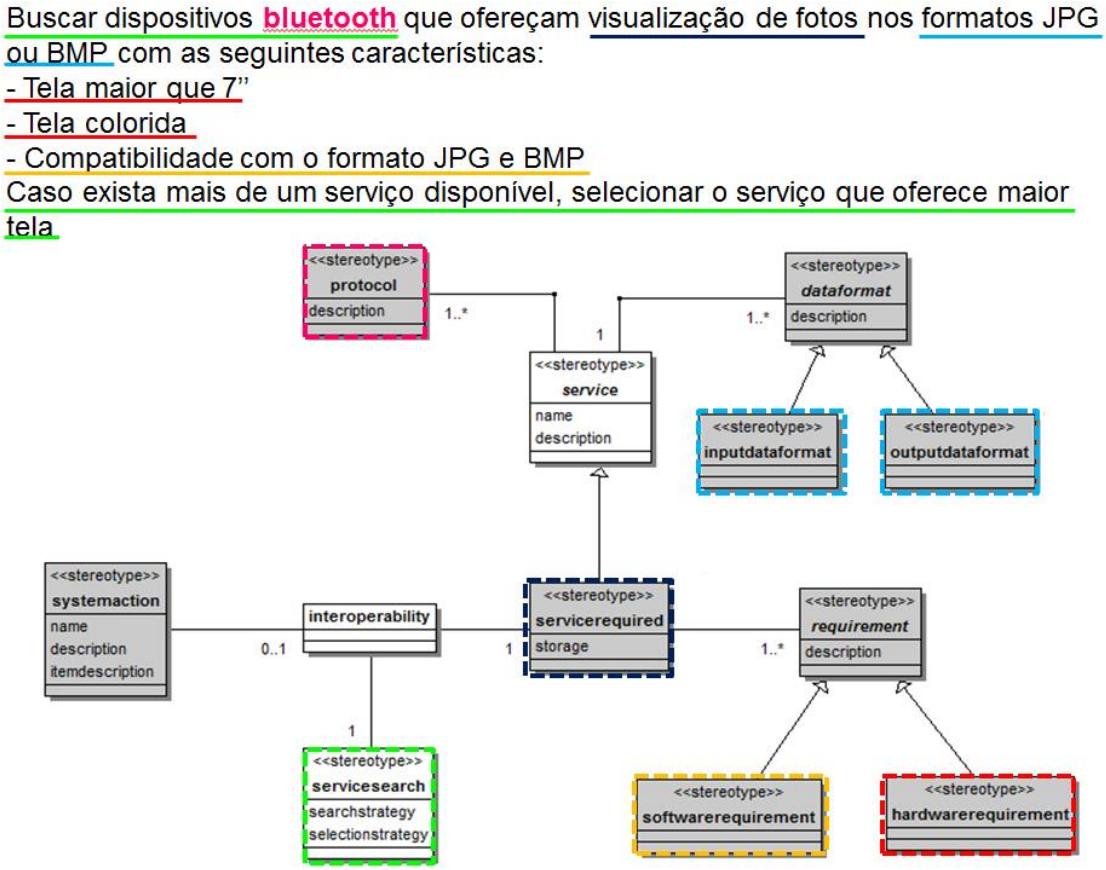


Figura 3-12–Exemplo de requisitos no modelo para Interoperabilidade Espontânea

3.5 Conclusão

Este capítulo apresentou *UbiModel*: um metamodelo que organiza uma estrutura para capturar as informações necessárias na especificação de requisitos de ubiquidade. *UbiModel* foi elaborado a partir da interpretação e organização dos conceitos contidos em *Ubicheck* (PINTO, 2009; SPINOLA, 2010). *UbiModel* se propõe a dar um direcionamento mais concreto acerca da especificação de requisitos de ubiquidade através da modelagem dos elementos necessários para a especificação desse tipo de requisito.

A primeira versão do metamodelo foi desenvolvida basicamente através da tradução literal dos elementos de *UbiCheck*, ou seja, todos os elementos e relações presentes em *UbiCheck* aparecem no metamodelo inicial, sem nenhum tipo de análise ou crítica. A partir dessa versão, foi realizado um esforço para detalhar o metamodelo através de consultas aos artefatos produzidos por PINTO (2009) e SPÍNOLA (2010) na busca de informações que apoiassem o processo de elaboração do *UbiModel*. Ao consultar esses artefatos alguns problemas foram encontrados. Para resolver tais problemas, alguns conceitos foram redefinidos, novos elementos surgiram, alguns

sumiram ao serem englobados por outros elementos e outros foram modificados de acordo com as novas definições.

Com o objetivo de garantir a completude do *UbiModel* em relação a *UbiCheck*, foi realizado um trabalho para verificar se os elementos do *UbiModel* capturavam todas as informações necessárias que respondessem as perguntas do guia original. Esse processo se repetiu até que o objetivo fosse atingido.

No próximo capítulo, será apresentado como *UbiModel* foi elaborado para ser utilizado em conjunto com *UCModel* (MASSOLLAR, 2011), um metamodelo que apoia a especificação de requisitos para descrição do comportamento esperado do sistema.

4 Integração entre os Modelos *UbiModel* e *UCModel*

Neste capítulo, é apresentado como o metamodelo UbiModel foi elaborado para considerar a descrição dos requisitos de ubiquidade sob o ponto de vista do comportamento do sistema, de forma que a especificação possa ser utilizada em etapas posteriores a etapa de requisitos no processo de desenvolvimento do sistema.

4.1 Introdução

Tanto o trabalho realizado por PINTO (2009) e SPÍNOLA (2010) quanto o presente trabalho tentam, de certa forma e em níveis diferentes de abstração, interpretar o conjunto de perguntas do guia de especificação de requisitos de ubiquidade e dar um direcionamento a questões abertas, ou seja, questões em um alto nível de abstração que dependem muito da interpretação de quem está especificando os requisitos. Do ponto de vista de desenvolvimento de software, o que se tem como resultado da aplicação dessas abordagens são requisitos, representados basicamente como necessidades e regras de alto nível que precisam ser atendidas. Desta forma, os requisitos especificados com essas abordagens equivalem a uma lista dos comportamentos que o usuário deseja que o sistema apresente, sem o detalhamento de como esse comportamento deve ser realizado pelo sistema. Esses requisitos representam, de uma maneira geral, a ideia de demandas, necessidades e restrições do ponto de vista do usuário e são descritos de modo que sejam compreensíveis pelos usuários do sistema.

A princípio, é importante entender o que o sistema deve fazer, sem a preocupação de detalhar como isso deve ser feito. O objetivo é que a especificação sirva de base para o entendimento entre clientes e desenvolvedores acerca do que o sistema deve contemplar. Para que a especificação dos requisitos possa ser utilizada para apoiar de forma efetiva atividades em outras fases do projeto de desenvolvimento, como projeto e implementação, é preciso que esses requisitos sejam descritos em mais detalhes sob o ponto de vista do sistema (SOMMERVILLE, 2007), ou seja, é preciso trabalhar em um nível intermediário de definição que mostre efetivamente qual é o comportamento que se espera do sistema para atender esses

requisitos, de forma que os responsáveis pelas fases posteriores do projeto tenham informações suficientes e adequadas para realização de suas atividades e produção de artefatos com quantidade reduzida de defeitos, redução de retrabalho e custos (WIEGERS, 2003). Os requisitos devem ser detalhados sob o ponto de vista do sistema, com o objetivo de facilitar a transição do software para o mundo computacional. A Seção 4.2 apresenta a abordagem utilizada para descrição dos requisitos de ubiquidade sob o ponto de vista do comportamento do sistema.

4.2 Detalhamento dos Requisitos

A partir do contexto apresentado na Seção 4.1, a ideia é tentar descrever os comportamentos em algum outro formato que possibilite o detalhamento adequado da especificação dos requisitos. A partir da lista de requisitos definida através do *UbiModel*, que tenta fornecer uma lista mais elaborada em comparação com aquela fornecida por *UbiCheck* – checklist utilizado pela abordagem de PINTO (2009) e SPÍNOLA (2010) – buscou-se alternativas de detalhamento desses requisitos, pois eles ainda se encontram descritos, mesmo que de forma mais objetiva, sob o ponto de vista do usuário. Vários mecanismos podem ser utilizados com esse objetivo, tais como, algoritmos estruturados (CORMEN *et. al.*, 2002), casos de uso, diagramas de atividades e diagramas de estado (OMG, 2010). Nesse trabalho, são utilizados casos de uso e diagramas de atividades, pois eles permitem a descrição do comportamento do sistema em um nível de detalhe que permite que a especificação seja utilizada em fases posteriores no processo de desenvolvimento do software. Para isso, considerou-se a proposta apresentada por MASSOLLAR (2011) que permite a utilização de diagramas de atividades estereotipados para descrição de casos de uso, além da utilização de diagramas de atividades normais, de acordo com o padrão da UML (OMG, 2010), para descrição de comportamentos não associados a casos de uso. Para mapear todos os conceitos associados a casos de uso, de forma que os mesmos possam ser descritos utilizando-se diagrama de atividades, foi definido o metamodelo *UCModel* (MASSOLLAR, 2011), que torna possível a descrição de um caso de uso completo com todos os elementos relacionados, através de diagramas de atividades. Com isso, os requisitos de ubiquidade definidos inicialmente em *UbiModel* podem ser detalhados através de diagramas de atividades e casos de uso descritos em *UCModel*.

Ao detalhar os requisitos de ubiquidade, entende-se que a especificação dos requisitos torna-se muito mais completa do que se fosse descrita pela aplicação de apenas uma das abordagens. O objetivo é determinar um contexto mais amplo a partir do entendimento das necessidades e demandas até o mapeamento dessas

informações para o conjunto de comportamentos esperados do sistema para realização desses requisitos. A partir da lista de requisitos sob o ponto de vista do usuário e do detalhamento desses requisitos sob o ponto de vista do sistema, é necessário realizar a integração desses requisitos. Com isso, a partir de uma demanda inicial, é possível obter os elementos que descrevem em detalhes o comportamento do sistema para realizá-la. No sentido oposto, é possível obter as demandas e restrições que deram origem a determinados comportamentos do sistema. Para que seja possível rastrear os elementos como descrito anteriormente, é preciso estabelecer os pontos de integração entre as abordagens e como seus elementos se relacionam. A Figura 4-1 ilustra o contexto descrito anteriormente.

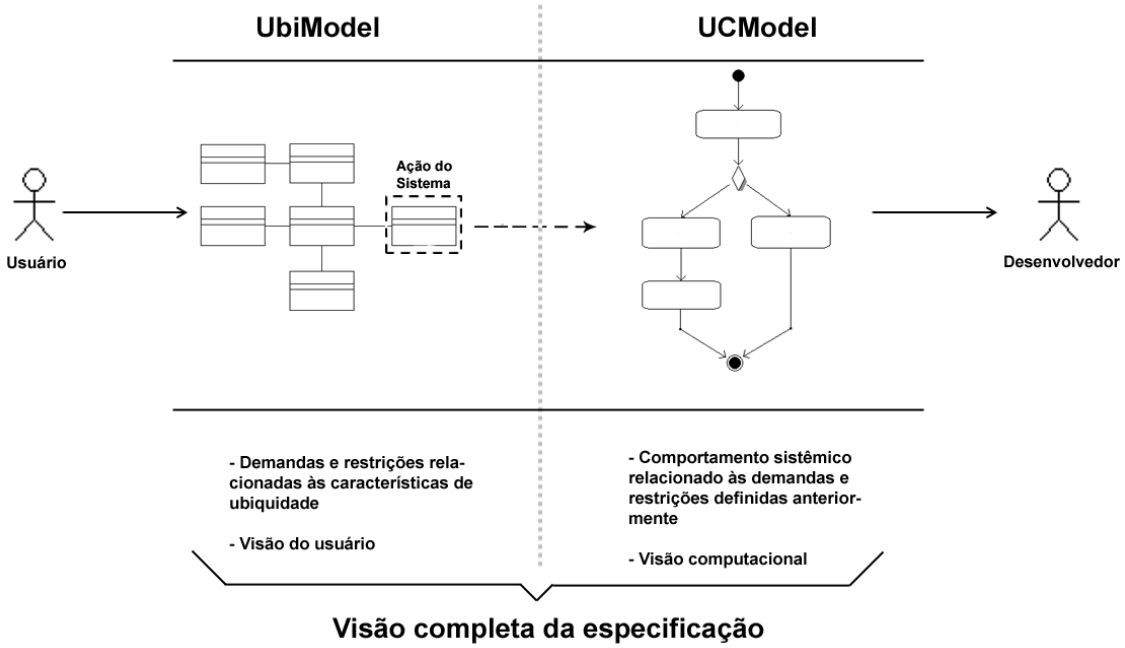


Figura 4-1–Contexto de atuação da abordagem proposta

A Seção 4.3 descreve como é feita a integração dos elementos definidos em *UbiModel* com os elementos de *UCModel*.

4.3 Mapeamento dos Requisitos

Como primeira proposta de solução, considerando-se a existência de um conjunto de elementos em *UbiModel* que referenciam comportamentos esperados do sistema, o objetivo é mapear esses comportamentos nos principais elementos de *UCModel*, que são os casos de uso e os diagramas de atividades que descrevem detalhadamente esses comportamentos. Quando a realização de um requisito é dada por elementos específicos que fazem parte de um caso de uso/diagrama de

atividades, como fluxo alternativo, passo de caso de uso, dentre outros, deve-se indicar primeiramente o caso de uso/diagrama de atividades ao qual esses elementos pertencem e depois descrever qual ou quais elementos do caso de uso realizam o requisito.

É importante destacar que as indicações de quais elementos de *UCModel* realizam os requisitos descritos em *UbiModel* ficam a cargo do engenheiro de software. Não existem regras de mapeamento e multiplicidade entre os requisitos do *UbiModel* e os elementos de *UCModel*, pois essas questões dependem muito do estilo do especificador. Por exemplo, o detalhamento de um requisito definido em *UbiModel* pode ser especificado em *UCModel* como um fluxo alternativo de um caso de uso ou como um caso de uso inteiro (Figura 4-2). Outro requisito pode ser desmembrado em *UCModel* como uma ou mais regras de negócio em um caso de uso.

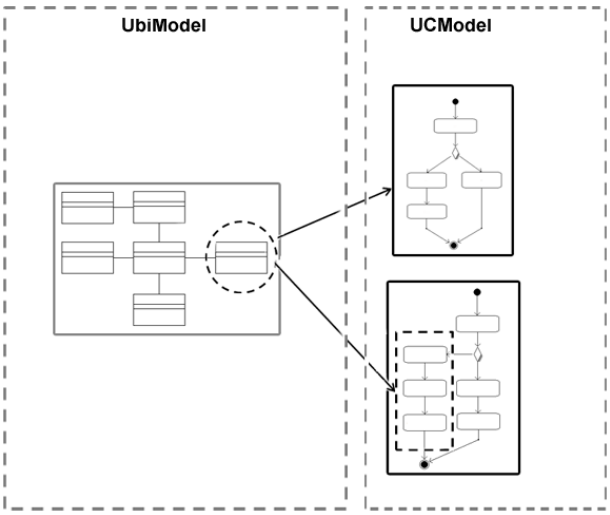


Figura 4-2–Item de especificação dependente do estilo do especificador

Foi definido que o elemento de ligação do *UbiModel* com os elementos de *UCModel* seria a ação do sistema (***systemaction*** – elemento 7 da Figura 4-3), pois esse elemento representa o comportamento esperado do sistema. Do ponto de vista do usuário, as informações capturadas pela ação do sistema no *UbiModel* são suficientes, pois representam as demandas e necessidades do sistema. Do ponto de vista do sistema, a simples descrição de uma ação do sistema pode ocultar uma complexidade que precisa ser descrita em detalhes para que o requisito seja tratável em fases posteriores no processo de desenvolvimento do software. Dessa forma, é importante indicar que elementos descrevem em detalhes, ou seja, como as ações do sistema devem ocorrer.

É importante destacar que o elemento ***systemaction*** está presente em todas as características de ubiquidade consideradas pelo modelo, ou seja, em todas as características de ubiquidade, em algum momento, os requisitos definidos fazem referência a algum comportamento do sistema que precisa ser detalhado de forma adequada. A seguir é apresentada como ***systemaction*** está relacionada com as características de ubiquidade:

- Na característica Sensibilidade ao Contexto, a ação do sistema representa o que o sistema deve fazer ao perceber determinados cenários de utilização. No modelo, esse relacionamento é dado pela associação de ***systemaction*** com o elemento ***behavior*** (elemento 1 da Figura 4-3).
- Na característica Captura de Experiências, a ação do sistema representa o que o sistema deve fazer ao identificar padrões de interação. No modelo, esse relacionamento é dado pela associação de ***systemaction*** com ***interactionpattern*** (elemento 2 da Figura 4-3).
- Na característica Comportamento Adaptável, as adaptações são realizadas pelas ações do sistema. No modelo, esse relacionamento é dado pela associação de ***systemaction*** com o elemento ***systemadaptation*** (elemento 3 da Figura 4-3).
- Na característica Onipresença de Serviços, a ação do sistema representa os serviços disponibilizados a outras aplicações. No modelo, esse relacionamento é dado pela especialização do elemento ***serviceprovided*** (elemento 4 da Figura 4-3) com o elemento ***systemaction***.
- Na característica Heterogeneidade de Dispositivos, as adaptações do sistema para a migração são realizadas pelas ações do sistema. No modelo, esse relacionamento é dado pela associação de ***systemaction*** com o elemento ***requirement*** (elemento 5 da Figura 4-3).
- Na característica Interoperabilidade Espontânea, a ação do sistema representa que serviços externos o sistema pode utilizar. No modelo, esse relacionamento é dado pela associação de ***systemaction*** com o elemento ***interoperability*** (elemento 6 da Figura 4-3).

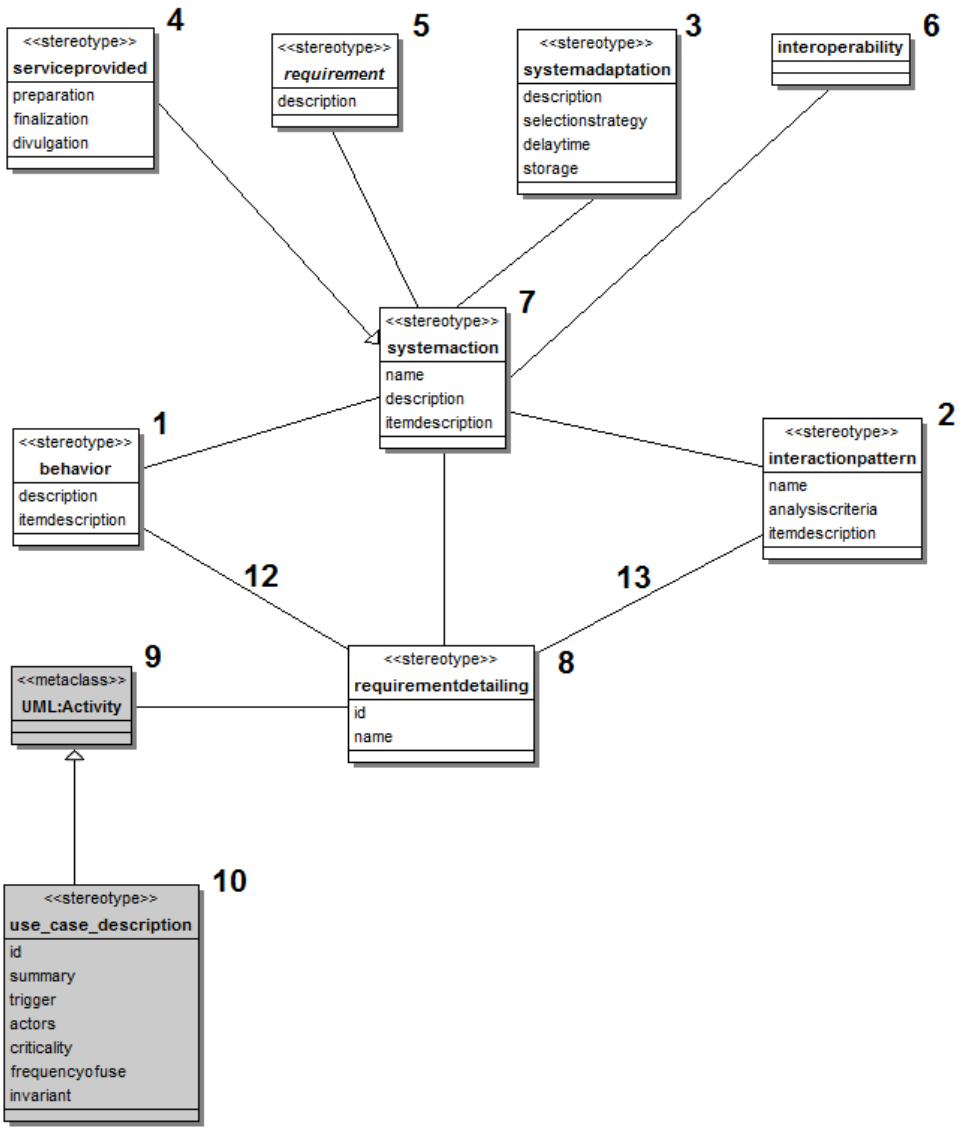


Figura 4-3–Elementos envolvidos na integração dos modelos

Nas características Sensibilidade ao Contexto e Captura de Experiências, o comportamento esperado do sistema é determinado por um conjunto de *systemactions*. Nesses casos, o detalhamento do requisito pode ser feito também a partir dos elementos que agregam as ações do sistema (relacionamentos 12 e 13 da Figura 4-3): ***behavior*** (Sensibilidade ao Contexto – elemento 1 da Figura 4-3) e ***interactionpattern*** (Captura de Experiências – elemento 2 da Figura 4-3), ficando a cargo do especificador a decisão em especificar um elemento de detalhamento do requisito para cada ação do sistema que compõe o comportamento ou apenas pelos elementos agregadores das ações do sistema. A Figura 4-3 mostra o relacionamento do elemento ***systemaction*** com os elementos das características de ubiquidade, além do elemento ***requirementdetailing*** (elemento 8 da Figura 4-3) que representa os

casos de uso e diagramas de atividade de *UCModel*. Em *UCModel*, os elementos responsáveis pela integração dos modelos são UML:Activity (elemento 9 da Figura 4-3) e ***use_case_description*** (elemento 10 da Figura 4-3). O primeiro pertence ao metamodelo da UML para diagramas de atividade e representa uma atividade, o segundo pertence ao metamodelo *UCModel* para diagramas de casos de uso e representa um caso de uso e seus elementos básicos. Dessa forma, os requisitos especificados em *UbiModel* podem referenciar diretamente casos de uso/atividades definidas em *UCModel*. Caso o elemento em *UCModel* que concretize o requisito definido em *UbiModel* seja um elemento específico de um caso de uso/atividade, essa informação deve ser descrita na própria ação do sistema a ser detalhada.

4.4 Conclusão

Este capítulo apresentou a integração proposta para os modelos *UbiModel* e *UCModel*. Como os requisitos definidos em *UbiModel* representam as demandas e necessidades sob o ponto de vista do usuário e tendo em vista a necessidade de se ter os requisitos descritos de forma mais detalhada, foi proposta uma integração que tenta definir um contexto mais amplo para a especificação dos requisitos com a finalidade de torná-la mais completa, de maneira que esta possa ser utilizada como base na execução de atividades em fases posteriores no processo de desenvolvimento do sistema. Assim, a partir das descrições iniciais dos requisitos, é possível chegar as descrições do comportamento do sistema que realizam os mesmos. Além disso, é possível identificar as demandas que justificam determinado comportamento do sistema.

A alternativa escolhida para a descrição dos comportamentos do sistema foram os casos de uso/diagramas de atividade, através de *UCModel*, por permitirem a descrição dos requisitos em um nível adequado às necessidades apresentadas.

Foram determinados os elementos das abordagens envolvidas que seriam responsáveis pela integração dos modelos. Em *UbiModel*, o elemento responsável pela integração é o ***systemaction***, por ser exatamente o elemento do modelo que representa as ações do sistema, ou seja, descreve o que ele deve fazer de acordo com as características de ubiquidade. Os elementos ***UML:Activity*** e ***Use_case_description*** representam, respectivamente, as atividades e casos de uso, de acordo com o *UCModel*. Esses elementos são responsáveis pela integração de *UCModel* com *UbiModel*.

No próximo capítulo será apresentada a infraestrutura computacional elaborada com o objetivo de apoiar a aplicação da abordagem de especificação proposta neste trabalho.

5 Infraestrutura de Apoio à Especificação de Requisitos Funcionais de Ubiquidade

Este capítulo apresenta a infraestrutura computacional para apoiar a abordagem de especificação de requisitos de ubiquidade proposta nesta pesquisa e detalha as três ferramentas desenvolvidas no escopo deste trabalho que compõem essa infraestrutura.

5.1 Introdução

Buscando apoiar a abordagem de especificação de requisitos de ubiquidade apresentada nos capítulos 3 e 4, uma infraestrutura computacional composta por um conjunto de ferramentas foi especificada e implementada. As principais funcionalidades implementadas na infraestrutura são:

1. Caracterização de projetos de software ubíquos de acordo com um modelo de características e fatores de ubiquidade;
2. Criação das especificações dos requisitos de acordo com o metamodelo *UbiModel*;
3. Associação dos requisitos definidos com elementos de *UCModel*;
4. Verificação automática das restrições previstas no metamodelo *UbiModel*, ou seja, verificação sintática da especificação do requisito;
5. Geração da especificação de requisitos em formato texto (padrão RTF – *Rich Text Format*).

As ferramentas que compõem a infraestrutura são:

UbiProject: tem o objetivo de apoiar a caracterização do projeto de software ubíquo, considerando-se que nem todos os projetos apresentam todas as características de ubiquidade. Utilizar todo o corpo de conhecimento estruturado para apoiar o desenvolvimento de projetos de software ubíquo sem levar em consideração as restrições de cada projeto, pode reduzir a efetividade do uso deste conhecimento. Assim, é importante caracterizar o projeto frente às características de ubiquidade para especializar o apoio à especificação dos requisitos de acordo com as necessidades do projeto.

UbiSpecification: tem o objetivo de apoiar a especificação de requisitos de ubiquidade tendo como base o metamodelo *UbiModel* que funciona como um roteiro de especificação ao organizar os elementos e relações importantes na descrição dos requisitos de ubiquidade. Como resultado, é produzido um modelo de especificação de requisitos através da instanciação do metamodelo *UbiModel*. Essa ferramenta permite, ainda, a associação dos requisitos definidos através de *UCModel*, metamodelo de especificação integrado a *UbiModel*, para permitir um detalhamento dos requisitos por meio de casos de uso/diagramas de atividade.

UbiDocument: tem o objetivo de percorrer os elementos do modelo produzido pela ferramenta *UbiSpecification* e gerar um documento estruturado no formato RTF com a especificação de requisitos de ubiquidade.

A Figura 5-1 apresenta as etapas de utilização da infraestrutura de apoio previstas na abordagem proposta nessa dissertação, descritas a seguir:

Etapa 1: nesta etapa, o especificador usa a ferramenta *UbiProject* para caracterizar o projeto de software através da seleção das características e fatores de ubiquidade relacionados ao projeto. O *UbiModel* é especializado a partir dessa caracterização;

Etapa 2: nesta etapa, o especificador usa a ferramenta *UbiSpecification* para descrever os requisitos de ubiquidade. A infraestrutura também apoia a verificação sintática da especificação segundo as restrições do *UbiModel*.

Etapa 3: nesta etapa, o especificador usa a ferramenta *UbiDocument* para gerar parte do documento de especificação relacionada aos requisitos de ubiquidade.

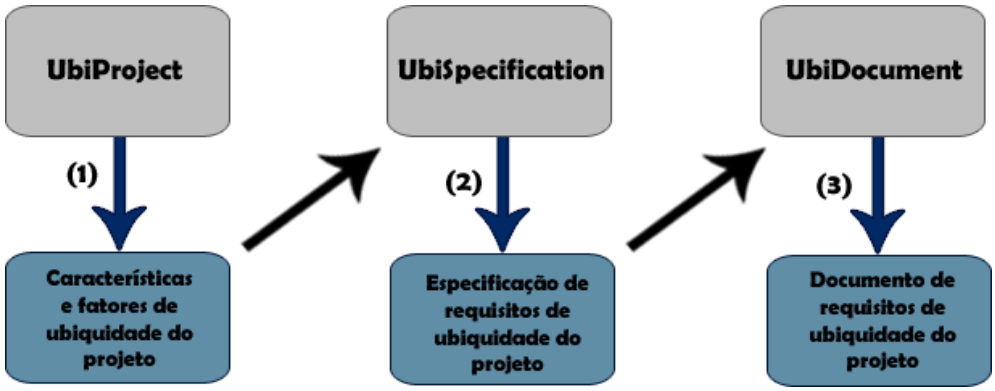


Figura 5-1- Etapas de utilização da infraestrutura

A Seção 5.2 apresenta detalhes sobre como foi realizada a implementação da infraestrutura aqui proposta.

5.2 Implementação da Infraestrutura Computacional

A proposta para especificação de requisitos de ubiquidade, apresentada neste trabalho, está fortemente baseada no uso de modelos UML estereotipados. Por esse motivo, qualquer infraestrutura computacional construída para apoiar a criação de modelos de especificação baseados no *UbiModel* tem que ter como premissa a possibilidade de manipular modelos UML. A fim de tornar mais simples a tarefa de implementação da infraestrutura proposta, foram avaliadas ferramentas UML que oferecessem a possibilidade de extensão, a fim de que as funcionalidades específicas aqui definidas pudessem ser implementadas. Por fim, foi selecionada a ferramenta BOUML (PAGÉS, 2011) como ferramenta UML devido às suas características e recursos disponíveis. São eles:

- Executável em várias plataformas (Windows©, MacOS X©, distribuições Linux);
- Implementa a especificação da UML 2;
- Trabalha com perfis UML, e;
- É extensível através de plug-ins escritos em C++ ou Java.

A infraestrutura computacional desenvolvida foi implementada na linguagem Java por meio de plug-ins para a ferramenta BOUML. A arquitetura oferecida pela BOUML permite que seus plug-ins leiam, criem, alterem ou excluam modelos ou elementos de modelos UML programaticamente. Dessa forma, as ferramentas desenvolvidas são capazes de manipular modelos UML, criando ou editando tais modelos segundo o metamodelo *UbiModel* e suas restrições.

A Seção 5.3 apresenta a estratégia de especificação adotada através da descrição dos elementos relacionados que formam a base para o desenvolvimento das funcionalidades que compõem a infraestrutura.

5.3 Estratégia de Especificação

A partir do *UbiModel*, apresentado no capítulo 3, foi projetada a infraestrutura computacional para apoiar a especificação de requisitos ubíquos. Um conjunto de modelos foi desenvolvido, utilizando a ferramenta BOUML, a fim de apoiar a infraestrutura em aspectos relacionados à estratégia de especificação adotada.

A estratégia de especificação dos requisitos de ubiquidade foi desenvolvida a partir da ideia de que as características de ubiquidade possuem um conjunto de fatores de ubiquidade que representam como essas características podem estar contempladas em projetos de softwares (SPÍNOLA, 2010). Esses fatores, por sua vez,

dão origem a um conjunto de diretivas, que são indicações sobre quais informações devem ser descritas ao especificar os requisitos de ubiquidade. É importante ressaltar que as diretivas são definidas a partir de termos específicos do domínio da computação ubíqua que precisam ser compreendidos para que os requisitos possam ser especificados corretamente. Além disso, a especificação de uma diretiva pode depender da especificação prévia de outras diretivas, o que leva ao conceito de pré-condição entre as diretivas. Como o escopo dos projetos de software ubíquos não compreendem necessariamente todas as características e fatores de ubiquidade é importante caracterizá-los frente a essas características e fatores para que seja possível realizar um corte nesse conjunto de informações para que sejam tratadas apenas as diretivas que fazem parte do escopo do projeto de acordo com suas características.

A Figura 5-2 apresenta o modelo que representa os elementos da estratégia de especificação na qual a infraestrutura computacional está baseada.

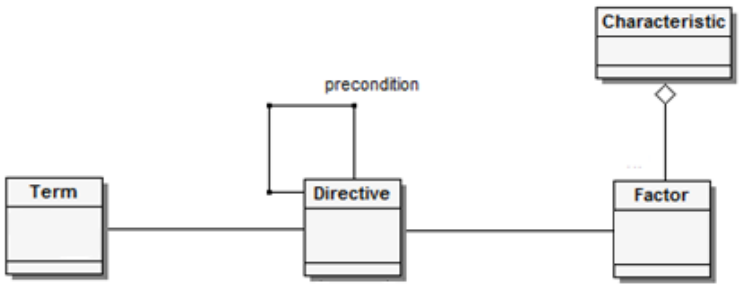


Figura 5-2- Modelo da estratégia de especificação

A partir do modelo da Figura 5-2 foi criado o modelo de características de ubiquidade, o modelo de fatores associados a essas características e o modelo de diretivas associadas a esses fatores. A Seção 5.4 apresenta as diretivas de apoio à definição de requisitos de ubiquidade através de Ubimodel e suas respectivas definições. A Seção 5.5 apresenta os modelos utilizados pela infraestrutura desenvolvida para apoiar a especificação de requisitos de ubiquidade.

5.4 Diretivas de *UbiModel*

A seguir, são apresentadas as diretivas consideradas por *UbiModel*:

Definir Usuários

Objetivo: definir os usuários do sistema informando um nome que os identifique e uma descrição dos seus papéis no uso do sistema.

Definir Localizações

Objetivo: definir os locais do ambiente controlados pelo sistema informando um nome que os identifique e uma descrição de suas características.

Definir Identificação

Objetivo: definir como os usuários são identificados pelo sistema. Para cada forma de identificação deve-se informar o conjunto de usuários e suas respectivas localizações onde aquela forma de identificação é válida.

Definir Atividades

Objetivo: definir as atividades controladas pelo sistema informando um nome que as identifique e uma descrição do seu significado.

Definir Ações do Sistema

Objetivo: definir as ações do sistema relacionadas à ubiquidade, ou seja, o que o sistema deve realizar. Devem ser informados os itens de especificação que descrevem em detalhes cada ação do sistema.

Definir Fontes de Dados

Objetivo: definir as fontes a partir das quais as informações de contexto podem ser obtidas informando um nome que as identifique e se as mesmas são do tipo software ou dispositivo. Caso seja um dispositivo, informar se a mesma refere-se a um sensor. Por fim, informar uma descrição das principais características da fonte.

Definir Informações de Contexto

Objetivo: definir as informações relacionadas aos elementos do ambiente relevantes para o sistema. As informações de contexto podem ser do tipo física quando estão relacionadas ao ambiente (temperatura, iluminação, ruído, etc.), podem ser informações de sistema quando estão relacionadas às restrições de infraestrutura (uso de memória, cpu, tamanho de tela, etc.), podem ser do tipo infraestrutura quando estão relacionadas ao correto funcionamento da infraestrutura (estado de funcionamento dos sensores, etc.) e, finalmente, podem ser do tipo usuário quando estão relacionadas aos usuários do sistema (temperatura corporal, pressão sanguínea, batimento cardíaco, etc.). Deve-se informar a fonte de dados ou a transformação que gera a informação de contexto. É importante também determinar a

validade das informações em relação ao tempo, além do critério e do meio de armazenamento das informações.

Definir Serviços Fornecidos

Objetivo: definir as ações do sistema que devem ser disponibilizadas como serviços a outras aplicações. Além de informações como nome e descrição dos serviços, é importante descrever o processo de inicialização dos mesmos, o que é feito antes dos serviços iniciarem, o que ocorre em suas finalizações, quais as condições mínimas para que estejam disponíveis e como são divulgados. Informações básicas sobre os serviços como protocolos e formato de dados também devem ser descritos.

Definir Comportamentos

Objetivo: definir o que o sistema deve fazer ao perceber determinados contextos. Um contexto é um cenário de utilização do sistema. Em outras palavras, na utilização do sistema, sempre haverá algum usuário participando de alguma atividade em algum lugar, sob determinadas condições do ambiente. Quando relevantes, deverão ser especificados para cada contexto quais são os usuários, atividades, localizações e condições que o definem. A partir desses contextos o sistema pode se comportar de maneira distinta para cada um ou para cada conjunto de contextos.

Definir Padrões de Interação

Objetivo: definir as sequências de interações que ocorrem de maneira recorrente, segundo critérios, que ao serem identificadas fazem com que o sistema realize ações em benefício do usuário. Interações são ações que o usuário realiza no ambiente controlado pelo sistema (acender a luz, ligar a TV, ligar o computador, etc.) e têm um conjunto de informações de contexto relacionadas. Além disso, elas ocorrem em determinados contextos de utilização do sistema. Na definição dos padrões é importante informar a ordem em que as interações devem ocorrer e os critérios utilizados para determinar a validade de um padrão (tempo decorrido entre as interações, tempo decorrido entre as sequências de interações, etc.). Na definição do que o sistema deve fazer na ocorrência de padrões é importante informar a ordem em que as ações devem ser realizadas.

Definir Adaptações do Sistema

Objetivo: definir as adaptações que o sistema deve realizar caso alguma preferência seja violada. Uma preferência é uma condição do sistema, definida em

função de uma informação de contexto, que deve ser obedecida para determinados contextos. As adaptações são compostas por alternativas de adaptação que o sistema deve selecionar considerando seus impactos. É importante informar o tempo que o sistema deve aguardar para a efetivação de uma adaptação, já que o contexto no qual a preferência é violada pode se desfazer rapidamente, não havendo a necessidade da adaptação. Deve-se informar, também, a estratégia de seleção utilizada pelo sistema para selecionar uma alternativa de adaptação e como as informações sobre as adaptações realizadas devem ser armazenadas.

Definir Serviços Requeridos

Objetivo: definir os serviços externos que o sistema deve buscar e utilizar para atender determinadas necessidades que ele não é capaz de atender, pela não realização das ações ou pela inexistência no ambiente de serviços que realizem as ações de maneira a trazer algum ganho em relação a forma como elas são realizadas no momento. Ambos os cenários são descritos por requisitos mínimos que os serviços devem apresentar para que sejam efetivamente utilizados. Além de informações de nome e descrição dos serviços, é importante descrever como as informações relevantes durante a utilização dos serviços serão armazenadas e as estratégias utilizadas pelo sistema para buscar e selecionar os serviços. Devem ser informadas quais ações do sistema os serviços a serem utilizados devem atender. Informações básicas sobre os serviços como protocolos e formato de dados também devem ser descritos.

Definir Migrações

Objetivo: definir as informações necessárias para que a aplicação possa migrar para outros dispositivos diferentes do utilizado originalmente pelo sistema. É importante descrever os requisitos que os dispositivos devem atender e as adaptações que devem ocorrer para que a migração seja possível. Existem 2 tipos de migração. No primeiro o usuário é responsável por iniciar o sistema via dispositivo heterogêneo e, a partir daí, o sistema se adapta de acordo com as características do dispositivo. A esse tipo de migração foi dado o nome de Migração Estática. O segundo tipo, denominado Migração Dinâmica, caracteriza-se por ser iniciado pelo próprio sistema, a partir da ocorrência de algum evento (usuário bloquear o computador, fechar a tampa do *notebook*, etc.). Após o evento ser percebido, o sistema busca e seleciona dispositivos no ambiente de acordo com estratégias e critérios de disponibilidade dos dispositivos que devem estar presentes na especificação. Na Migração Dinâmica, diferentemente da Migração Estática, os requisitos são organizados por grupos de

dispositivos, ou seja, o sistema pode se adaptar de diferentes maneiras para cada grupo. Na Migração Estática o conjunto de requisitos é único, pois existe apenas um conjunto de adaptações do sistema, sendo portanto desnecessária a definição de grupos de dispositivos para esse tipo de migração.

5.5 Modelos de Apoio à Infraestrutura

O modelo de características de ubiquidade (Figura 5-3) foi criado para apoiar a ferramenta *UbiProject* na caracterização de projetos ubíquos. Esse modelo representa a relação entre as características e os fatores de ubiquidade. Uma característica é formada por um conjunto de fatores de ubiquidade e um fator de ubiquidade se relaciona a uma única característica de ubiquidade. Este modelo foi criado a partir do corpo de conhecimento definido em SPÍNOLA (2010). A Figura 5-3 apresenta a parte do modelo referente à característica Sensibilidade ao Contexto. O Apêndice C apresenta as partes do modelo das demais características de ubiquidade abordadas no trabalho.

O modelo é constituído por elementos estereotipados que representam as características de ubiquidade (Tabela 5-1), fatores de ubiquidade (Tabela 5-2) e a relação entre eles (Tabela 5-3).

Tabela 5-1 - Estereótipo usado na definição da característica de ubiquidade

Estereótipo	**Definição**
<<*characteristic*>>	Característica de ubiquidade
Tagged-values	
id	Número único que identifica a característica de ubiquidade
state	Define o estado da característica: valid (v) ou invalid (i)
description	Descrição da característica de ubiquidade

Tabela 5-2 - Estereótipo usado na definição do fator de ubiquidade

Estereótipo	**Definição**
<<*factor*>>	Fator de ubiquidade
Tagged-values	
id	Número único que identifica o fator de ubiquidade
state	Define o estado do fator: valid (v) ou invalid (i)
description	Descrição do fator de ubiquidade

Tabela 5-3 - Estereótipo usado na definição do relacionamento entre características e fatores de ubiquidade

Estereótipo	**Definição**
<<own>>	Relação entre uma característica e um fator de ubiquidade
Tagged-values	
order	Ordem de exibição do fator

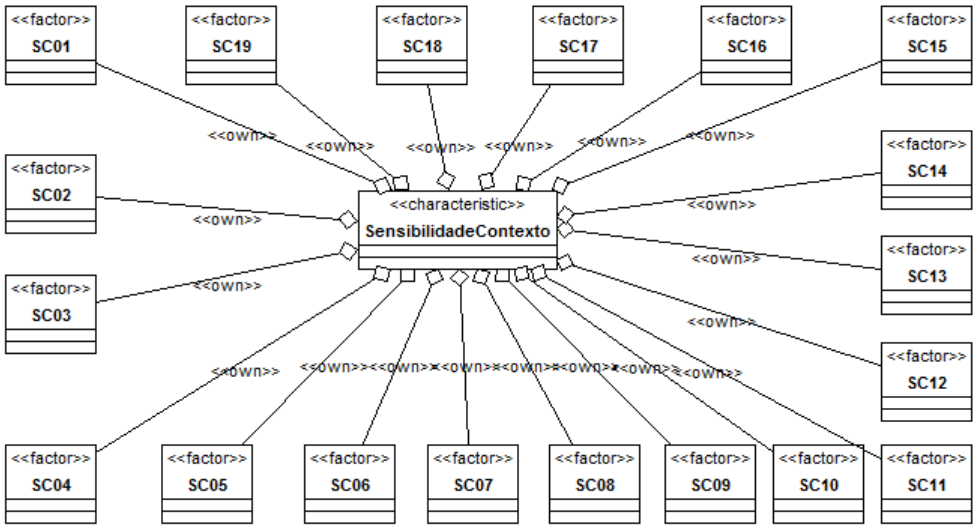


Figura 5-3 - Modelo de Características de Ubiquidade para Sensibilidade ao Contexto

O modelo de fatores (Figura 5-4) foi criado para apoiar a ferramenta *UbiSpecification* na elaboração do roteiro de especificação de requisitos de ubiquidade de acordo com a caracterização do projeto. Ele representa a relação entre os fatores de ubiquidade e as diretivas.

É importante ressaltar que uma diretiva pode ter como origem um ou mais fatores de ubiquidade e um fator de ubiquidade pode originar mais de uma diretiva de apoio a especificação. Para que uma diretiva seja habilitada e faça parte do roteiro de especificação de requisitos, basta que apenas um fator de ubiquidade relacionado a ela seja válido. A Figura 5-4 apresenta a parte do modelo referente a característica Sensibilidade ao Contexto. O Apêndice D apresenta as partes do modelo das demais características de ubiquidade abordadas no trabalho.

O modelo de fatores é constituído por elementos estereotipados que representam as diretivas (Tabela 5-4), os fatores de ubiquidade (Tabela 5-2) e a associação entre eles (Tabela 5-5).

Tabela 5-4 - Estereótipo usado na definição da diretiva do roteiro de especificação

Estereótipo	Definição
<<directive>>	Diretiva de apoio a especificação de requisitos de ubiquidade
Tagged-values	
order	Ordem de exibição da diretiva
state	Define o estado da diretiva: valid (v) ou invalid (i)
description	Descrição da diretiva
filled	Indica se um requisito para a diretiva já foi especificado: yes (y) ou no (n).

Tabela 5-5 - Estereótipo usado na definição do relacionamento entre fatores de ubiquidade e diretivas

Estereótipo	Definição
<<support>>	Relação entre um fator de ubiquidade e uma diretiva

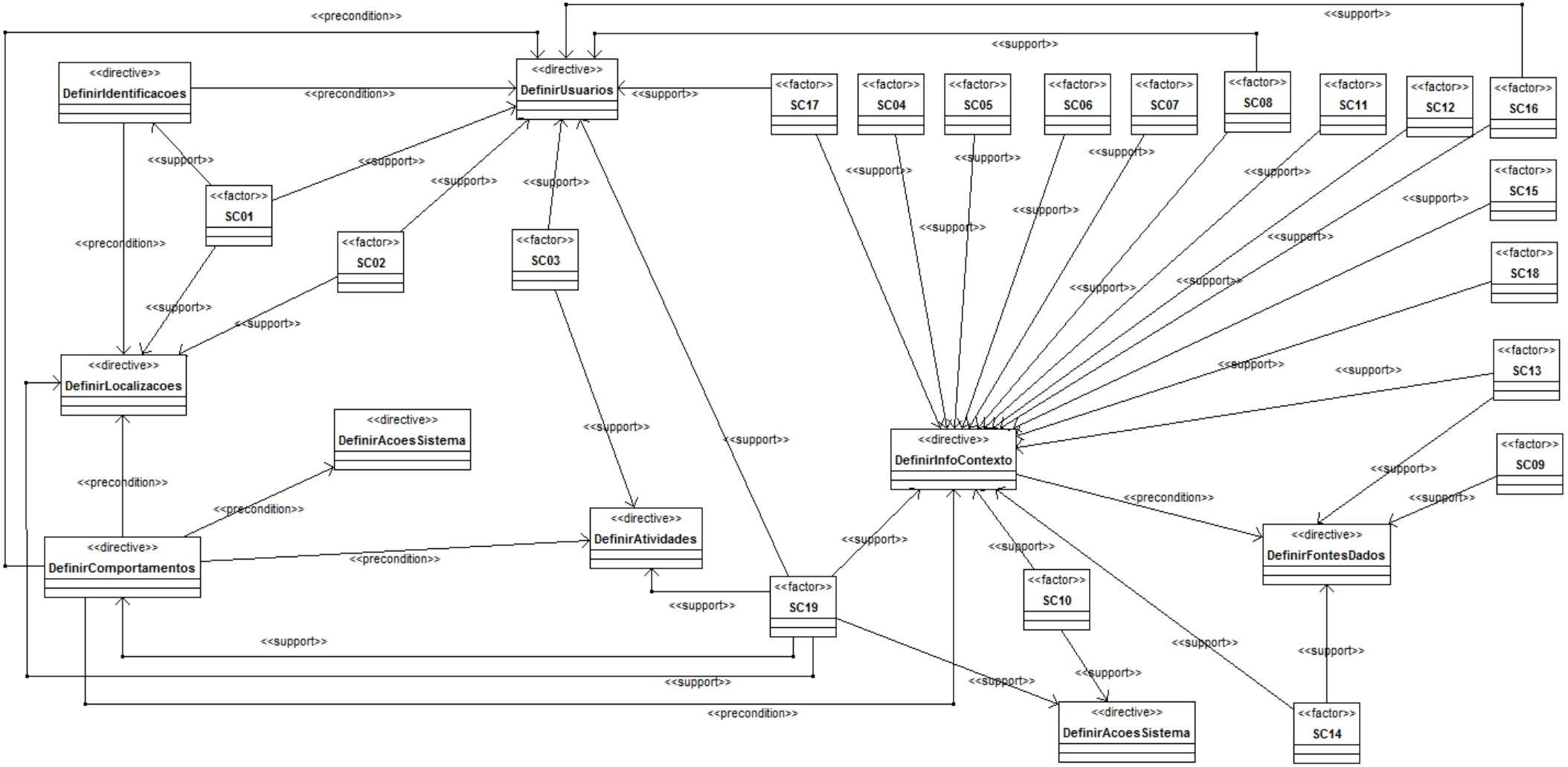


Figura 5-4- Modelo de fatores para Sensibilidade ao Contexto

O modelo de pré-condições (Figura 5-5) foi criado para apoiar a ferramenta *UbiSpecification* na elaboração da estratégia de ordenação do roteiro de especificação a ser seguido, tendo em vista as dependências entre as diretivas.

O modelo de pré-condições representa a relação de dependência entre as diretivas. Uma diretiva pode ser pré-condição para zero ou mais diretivas e ter zero ou mais diretivas como pré-condição. O modelo é constituído por elementos estereotipados que representam as diretivas (Tabela 5-4) e a associação existente entre elas (Tabela 5-6).

Tabela 5-6 - Estereótipo usado na definição do relacionamento entre as diretivas

Estereótipo	Definição
<<precondition>>	Relação entre as diretivas

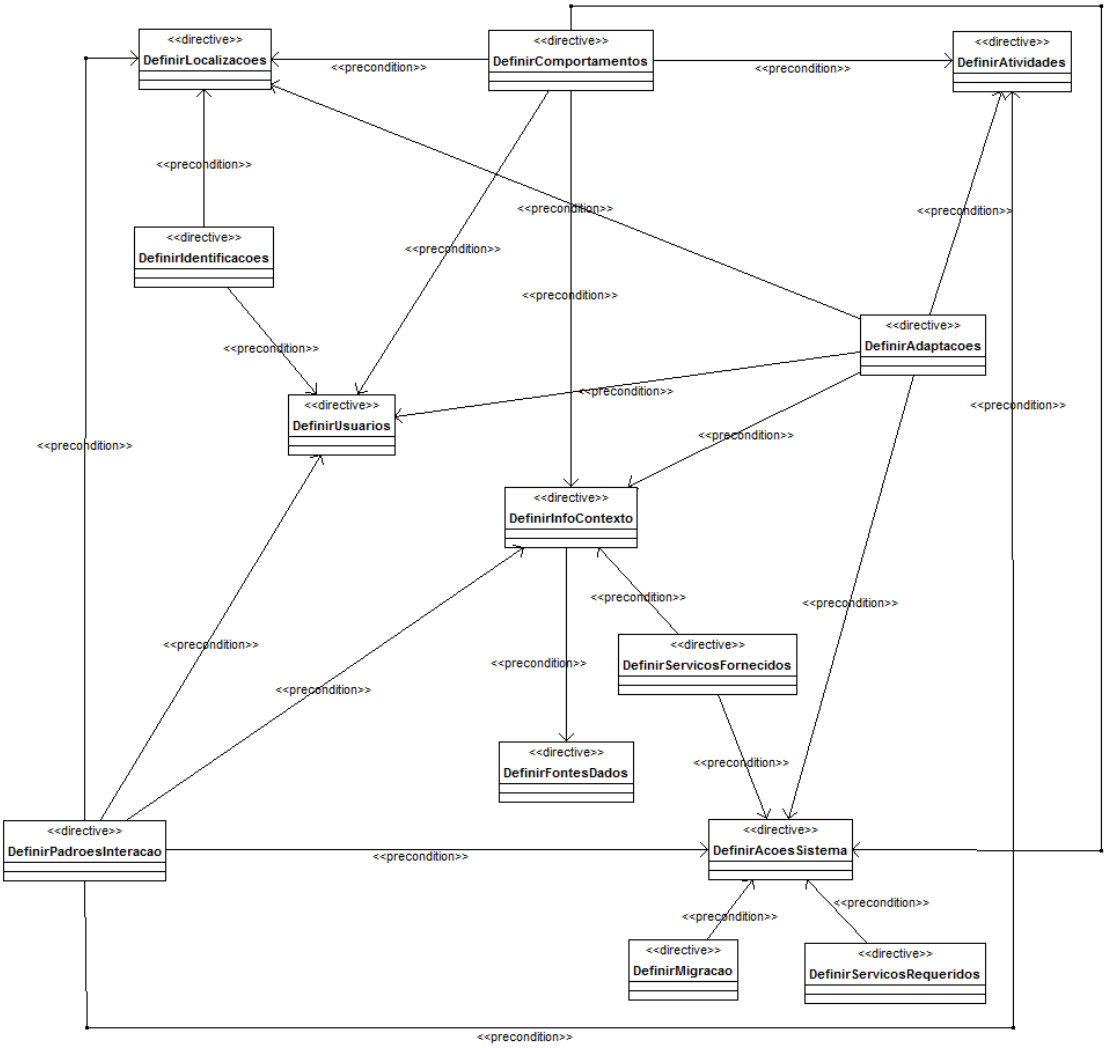


Figura 5-5 – Modelo de pré-condições

O modelo de diretivas (Figura 5-6) foi elaborado para apoiar a construção de um glossário de termos da computação ubíqua relacionado às diretivas. Dessa forma, uma lista de termos com suas respectivas definições é fornecida pela ferramenta para cada diretiva, no momento em que elas são utilizadas na especificação dos requisitos de ubiquidade, para que o foco do especificador não seja desviado com definições de termos não relacionados a diretiva em questão.

O modelo de diretivas representa a relação entre as diretivas e os termos da computação ubíqua. Uma diretiva possui diversos termos da computação ubíqua e um termo pode pertencer a mais de uma diretiva. A Figura 5-6 apresenta a parte do modelo referente à característica Sensibilidade ao Contexto. O Apêndice E apresenta as partes do modelo das demais características de ubiquidade abordadas no trabalho.

O modelo é constituído por elementos estereotipados que representam as diretivas (Tabela 5-4), termos (Tabela 5-7) e a associação entre eles (Tabela 5-8).

Tabela 5-7 - Estereótipo usado na definição do termo da computação ubíqua

Estereótipo	**Definição**
<<term>>	Termo da computação ubíqua
Tagged-values	
name	Nome do termo
definition	Definição do termo

Tabela 5-8 - Estereótipo usado na definição do relacionamento entre diretivas e termos da computação ubíqua

Estereótipo	**Definição**
<<reference>>	Relação entre uma diretiva e um termo da computação ubíqua

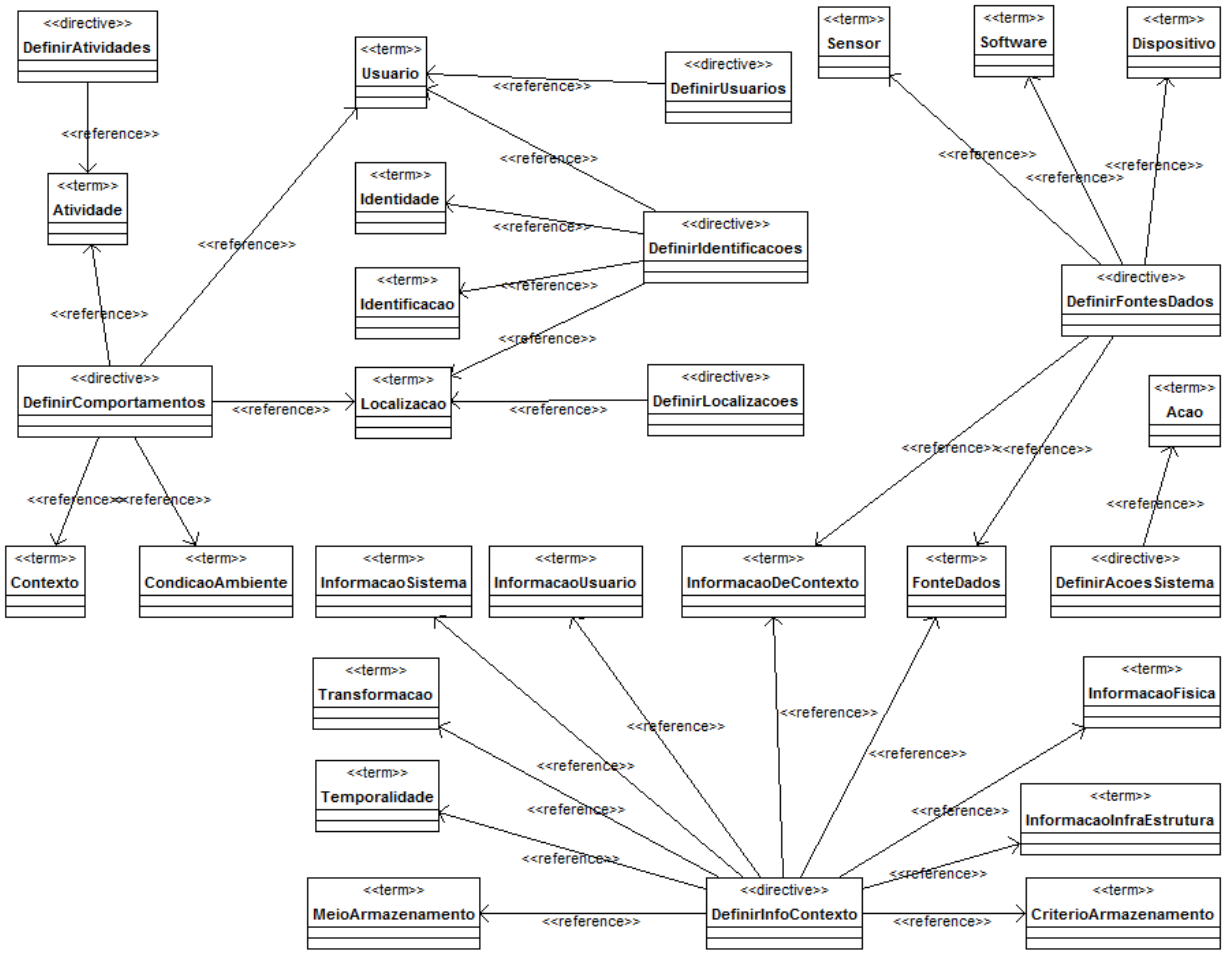


Figura 5-6 – Parte do modelo de diretivas para Sensibilidade ao Contexto

A Seção 5.5 apresenta as ferramentas que compõem a infraestrutura através de um exemplo hipotético do seu uso.

5.6 Ferramentas de Apoio à Infraestrutura Computacional

5.6.1 *UbiProject*

A ferramenta *UbiProject* foi desenvolvida para apoiar a atividade de caracterização de projetos ubíquos. *Ao caracterizar* um projeto, o especificador indica quais características e fatores de ubiquidade estão presentes no projeto. As características e os fatores de ubiquidade selecionados na etapa de caracterização serão utilizados para configurar as informações que devem ser especificadas, ou seja, quais diretivas estarão disponíveis no roteiro de especificação que será construído posteriormente. A Figura 5-7 apresenta a tela de caracterização de projeto. As características de ubiquidade (obtidas a partir do modelo de características - Figura 5-3) são apresentadas como abas e os fatores associados a cada característica (obtidos

a partir do modelo de fatores - Figura 5-4) são apresentados como itens selecionáveis em uma lista. O analista tem a liberdade de selecionar as características e/ou fatores de acordo com as particularidades do projeto. A Figura 5-7 ilustra a seleção de apenas um fator de ubiquidade relacionado à característica Sensibilidade ao Contexto.

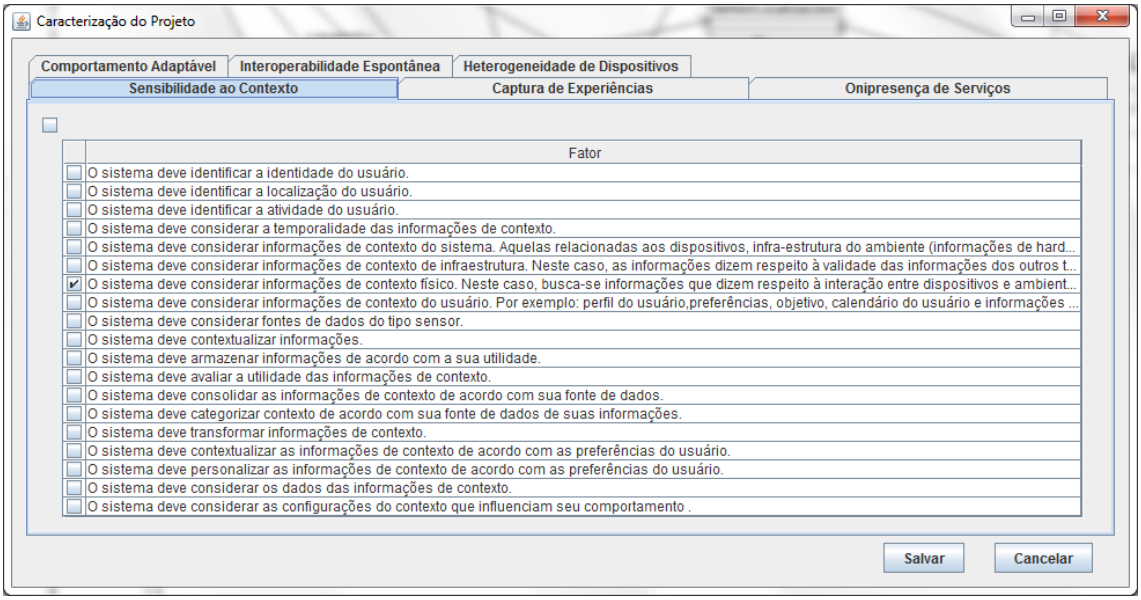


Figura 5-7- Tela de caracterização do projeto

Uma vez que as características/fatores são selecionados pelo especificador, é possível definir as diretivas que estarão habilitadas para especificação (obtidas a partir do modelo de fatores - Figura 5-4) e a ordem em que essas diretivas serão tratadas (obtidas a partir do modelo de diretivas - Figura 5-5).

5.6.2 *UbiSpecification*

Quando a etapa de caracterização do projeto é finalizada, os fatores que foram selecionados servirão para habilitar as diretivas relacionadas a esses fatores que serão utilizadas na elaboração do roteiro de especificação dos requisitos de ubiquidade. Uma diretiva possui um conjunto de informações que são usadas para que o especificador compreenda, de forma clara, o que se espera na definição de um requisito. O roteiro de especificação funciona como um orquestrador ao fornecer as diretivas de apoio à especificação dos requisitos compatíveis com a caracterização do projeto em uma ordem que faça sentido de serem especificados. Ao especificar os requisitos de ubiquidade de um projeto, serão apresentadas apenas as diretivas relacionadas aos fatores e características selecionados na etapa de caracterização.

A especificação dos requisitos não pode ser realizada de forma aleatória. Foi observada a existência de dependências entre as diretivas na definição dos requisitos. Só faz sentido especificar o requisito da diretiva X que depende da diretiva Y se o requisito da diretiva Y tiver sido especificado anteriormente, ou seja, algumas diretivas podem ser pré-condições para outras. Por exemplo, não faz sentido especificar o conjunto de informações de contexto sem ter as fontes de dados dessas informações especificadas previamente, pois as informações de contexto são obtidas a partir das fontes de dados. Desta forma, é estabelecida uma ordem onde estas dependências são consideradas.

O metamodelo *UbiModel* foi elaborado com base nos fatores das características de ubiquidade. Como esses fatores utilizam termos específicos do domínio da computação ubíqua, as diretivas herdaram esses termos.

A Figura 5-8 apresenta a tela principal da ferramenta *UbiSpecification*. Nessa tela é possível observar duas diretivas relacionadas ao fator de ubiquidade previamente selecionado na ferramenta *UbiProject* (Figura 5-7). O sinal verde indica que a diretiva não possui pré-condições, estando liberada para a especificação do requisito associado. O sinal vermelho indica que a diretiva depende da definição prévia de outras diretivas para ser liberada. O especificador pode clicar sobre as diretivas com sinal verde a fim de realizar a especificação propriamente dita. Nesse caso, a ferramenta *UbiSpecification* apresentará uma tela projetada para especificação daquela diretiva.

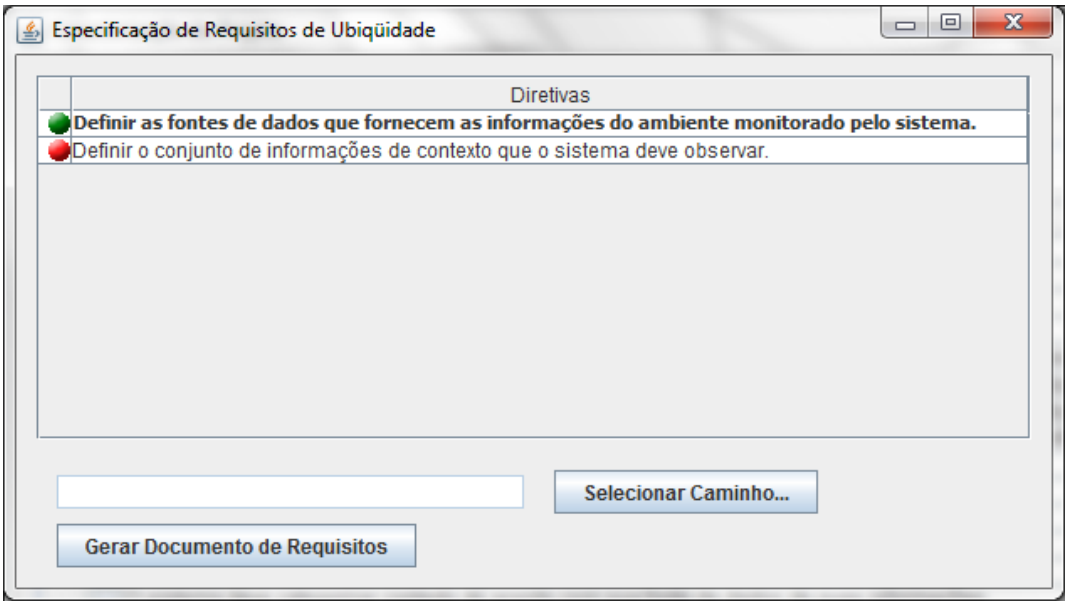


Figura 5-8- Roteiro de especificação

Por exemplo, no momento em que o especificador clicar na primeira diretiva da tela da Figura 5-4, é apresentada a tela da Figura 5-9 para manutenção (inclusão, exclusão e alteração) das Fontes de Dados. Nesse exemplo em particular, a lista de Fontes de Dados está vazia, pois nenhuma Fonte de Dados foi definida até o momento.

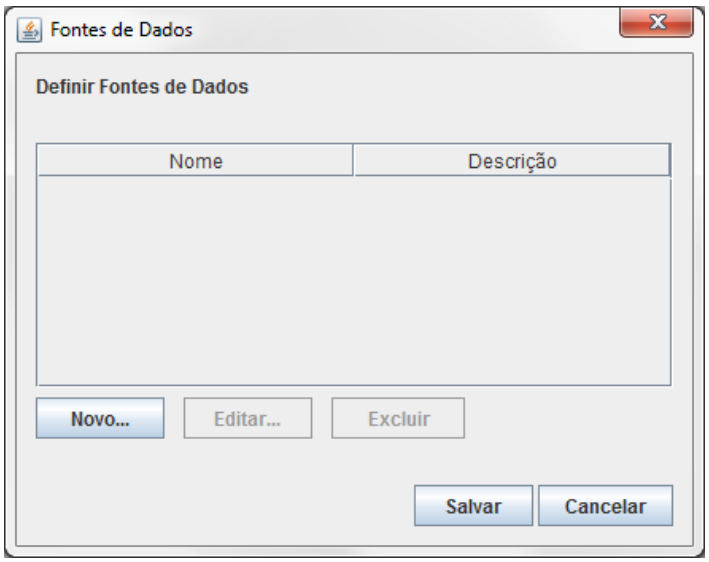


Figura 5-9- Tela de manutenção de Fontes de Dados

Ao clicar no botão "Novo..." a tela da Figura 5-10 é apresentada para que uma nova Fonte de Dados possa ser especificada.

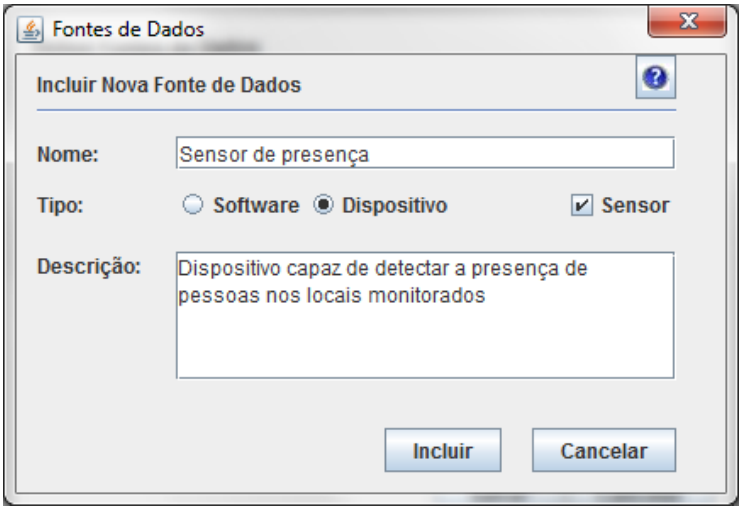


Figura 5-10- Tela de especificação de Fonte de Dados

Ao concluir o preenchimento das informações e clicar no botão "Incluir" é exibida a tela da Figura 5-11, onde a diretiva especificada aparece com um símbolo de verificação, indicando que requisitos para essa diretiva já foram especificados. Repare

que a diretiva que estava bloqueada anteriormente passou para o estado "liberada para especificação", pois a sua pré-condição foi resolvida.

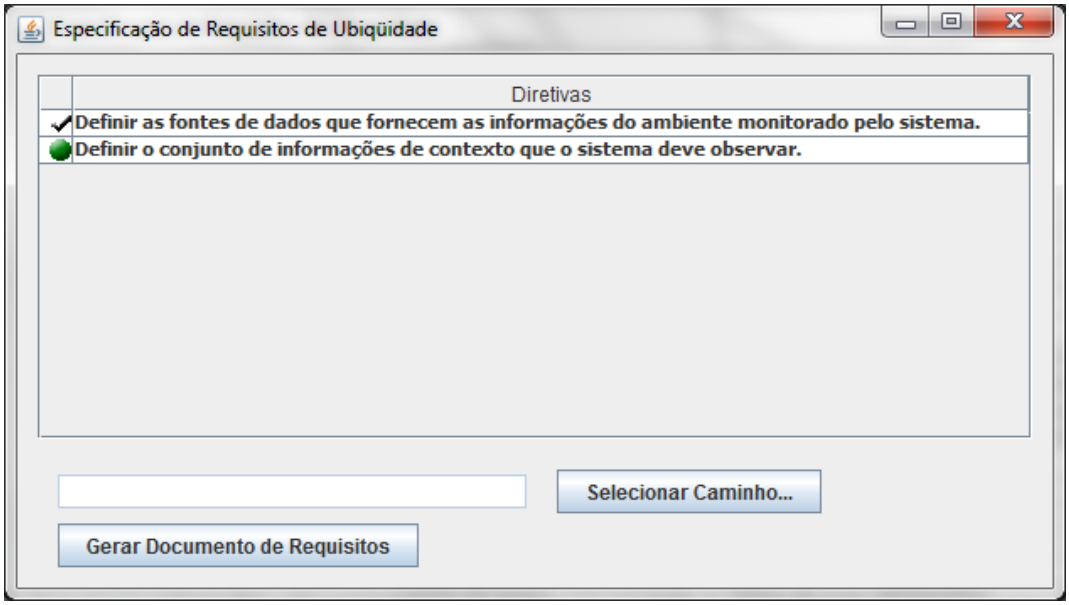


Figura 5-11- Liberação da diretiva bloqueada

Ao clicar na segunda diretiva da tela da Figura 5-11 é apresentada a tela da Figura 5-12 para manutenção (inclusão, exclusão ou alteração) das Informações de Contexto. Novamente uma lista vazia é apresentada inicialmente, pois nenhuma Informação de Contexto foi definida até o momento.

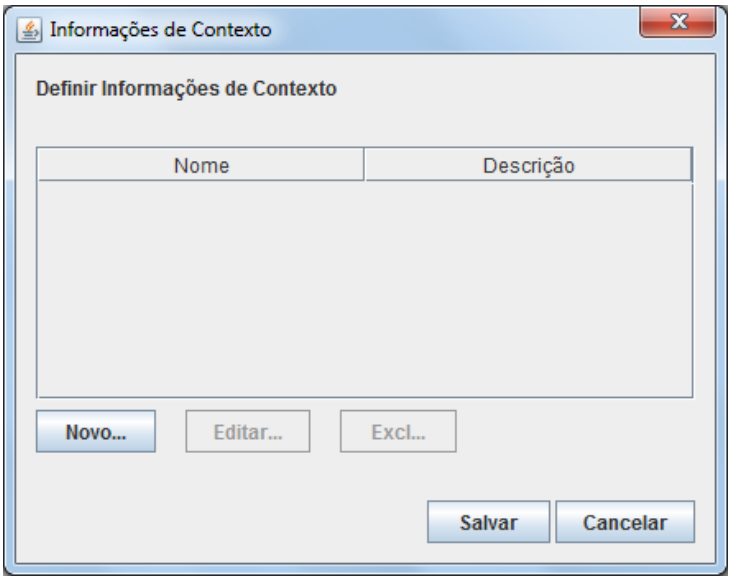


Figura 5-12- Tela de manutenção de Informações de Contexto

Ao clicar no botão "Novo..." a tela da Figura 5-13 é apresentada para que uma nova Informação de Contexto possa ser especificada.

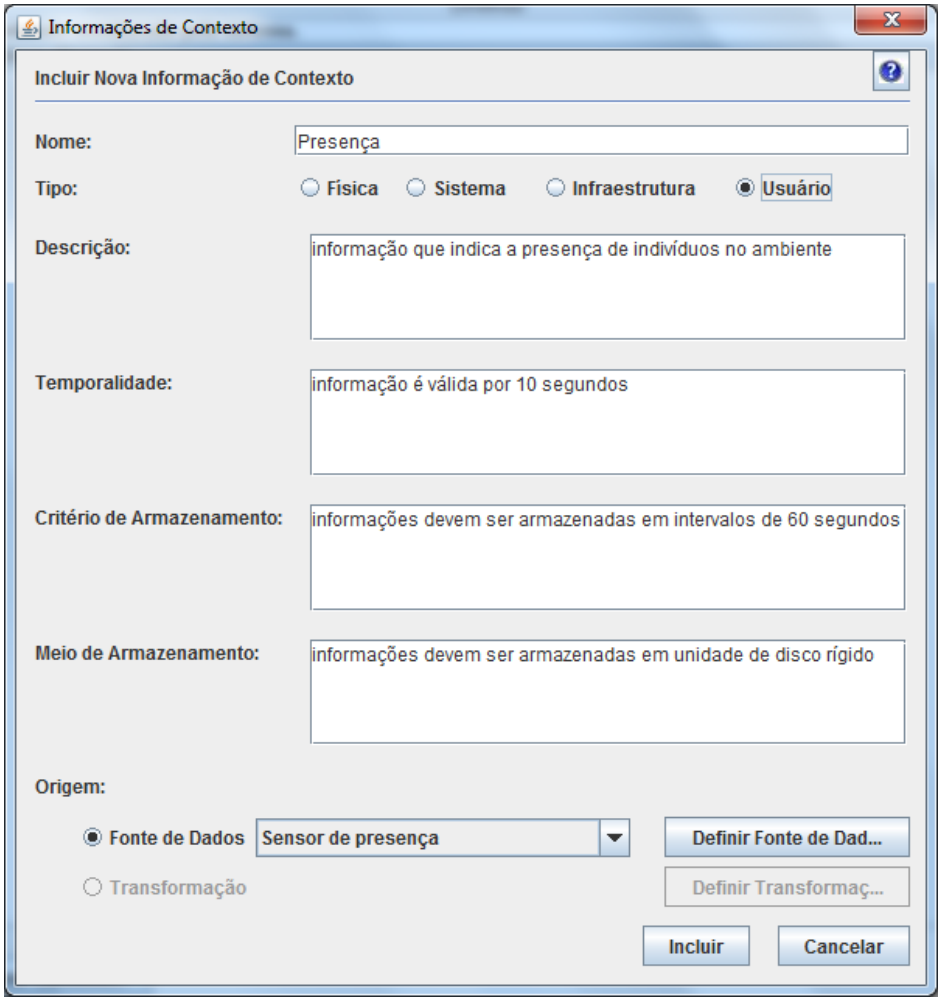


Figura 5-13- Tela de especificação de Informação de Contexto

Ao clicar no botão "Incluir", a segunda diretiva da tela da Figura 5-11 também passará para o estado "especificada". Dessa forma, todas as diretivas definidas para especificação desse projeto em particular estarão especificadas.

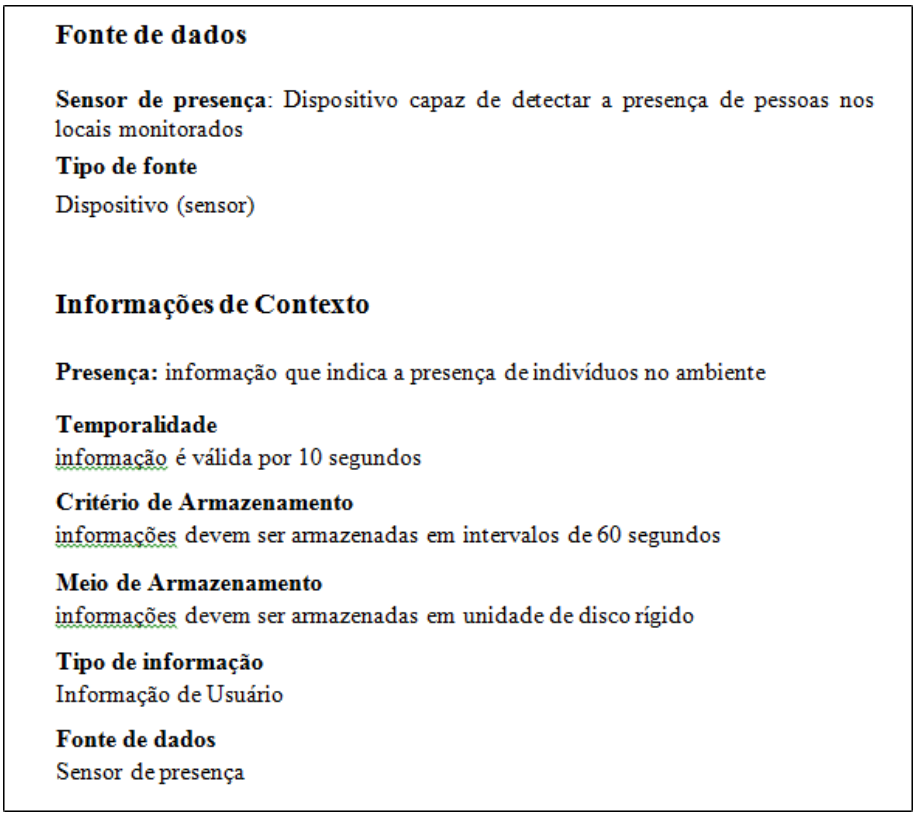
É importante ressaltar que, a partir das informações coletadas pelo especificador, a ferramenta *UbiSpecification* gera um modelo de requisitos aderente ao metamodelo *UbiModel*. A Figura 5-14 apresenta o modelo de especificação gerado pela *UbiSpecification* para o exemplo em questão.

Figura 5-14- Modelo de especificação gerado por *UbiSpecification*

O próximo passo seria a geração do documento de especificação, apresentado na próxima subseção.

5.6.3 *UbiDocument*

A ferramenta *UbiDocument* foi desenvolvida para apoiar a geração do documento de especificação a partir do modelo gerado pela ferramenta *UbiSpecification*. Para isso, basta selecionar o diretório no qual o documento deve ser salvo e clicar no botão "Gerar Documento de Requisitos" (Figura 5-11). A Figura 5-15 apresenta um fragmento da especificação gerada de acordo com o exemplo apresentado anteriormente.



Fonte de dados

Sensor de presença: Dispositivo capaz de detectar a presença de pessoas nos locais monitorados

Tipo de fonte

Dispositivo (sensor)

Informações de Contexto

Presença: informação que indica a presença de indivíduos no ambiente

Temporalidade

informação é válida por 10 segundos

Critério de Armazenamento

informações devem ser armazenadas em intervalos de 60 segundos

Meio de Armazenamento

informações devem ser armazenadas em unidade de disco rígido

Tipo de informação

Informação de Usuário

Fonte de dados

Sensor de presença

Figura 5-15- Fragmento da especificação gerada por *UbiDocument*

5.7 Conclusão

O objetivo principal da infraestrutura computacional organizada no escopo deste trabalho é oferecer apoio à especificação de requisitos de ubiquidade baseados no modelo *UbiModel*. Para tal, foram implementadas três ferramentas chamadas *UbiProject*, *UbiSpecification* e *UbiDocument*. A concepção dessas ferramentas levou em consideração a organização de um ambiente de trabalho no qual o especificador tivesse acesso às várias funcionalidades oferecidas pelas ferramentas da forma mais transparente possível. Desta forma, elas foram implementadas como *plug-ins* da

ferramenta BOUML (PAGÉS, 2011), o que possibilita a execução dessas ferramentas e a manipulação de modelos UML dentro do mesmo ambiente de trabalho.

O uso dessa infraestrutura computacional tem como objetivo:

- Reduzir o esforço necessário para especificação dos requisitos de ubiquidade, minimizando a quantidade de interações necessárias para criação do modelo de especificação.
- Aumentar a qualidade da especificação, já que o especificador é guiado por um roteiro que o direciona para quais informações devem ser fornecidas, minimizando a falta de informações ou a entrada de informações inadequadas.

No próximo capítulo, será apresentado um estudo com a finalidade de observar a utilidade e a facilidade de uso da infraestrutura de apoio computacional para especificação de requisitos de ubiquidade.

6 Avaliação

Neste capítulo, apresentamos uma avaliação da abordagem proposta realizada com membros do Grupo de Engenharia de Software Experimental (ESE) da COPPE/UFRJ para identificar indícios do seu apoio na definição de requisitos de ubiquidade em projetos de software através da utilização da infraestrutura de apoio computacional desenvolvida, além dos resultados extraídos a partir da análise dos dados obtidos nessa avaliação.

6.1 Introdução

Com o objetivo de se obter resultados sobre a utilização da abordagem de apoio a especificação de requisitos de ubiquidade, uma avaliação foi planejada e executada com membros do grupo ESE. Essa avaliação se baseou na obtenção de dados de uso da infraestrutura de apoio computacional proposta.

O propósito deste estudo é observar a aplicação do *UbiModel* em um projeto de desenvolvimento de software ubíquo para compreender se a abordagem pode ser utilizada para apoiar a definição de requisitos, considerando sua facilidade de uso e utilidade. O estudo foi realizado com membros do grupo ESE e os resultados foram analisados com o intuito de se obter um direcionamento das próximas ações em relação à evolução da abordagem.

6.2 Estudo de caso

6.2.1 Objetivo

O objetivo do estudo pode ser formalizado, usando o paradigma GQM (BASILI e ROMBACH, 1998), como:

Analisar	a infraestrutura de apoio computacional desenvolvida
com o propósito de	caracterizar
com respeito a	facilidade de uso (G1) e utilidade (G2)
do ponto de vista dos	especificadores de requisitos
no contexto de	especificação de requisitos de ubiquidade descrevendo requisitos de um sistema de gestão de inventário patrimonial por estudantes de pós graduação de Engenharia de Software.

6.2.2 Seleção do Contexto

O cenário de ubiquidade selecionado para o estudo está relacionado ao controle de bicicletários, onde os usuários podem retirar bicicletas, usá-las para realizar um deslocamento qualquer e, posteriormente, devolvê-las. Esse cenário foi selecionado por conveniência, pois as regras de funcionamento do bicicletário já haviam sido definidas no contexto de outro trabalho, e por representar um alto nível de cobertura para a característica Sensibilidade ao Contexto, pois esta concentra o maior número de elementos que fazem interseção com as demais características.

6.2.3 Seleção de Variáveis

Foram observados os aspectos de utilidade e facilidade de uso segundo o modelo TAM (*Technology Acceptance Model*) (DAVIS *et. al.,* 1989) que tem por objetivo identificar porque determinada tecnologia é aceita ou rejeitada pelos usuários. A avaliação é baseada nos conceitos de percepção sobre utilidade e percepção sobre facilidade de uso, que são definidos como:

- percepção sobre utilidade: o quanto o usuário acredita que usando uma determinada tecnologia ele pode melhorar seu desempenho;
- percepção sobre facilidade de uso: o quanto o indivíduo acredita que usando uma determinada tecnologia ele pode ficar livre de esforço físico ou mental (DAVIS, 1993).

As questões relacionadas ao TAM foram elaboradas com base no estudo de LAITENBERGER e DREYER (1998), usando a escala Likert (MCIVER e CARMINES, 1991) como opções de resposta (concordo totalmente, concordo amplamente, concordo parcialmente, discordo parcialmente, discordo amplamente e discordo totalmente). Seis questões avaliavam a facilidade de uso da infraestrutura computacional (Q1 a Q6) e cinco a sua utilidade (Q7 a Q11) (Tabela 6-1). O questionário fornecia, ainda, um espaço para que os participantes pudessem fazer comentários. O Apêndice E apresenta o questionário utilizado.

A Figura 6-1 apresenta o modelo GQM, que é composto por dois objetivos, onze questões (Tabela 6-1) e seis métricas (Tabela 6-2). Os objetivos abordam os itens propostos pelo TAM.

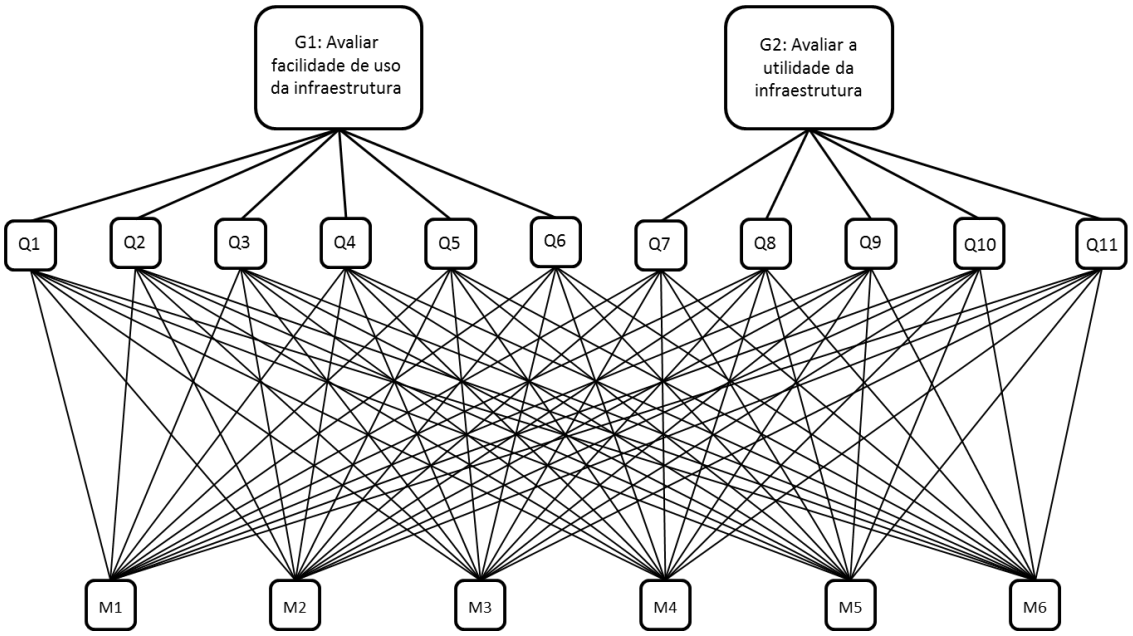


Figura 6-1- GQM para avaliar a infraestrutura

Tabela 6-1 – Questões utilizadas na avaliação

Questão	Descrição
Q1	Foi fácil aprender a utilizar a infraestrutura
Q2	Consegui utilizar a infraestrutura para realizar a especificação
Q3	A interação com a infraestrutura foi clara e compreensível
Q4	Seria fácil ganhar habilidade no uso da infraestrutura
Q5	Foi fácil compreender como realizar uma especificação de requisitos com o uso da infraestrutura
Q6	Considero a infraestrutura fácil de usar
Q7	Usar a infraestrutura melhorou o meu desempenho durante a especificação de requisitos de ubiquidade
Q8	Usar a infraestrutura facilitou a especificação de requisitos de ubiquidade
Q9	Usar a infraestrutura melhorou minha produtividade na especificação de requisitos de ubiquidade
Q10	Eu considero a infraestrutura útil para especificar requisitos de ubiquidade
Q11	Eu considero que a infraestrutura melhora a efetividade da especificação de requisitos

Tabela 6-2 – Métricas usadas no GQM

Métrica	Descrição
M1	Número de participantes que escolheram "Concordo totalmente"
M2	Número de participantes que escolheram "Concordo amplamente"
M3	Número de participantes que escolheram "Concordo parcialmente"
M4	Número de participantes que escolheram "Discordo totalmente"
M5	Número de participantes que escolheram "Discordo amplamente"
M6	Número de participantes que escolheram "Discordo parcialmente"

O intuito da avaliação é verificar se os desenvolvedores são capazes de aplicar a abordagem em um projeto de software e suas percepções de utilidade e facilidade de uso ao utilizar a infraestrutura de apoio.

6.2.4 Participantes

Os participantes foram selecionados por conveniência dentro do grupo ESE. No total, treze participantes assinaram o termo de consentimento e participaram do estudo.

6.2.5 Projeto do Estudo

A caracterização do cenário selecionado, na qual é baseada a construção do roteiro de especificação, foi realizada previamente pelo pesquisador para que todos os participantes tivessem acesso às mesmas diretivas de especificação. Desta forma, todos os participantes, independente da sua caracterização, trabalharam com o mesmo cenário de ubiquidade, o mesmo roteiro de especificação, a mesma abordagem e a mesma infraestrutura de apoio. Não foi realizada, portanto, a comparação da infraestrutura proposta com nenhuma outra abordagem. No Apêndice F estão disponíveis os cenários de ubiquidade completos utilizados no treinamento e na execução do estudo.

6.2.6 Instrumentação

Os instrumentos preparados e usados como apoio ao estudo foram:

1. Formulário de consentimento;
2. Material de treinamento sobre o metamodelo *UbiModel*;
3. Material de treinamento sobre as ferramentas que compõem a infraestrutura de apoio a especificação de requisitos de ubiquidade;
4. Documento com a descrição do cenário de controle do bicicletário;
5. Projeto na ferramenta case BOUML vazio, ou seja, sem nenhuma especificação, contendo somente o projeto previamente caracterizado e as ferramentas de apoio para criação da especificação.

Os instrumentos do estudo podem ser encontrados no Apêndice F e G.

6.2.7 Preparação

O pacote de treinamento dos participantes foi dividido em dois módulos:

1. Treinamento sobre o metamodelo *UbiModel*, suas definições e restrições;

2. Treinamento sobre a criação e edição da especificação dos requisitos de ubiquidade através da infraestrutura de apoio.

Os dois treinamentos foram ministrados para todos os participantes, um após o outro, na mesma sessão de treinamento. O treinamento 2 foi realizado com o apoio de um cenário relacionado ao controle dos itens de patrimônio, onde a entrada e saída de um item da organização, realizada por usuários, é controlada pelo sistema. Todos os requisitos de ubiquidade para esse cenário foram especificados passo a passo com o objetivo de apresentar todos os detalhes da infraestrutura de apoio.

6.2.8 Execução

Houve uma única sessão de treinamento com todos os participantes e a execução do estudo foi realizada logo em seguida. Cada participante teve 1 hora para descrever os requisitos com o apoio das ferramentas da infraestrutura computacional e ao final, entregaram a especificação de requisitos e o questionário de avaliação. A execução do estudo foi realizada em um dos laboratórios do Programa de Engenharia de Sistemas e Computação - PESC.

6.2.9 Validação dos dados

Todos os treze participantes entregaram seus questionários e nenhuma especificação ou questionário foram descartados.

6.2.10 Resultados

A Tabela 6-3 apresenta a quantidade de respostas fornecidas pelos participantes para cada questão. A Tabela 6-4 apresenta o modelo de interpretação do GQM, o qual deve ser lido da seguinte forma: “Se Expressão então Interpretação”, ou seja, tomando como exemplo a linha 4, na qual as questões de referência variam de Q7 a Q11, “Se M1 >= M2+M3 então a infraestrutura é vista como um recurso útil para apoiar a especificação de requisitos de ubiquidade”.

Tabela 6-3 – Dados coletados no questionário de avaliação da infraestrutura

Questão	Concordo totalmente	Concordo amplamente	Concordo parcialmente	Discordo parcialmente	Discordo amplamente	Discordo totalmente
Q1	6	3	3	1		
Q2	4	4	4		1	
Q3	2	2	8	1		
Q4	6	7				
Q5	3	5	1	1	2	1
Q6	4	5	3	1		

Q7	5	4	3			
Q8	6	4	3			
Q9	6	2	4			
Q10	8	3	1	1		
Q11	6	5	2			

Tabela 6-4 – Modelo de interpretação do GQM

Nº	Expressão	Interpretação
1	Para Q_i, i = 1 a 6: M1 >= M2+M3	A infraestrutura é fácil de ser utilizada.
2	Para Q_i, i=1 a 6: M1 <= M2+M3 e M2 >= M3	A infraestrutura é fácil de ser utilizada, porém, deve ser feita a análise dos dados enviados pelos participantes, por meio de comentários, com o intuito de identificar as melhorias necessárias para facilitar o uso das ferramentas.
3	Para Q_i, i=1 a 6: M1 <= M2+M3 e M3 >= M2	A infraestrutura não possui facilidade de uso e nesse caso deve ser estudado padrões de usabilidade e heurísticas definidos na literatura para realizar uma auto-avaliação sobre a interface das ferramentas, os comentários enviados pelos participantes devem ser analisados e, com base nessa análise, o projeto da infraestrutura deve ser revisto e as melhorias implementadas.
4	Para Q_i, i=7 a 11: M1 >= M2+M3	A infraestrutura é útil para apoiar a especificação de requisitos de ubiquidade.
5	Para Q_i, i=7 a 11: M1 <= M2+M3 e M2 >= M3	A infraestrutura é útil para apoiar a especificação de requisitos de ubiquidade, porém, deve ser feita a análise dos dados enviados pelos participantes, por meio de comentários, com o intuito de identificar as melhorias necessárias para tornar as ferramentas úteis.
6	Para Q_i, i=7 a 11: M1 <= M2+M3 e M3>=M2	A infraestrutura não é de grande utilidade e nesse caso o projeto da infraestrutura deve ser revisto com o intuito de identificar se há funcionalidades que possam ser implementadas para prover maior utilidade.
7	Para Q_i, i=1 a 6: M6 + M5 + M4 >= M1 + M2 + M3	A implementação de novas funcionalidades deve ser abortada, os comentários enviados pelos participantes devem ser analisados e o projeto da infraestrutura deve ser totalmente revisto, principalmente a interface com o usuário.
8	Para Q_i, i = 7 a 11: M6 + M5 + M4 >= M1 + M2 + M3	A implementação de novas funcionalidades deve ser abortada, os comentários enviados pelos participantes devem ser analisados e o projeto da infraestrutura deve ser totalmente revisto.

Nas figuras 6-2 e 6-3, são apresentados gráficos com as questões e respostas do questionário. Nos gráficos, a ordem das barras obedece a ordem das questões.

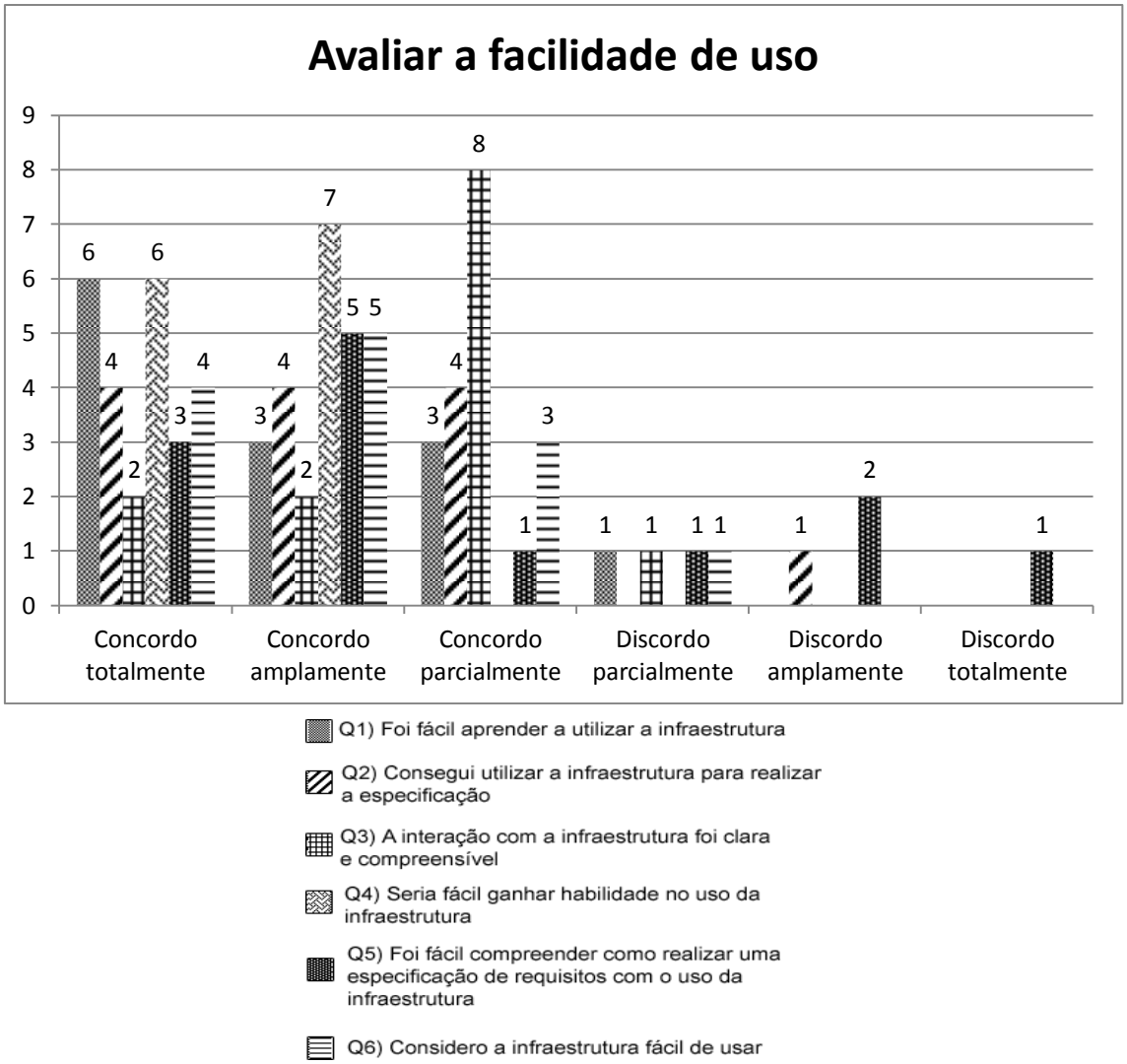


Figura 6-2- Questões e respostas relacionadas à facilidade de uso

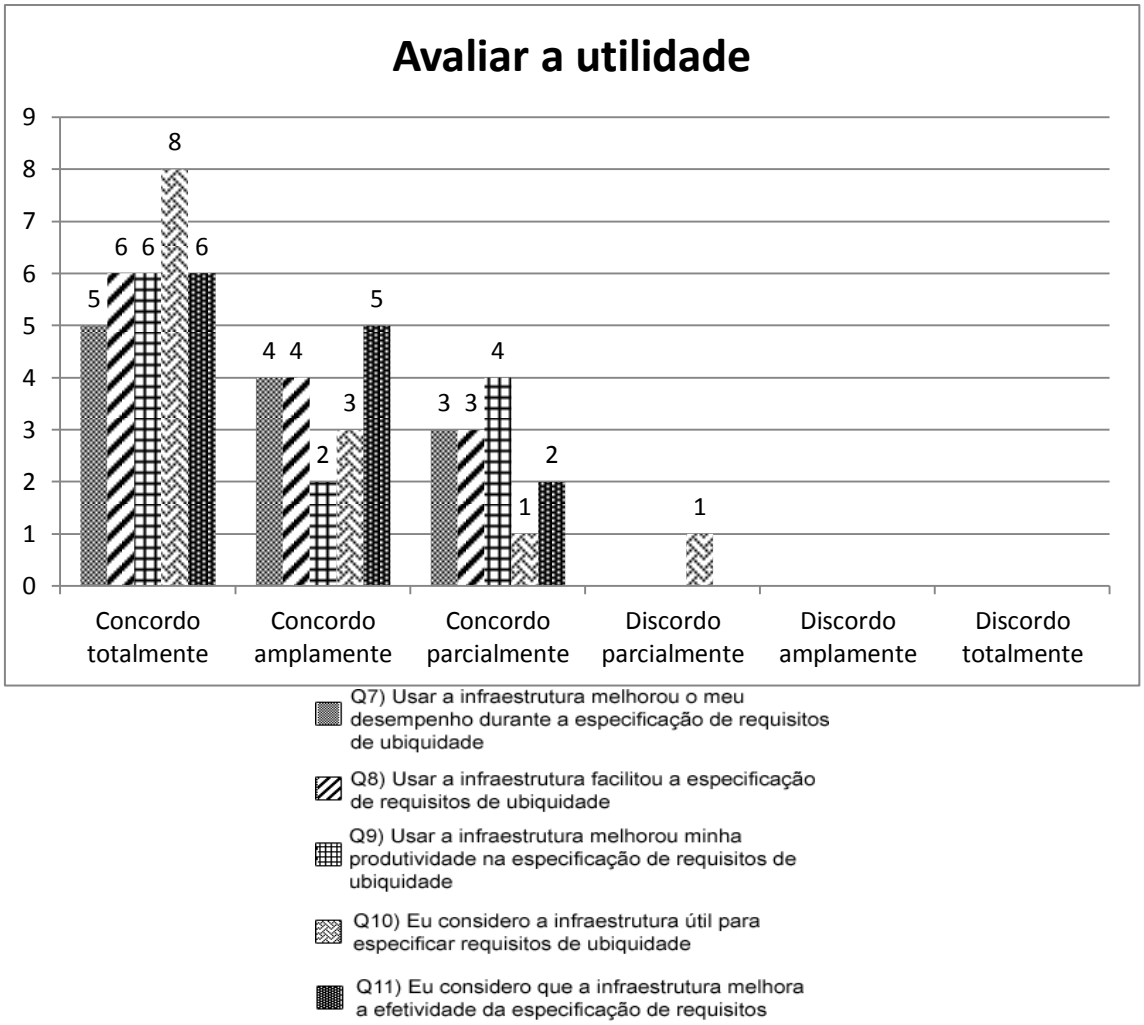


Figura 6-3- Questões e respostas relacionadas à utilidade

Considerando o modelo de interpretação apresentado na Tabela 6-4 e os dados coletados durante a avaliação, representados nas figuras acima, todos os objetivos da avaliação foram contemplados e o que se pode dizer acerca deles é:

- G1: a maioria dos participantes concorda amplamente com a facilidade de uso da infraestrutura. Isso pode ser constatado na Figura 6-2, em que 26 respostas dos participantes estão concentradas na opção “Concordo amplamente”, o que satisfaz a expressão 2 (ou seja, 25 <= 26 + 19 e 26 >= 19) e de acordo com o modelo de interpretação da Tabela 6-4, indica que o próximo passo deve ser a análise dos dados enviados pelos participantes por meio de comentários com o intuito de identificar as melhoras necessárias para facilitar o uso da infraestrutura.
- G2: a maioria dos participantes concorda totalmente com a utilidade da infraestrutura. Isso pode ser confirmado na Figura 6-3, em que 31 respostas dos participantes estão concentradas na opção “Concordo totalmente”, o que satisfaz a expressão 4 (ou seja 31 >= 18 + 13) e de

acordo com o modelo de interpretação da Tabela 6-4, indica que a infraestrutura é útil para apoiar a especificação de requisitos de ubiquidade.

Em relação à facilidade de uso da infraestrutura, uma análise dos dados fornecidos pelos participantes por meio de comentários foi realizada. O entendimento de alguns conceitos da computação ubíqua foi a principal dificuldade relatada pelos participantes. Essa dificuldade pode ter sido causada pela não exploração, em um nível de detalhe adequado, dos conceitos durante o treinamento. Alguns participantes sugeriram que o glossário de termos apresente informações mais detalhadas acerca dos conceitos ou ainda que apresente exemplos dentro da própria ferramenta. Além disso, um dos participantes sugeriu que a ferramenta de especificação dos requisitos permita a edição de contextos. Na solução atual, caso haja necessidade de editar um contexto o mesmo deve ser excluído e um novo contexto deve ser criado. O mesmo participante sugeriu, ainda, a utilização de um componente na listagem das ações do sistema que permita a alteração da ordem em que as ações ocorrem de forma mais intuitiva. Na solução atual, a ordenação das ações é alterada pela edição de um campo que recebe um número inteiro.

Essas melhorias deverão ser implementadas em uma próxima versão da ferramenta.

6.3 Conclusão

Este capítulo apresentou uma avaliação das ferramentas que compõem a infraestrutura de apoio a especificação de requisitos de ubiquidade (*UbiSpecification* e *UbiDocument*) que foi planejada usando o GQM e estabeleceu como objetivos os aspectos tratados pelo método TAM – facilidade de uso e utilidade. O uso do TAM tornou a avaliação rápida e objetiva, enquanto as diretrizes do GQM propiciaram objetividade à avaliação e à definição e elaboração do formulário para coleta de dados.

A avaliação tinha como objetivo caracterizar a facilidade de uso e utilidade de se especificar requisitos de ubiquidade utilizando a infraestrutura de apoio desenvolvida.

Assim, o resultado da avaliação trás indícios de que a infraestrutura é útil, uma vez que 49,02% dos participantes concordaram totalmente com a utilidade da infraestrutura, 28,57% dos participantes concordaram amplamente, 20,63% dos participantes concordaram parcialmente e apenas 1,58% dos participantes discordaram parcialmente sobre a utilidade da infraestrutura. Quanto à facilidade de

uso, 32,05% dos participantes concordaram totalmente com esse item, 33,33% dos participantes concordaram amplamente, 24,35% dos participantes concordaram parcialmente, 5,12% dos participantes discordaram parcialmente, 3,84% dos participantes discordaram amplamente e 1,28% dos participantes discordaram totalmente. De acordo com o resultado da avaliação, ações devem ser tomadas para que a facilidade de uso da infraestrutura seja aprimorada. Em resumo, apesar de não ser possível generalizar os resultados, a avaliação forneceu indícios de que a infraestrutura é útil para apoiar a especificação de requisitos de ubiquidade e que a infraestrutura é fácil de usar, mas pode ser aprimorada.

7 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.

7.1 Considerações Finais

A computação ubíqua é definida por um conjunto de características específico que tende a aumentar a complexidade e trazer desafios ao desenvolvimento de software nesse domínio. Segundo DUCATEL *et. al.* (2003), essas características 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.

A partir desse cenário, SPÍNOLA (2010) define uma abordagem para auxiliar as atividades de definição e verificação dos requisitos funcionais de ubiquidade do projeto através de um conjunto de guias. Com o objetivo de se obter uma especificação de software mais completa, a abordagem foi elaborada a partir de um conjunto de características e fatores da computação ubíqua (SPÍNOLA *et. al.* 2006) que considera as especificidades desse domínio frente a ubiquidade computacional.

Nesse contexto, o objetivo dessa dissertação foi estender a abordagem apresentada por SPÍNOLA (2010) através da definição de um metamodelo, denominado *UbiModel*, que tem o objetivo de organizar as características e fatores de ubiquidade e suas interrelações de forma mais concreta e detalhada para guiar a geração da especificação e minimizar o caráter interpretativo sobre essas características e fatores presentes na abordagem original. A partir da definição do metamodelo *UbiModel* foi possível, também, definir uma infraestrutura computacional de apoio à especificação de requisitos de ubiquidade. O uso dessa infraestrutura computacional tem a finalidade de reduzir o esforço dedicado à especificação de requisitos de ubiquidade, pois grande parte das atividades são automatizadas.

Foi realizada, ainda, a junção do *UbiModel*, que apresenta os requisitos sob a ótica do usuário, à abordagem apresentada por MASSOLLAR (2011), que apresenta os requisitos sob a ótica do comportamento esperado do sistema, com o objetivo de tornar a descrição dos requisitos mais completa.

Por fim, foi realizada uma avaliação que apresentou indícios de que a ferramenta para especificação de requisitos de ubiquidade é útil e que sua usabilidade deve ser aprimorada através da evolução do glossário de termos para apresentar informações mais detalhadas.

7.2 Contribuições da Pesquisa

As contribuições desta pesquisa estão relacionadas à definição de uma abordagem para definição de requisitos alinhada aos conceitos relacionados a computação ubíqua, fornecendo uma estrutura para apoiar a especificação, que envolve:

- Revisão dos artefatos produzidos nas pesquisas desenvolvidas por PINTO (2009) e SPÍNOLA (2010), por constituírem a principal fonte de informação para a elaboração do *UbiModel*;
- Atualização das descrições das características de ubiquidade de acordo com a interpretação dada na elaboração do metamodelo *UbiModel* em função de novos elementos originados ao detalhar os conceitos relacionados a abordagem proposta em (SPÍNOLA, 2010);
- Elaboração de um glossário de termos sobre computação ubíqua para facilitar a leitura por parte do especificador dos diversos conceitos e orientações específicos a esse domínio;
- Estruturação do metamodelo *UbiModel*, que permite a organização da especificação dos requisitos segundo um conjunto de critérios e restrições bem definido e oferece a estruturação a partir do qual é possível tratar questões relacionadas à garantia da qualidade;
- Elaboração de um conjunto de orientações para auxiliar na redação das descrições dos requisitos de ubiquidade;
- Obtenção de especificação mais abrangente, ao realizar a integração dos requisitos sob a ótica das demandas e necessidades do usuário a outra abordagem onde os requisitos são definidos sob a ótica do comportamento do sistema (MASSOLLAR, 2011).
- Implementação da infraestrutura computacional, destinada a apoiar a definição de requisitos de ubiquidade, que implementa a estrutura e restrições definidas no metamodelo *UbiModel*;
- Planejamento execução e análise de estudo de viabilidade que apoia a avaliação e evolução das tecnologias propostas;

7.3 Limitações e Trabalhos Futuros

No decorrer deste trabalho algumas limitações foram identificadas. A primeira limitação deste trabalho foi que não houve oportunidade para realizar um estudo para caracterizar a eficiência e eficácia do apoio fornecido por *UbiModel* durante a definição de requisitos. No entanto, foi realizado um estudo (Capítulo 6) que sugere que a infraestrutura de apoio computacional é útil para apoiar a definição de requisitos em projetos de software ubíquo.

Outra limitação deste trabalho está relacionada ao fato da abordagem desenvolvida fornecer apoio somente às características funcionais da computação ubíqua acompanhando a abordagem original de SPÍNOLA (2010), onde as características restritivas não são consideradas. Sendo assim, uma evolução natural deste trabalho seria criar mecanismos de apoio à definição e verificação dos requisitos não funcionais de ubiquidade.

Por fim, a abordagem proposta nesta pesquisa oferece oportunidades de pesquisa no que diz respeito a:

- definição ou adaptação de técnicas de inspeção à luz dos conceitos definidos na abordagem e que explorem os elementos que a compõem: modelos conceituais, regras, descrições, dentre outras;
- mapeamento dos requisitos definidos por meio do *UbiModel* para os possíveis itens de especificação que constituem o detalhamento desses requisitos, e;
- realização de estudos experimentais para obter indícios da eficiência e eficácia do apoio fornecido por *UbiModel* e a redução de defeitos em documentos de especificação.

REFERÊNCIAS BIBLIOGRÁFICAS

APPOLINÁRIO, F. (2004). “Dicionário de Metodologia Científica: Um Guia para a Produção do Conhecimento Científico”. Atlas, São Paulo, SP, Brasil.

AURUM, A., WOHLIN, C. (2005). “Engineering and Managing Software Requirements”, Springer-Verlag, New York, EUA.

BARROS, A. J. S., LEHFELD, N. A. S. (2000). “Fundamentos de Metodologia: Um Guia para a Iniciação Científica”. Makron Books, São Paulo, SP, Brasil, 2 edição.

BOEHM, B.W. (1981). “Software Engineering Economics”. Prentice Hall

BOEHM, B., BASILI, V. R. (2001). “Software Defect Reduction Top 10 List”. IEEEComputer Society Press, v. 34, n. 1, pp. 135-137

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., STEIN, C. (2002). “Algoritmos – Teoria e Prática”. Campus, Rio de Janeiro, RJ, Brasil, 2 edição.

DAVIS, F. D.; BAGOZZI, R.; WARSHAW, P. R. (1989). “User acceptance of computer technology: a comparison of two theoretical models”. In Management Science, pp.982-1003.

DAVIS, F. D. (1993). “User acceptance of information technology: system characteristics, user preceptions and behavioral impacts”. In Int. J. Man-Macgine Studies (v.38), pp.475-487.

DRYER, D. C.; EISBACH, C. & ARK, W. S. (1999).“At what cost pervasive? A social computing view of mobile computing systems”, IBM Systems Journal, Vol.38, No.4, pp.652-676

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

JACOBSON, I. (1992). “Object-oriented software engineering: a use case driven approach”, Addison- Wesley.

KAPPEL, G., PRÖLL. B., RETSCHITZEGGER, W., SCHWINGER, W., HOFER, T. (2001) “Modeling Ubiquitous Web Applications - A Comparison of Approaches”, Proceedings of the Interational Conference on Information Integration and Web-based Applications and Services (iiWAS2001), Austria.

LAITENBERGER, O., DREYER, H. M. (1998). “Evaluating the Usefulness and the Ease of Use of a Web-based Inspection Data Collection Tool”. In International Symposium on Software Metrics, pp. 122-135.

MCIVER, J. P.AND CARMINES, E. G. (1991).Unidimensional Scaling.Sage Publications.

MASSOLLAR, J.L. (2011). “Uma Abordagem para Especificação de Requisitos Dirigida por Modelos Integrada ao Controle da Qualidade de Aplicações Web”. Tese de Doutorado. COPPE/UFRJ. Rio de Janeiro.

MASSOLLAR, J L., MELLO, R. M.de, TRAVASSOS, G. H. (2011). “Investigating the Feasibility of a Specification and Quality Assessment Approach Suitable for Web Functional Requirements”. 30th International Conference of the Chilean, Computer Science Society (SCCC),Curico, Chile. pp.108,117, 9-11.

MASSOLLAR, J. L., MELLO, R. M. de, TRAVASSOS, G. H. (2012). “Structuring and Verifying Requirements Specifications through Activity Diagrams to Support the Semi-automated Generation of Functional Test Procedures”.QUATIC. Lisboa, Portugal.

MOTA, L. S., MASSOLLAR, J. L., TRAVASSOS, G. H. (2013). “Usando Modelos Para Apoiar a Especificação e Verificação de Requisitos de Ubiquidade”. RequirementsEngineering@Brazil, ER@BR2013, Rio de Janeiro, Brasil. pp.178-183.

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.

OMG (2010). “Unified Modeling Language Superstructure - version 2.3”, Object Management Group, http://www.omg.org/spec/UML/2.3/.

PAGÉS, B. (2011). BOUML, http://bouml.free.fr.

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.

PINTO, F. (2009). “Uma Abordagem para Apoiar Especificação de Requisitos para Projetos de Software Ubíquo”. Dissertação de Mestrado. COPPE/UFRJ. Rio de Janeiro.

SHULL, F., RUS, I., BASILI, V. (2000) “How Perspective-based Reading Can Improve Requirements Inspections”. July, IEEE Software, pp: 73-79

SOMMERVILLE, I. (2007). “Engenharia de Software”. Pearson, ISBN: 8588639289, 8a edição.

SPÍNOLA, R., SILVA, J., TRAVASSOS, G. (2006). “Towards a Conceptual Framework to Classify Ubiquitous Software Projects”.Proceedings of the 8th International Conference on Software Engineering and Knowledge Engineering

SPÍNOLA, R., SILVA, J., TRAVASSOS, G.H. (2007). “Characterizing Ubicomp Software Projects through a Checklist”, Proceedings of the I Workshop on Pervasive and Ubiquitous Computing.

SPÍNOLA, R. O., TRAVASSOS, G.H. (2008). “Arcabouço para Apoiar a Definição e a Garantia de Qualidade de Requisitos de Ubiqüidade em Projetos de Software”. II Workshop on Pervasive and Ubiquitous Computing. Campo Grande, Brasil.

SPÍNOLA, R. O., TRAVASSOS, G.H. (2010). “Characteristics of Ubiquitous Software Projects: Pertinence, Relevance, and Use”. Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering (SEKE 2010), San Francisco Bay, CA, USA.

SPÍNOLA, R., PINTO, F.C.R, TRAVASSOS, G. H. (2008). “Ubicheck: An Approach to Support Requirements Definition in the Ubicomp Domain”. 25th Symposium On Applied Computing, pp 306-310. Sierre, Suíça.

SPÍNOLA, R., PINTO, F.C.R, TRAVASSOS, G. H. (2009). “Supporting Requirements Definition and Quality Assurance in Ubiquitous Software Project”. Communications in Computer and Information Science, Vol. 17, Part 10, 587603, DOI: 10.1007/978-3-540-88479-8_42.

SPÍNOLA, R.O. (2010). “Apoio à Especificação e Verificação de Requisitos Funcionais de Ubiquidade em Projetos de Software”. Tese de Doutorado. COPPE/UFRJ. Rio de Janeiro.

WHEELER, D.A., BRYKEZYNSKI, B., MEESON, R.N. (1996). “Software Inspections: An Industry Best Practice”, IEEE Computer Society.

WIEGERS, K. (2003). “Software Requirements”. Microsoft Press, 2a edição.

APÊNDICE A – Lista de Defeitos

Lista de defeitos oriundos da revisão das abordagens de PINTO (2009) e SPÍNOLA (2010).

1. No meta-modelo das características de software ubíquo, a multiplicidade da associação entre Serviço e Entidade, no lado da Entidade, deve ser 1..*. Esse defeito ocasiona perguntas incompletas relacionadas a essas entidades no *checklist*.
2. No modelo de Captura da Experiência de acordo com a lista de fatores que originou o modelo, o tipo do relacionamento entre Relacionamento (entidade) e Analisar (serviço) deve ser uma associação. Esse defeito ocasiona perguntas incorretas relacionadas a essas entidades no *checklist*.
3. No modelo de Comportamento Adaptável de acordo com a lista de fatores que originou o modelo, o tipo do relacionamento entre Alteração (entidade) e Prever (serviço) deve ser uma associação. Esse defeito ocasiona perguntas incorretas relacionadas a essas entidades no *checklist*.
4. No Guia de Definição de Requisitos de Ubiquidade a pergunta “Como analisar as informações do ambiente?” aparece duplicada, porém com orientações diferentes. Esse defeito poderia ocasionar a inserção de informações inconsistentes ou redundantes comprometendo a qualidade da especificação de requisitos.
5. Diversos direcionamentos não apresentam os campos Orientação e Item de Especificação preenchidos no Guia Geral de Definição de Requisitos de Ubiquidade com Direcionamento. A falta desses campos poderia dificultar a especificação desses requisitos, comprometendo a qualidade da especificação de requisitos.

APÊNDICE B – Glossário de Termos

Context Information: Informação que está relacionada aos elementos do ambiente controlado pelo sistema. Ela pode ser obtida através de fontes de dados ou por meio de transformações de outras informações de contexto. Deve ser considerado o momento que a informação é obtida, além do critério e do meio de armazenamento da informação.

User Information: Informações de contexto relacionadas aos usuários.

System Information: Informações de contexto relacionadas às restrições de infraestrutura (uso de memória, cpu, tamanho de tela, energia, largura de banda disponível).

Infrastructure Information: Informações de contexto relacionadas ao correto funcionamento da infraestrutura (estado de funcionamento dos sensores, por exemplo).

Physical Information: Informações de contexto relacionadas ao ambiente (temperatura, iluminação, ruído).

Data Source: Fonte de onde uma informação de contexto é obtida pelo sistema.

Software: Fonte de dados, do tipo software, que fornece informações para o sistema ou consome informações geradas pelo mesmo. Pode ser um ator do sistema.

Hardware: Fonte de dados, do tipo hardware, que fornece informações para o sistema ou consome informações geradas pelo mesmo. Pode ser um ator do sistema.

Device: Hardware, do tipo dispositivo, que fornece informações para o sistema ou consome informações geradas pelo mesmo. Pode ser um ator do sistema. Considera dispositivos do tipo sensor.

Transformation: Como informações de contexto obtidas via fonte de dados podem originar outra informação de contexto.

Actor: Usuário ou meio externo que interage com o sistema.

User: Indivíduo que interage com o ambiente controlado pelo sistema.

Location: Localização controlada pelo sistema.

Identity: Forma de identificar os usuários nos locais controlados pelo sistema.

Identification: Localizações do ambiente, seus respectivos usuários e como esses usuários são identificados em cada localização.

Activity: Evento controlado pelo sistema (reunião, palestra, aula, etc.).

Behavior: Como o sistema deve se comportar ao perceber um determinado contexto.

Trigger: Ordem de execução da ação do sistema para um comportamento.

System Action: Algo que o sistema deve fazer (ligar internet, enviar email, etc.).

Condition: Uma informação sobre o estado desejado do ambiente, em função de uma informação de contexto, definida por meio de uma expressão.

Context: Um cenário de utilização do sistema, composto por localizações do ambiente, usuários, atividades e condições determinada.

Interaction: Uma ação que o usuário realiza no ambiente controlado pelo sistema.

Interaction Pattern: Sequência de interações que ocorre de maneira recorrente, de acordo com um critério, que ao ser identificada pelo sistema o mesmo deve realizar ações que beneficiem o usuário.

Own: Ordem da interação dentro de um padrão específico.

Trigger: Ordem de execução da ação do sistema para um padrão de interação.

Preference: Característica do ambiente, para um contexto específico, que ao ser violada faz com que o sistema tente se adaptar.

Restriction: Limita uma informação de contexto a valores determinados por meio de expressões que representam as preferências que o sistema deve garantir.

System Adaptation: Adaptação do sistema para uma preferência violada, que considera a estratégia de seleção de uma alternativa de adaptação, o tempo de espera até que a adaptação seja realizada de fato e o armazenamento das informações sobre a adaptação.

Adaptation Alternative: Forma como uma adaptação pode ser realizada, considerando seu impacto sobre o sistema.

Service: Funcionalidade de um sistema disponível a outros sistemas fora dos seus limites.

Service Provided: Serviço do sistema disponível a outros sistemas.

Restriction: Limita uma informação de contexto a valores determinados por meio de expressões que representam as condições mínimas para que o serviço esteja disponível.

Protocol: Forma como os sistemas se comunicam. Comunicação se dá via serviços.

Data Format: Tamanho e tipo do dado utilizado por um serviço.

Input Data Format: Formato de dado do tipo entrada.

Output Data Format: Formato de dado do tipo saída.

Interoperability: Capacidade do sistema em utilizar serviços de outros sistemas.

Service Required: Serviço disponível por outros sistemas a ser utilizado para atender a uma demanda do sistema principal, considerando-se os requisitos mínimos para sua utilização.

Service Search: Forma como o sistema procura por serviços no ambiente, considerando-se uma estratégia para busca e para seleção dos serviços disponíveis.

Minimum Requirements: Requisitos mínimos para que um serviço externo seja utilizado pelo sistema.

Migration: Capacidade do sistema em continuar sua execução em um outro dispositivo com características diferentes do dispositivo atual.

Dynamic: O sistema é responsável por perceber eventos que iniciam uma migração, a partir daí, o mesmo deve buscar e selecionar dispositivos no ambiente segundo estratégias e critérios de disponibilidade dos dispositivos. A migração é iniciada pela ocorrência de um evento.

Static: Sistema executado por dispositivos heterogêneos onde todas ou algumas funcionalidades são adaptadas de acordo com os requisitos de migração. O usuário é o responsável por iniciar o sistema no dispositivo heterogêneo.

Event: Algo que inicia o processo de migração dinâmica (deslocamento do usuário, etc.).

Device Search: Forma como o sistema busca os dispositivos, considerando-se uma estratégia de busca e seleção dos dispositivos disponíveis no ambiente.

Availability: Critério que define a disponibilidade de migração.

Device Group: Grupo de dispositivos compatível para a realização de uma migração caracterizado por um conjunto de requisitos.

Requirement: Requisitos que os dispositivos heterogêneos devem ter para que a migração seja possível.

Software Requirement: Requisito relacionado a software.

Hardware Requirement: Requisito relacionado a hardware.

Adaptation: Adaptação de uma ação do sistema de acordo com os requisitos de migração.

APÊNDICE C – Modelo de Características de Ubiquidade

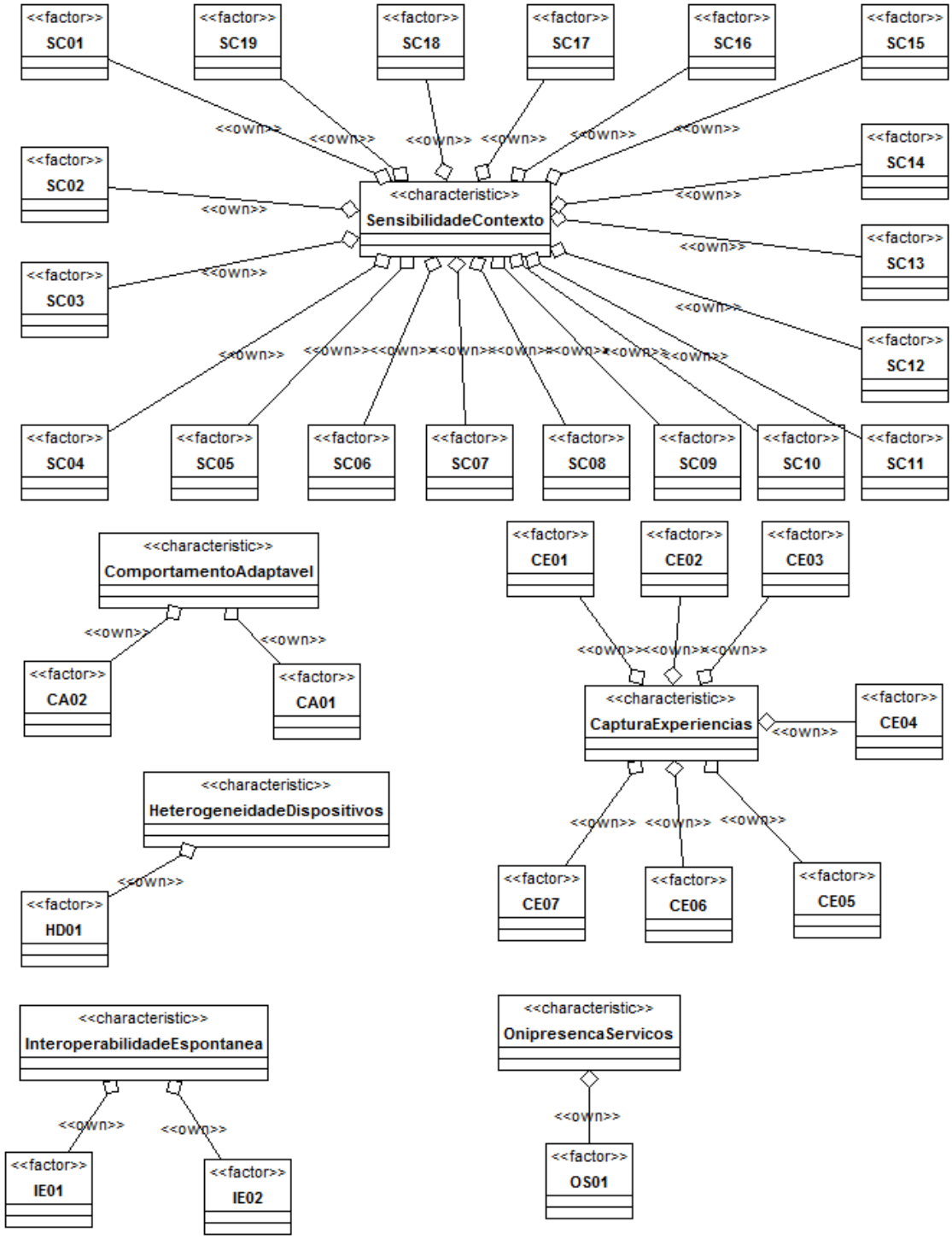


Figura C-1– Modelo de Características de Ubiquidade

APÊNDICE D – Modelo de Fatores

D.1 Captura de Experiências

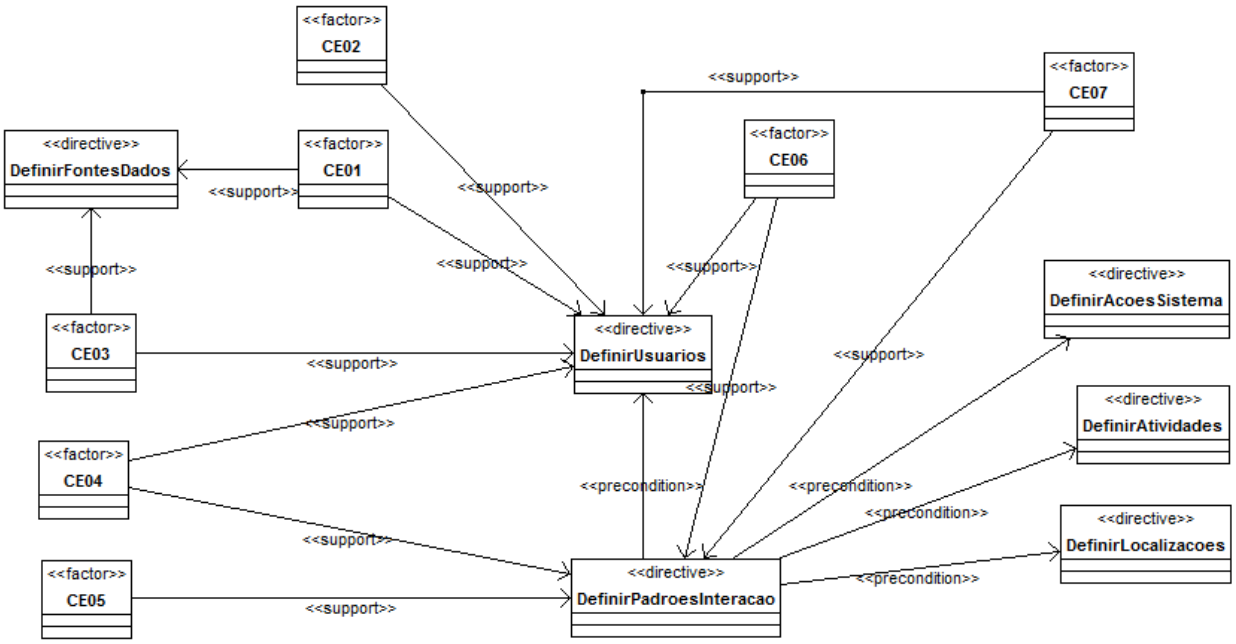


Figura D-1– Parte do modelo referente à Captura de Experiências

D.2 Comportamento Adaptável

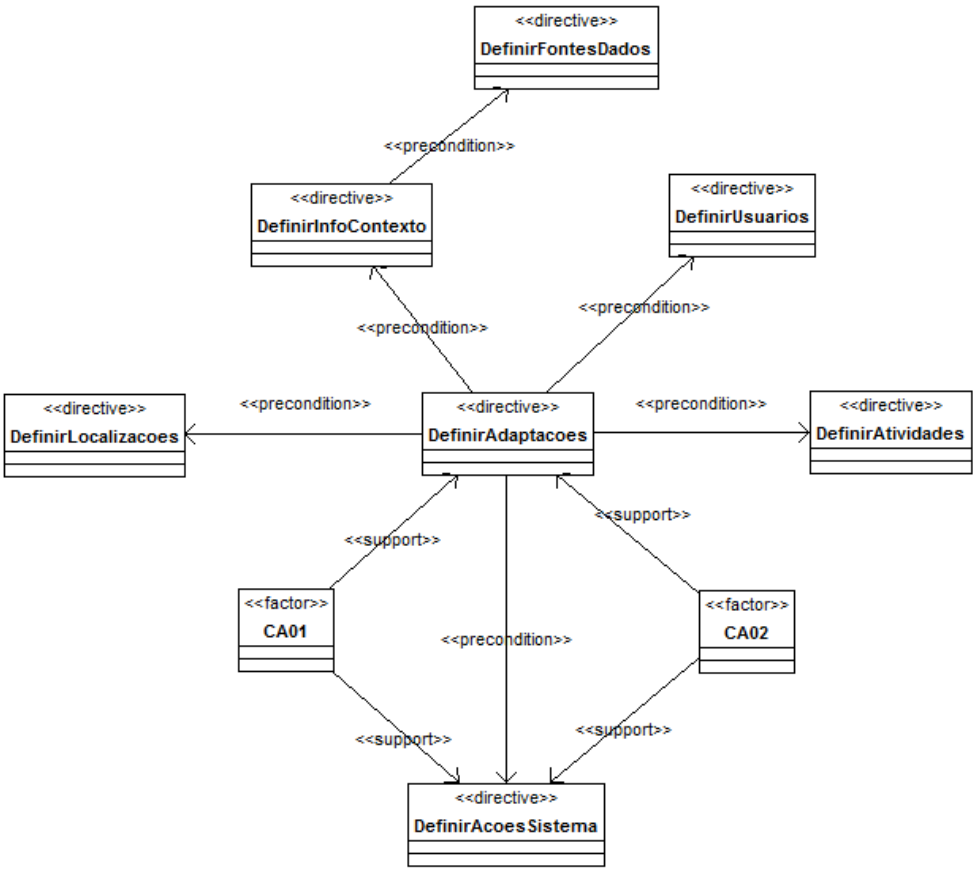


Figura D-2– Parte do modelo referente à Comportamento Adaptável

D.3 Onipresença de Serviços

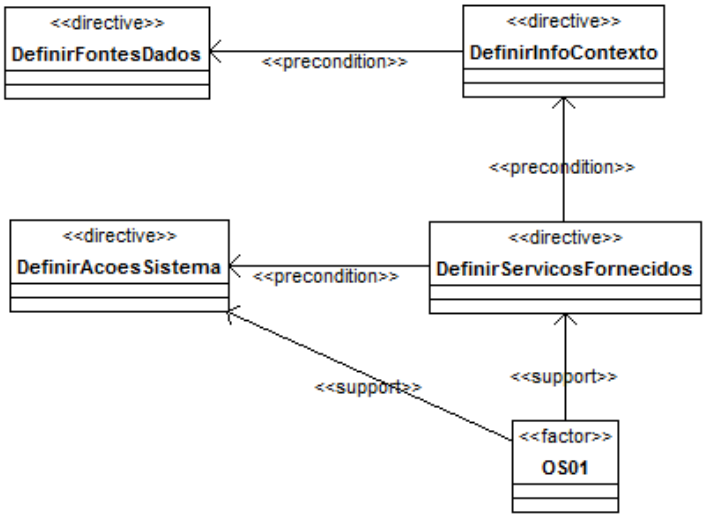


Figura D-3– Parte do modelo referente à Onipresença de Serviços

D.4 Heterogeneidade de Dispositivos

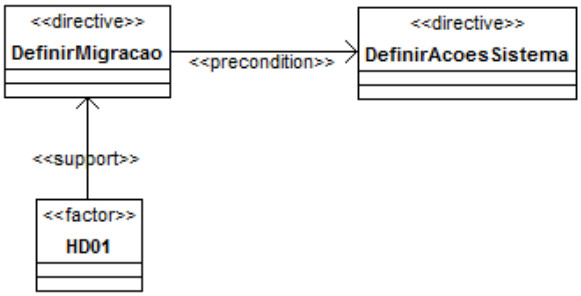


Figura D-4– Parte do modelo referente à Heterogeneidade de Dispositivos

D.5 Interoperabilidade Espontânea

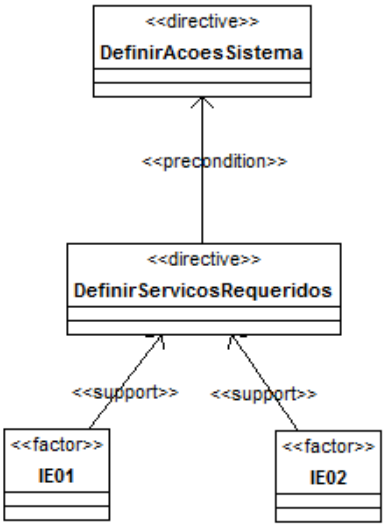


Figura D-5– Parte do modelo referente à Interoperabilidade Espontânea

APÊNDICE E – Modelo de Diretivas

E.1 Captura de Experiências

Figura

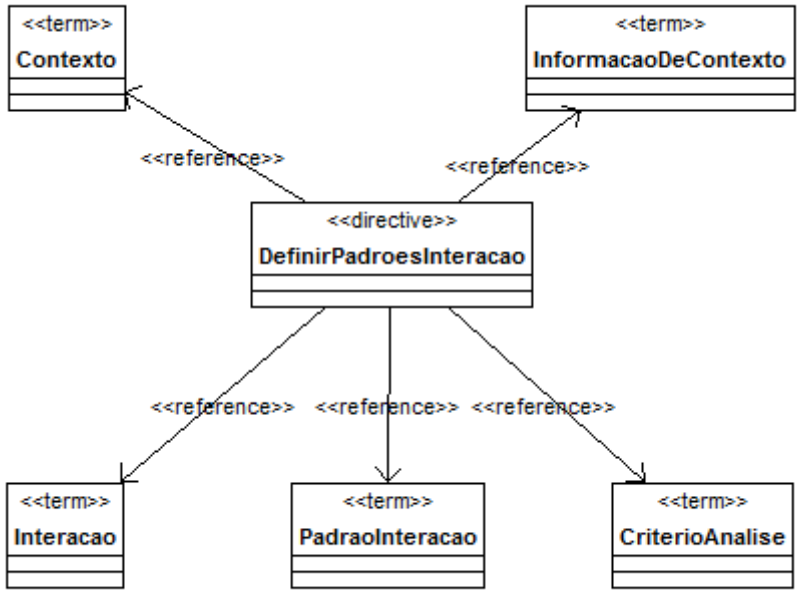


Figura E-1– Modelo de Diretivas para a característica Captura de Experiências

E.2 Comportamento Adaptável

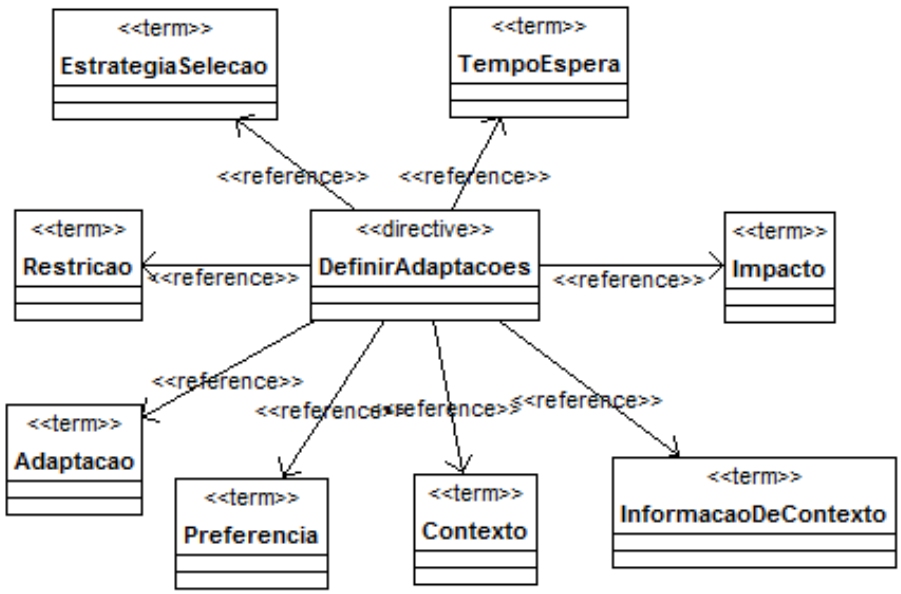


Figura E-2–Modelo de Diretivas para a característica Comportamento Adaptável

E.3 Onipresença de Serviços

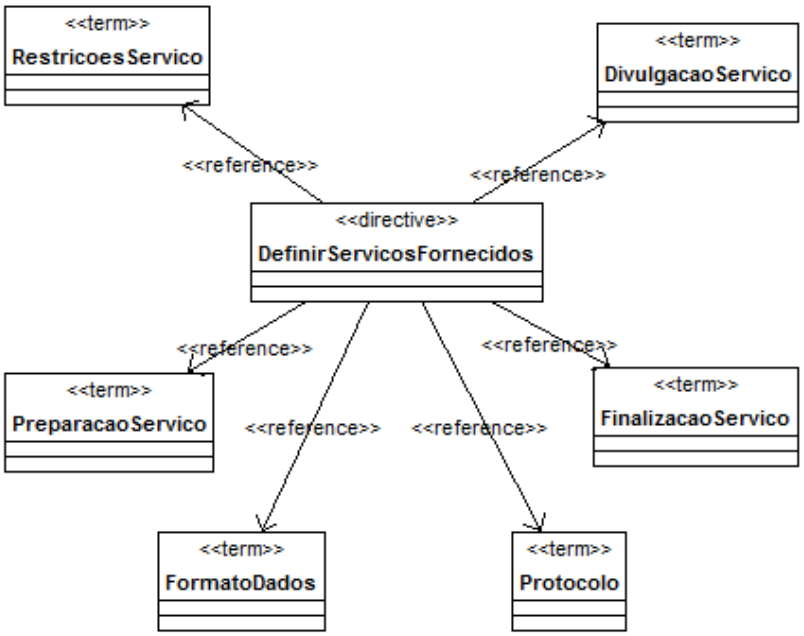


Figura E-3–Modelo de Diretivas para a característica Onipresença de Serviços

E.4 Heterogeneidade de Dispositivos

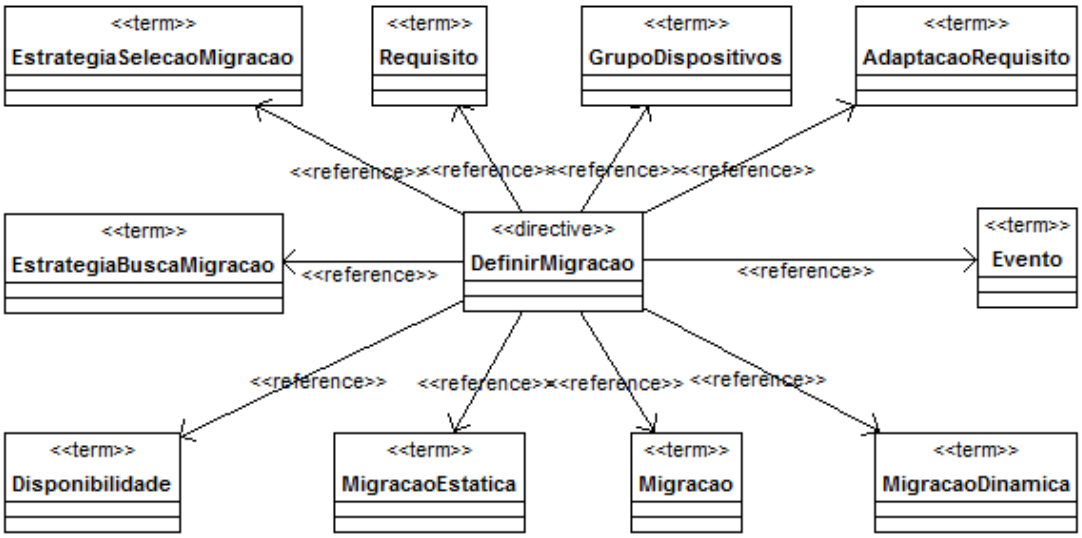


Figura E-4–Modelo de Diretivas para a característica Heterogeneidade de Dispositivos

E.5 Interoperabilidade Espontânea

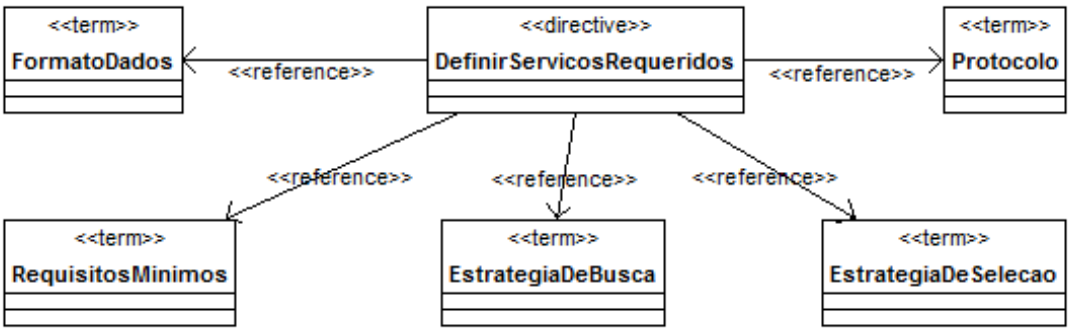


Figura E-5–Modelo de Diretivas para a característica Interoperabilidade Espontânea

APÊNDICE F – Cenários de Ubiquidade

F.1 Cenário de 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 ubiquidade 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.

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.

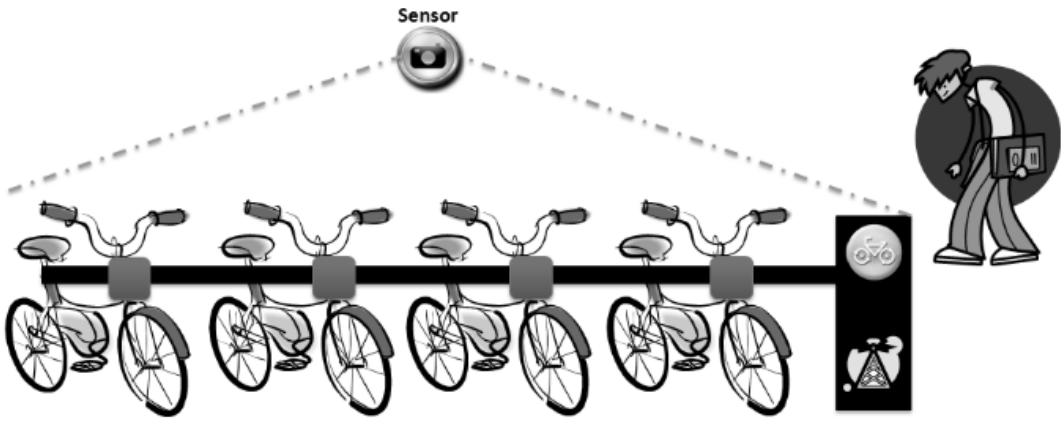
Para lidar com esta nova demanda, as seguintes necessidades adicionais foram definidas e devem ser consideradas no projeto do SGP:

- As bicicletas deverão ser classificadas como item rastreável. Uma bicicleta rastreável é aquela que, além do número de patrimônio, possui uma etiqueta eletrônica de identificação;
- Cada indivíduo da organização deve possuir um 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 número da bicicleta data e hora e o indivíduo da organização que ficará responsável pela bicicleta até que esta seja devolvida a um dos bicicletários. Em caso negativo, o sistema informa ao indivíduo 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.

F.2 Cenário de 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 ubiquidade 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.

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.

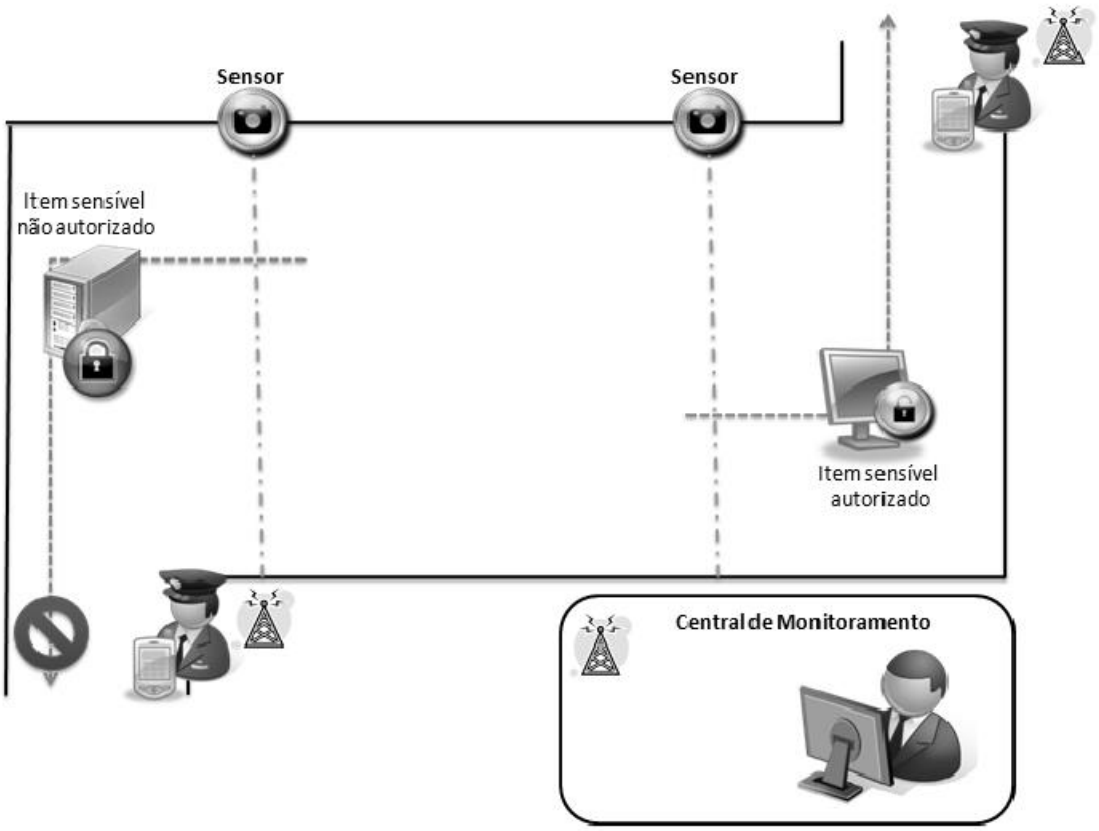
Para lidar com estas limitações, as seguintes necessidades adicionais foram definidas e devem ser consideradas no projeto do SGP:

- Os bens de capital passam a ser classificados também como item rastreável. Um item rastreável é aquele que, além do número de patrimônio, possui uma etiqueta eletrônica de identificação;
- Para ser classificado como rastreável, um item deve ter seus dados incluídos no cadastro de itens rastreáveis;
- As entradas e saídas dos prédios terão sensores para estas etiquetas. Ao passar com um item rastreável, deve ser registrado no sistema o evento, enviada uma mensagem para o setor de segurança e fazer com que o agente de portaria seja avisado que um item rastreável está saindo/entrando na instalação;
- A saída de um item rastreável passa a ser feita em duas etapas: (1) deve haver um cadastro de transferência considerando a solução atual descrita no documento de referência; (2) deve haver uma confirmação da saída do item pelo setor de segurança – essa confirmação envolve tanto a verificação se o item que está sendo retirado foi autorizado e se

o indivíduo que o está retirando está associado ao item como responsável pela sua retirada;

- A confirmação ou proibição de saída de um item rastreável deve sempre ser notificada para a central de monitoramento que acompanhará em tempo real as entradas e saídas de itens rastreáveis.

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:

1. 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 rastreável antes de sua saída;
2. Para que um item possa ser retirado, deve existir uma autorização prévia indicando também quem poderá retirar o item;

3. Ao passar por um sensor, uma mensagem é enviada ao setor de segurança indicando se o item está autorizado e qual indivíduo pode fazer sua retirada;
4. O setor de segurança recebe esta mensagem em um smartphone, faz a verificação da saída do item e autoriza ou não sua saída com base nas informações recebidas;
5. O resultado desta ação é enviado para a central de monitoramento.

APÊNDICE G – Instrumentos do Estudo

G.1 Termo de Consentimento

TERMO DE CONSENTIMENTO

Eu declaro ter mais de 18 anos de idade e que concordo em participar de um estudo conduzido pelos pesquisadores Prof. Guilherme Horta Travassos, Jobson Luiz Massollar da Silva e Leonardo da Silva Mota. Este estudo visa caracterizar a utilidade e a facilidade de uso da infraestrutura de apoio computacional para a definição de requisitos de ubiquidade.

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 fácil e útil é o uso da infraestrutura 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 participo de livre e espontânea vontade com o único intuito de contribuir para o avanço e desenvolvimento de técnicas, ferramentas e processos para a Engenharia de Software.

PESQUISADORES RESPONSÁVEIS
Prof. Guilherme Horta Travassos
Jobson Luiz Massollar da Silva
Leonardo da Silva Mota
Programa de Engenharia de Sistemas e Computação
COPPE/UFRJ

Nome (em letra de forma): _____

Assinatura: _____ Data:_____

G.2 Orientações para a Operação do Trabalho

Você está recebendo:

- Termo de consentimento
- Cenário de ubiquidade a ser especificado
- Questionário de avaliação da infraestrutura
- Ambiente com as ferramentas instaladas e configuradas

Acesso às ferramentas

A infraestrutura computacional foi implementada por meio de plug-ins para a ferramenta BOUML. Para acessar as ferramentas necessárias basta:

- Abrir a BOUML através do ícone localizado na área de trabalho;
- Com a BOUML aberta, abrir o projeto onde a especificação dos requisitos será criada. O arquivo do projeto está localizado na área de trabalho, na pasta avaliacao/TemplateUbiModel (TemplateUbiModel.prj)
- Para acessar o roteiro de especificação (lista de diretivas), com o projeto aberto, clique com o botão direito no pacote **<<ubispecification>>Especificacao** e selecione a opção **Tool →*UbiSpecification***

Lista de Tarefas da Avaliação

1. Descrever os requisitos de ubiquidade, de acordo com o treinamento recebido, para o cenário de ubiquidade fornecido.
2. Gerar a especificação textual dos requisitos.

G.3 Questionário de Avaliação da Infraestrutura

Questionário de Avaliação da Infraestrutura

Nome: _____

*Em relação a sua percepção sobre a **facilidade de uso** da infraestrutura, qual o seu grau de concordância com as afirmações abaixo:*

1. Foi fácil aprender a utilizar a infraestrutura
( ) Concordo totalmente
( ) Concordo amplamente
( ) Concordo parcialmente
( ) Discordo parcialmente
( ) Discordo amplamente
( ) Discordo totalmente

2. Consegui utilizar a infraestrutura para realizar a especificação
( ) Concordo totalmente
( ) Concordo amplamente
( ) Concordo parcialmente
( ) Discordo parcialmente
( ) Discordo amplamente
( ) Discordo totalmente

3. A interação com a infraestrutura foi clara e compreensível
( ) Concordo totalmente
( ) Concordo amplamente
( ) Concordo parcialmente
( ) Discordo parcialmente
( ) Discordo amplamente
( ) Discordo totalmente

4. Seria fácil ganhar habilidade no uso da infraestrutura
( ) Concordo totalmente
( ) Concordo amplamente
( ) Concordo parcialmente
( ) Discordo parcialmente
( ) Discordo amplamente
( ) Discordo totalmente

5. Foi fácil compreender como realizar uma especificação de requisitos com o uso da infraestrutura
( ) Concordo totalmente
( ) Concordo amplamente
( ) Concordo parcialmente
( ) Discordo parcialmente
( ) Discordo amplamente
( ) Discordo totalmente

6. Considero a infraestrutura fácil de usar
( ) Concordo totalmente
( ) Concordo amplamente
( ) Concordo parcialmente
( ) Discordo parcialmente
( ) Discordo amplamente
( ) Discordo totalmente

*Em relação a sua percepção sobre a **utilidade** da infraestrutura, qual o seu grau de concordância com as afirmações abaixo:*

7. Usar a infraestrutura melhorou o meu desempenho durante a especificação de requisitos de ubiquidade
( ) Concordo totalmente

( ) Concordo amplamente
( ) Concordo parcialmente
( ) Discordo parcialmente
( ) Discordo amplamente
( ) Discordo totalmente

8. Usar a infraestrutura facilitou a especificação de requisitos de ubiquidade
( ) Concordo totalmente
( ) Concordo amplamente
( ) Concordo parcialmente
( ) Discordo parcialmente
( ) Discordo amplamente
( ) Discordo totalmente

9. Usar a infraestrutura melhorou minha produtividade na especificação de requisitos de ubiquidade
( ) Concordo totalmente
( ) Concordo amplamente
( ) Concordo parcialmente
( ) Discordo parcialmente
( ) Discordo amplamente
( ) Discordo totalmente

10. Eu considero a infraestrutura útil para especificar requisitos de ubiquidade
( ) Concordo totalmente
( ) Concordo amplamente
( ) Concordo parcialmente
( ) Discordo parcialmente
( ) Discordo amplamente
( ) Discordo totalmente

11. Eu considero que a infraestrutura melhora a efetividade da especificação de requisitos
( ) Concordo totalmente
( ) Concordo amplamente
( ) Concordo parcialmente
( ) Discordo parcialmente
( ) Discordo amplamente
( ) Discordo totalmente

Utilize o espaço abaixo para fazer comentários que considere relevantes a avaliação.

APÊNDICE H – Manual da Ferramenta

H.1 Instalação e Configuração

Para instalação da ferramenta desenvolvida são distribuídos 3 arquivos:

- Plugout.zip
- Template.zip
- BOUML.exe

A configuração do ambiente deve ser feita em 3 passos:

1. Instalação do BOUML: execute o arquivo BOUML.exe e siga as instruções do wizard.
2. Descompactação do plug-in: o arquivo plugout.zip deve ser descompactado no mesmo diretório onde foi instalado o BOUML.
3. Descompactação do projeto Template.zip: descompacte o arquivo template.zip em um diretório qualquer.

Depois desses passos, para cada projeto que desejamos utilizar a ferramenta devemos configurar o acesso do BOUML ao plug-in e aos perfis UML necessários. Infelizmente essas duas tarefas devem ser feita para cada projeto, porque o BOUML não leva essa configuração de um projeto para o outro. Visando facilitar a utilização da ferramenta é distribuído, juntamente com o plug-in, um projeto BOUML chamado Template, que já possui todas as configurações necessárias.

Dessa forma, para criar um novo projeto basta abrir o projeto Template e usar a opção Save as para criar efetivamente o projeto desejado.

Importante: como a ferramenta foi desenvolvida em Java, para que a sua execução seja bem sucedida, o JRE versão 1.6 ou superior deve estar instalado e disponível a partir de um diretório qualquer. Para verificar essa condição basta abrir uma janela do console e digitar “java –version”.

H.2 Caracterização do projeto

Uma das facilidades da ferramenta é a possibilidade de caracterização do projeto, que significa a escolha dos fatores que o projeto a ser especificado apresenta. Para caracterizar o projeto, basta acionar o menu de contexto no pacote <<*ubimodel*>> selecionar a opção Tool →*UbiProject*, conforme mostra a Figura H-1.

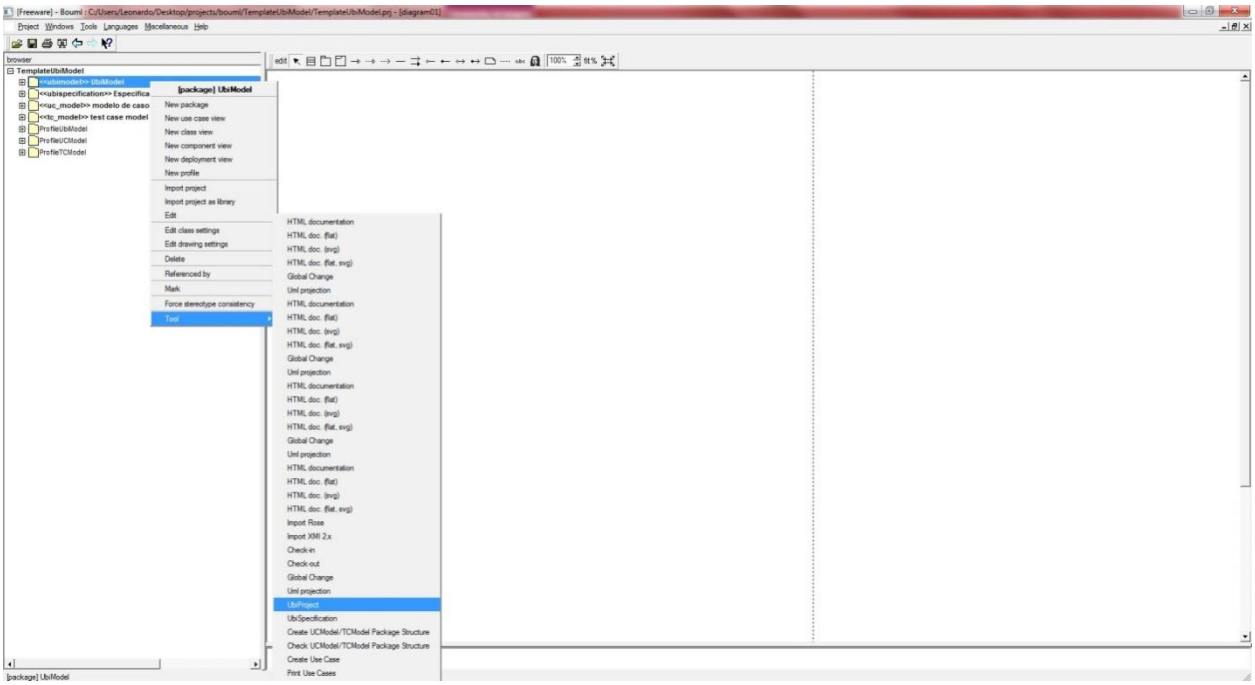


Figura H-1 –Acionamento da tela de caracterização do projeto

Na tela seguinte, uma lista de fatores selecionáveis para cada característica de ubiquidade é apresentada, conforme mostra a Figura H-2.

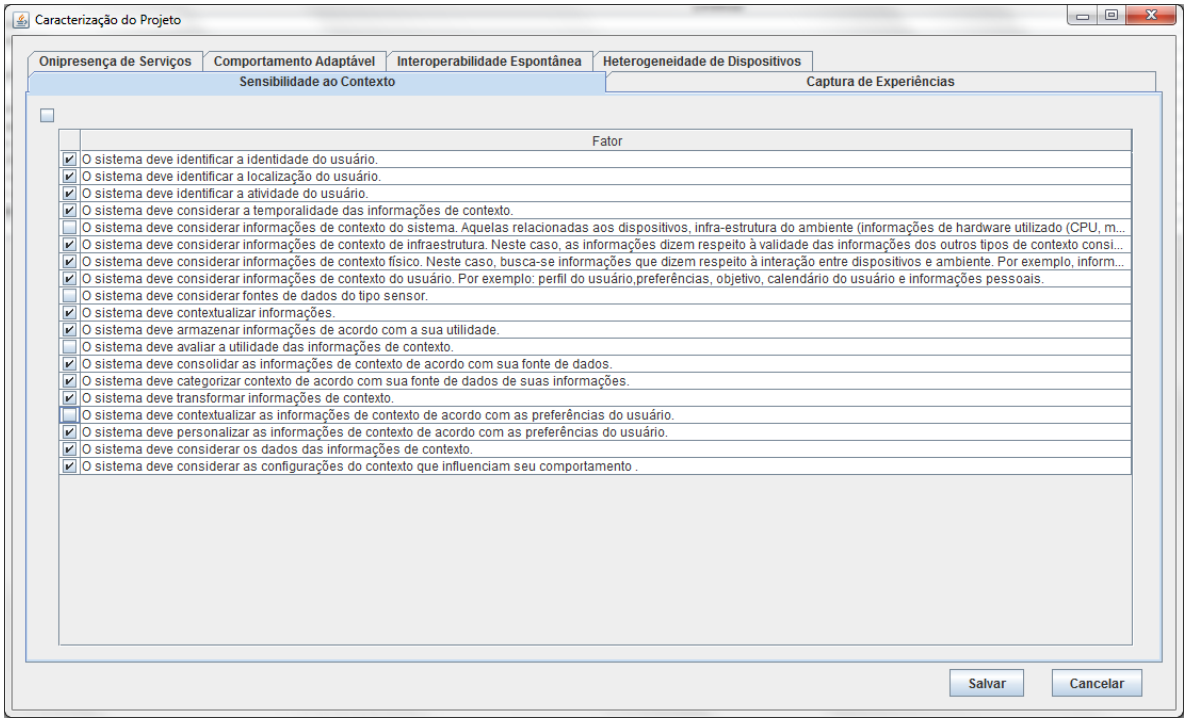


Figura H-2 –Tela de caracterização do projeto

Na tela apresentada na Figura H-2, os fatores de ubiquidade se encontram organizados em guias que contém os nomes das características de ubiquidade ao qual se relacionam. O analista pode navegar por essas guias e selecionar os fatores

presentes no projeto. Após selecioná-los, basta clicar em “Salvar”, as informações são salvas e a tela é fechada.

A partir desse momento, a caracterização do projeto está finalizada e, baseado nessa caracterização, a especificação do projeto pode ser realizada.

H.3 Especificação do projeto

Para especificar o projeto, basta acionar o menu de contexto no pacote <<ubispecification>> Tool →*UbiSpecification*. A tela inicial para a especificação do projeto (Figura H-3) possui uma lista com as diretivas a serem preenchidas. Essa lista pode variar de acordo com os fatores selecionados na caracterização do projeto, ou seja, dependendo dos fatores que forem selecionados na caracterização, pode-se obter uma lista distinta de diretivas a serem especificadas.

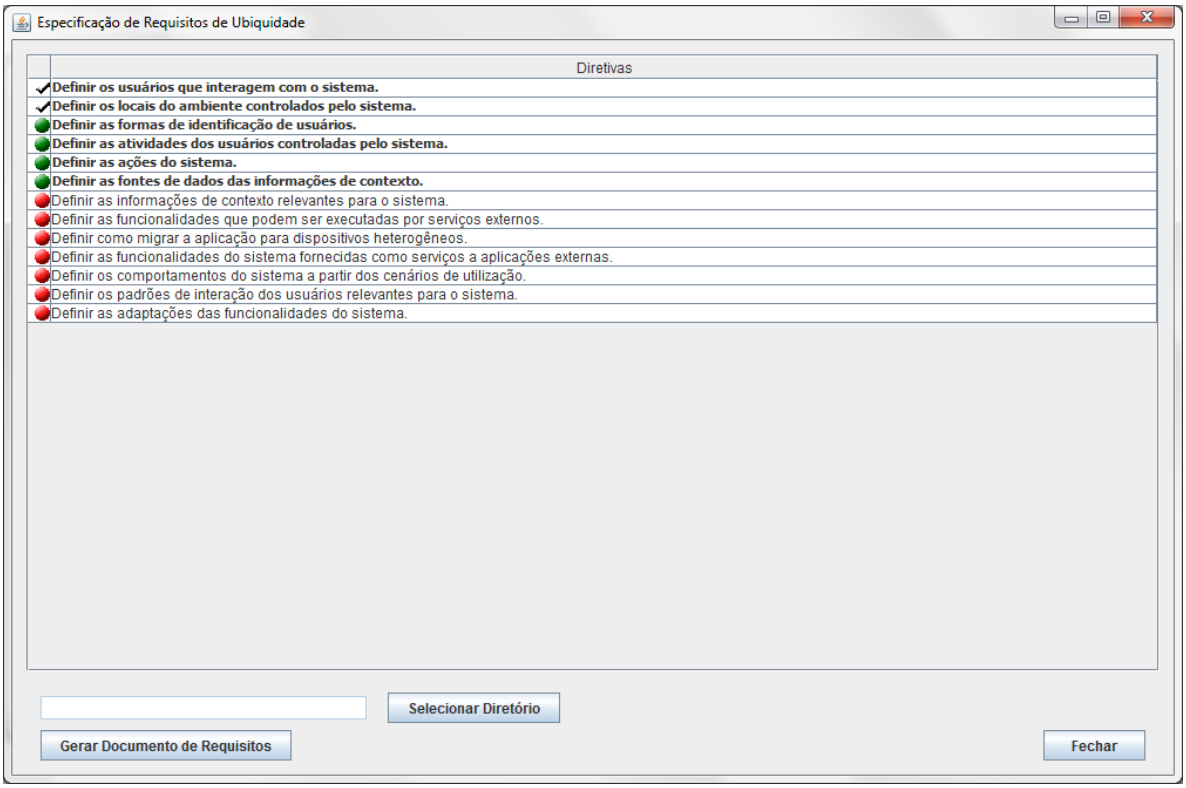


Figura H-3–Tela de Especificação de Requisitos de Ubiquidade

Ainda na Figura H-3 é possível observar que ao lado de cada diretiva existem ícones que indicam seu estado. O ícone verde  significa que a diretiva não possui pendências e está disponível para especificação, bastando para isso clicar sobre a diretiva e iniciar a especificação. Além disso, este ícone também indica que ainda não existem especificações realizadas para a diretiva em questão.

O ícone de check preto ✔ indica que requisitos para a diretiva já foram especificados, ou seja, já está preenchida. De qualquer forma, é possível especificar quantos requisitos mais forem necessários associados a essa diretiva.

O ícone vermelho 🔴 indica que a diretiva não está disponível para especificação, ou seja, que a mesma possui pendências, ou pré-condições que devem ser resolvidas antes de especificar tal diretiva. Por exemplo, ao especificar uma informação de contexto, através da diretiva “Definir o conjunto de informações de contexto que o sistema deve observar” é necessário que a diretiva relacionada a fonte de dados (“Definir as fontes de dados que fornecem informações do ambiente monitorado pelo sistema”) tenha sido preenchida anteriormente, isso porque o conceito de fonte de dados é utilizado na especificação de informações de contexto. Desta forma, para o exemplo em questão, enquanto não houver especificação de pelo menos uma fonte de dados, a diretiva de informação de contexto permanecerá com o ícone vermelho associado. Assim que for especificada uma fonte de dados, o ícone da diretiva de informação de contexto é automaticamente alterado de vermelho para verde, indicando que a mesma já pode ser especificada, já que sua pré-condição foi resolvida.

H.3.1 Definir Usuários

Objetivo: definir os usuários do sistema informando um nome que os identifique e uma descrição dos seus papéis no uso do sistema.

A definição de usuários pode ser realizada acionando a diretiva “Definir os usuários que interagem com o sistema”. Ao abrir a tela inicial, será apresentada uma listagem de usuários, juntamente com as opções para incluir, editar ou excluir um usuário.

Ao incluir um novo usuário, os campos Nome e Descrição devem ser especificados, conforme mostra a Figura H-4.

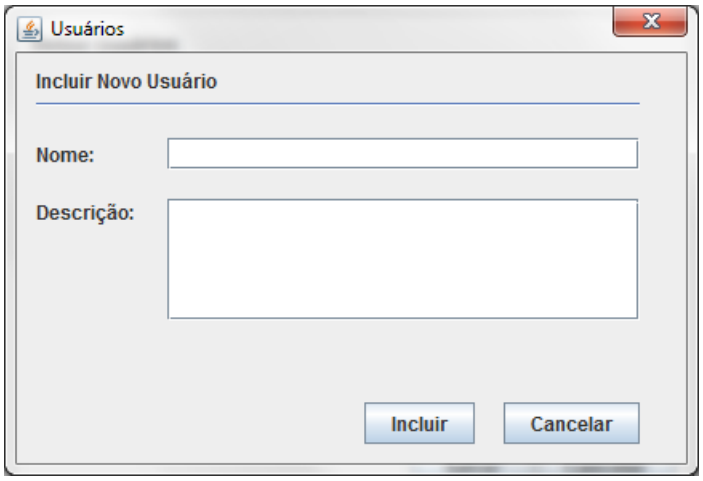


Figura H-4–Tela de Inclusão de Novo Usuário

H.3.2 Definir Localizações

Objetivo: definir os locais do ambiente controlados pelo sistema informando um nome que os identifique e uma descrição de suas características.

A definição de localizações pode ser realizada acionando a diretiva “Definir os locais do ambiente controlados pelo sistema”. Ao abrir a tela inicial, será apresentada uma listagem de localizações, juntamente com as opções para incluir, editar ou excluir uma localização.

Ao incluir uma nova localização, os campos Nome e Descrição devem ser especificados, conforme mostra a Figura H-5.

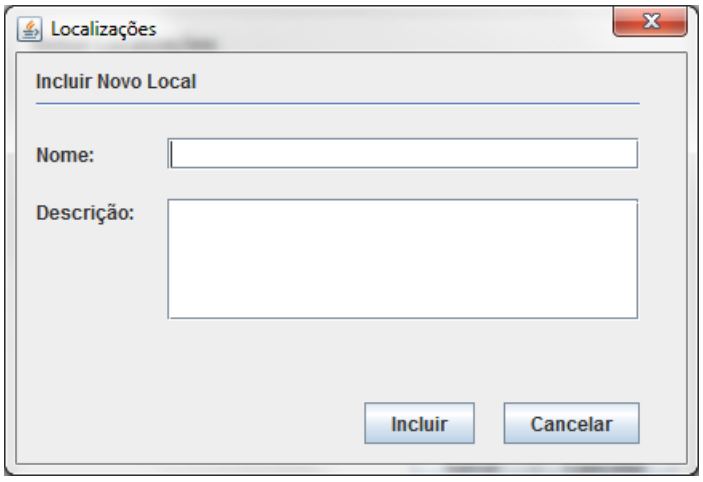


Figura H-5 – Tela de Inclusão de Novo Local

H.3.3 Definir Identificação

Objetivo: definir como os usuários são identificados pelo sistema. Para cada forma de identificação deve-se informar o conjunto de usuários e suas respectivas localizações onde aquela forma de identificação é válida.

A definição de identificação pode ser realizada acionando a diretiva “Definir as formas de identificação de usuários”. Ao abrir a tela inicial, será apresentada uma listagem de identificações, juntamente com as opções para incluir, editar ou excluir uma identificação.

Ao incluir uma nova identificação, os campos Nome e Caracterização devem ser especificados, assim como os usuários e suas respectivas localizações devem ser adicionados a lista de identificações conforme mostra a Figura H-6.

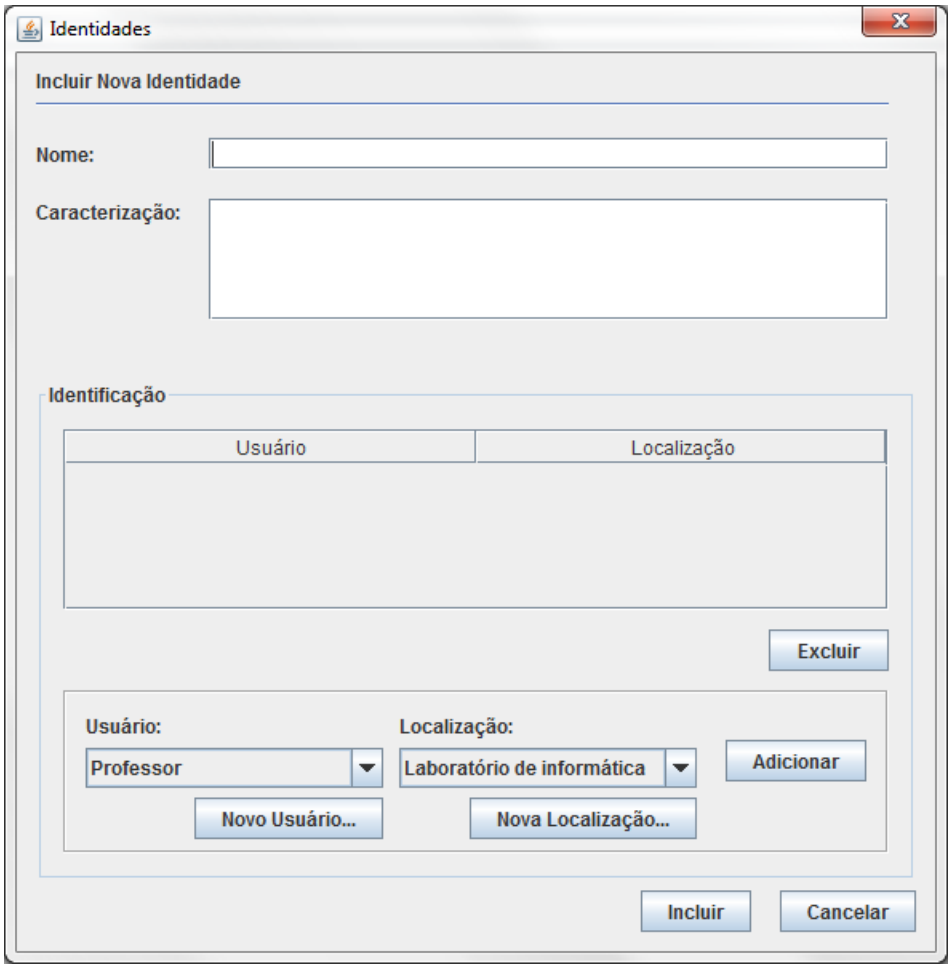


Figura H-6 – Tela de Inclusão de Nova Identificação

H.3.4 Definir Atividades

Objetivo: definir as atividades controladas pelo sistema informando um nome que as identifique e uma descrição do seu significado.

A definição de atividades pode ser realizada acionando a diretiva “Definir as atividades dos usuários controladas pelo sistema”. Ao abrir a tela inicial, será apresentada uma listagem de atividades, juntamente com as opções para incluir, editar ou excluir uma atividade.

Ao incluir uma nova atividade, os campos Nome e Descrição devem ser especificados, conforme mostra a Figura H-7.

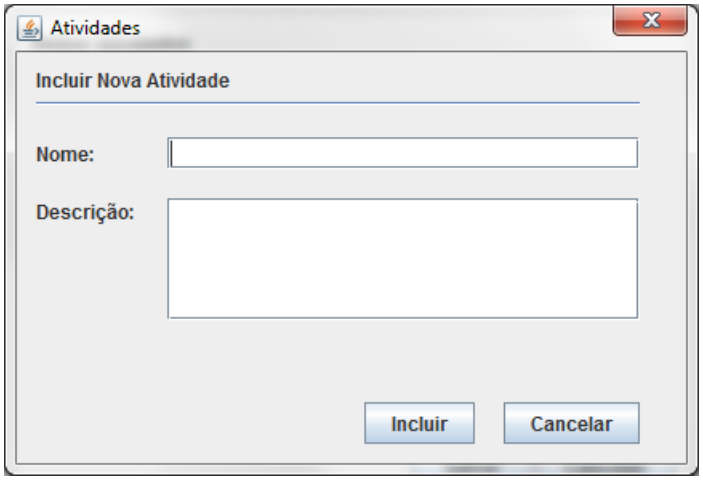


Figura H-7–Tela de Inclusão de Nova Atividade

H.3.5 Definir Ações do Sistema

Objetivo: definir as ações do sistema relacionadas à ubiquidade, ou seja, o que o sistema deve realizar. Devem ser informados os itens de especificação que descrevem em detalhes cada ação do sistema.

A definição de ações do sistema pode ser realizada acionando a diretiva “Definir as ações do sistema”. Ao abrir a tela inicial, será apresentada uma listagem de ações do sistema, juntamente com as opções para incluir, editar ou excluir uma ação do sistema.

Ao incluir uma nova ação do sistema, os campos Nome e Descrição devem ser especificados. Existe também a possibilidade de associar a ação a elementos do modelo de especificação *UCModel* na área “Item de Especificação” (Figura H-8).

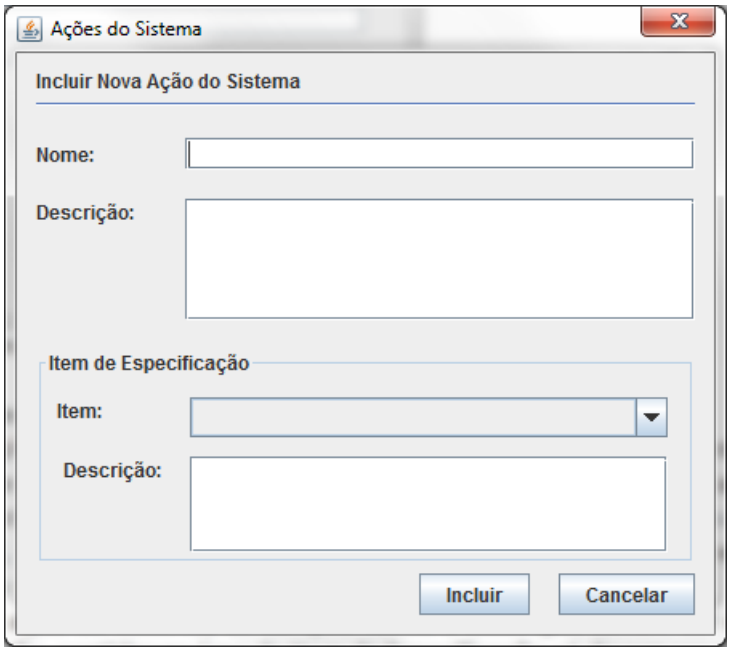


Figura H-8–Tela de Inclusão de Nova Ação do Sistema

H.3.6 Definir Fontes de Dados

Objetivo: definir as fontes a partir das quais as informações de contexto podem ser obtidas informando um nome que as identifique e se as mesmas são do tipo software ou dispositivo. Caso seja um dispositivo, informar se a mesma refere-se a um sensor. Por fim, informar uma descrição das principais características da fonte.

A definição de fontes de dados pode ser realizada acionando a diretiva “Definir as fontes de dados que fornecem as informações do ambiente monitorado pelo sistema”. Ao abrir a tela inicial, será apresentada uma listagem de fontes de dados, juntamente com as opções para incluir, editar ou excluir uma fonte de dados.

Ao incluir uma nova fonte de dados, os campos Nome e Descrição devem ser especificados, além disso, deve ser selecionado o tipo da fonte de dados, software ou dispositivo e caso seja dispositivo há, ainda, a possibilidade de indicar se o mesmo é sensor, conforme mostra a Figura H-9.

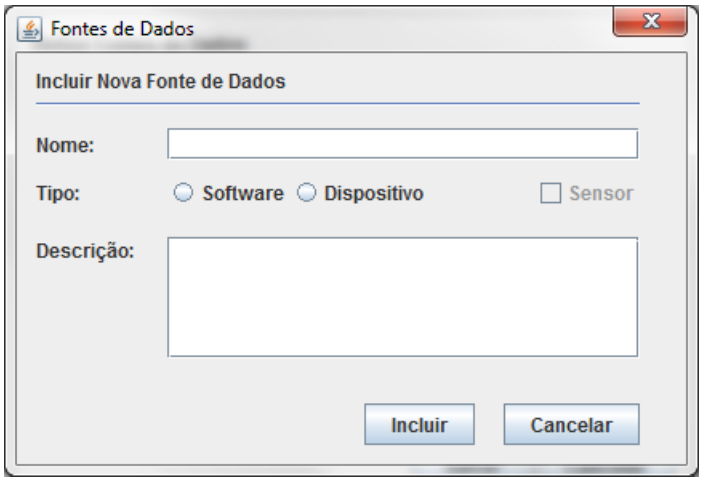


Figura H-9–Tela de Inclusão de Nova Fonte de Dados

H.3.7 Definir Informações de Contexto

Objetivo: definir as informações relacionadas aos elementos do ambiente relevantes para o sistema. As informações de contexto podem ser do tipo física quando estão relacionadas ao ambiente (temperatura, iluminação, ruído, etc.), podem ser informações de sistema quando estão relacionadas às restrições de infraestrutura (uso de memória, cpu, tamanho de tela, etc.), podem ser do tipo infraestrutura quando estão relacionadas ao correto funcionamento da infraestrutura (estado de funcionamento dos sensores, etc.) e, finalmente, podem ser do tipo usuário quando estão relacionadas aos usuários do sistema (temperatura corporal, pressão sanguínea, batimento cardíaco, etc.). Deve-se informar a fonte de dados ou a transformação que gera a informação de contexto. É importante também determinar a validade das informações em relação ao tempo, além do critério e do meio de armazenamento das informações.

Ao incluir uma nova informação de contexto, os seguintes campos devem ser especificados: Nome, Tipo de informação de contexto, Descrição, Temporalidade, Critério de Armazenamento, Meio de Armazenamento e Origem da informação de contexto. Neste último campo é possível escolher se a origem é uma fonte de dados, bastando selecionar uma fonte na lista ou até mesmo criar uma nova fonte ou criar uma nova transformação a partir de informações de contextos já existentes (Figura H-10).

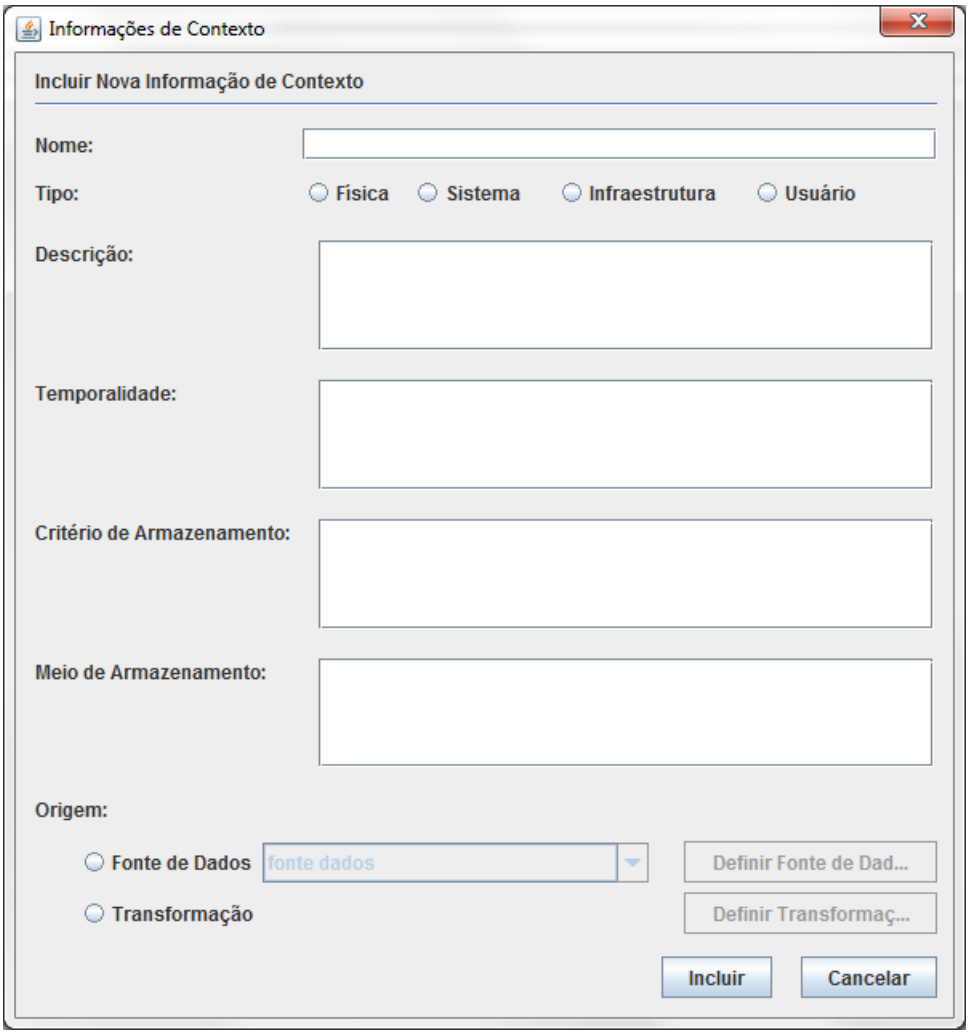


Figura H-10 – Tela de Inclusão de Nova Informação de Contexto

H.3.8 Definir Serviços Fornecidos

Objetivo: definir as ações do sistema que devem ser disponibilizadas como serviços a outras aplicações. Além de informações como nome e descrição dos serviços, é importante descrever o processo de inicialização dos mesmos, o que é feito antes dos serviços iniciarem, o que ocorre em suas finalizações, quais as condições mínimas para que estejam disponíveis e como são divulgados. Informações básicas sobre os serviços como protocolos e formato de dados também devem ser descritos.

A definição de serviços fornecidos pode ser realizada acionando a diretiva “Definir as funcionalidades do sistema fornecidas como serviços a aplicações externas”. Ao abrir a tela inicial, será apresentada uma listagem de serviços fornecidos, juntamente com as opções para incluir, editar ou excluir um serviço fornecido.

Ao incluir um novo serviço fornecido os campos Nome, Descrição, Preparação, Finalização e Divulgação devem ser preenchidos. Existe também a possibilidade de associar a ação a elementos do modelo de especificação *UCModel* na área “Item de Especificação” (Figura H-11). Nessa tela, ainda é possível especificar restrições associadas ao serviço, bastando selecionar uma informação de contexto, descrever a restrição na caixa de texto logo abaixo da lista e adicionar na tabela de restrições. Deve-se também especificar os protocolos (Figura H-12) e os formatos de dados (Figura H-13) do serviço.

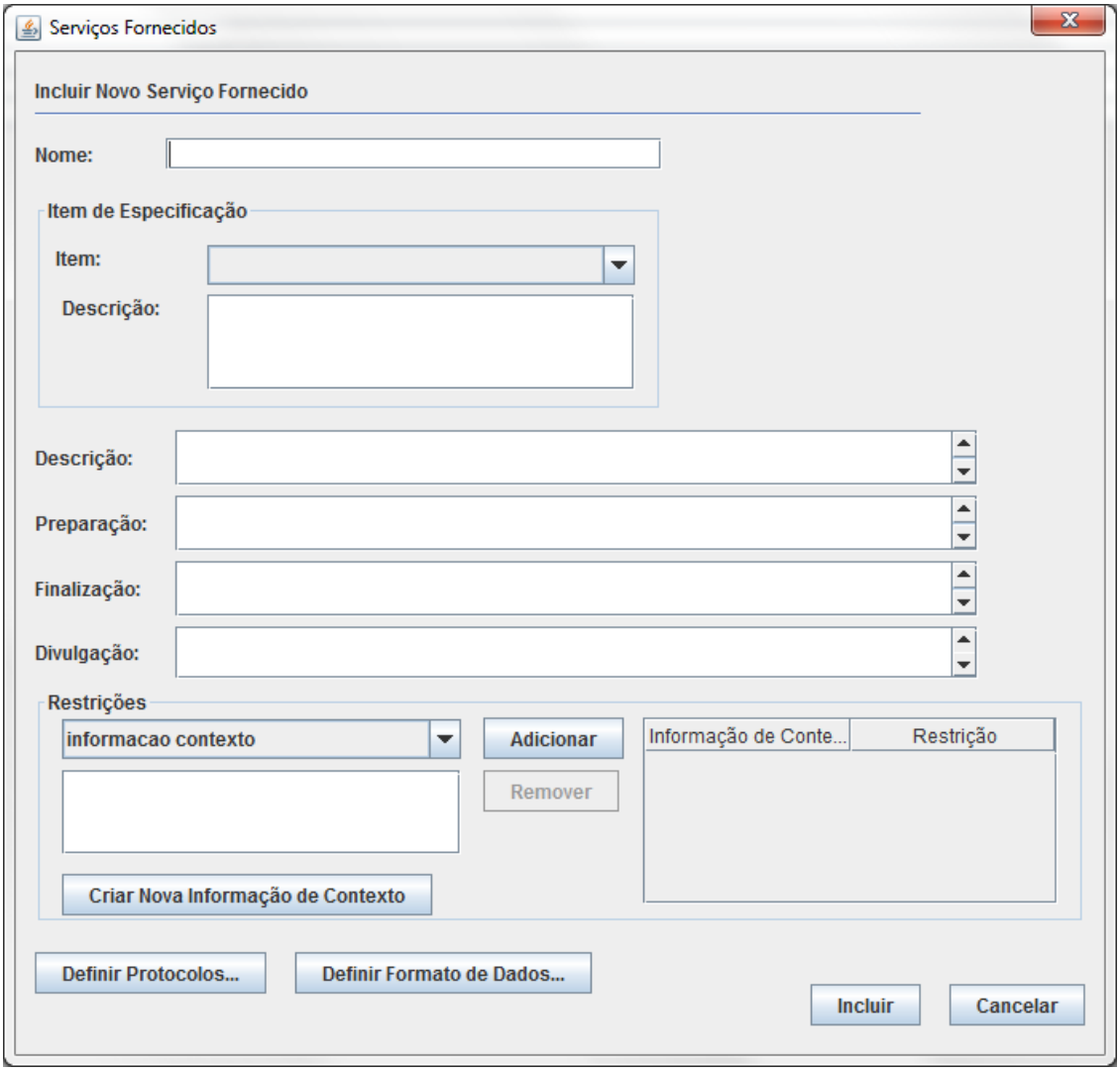


Figura H-11–Tela de Inclusão de Novo Serviço Fornecido

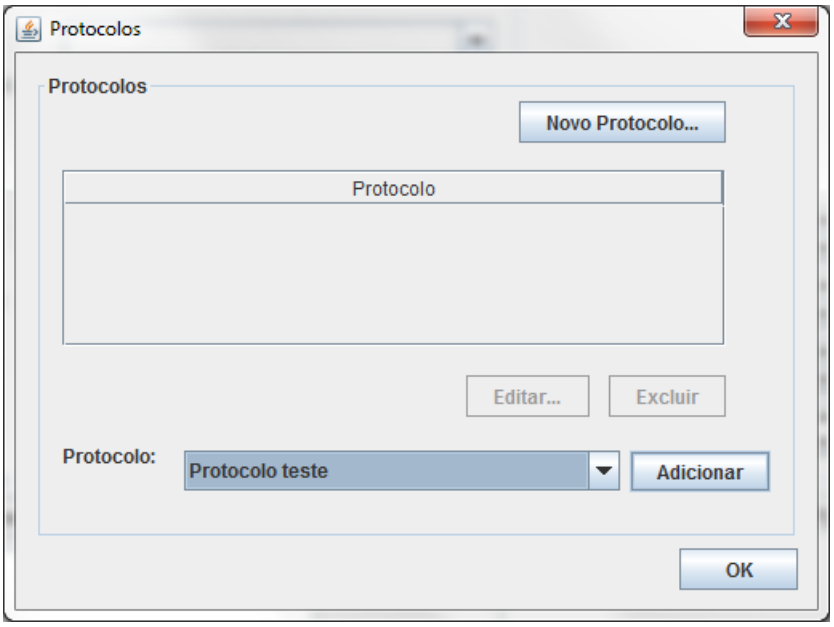


Figura H-12 – Tela de seleção de Protocolo

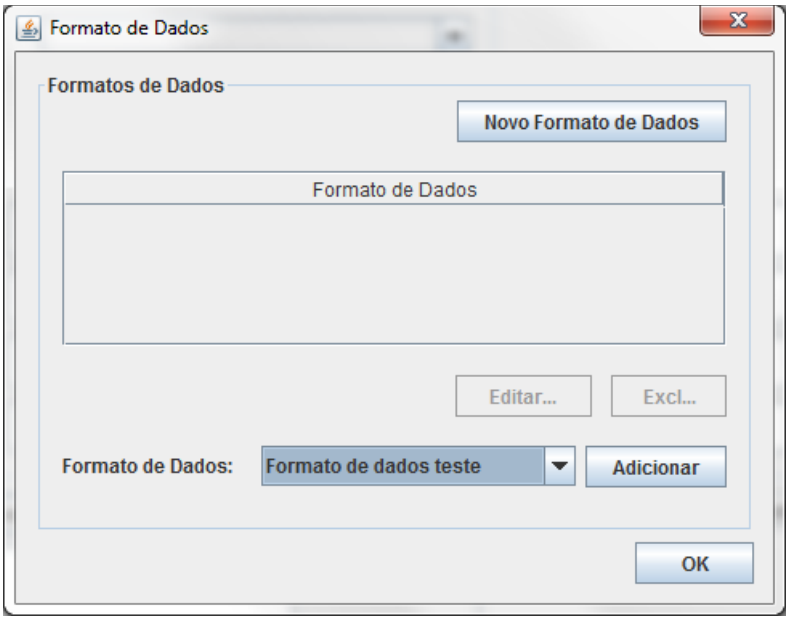


Figura H-13–Tela de seleção de formato de dados

H.3.9 Definir Comportamentos

Objetivo: definir o que o sistema deve fazer ao perceber determinados contextos. Um contexto é um cenário de utilização do sistema. Em outras palavras, na utilização do sistema, sempre haverá algum usuário participando de alguma atividade em algum lugar, sob determinadas condições do ambiente. Quando relevantes, deverão ser especificados para cada contexto quais são os usuários, atividades,

localizações e condições que o definem. A partir desses contextos o sistema pode se comportar de maneira distinta para cada um ou para cada conjunto de contextos.

A definição de comportamentos pode ser realizada acionando a diretiva “Definir os comportamentos do sistema em função dos cenários de utilização”. Ao abrir a tela inicial, será apresentada uma listagem de comportamentos, juntamente com as opções para incluir, editar ou excluir um comportamento.

Ao incluir um novo comportamento um Nome deve ser informado, além dos contextos envolvidos e ações que o sistema deve realizar ao perceber esses contextos. Existe também a possibilidade de associar a ação a elementos do modelo de especificação *UCModel* na área “Item de Especificação” (Figura H-14).

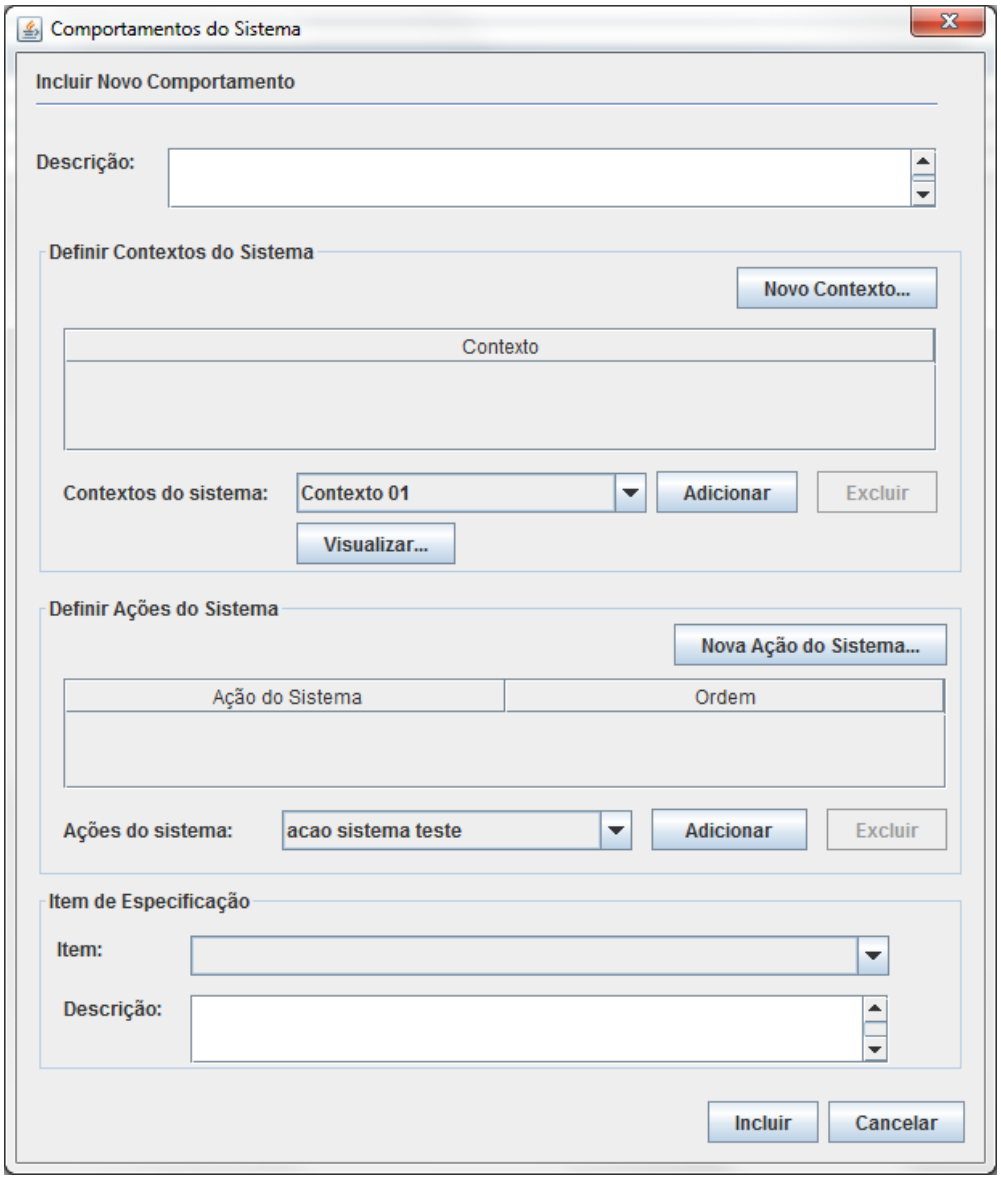


Figura H-14 – Tela de Inclusão de Novo Comportamento

É possível especificar contextos a partir da tela de inclusão de comportamentos. Na tela de criação de novo contexto é possível definir o Nome do Contexto, além de selecionar usuários, atividades e condições de contexto, conforme mostra a Figura H-15.

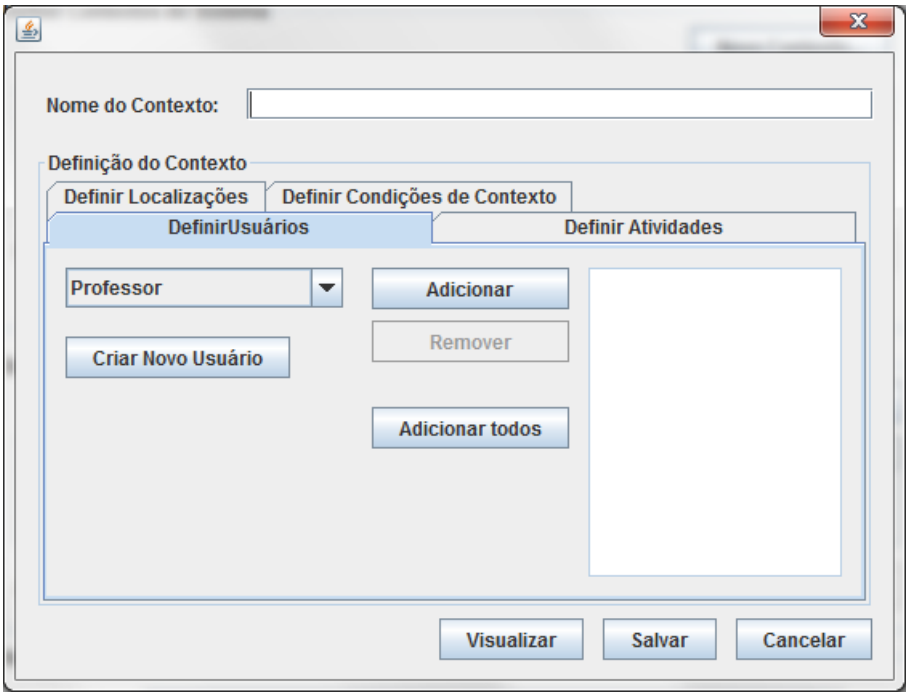


Figura H-15 – Tela de Inclusão de Novo Contexto

H.3.10 Definir Padrões de Interação

Objetivo: definir as sequências de interações que ocorrem de maneira recorrente, segundo critérios, que ao serem identificadas fazem com que o sistema realize ações em benefício do usuário. Interações são ações que o usuário realiza no ambiente controlado pelo sistema (acender a luz, ligar a TV, ligar o computador, etc.) e têm um conjunto de informações de contexto relacionadas. Além disso, elas ocorrem em determinados contextos de utilização do sistema. Na definição dos padrões é importante informar a ordem em que as interações devem ocorrer e os critérios utilizados para determinar a validade de um padrão (tempo decorrido entre as interações, tempo decorrido entre as sequências de interações, etc.). Na definição do que o sistema deve fazer na ocorrência de padrões é importante informar a ordem em que as ações devem ser realizadas.

A definição de padrões de interação pode ser realizada acionando a diretiva “Definir padrões de interação do usuário relevantes para o sistema”. Ao abrir a tela

inicial, será apresentada uma listagem de padrões de interação, juntamente com as opções para incluir, editar ou excluir um padrão de interação.

Ao incluir um novo padrão de interação, deve-se informar um Nome, uma ou mais interações que o sistema deve considerar, o Critério de Análise e as ações que o sistema deve tomar ao identificar o padrão de interação. Existe também a possibilidade de associar a ação a elementos do modelo de especificação *UCModel* na área “Item de Especificação” (Figura H-16).

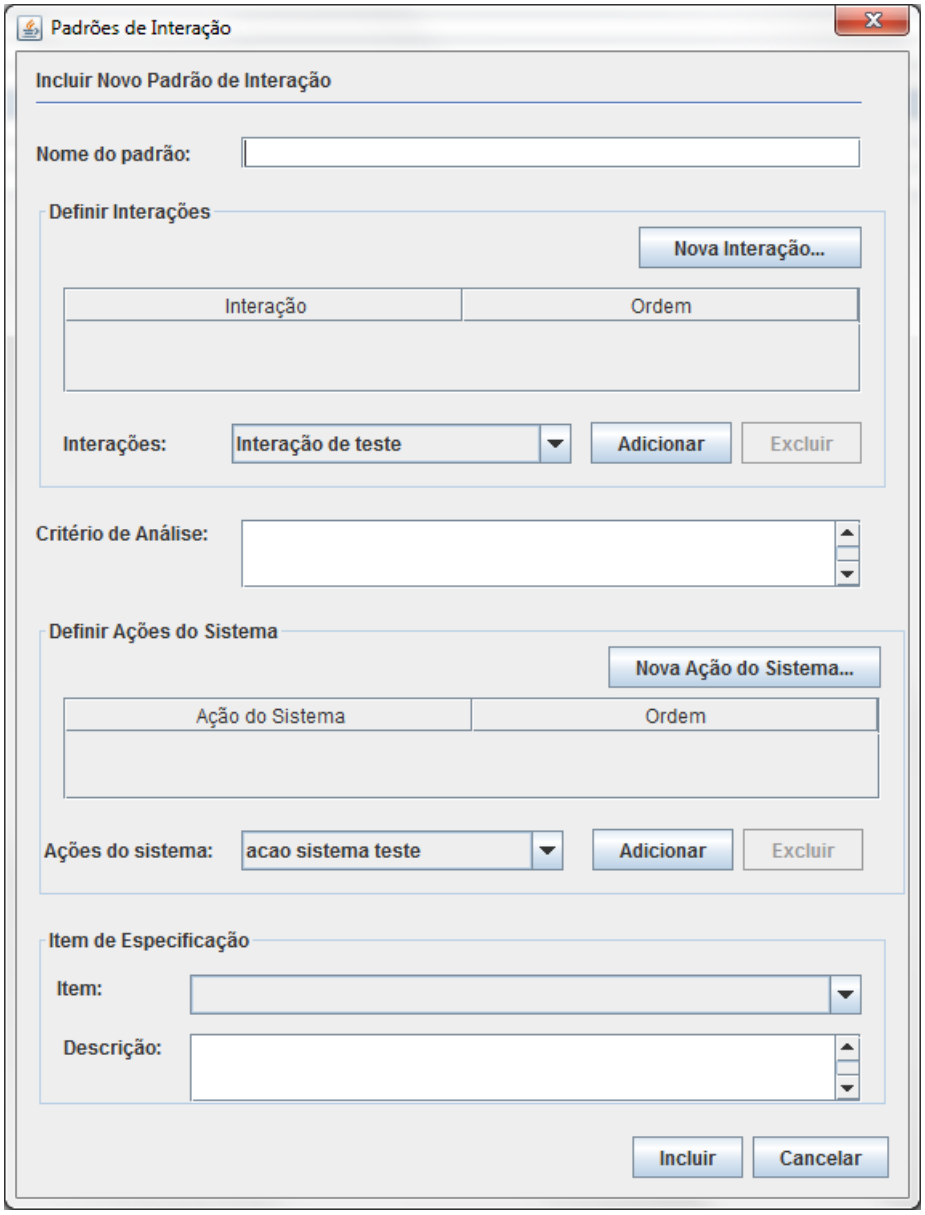


Figura H-16 – Tela de Inclusão de Novo Padrão de Interação

É possível especificar uma interação a partir da tela de inclusão de padrão de interação. Na tela de criação de interação é necessário especificar Nome da Interação, Descrição, Armazenamento, contexto ao qual está relacionada e uma lista de informações de contexto relevantes na interação, conforme mostra a Figura H-17.

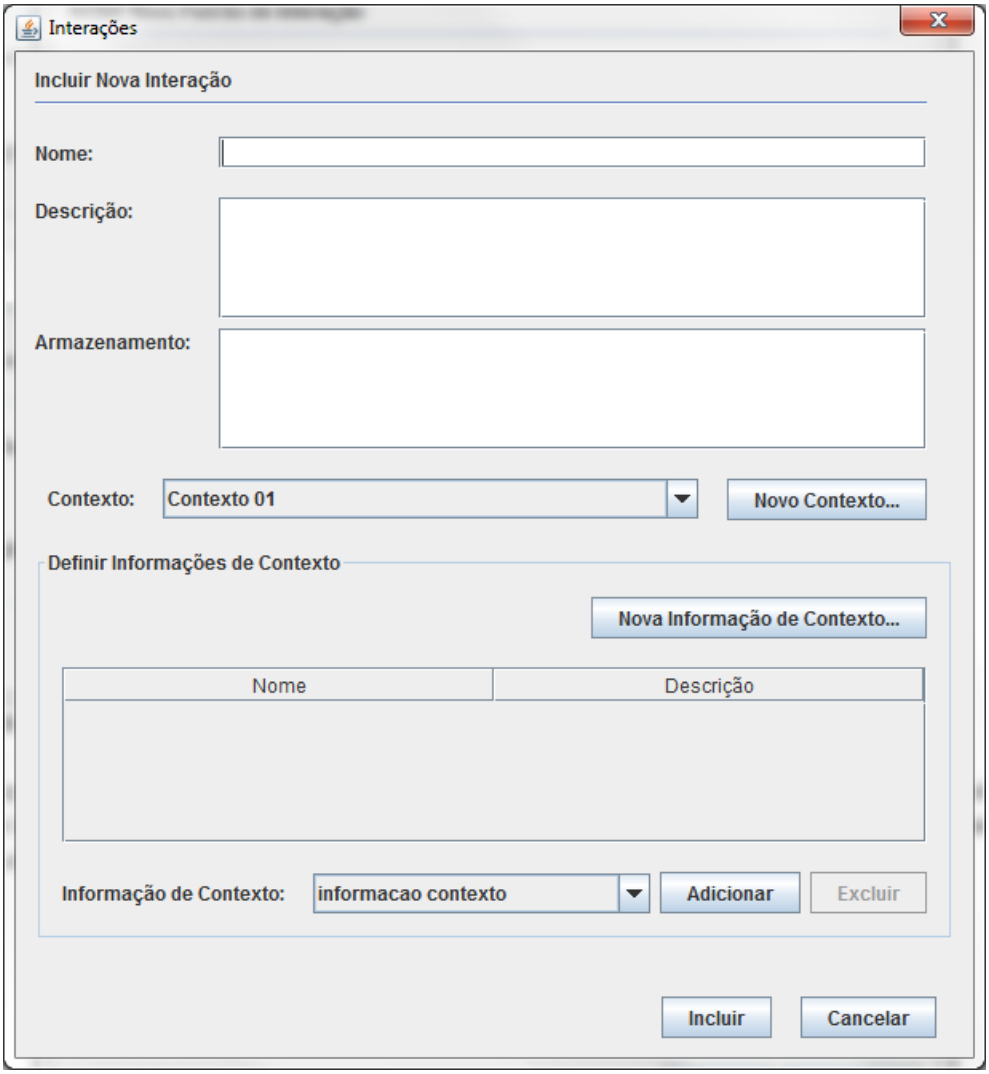


Figura H-17 – Tela de Inclusão de Nova Interação

H.3.11 Definir Adaptações do Sistema

Objetivo: definir as adaptações que o sistema deve realizar caso alguma preferência seja violada. Uma preferência é uma condição do sistema, definida em função de uma informação de contexto, que deve ser obedecida para determinados contextos. As adaptações são compostas por alternativas de adaptação que o sistema deve selecionar considerando seus impactos. É importante informar o tempo que o sistema deve aguardar para a efetivação de uma adaptação, já que o contexto no qual a preferência é violada pode se desfazer rapidamente, não havendo a necessidade da adaptação. Deve-se informar, também, a estratégia de seleção utilizada pelo sistema para selecionar uma alternativa de adaptação e como as informações sobre as adaptações realizadas devem ser armazenadas.

A definição de adaptações do sistema pode ser realizada acionando a diretiva “Definir as adaptações para as funcionalidades do sistema”. Ao abrir a tela inicial, será apresentada uma listagem de adaptações do sistema, juntamente com as opções para incluir, editar ou excluir uma adaptação do sistema.

Ao incluir uma nova adaptação, é necessário especificar sua Descrição, Estratégia de Seleção, Critério de Espera e Armazenamento, além de definir uma lista de preferências que será observada e as alternativas de adaptação (Figura H-18).

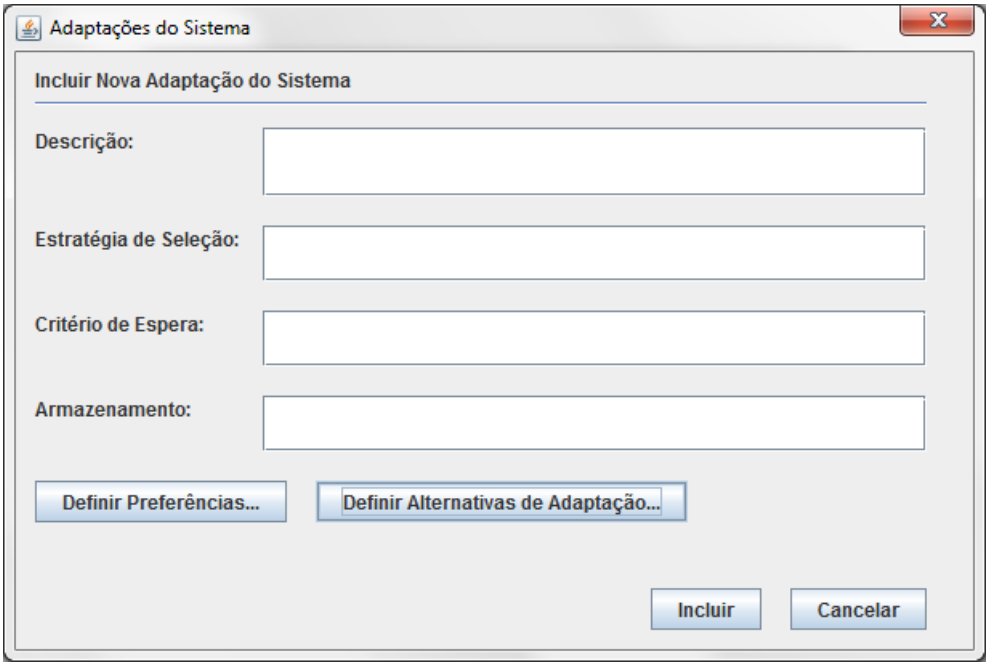


Figura H-18 – Tela de Inclusão de Nova Adaptação do Sistema

A partir da tela de inclusão de nova adaptação, é possível selecionar e criar preferências. Ao incluir uma nova preferência, é necessário informar o seu nome, além de restrições baseadas em informações de contextos e contextos que devem ser obedecidos para que a preferência seja estabelecida (Figura H-19).

Além da criação de preferências, é possível criar também alternativas de adaptação (Figura H-20) onde é possível escolher uma lista de alternativas de adaptação que contém a ação do sistema que deve ser realizada, bem como a descrição dessa alternativa de adaptação e seu impacto.

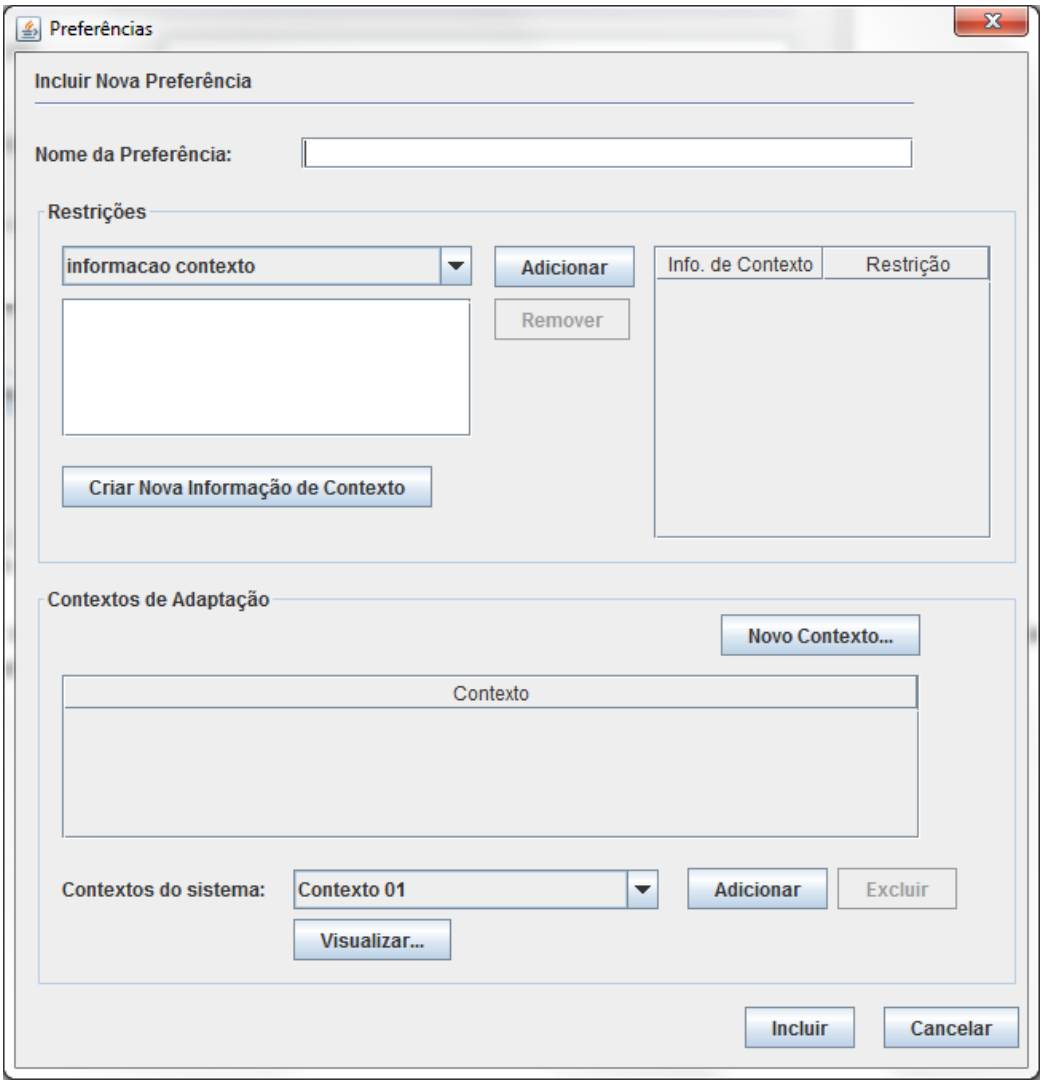


Figura H-19–Tela de Inclusão de Nova Preferência

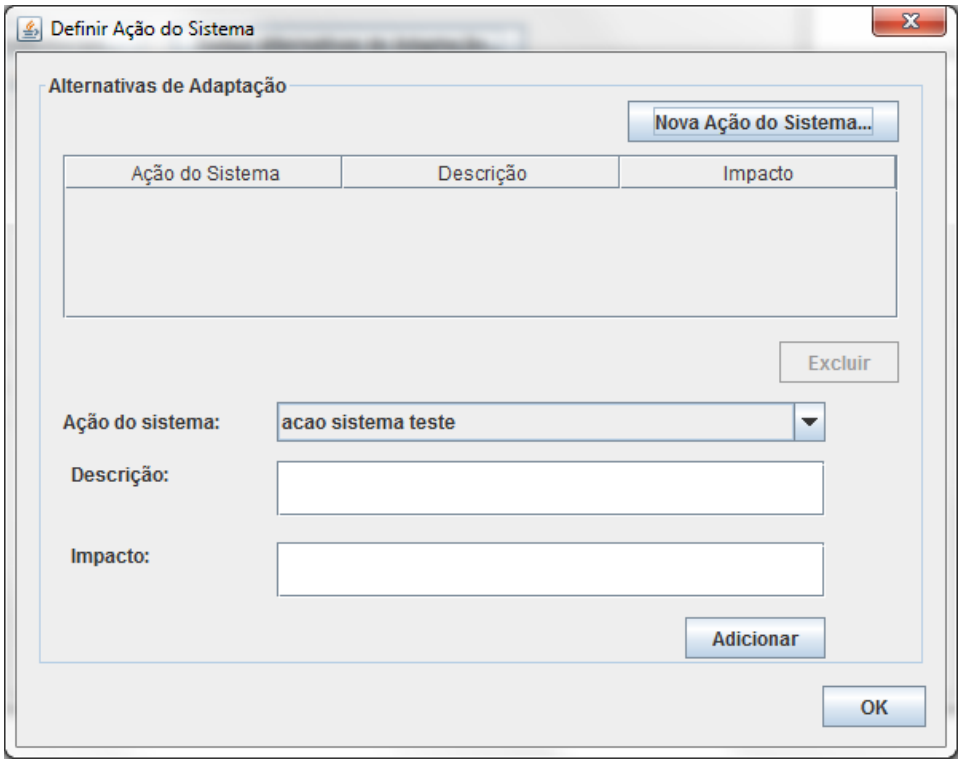


Figura H-20 –Tela de Inclusão de Alternativas de Adaptação

H.3.12 Definir Serviços Requeridos

Objetivo: definir os serviços externos que o sistema deve buscar e utilizar para atender determinadas necessidades que ele não é capaz de atender, pela não realização das ações ou pela inexistência no ambiente de serviços que realizem as ações de maneira a trazer algum ganho em relação a forma como elas são realizadas no momento. Ambos os cenários são descritos por requisitos mínimos que os serviços devem apresentar para que sejam efetivamente utilizados. Além de informações de nome e descrição dos serviços, é importante descrever como as informações relevantes durante a utilização dos serviços serão armazenadas e as estratégias utilizadas pelo sistema para buscar e selecionar os serviços. Devem ser informadas quais ações do sistema os serviços a serem utilizados devem atender. Informações básicas sobre os serviços como protocolos e formato de dados também devem ser descritos.

A definição de serviços requeridos pode ser realizada acionando a diretiva “Definir as funcionalidades do sistema que podem ser executadas por serviços externos”. Ao abrir a tela inicial, será apresentada uma listagem de serviços requeridos, juntamente com as opções para incluir, editar ou excluir um serviço requerido.

Ao incluir um novo serviço requerido, é necessário informar seu Nome, Descrição e Armazenamento. Além disso, é necessário informar qual Ação do Sistema está associada ao serviço requerido, além da Estratégia de Busca e Estratégia de Seleção do serviço externo. É necessário informar também os protocolos, formatos de dados e Requisitos do serviço a ser executado externamente (Figura H-21).

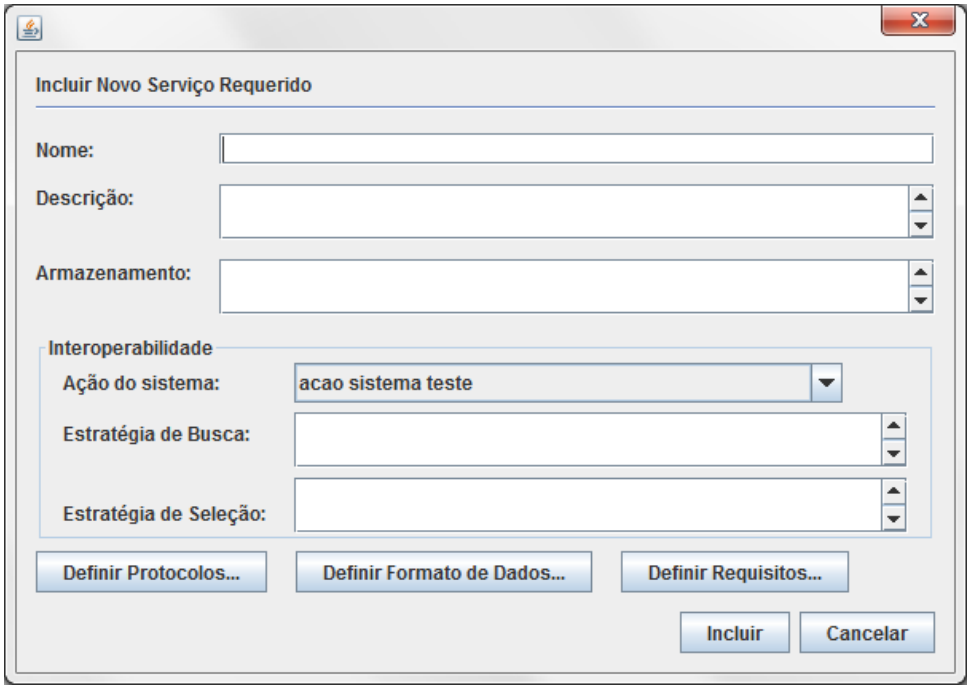


Figura H-21 – Tela de Inclusão de Novo Serviço Requerido

As telas de Definição de Protocolos (Figura H-12) e Definição de Formato de Dados (Figura H-13) já foram mostradas anteriormente. A seguir é apresentada a Tela de Definição de Requisito (Figura H-22), que também pode ser acionada na tela de inclusão de novo serviço requerido.

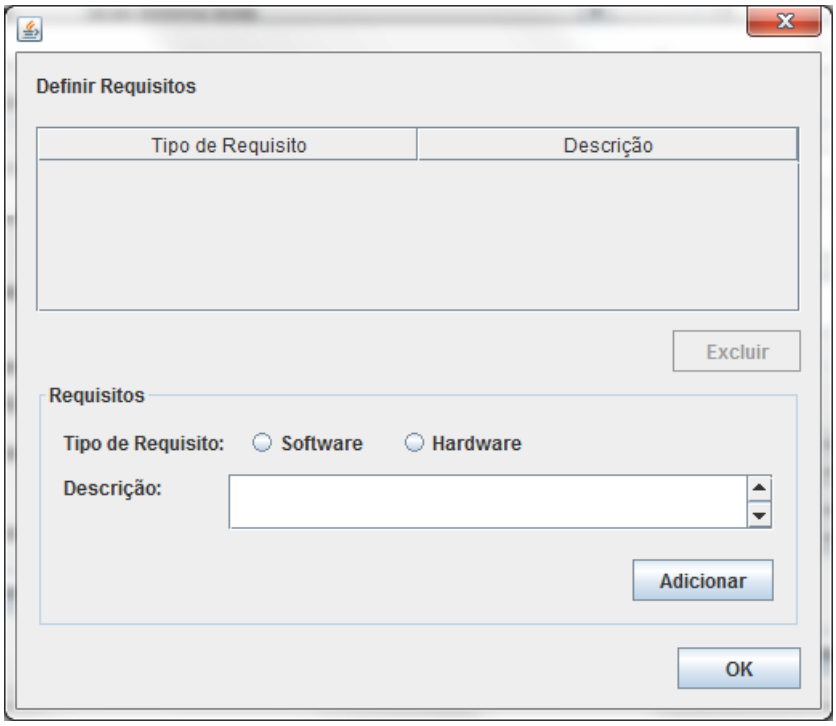


Figura H-22 – Tela de Definição de Requisito

H.3.13 Definir Migrações

Objetivo: definir as informações necessárias para que a aplicação possa migrar para outros dispositivos diferentes do utilizado originalmente pelo sistema. É importante descrever os requisitos que os dispositivos devem atender e as adaptações que devem ocorrer para que a migração seja possível. Existem 2 tipos de migração. No primeiro o usuário é responsável por iniciar o sistema via dispositivo heterogêneo e, a partir daí, o sistema se adapta de acordo com as características do dispositivo. A esse tipo de migração foi dado o nome de Migração Estática. O segundo tipo, denominado Migração Dinâmica, caracteriza-se por ser iniciado pelo próprio sistema, a partir da ocorrência de algum evento (usuário bloquear o computador, fechar a tampa do notebook, etc.). Após o evento ser percebido, o sistema busca e seleciona dispositivos no ambiente de acordo com estratégias e critérios de disponibilidade dos dispositivos que devem estar presentes na especificação. Na Migração Dinâmica, diferentemente da Migração Estática, os requisitos são organizados por grupos de dispositivos, ou seja, o sistema pode se adaptar de diferentes maneiras para cada grupo. Na Migração Estática o conjunto de requisitos é único, pois existe apenas um conjunto de adaptações do sistema, sendo portanto desnecessária a definição de grupos de dispositivos para esse tipo de migração.

A definição de migrações pode ser realizada acionando a diretiva “Definir migrações”. Ao abrir a tela inicial, será apresentada uma listagem de migrações, juntamente com as opções para incluir, editar ou excluir uma migração.

Ao incluir uma nova migração, é necessário selecionar se a migração é estática ou dinâmica. Na migração estática, é necessário especificar a Descrição da Migração, além de especificar uma ou mais adaptações, onde o requisito deve ser descrito, o tipo de requisitos deve especificado, ações do sistema devem ser associadas e a descrição da adaptação deve ser realizada (Figura H-23).

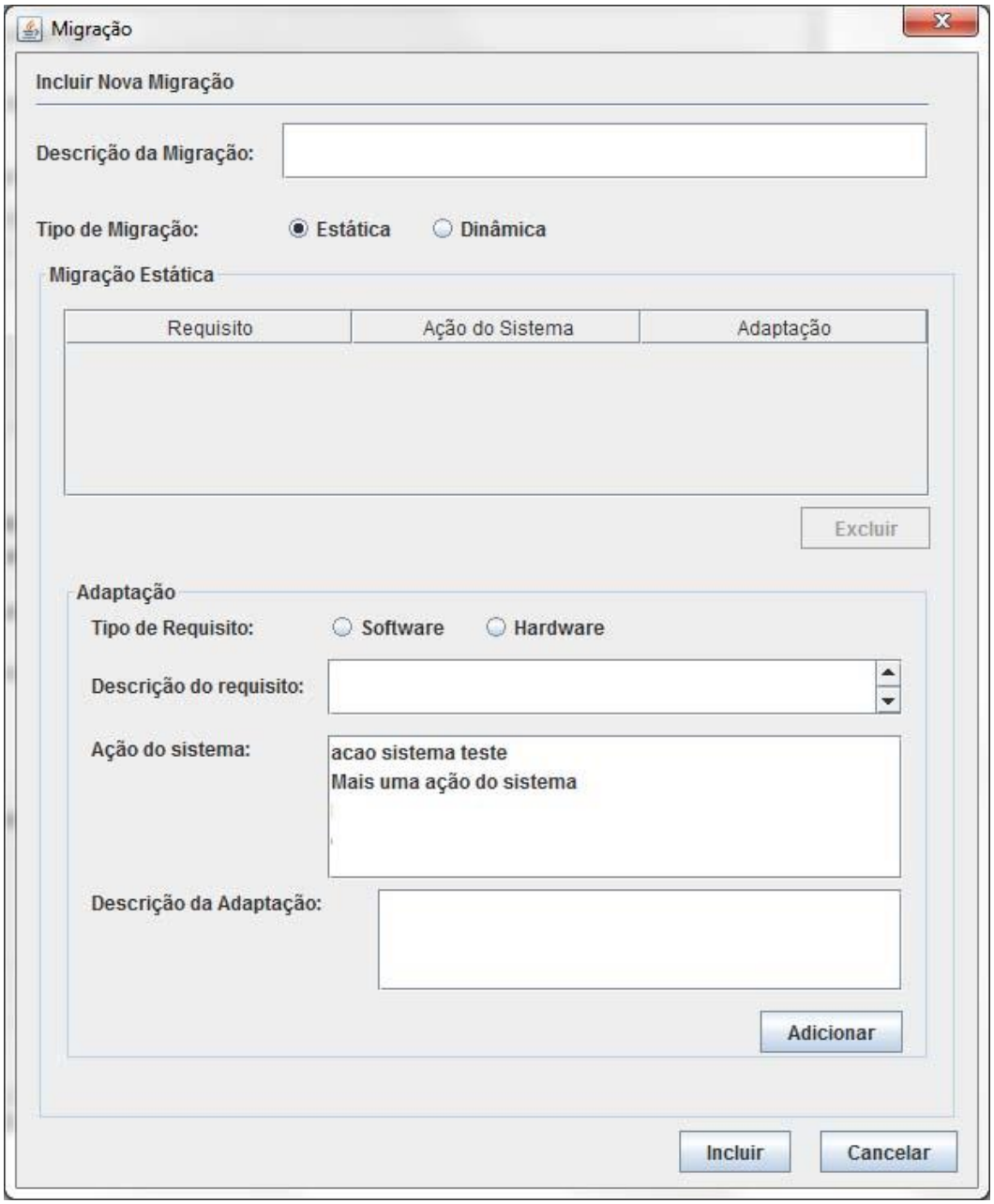


Figura H-23 – Tela de Inclusão de Nova Migração Estática

Ao selecionar a migração dinâmica, é necessário especificar a Descrição da Migração, a Estratégia de Busca e Estratégia de Seleção de dispositivos, além do grupo de dispositivos que a migração é compatível e a disponibilidade dos recursos para a migração (Figura H-24).

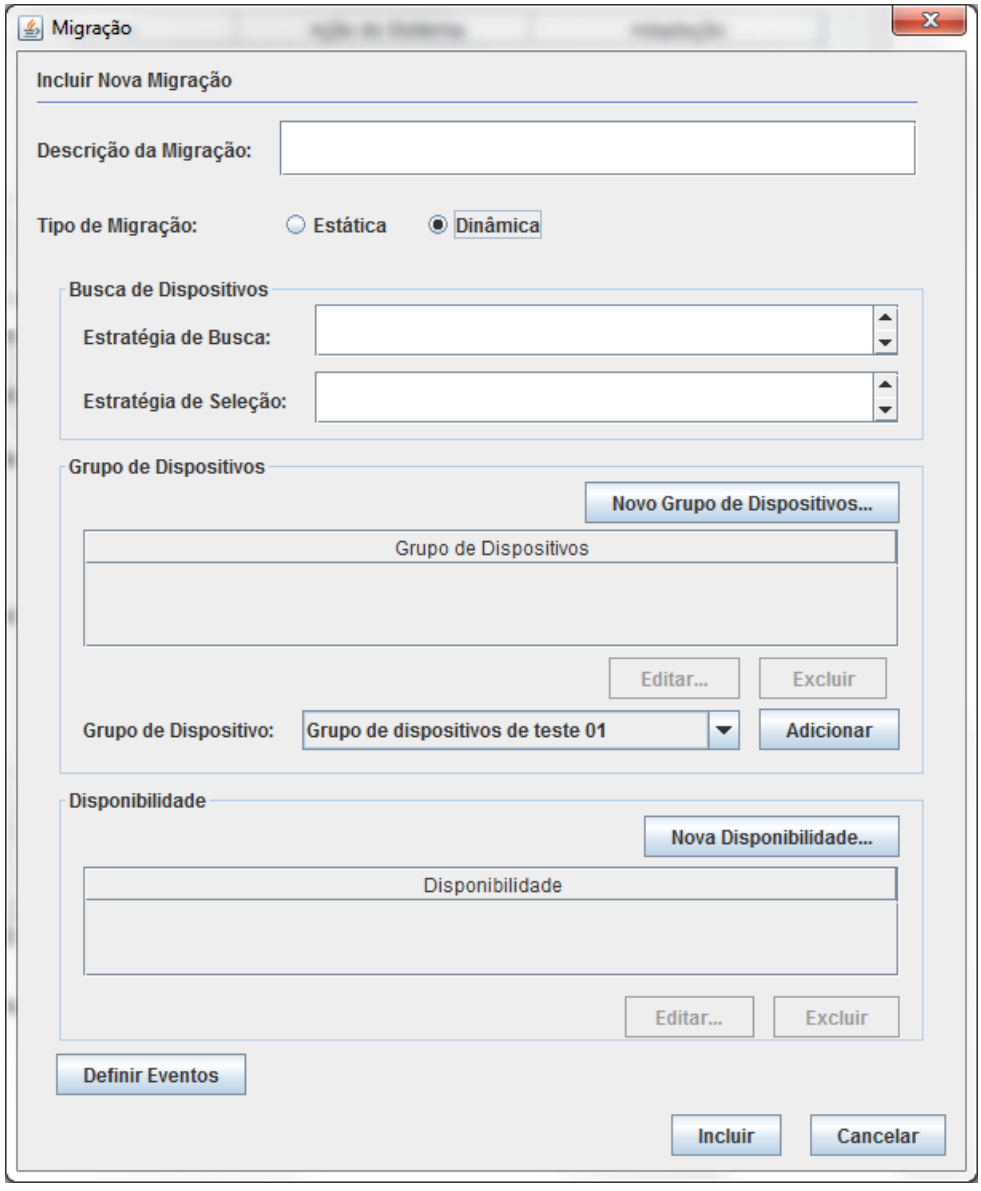


Figura H-24 – Tela de Inclusão de Nova Migração Dinâmica

Na própria tela de inclusão de migração dinâmica, é possível criar um novo grupo de dispositivos (Figura H-25) em que é necessário especificar um nome para o grupo de dispositivos, além de especificar uma ou mais adaptações, onde o requisito deve ser descrito, o tipo de requisitos deve especificado, ações do sistema devem ser associadas e a descrição da adaptação deve ser realizada(Figura H-25).

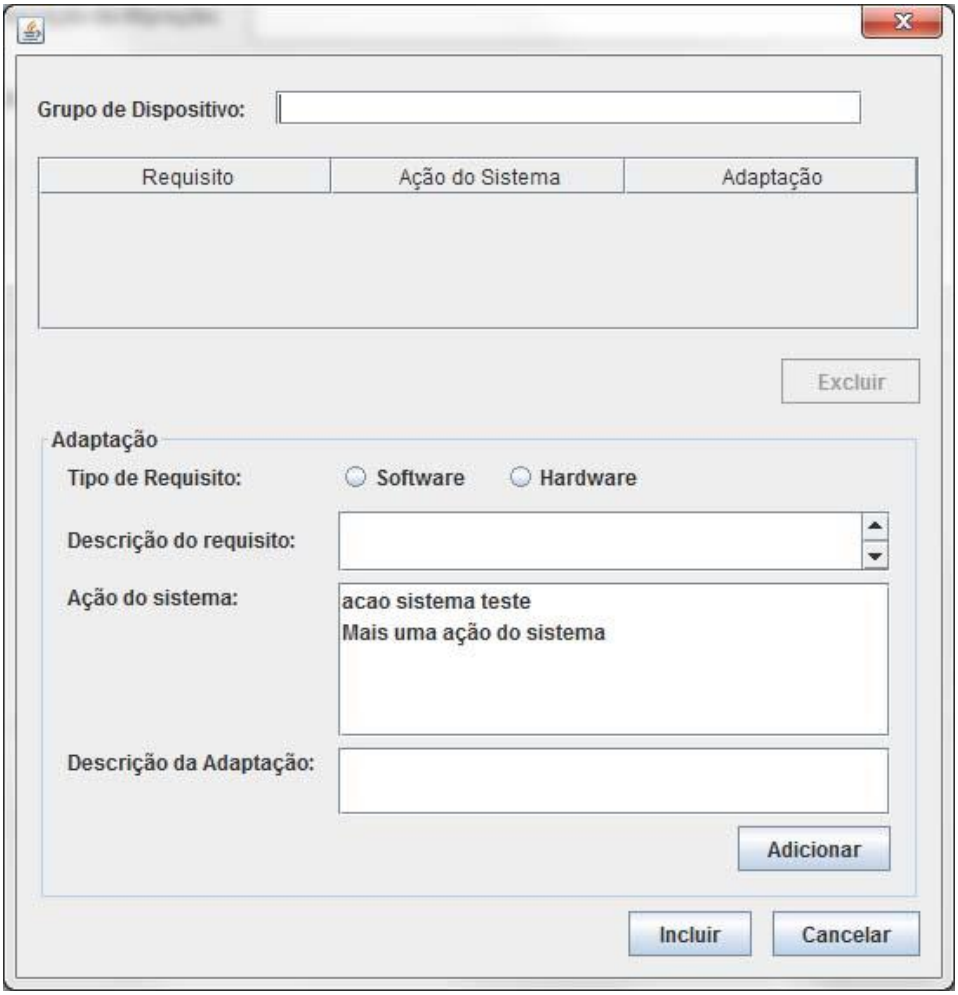


Figura H-25 – Tela de Inclusão de Grupo de Dispositivos

H.4 Geração do Documento de Requisitos

A geração do documento de requisitos pode ser feita após a especificação dos requisitos do sistema. Para isso, na tela principal, basta indicar o caminho no qual o documento será salvo e acionar o botão para a geração do documento de requisitos, conforme mostra a Figura H-26.

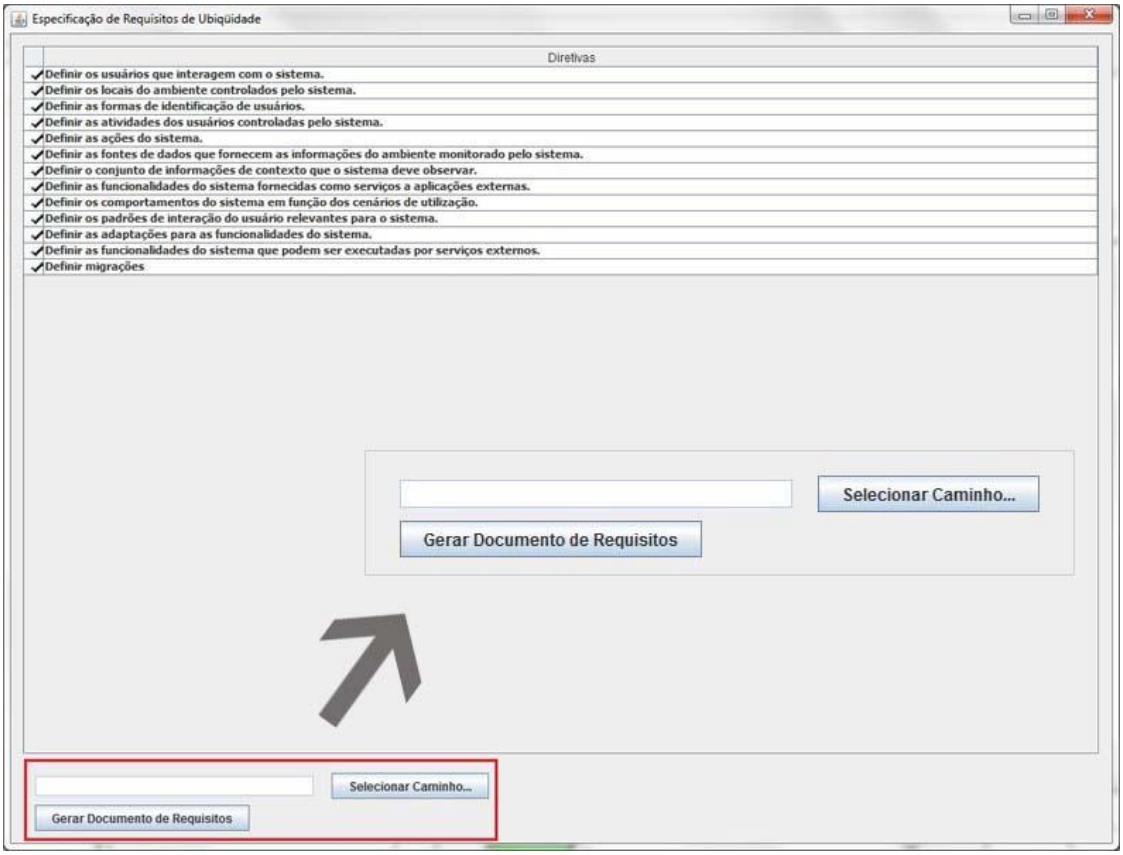


Figura H-26 – Detalhe da geração do documento de requisitos