Uma Proposta de Sintaxe Concreta para Caso de Uso

José O. B. Colombini[1], **Fábio Levy Siqueira**[1]

[1]Departamento de Computação e Sistemas Digitais (PCS)
Escola Politécnica da Universidade de São Paulo (EP - USP)

{josecolombini, levy.siqueira}@usp.br

Abstract. *Problems in requirements elicitation and documentation may affect the entire software development process. Out of the possibles problems there are ambiguity, inconsistency, not in conformance to a standard, unclear documentation, etc. One way to avoid these problems is by using techniques that tries to mitigate the informality of this process. In this work we use the concepts of Model Driven Engineering (MDE) to represent behavioral requirements using the use case representation. We propose a concrete syntax for an existing abstract syntax and semantics. By using MDE it is possible to clearly indentify use case elements and their relationship, creating an structured format that improves model quality.*

Resumo. *Problemas na elicitação e documentação de requisitos podem afetar todo o desenvolvimento do software. Dentre os possíveis problemas estão ambiguidade, inconsistência, conteúdo fora do padrão e documentação não clara. Um dos meios de evitar tais problemas é através de técnicas que buscam reduzir a informalidade deste processo. Neste trabalho foram aplicados conceitos de Engenharia Dirigida por Modelos (MDE) para representar requisitos funcionais seguindo a representação de caso de uso, propondo uma sintaxe concreta para uma sintaxe abstrata e semântica já existentes. O uso de MDE permite uma identificação mais clara dos elementos do caso de uso e suas relações, criando um formato estruturado de modo a aumentar a qualidade do modelo.*

1. Introdução

A Engenharia de Requisitos (ER) é uma abordagem sistemática e disciplinada para tratar dos requisitos de um sistema [Pohl and Rupp 2011]. A execução adequada das atividades de ER ajuda a reduzir falhas no processo de desenvolvimento de software [Pohl and Rupp 2011]. Porém a ER pode ser afetada por diversos problemas. No caso da documentação, que é uma das atividades principais da ER, alguns problemas da representação de requisitos estão relacionados à informalidade da linguagem (por exemplo, ambiguidade, documentação fora do padrão e inconsistência).

O objetivo deste trabalho é minimizar esses problemas. Nele foram aplicados conceitos da Engenharia Dirigida por Modelos (*Model Driven Engineering*, MDE) para apresentar uma proposta de sintaxe concreta para uma representação de requisitos: o caso de uso. A sintaxe concreta é baseada em uma semântica e sintaxe abstrata já existentes [Nguyen et al. 2015] e é capaz de representar casos de uso típicos de uma forma padronizada em uma ferramenta, denominada UCWriter, desenvolvida como um dos resultados desta pesquisa.

Este trabalho está organizado nas seguintes seções: Fundamentação Teórica, Metodologia, Resultados e Conclusões.

2. Fundamentação Teórica

Caso de uso é uma representação de requisitos funcionais, proposto para ser entendido por pessoas com e sem conhecimento em desenvolvimento de software [Cockburn 2000, Tiwari and Gupta 2015]. Uma representação textual de caso de uso é composta principalmente pelos seus atores, pré-condições, pós-condições e fluxos de eventos [Cockburn 2000, Siqueira and Muniz Silva 2011]. Em especial, os fluxos de eventos descrevem as interação dos atores com o sistema, podendo levar a um sucesso ou fracasso no objetivo do caso de uso.

A MDE, permite trabalhar com modelos que podem ser usados para automatizar e agilizar processos. Um conceito central da MDE é o de metamodelo, que é um modelo de uma linguagem de modelagem [Clark et al. 2015]. Tipicamente ele é composto por três partes: (1) a sintaxe abstrata, que define os elementos e suas relações; (2) a semântica, que define o significado de todos os elementos e relações da sintaxe abstrata; (3) a sintaxe concreta, que é a forma como o metamodelo é expresso pelo usuário para a criação de modelos [Clark et al. 2015].

3. Metodologia

Esta pesquisa foi organizada nas etapas de: revisão bibliográfica, escolha da sintaxe abstrata e semântica, seleção do *framework*, e produção e implementação da sintaxe concreta de forma iterativa.

Na revisão bibliográfica foram pesquisados trabalhos relacionados no IEEEXplorer, ACM e SpringerLink. Selecionando os 10 trabalhos mais relevantes, datados a partir de 2015, utilizando a mesma *string* de busca[1]: `("metamodel" OR "meta model"-OR "meta-model" OR "abstract syntax") NEAR/1 "use case"`. Devido ao grande número de resultados na ACM, apenas nesta base foi feita uma segunda busca aplicando os mesmos critérios apenas para o *abstract*.

Na revisão bibliográfica foram encontrados 5 metamodelos: UCMeta [Yue et al. 2015], UML Extended [Misbhauddin and Alshayeb 2015], GUIMeta [Nguyen et al. 2015], URN, estendido no TimedURN [Aprajita et al. 2017] e o PUM [Hajri et al. 2015]. Destes modelos foi avaliado o escopo da representação, eliminando o PUM, que tem o foco em linha de produto (já que não se desejava um metamodelo de domínio específico). Dos restantes: o UCMeta possui uma ferramenta que utiliza processamento de linguagem natural, e os modelos URN e UML Extended possuem propostas de sintaxe concreta gráfica. Portanto foi selecionado o GUIMeta por não possuir nenhuma proposta e condizer com o escopo deste trabalho. A sintaxe abstrata usada, baseada no GUIMeta, é apresentado na Figura 1. Ela possui algumas alterações do original devido a questões tecnológicas (como a dificuldade de implementar paralelismo) e falta de detalhamento (*RepeatingStep* com composição de *UseCaseSteps*). Além disso, a metaclasse *Step* foi adicionada para impedir a presença de *ExtensionSteps* no fluxo principal.

[1]Na ACM o *NEAR* foi substituído por *AND* por limitações do site.

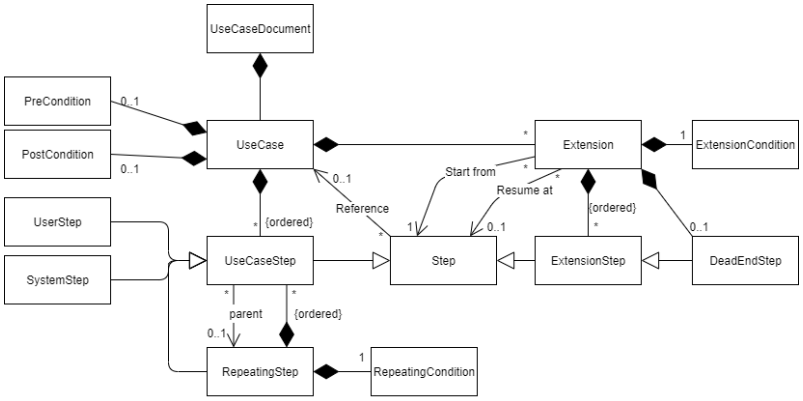

Figura 1. Sintaxe abstrata alterada para a implementação.

A implementação foi realizada de forma iterativa e concomitante ao projeto da sintaxe concreta. Isso permitiu que as etapas fossem testadas independentemente, e aprimorar os detalhes da sintaxe concreta conforme a complexidade da representação crescia. Como primeira etapa foi implementado o fluxo principal (sem *RepeatingStep*), posteriormente os fluxos alternativos, as demais partes do metamodelo (condições e *RepeatingStep*) e por fim as funcionalidades de verificação.

4. Resultados e Conclusões

O resultado deste trabalho foi a ferramenta *open source* UCWriter, a qual usa o Xtext para implementar a proposta de sintaxe concreta para o GUIMeta. Ela pode ser usada como um *plugin* no Eclipse ou através de um navegador, como apresentado na Figura 2. A ferramenta também é capaz de verificar: (1) ordem dos *Steps* e unicidade nos números; (2) se os objetos apontados existem; (3) unicidade no nome das *Extension*; (4) se um caso de uso referencia a si mesmo, (*Reference*) criando um laço infinito.

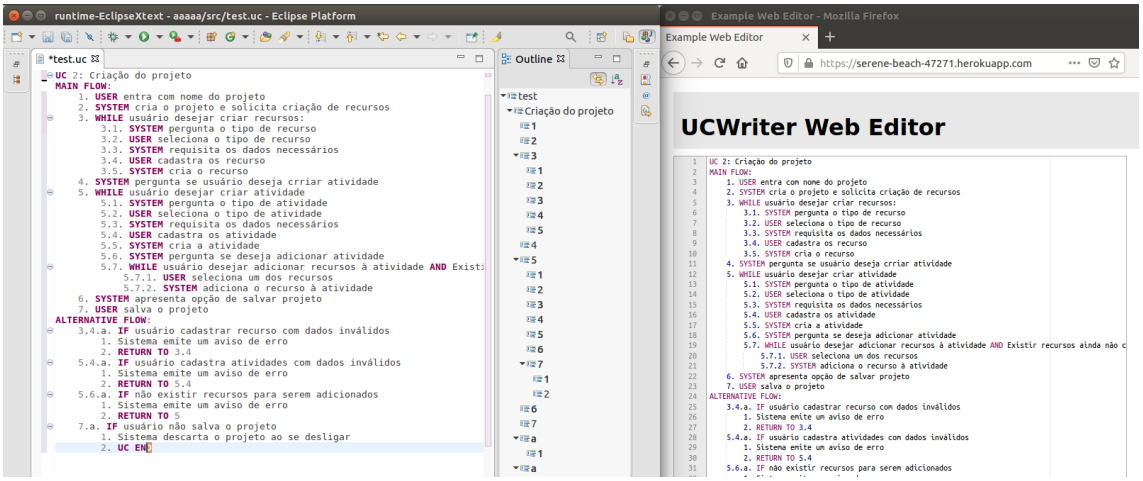

Figura 2. Exemplo de uso com UCWriter: a direta em Eclipse e a esquerda e web

Foram realizados testes para avaliar a representação de casos de uso típicos, usando exemplos do Cockburn [Cockburn 2000] e um criado especificadamente para o teste (apresentado na Figura 2). A única dificuldade encontrada foi a de representar fluxos alternativos que se referem a mais de um *Step* (como extensões globais).

Diferentemente de outras propostas, como o ZenUCM[2] que só pode ser utilizada para estudos com devida autorização dos desenvolvedores, o UCWriter é *open source* e está disponível para testes e melhorias[3].

Como trabalhos futuros pode-se melhorar a sintaxe abstrata para corrigir os problemas já identificados, realizar testes com problemas reais e avaliar características de qualidade, como a usabilidade. Um outro trabalho é avançar na representação de modelos e permitir a geração de outros modelos através de um caso de uso.

5. Agradecimentos

Este trabalho foi apoiado através do processo nº 2019/12641-7, Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP).

Referências

Aprajita, Luthra, S., and Mussbacher, G. (2017). Specifying evolving requirements models with timedurn. In *2017 IEEE/ACM 9th International Workshop on Modelling in Software Engineering (MiSE)*, pages 26–32.

Clark, T., Sammut, P., and Willans, J. S. (2015). Applied metamodelling: A foundation for language driven development (third edition). *CoRR*, abs/1505.00149.

Cockburn, A. (2000). *Writing effective use cases*. Addison-Wesley Professional.

Hajri, I., Goknil, A., Briand, L. C., and Stephany, T. (2015). Applying product line use case modeling in an industrial automotive embedded system: Lessons learned and a refined approach. In *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 338–347.

Misbhauddin, M. and Alshayeb, M. (2015). Extending the uml use case metamodel with behavioral information to facilitate model analysis and interchange. *Software & Systems Modeling*, 14(2):813–838.

Nguyen, T. H., Grundy, J., and Almorsy, M. (2015). Integrating goal-oriented and use case-based requirements engineering: The missing link. In *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 328–337. IEEE.

Pohl, K. and Rupp, C. (2011). *Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam - Foundation Level - IREB Compliant*. Rocky Nook, 1st edition.

Siqueira, F. L. and Muniz Silva, P. S. (2011). An Essential Textual Use Case Meta-model Based on an Analysis of Existing Proposals. pages 419–430, Rio de Janeiro.

Tiwari, S. and Gupta, A. (2015). A systematic literature review of use case specifications research. *Information and Software Technology*, 67:128–158.

Yue, T., Briand, L. C., and Labiche, Y. (2015). atoucan: an automated framework to derive uml analysis models from use case models. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 24(3):1–52.

[2] `http://zen-tools.com/tools/zen-rucm.html`

[3] `https://github.com/JoseColombini/UCWriter`