Um processo para construção de software mais transparente

Eduardo Almentero[1], and Julio Cesar Sampaio do Prado Leite[1]

[1]Pontifícia Universidade Católica do Rio de Janeiro, PUC - Rio, Brasil

{ealmentero, julio}@inf.puc-rio.br

Resumo. À medida que o software está cada vez mais presente em nossas vidas, se tornando parte de nosso cotidiano, a sociedade também exigirá sua transparência. Cabe, portanto à engenharia de software, como disciplina, propor métodos e ferramentas para apoiar esta nova demanda. Um software é transparente quando manipula informações transparentes e informa sobre si mesmo, como funciona, o que faz e porque o faz. Devido à complexidade deste novo requisito, ele foi entendido como um conjunto de qualidades que contribuem individualmente para transparência. Neste trabalho, procuramos abordar o requisito de transparência sob a ótica da simbiose entre modelos de requisitos e artefatos de arquitetura e código, com o objetivo de abranger um maior conjunto das qualidades exigidas para tornar o software transparente.

Palavras-chave: transparência de software, rastreabilidade, requisitos, cenários, arquitetura, documentação.

1 Introdução

À medida que o software está cada vez mais presente em nossas vidas, se tornando parte de nosso cotidiano, a sociedade também exigirá sua transparência, principalmente daqueles que utilizamos com mais freqüência e que tem grande importância para nós, como por exemplo, os sistemas bancários. Atualmente, sabemos que a comunicação feita através do software é obscura, porém é essencial que o software em si seja totalmente transparente, para que todos possam ter o conhecimento sobre exatamente o que ele faz e como faz.

Um software é transparente quando manipula informações transparentes (transparência da informação) e informa sobre si mesmo, como funciona, o que faz e porque o faz (transparência do processo) [1]. Segundo Leite [2], a necessidade de transparência terá um grande impacto no processo de desenvolvimento de software. Portanto, é necessário que pesquisas sejam realizadas no sentido de atender esta demanda para aqueles que usam o software ou são de alguma forma afetados por ele.

O grupo de engenharia de requisitos da PUC-Rio (Grupo ER) [7] é pioneiro na pesquisa de transparência de software. Nesse sentido, o grupo entendeu que o conceito de transparência era muito complexo e abstrato, mas que poderia ser mais bem entendido como um conjunto de qualidades (metas flexíveis) que contribuem individualmente para a transparência. A fim de definir que qualidades deveriam ser consideradas e de que forma contribuiriam para transparência, realizamos um extensivo "survey" e validações guiadas por questionário. O NFR framework foi utilizado para modelar a rede de metas flexíveis criada para definir a transparência. Esta rede é composta por 27 nós folha e quatro grupos [3]. Cada nó é uma característica ou meta flexível necessária para alcançarmos a transparência.

Em particular, no trabalho [4] evoluído posteriormente em [5] requisitos modelados através de cenários foram associados ao código fonte, de forma que este se tornasse mais fácil de ser entendido, aumentando assim sua transparência. Cenário é a descrição de situações comuns ao cotidiano [6], ou seja, é uma técnica de descrição que auxilia a compreensão de uma situação específica de um software, priorizando seu comportamento. Os cenários devem abordar aspectos de usabilidade, permitir um maior detalhamento do conhecimento do problema, a unificação de critérios, a obtenção do compromisso de usuários, a organização de detalhes e o treinamento de pessoal. Escolhemos uma abordagem baseada em cenários porque quando utilizados na engenharia de requisitos, os cenários ajudam tanto na compreensão do sistema que será desenvolvido quanto na validação do conhecimento do engenheiro pelo cliente. Isto se dá pelo fato de que os cenários são descritos em linguagem natural e modelam situações do sistema, fazendo com que clientes se sintam mais à vontade e entendam melhor o processo de descobrimento dos requisitos.

Como a transparência impacta o processo de desenvolvimento de software como um todo, sabemos que é essencial não limitá-la apenas ao artefato de código fonte. Desta forma, utilizamos a experiência adquirida na definição da transparência de software, com objetivo de viabilizar a transparência do processo e dos artefatos produzidos durante o desenvolvimento do software. **Nossa principal estratégia é a associação de modelos de requisitos aos artefatos**, de forma que seja mantida a rastreabilidade entre eles e entre os artefatos produzidos em cada fase, durante o ciclo de desenvolvimento do software. Desta forma, procuramos atender às diferentes qualidades requeridas para se obter um software transparente.

Este artigo está organizado da seguinte forma: na seção 2 são apresentados os objetivos delineados para a pesquisa. Na seção 3 descrevemos as contribuições alcançadas com o trabalho realizado. As conclusões obtidas durante a realização da pesquisa são discutidas na seção 4. E por fim, na seção 5, os trabalhados que estão sendo realizados atualmente e os futuros são detalhados.

2 Objetivos da Pesquisa

O presente trabalho de pesquisa tem como objetivo principal investigar e definir uma estratégia para alcançar a transparência de um software. Tal objetivo implica na adequação do processo de desenvolvimento, de forma que em cada fase a transparência seja abordada com o uso de modelos de requisitos associados aos artefatos produzidos.

A primeira etapa investigada foi a fase de definição da arquitetura do software. Esta arquitetura será composta de cenários organizados em grupos. Tais grupos serão identificados a partir de uma análise do léxico ampliado da linguagem (LAL) [9], que é uma técnica baseada na descrição de termos através de noção e impacto. Esta técnica foi criada para ajudar no entendimento da linguagem específica do domínio, contribuindo desta forma para o entendimento do domínio como um todo. Esta análise se baseia na idéia que termos muito utilizados dentro do domínio representam conceitos importantes, e podem ser aproveitados para organizar as situações do sistema. A inclusão dos cenários no grupo se baseia na comparação dos elementos que compõem sua descrição com a definição do termo que representa o grupo. A Fig. 1exibe o processo de divisão dos cenários em grupos.

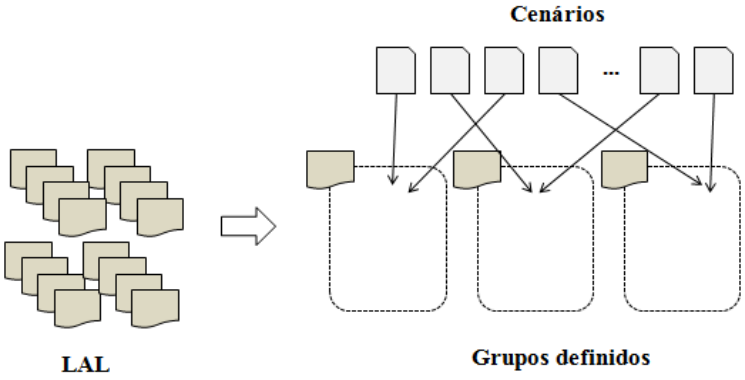

Fig. 1. Processo de organização dos cenários em grupos

Com base nesta organização inicial, propomos a divisão dos cenários em camadas, de acordo com o framework MVC [10]. Esta divisão visa à separação da interface do comportamento do sistema, permitindo um melhor entendimento de ambos, além de estabelecer os cenários de controle, que serão responsáveis por administrar o fluxo de informações do sistema e aplicar as regras de negócio. Após a divisão, os cenários de cada camada são detalhados de acordo com uma série de heurísticas já estabelecidas e são posteriormente operacionalizados, de forma que os cenários fiquem associados ao código, formando um só documento.

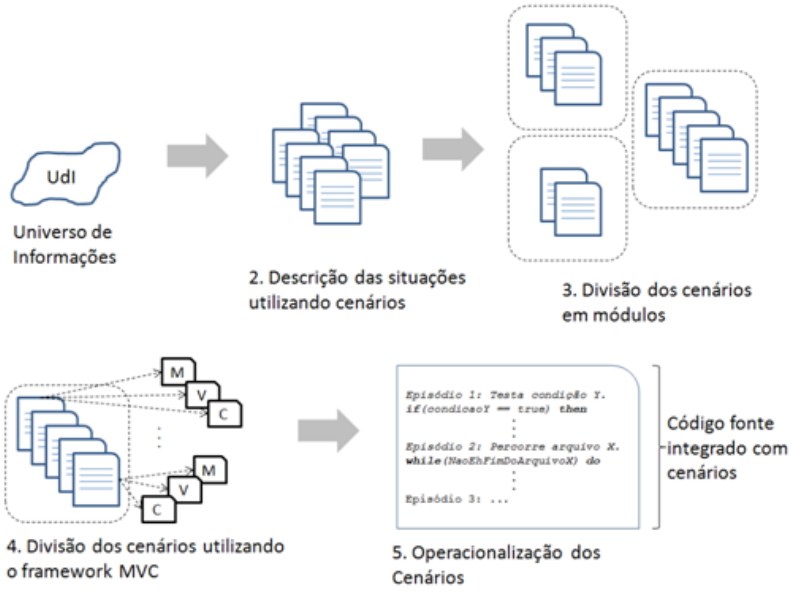

Fig. 2. Processo de desenvolvimento com viés de transparência

Este processo permite a rastreabilidade do código para os cenários que o documentam e destes para os cenários da arquitetura, que por sua vez podem ser imediatamente relacionados aos cenários iniciais, isto é, a especificação de requisitos do sistema, como mostra a Fig. 2. Diante de tal estrutura, toda alteração, tanto da dos cenários utilizados como documentação quanto do código, pode ser propagada facilmente, ajudando a evitar a erosão da documentação.

3 Contribuições Científicas

Acreditamos que a grande contribuição deste trabalho é a definição de um processo com passos bem definidos, que pode ser facilmente transformado em um processo semiautomático, facilitando a construção de software mais transparente. O processo proposto engloba a definição da arquitetura de forma guiada a partir do LAL e propõe sua descrição através de cenários. A abordagem de cenários escolhida permite a descrição de requisitos não funcionais. Desta forma conseguimos tratar outro problema interessante, que é a descrição destes requisitos no documento de arquitetura. O detalhamento destes cenários evolui durante o processo de desenvolvimento, proporcionando uma divisão conceitual da arquitetura com base no framework MVC, até chegar à efetiva implementação do software. Como todo o processo é essencialmente baseado na evolução de cenários, e estes artefatos, quando evoluídos, mantêm naturalmente um elo com os anteriores, temos a possibilidade de rastrear o impacto de

cada decisão até o código fonte e também seguir o sentido oposto. Tendo associados a estes cenários o código e a arquitetura, dispomos do rastro da arquitetura para o código e vice versa.

Outra importante contribuição foi a evolução da plataforma C&L [8] e construção de uma linha de produtos para bibliotecas digitais, ambas guiadas por qualidades de transparência. Esta evolução do C&L iniciou sua transformação em uma plataforma de desenvolvimento de software transparente. Os esforços deste trabalho visam à evolução desta ferramenta para atender cada vez mais as necessidades de desenvolvedores e outros interessados no desenvolvimento de software transparente. Os trabalhos realizados pelo grupo neste sentido incentivam outros que também contribuem para o enriquecimento da plataforma.

4 Conclusões

Neste trabalho abordamos uma estratégia para possibilitar o desenvolvimento de software mais transparente. Embora mais experimentos sejam necessários, durante a aplicação deste método em sistemas simples, como o de uma biblioteca digital, tivemos indícios de que o software produzido realmente atende uma parcela significativa das qualidades necessárias para a transparência, nos permitindo dizer que o tornamos mais transparente. Além disto, percebemos que a manutenção da documentação do software se tornou mais simples, devido à sua proximidade do código. E por fim, outro benefício foi a possibilidade de especificar o impacto de requisitos não funcionais, devido ao uso de cenários, já na descrição da arquitetura do sistema.

5 Trabalhos em Andamento e Futuros

Atualmente trabalhamos na criação de um experimento para a aplicação da estratégia proposta na prática. O desenho deste experimento visa identificar a complexidade de aplicação da solução e de que forma ela contribui para a manutenção do software e de sua documentação. Desejamos também verificar de que forma um software desenvolvido de acordo com tal proposta atende a alguns requisitos de qualidade essenciais para que a transparência seja alcançada, como entendimento e leiturabilidade, por exemplo.

Como trabalho futuro, pretendemos evoluir a ferramenta C&L de forma que ela dê suporte à aplicação do método, automatizando as etapas com regras bem definidas, como a identificação de grupos a partir do LAL e organização de cenários, e facilitando as etapas mais complexas, como a divisão dos cenários em camadas e seu detalhamento. Desejamos também evoluir a ferramenta para que ajude na evolução dos cenários associados ao código fonte, de forma que seja mais fácil evitar a erosão da documentação devido à evolução dos requisitos do software. A disponibilização de uma ferramenta para facilitar a utilização das abordagens propostas certamente irá atrair mais atenção ao uso de técnicas para construção de software mais transparente.

6 Referências

1. Leite, J. C. P., Cappelli, C.; Exploring i* Chracteristics that Support Software Transparency. In Proceedings, Vol. 322, 2008, PP.51-54.
2. http://amazinnggg.blogspot.com/2006/07/software-transparency.html. Acessado em 10/03/2013.
3. Cappelli, C.; An Approach for Business Processes Transparency Using Aspects. Tese de Doutorado. Departamento de Informática, PUC-Rio, 2009.
4. Silva, Lyrene Fernandes da, Sayão, Miriam, Leite, Julio Cesar S. P., Breitman, Karin Koogan; Enriquecendo o Código com Cenários; XVII Brazilian Symposium on Software Engineering (SBES2003), Manaus-AM, 2003.
5. Almentero, E. ; Re-engenharia do Software C&L para Plataforma Lua-Kepler utilizando princípios de transparência. Dissertação de Mestrado. Departamento de Informática, PUC-Rio, 2009.
6. Leite, J.C.S.P, Rossi, G., Balaguer, F., Maiorana, V., Enhancing a requirements baseline with scenarios; In: Third IEEE International Symposium on Requirements Engineering - RE97, Proceedings. IEEE Computer Society Press, January, 1997, pp 44-53.
7. URL: http://www-di.inf.puc-rio.br/~julio/Slct-pub/transp-sbes.pdf
   Grupo de Pesquisas em Engenharia de Requisitos da Puc-Rio. Disponível em: http://transparencia.inf.puc-rio.br/wiki/index.php/Integrantes. Acessado em 10/03/2013.
8. C&L – Cenários e Léxicos – Disponível em: http://pes.inf.puc-rio.br/cel. Acesso em: 12/03/2013.
9. Leite, JCS do, and Ana PM Franco. "A strategy for conceptual model acquisition." Requirements Engineering, 1993., Proceedings of IEEE International Symposium on. IEEE, 1993.
10. Krasner, Glenn E., and Stephen T. Pope. "A description of the model-view-controller user interface paradigm in the smalltalk-80 system." Journal of object oriented programming 1.3 (1988): 26-49.