Abordagem para Apoiar a Definição de Requisitos de Software Ubíquo

(*position paper*)

Felipe C. do R. Pinto, Rodrigo O. Spínola, Guilherme H. Travassos
COPPE / UFRJ – Programa de Engenharia de Sistemas e Computação
Caixa Postal 68.511 – 21.945-970 – Rio de Janeiro – RJ – Brasil
{felipecrp,ros,ght}@cos.ufrj.br

Resumo

As aplicações de software ubíquo intencionam tornar o uso do computador tão fácil que o usuário não perceberia sua presença. Ao mesmo tempo em que esse tipo de aplicação habilita o uso da computação por um público mais abrangente, o aumento da complexidade no desenvolvimento e a falta de técnicas para apoiar a construção do software ubíquo dificultam a sua construção. Com vistas a facilitar a construção do software ubíquo, bem como reduzir o número de defeitos do software construído, este trabalho apresenta uma proposta de abordagem para apoiar a definição de requisitos de ubiqüidade do software. A abordagem é baseada em um guia que através de diretrizes auxilia o desenvolvedor a elaborar o documento de requisitos do software.

1. Introdução

A computação ubíqua, segundo Weiser [1], representa uma evolução onde o uso da computação se torna tão simples que o usuário sequer a percebe. Esse conceito permite o uso da computação por um público mais abrangente em aplicações cotidianas, algumas delas exemplificadas a seguir [3]:

- rastreamento de alimentos para permitir acompanhar a sua distribuição desde o produtor até o consumidor;
- assistência a mobilidade para auxiliar deficientes, idosos e visitantes a se locomoverem em locais públicos, e;
- casa inteligente que proporciona controles avançados para melhorar qualidade de vida de seus moradores.

Além de estar ganhando espaço na indústria, a computação ubíqua caracteriza uma nova fase para a tecnologia digital [2]. Ao mesmo tempo, poucas ferramentas, técnicas, metodologias e processos estão disponíveis para auxiliar na construção deste tipo de software [6]. Essa carência tem motivado a elaboração de abordagens para apoiar a construção de software ubíquo [3,6]. Neste trabalho, optou-se por trabalhar com o apoio à definição de requisitos de ubiqüidade, pois, além de normalmente constituírem a primeira etapa do processo de desenvolvimento, defeitos nesta fase, segundo Boehm and Basili [7], podem resultar em até 50% de retrabalho desnecessário e custo de correção 100 vezes maior quando o software estiver em produção.

Neste contexto, este artigo apresenta uma abordagem para apoiar o analista na definição de requisitos de ubiqüidade do software. Para isto, é fornecido um guia com vistas a explicitar as preocupações que o desenvolvedor deve ter em mente quando pretende especificar requisitos para este tipo de software. O principal benefício esperado para este guia é a redução de defeitos relacionados a omissões de requisitos e a definição de requisitos incorretos. Adicionalmente, esse trabalho também explica como adaptar o guia citado para atender as necessidades específicas de um projeto de computação ubíqua.

Esse artigo é organizado em quatro seções, incluindo esta introdução. A seção 2 apresenta o contexto no qual a abordagem de definição de requisitos apresentada neste trabalho se insere. Já a seção 3 descreve a abordagem proposta. Por fim, a seção 4 apresenta as considerações finais do trabalho.

2. Arcabouço de Apoio à Definição e Garantia de Qualidade de Requisitos de Ubiqüidade em Projetos de Software

Spínola *et al.* [5] organizaram um corpo de conhecimento em computação ubíqua considerando características e fatores de ubiqüidade. Uma vez organizado este conhecimento, um arcabouço de apoio à definição e garantia de qualidade de requisitos de ubiqüidade foi proposto por Spínola *et al.[6]* para permitir seu uso durante à construção de projetos de software ubíquos. Sua utilização pelo analista de requisitos objetiva a redução dos riscos de projeto como retrabalho associados à presença de defeitos inseridos durante a especificação dos requisitos.

A figura 1 apresenta uma visão geral do arcabouço, que contempla:

- as facilidades apoiadas (caixas brancas) pela abordagem;

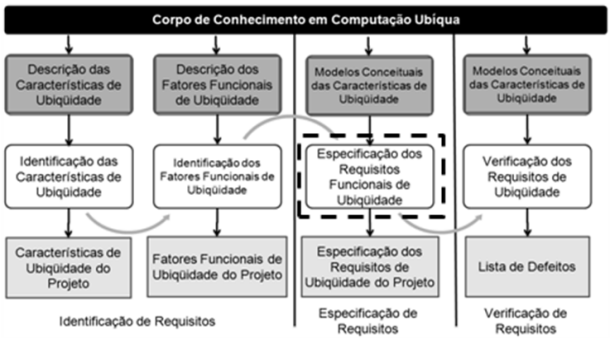

Figura 1: Visão geral do arcabouço

- as macro-atividades da engenharia de requisitos associadas às facilidades disponibilizadas (identificação, especificação e verificação de requisitos);
- o conhecimento utilizado para apoiar a realização das facilidades (caixas em cinza escuro);
- os principais artefatos gerados (caixas em cinza claro): **(1)** Características de Ubiqüidade do Projeto: define o conjunto de características de ubiqüidade contempladas no projeto; **(2)** Fatores Funcionais de Ubiqüidade do Projeto: define o conjunto de fatores funcionais de ubiqüidade contemplados no projeto e os requisitos funcionais elaborados a partir deles; **(3)** Especificação dos Requisitos de Ubiqüidade do Projeto: define de forma detalha os requisitos de ubiqüidade do software para serem utilizados posteriormente pelas equipes de projeto e desenvolvimento do projeto; **(4)** Lista de Defeitos: define o conjunto de defeitos que foram identificados durante a revisão da especificação de requisitos de ubiqüidade do software.

Tabela 1. Características de Ubiqüidade

Característica
Invisibilidade – permitir interagir com o sistema através de interfaces que tornem o uso do software transparente.
Onipresença – permitir que o software esteja sempre disponível.
Sensibilidade ao Contexto – permitir que o software perceba o que ocorre em sua volta.
Comportamento Adaptativo – permitir que o software mude seu comportamento de acordo com o seu contexto de atuação.
Captura da Experiência – permitir que o software aprenda.
Descoberta de Serviços – permitir que o software descubra outros serviços disponíveis na região.
Interoperabilidade Espontânea – permitir que o software se adapte para utilizar os serviços disponíveis.
Composição de Funções – permitir que o software ofereça uma funcionalidade através da composição de outras.
Heterogeneidade de Dispositivos – permitir que o software tenha comportamento equivalente em diferentes dispositivos.

O arcabouço proposto visa apoiar o analista de requisitos desde as etapas iniciais, onde são identificadas quais características de ubiqüidade estarão presentes no sistema, passando pelo apoio à definição e ao detalhamento dos requisitos de ubiqüidade do projeto, e fechando o ciclo com a revisão dos requisitos definidos para o sistema.

Este trabalho está inserido no contexto deste arcabouço e tem seu foco voltado para a facilidade **Especificação dos Requisitos Funcionais de Ubiqüidade** (quadrado tracejado na Figura 1).

3. Abordagem de Apoio à Definição de Requisitos de Ubiqüidade

Na seção 1, algumas aplicações que envolvem computação ubíqua foram exemplificadas. Embora elas sejam bem diferentes, é factível pensar que elas devem possuir características comuns.

Neste sentido, Niemela e Latvakoski [4] e Spínola *et al.* [5] realizaram trabalhos com vistas a identificar características específicas da computação ubíqua comuns entre projetos. Além de descrevê-las, o último trabalho lista também requisitos genéricos associados a cada uma. Esses requisitos são denominados fatores e podem ser utilizados para caracterizar projetos de software ubíquos [6]. A tabela 1 descreve as características ditas funcionais [5], as quais compõem a base para este trabalho.

A partir do detalhamento destas características de ubiqüidade através dos fatores funcionais, foi possível derivar um modelo conceitual para cada uma delas (observar o modelo para a característica sensibilidade ao contexto apresentado na Figura 3). Esses modelos não só explicitam os conceitos importantes das características de ubiqüidade, mas também ilustram os seus relacionamentos.

Os modelos foram construídos utilizando a notação UML, alinhada com os princípios de modelagem descritos por Clements [8], o que permitiu, através do uso de poucos estereótipos, representar as características de ubiqüidade e seus fatores associados. Outras notações foram avaliadas, mas com vistas a simplificar o entendimento e dispensar a necessidade de aprendizado de novas notações, elas foram descartadas.

A. Definir o contexto de operação do sistema

A1. Definir os contextos em que o sistema opera.
A2. Definir as informações que caracterizam os contextos de operação do sistema.
A3. Definir os conteúdos e valores válidos para as informações que definem os contextos de operação do sistema.

B. Definir as fontes de dados

B1. Enumerar as informações utilizadas para definir os contextos de operação do sistema.
B2. Definir as fontes de dados utilizadas para adquirir cada informação.
B3. Definir os contextos de operação do sistema onde as fontes de dados são válidas.
B4. Verificar para todos os contextos de operação do sistema, se as informações utilizadas para defini-lo possuem pelo menos uma fonte de dados operacional neste contexto.
B5. Definir como as informações adquiridas podem ser avaliadas quanto a sua validade.

C. Definir os dispositivos habilitados

C1. Definir os dispositivos habilitados a acessar o sistema
C2. Definir os contextos de operação que cada dispositivo suporta.

Figura 2. Exemplo de guia para definição de requisitos da característica de ubiqüidade "Sensibilidade ao Contexto"

Estes modelos conceituais podem ser usados para orientar a elaboração de requisitos relacionados à computação ubíqua. Para isto, os modelos foram utilizados para gerar um guia para auxiliar a definição da especificação de requisitos de ubiqüidade do software.

A idéia até este ponto é que o guia oriente o analista de requisitos para evitar que ele insira defeitos de omissão e fatos incorretos. Para isto, este guia reúne necessidades que podem ser relevantes para projetos de software ubíquos em geral. Estas necessidades são derivadas dos conceitos contemplados nos modelos conceituais e dos relacionamentos entre eles.

Porém, uma vez que cada projeto possui necessidades específicas, é necessário derivar um guia específico que aborde apenas o que é relevante para o projeto. Em linha com esse objetivo, Spínola *et al.* [6] propõem que um projeto de software ubíquo pode ser caracterizado de acordo com a presença dos fatores de ubiqüidade. Uma vez que as características de ubiqüidade são compostas por estes fatores e que os modelos são derivados deles, é possível reduzir os modelos para que eles representem apenas os fatores relevantes para o projeto. Em complemento, como o guia está associado aos modelos conceituais, ele também pode ser reduzido para orientar apenas a definição dos requisitos presentes no modelo reduzido.

A figura 2 ilustra um exemplo do guia baseado no modelo da característica Sensibilidade ao Contexto, ilustrado na figura 3. Ele não só explicita preocupações que podem ser pertinentes para o analista de requisitos, mas introduz um passo a passo a ser seguido para definir os requisitos relacionados à ubiqüidade.

Assim, mantendo os rastros entre os fatores e o modelo e entre o modelo e o guia, é possível a partir da caracterização de um projeto gerar um guia de apoio à definição de requisitos de ubiqüidade específico para ele.

A figura 4 ilustra um exemplo do modelo da característica de ubiqüidade Sensibilidade ao Contexto após a filtragem referente à caracterização de um projeto fictício. Ao compará-la com a figura 3, é possível perceber que diversos elementos foram removidos por

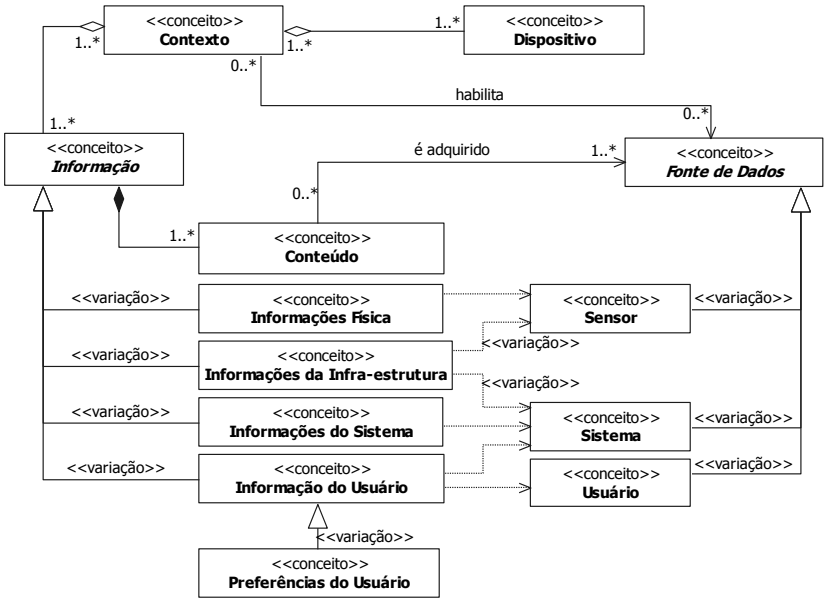

Figura 3. Modelo de exemplo da característica de ubiqüidade "Sensibilidade ao Contexto"

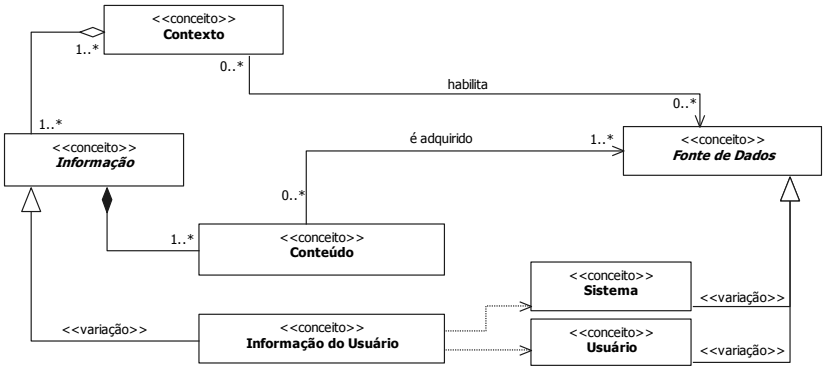

Figura 4. Modelo de exemplo da característica de ubiqüidade "Sensibilidade ao Contexto" filtrado para atender as necessidades de um projeto específico

não serem mais relevantes para o projeto em questão. Da mesma forma, essas alterações podem ser replicadas para o guia exemplificado na Figura 2, e desta forma, por exemplo, poderiam ser removidas as diretrizes C, C1 e C2. Percebe-se que o conceito Dispositivo não está contemplado no modelo conceitual e por conta disso, as diretrizes associadas a ele foram removidas do guia de apoio à definição de requisitos.

Com o guia especializado, o analista tem em mãos um conjunto de conhecimento estruturado e direcionado de forma a apoiar suas atividades relacionadas à especificação dos requisitos de ubiqüidade.

4. Conclusão

A necessidade de apoio para o desenvolvimento de software ubíquo [3] motivou a elaboração de uma abordagem para apoiar a definição de requisitos de projetos de software ubíquo. O trabalho proposto elabora um guia (fundamentado em conhecimento do domínio da área de computação ubíqua) para auxiliar o desenvolvedor a evitar que defeitos relacionados à omissão de requisitos e a definição de fatos incorretos sejam inseridos na especificação dos requisitos de ubiqüidade. A viabilidade desta hipótese é reforçada pelo fato de que abordagens de apoio a atividades de desenvolvimento de software para um domínio específico podem ter sua eficácia aumentada ao fazer uso do conhecimento do domínio [9].

Embora este trabalho não tenha foco em auxiliar o desenvolvedor nas fases seguintes à definição de requisitos, acreditamos que com alguma adaptação, os artefatos gerados podem ser utilizados em fases posteriores, como projeto e testes.

No estágio atual deste trabalho de mestrado foram elaborados modelos para as características de ubiqüidade apresentadas na Tabela 1. Além disso, está em construção o guia de apoio à definição de requisitos para as características de ubiqüidade "Sensibilidade ao Contexto" e "Comportamento Adaptativo".

5. Agradecimentos

Os autores gostariam de agradecer à CAPES, CNPq (Engenharia de Software Experimental e Ciência em Larga Escala: 475459/2007-5) e FAPERJ – Cientista do Nosso Estado pelo apoio financeiro.

6. Referências

[1] Weiser, M. "The Computer for the 21st Century". Scientific American 1991, pp. 94-104.

[2] Krikke, J. (2005) "T-Engine: Japan's Ubiquitous Computing Architecture Is Ready for Prime Time", IEEE pervasive computing, Vol.4, No.2, ISSN: 1536-1268.

[3] Sakamura, K. (2006) "Challenges in the age of ubiquitous computing: a case study of T-Engine, an open development platform for embedded systems", ICSE '06: Proceedings of the 28th international conference on Software engineering, ACM, 713-720.

[4] Niemelä, E. and Latvakoski, J. (2004) "Survey of requirements and solutions for ubiquitous software", MUM '04: Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia, ACM, 71-78.

[5] Spínola, R. O., Massollar, J. and Travassos, G. H. (2007) "Checklist to characterize Ubiquitous Software Projects", proceeding of the XXI Brazilian Symposium on Software Engineering (SBES).

[6] Spínola, R.O., Pinto, F.C.R., Travassos, G.H. (2008) "Supporting Requirements Definition and Quality Assurance in Ubiquitous Software Project". Proceedings of the 3rd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation.

[7] Boehm, B. and Basili, V. R. (2001) "Software Defect Reduction Top 10 List", IEEE Computer, 34, n 1, 135-137.

[8] Clements, P.; Bachmann, F.; Bass, L.; Garlan, D.; Ivers, J.; Little, R.; Nord, R. & Stafford, J. (2004) "Documenting Software Architectures: Views and Beyond" Addison-Wesley.

[9] Oliveira, K. M.; Travassos, G. H.; Menezes, C. *et al.* Ambientes de Desenvolvimento de Software Orientados a Domínio. XIV Simpósio Brasileiro de Engenharia de Software, Sessão de Ferramentas. João Pessoa, Brasil, 2000.