Usando Modelos Para Apoiar a Especificação e Verificação de Requisitos de Ubiquidade

Leonardo Mota, Jobson Massollar, Guilherme Horta Travassos

Federal University of Rio de Janeiro/COPPE/PESC
Caixa Postal 68.511, CEP 21945-970, Rio de Janeiro - Brasil
{lsm, jobson, ght}@cos.ufrj.br

Abstract. A computação ubíqua é caracterizada como um novo paradigma onde o poder de processamento está disponível de forma onipresente e imperceptível no ambiente do usuário. Explorar este paradigma em projetos de software permite tratar soluções para problemas até então inviáveis devido a suas características gerais de utilização e acesso. Assim, essa pesquisa visa evoluir uma abordagem de apoio à definição e verificação de requisitos de ubiquidade ao definir estruturas (através de um metamodelo chamado *UbiModel*) para guiar a geração da especificação. Nesse contexto foi desenvolvida uma ferramenta que usa *UbiModel* para apoiar a descrição dos requisitos de ubiquidade de forma mais robusta. Considerando que os requisitos são usualmente definidos sob o ponto de vista das demandas e necessidades do usuário, a ferramenta permite a integração das características de ubiquidade com o ponto de vista do comportamento esperado do sistema, propiciando a descrição dos requisitos de forma abrangente.

Keywords. Requisitos, Ubiquidade Computacional, Ferramenta CASE.

1 Introdução

Atualmente, sistemas de software são importantes ativos estratégicos para as organizações e fazem parte do dia a dia dos indivíduos de forma cada vez mais intensa, tornando fundamental que o seu funcionamento esteja de acordo com os requisitos estabelecidos. Nesse contexto, torna-se cada vez mais essencial identificar, compreender, documentar e avaliar os requisitos que um sistema em particular vai apoiar [1]. Os requisitos possuem papel central no desenvolvimento de software, pois são base para estimativas, modelagem, projeto, implementação e testes, estando presentes ao longo de todo o ciclo de desenvolvimento do sistema. Dada a importância dos requisitos, atividades de controle da qualidade devem ser realizadas para verificar, validar e garantir a qualidade dos requisitos. De acordo com [2], 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 permitir sua propagação para fases posteriores do desenvolvimento, o que

aumentaria o custo de correção, retrabalho e os riscos de falha do software. Ou seja, quanto mais cedo os defeitos são corrigidos ou minimizados, menores serão os custos de sua correção.

Na construção de softwares ubíquos existem características específicas que normalmente não são consideradas pelas abordagens de desenvolvimento de software tradicionais disponíveis no corpo de conhecimento da engenharia de software, tais como sensibilidade ao contexto e heterogeneidade de dispositivos. Desta forma, ao utilizar estas tecnologias no desenvolvimento de software ubíquo em diferentes domínios de problema, é possível que sua eficiência e/ou eficácia não atinjam o desempenho esperado [3]. Por isso, observa-se a importância de entender como características de ubiquidade podem influenciar no desenvolvimento do software e como podem ser consideradas quando aplicando as abordagens correntes de apoio ao desenvolvimento. Atualmente, ainda é possível observar a existência de poucas investigações sobre como a engenharia de software pode apoiar o desenvolvimento de software ubíquo, o que aumenta a dificuldade e os riscos associados ao desenvolvimento desta categoria de software [4].

A partir desse cenário, Spínola [4] apresenta uma abordagem para apoiar a definição de requisitos de ubiquidade em conjunto com preocupações relacionadas à garantia da qualidade da especificação. Esta abordagem foi elaborada a partir de um conjunto de 6 características e 92 fatores da computação ubíqua [5], com a finalidade de se obter uma especificação de software mais robusta ao considerar as especificidades desse domínio frente a ubiquidade computacional.

2 Objetivos da pesquisa

O trabalho de Spínola [4] apresenta um conjunto baseado em evidência de características e fatores de ubiquidade e suas respectivas definições, obtido a partir de revisões sistemáticas da literatura [5] e de pesquisas de opinião realizadas com especialistas nesse domínio. A partir da construção desse corpo de conhecimento, Spínola [4] propôs um *checklist*, denominado *UbiCheck* [6], que tem como objetivo apoiar e guiar através de um conjunto de perguntas a captura e a elaboração da especificação de requisitos de ubiquidade. Estudos experimentais realizados indicaram que *UbiCheck* ajuda o engenheiro de software a direcionar sua atenção para as informações importantes na definição dos requisitos de ubiquidade. Entretanto, por se tratar simplesmente de uma lista de perguntas abertas e não de um modelo formalizado, sua estrutura não permite a exploração de cenários mais proeminentes, principalmente aqueles relacionados à organização automatizada de apoio adequado à especificação e avaliação da qualidade das especificações de requisitos. Além disso, as perguntas presentes no guia são abertas e possuem um nível de abstração muito elevado, o que as torna demasiadamente interpretativas, ou seja, a definição dos requisitos é muito dependente da interpretação e experiência do engenheiro de software.

Nesse contexto, o objetivo dessa pesquisa é aprimorar a abordagem apresentada por Spínola [4] através da definição de um metamodelo, chamado *UbiModel*, que procura organizar as características e fatores de ubiquidade e suas interrelações de forma mais concreta e detalhada, minimizando o caráter interpretativo sobre essas

características e fatores presente na abordagem original. Assim, ao invés de ser guiado por um conjunto de perguntas, o especificador constrói a especificação a partir da instanciação do metamodelo *UbiModel*, ou seja, a partir da um conjunto rígido e bem definido de elementos, seus relacionamentos e definições. Nesse sentido, *UbiModel* se propõe a prover um direcionamento mais concreto acerca da especificação dos requisitos de ubiquidade através da construção de um modelo com os elementos necessários para a especificação desse tipo de requisito. É importante destacar que *UbiModel* foi totalmente definido a partir das características e fatores de ubiquidade apresentados por Spínola [4]. A Figura 1 mostra os elementos de *UbiModel* referentes a característica *Sensibilidade ao Contexto* que representa o comportamento (*behavior*) esperado do sistema frente a um contexto que se apresenta. Um contexto (*context*) é um cenário de utilização do sistema que considera usuários (*user*) participando de alguma atividade (*activity*) em alguma localização (*location*), sob determinadas condições (*condition*) do ambiente.

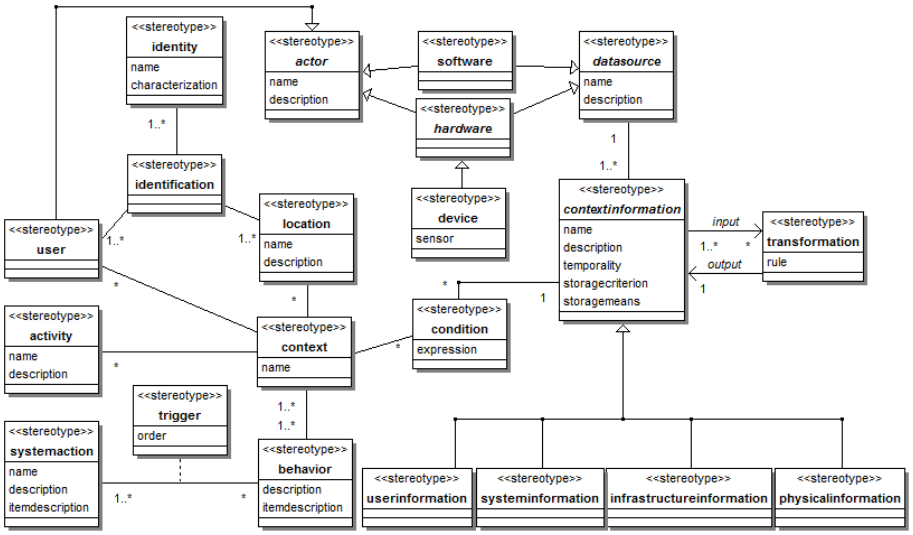

Fig. 1. Visão parcial do *UbiModel*

Outro aspecto explorado na pesquisa está relacionado às perspectivas nas quais os requisitos se apresentam. Os requisitos de ubiquidade descritos com *UbiModel* intencionam representar 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 caso fosse possível incluir também a descrição dos requisitos sob a perspectiva do sistema, ou seja, como o sistema irá se comportar a fim de atender às demandas, necessidades e restrições dos usuários. Nesse contexto, a abordagem proposta por Massollar [7] também define um metamodelo, chamado *UCModel*, que apoia a especificação funcional sob a ótica do comportamento esperado do sistema. Assim, o metamodelo *UbiModel* foi ampliado para que os modelos contendo os requisitos de ubiquidade também pudessem referenciar os modelos definidos com

o metamodelo *UCModel*, criando um rastro entre as necessidades/restrições de ubiquidade e o comportamento sistêmico associado à essas necessidades/restrições.

3 Contribuições da Pesquisa

As contribuições desta pesquisa estão relacionadas à definição de uma abordagem para especificação de requisitos de ubiquidade alinhada com os conceitos da computação ubíqua e as características funcionais, visando fornecer uma estrutura que garanta a qualidade da especificação, com expectativa de oferecer apoio à:

- 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 [4];
- Elaboração de um glossário de termos sobre computação ubíqua para apoiar o especificador;
- 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 voltadas para a redação das descrições dos requisitos de ubiquidade;
- Obtenção de especificação mais robusta, ao permitir a definição dos requisitos de ubiquidade tanto sob a perspectiva do usuário quanto do sistema.

3.1 Infraestrutura de Apoio à Especificação de Requisitos de Ubiquidade

A partir da definição do metamodelo *UbiModel* foi possível, também, definir uma infraestrutura computacional de apoio à especificação de requisitos de ubiquidade. Como os requisitos estão estruturados em um modelo (*UbiModel*) é possível realizar automaticamente a verificação sintática do mesmo. As principais funcionalidades construídas e oferecidas pela infraestrutura são:

- Caracterização de projetos de software ubíquos de acordo com um modelo de características e fatores de ubiquidade;
- Criação das especificações dos requisitos de acordo com o metamodelo *UbiModel*;
- Associação dos requisitos de ubiquidade definidos com elementos de *UCModel* [7];
- Verificação automática das restrições sintáticas previstas no metamodelo *UbiModel* da especificação do requisito;
- Geração da especificação de requisitos em formato texto (padrão RTF – *Rich Text Format*).

O uso dessa infraestrutura computacional se propõe a reduzir o esforço dedicado à especificação de requisitos de ubiquidade, pois grande parte das atividades são auto-

matizadas. Adicionalmente, a automação proporcionada pode evitar erros durante a execução das tarefas de especificação, visto que o número de interações manuais também é reduzido. A seguir são descritas as ferramentas que fazem parte desta infraestrutura.

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 em [4] para apoiar o desenvolvimento de projetos de software ubíquo sem levar em consideração as restrições e particularidades de cada projeto, pode reduzir a efetividade do uso deste conhecimento. Assim, é importante caracterizar o projeto no que diz respeito às características de ubiquidade, a fim de especializar o apoio à especificação dos requisitos de ubiquidade de acordo com as necessidades do projeto.

UbiSpecification: tem o objetivo de apoiar a especificação de requisitos de ubiquidade propriamente dita usando o metamodelo *UbiModel*, que funciona como um roteiro de especificação ao estruturar os elementos e relações importantes na descrição dos requisitos de ubiquidade. Além disso, permite a associação dos requisitos definidos com elementos de *UCModel* [7], modelo de especificação integrado a *UbiModel* que permite detalhar os requisitos por meio de casos de uso/diagramas de atividade.

UbiDocument: tem o objetivo de gerar um documento estruturado com a especificação de requisitos em formato textual a partir dos modelos gerados na ferramenta *UbiSpecification*.

4 Conclusão

Essa pesquisa tem por objetivo a elaboração de um arcabouço que forneça um contexto mais concreto e robusto para especificação de requisitos de ubiquidade, ao substituir o conjunto de perguntas abertas proposto por Spínola [4] por um metamodelo UML (UbiModel) a partir do qual essas especificações podem ser elaboradas.. Como o *UbiModel* tem o propósito de guiar a elaboração dos requisitos, espera-se que o especificador possa definir requisitos menos ambíguos e de forma menos dependente da experiência e de interpretações individuais do especificador. Além disso, espera-se também que a infraestrutura proposta reduza o esforço necessário para especificação dos requisitos, haja vista que ela oferece apoio para grande parte das atividades previstas. Por fim, também é uma meta do presente trabalho a redução de defeitos inseridos durante o processo de especificação dos requisitos, principalmente aqueles relacionados a omissões e inconsistências, visto que *UbiModel* possui uma estrutura bem definida, capaz de direcionar o especificador para as informações realmente relevantes. Ainda nesse contexto, o *UbiModel* oferece a possibilidade de explorar a verificação automática dos modelos de requisitos, principalmente na questão sintática.

5 Trabalhos Futuros e Em Andamento

Atualmente os modelos estão formalizados, a infraestrutura computacional construída e a pesquisa encontra-se em fase de planejamento de estudo (prova de conceito)

para avaliar a adequação do metamodelo *UbiModel* por meio da utilização da infraestrutura de apoio computacional desenvolvida.

Por fim, a abordagem proposta nesta pesquisa oferece oportunidades de pesquisa no que diz respeito a: (1) definição ou adaptação de técnicas de inspeção, alinhadas aos conceitos definidos na abordagem, e que possam explorar os elementos que a compõem: modelos conceituais, regras, descrições, dentre outras; (2) mapeamento dos requisitos definidos através de *UbiModel* para os possíveis itens de especificação que representam o detalhamento desses requisitos, e; (3) realização de estudos experimentais com o objetivo de verificar a redução de defeitos em documentos de especificação.

6 Referências

1. Aurum, A., Wohlin, C.: Engineering and Managing Software Requirements. Springer-Verlag, New York (2005)
2. Boehm, B. W.M Basili, V.R.: Software Defect Reduction Top 10 List. In: IEEE Computer, vol. 34 pp. 135-137. (2001)
3. Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., Burgelman, J. C.: Ambient Intelligence: From Vision to Reality. In: IST Advisory Group Draft Report, pp. 45-48 (2003)
4. Spínola, R.O.: Apoio à Especificação e Verificação de Requisitos Funcionais de Ubiquidade em Projetos de Software. Tese de Doutorado. COPPE/UFRJ. Rio de Janeiro (2010)
5. Spínola, R.O., Massollar, J.L., Travassos, G.H.: Towards a Conceptual Framework to Classify Ubiquitous Software Projects. In: Proceedings of the 8th International Conference on Software Engineering and Knowledge Engineering (SEKE), San Francisco (2006)
6. Pinto, F.: Uma Abordagem para Apoiar Especificação de Requisitos para Projetos de Software Ubíquo. Dissertação de Mestrado. COPPE/UFRJ. (2009)
7. Massollar, J.L.: 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 (2011)
8. Spínola, R. O., Travassos, G.H.: Arcabouço para Apoiar a Definição e a Garantia de Qualidade de Requisitos de Ubiqüidade em Projetos de Software. In: II Workshop on Pervasive and Ubiquitous Computing. Campo Grande (2008)
9. Spínola, R., Pinto, F.C.R, Travassos, G. H.: Ubicheck: An Approach to Support Requirements Definition in the Ubicomp Domain. In: 25th Symposium On Applied Computing, pp 306-310. Sierre (2008)
10. Massollar, J. L., Mello, R. M. de, Travassos, G. H.: Structuring and Verifying Requirements Specifications through Activity Diagrams to Support the Semi-automated Generation of Functional Test Procedures. In: QUATIC. Lisboa (2012)
11. Massollar, J L., Mello, R. M.de, Travassos, G. H.: Investigating the Feasibility of a Specification and Quality Assessment Approach Suitable for Web Functional Requirements. In: 30th International Conference of the Chilean, pp.108,117, 9-11. Computer Science Society (SCCC), Curico (2011)