WebAtlas: Uma ferramenta de Modelagem de Características

Lucas S. Corrêa[1], **Luciano Marchezan**[1], **Elder Rodrigues**[1]

[1]Laboratory of Empirical Studies in Software Engineering
Universidade Federal do Pampa - (UNIPAMPA)
Caixa Postal 15.064 - 97.546-550 - Alegrete - RS - Brasil

{eslucascorrea,lucianomarchp}@gmail.com,

elder.rodrigues@unipampa.edu.br

Abstract. *Software product line (SPL) is an approach used by industries to systematically generate several products with lower costs, shorter time-to-market and higher quality. Among the stages of the SPL development process, is the feature model, which consists of presenting the attributes of a given SPL. Currently, there are several tools that give support for feature modeling. However, there is lack tools focusing on teaching feature modeling. This work presents WebAtlas, an open-source tool that aids students to learn SPL concepts. To initially evaluate the tool, a comparison between WebAtlas and other tools was conducted to measure time and effort creating the feature model.*

Resumo. *Linha de produto de software (LPS) é uma abordagem que vem sendo utilizada pelas indústrias para se gerar sistematicamente diversos produtos com custos mais baixos, tempos de produção menor e maior qualidade. Dentre as etapas do processo de desenvolvimento de LPS existe a modelagem de características. Existem diversas ferramentas que podem auxiliar no processo de modelagem. Porém, identificou-se que não existem ferramentas voltadas para o ensino de modelagem de LPS. Este trabalho apresenta a ferramenta WebAtlas que tem como objetivo ser um ferramenta open-source para modelagem de características com o intuito de auxiliar o aprendizado dos conceitos de LPS. Para uma avaliação inicial da ferramenta, foi executada uma comparação da WebAtlas com outras ferramentas de modelagem para se medir o tempo e o esforço para se elaborar um modelo de características.*

1. Introdução

Quando tratamos de desenvolvimento de software, estamos visando as melhores técnicas, abordagens e ferramentas com a finalidade de nos auxiliar em tarefas que a envolve. Empresas estão sempre em busca de melhores práticas de desenvolvimento e gerenciamento, onde visam diminuir custos, tempo de produção e aumentar a qualidade e robustez de seus produtos. Linha de Produto de Software (LPS) é uma abordagem que surgiu com métodos e técnicas eficazes para produção de uma família de sistemas de um mesmo domínio, compartilhando artefatos como: arquitetura, decisões de projeto, planos de teste, entre outros. Durante os últimos anos, técnicas de modelagem e ferramentas surgiram para dar suporte durante a modelagem da variabilidade da família de sistemas [Pohl et al. 2005]. Entretanto, segundo [Acher et al. 2017], LPS não faz parte do currículo nem é um tópico estudado nas Instituições de Ensino Superior. Acher et al. 2017 cita que ensinar LPS é

um desafio, por ser um tema recente e não possuir métodos sendo comprovados, além da falta de ferramentas acadêmicas. Este é um dos principais problemas encontrados por não ser um tópico abordado nas Instituições de Ensino Superior e ter pouco material.

Com o intuito de verificar o maior número de ferramentas, foi executada uma busca na literatura com o intuito de analisar ferramentas de modelagem de características para LPS. Como resultado, identificou-se que o cenário atual possui diversas ferramentas, porém dentre estas, nenhuma tem foco voltado ao auxilio do aprendizado de LPS. Devido a este deficit, foi desenvolvida uma ferramenta de modelagem de características, WebAtlas, voltada para o meio acadêmico, auxiliando o ensino de LPS. A WebAtlas permite ao usuário criar e editar modelos de características de acordo com diversas notações. Além disso, professores poderão criar e avaliar atividades realizadas por alunos.

Neste trabalho é apresentada uma avaliação inicial que foi executada comparando o *UI Design* das ferramentas encontradas na literatura com a WebAtlas. O objetivo era comparar a eficiência das ferramentas durante a criação de modelos de características e elaboração de restrições. A avaliação é uma replicação de um estudo proposto em [El Dammagh and De Troyer 2011]. Através desta análise foi observada a capacidade das ferramentas, além de se verificar requisitos de usabilidade funcional como *Feedback*, *Undo*, *shortcuts*, entre outros. Os resultados serão utilizados como base para aprimorar a WebAtlas no ponto de vista de usabilidade. Este trabalho está organizado da seguinte maneira: a Seção 2 apresenta os conceitos utilizados na pesquisa; a Seção 3 apresenta a ferramenta WebAtlas e suas funcionalidades; a Seção 4 apresenta a busca realizada na literatura por ferramentas semelhantes assim como a avaliação comparativa entre elas com a WebAtlas; por fim a Seção 5 apresenta as conclusões do trabalho.

2. Fundamentação Teórica

2.1. Linha de Produto de Software

LPS é uma técnica de produção de sistemas de uma mesma família de produto, tal técnica possibilita aumentar a produção de sistemas diminuindo os custos e prazos de sua produção [Clements and Northrop 2001]. O Software Engineering Institute (SEI) define LPS como "Uma linha de produtos de software é um conjunto de sistemas de software que compartilham o mesmo conjunto de funcionalidades comum e gerenciáveis que atendem a uma área específica de mercado ou missão e são construídos a partir de um conjunto comum de ativos principais de uma maneira prescrita."

A engenharia de software tradicional costuma implementar sistemas individuais um por vez, e com oportunidades ou necessidades acaba reutilizando os códigos. Em vez disso, a engenharia de LPS adota uma abordagem diferente, onde foca na reutilização de todos os ativos desenvolvidos como requisitos, arquitetura, códigos-fonte, entre outros artefatos durante o ciclo de desenvolvimento de um determinado software [de M. Rodrigues 2013]. Na Figura 1 podemos ver o ciclo de vida da LPS, dividida em Engenharia de Domínio e Engenharia de Aplicação. Engenharia de Domínio possui as seguintes atividades [Pohl et al. 2005]: ***Product Management*** que define o que pertence ao escopo da LPS, na entrada consiste com os objetivos da LPS e saída é um roteiro com as características comuns e variabilidade de produtos futuros; ***Domain Requirements Engineering*** que identifica o escopo do domínio juntamente com os ativos principais (requisitos, modelos, componentes, arquitetura, plano de teste, entre outros); ***Domain Design***

que define uma arquitetura de referência para a LPS e determina um plano de produção; ***Domain Realisation*** que implementa os ativos principais e ***Domain Testing*** que valida e verifica os ativos principais.

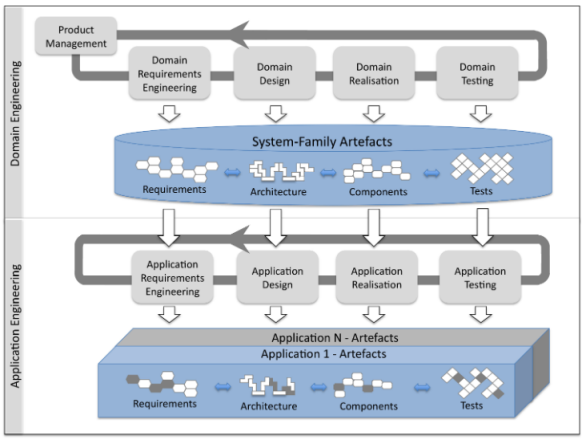

Figura 1. Ciclo do Processo de Linha de Produto de Software [Pohl et al. 2005]

Os artefatos que são gerados pela Engenharia de Domínio serão utilizados na Engenharia de Aplicação para produzir novos sistemas. As atividades da Engenharia de Aplicação são as seguintes: ***Application Requirements Engineering*** que especifica os requisitos da aplicação e realiza uma consistência entre tais requisitos e o requisitos gerados na Engenharia de Domínio, ***Application Design*** que cria uma arquitetura da aplicação utilizando a arquitetura de referência gerada no ciclo anterior, assim incorporando as adaptações específicas da aplicação, ***Application Realisation*** que produz a aplicação, utilizando a arquitetura desenvolvida na etapa anterior e os artefatos gerados na Engenharia de Domínio e ***Application Testing*** que valida e verifica a aplicação de acordo com os requisitos levantados.

2.2. Ferramentas de Modelagem de Características

Como apresentado na seção anterior, Linha de Produto de Software é um ciclo em que engloba muitas tarefas. Com o objetivo de auxiliar sua modelagem, facilitando seu entendimento e até em certos casos ajudando seu gerenciamento, ferramentas de modelagem de características foram desenvolvidas. Sua função é mapear todas as características, limitações e restrições possíveis em uma Linha de Produto de Software.

Atualmente existem diversas ferramentas disponíveis como FeatureIDE [Meinicke 2017], Pure:variants [Constantino et al. 2016], TouchCore [Schöttle et al. 2015], S.P.L.O.T [Mendonca et al. 2009], entre outras. Muitas dessas ferramentas possuem características e funcionalidades distintas, como por exemplo a S.P.L.O.T é uma ferramenta de plataforma web e com a modelagem das características de forma textual.

3. WebAtlas

WebAtlas[1] é uma ferramenta de modelagem de características, com foco principal no meio acadêmico, podendo ser utilizada para auxiliar o aprendizado de LPS. O projeto pos-

[1]WebAtlas encontra-se em: http://lesse.com.br/webatlas/

sui dois principais objetivos: a) auxiliar o aprendizado de LPS, possibilitando a criação, edição de modelos por meio de diferentes notações; b) atuar como um repositório de modelos já desenvolvidos e validados, para que possam ser utilizados como base de estudo.

3.1. Requisitos Funcionais (RF) E Requisitos Não Funcionais (RNF)

• **RF1:** Possibilidade de criar, editar ou excluir um modelo de características. • **RF2:** Possibilidade de importar/exportar um modelo de característica, inclusive de/para outras ferramentas. • **RF3:** Possibilidade de criar modelos com as notações FODA, *Generative Programming* e *Cardinality-Based*. Além de poder fazer conversão entre as notações. • **RF5:** Criar avaliações para enviar aos alunos, realizar correções e avaliar os modelos criados. • **RF6:** Enviar modelos para correção e visualizar a correção junto à nota. • **RNF1:** Visualizar um modelo através de uma representação gráfica. • **RNF2:** Possibilidade de autenticação para ter acesso a demais funcionalidades do sistema. • **RNF3:** Ser *open-source* e de fácil acesso.

3.2. Decisões de Projeto

Baseado nos requisitos apresentados foram elaboradas decisões de projeto durante a fase de planejamento. A seguir apresentamos as sete decisões e a relação com os requisitos:

1. Ferramenta deverá ser Web, possibilitando fácil acesso e não necessitando processos de instalação (**RF1**); 2. Utilizar o padrão de arquitetura *Model View Controller* (MVC) (**RF1**); 3. Utilizar do padrão estrutural *Facade* [Gamma et al. 1995], possibilitando a implementação de diferentes notações independentes (**RF3**); 4. Utilizar uma biblioteca de diagramação *open-source* (**RNF1 e RNF3**); 5. Separar a estrutura dos diagramas de sua visualização, fazendo com que a biblioteca de diagramação possa ser trocada dependendo do contexto (**RF2, RF3 e RNF1**); 6. Permitir exportar e importar modelos e diversos formatos (.pdf, .png, .xml, .json) (**R2**); 7. Apresentar comportamentos e funcionalidades acadêmicas como avaliações e fórum para discussão (**RF5 e RF6**).

A Figura 2 apresenta a ferramenta WebAtlas onde é possível observar um modelo de características elaborado na mesma com *features* obrigatórias e alternativas, além de restrições criadas para o determinado modelo.

4. Trabalhos Relacionados

As ferramentas apresentadas nesta seção foram encontradas através de uma busca em bases de dados. O objetivo foi encontrar trabalhos que apresentam ou citam ferramentas de modelagem de características.[2]

4.1. Análise Comparativa entre as Ferramentas

As análises foram baseadas no estudo de [El Dammagh and De Troyer 2011], onde os autores comparam ferramentas de modelagem de características em questões de usabilidade. Seus resultados são utilizados para se formular recomendações de *design* para autores propondo novas ferramentas. A análise foi realizada utilizando critérios de qualidade: usabilidade, segurança e suporte ao requisitos de usabilidade funcional. Tais critérios são avaliados através da criação de um modelo de características[3]. Com o intuito de mitigar

[2]O protocolo detalhado encontra-se em: http://bit.do/webatlasfiles

[3]O modelo se encontra em: http://bit.do/webatlasfiles

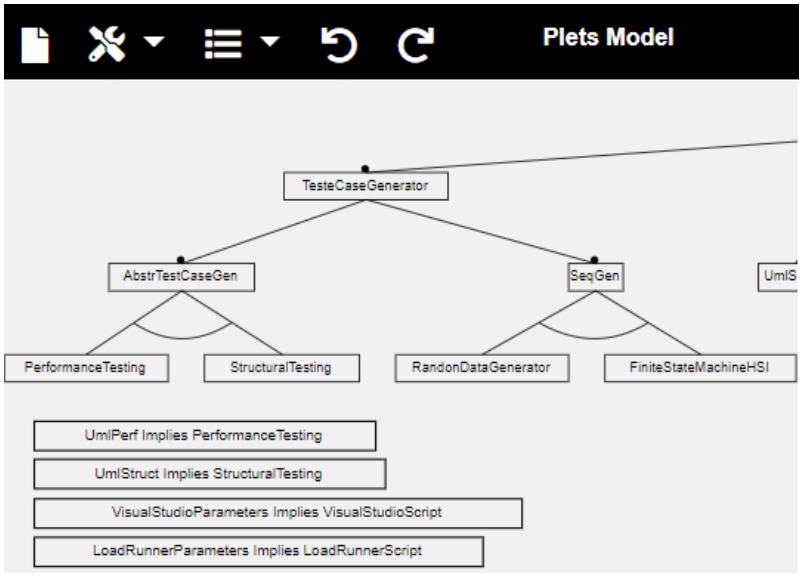

Figura 2. Modelo de Características Criado na WebAtlas

a experiência nas comparações, adoto-se a prática de aprender a manusear as ferramentas e posteriormente executar a comparação, tal prática foi possível pois foi a primeira experiência do usuário com as demais ferramentas. Com relação a experiência da WebAtlas não foi possível mitigar está ameaça. As seis ferramentas selecionadas para a comparação com WebAtlas são apresentadas na Tabela 1. Muitas ferramentas não foram analisadas por conta de se tratar de ferramentas para melhoria de restrições, melhoria de LPS, conjunto de ferramentas que auxiliam LPS, algumas ferramentas tiveram problemas na hora da instalação como por exemplo dependências ou problemas com a IDE que a suporta, links inválidos para download ou ferramentas com disponibilidade comercial.

Tabela 1. Ferramentas analisadas

ID	Ferramentas
F1	WebAtlas
F2	FeatureIDE
F3	Captain Feature
F4	Feature Diagram Editor
F5	TouchCore
F6	SPLOT

4.1.1. Critérios de Avaliação

Os critérios utilizados nas avaliações são apresentados a seguir:

- **Usabilidade:** medida através da eficiência. Segundo [El Dammagh and De Troyer 2011] eficiência é medida pelo tempo de esforço para se modelar um modelo de característica. O esforço é medido pela soma do número de cliques do mouse, número de cliques do teclado, número de pixeis percorridos pelo cursor do mouse. • **Segurança:** Foi analisada a capacidade das ferramentas quanto as suas regras de redundância, anomalias, inconsistências e semânticas inerentes a estrutura

de árvore, como por exemplo cada filho ter apenas um pai. Redundância refere-se a informações modeladas de varias maneiras e é considerada um problema leve. Anomalias refere-se a modelagem de informações sem sentido, embora não afetem a consistência do modelo, podem afetar na perda de configurações potenciais e é considerado um problema médio. Inconsistência refere-se a informações contraditórias, informações que entrem em conflito entre si, esse critério é considerado um problema grave. Semântica refere-se em verificar se as ferramentas seguem a regra da semântica de estrutura em árvore, como por exemplo um filho ter apenas um pai. Neste critério, as ferramentas foram classificadas utilizando uma escala de 0 a 3 de acordo com o suporte fornecido. Neste caso, 0 - representa não da suporte a funcionalidade até 3 - da suporte totalmente a funcionalidade. • **Recursos Funcionais:** Foram analisados 6 recursos de usabilidade funcional: *Feedback:* dar retorno ao usuário sobre informações importantes; *Undo:* opção de desfazer uma ação, afim de evitar retrabalho; Atalhos: permitir que o usuário execute ações através de atalhos do teclado; *User Expertise:* possibilidade do usuário personalizar seu ambiente de trabalho e atalhos; Reutilizar informações: permitir que o usuário reutilize informações através de funções como, copiar, colocar e recortar; Ajuda: fornecer ajuda ao usuário como dicas e tutoriais.

Para executar a análise foi utilizado o programa WinOMeter[4].

4.2. Resultados

Referente a análise de tempo de conclusão e esforço, ambos sendo métricas de eficiência, os resultados foram divididos em dois grupos de ferramentas (G1 - mais eficientes e G2 - menos eficientes). Os resultados são apresentados na Tabela 2, onde as Ferramentas F3 e F4 demandaram mais tempo na criação de características e a ferramenta F3 na criação de restrições. Algumas ferramentas, F4 e F5 não permitem a criação de tais restrições. A respeito do esforço, a Tabela 3 apresenta os resultados. Para essa métrica, foram dividido três grupos de ferramentas(G1 - menos esforço até G3 - mais esforço). Podemos observar que as ferramentas F2, F5 e F6 obtiveram os menores esforços, a ferramenta WebAtlas encontra-se no grupo G2 por conta que as associações entre *feature* pai e filha são modeladas separadamente, isto demanda um esforço maior ao construir um modelo de características. As ferramentas F3 e F4 são as consideradas com maior esforço, F3 apresenta uma forma de modelar que faz intercalar demasiadamente entre utilizar o mouse e teclado, além de ter que criar associações separadamente, F4 apresenta a utilização de *Drag-and-Drop*, o que torna uma ferramenta com um esforço enorme em caso de diagramas extensos. Com relação ao esforço para modelar as restrições, as ferramentas F1 e F2 tiveram os melhores resultando, F6 ficou no grupo G2 pois a linguagem formal utilizada para as restrições tráz algumas limitações, fazendo que certas restrições sejam divididas ocasionando na construção de mais restrições. F3 é considerada a ferramenta que exige maior esforço, pois suas restrições são elaboradas de forma textual utilizando identificadores relacionando com uma determinada *feature*, assim é necessário procurar qual o identificador da *feature* desejada para construir a restrição.

Os resultados da análise de segurança são apresentados na Tabela 4. Nenhuma ferramenta apresentou suporte a redundância, o que pode ser um ponto negativo por não aumentar a legibilidade ou compreensão do modelo, porem é positivo quanto a manutibilidade. Quanto a anomalias, todas ferramentas deixam modelar, porém a FeatureIDE

[4]http://winometer.findmysoft.com/

é a única que apresenta feedback quanto a esta métrica. Quanto a incosistência FeatureIDE e SPLOT permitem a construção porém lançam alertas quanto a estas restrições. Por último a semântica, onde a CaptainFeature é a única ferramenta que não da suporte a está métrica.

Tabela 2. Análise de tempo de conclusão de tarefa

Grupos	C	R
G1	F1 - F2 - F5 - F6	F1 - F2 - F6
G2	F3 -F4	F3

G1: até 5min, G2: +5min
C. Características, R. Restrições

Tabela 3. Análise de esforço

Grupos	C	R
G1	F2, F5 e F6	F1, F2
G2	F1	F6
G3	F3 e F4	F3

C. Características, R. Restrições

Tabela 4. Segurança dos modelos

Ferramentas	A	B	C	D
WebAtlas	0	0	0	3
FeatureIDE	0	1	1	3
Captain Feature	0	0	0	0
FDE	0	-	-	3
TouchCore	0	-	-	3
SPLOT	0	0	1	3

A. Redundância, B. Anomalias, C. Inconsistência, D. Semântica.

Tabela 5. Requisitos de usabilidade funcional

Ferramentas	A	B	C	D	E	F
WebAtlas	-	X	X	-	-	-
FeatureIDE	X	X	X	-	X	-
CaptainFeature	X	-	-	-	-	X
FDE	-	-	-	-	-	-
TouchCore	-	X	-	-	-	X
SPLOT	X	-	-	-	-	X

A. *Feedback*, B. *Undo*, C. Atalhos, D. *User Expertise*,
E. Reutilizar Informações, F. Ajuda.

Com relação aos resultados de requisitos de analise funcional apresentados na Tabela 5, os dados foram medianos onde esperava-se que as ferramentas *Standalone* (sem a necessidade de uma IDE (Ambiente Integral de Desenvolvimento) para sua utilização) tivessem resultados mais satisfatórios. Feature Diagram Editor é limitada ao *drag-and-drop*, desta forma impossibilita a utilização de atalhos e *user expertise*, porém através da integração com outras ferramentas é possível utilizar tais requisitos. Por sua vez, FeatureIDE destaca-se por ser um plugin da IDE Eclipse, uma ferramenta bem desenvolvida. SPLOT é a única ferramenta web e que utiliza a forma textual para elaborar modelos de características. Contudo, os resultados foram bons, pois conseguimos obter informações em todos requisitos de usabilidade funcional, exceto com reutilização de informações. Tais resultados serão utilizados para acrescentar requisitos de analise funcional na WebAtlas.

5. Considerações Finais

Afim de verificar a qualidade da ferramenta WebAtlas, comparamos com as demais ferramentas utilizadas para este propósito, realizamos uma avaliação de usabilidade, segurança e requisitos de usabilidade funcional. Com base nos resultados obtidos, podemos tirar conclusões que a ferramenta WebAtlas está no caminho correto quanto algumas ferramentas mais utilizadas para este propósito. A utilização de outras ferramentas pode ser utilizada como base para futuras melhorias, tanto na qualidade da ferramenta quanto na elaboração e confiabilidade do modelo. Os próximos passos são finalizar a implementação de uma nova notação, fazer as melhorias necessárias encontradas através

desta comparação e submeter um estudo de caso com alunos utilizando a WebAtlas e outras ferramentas de modelagem, desta forma reduzimos ameaças de experiência entre as ferramentas e podemos efetuar uma avaliação de satisfação quanto a modelagem de um modelo de característica.

Referências

Acher, M., Lopez-Herrejon, R. E., and Rabiser, R. (2017). Teaching software product lines: A snapshot of current practices and challenges. *ACM Trans. Comput. Educ.*, 18(1):2:1–2:31.

Clements, P. and Northrop, L. (2001). *Software Product Lines: Practices and Patterns*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Constantino, K., Pereira, J. A., Padilha, J., Vasconcelos, P., and Figueiredo, E. (2016). An empirical study of two software product line tools. In *Proceedings of the 11th International Conference on Evaluation of Novel Software Approaches to Software Engineering*, ENASE 2016, pages 164–171, Portugal. SCITEPRESS - Science and Technology Publications, Lda.

de M. Rodrigues, E. (2013). *Plets: a product line of model-based testing tools*. PhD thesis, Pontifícia Universidade Católica do Rio Grande do Sul, http://hdl.handle.net/10923/5577.

El Dammagh, M. and De Troyer, O. (2011). Feature modeling tools: Evaluation and lessons learned. In *Proceedings of the 30th International Conference on Advances in Conceptual Modeling: Recent Developments and New Directions*, ER'11, pages 120–129, Berlin, Heidelberg. Springer-Verlag.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Meinicke, J. (2017). *Mastering Software Variability with FeatureIDE*. Springer.

Mendonca, M., Branco, M., and Cowan, D. (2009). S.p.l.o.t.: Software product lines online tools. In *Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications*, OOPSLA '09, pages 761–762, New York, NY, USA. ACM.

Pohl, K., Böckle, G., and van der, F. J. (2005). *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag New York, New York, NY, USA.

Schöttle, M., Thimmegowda, N., Alam, O., Kienzle, J., and Mussbacher, G. (2015). Feature modelling and traceability for concern-driven software development with touchcore. In *Companion Proceedings of the 14th International Conference on Modularity*, MODULARITY Companion 2015, pages 11–14, New York, NY, USA. ACM.