Elicitando Semelhanças e Variabilidades de Linhas de Produtos de Software com Diagramas de Interação do Usuário

Jefferson Kobs, Patrícia Vilain

Departamento de Informática e Estatística – INE
Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC – Brasil

jeffersonkobs@gmail.com, patricia.vilain@ufsc.br

Abstract. *This paper presents a new process of identifying common and variable characteristics of a SPL (Software Product Line). This process uses the technique UID (User Interaction Diagram) along with the feature model during domain engineering. It is shown how the characteristics related to functional requirements can be identified from UIDs, making the definition of the feature model easier. Subsequently, both feature model and UIDs are used during the project and implementation of the core assets of domain engineering. The proposed method was successfully applied in a SPL composed by three applications: Document Management System (DMS), File Manager (FM), and Datebook System (DS).*

Resumo. *Este artigo apresenta um novo processo de levantamento das características comuns e variáveis de uma LPS (Linha de Produto de Software). Este processo utiliza a técnica UID (User Interaction Diagrams) em conjunto com o modelo de features durante a engenharia de domínio. É mostrado como as características relacionadas com os requisitos funcionais podem ser identificadas a partir de UIDs, tornando a definição do modelo de features mais fácil. Posteriormente, tanto o modelo de features como os UIDs são utilizados durante o projeto e implementação dos core assets da engenharia de domínio. O processo proposto foi aplicado com sucesso em uma LPS composta por três aplicações: gerenciador eletrônico de documentos (GED), gerenciador de arquivos(GAR) e agenda de compromissos (AGD).*

1. Introdução

Atualmente, com a grande quantidade de empresas desenvolvendo sistemas específicos para as diversas áreas de negócio, aquelas que conseguem entregar um sistema adequado para a realidade de cada empresa no menor tempo possível e com um custo reduzido, acabam levando vantagem com relação às demais.

Assim, a abordagem de Linhas de Produtos de Software (LPS) ganhou mais atenção nos últimos anos devido a esta competição no segmento de desenvolvimento de software [Pohl, Böckle e Linden 2005]. LPS tem foco na reutilização, onde o desenvolvimento de sistemas é modularizado, permitindo que estes vários componentes possam ser reutilizados em vários sistemas, reduzindo assim o tempo que seria necessário para desenvolvê-los novamente [Long 2002].

O desenvolvimento de uma LPS é dividido em duas atividades: Engenharia de Domínio e Engenharia de Aplicação. Na Engenharia de Domínio, uma etapa importante é a definição do modelo de *features*[1] [ALMEIDA 2007]. Para auxiliar a definição do modelo de *features*, uma abordagem que pode ser utilizada é a construção de Diagramas de Interação do Usuário (UID – User Interaction Diagram).

Os UIDs são diagramas que demonstram como a interação entre um usuário e uma aplicação é realizada e como ocorre a troca de informações entre eles, sem levar em consideração aspectos específicos de interface com usuário [Vilain 2002].

No processo proposto neste trabalho, os UIDs são utilizados para auxiliar a definição do modelo de *features*. Inicialmente, o levantamento de todas as funcionalidades e características do sistema é realizado através da utilização de UIDs e, posteriormente, estas funcionalidades e características são vinculadas às *features* do modelo de *feature*.

Este artigo está organizado da seguinte forma. A seção 2 descreve os UIDs (Diagramas de Interação do Usuário). Os conceitos de LPS são apresentados na seção 3. A seção 4 descreve como podem ser identificadas as semelhanças e variabilidades com UIDs. A seção 5 apresenta um exemplo de aplicação da proposta apresentada na seção 4. Finalmente, a seção 6 conclui este trabalho.

2. Diagramas de Interação do Usuário

Um Diagrama de Interação do Usuário ou UID (User Interation Diagram) representa a interação entre o usuário e uma aplicação que apresenta intensa troca de informações [Vilain 2002]. Um UID é composto por um conjunto de estados conectados através de transições. Os estados representam as informações que são trocadas entre o usuário e a aplicação, enquanto as transições são responsáveis pela troca do foco da interação de um estado para outro. Diz-se que um estado da interação é o foco da interação quando as informações contidas nesse estado representam as informações que estão sendo trocadas entre o usuário e a aplicação em um dado momento. As informações participantes da interação entre o usuário e a aplicação são apresentadas dentro dos estados de interação, mas algumas seleções e opções são associadas às transições. As transições são disparadas, geralmente, pela entrada ou seleção de informações pelo usuário [Vilain 2002].

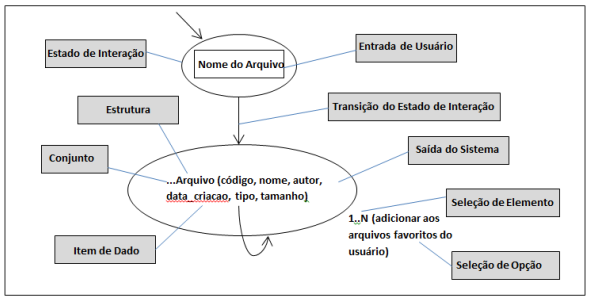

Figura 1. Exemplo de UID

[1] O termo *feature* é usado como sinônimo de característica.

O UID mostrado na Figura 1 representa a interação entre um usuário e um sistema durante a tarefa *Buscar um Arquivo pelo Nome*, onde o usuário fornece o nome de um determinado arquivo e o sistema retorna um conjunto com todos os arquivos que correspondem ao nome digitado. Os usuários podem incluir nos seus favoritos, os arquivos que desejam visualizar posteriormente. Para maior clareza, o nome de cada símbolo utilizado na Figura 1 está incluído nos retângulos cinzas.

3. Linhas de Produtos de Software (LPS)

As Linhas de Produtos de Software (LPS) são um paradigma de desenvolvimento que cria um portfólio de sistemas de software similares a partir de uma base de componentes compartilhados [KRUEGER 2011]. A ideia é usar artefatos genéricos que se aplicam a diferentes produtos que estão incluídos no mesmo escopo. Assim, é possível conseguir certo número de melhorias durante o seu desenvolvimento em relação ao tempo, custo e qualidade do que está sendo feito. Com este paradigma é possível ainda o ingresso no mercado de uma forma rápida e flexível com a característica de customização em massa [SEI 2013].

Uma LPS é dividida basicamente em duas atividades: Engenharia de Domínio e Engenharia de Aplicação, descritas a seguir.

3.1 Engenharia de domínio

Esta é a primeira atividade da elaboração da LPS. Ela consiste em coletar, organizar e armazenar a experiência adquirida na construção de sistemas em um domínio particular sob a forma de artefatos reutilizáveis, bem como proporcionar um meio adequado para a reutilização desses artefatos na construção de novos sistemas [ALMEIDA 2007].

Para a realização dessa atividade, três etapas podem ser seguidas [ALMEIDA 2007]:

A - Análise de Domínio: Definição de requisitos reutilizáveis para sistemas pertencentes a um determinado domínio.

B - Projeto de Domínio: Desenvolvimento da arquitetura comum para estes sistemas e um plano de produto.

C - Implementação de Domínio: Implementação do núcleo que será reutilizado em todas as aplicações.

3.2 Engenharia de aplicação

Neste trabalho são utilizadas as duas fases propostas pelo Feature-Oriented Reuse Method (FORM) [KANG, KIM, LEE e SHIN 1998]:
A - Análise de requisitos e Seleção de recursos. A engenharia de aplicação começa com a seleção das características definidas na engenharia de domínio, que serão integradas ao produto.
B - Seleção da arquitetura e o desenvolvimento de aplicativos. Um modelo de arquitetura de referência é gerado automaticamente a partir da seleção dos recursos feito anteri-

ormente. Em seguida, os módulos específicos de cada produto devem ser implementados para complementar a arquitetura reutilizável.

4. Elaborando a Análise de Domínio

A proposta deste trabalho consiste em apresentar uma forma de integrar a etapa de Análise de Domínio de uma LPS com os UIDs, para facilitar a análise, o projeto e implementação do domínio.

A seguir, cada etapa do processo proposto neste trabalho para definir o modelo de domínio a partir dos UIDs é detalhado.

4.1 Definição dos requisitos funcionais a partir de Casos de Uso e UIDs

Na análise de domínio são identificadas todas as características das aplicações que poderão estar na LPS, e que depois serão incluídas no modelo de *features*.

Para facilitar a identificação das características das aplicações, este trabalho propõe a utilização de UIDs para definir os requisitos funcionais das aplicações da LPS. É importante frisar que este trabalho abordará somente características das aplicações que podem ser definidas como requisitos funcionais.

Porém, conforme descrito em [Vilain 2002], cada UID corresponde a um caso de uso. Como através do caso de uso é possível fazer um detalhamento amplo da funcionalidade, primeiro define-se todos os casos de uso de cada um dos sistemas da LPS, para a partir deles definir quais podem ser agregados como um único UID. Para tanto, cada aplicação que vai compor a LPS deverá, então, ter uma descrição do que ela faz ou pretende fazer. A partir dessa descrição, é possível definir os casos de uso.

4.2 Definição das variabilidades e semelhanças nos UIDs a partir dos Casos de Uso

O principal objetivo de definir os requisitos do sistema através de UIDs é a facilidade em mostrar quais requisitos são comuns entre as aplicações e quais sofrem variabilidade. Para isto, para cada caso de uso definido nos diagramas dos sistemas, define-se um UID. Porém, ao encontrar semelhanças nos casos de uso, ao invés de criar dois UIDs, é possível criar apenas um UID, mapeando o que é variável e o que é comum entre os dois requisitos.

Na Figura 2 é apresentado, com um exemplo hipotético, a representação da relação entre as aplicações da LPS e os UIDs. Nesta figura são apresentados os UIDs que foram definidos, entretanto, aqueles que podem ser utilizados em mais de um sistema são representados através de uma linha dupla no diagrama.

Para cada UID que representa uma funcionalidade comum a mais de uma aplicação (representado na Figura 2 com a linha dupla), é necessário identificar as informações comuns e variáveis. Como a notação original do UID [Vilain 2002] não representa variabilidade, foi verificada a necessidade de estender a sua notação para dar suporte à modelagem das *features* variáveis e comuns de diferentes aplicações de uma LPS. A extensão

da notação do UID proposta neste trabalho para representar o que pode ser comum e variável na descrição do UID é baseada na notação de personalização apresentada em [Remáculo 2004].

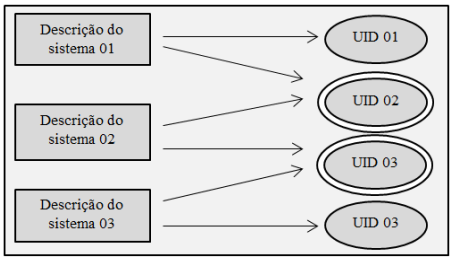

Figura 2. – Representação de UIDs comuns que sofrem variabilidade

A seguir são apresentados os casos onde a variabilidade é representada na extensão proposta para os UIDs:

- **Apresentação de itens de dados variáveis:** Nos UIDs, um item de dado representa uma informação simples que aparece durante a interação. Um item de dado variável representa uma informação simples que não aparece em todas as aplicações. Para representar que um item é variável em um UID, utilizamos o símbolo asterisco (*) como mostrado na Figura 3. Para facilitar a identificação da aplicação onde o item aparece, todas as aplicações candidatas da LPS, deverão receber uma sigla de três letras que irá ser o seu identificador nos UIDs. Desta forma, em todo item de dado que poderá sofrer variabilidade, deverá ser incluída a sigla da aplicação na qual a mesma irá aparecer. Para os itens de dados que aparecem normalmente em todas as aplicações, segue-se a notação normal.

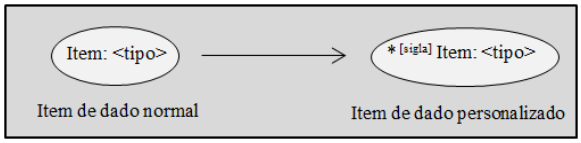

Figura 3. – Representação de item de dado personalizado

- **Transição com opção restrita:** Uma opção restrita é uma ação que não pode ser realizada pelos usuários de todas as aplicações da LPS. O nome da opção restrita é colocado entre parênteses, antecedido pelo símbolo asterisco (*) conforme mostrado na Figura 4. Da mesma forma que no caso anterior, além do asterisco também é utilizada a sigla do sistema. Assim, em toda transição que for restrita a determinadas aplicações, deverá ser incluída a sigla das aplicações em qual a mesma irá aparecer. Para as transições que aparecem normalmente em todas as aplicações, segue-se a notação normal.

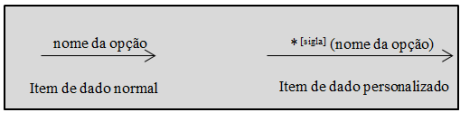

Figura 4. – Representação de transição com opção restrita

4.3 Integrando os UIDs com o Modelo de *Features*

Com a extensão da notação do UID proposta neste trabalho, podem ser definidos quais requisitos são comuns para mais de uma aplicação e quais requisitos podem sofrer variabilidade. Porém, de nada adianta definir quais requisitos são variáveis, se eles não forem relacionados com as *features* do modelo de *features*.

Assim, é proposto neste trabalho o modelo apresentado na Figura 5 para representar o relacionamento entre as *features* e os UIDs. Cada *feature* do modelo de *features* representa uma característica do sistema. Para cada uma dessas *features* podemos ter mais de um UID, pois pode haver mais de uma tarefa associada a cada um dos requisitos, assim como pode haver mais de uma *feature* vinculada ao mesmo UID. Desta forma, é feito um agrupamento dos UIDs que são vinculados com aquela *feature*.

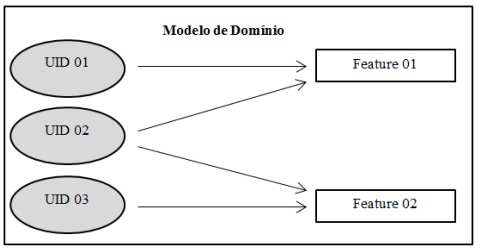

Figura 5. – Relacionamento entre UIDs e *Features*

Desta forma, tem-se a definição dos requisitos através de UIDs e a representação do que foi encontrado de comum e variável entre as *features* de toda a LPS.

Para a definição das *features* e a representação das características comuns e variáveis, são definidas algumas diretrizes de representação, baseadas na representação do modelo FODA:

- *Quando todas as informações relacionadas com uma feature, sejam elas mostradas em parte de um UID, em um UID ou em mais de um UID, são mostradas em todas as aplicações, a feature correspondente será obrigatória.*
- *Quando as informações de uma feature aparecem em somente alguns sistemas, elas são mapeadas para features opcionais.*
- *Quando as informações em uma mesma feature aparecem divididas, algumas em uma aplicação e outras em outras aplicações, elas são mapeadas para features alternativas.*

5. Exemplo de Utilização da Proposta

Nesta seção será descrita parte do desenvolvimento de uma LPS seguindo os passos propostos nesse trabalho. Esta LPS possui três aplicações:

- Gerenciador eletrônicos de documentos (GED) – é uma ferramenta *online* que permite o acesso a documentos através de um navegador web.
- Gerenciador de arquivos (GAR) – é um serviço para organizar o armazenamento de arquivos numa estrutura de diretórios.

- Agenda de compromissos (AGD) – é uma ferramenta *online* acessada por navegador que oferece uma agenda para cadastrar compromissos de um usuário individualmente ou de um grupo de usuários pertencentes a uma organização.

5.1 Definição dos requisites funcionais a partir dos casos de uso e UIDs

Baseado nas definições das três aplicações da LPS, os requisitos funcionais são descritos usando UIDs. Porém, inicialmente é necessário definir os casos de uso dos sistemas, que serão utilizados para exemplificar a proposta. Os diagramas de caso de uso estão exemplificados na Figura 6.

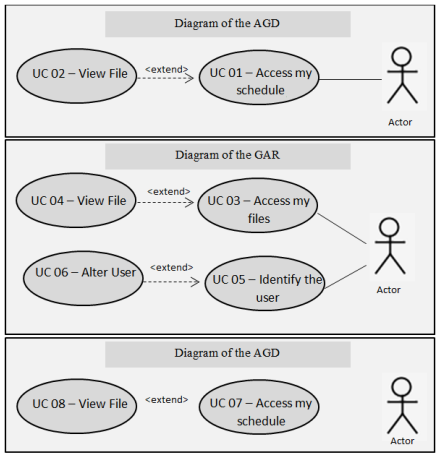

Figura 6. – Casos de Uso das Aplicações

5.2 Definição das variabilidades e semelhanças nos UIDs a partir dos casos de uso

Baseados nas definições das três aplicações da LPS, os requisitos funcionais das aplicações são descritos através de UIDs. Para exemplificar são utilizados dois requisitos funcionais: visualizar arquivos e alterar usuários.

UID 01 – Visualizar Arquivos

Descrição: Deve ser possível visualizar os arquivos através da aplicação GAR e da aplicação AGD. Tanto no GAR, quanto no AGD, estes arquivos são visualizados através de uma lista de arquivos. Ao visualizar o arquivo pelo GAR, é permitido ver o texto completo, o login e a data de alteração. Porém, visualizando pelo AGD, não é possível ver estes dados, porém é possível ver a data do agendamento vinculado a aquele arquivo. Na Figura 7 é apresentado o UID *Visualizar Arquivos* que é acessado a partir do UID *Acessar meus arquivos*.

UID 02 – Alterar Usuários

Descrição: Esta funcionalidade somente pode ser acessada através da aplicação GED e da aplicação GAR, sendo que no AGD esta opção não é disponibilizada, pois os usuários são controlados por um perfil de administrador. Para os usuários que possuem a opção, é permitido alterar *nome*, *login*, *senha* e *e-mail*. Quando for acessado pelo sistema GED, é permitido alterar o *nome de autor*, que aparecerá nos documentos que forem

criados por este usuário. Na Figura 8 é apresentado o UID *Alterar Usuários*, acessado quando o sistema é o GED ou o GAR.

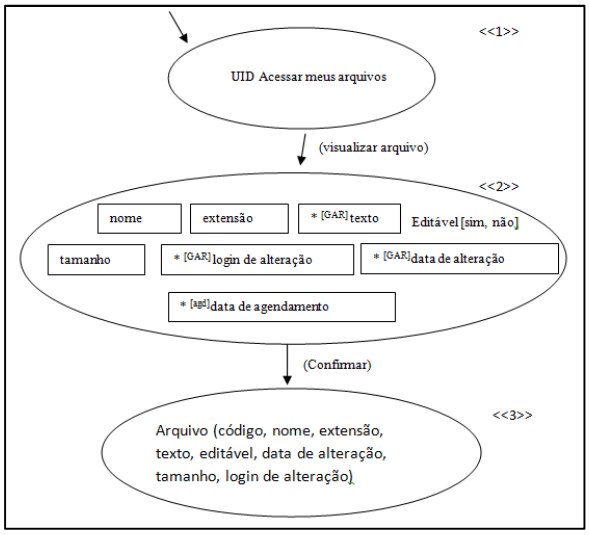

Figura 7. UID Visualizar Arquivo

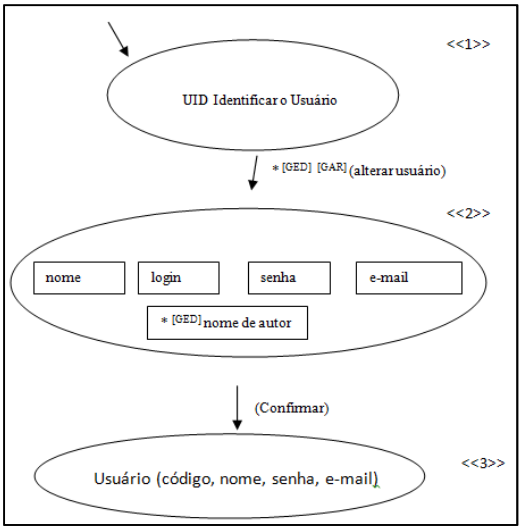

Figura 8. UID Alterar Usuário

5.3 Integração dos UIDs com o Modelo de *Features*

Depois de definir os requisitos comuns e com variabilidade da LPS, podemos especificar o modelo de *features* a partir dos UIDs. Na Figura 9 é apresentado o modelo parcial de *features* da LPS, seguindo o modelo FODA [KANG, COHEM, HESS, NOVAK e PETERSON 1990].

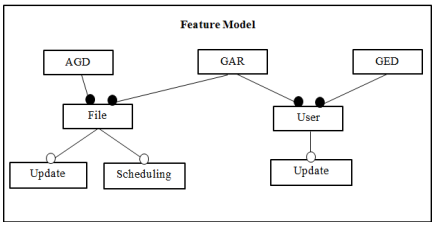

Figura 9. Exemplo do modelo de features

Depois deste passo é apresentado um exemplo na Figura 10, onde é feito o vínculo entre os UIDs de cada *feature*.

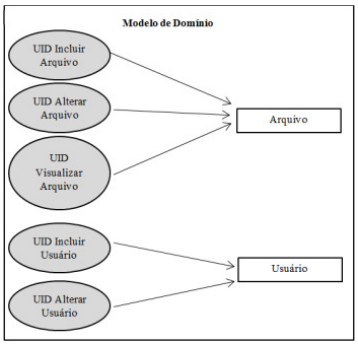

Figura 10. Exemplo de relação entre UIDs e *features*

5.4 Utilização da proposta na Engenharia de Aplicação da LPS

Conforme descrito, o desenvolvimento das aplicações é feito de acordo com as *features* selecionadas para cada uma delas. A partir de cada *feature*, o desenvolvedor da aplicação deve verificar quais UIDs estão vinculados a ela, e assim saber o que deve ser desenvolvido em comum para cada aplicação e o que deve ser variável.

Na Figura 11, é possível verificar que o código Java, através da ferramenta Eclipse Kepler, faz a verificação de qual sistema está acessando a listagem de arquivos, para saber quais campos podem ser visualizados. Na implementação do método *retornaListaDeArquivo*, é repassado a sigla do sistema, permitindo que sejam retornadas somente determinadas informações conforme a variabilidade do UID Visualizar Arquivo.

```
public class VisualizadorDeArquivos {
    public Arquivo retornaArquivo(String usuario, String nomeDoArquivo){
        DownloadUpload download = new DownloadUpload();
        InputStream is = download.retornaObjeto(nomeDoArquivo, usuario);
        if (is == null){
            return null;
        }
        return new Arquivo(usuario, is);
    }

    public ArrayList<Arquivo> retornaListaDeArquivo(String usuario, String sistema){
        DownloadUpload download = new DownloadUpload();
        ArrayList<Objeto> lista = download.retornaListaDeObjeto(usuario);
        ArrayList<Arquivo> listaDeArquivos = new ArrayList<Arquivo>();
        for(Objeto is : lista){
            listaDeArquivos.add(new Arquivo(is.getNome(), is.getExtensao(), is.getTexto(),
                    is.getTamanho(), is.eetLogin(), is.getDataAlt(), is.getDataAed(), is.setSistema()));
        }
        return listaDeArquivos;
    }
}
```

Figura 11. Implementação do UID Visualizar Arquivo

Desta forma, a proposta apresentada neste trabalho se torna útil pois auxilia o analista da LPS na definição de quais *features* possuem variabilidade e também deixa claro como desenvolver a LPS considerando a variabilidade proposta pelo analista.

6. Conclusão

Este trabalho apresentou uma forma de integrar os UIDs com uma das fases principais do desenvolvimento de uma LPS, a Análise de Domínio. Após a definição dos UIDs de todas as aplicações, estes são mapeados para *features* e é definido o modelo de *features*.

A proposta apresentada foi mostrada através de um exemplo. Para cada uma das aplicações candidatas da LPS, foram utilizados casos de uso para fazer a definição dos seus requisitos funcionais. Para cada caso de uso, foi feito o mapeamento de UIDs, analisando quais casos de uso poderiam ser agrupados em um único UID por tratarem da mesma tarefa. Desta forma, somente os UIDs com variabilidade foram utilizados para exemplificação da proposta. Como uma LPS apresenta variabilidade entre os componentes que podem ser utilizados em cada sistema candidato, a notação dos UIDs foi estendida para representar esta variabilidade.

A utilização do mapeamento dos UIDs para as *features*, permite que o desenvolvedor entenda melhor como funciona a variabilidade na implementação dos sistemas candidatos, pois os UIDs apresentam informações mais detalhadas. É neste ponto que está a importância de representar o relacionamento entre os UIDs e as *features*, pois quando o desenvolvedor fizer a implementação de cada *feature*, ele terá conhecimento de todos os UIDs relacionados com ela e poderá saber mais claramente o que deve ser variável para cada sistema que usará esta *feature*.

Também foi observado que utilizar os UIDs para auxiliar na definição do modelo de features torna o mapeamento da variabilidade entre as *features* mais ágil, por demonstrar num contexto geral todas as funcionalidades do sistemas da LPS.

A proposta apresentada neste trabalho pode ser usada em conjunto com abordagens que descrevem funcionalidades a partir de casos de uso [Chastek, Donohoe, Kang e Thiel 2001] ou cenários [Rommes 2003] [Ullah e Ruhe 2006].

Referências

KANG, Kyo C. , COHEM, Sholom G. , HESS, James A. , NOVAK, William E. e PETERSON, A. Spencer, Feature-Oriented Domain Analysis (FODA), 1990.

K. Pohl, G. Böckle, and F. J. v. d. Linden, Software Product Line Engineering: Foundations, Principles, and Techniques. Secaucus, NJ, USA: Springer-Verlag, 2005.

C.A. Long. Software product lines: practices and patterns [book review]. Software, IEEE, 19(4):131 –132, jul/aug 2002.

P. Vilain, User Interaction Modeling in Hypermedia Applications, PhD Thesis, PUC-Rio, 2002. (in Portuguese)

P. Vilain, D. Schwabe, and C.S. de Souza, "A Diagrammatic Tool for Representing User Interaction in UML", UML 2000 Conference, 2000, pp. 133-147.

KRUEGER, Charles W. Introduction to Software Product Lines for Systems and Software, 2011.

Software Product Lines - Overview. Disponível em <http://www.sei.cmu.edu/productlines/> acessado em novembro de 2013.

ALMEIDA, Eduardo Santana. RiDE: The RiSE Process for Domain Engineering, 2007.

KANG Kyo, C. , KIM, Sajoong, LEE, Jaejoon,KIM, Kijoo, KIM, Gerard Jounghyun e SHIN, Euiseob. FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures, 1998.

Remáculo, Luanda Philippi, Personalização de Diagramas de Interação do Usuário e Mapeamento para a Ontologia de Widgets Abstratos, 2004.

Chastek, G.; Donohoe, P.; Kang, K.; Thiel, S. Product Line Analysis: A Practical Introduction, SEI, 2001.

Rommes, E., A People Oriented Approach to Product Line Scoping. "International Workshop on Product Line Engineering ? The Early Steps", 2003.

Ullah, M.I., Ruhe, G. Towards, Comprehensive Release Planning for Software Product Lines. International Workshop on Software Product Management, 2006.