Proposta de Linguagem de Modelagem Gráfica com o Sirius: Uma Versão Open Source da DSL Canopus

João Carbonell, Bruno Medeiros, Pedro Silva,
Elder Rodrigues, Maicon Bernardino

[1]Universidade Federal do Pampa (UNIPAMPA)
Código Postal 97.546-550 – Alegrete – RS – Brasil

{joaocarbonellpc,brunobragamedeiros}@gmail.com, peu06@hotmail.com
elderrodrigues@unipampa.edu.br, bernardino@acm.org

Abstract. *Domain-Specific Languages (DSL) are programming languages developed to solve a specific problem of a domain, which can be represented graphically and/or textually. Thus, considering that many DSL development supporting tools are under commercial license, this study aims to propose the implementation of a new version of a DSL for modeling performance testing, called Canopus. For this, the study presents a proposal of a graphical DSL developed based on open source technologies, highlighting its requirements and design decisions, as well as a vision of the Eclipse Sirius modeling language development framework.*

Resumo. *Linguagem Específica de Domínio (Domain-Specific Language-DSL) é uma linguagem desenvolvida com o intuito de resolver um problema específico de um domínio, podendo ser representada de forma gráfica e/ou textual. Assim, considerando que muito do ferramental de apoio à criação de uma DSL é pago, este estudo tem como objetivo propor a implementação de uma nova versão da DSL Canopus para modelagem de testes de desempenho. Para isso, o estudo apresenta uma proposta de DSL gráfica desenvolvida com base em tecnologias open source, discutindo seus requisitos e decisões de projeto, bem como apresentando uma visão do framework de desenvolvimento de linguagem de modelagem Eclipse Sirius.*

1. Introdução

Linguagem específica de domínio (*Domain-Specific Language* - DSL) [Fowler 2010] é uma técnica para o desenvolvimento de sistemas computacionais com características particulares de um determinado domínio. Sua utilização é ampla, estando no desenvolvimento de aplicações relacionadas a campo geológico [De Sousa and Da Silva 2016] até a uma linguagem desenvolvida como auxílio a comunicação com banco de dados, *e.g.* SQL. O desenvolvimento de uma DSL é uma tarefa que envolve conhecimento teórico e prático, sendo necessário o entendimento do funcionamento de linguagens de programação de propósito geral para criar uma linguagem específica de domínio, como conhecimentos de gramática, sintaxe e semântica. Além disso, para o desenvolvimento de uma DSL devem ser utilizadas ferramentas que auxiliem no processo de criação da linguagem, fornecendo as funções para que seja possível a utilização da DSL ao seu usuário final. Estas ferramentas são conhecidas como *Language Workbench* (LW). Entretanto, muitas destas

LW's estão disponíveis comercialmente, gerando linguagens restritas e financeiramente custosas.

Um exemplo de DSL desenvolvida usando um LW comercial é a DSL Canopus [Bernardino et al. 2016], a qual foi criada com a MetaEdit+ da MetaCase[1]. A Canopus tem como objetivo a modelagem de testes de desempenho e geração automática de cenários e *scripts* de testes. Ela foi desenvolvida junto à indústria, por meio de uma parceria entre os pesquisadores e uma multinacional de TI. Apesar de considerarem a experiência com a MetaEdit+ satisfatória, tanto do ponto de vista dos pesquisadores ao criarem a linguagem, quanto do ponto de vista da empresa, o alto custo das licenças foi preponderante para a não adoção da linguagem.

Este estudo propõe uma nova versão *open source* da Canopus, com o propósito de disponibilizá-la em repositórios abertos a fim de que outros grupos de pesquisa e/ou comunidades de desenvolvedores possam contribuir, bem como incentivar que empresas a adotem na modelagem dos testes de desempenho.

Neste estudo, serão discutidos os requisitos e decisões de projetos necessários para a criação de desta nova versão *open source* da DSL gráfica, assim como uma visão geral dos conceitos e funcionamentos do *framework* para a modelagem Sirius, projeto do Eclipse Foundation [Eclipse 2017]. Este estudo foi organizado da seguinte forma: Seção 2 discute a fundamentação relacionada a DSL, *Language Wokbench* (LW) e trabalhos relacionados, bem como apresenta a Canopus. A Seção 3 discute a proposta da versão *open source* da Canopus, destacando seus requisitos e decisões de projeto para sua implementação. Por fim, a Seção 4 apresenta as conclusões relacionadas ao estudo e futuros passos em direção a nova versão gráfica da DSL.

2. Fundamentação Teórica

Linguagem Específica de Domínio (*Domain-Specific Language*-DSL) é uma linguagem de programação de computador, a qual têm foco no domínio de aplicação [Fowler 2010]. Focada em um domínio particular. Uma DSL pode ser dividida em duas categorias: **DSL Interna**: Linguagem que, geralmente, utiliza a estrutura da linguagem de propósito geral hospedeira, auxiliando em uma tarefa específica. Um exemplo: muitos dos mecanismos usados pelo Rails vêm do Ruby [Fowler 2010]. **DSL Externa**: Linguagem separada da linguagem host de prooósito geral(*General Purpose Language* - GPL), que realiza tarefas de apoio, mas de forma independente. Um exemplo: SQL (*Structured Query Language*) utilizada por outras linguagens, como o JAVA, para consulta e comunicação com bancos de dados. **Language Workbench (LW)**: São ferramentas que proporcionam um ambiente para a criação e suporte de DSL, por meio do desenvolvimento de metamodelos [Fowler 2010]. Suportando a criação, edição e manutenção de DSL, bem como uma IDE, que oferece um ambiente propicio à manipulação de GPL.

Um exemplo de LW que auxilia a notação gráfica de uma DSL é a MetaEdit+, que tem como objetivo tornar mais fácil a modelagem de domínios, possibilitando que os usuários se concentrem na criação de linguagens e modelos mais expressivos. Outra LW é a Whole Plataform, que é uma solução *open source* utilizada no auxílio de programação orientada a linguagem, mas que atualmente encontra-se sem suporte. Existe, também, o

[1]MetaEdit+ http://www.metacase.com/mwb/

framework Eclipse Sirius, cujas informações encontram-se mais detalhadas na Seção 3.3. Além de ferramentas que auxiliam no desenvolvimento da notação gráfica, existem ferramentas que contribuem no desenvolvimento de DSL textuais. Um exemplo é o Xtext, um *framework* da Eclipse Foundation para criação de novas linguagens usando representação textual.

2.1. Trabalhos Relacionados

Nos trabalhos relacionados, nota-se que existe uma carência de ferramentas gráficas *open source* para testes de desempenho. Existem alguns estudos propostos com o *framework* Sirius [Jäger et al. 2016] [Vujović et al. 2014], mas que não são para o domínio de teste de desempenho. O estudo de Jäger [Jäger et al. 2016] apresenta uma DSL gráfica, desenvolvida com a Sirius e o *Eclipse Modeling Framework*. A DSL foi proposta para um sistema de trem de brinquedos, visando ensinar a implementação de sistemas embarcados em tempo real, projetando um bloco central para controlar os trens e trocar de acordo com uma programação pré-definida. Foi relatado que, o conjunto de modelos específicos de domínio, editor gráfico, modelos de sistemas e simulações, levaram em torno de três dias para serem projetados e implementados com a abordagem proposta.

Já no estudo [Vujović et al. 2014], o objetivo é apresentar um editor gráfico de modelos específicos de domínio. Como exemplo de uso do editor gráfico, foi modelado um sensor RESTful para web. Cada elemento gráfico modelado representa um certo elemento dos sensores RESTful da web ou um serviço (elementos como números de nós sensores, métodos, abordagens) e é descrito pelas propriedades que descrevem esta condição. O estudo conclui que, desenvolvedores sem experiência com o domínio ou a solução específica, puderam facilmente definir tarefas e implementar serviços RESTFul.

2.2. Canopus

A Canopus [Bernardino et al. 2016] consiste em uma DSL gráfica e textual para modelagem de testes de desempenho, criada com a LW MetaEdit+. A Canopus faz usos de processos relacionados a abordagem de testes baseados em modelos (*Model-Based Testing* - MBT), a fim de melhorar e aperfeiçoar a modelagem e criação dos testes de desempenho. Assim, sendo possível gerar *scripts* por meio de modelos de desempenho que imitam a interação do usuário virtual (*Virtual User* - VU) com o sistema sob teste (*System Under Test* - SUT), então permitindo gerar representações textuais destes modelos ou *scripts*, os quais podem ser executados por ferramentas de desempenho, tais como o HP LoadRunner. Além disso, é possível gerar arquivos XML, para uso em outras ferramentas de teste de desempenho, como a PLeTsPerf [Rodrigues et al. 2015], capaz de interpretar o XML para gerar automaticamente cenários e *scripts* de testes de desempenho.

Para definir a Canopus, foi criado um metamodelo para a criação de modelos de testes de desempenho. Este metamodelo é composto por metatipos, os quais são usados para implementar uma DSL específica. Os metamodelos da Canopus, representados por 7 pacotes. São eles: *Modelo de Desempenho Canopus*, *Monitoramento de Desempenho Canopus*, *Cenário de Desempenho Canopus*, *Scripting de Desempenho Canopus*, *Métricas de Desempenho Canopus*, *Carga de Trabalho Canopus*, *Arquivos Externos de Desempenho Canopus*. Sendo que o Monitoramento de Desempenho Canopus, Cenário de Desempenho Canopus e *Scripting* de Desempenho Canopus, são os metamodelos principais, que juntos compõe o Modelo de Desempenho Canopus.

3. Proposta de DSL

Como apresentado anteriormente, a Canopus foi desenvolvida dentro da LW MetaEdit+, uma ferramenta de licença comercial. Este fator torna custoso financeiramente a aquisição e utilização da Canopus. Por isto, foi tomada a decisão de desenvolver uma nova versão da Canopus, a qual possuísse as mesmas características da sua versão antecessora, embora novas características possam ser adicionadas, mas que fundamentalmente estivesse disponível sob uma licença *open source*. Este trabalho apresenta uma proposta para a criação da representação gráfica da Canopus. Assim, primeiramente, será abordado o levantamento dos requisitos para a linguagem. Depois, as decisões de projeto da DSL serão discutidas, bem como as tecnologias disponíveis e desafios encontrados para a implementação da nova versão *open source* da Canopus.

3.1. Requisitos da Linguagem

Esta seção enumera os requisitos aplicados a Canopus, além de novos requisitos contemplados para a proposta da nova versão *open source* da DSL. A maioria dos requisitos foram previamente estabelecidos no estudo de Análise de Requisitos e Decisões de Projeto para a Canopus [Bernardino et al. 2014]. Nesta seção consta a releitura destes requisitos com algumas melhorias e apresenta alguns novos requisitos da linguagem.

RQ1) *A DSL deve permitir representar as características do teste de desempenho.* Uma das principais funções de um teste de desempenho é a identificação de limitações do sistema. Com isso, é necessário que as aplicações sejam medidas em pequenas partes chamadas transações. Dessa forma, cada atividade do sistema pode ser medida.

RQ2) *A linguagem deve ser implementada e disponibilizada sob uma licença open source.* Como principal mudança em relação a primeira versão da Canopus, qualquer ferramenta utilizada na implementação da linguagem deve ser *open source*, permitindo que ela seja disponibilizada em repositórios abertos a comunidade, academia e indústria.

RQ3) *A DSL deve prover uma representação gráfica das características do teste de desempenho.* Esse requisito não diz respeito a linguagem, mas sim a ferramenta utilizada para sua criação. Dessa forma, a ferramenta deve ser capaz de fornecer mecanismos de representar graficamente todas as características do domínio.

RQ4) *A DSL deve prover uma representação textual.* Dessa forma, engenheiros de testes, que estão acostumados com representação textual, irão ter mais facilidade em adotar a DSL, bem como documentar suas aplicações. A linguagem deve ter características e palavras-chaves que lembram o domínio de teste de desempenho.

RQ5) *A DSL deve incluir recursos que ilustrem os contadores de desempenho.* Dessa forma, é possível analisar o nível de qualidade da aplicação e a infraestrutura do servidor que a hospeda por meio de indicadores e métricas de desempenho.

RQ6) *A DSL deve permitir modelar comportamentos de diferentes perfis de usuários.* Esse requisito é específico do domínio de teste de desempenho. Esses comportamentos são modelados de acordo com os sistemas que estão em testes. Vale ressaltar que o foco da Canopus é para o domínio de aplicações Web.

RQ7) *A DSL deve realizar a rastreabilidade entre os elementos das notações gráficos e textuais.* A DSL proposta deve ser capaz de automatizar a conversão/tradução dos elementos gráficos para as suas contrapartes textuais e vice-versa.

RQ8) *A DSL deve permitir a exportação de modelos para formatos de tecnologias específicas.* Com esse requisito é possível garantir que a DSL proposta exporte seus modelos para ferramentas de testes de desempenho, *e.g.* HP LoadRunner, Apache JMeter.

RQ9) *A DSL deve exportar os modelos para o formato de arquivo XML.* Com isso, é possível que qualquer pessoa que queira usar as informações dos modelos Canopus, tenha apenas que interpretar o XML gerado para a sua tecnologia.

RQ10) *A DSL deve representar diferentes elementos do teste de desempenho em scripts de testes.* O diagrama modelado usando a DSL proposta deve representar múltiplos elementos de *scripts* de teste, tais como fluxos condicionais ou de repetição, entre outros.

RQ11) *A DSL deve permitir a modelagem de múltiplos cenários de testes.* Dessa forma, a DSL será capaz de modelar cenários de testes que representem cargas de trabalho normais, bem como cargas de estresse. Além disso, deve permitir a possibilidade de gerenciar a hierarquia de cenários, que é a decomposição de um cenário de teste em outros múltiplos cenários, a fim de permitir maior reuso entre eles.

3.2. Decisões de Projeto

Esta seção descreve as decisões de projeto para a criação da nova versão da Canopus, relacionadas aos requisitos mencionados na Seção 3.1. Assim, para cada decisão será apresentado seu(s) respectivo(s) requisito(s).

DD1) *O uso de soluções open source no auxílio da implementação de DSL gráficas* (RQ2, RQ3). Estes requisitos foram atendidos por meio de um mapeamento da literatura realizado, em que foram encontradas algumas ferramentas que atendiam a este requisito. Sendo assim, foi decidido pelo uso do *framework* Eclipse Sirius, de código aberto e que apoia a criação de DSL com notação gráfica.

DD2) *As características do domínio de teste de desempenho devem ser aplicadas de forma incremental* (RQ1, RQ5). Para estes requisitos foi adotado a estratégia de usar uma ontologia [Freitas and Vieira 2014], a fim de determinar e identificar as características que representam o domínio do teste de desempenho. Esta ontologia determina conceitos básicos, relações e restrições do domínio do problema.

DD3) *Fornecer uma linguagem gráfica que represente o comportamento de perfis de usuários para diferentes cenários de testes* (RQ6, RQ11). Foram analisados diferentes modelos e representações gráficas que auxiliam testes de desempenho para atender esses requisitos. A linguagem deve ter elementos capaz de representar o comportamento de diferentes perfis de usuários e que também representem as configurações dos cenários de teste, como as informações sobre o domínio (duração dos testes, tempo de processamento). Também a possibilidade de incluir a escolha aleatória e execução de probabilidades para as interações dos usuários virtuais. Além disso, deve ter a funcionalidade de representar de forma abstrata os dados que devem ser instanciados nas atividades do processo de teste de desempenho, como a geração de *scripts* de teste.

DD4) *Criar uma representação textual em linguagem semi-natural.* Para facilitar o entendimento dos testes por parte da equipe e dos intervenientes, deve-se permitir a representação textual em linguagem semi-natural (RQ4). Linguagens que utilizam linguagens naturais, como a Gherkin [Wynne and Hellesøy 2012], as quais permitem descrever testes funcionais, não costumam oferecer mecanismos para avaliar o teste de de-

sempenho. Por isto, pretende-se incluir as características do teste de desempenho visando estender esta linguagem. A Gherkin é comumente usada para ser interpretada por uma ferramenta de linha de comando chamada Cucumber, a qual automatiza a execução do teste de aceitação.

DD5) *Permitir a rastreabilidade entre as notações gráficas e textuais* (RQ3, RQ4, RQ10). A LW deve permitir a conversão/tradução entre as regras dos modelos. Ou seja, a modelagem bidirecional entre os elementos gráficos e seus respectivos ativos na notação textual devem ser mapeadas. Não sendo este mapeamento apenas de um para um, mas que permita que uma notação gráfica seja mapeada para várias instâncias textuais. Como não existe uma única ferramenta *open source* que atenda a isso, será realizada uma integração entre os *frameworks* Sirius [Eclipse 2017] e Xtext [Efftinge and Völter 2006].

DD6) *Integração entre a DSL e outras tecnologias (RQ8, RQ9).* Deve permitir a exportação dos modelos, cenários e *scripts* modelados para outros formatos, tanto formatos genéricos, *e.g.* XML, quanto formatos específicos, tal como HP LoadRunner.

3.3. Framework Eclipse Sirius

O desenvolvimento de uma DSL gráfica requer ferramentas que auxiliem o processo de criação de seus elementos. Após a pesquisa em estudos da área de desenvolvimento de DSL, foi encontrado um *framework* para auxiliar no desenvolvimento da versão *open source* da Canopus, o *framework* Sirius [Eclipse 2017]. O *framework* Sirius é um projeto da *Eclipse Foundation*, cujo objetivo é permitir a criação de ferramentas de modelagem. De maneira geral, o Sirius é utilizado para elaborar e analisar um sistema a fim de enriquecer a comunicação com outros membros da equipe, parceiros ou clientes [Eclipse 2017]. No Sirius, os editores são definidos por um modelo que tem o objetivo de estabelecer o comportamento e todas as ferramentas de edição e navegação utilizadas para adaptar uma DSL.

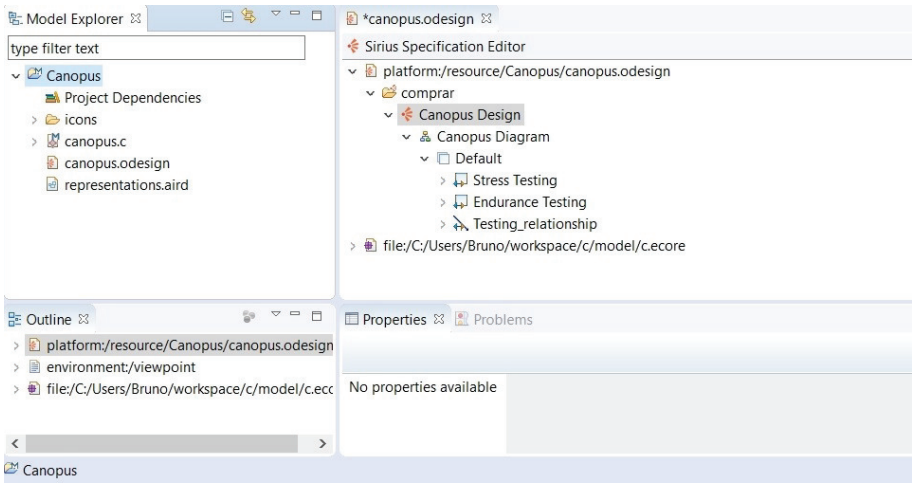

Figura 1. Interface da Ferramenta Sirius

Pode ser observado na Figura 1 o início da nova versão da Canopus. Ressalta-se que o estudo ainda está em andamento, porém é importante dissertar sobre o funcionamento da ferramenta e suas características em relação a DSL abordada no estudo. Ao lado esquerdo da Figura 1, na aba denominada `Model Explorer`, encontram-se os

arquivos originados na criação do projeto Canopus. A modelagem da DSL é realizada no arquivo de extensão `.odesign`. A tarefa de modelagem é feita na aba denominada `Sirius Specification Editor`, situada na parte direita da imagem. Todos os elementos gráficos da DSL, suas interações e representações são trabalhadas nessa aba. Como pode ser visto na Figura 1, a representação gráfica da Canopus possui três elementos gráficos. Dentro do pacote `canopus`, existe o `Canopus Diagram`, em que os elementos gráficos são inseridos. Esse diagrama possui três elementos: "Stress Testing" e "Endurance Testing"; além da conexão entre os dois, chamada de "Testing Relationship".

Os elementos gráficos contidos na modelagem de uma DSL dentro do *framework* Sirius têm suas características descritas na aba `Properties`, em que, para cada elemento, o desenvolvedor deve inserir desde características gráficas à lógicas de relacionamento entre elementos. Na aba denominada `Problems`, os problemas descobertos durante a validação da modelagem são expostos. O *framework* Sirius possui mecanismos para especificar a representação dos modelos em formas de diagramas, tabelas e árvores. Os diagramas são utilizados para modelagem gráfica de informações; as árvores, utilizadas em representações hierárquicas; e as tabelas têm o intuito de representar elementos posicionados em uma matriz, formada por linhas e colunas. Além disso, dentro de um elemento de modelagem podem ser criados "Regras de Validação", em que as informações de dados inseridos na modelagem dos elementos podem ser verificadas. Sendo assim, o *framework* Sirius apresenta uma série de características que possibilita o desenvolvimento da nova versão *open source* da Canopus.

4. Conclusão

Diante da dificuldade em encontrar ferramentas que auxiliem o processo de criação de DSL gráficas e que também seja de licença gratuita, o *framework* Sirius é uma solução que oferece a maioria das funcionalidades necessárias a implementação de linguagens gráficas, possuindo uma interface familiar a quem utiliza o ambiente Eclipse e abrindo a possibilidade de integração com outros *frameworks*. Nesse estudo foi apresentado uma proposta de implementação de uma versão *open source* da Canopus. Assim, foi apresentado uma visão geral sobre a Canopus e o *framework* de desenvolvimento Sirius. Além de serem levantados e discutidos os requisitos necessários para criação dessa versão da DSL, seus requisitos originais e os da nova versão foram relatados e debatidos, bem como foram discutidas as decisões de projetos necessárias para atender aos requisitos. Como trabalho futuro, será realizada a implementação da versão *open source* da Canopus. Outro ponto importante a ser destacado é, que além da implementação gráfica, a textual também será desenvolvida. Portanto, será necessário implementar a bidirecionalidade na conversão entre as notações, transformando a Canopus em uma DSL gráfica e textual, por intermédio da integração dos *frameworks* Sirius com o Xtext.

Referências

Bernardino, M., Zorzo, A., Rodrigues, E., Oliveira, F., and Saad, R. (2014). A domain-specific language for modeling performance testing: requirements analysis and design decisions. In *9th Int. Conf. on Software Engineering Advances*, pages 609–614.

Bernardino, M., Zorzo, A. F., and Rodrigues, E. M. (2016). Canopus: A domain-specific language for modeling performance testing. In *Int. Conf. on Software Testing, Verification and Validation*, pages 157–167, Chicago, USA.

De Sousa, L. M. and Da Silva, A. R. (2016). A domain specific language for spatial simulation scenarios. *GeoInformatica*, 20(1):117–149.

Eclipse (2017). Framework Eclipse Sirius. `https://eclipse.org/sirius/`.

Efftinge, S. and Völter, M. (2006). oAW xText: A framework for textual DSLs. In *Workshop on Modeling Symposium at Eclipse Summit*, volume 32, page 118.

Fowler, M. (2010). *Domain-specific languages*. Pearson Education.

Freitas, A. and Vieira, R. (2014). An ontology for guiding performance testing. In *IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies*, pages 400–407. IEEE Computer Society.

Jäger, S., Maschotta, R., Jungebloud, T., Wichmann, A., and Zimmermann, A. (2016). Creation of domain-specific languages for executable system models with the eclipse modeling project. In *Systems Conference (SysCon), 2016 Annual IEEE*, pages 1–8. IEEE.

Rodrigues, E., Bernardino, M., Costa, L., Zorzo, A., and Oliveira, F. (2015). PLeTsPerf-A Model-Based Performance Testing Tool. In *8th Int. Conf. on Software Testing, Verification and Validation*, pages 1–8.

Vujović, V., Maksimović, M., and Perisić, B. (2014). Sirius: A rapid development of dsm graphical editor. In *18th Int. Conf. on Intelligent Engineering Systems*, pages 233–238.

Wynne, M. and Hellesøy, A. (2012). *The Cucumber Book: Behaviour-Driven Development for Testers and Developers*. The Pragmatic Bookshelf.