Doc. 3

Conceitos Essenciais sobre Linha de Produto de Software[1]

I. Linha de Produto de Software

Uma linha de produto de software (LPS) corresponde a um conjunto de sistemas de *software* que compartilham características (*features*) comuns e gerenciáveis que satisfazem a necessidade de um segmento particular ou de uma missão. Este conjunto de sistemas é denominado também, família de produtos. Os membros da família são produtos específicos desenvolvidos de maneira sistemática a partir da instanciação de uma infraestrutura comum de uma LPS, chamada núcleo de artefatos.

O núcleo de artefatos é formado por um conjunto de características comuns (similaridades) e características variáveis (variabilidades). As variabilidades podem estar associadas a diferentes níveis de abstração, como a descrição da arquitetura, o código fonte, etc., e auxiliam na geração de produtos específicos distintos em um mesmo domínio e, desta forma, diminuem o custo e o tempo de desenvolvimento, reduzem riscos e perdas, além de reduzirem o *time to market* e justificarem o retorno de investimento (ROI).

Uma das formas de se implementar características de LPS é tratando cada uma delas como um interesse presente na família. Desta forma, pode-se focar na melhor maneira para desenvolver e modularizar o referido interesse/característica. Vale ressaltar que além da presença de características, outros interesses podem estar presentes. É comum a presença de interesses não funcionais, tais como persistência, *logging*, tratamento de exceções, dentre outros. Os interesses podem ser mapeados em artefatos da LPS usando estereótipos UML, como pode ser visto na Figura 1 em que a interface ISaveScore está associada ao interesse save que é uma característica da LPS e algumas de suas operações também estão associadas a esta característica. Além disso, a operação saveInRanking está associada ao interesse ranking que é uma característica e ao interesse não funcional logging.

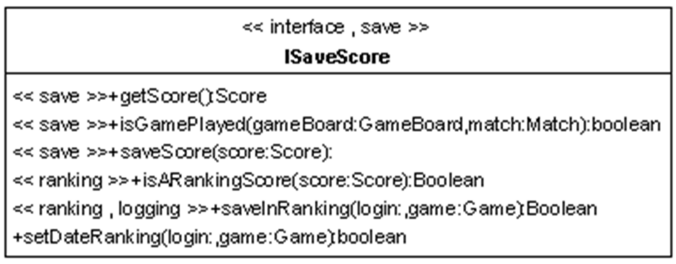

Figura 1 – Exemplo de Interface com Interesses associados usando estereótipos UML

O gerenciamento de variabilidades é uma atividade muito importante no gerenciamento de uma LPS. Em síntese variabilidade é a forma como os membros de uma família de produtos podem se diferenciar entre si, ou seja, é o que permite distinguir os diversos produtos de uma LPS.

A variabilidade é descrita por pontos de variação e variantes:

- **Ponto de variação:** Um local específico de um artefato em que uma decisão de projeto ainda não foi tomada, ou seja, foi adiada;
- **Variante:** Corresponde a uma alternativa de projeto para resolver uma determinada variabilidade.
- **Restrições entre variantes:** define os relacionamentos entre duas ou mais variantes para que seja possível resolver um ponto de variação ou uma variabilidade.

A aplicação destes conceitos é apresentada na Figura 2.

[1] Adaptação e complementação do material gentilmente cedido por Anderson Marcolino e Edson Oliveira Junior (DIN/UEM).

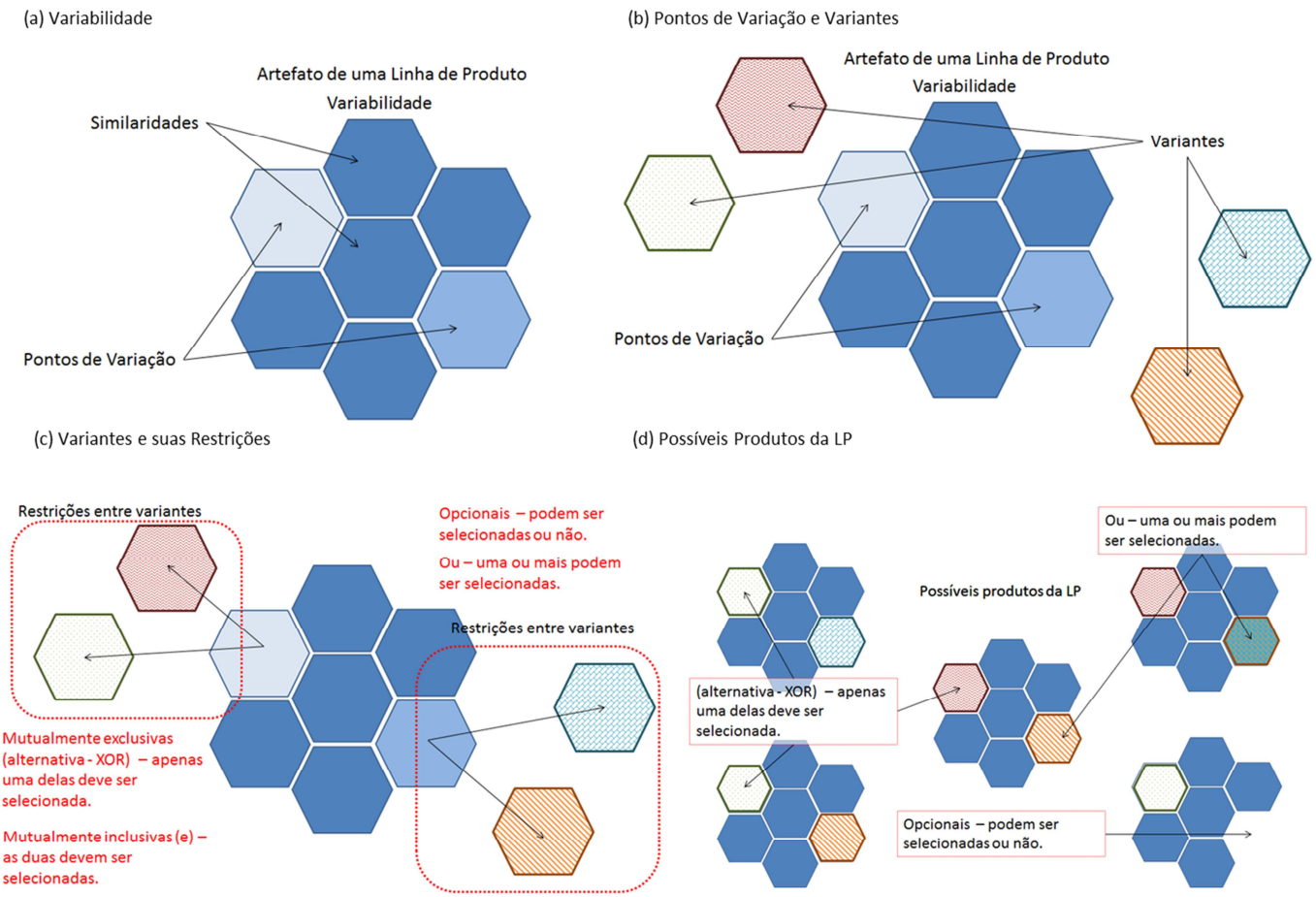

Figura 2 – Exemplo dos Conceitos de Variabilidade, Pontos de Variação, Variantes e Restrições entre Variantes.

II. Abordagem SMarty para Gerenciamento de Variabilidades

A maioria das abordagens desenvolvidas para auxiliar no gerenciamento de variabilidades envolvem diversos conceitos e modelos de representação. A abordagem SMarty apresentada a seguir possui como base a UML – *Unified Modeling Language*. Esta abordagem utiliza diversos conceitos da UML como o uso de estereótipos e diagramas. Entre os itens que a SMarty contempla, temos os que seguem na Tabela I.

Tabela I – Visão Geral SMarty

Item	Identificação
Baseada em UML	SMarty utiliza os modelos UML, meta atributos, etc., como forma de representação da LPS e de suas variabilidades.
Perfil	SMarty apresenta um perfil específico que é formado por estereótipos e meta atributos, geralmente derivados de uma linguagem de modelagem, como a UML.
Processo	O processo contempla a sistematização da utilização de um perfil para o gerenciamento de variabilidades, guiando o usuário no uso das definições do perfil.
Estereótipos	Estereótipos, como os da UML, são um padrão de mecanismo de extensão e são usados para distinguir diferentes tipos de elementos modelados. Em LPS são ferramentas úteis para identificar variabilidade, seus pontos de variação, variantes e outros itens necessários ao seu gerenciamento, tal como interesses associados aos elementos arquiteturais. SMarty usa estereótipos específicos padrões para representar variabilidades em todos os modelos UML.
Diretrizes	São os passos sistematizados, definidos no processo, que permitem a aplicação facilitada do perfil da abordagem a que corresponde.

II.1 Estereótipos e Diretrizes

Nesta seção são apresentados os estereótipos para aplicação em diagrama de classes, existentes no perfil da abordagem SMarty por meio da Tabela II, em seguida são apresentados exemplos do uso destes, seguidos pelas diretrizes para cada tipo de modelo.

Tabela II – Estereótipos da Abordagem SMarty para Classes.

Estereótipos Abordagem SMarty (aplicáveis também aos demais modelos da abordagem)		
Estereótipo	**Utilização**	**Exemplo**
<<variationPoint>>	Representa o local em que ocorre uma variabilidade. Um ponto de variação está sempre associado a uma ou mais variantes.	Figura 3
<<mandatory>>	A variante estará obrigatoriamente presente na configuração de qualquer produto da linha de produto.	Figura 3
<<optional>>	A variante pode ou não estar presente na configuração de um produto da linha de produto. Variantes opcionais também podem ou não estar associadas a um ponto de variação.	Figura 5
<<alternative_OR>>	Estão sempre associadas aos pontos de variação. Pelo menos uma das variantes deverá ser escolhida para resolver o ponto de variação, ou seja, para estar presente na configuração de um produto da linha de produto.	Figura 3
<<alternative_XOR>>	Estão sempre associadas aos pontos de variação. Somente uma das variantes deverá ser escolhida para resolver o ponto de variação.	-
<<variability>>	Indica uma variabilidade existente em um modelo UML.	Figuras 3 e 4
<<requires>>	Indica um relacionamento de dependência (em UML) entre variantes no qual a variante dependente (origem da dependência) só existirá em uma configuração se a variante relacionada (destino da dependência) existir.	Figura 4
<<mutex>>	Indica um relacionamento de dependência (em UML) entre variantes no qual a variante dependente (origem da dependência) só existirá em uma configuração se a variante relacionada (destino da dependência) obrigatoriamente não existir. São conhecidas como variantes mutuamente exclusivas.	Figura 4

II.1.1 Exemplos

Na Figura 3 observamos a aplicação da abordagem SMarty, e seus elementos. Passamos a analisar cada um deles, bem como as diretrizes presentes no processo da SMarty, que auxiliam sua utilização em outras LPSs:

A classe **AlgoritmoOrdenacao** identifica uma classe obrigatória (<<*mandatory*>>) e representa também um ponto de variação (<<*variationPoint*>>), com três variantes. Estas variantes estão descritas no elemento comentário, relacionado a classe, por meio do *Tagged Value (***variants**). As três variantes desta classe são **OrdenacaoPorTroca**, **OrdenacaoPorSelecao** e **OrdenacaoPorInsercacao**. Todas estas são estereotipadas como <<*alternative_OR*>>, o que indica o tipo de restrição para tais variantes, neste caso, significa que ao menos uma ou todas elas podem solucionar o ponto de variação.

OrdenacaoPorTroca, **OrdenacaoPorSelecao** e **OrdenacaoPorInsercacao**, além de variantes, são, por sua vez, pontos de variação (<<*variationPoint*>>), e assim, cada uma delas apresenta um comentário, que descreve as suas variantes (**variants**), bem como o nome da mesma (**name**). Neste caso, todas as variantes são marcadas como <<*alternative_OR*>> e, como anteriormente, uma delas , ao menos, deve ser selecionada ou todas.

A classe **ProgramaPrincipalOrdenacao**, representa uma classe obrigatória, portanto é marcada como <<*mandatory*>>, e estará presente em todos os produtos desta LPS.

A classe **ElementoOrdenacao**, também é obrigatória (<<*mandatory*>>) e representa um ponto de variação (<<*variationPoint*>>). Ela está ligada a um comentário com o estereótipo <<*variability*>>, que identifica os dados da variabilidade, que é nomeada, por exemplo, de "sorting element" e possui duas classes variantes (**variants**): **ElementoNumerico** e **ElementoString**, marcadas como variantes alternativas <<*alternative_OR*>>, onde, ambas podem ser selecionadas, ou ao menos uma.

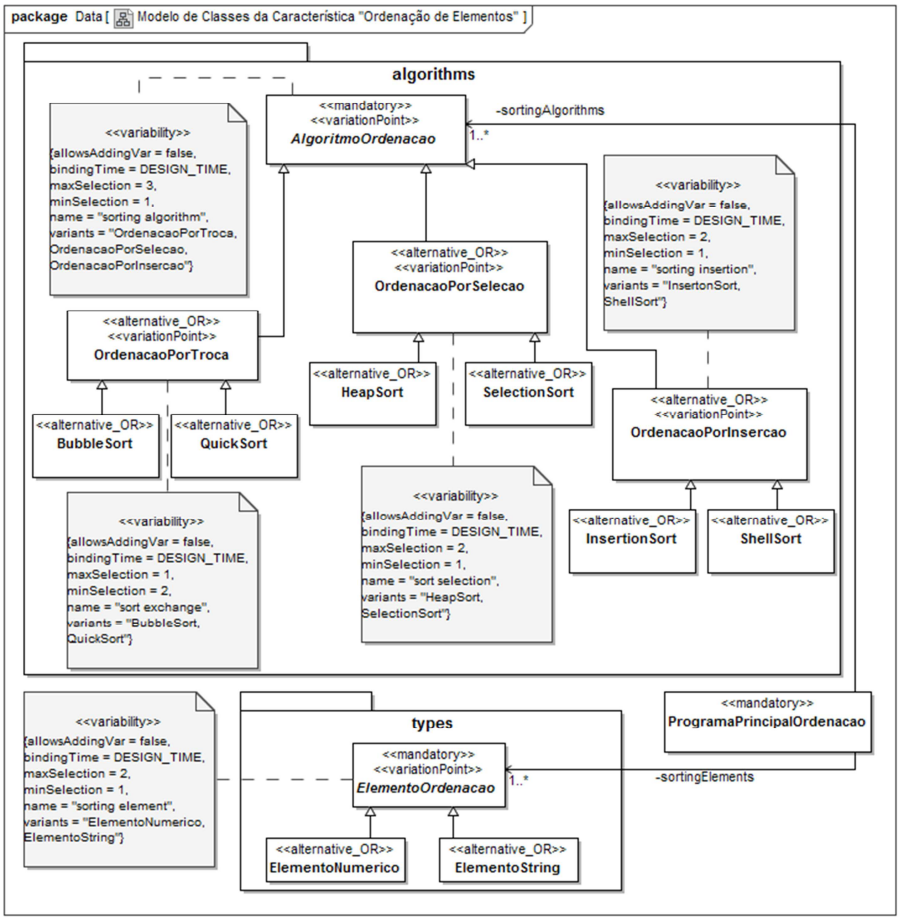

Figura 3 – Exemplo de Modelo de Variabilidade em Diagrama de Classes com a Abordagem SMarty.

Desta forma, as variabilidades são identificadas por meio do comentário UML, estereotipada com <<*variability*>>. Nestes comentários estão contidos os meta atributos que seguem:

- ***Name***: nome da variabilidade;
- ***minSelection***: a quantidade mínima de variantes a serem selecionadas;
- ***maxSelection***: a quantidade máxima de variantes a serem selecionadas;
- ***bindingTime***: em qual momento a variabilidade será resolvida;
- ***allowsAddingVar***: se permite incluir novas variantes para resolver o ponto de variação; e
- ***variants***: quais as variantes para resolver o ponto de variação (casos de uso ligados ao ponto de variação).

Estas notas são inseridas em todas as variabilidades.

Diretrizes para Identificação e Representação de Variabilidade - Para a criação e classificação de variabilidades, as seguintes diretrizes presentes no processo da abordagem, sugerem ser seguidas:

D.3.1 Variabilidades com variantes opcionais possuem multiplicidade ***minSelection = 0*** e ***maxSelection = 1***;
D.3.2 Variabilidades com variantes exclusivas possuem multiplicidade ***minSelection = maxSelection = 1***;
D.3.3 Variabilidades com variantes inclusivas possuem multiplicidade ***minSelection = 1*** e ***maxSelection = size(variants)*** em que ***size(x)*** é uma função que retorna a quantidade de elementos da coleção ***x***;
D.3.4 O valor ***bindingTime*** deve ser definido escolhendo-se um dos valores da classe de enumeração ***BindingTime***, que são: DESIGN_TIME, LINK_TIME, COMPILE_TIME, RUNTIME;
D.3.5 O valor *booleano* do atributo ***allowsAddingVar*** deve ser analisado de acordo com a possibilidade de manter o ponto de variação aberto (*true*) ou fechado (*false*); e
D.3.6 O valor da coleção ***variants*** é o conjunto formado pelas instâncias das variantes associadas ao ponto de variação ou variabilidade.

Diretrizes para Diagrama de Classes - As diretrizes especificadas para auxiliar na identificação das variabilidades em diagramas de classes são expressas abaixo:

D.2.2 Em modelos de classes, pontos de variação e suas variantes são identificadas nos seguintes relacionamentos:

a) **generalização**, os classificadores mais gerais são os pontos de variação, enquanto os mais específicos são as variantes;
b) **realização de interface**, os "*suppliers*" (especificações) são os pontos de variação e as implementações (clientes) são as variantes;
c) **agregação**, as instâncias tipadas com losangos não preenchidos são os pontos de variação e as instâncias associadas são as variantes; e
d) **composição**, as instâncias tipadas com losangos preenchidos são os pontos de variação e as instâncias associadas são as variantes.

D.2.4 Elementos de modelos de classes, relacionados à associações nas quais os seus atributos *aggregationKind* possuem valor *none*, ou seja, não representam nem agregação nem composição, sugerem variantes obrigatórias ou opcionais. Na Figura 3, a classe **ProgramaPrincipalOrdenacao**, é um exemplo de variante obrigatória.

Elementos de modelos de casos de uso relacionados aos mecanismos de extensão e de pontos de extensão sugerem pontos de variação com variantes associadas, as quais podem ser inclusivas ou exclusivas;
D.2.3 Em modelos de caso de uso relacionadas com a associação de inclusão (<<*include*>>) ou associados a atores sugerem variantes obrigatórias ou opcionais;
D.2.5 Variantes que, ao serem selecionadas para fazer parte de um produto, exigem a presença de outra(s) determinada(s) variante(s) devem ter seus relacionamentos de dependência marcados com o estereótipo <<*requires*>>;
D.2.6 Variantes mutuamente exclusivas para um determinado produto devem ter seus relacionamentos de dependência marcados com o estereótipo <<*mutex*>>.
D.2.7 Componentes formados por classes com variabilidades são marcados com o estereótipo <<variable>>.

Na Figura 4, que representa fragmento de um diagrama de classes, nele notas o uso dos estereótipos para identificar a dependência entre classes, bem como a seleção mutualmente exclusiva.

A classe **ElementoNumerico** requer a presença da classe **QuickSort**, para que possa ser incluída no produto, já a classe **ElementoString** restringe que a classe **InsertionSort** não seja inserida no

produto, para que possa ser selecionada, ou seja, se selecionada a classe **ElementoString** como variante para o ponto de variação **ElementoOrdenacao**, a classe **InsertionSort**, que é ponto de variação de outra classe, não pode ser selecionada. No caso da classe **ElementoNumerico**, ser selecionada, a classe **QuickSort**, deverá ser selecionada como variante para o ponto de variação a qual pertence.

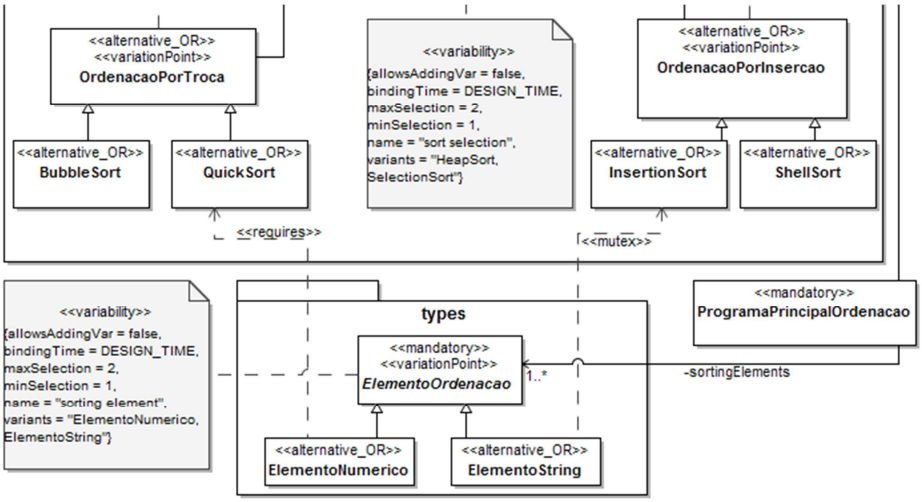

Figura 4 – Exemplo de Identificação de Variabilidade em Classes – *requires e mutex.*

A Figura 5 representa a aplicação do estereótipo <<*optional*>>. A classe **SaveGame** é opcional, e assim é marcada como uma variabilidade, pelo comentário da UML – quando opcional a classe pode ou não estar inserida no produto da LPS.

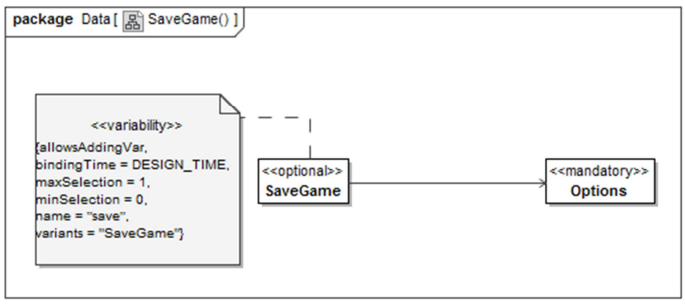

Figura 5 – Exemplo de Identificação de Variabilidade em Classes – *optional.*