Aumentando a Transparência no Software C&L - Lua

Edgar Sarmiento Calisaya, Eduardo Almentero e Julio Cesar Sampaio do Prado Leite

Pontifícia Universidade Católica do Rio de Janeiro, PUC - Rio, Brasil

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

Abstract. *Software is transparent when the information processed is transparent and it informs about itself, how it does, what it does and why it does. The use of scenario and lexicon languages for modeling and specification of software requirements enable the construction of transparent software, because they make explicit: (1) the entities and the information processed by the system, (2) the operations that process this information, (3) the relationships between entities, information and operations, This work shows, how the C&L – Lua software enables the comprehension, the transparency and the evolution of software requirements in test cases of the different execution flows of the software.*

Resumo. *Um software é transparente quando as informações que manipula são transparentes e informa sobre si mesmo, como funciona, o que faz e por que o faz. O uso das linguagens de Cenário e Léxico para a modelagem e descrição de requisitos de software possibilitam a construção de software mais transparente; pois estes explicitam: (1) as entidades e informações manipuladas pelo sistema, (2) as operações que manipulam estas informações, (3) os relacionamentos entre as entidades, informações e as operações. Neste trabalho, procuramos mostrar cómo o software C&L - Lua facilita a compreensão, a transparência e a evolução dos requisitos em casos de teste dos diferentes fluxos de execução do software.*

1. Introdução

Diversas pesquisas demonstraram que o processo de desenvolvimento de software deveria ser dirigido pelos requisitos [Leite 2010 e Van Lamsweerde 2003]. Isto principalmente porque os conceitos ou entidades referenciadas nas descrições dos requisitos de um determinado sistema são (e devem ser) usados no restante das etapas do processo de desenvolvimento (projeto e implementação). Alem disso, demonstrou - se que o caminho para construir software transparente precisa de métodos de desenvolvimento de software dirigidos por requisitos [Leite 2010 e Almentero 2013]. Estes requisitos devem ser entendíveis e transparentes para todos os envolvidos (cidadãos, usuários, clientes e desenvolvedores) neste processo.

Neste contexto, cabe à engenharia de software, propor os métodos, técnicas, linguagens e ferramentas que explicitem e aumentem a transparência dos conceitos, entidades e operações implementadas pelo sistema desde as etapas inicias do processo de construção de software adotado.

O uso de tecnicas ou linguagens de descrição de requisitos que explicitem de forma transparente e compreensível os conceitos, entidades, operações de um sistema e seus relacionamentos, facilitam a transição ou evolução dos requisitos para outros artefatos de software. A transição de requisitos para outros artefatos de software tem sido um dos principais desafios no desenvolvimento de software e diversas abordagens que visam ajudar neste processo têm sido propostas (Model-driven Development).

Uma abordagem de especificação de requisitos baseada nas linguagens de cenário e léxico (LAL) [Leite 2000] pode aumentar a transparência dos requisitos. O uso destas linguagens também facilita a evolução dos requisitos em outros artefatos de software, pois, os cenários também são usados para descrever: interações entre o sistema e seu ambiente, interações entre conceitos ou módulos do sistema, e operações ou procedimentos executados pelo sistema.

O Léxico Ampliado da Linguagem – LAL [Leite 2000] expressa a denotação (noção) e a conotação (impacto) dos conceitos utilizados na aplicação - símbolos. A noção explica o significado literal do símbolo, e o impacto descreve os efeitos do uso ou ocorrência do símbolo no domínio sob estudo. Podemos classificar os símbolos do léxico gramaticalmente como sujeito, verbo, estado ou objeto.

Cenário [Leite 2000] é uma técnica que auxilia o entendimento de uma situação específica de uma aplicação de software, priorizando o seu comportamento. A estrutura deste modelo é composta das seguintes entidades: Título, Objetivo, Contexto, Recursos, Atores, Episódios, Exceções e Restrições. Atores e Recursos são representados por listas; e, título, objetivo, contexto e exceção são sentenças declarativas simples. Os episódios representam o curso principal das ações de um cenário, mas também incluem variações e outras alternativas possíveis. Uma exceção pode ocorrer durante a execução dos episódios, sinalizando que existe um empecilho para satisfazer o objetivo. Restrições detalham aspectos não funcionais nos recursos ou nos episódios.

O uso das linguagens de cenário e léxico (apoiado por uma ferramenta) no processo de construção de software pode aumentar a transparencia tanto do produto desenvolvido quanto do processo adotado. Isto pelo fato de que o uso destas linguagens operacionaliza ou implementa alguns dos requisitos não funcionais que ajudam a alcançar a transparência do software. Por exemplo: Dependência, Detalhamento, Consição, Clareza, Explicação e Rastreabilidade. E, para serem efetivamente utilizados, faz-se necessário o uso de boas ferramentas de edição, visualização e gerenciamento de cenários e léxicos, para ganho de produtividade.

O grupo de engenharia de requisitos da PUC-Rio [Grupo ER 2014] classificou a transparência como um conjunto de requisitos não funcionais (metas flexíveis), organizados hierarquicamente através de um grafo de interdependências (SIG – NFR Framework [Chung 2000]).

2. Objetivos da Pesquisa

O impacto do uso das linguagens de cenário e léxico na transparência de alguns artefatos de software já foi avaliado em [Almentero 2013]; nesta pesquisa os requisitos modelados como cenários foram associados ao código fonte da aplicação (C&L - Lua,

WERPapers). Desta forma, o entendimento, explicação, clareza, rastreabilidade e a evolução dos requisitos em outros artefatos foram positivamente impactados.

A descrição de requistos através de cenários tem um impacto positivo sobre a transparência do processo de desenvolvimento de software como um todo e os artefatos produzidos durante as fases deste processo, e, o uso destas linguagens facilita a transição dos modelos de requisitos a outros artefatos, de forma que seja mantida a rastreabilidade entre os modelos de requisitos e os artefatos produzidos em cada fase.

Neste contexto, a transição de modelos de requisitos para modelos de teste (Model-driven Testing) sempre foi um tópico desafiante, pois as tarefas de testes são custosas, demoradas e obscuras. Então, a especificação transparente de requisitos pode facilitar a transformação automatizada de descrições de requisitos (cenários) para casos de teste. Desta forma, procuramos ampliar o número de requisitos não funcionais operacionalizados e requeridos para se obter um software mais transparente: Controlabilidade, Verificabilidade e Validade.

O objetivo desta pesquisa é facilitar a transição de descrições de requisitos (cenários) a modelos de casos de teste de uma maneira a seguir os princípios de transparência de software e paralelamente melhorar a testabilidade [Bach 2003] do produto de software.

3. Contribuições Esperadas

A contribuição principal deste trabalho é a demonstração de que a geração de casos de teste a partir da especificação de requisitos (cenários e léxico) aumenta a transparência tanto dos artefatos produzidos ao longo do processo quanto do produto final. Desta forma, um número maior de requisitos não funcionais relacionados à transparência (e testabilidade) são operacionalizados.

A implementação da abordagem para a geração de casos de teste a partir de descrições de requisitos, foi baseada na ferramenta para a edição e visualização de cenários e léxico C&L-Lua desenvolvida pelo grupo de engenharia de requisitos da PUC-Rio [Almentero 2013]. O processo de desenvolvimento adotado para as implementações foi o proposto por [Almentero 2013]. A abordagem para a geração de casos de teste foi desenvolvida com o objetivo de reduzir os custos do processo de testes, além de visualizar em forma gráfica a apresentação dos casos de teste e os elementos de teste [Sarmiento 2014].

Na nossa abordagem, a linguagem de Cenário [Leite 2000] é usado para descrever o comportamento da aplicação através de episódios; palavras ou frases relevantes da aplicação (Léxico) referenciados dentro das descrições de cenários: (1) tornam os dados de entrada e as condições que executam os diferentes cenários de teste explícitas a partir de descrições iniciais de requisitos, (2) permitem a evolução e derivação de modelos, (3) estas informações podem também ser utilizadas para derivar e reduzir o número de cenários de teste.

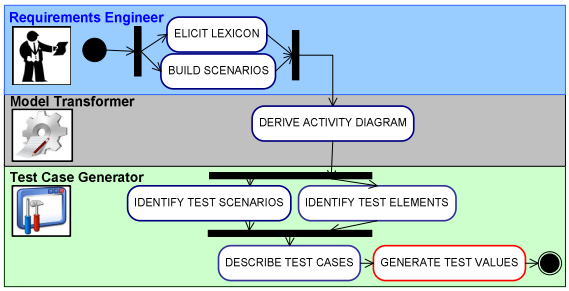

Figura 1. Processo de Derivação de Casos de Teste.

4. Resultados já Alcançados

Cada uma das etapas da abordagem para a geração de casos de teste a partir de cenários, e descritas na Figura 1, estão detalhadas em [Sarmiento 2014]. Os algoritmos de transformação de modelos estão sendo implementados junto com uma ferramenta de apoio a este processo. Os algoritmos propostos são capazes de lidar com as descrições de requisitos que contém estruturas complexas como a concorrência. O desenvolvimento esta sendo baseado na ferramenta C&L - Lua [C&L 2014]. C&L - Lua é uma aplicação web desenvolvida na linguagem Lua.

A arquitetura do C&L - Lua é baseada no framework MVC. Na arquitura existe a divisão física do sistema em camadas e a divisão lógica em módulos. Cada um dos módulos está distribuído pelas camadas de visão, controle e modelo, como mostra a Figura 2. A camada de Util a grupa os módulos para manipulação de “grafos”.

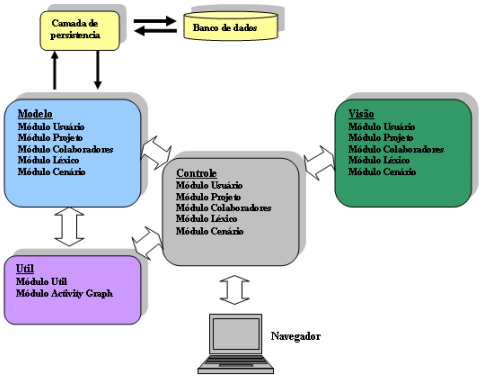

Figura 2. Arquitetura do C&L – Lua.

A Figura 4 ilustra através de um exemplo as principais tarefas da nossa abordagem de geração de casos de teste a partir de cenários. O exemplo ilustrado é a derivação de casos de teste para o “Processo de Transformação de Cenário em Casos de Teste” descrito na Figura 3.

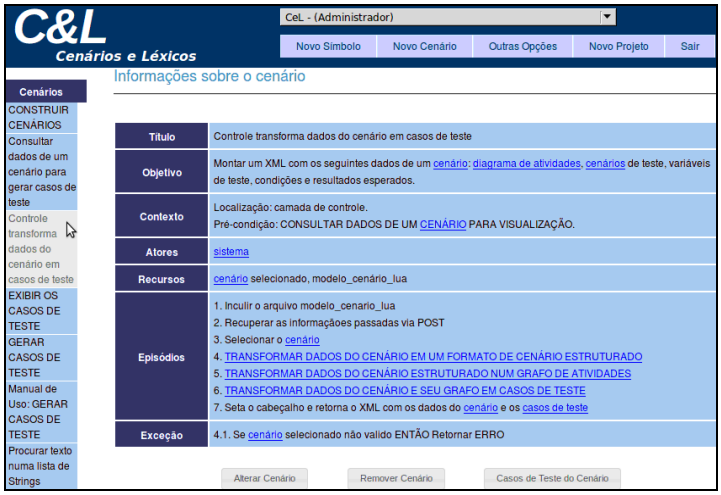

Figura 3. Funcionamento do C&L – Lua: Cenários.

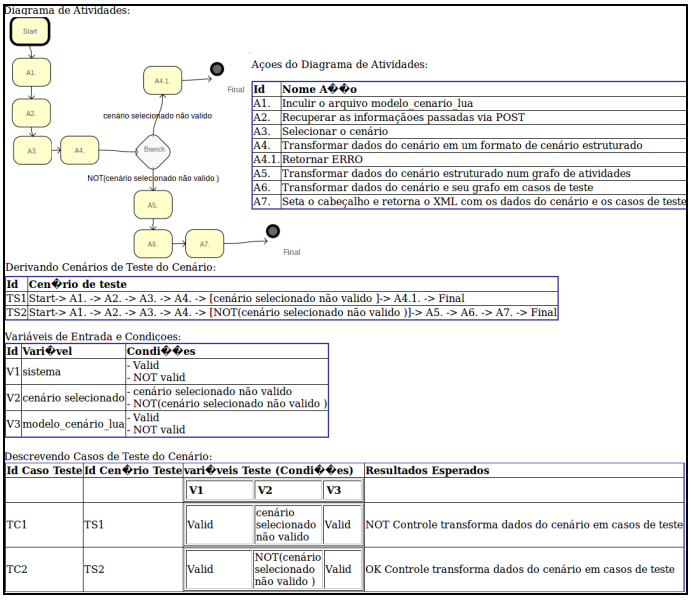

Figura 4. Funcionamento do C&L – Lua: Geração de Casos de Teste a partir de Cenários.

Na Figura 4 podemos observar os Cenários de teste (TS1,TS2), as Variáveis de teste (Sistema, Cenário selecionado) e suas Condições identificadas para o processo de transformação de cenários em casos de teste.

Também são descritos os Casos de teste considerando os cenários de teste e as condições das variáveis necessárias. O caso de teste TC1 executa uma condição não valida da variável Cenário (cenário não válido) e o caso de teste TC2 executa uma condição valida da variável Cenário (cenário válido).

5. Conclusão

Neste trabalho abordamos uma estratégia para contribuir com a transparência de software. Nesta abordagem, a especificação de requistos baseada em cenários e léxicos ajuda os desenvolvedores na identificação dos diferentes casos de teste que exercitam os diferentes fluxos de execução do software e, consequentemente, melhorar a qualidade

do produto, reduzir os custos de falhas em uso e os custos de manutenção depois de entregue o produto final. Isto porque os conceitos e os fluxos de execução das operações implementadas são explicitas ou transparentes para todos os envolvidos no processo de software desde as etapas inciais deste processo.

O software C&L vem sendo utilizado e evoluído, pelo grupo de engenharia de requisitos da PUC-Rio. Seus resultados são positivos e, por isso, sua evolução continua.

No futuro, serão implementadas abordagens para a verificação de cenários e léxicos de acordo com suas regras específicas de construção.

6. Referências

Leite, J. C. S. P., Hadad, G., Doorn, J., Kaplan, G. (2000) A scenario construction process, Requirements, Engineering Journal, Springer-Verlag London Limited, 5, 1, p. 38-61.

Leite, J. C. S. P., Cappelli, C. (2010) Software Transparency, *Business & Information Systems Engineering*, Vol. 2: Iss. 3, 127-139.

Heumann, J. (2001) Generating test cases from use cases, IBM.

Binder, R. V. (2000) Testing object-oriented systems, Addison Wesley.

Cockburn, A. (2001) Writing Effective Use Cases, Addison-Wesley, Reading, MA.

Denger, C., Medina, M. (2003) Test Case Derived from Requirement Specifications, Fraunhofer IESE Report.

Grupo de Pesquisas em Engenharia de Requisitos da PUC-Rio. Disponível em: http://transparencia.inf.puc-rio.br/wiki/index.php/Integrantes.

C&L – Cenários e Léxicos - Disponível em: http://pes.inf.puc-rio.br/cel.

Van Lamsweerde, A. (2003) From System Goals to Software Architecture, Formal Methods for Software Architectures.

Chung L., Nixon B.A., Yu E., Mylopoulos J. (2000) Non- Functional Requirements in Software Engineering, Kluwer Academic Publishing, Boston Hardbound.

Bach, J. (2003) Heuristics of Software Testability. Disponível em: http://www.satisfice.com/tools/testable.pdf Acessado em 20/07/2013.

Almentero, E. (2013) Dos Requisitos ao Código: Um Processo para Desenvolvimento de Software mais Transparente, Tese de Doutorado, Departamento de Informática, PUC-Rio.

Sarmiento, E., Leite, J.C.S.P., Rodriguez, N., von Staa, A. (2014) An Automated Approach of Test Case Generation for Concurrent Systems from Requirements Descriptions, In Proceedings of ICEIS.