ArgoUML-SPL: Uma Linha de Produtos para Modelagem de Sistemas usando UML

Marcus Vinicius Couto[1], Marco Túlio Valente[2]

[1]Instituto de Informática, PUC Minas
[2]Departamento de Ciência da Computação, UFMG

marcusvnac@gmail.com, mtov@dcc.ufmg.br

Abstract. *In this paper we present a software product line extracted from the ArgoUML tool, an open source tool widely used for designing systems in UML. By extracting this product line, our central goal was to provide to researchers in the area a realistic product line, extracted from a complex, large, and mature system.*

1 Introdução

Linhas de Produtos de Software (LPS) constitui uma abordagem emergente para projeto e implementação de sistemas que procura promover o reúso sistemático de componentes e outros artefatos de software. O objetivo final é migrar para uma cultura de desenvolvimento onde novos sistemas são derivados a partir de um conjunto de componentes e artefatos comuns (os quais constituem o núcleo da linha de produtos). Além de componentes do núcleo, uma linha de produtos inclui componentes responsáveis pela implementação de *features* que somente são necessárias em determinados domínios ou ambientes de uso.

Os princípios básicos de linhas de produtos de software foram propostos há mais de dez anos (ou até há mais tempo, visto que em essência eles apenas procuram sistematizar estratégias de reúso que sempre nortearam o projeto e implementação de sistemas). No entanto, ainda não se tem conhecimento de implementações públicas de linhas de produtos de software envolvendo sistemas de médio para grande porte. Normalmente, os trabalhos de pesquisa nessa área se valem de sistemas de demonstração, que foram integralmente sintetizados em laboratório pelos próprios autores desses trabalhos. Como exemplo, podemos citar as seguintes linhas de produtos: *Expression Product Line* [4], *Graph Product Line* [5] e *Mobile Media Product Line* [3].

Assim, descreve-se neste artigo uma ferramenta de modelagem de sistemas baseada em princípios de linhas de produtos de software. A linha de produtos descrita foi extraída do sistema ArgoUML, uma ferramenta de código aberto largamente utilizada para projeto de sistemas em UML. Na versão original do ArgoUML, a implementação de diversas *features* opcionais encontra-se entrelaçada com a implementação de funcionalidades mandatórias da ferramenta. Mais especificamente, descreve-se no trabalho uma experiência por meio da qual o código responsável por oito *features* importantes – porém não imprescindíveis ao funcionamento do sistema – foi delimitado usando-se diretivas de pré-processamento. Com isso, viabilizou-se a geração de produtos do ArgoUML sem uma ou mais dessas *features* opcionais. A versão considerada do ArgoUML possui cerca de 120 KLOC, sendo que cerca de

37 KLOC foram marcadas como pertencentes a pelo menos uma das oito *features* consideradas no estudo. Tais números são largamente superiores a qualquer outra experiência de extração de *features* documentada na literatura [2].

O restante deste artigo está organizado conforme descrito a seguir. Na Seção 2, apresenta-se uma visão geral da linha de produtos proposta, denominada ArgoUML-SPL, incluindo uma descrição da arquitetura do sistema base (ArgoUML) e das *features* extraídas. Na Seção 3, descreve-se o processo de extração, detalhando a metodologia usada para delimitação das *features* por meio de diretivas de pré-processamento. Por fim, a Seção 4 discute trabalhos relacionados e a Seção 5 conclui o artigo.

2 ArgoUML-SPL

A extração de uma LPS a partir de uma versão monolítica da ferramenta ArgoUML teve como objetivo principal disponibilizar à comunidade de pesquisadores da área uma linha de produtos real, criada a partir de um software relevante, maduro e complexo. A ferramenta ArgoUML é um sistema de código aberto, largamente utilizado para modelagem de sistemas em UML.

2.1 Arquitetura

O ArgoUML possui uma arquitetura com os subsistemas mostrados na Figura 1, onde as setas pontilhadas representam dependências entre os subsistemas. As principais funções desses subsistemas são as seguintes [8]:

- Externos: bibliotecas de terceiros, tais como bibliotecas para edição gráfica e para especificação de restrições em OCL.
- Baixo Nível: subsistemas responsáveis por tarefas como configuração, internacionalização, gerenciamento de tarefas, gerenciamento de modelos e pela implementação de interfaces gráficas.
- Controle e Visão: subsistemas responsáveis pela criação e manipulação dos diversos diagramas, dentre outras tarefas.
- Alto Nível: subsistema responsável pela inicialização da aplicação.

2.2 Features

Para criação do ArgoUML-SPL, foram selecionadas oito *features* representando requisitos funcionais e não funcionais relevantes do sistema, conforme descrito a seguir:

- A *feature* Logging é um representante dos requisitos não funcionais do sistema. Essa *feature* tem por finalidade registrar eventos de exceções e informações para *debug*, sendo que a sua escolha deveu-se à sua natureza espalhada e entrelaçada ao código base do sistema.

- As próximas seis *features* escolhidas representam diagramas UML, incluindo: Diagrama de Estados, Diagrama de Atividades, Diagrama de Colaboração, Diagrama de Sequência, Diagrama de Casos de Uso e Diagrama de Implantação.

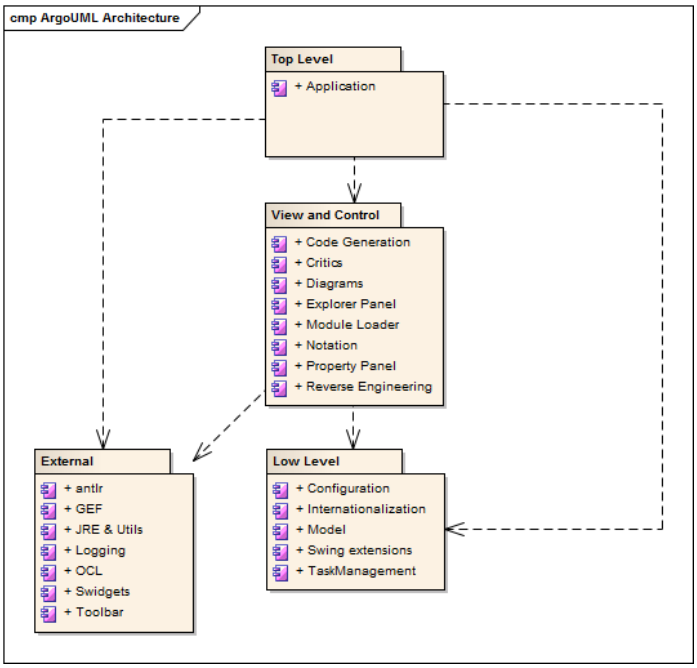

Figura 1. Arquitetura do ArgoUML

- Por fim, a *feature* SUPORTE COGNITIVO tem por finalidade prover informações aos projetistas para auxiliá-los a detectar e resolver problemas em seus projetos. Essa *feature* é implementada por agentes de software que ficam em execução contínua em segundo plano efetuando análises sobre os diagramas criados, com a finalidade de detectar eventuais problemas nos mesmos. Esses agentes podem recomendar boas práticas a serem seguidas na construção do modelo, fornecer lembretes sobre partes do projeto que ainda não foram finalizadas ou sobre a presença de erros de sintaxe, dentre outras tarefas.

O modelo de *features* do ArgoUML-SPL é mostrado na Figura 2 (nesse modelo, círculos cheios representam *features* mandatórias; círculos vazios, representam *features* opcionais).

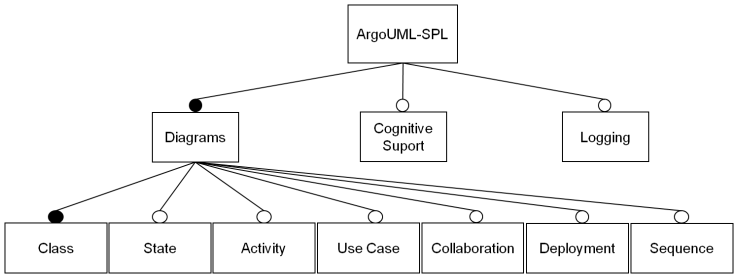

Figura 2. Modelo de Features

3 Processo de Extração

A tecnologia de pré-processadores foi empregada na extração da linha de produtos de software que originou o projeto ArgoUML-SPL. Claramente, diretivas de pré-processamento são conhecidas por sua capacidade de poluir o código com anotações

extras, tornando-o menos legível e mais difícil de entender, manter e evoluir [1]. Por outro lado, pré-processadores também possuem algumas vantagens, incluindo a facilidade de utilização e entendimento. Mais importante, pré-processadores permitem a anotação de qualquer trecho de código. Ou seja, eles constituem a tecnologia mais primitiva e ao mesmo tempo mais poderosa para marcação de *features*. Assim, ao utilizar pré-processadores no trabalho, pretendeu-se disponibilizar uma LPS que servisse como *benchmark* para outras tecnologias de modularização, como orientação por aspectos. Em outras palavras, pretende-se com o ArgoUML-SPL disponibilizar um sistema-desafio, que contribua para avaliar o real poder de expressão de técnicas modernas de modularização e separação de interesses.

Como Java não possui suporte nativo para diretivas de pré-processamento, foi utilizada uma ferramenta denominada `javapp`[1]. Essa ferramenta disponibiliza diretivas similares àquelas existentes em C/C++, incluindo `#ifdef`, `#ifndef` e `#else`. Basicamente, tais diretivas informam ao pré-processador se o código fonte delimitado pela diretiva deve ou não ser compilado. Dessa maneira, é possível selecionar os trechos de código que estarão presentes na versão final de um determinado produto. A Figura 3 apresenta um exemplo de código anotado. Nessa figura, o código da linha 3 foi anotado como pertencente à *feature* Suporte Cognitivo, ou seja, essa linha será incluída apenas em produtos que incluam tal *feature*. Por outro lado, o código da linha 5 somente será incluído em produtos que não incluam a *feature* Suporte Cognitivo.

```
1 JPanel todoPanel;
2 //#if COGNITIVE
3 todoPanel = new ToDoPane(splash);
4 //#else
5 todoPanel = new JPanel();
6 //#endif
```

Figura 3. Exemplo de código anotado

Anotação do Código: A identificação dos elementos de código do ArgoUML pertencentes a uma determinada *feature* foi feita manualmente por um dos pesquisadores envolvidos na criação da LPS. Como esse pesquisador não era especialista na ferramenta, o primeiro passo para essa extração consistiu em um estudo detalhado da arquitetura do sistema, principalmente com o auxílio de seu *cookbook* [8]. Após um entendimento básico da arquitetura do ArgoUML, o seu código fonte foi estudado a fim de identificar os pontos em que cada *feature* era demandada. Nesta tarefa de exploração do código fonte, utilizou-se o ambiente de desenvolvimento Eclipse. Os recursos de busca textual e busca por referências dessa IDE foram extensivamente usados para auxiliar na localização dos trechos de código relativos a cada uma das *features*. Neste processo, uma vez identificado que um determinado trecho de código `X` apenas deveria ser compilado caso uma determinada feature `F` fosse selecionada, procedia-se à sua delimitação por meio de anotações `#ifdef` e `#endif`.

Inicialmente, a validação dos diversos produtos gerados pela LPS extraída foi feita por meio de testes funcionais, do tipo caixa preta. Esses testes visaram

[1]Disponível em `http://www.slashdev.ca/javapp`.

principalmente a validação do produto gerado de acordo com a seleção de *features* efetuada. Em tais testes foi avaliada a estabilidade do sistema e comparado o funcionamento do ArgoUML original com o produto gerado pela LPS. Por exemplo, suponha um produto com todas as *features*, exceto Diagrama de Estados. Durante os testes, esse produto foi gerado e então executado para avaliar o seu correto funcionamento (isto é, se no produto gerado a opção de criação de diagrama de estados tinha sido de fato removida). Esse procedimento foi repetido para todas as *features* opcionais da LPS extraída. A segunda parte da validação consistiu em uma inspeção manual de todo código fonte do sistema, a fim de verificar se os elementos sintáticos responsáveis por cada *feature* haviam sido de fato anotados.

Ao final desse processo, a LPS extraída foi enviada para os projetistas do sistema ArgoUML, que após avaliação aprovaram a sua hospedagem no website oficial da comunidade. O principal critério para aprovação do ArgoUML-SPL como um dos sub-projetos do ArgoUML foi a sua potencial contribuição para evolução e modernização da ferramenta. Espera-se que essa hospedagem pública no repositório de versões do ArgoUML possa atrair desenvolvedores interessados em ampliar o número de *features* atualmente disponibilizadas pela LPS.

Derivação de Produtos: Para automatizar a geração dos possíveis produtos da LPS foram implementados um conjunto de scripts `ant`. Basicamente, esses scripts permitem que os desenvolvedores informem quais *features* devem ser incluídas em um determinado produto. Feito isso, eles se encarregam da execução do pré-processador `javapp`, bem como da compilação e empacotamento do produto selecionado pelo desenvolvedor.

Tamanho da Linha de Produtos Extraída: As Tabelas 1 e 2 apresentam informações sobre o tamanho de alguns produtos e *features*, respectivamente. A Tabela 2 apresenta também o percentual de código dedicado à implementação de cada *feature* (em relação à versão original – não baseada em LPS – do ArgoUML). Nessas tabelas, são usadas as seguintes métricas de tamanho: número de linhas de código (LOC), número de pacotes (NOP), número de classes (NOC) e número de linhas de código anotadas para cada *feature* (LOF).

4 Trabalhos Relacionados

A extração do ArgoUML-SPL foi motivada pelo fato de a maioria dos trabalhos na área de linhas de produtos se basear em sistemas de demonstração, construídos em laboratório. Por exemplo, três linhas de produtos são frequentemente usadas em tais trabalhos: *Expression Product Line* (EPL) [4], *Graph Product Line* (GPL) [5] e *Mobile Media Product Line* (MMPL) [3]. A EPL consiste em uma gramática de expressões cujas variabilidades incluem tipos de dados (literais), operadores (negação, adição etc) e operações (impressão e avaliação). A EPL completa, implementada em Java, possui apenas 2 KLOC. A GPL é uma linha de produtos na área de grafos, cujas variabilidades incluem características das arestas (direcionadas ou não direcionadas, com peso ou sem peso etc), métodos de busca (em profundidade ou em largura) e alguns algoritmos clássicos de grafos (verificação de *loops*, caminho mais

Tabela 1. Métricas de Tamanho para Produtos

Produto	LOC	NOP	NOC
Original, não baseada em LPS	120.348	81	1.666
Apenas Suporte Cognitivo desabilitada	104.029	73	1.451
Apenas Diagrama de Atividades desabilitada	118.066	79	1.648
Apenas Diagrama de Estados desabilitada	116.431	81	1.631
Apenas Diagrama de Colaboração desabilitada	118.769	79	1.647
Apenas Diagrama de Sequência desabilitada	114.969	77	1.608
Apenas Diagrama de Casos de Uso desabilitada	117.636	78	1.625
Apenas Diagrama de Implantação desabilitada	117.201	79	1.633
Apenas Logging desabilitada	118.189	81	1.666
Todas as *features* desabilitadas	82.924	55	1.243

Tabela 2. Métricas de Tamanho para Features

Feature	LOF	
Suporte Cognitivo	16.319	13,56%
Diagrama de Atividades	2.282	1,90%
Diagrama de Estados	3.917	3,25%
Diagrama de Colaboração	1.579	1,31%
Diagrama de Sequência	5.379	4,47%
Diagrama de Casos de Uso	2.712	2,25%
Diagrama de Implantação	3.147	2,61%
Logging	2.159	1,79%
Todas as *features*	37.424	31,10%

curto, árvore geradora mínima etc). A GPL completa possui apenas 2 KLOC. Por fim, a MMPL é uma linha de produtos para dispositivos móveis, cujo objetivo é implementar aplicações para gerência de documentos multimídia, incluindo fotos, vídeo e músicas. Além do tipo de mídia, a MMPL inclui outras variabilidades, como transferência de documentos via SMS e algumas operações sobre os documentos gerenciados (criação, remoção, visualização etc). A MMPL completa possui apenas 3 KLOC.

Anualmente, durante a *Software Product Line Conference* (SPLC) são escolhidos exemplos de sucesso de implantação prática dos princípios de linhas de produto, que passam a integrar o Hall da Fama dessa abordagem de desenvolvimento [7]. No entanto, esses casos não necessariamente implicam na separação física ou na anotação de código responsável pela implementação de variabilidades.

Quando comparada com extrações de linhas de produtos usando aspectos, acreditamos que uma solução baseada em anotações de código (como diretivas de pré-processamento) apresenta uma importante vantagem prática: ela dispensa a transformação prévia do código orientado a objetos a fim de viabilizar a sua instrumentação por meio de uma linguagem para definição de conjuntos de junção [6].

5 Conclusões

Neste trabalho, foi apresentada uma linha de produtos extraída a partir de uma ferramenta de modelagem UML relevante, madura e complexa. Uma análise quantitativa detalhada da linha de produtos extraída pode ser encontrada em [2]. Nessa análise, são providas as seguintes informações sobre as *features* extraídas: tamanho, comportamento transversal, granularidade e localização estática. São discutidos ainda os principais problemas encontrados na anotação do código do ArgoUML. Por fim, o sistema ArgoUML-SPL já foi usado para avaliar a ferramenta CIDE+, uma ferramenta para extração semi-automática de linhas de produtos [9].

O ArgoUML-SPL está disponível em: `http://argouml-spl.tigris.org`.

Agradecimentos: Este trabalho foi apoiado pela FAPEMIG e CNPq.

Referências

[1] Bram Adams, Wolfgang De Meuter, Herman Tromp, and Ahmed E. Hassan. Can we refactor conditional compilation into aspects? In *8th ACM International Conference on Aspect-Oriented Software Development (AOSD)*, pages 243–254, 2009.

[2] Marcus Vinicius Couto, Marco Tulio Valente, and Eduardo Figueiredo. Extracting software product lines: A case study using conditional compilation. In *15th European Conference on Software Maintenance and Reengineering (CSMR)*, pages 191–200, 2011.

[3] Eduardo Figueiredo et al. Evolving software product lines with aspects: an empirical study on design stability. In *30th International Conference on Software Engineering (ICSE)*, pages 261–270, 2008.

[4] Roberto Lopez-Herrejon, Don Batory, and William R. Cook. Evaluating support for features in advanced modularization technologies. In *19th European Conference on Object-Oriented Programming (ECOOP)*, pages 169–194, 2005.

[5] Roberto E. Lopez-Herrejon and Don S. Batory. A standard problem for evaluating product-line methodologies. In *Third International Conference on Generative and Component-Based Software Engineering (GPCE)*, pages 10–24, 2001.

[6] Marcelo Nassau, Samuel Oliveira, and Marco Tulio Valente. Guidelines for enabling the extraction of aspects from existing object-oriented code. *Journal of Object Technology*, 8(3):1–19, 2009.

[7] Software Product Line Conference. Product line hall of fame, 2010. http://splc.net.

[8] Linus Tolke, Markus Klink, and Michiel van der Wulp. ArgoUML cookbook, 2010. http://argouml.tigris.org/wiki/Cookbook.

[9] Marco Tulio Valente, Virgilio Borges, and Leonardo Passos. A semi-automatic approach for extracting software product lines. *IEEE Transactions on Software Engineering*, pages 1–39, 2011 (to appear).