Avaliação experimental do uso de algoritmos meméticos na otimizaçao de projeto de arquitetura de linha de produto de software

João Choma Neto

Orientadora: Prof. Dra. Thelma Elita Colanzi Lopes
Adaptação do material: Guilherme Conti e Anderson da Silva Marcolino

Linha de Produto de Software

- Abordagem que objetiva promover a geração de produtos específicos com base na reutilização de uma infraestrutura central - núcleo de artefatos - formada por uma arquitetura de software e seus componentes.

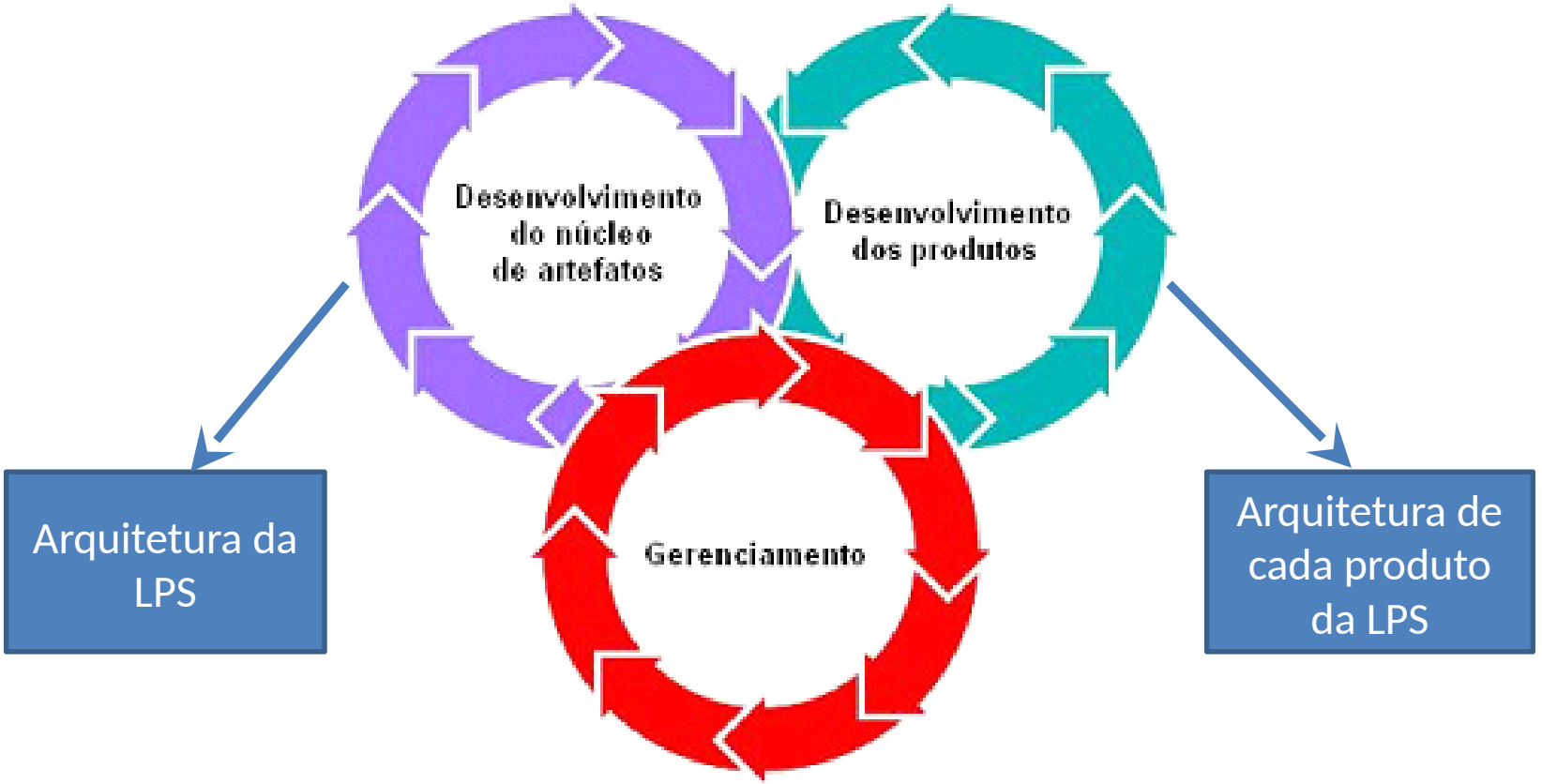
- Por meio desta abordagem, é possível explorar as semelhanças dos seus produtos para aumentar o **reúso** de artefatos.

Linha de Produto de Software

Quais seriam as possíveis diferenças entre os produtos dessa linha de produção?

Linha de Produto de Software

- Atividades Essenciais de LPS



Linha de Produto de Software

- A base de uma LPS é seu **núcleo de artefatos**.
  - arquitetura da LPS, os componentes reusáveis, modelos de domínio, requisitos da LPS, modelos de características e planos de teste.

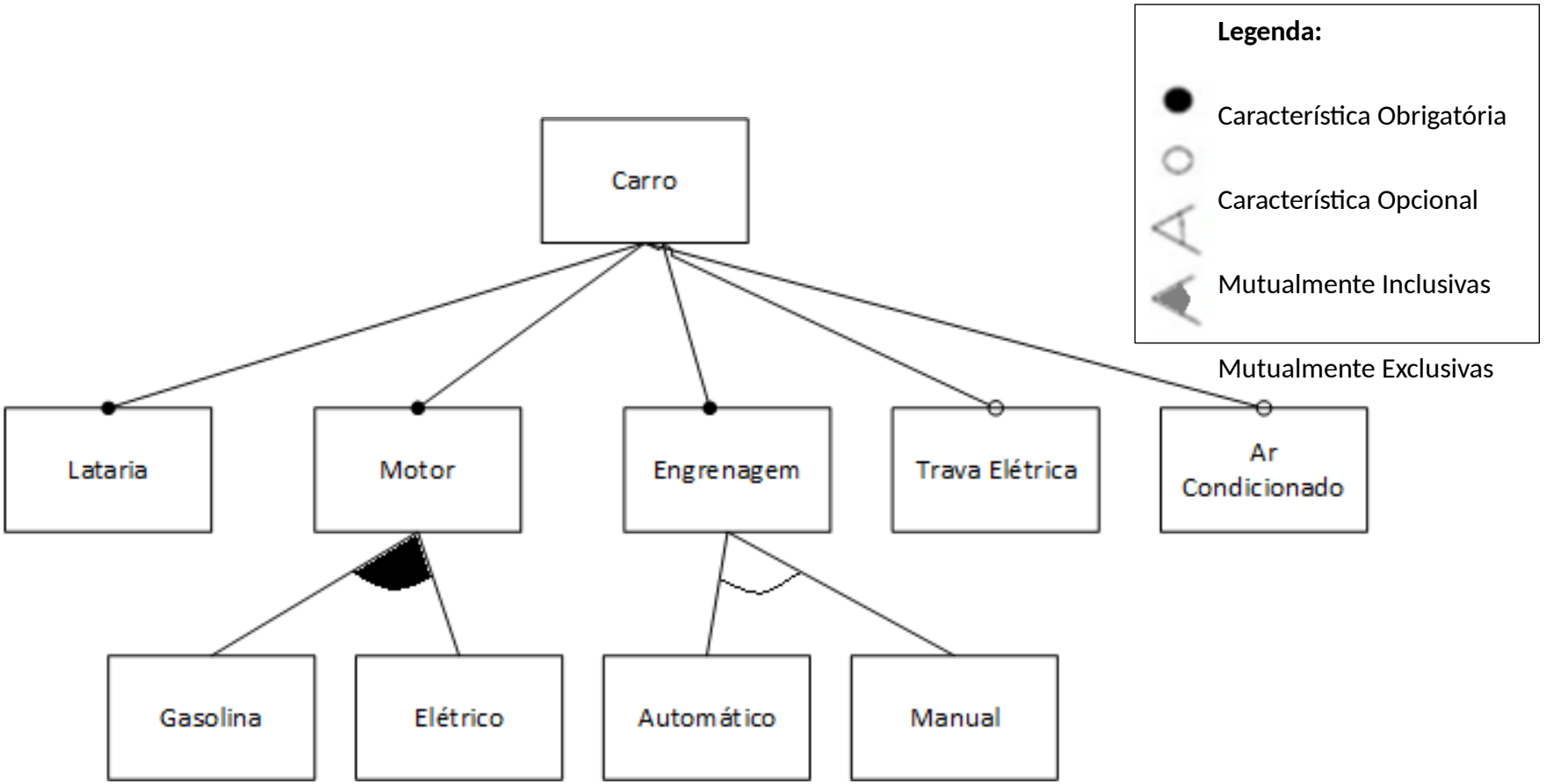
- Uma **característica** (*feature*) é uma capacidade do sistema que é relevante e visível para o usuário final.

Linha de Produto de Software

- O modelo de características inclui todas as características de uma LPS e os seus inter-relacionamentos.
- A definição explícita de **variabilidades** em LPS é a diferença chave entre o desenvolvimento de sistemas únicos e o desenvolvimento de LPS.

Linha de Produto de Software

- Exemplo de Modelo de Características



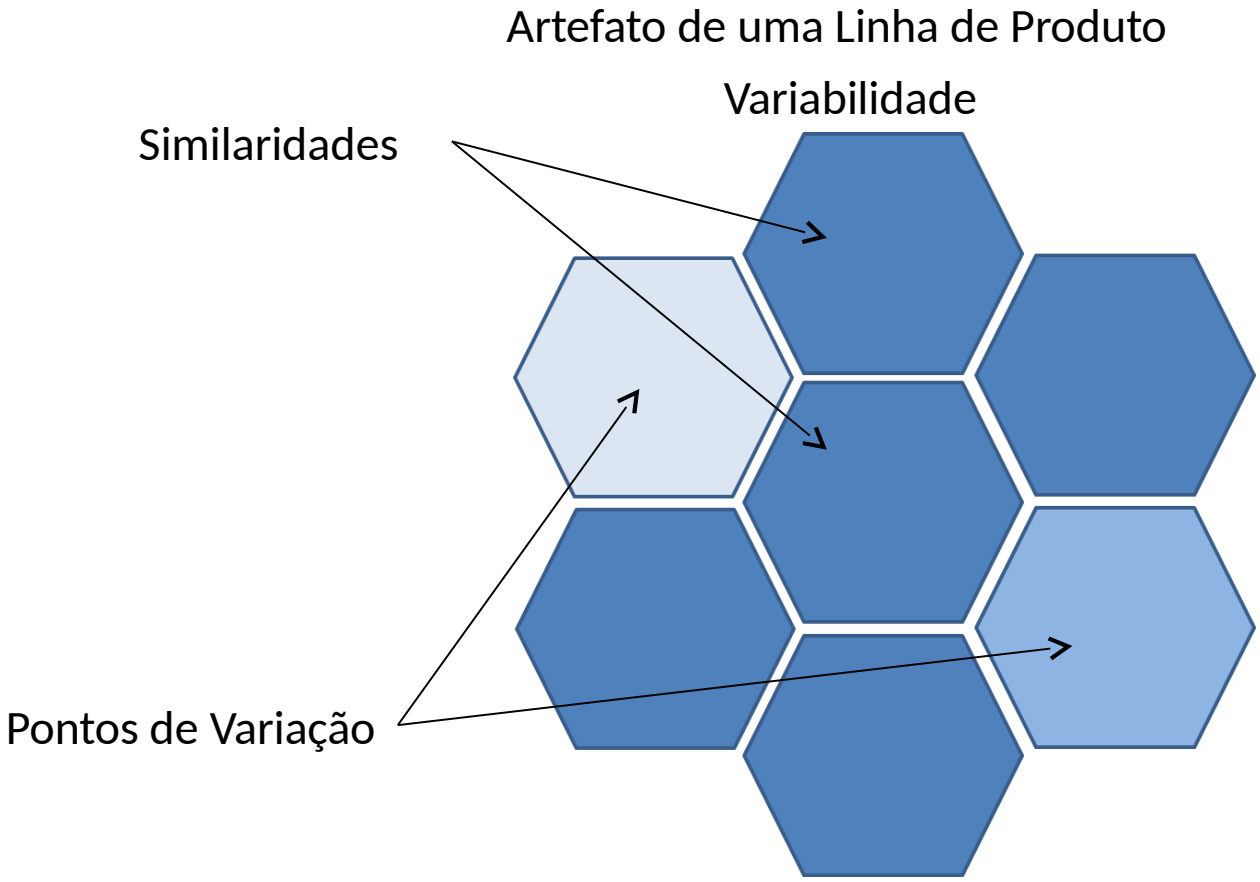
Linha de Produto de Software

- Variabilidade é a forma como os membros de uma família de produtos podem se diferenciar entre si.
- O gerenciamento de variabilidades é uma das atividades mais importantes no gerenciamento de uma LPS.
- A variabilidade é descrita por pontos de variação e variantes.

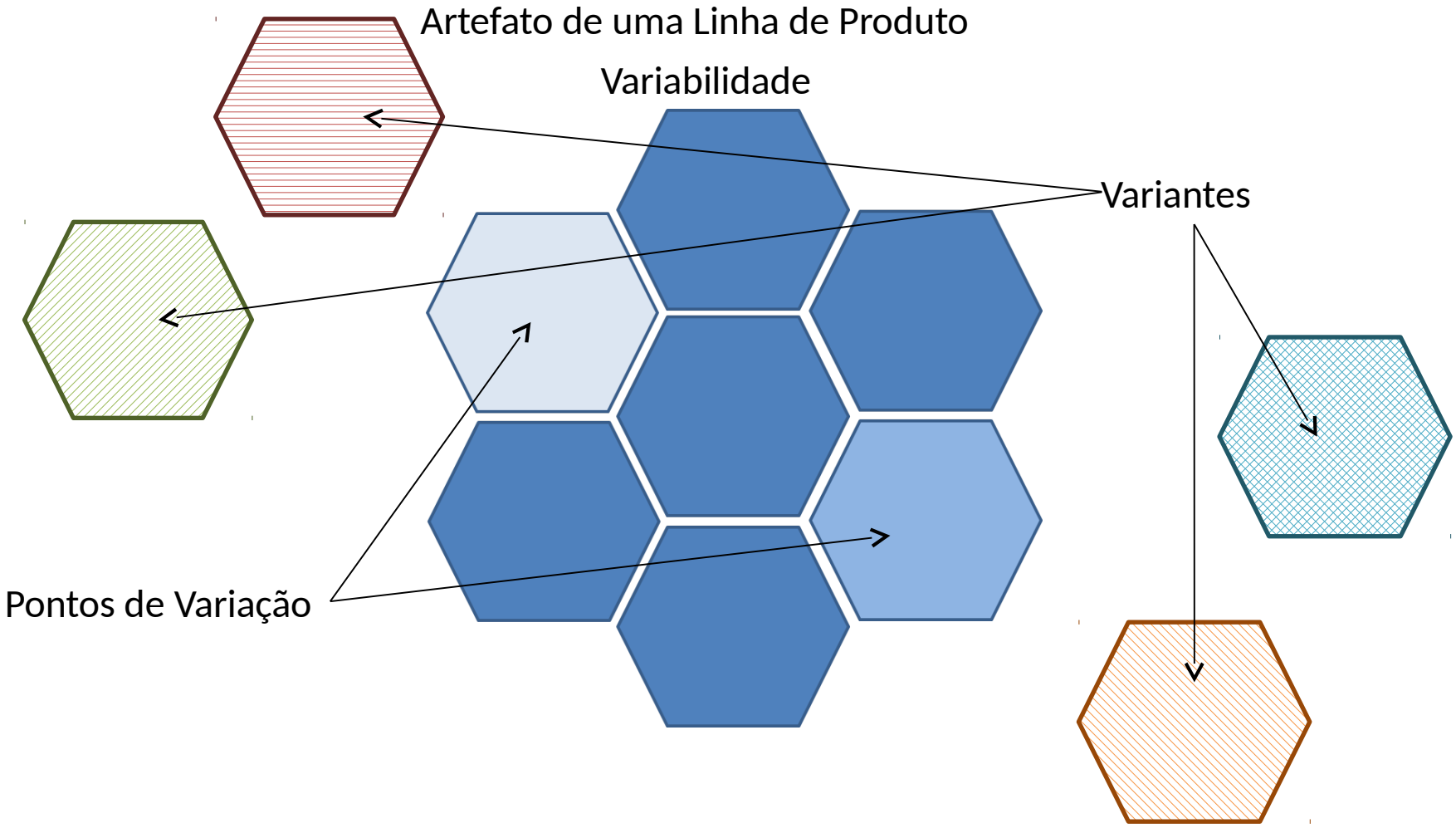
Linha de Produto de Software

- **Ponto de variação:** Um **local específico** de um artefato em que uma decisão de projeto ainda não foi tomada;
- **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.

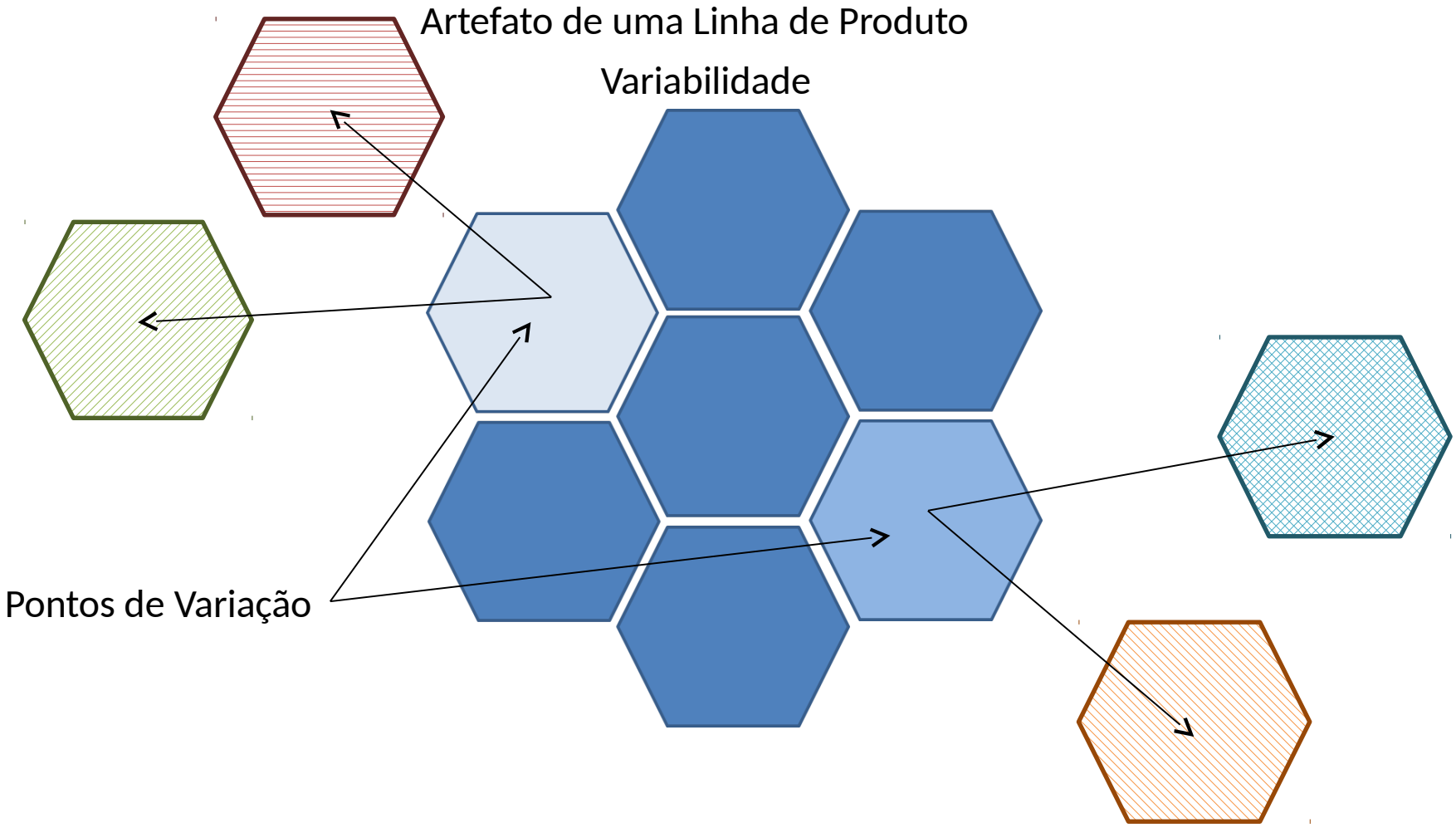
Gerenciamento de Variabilidade



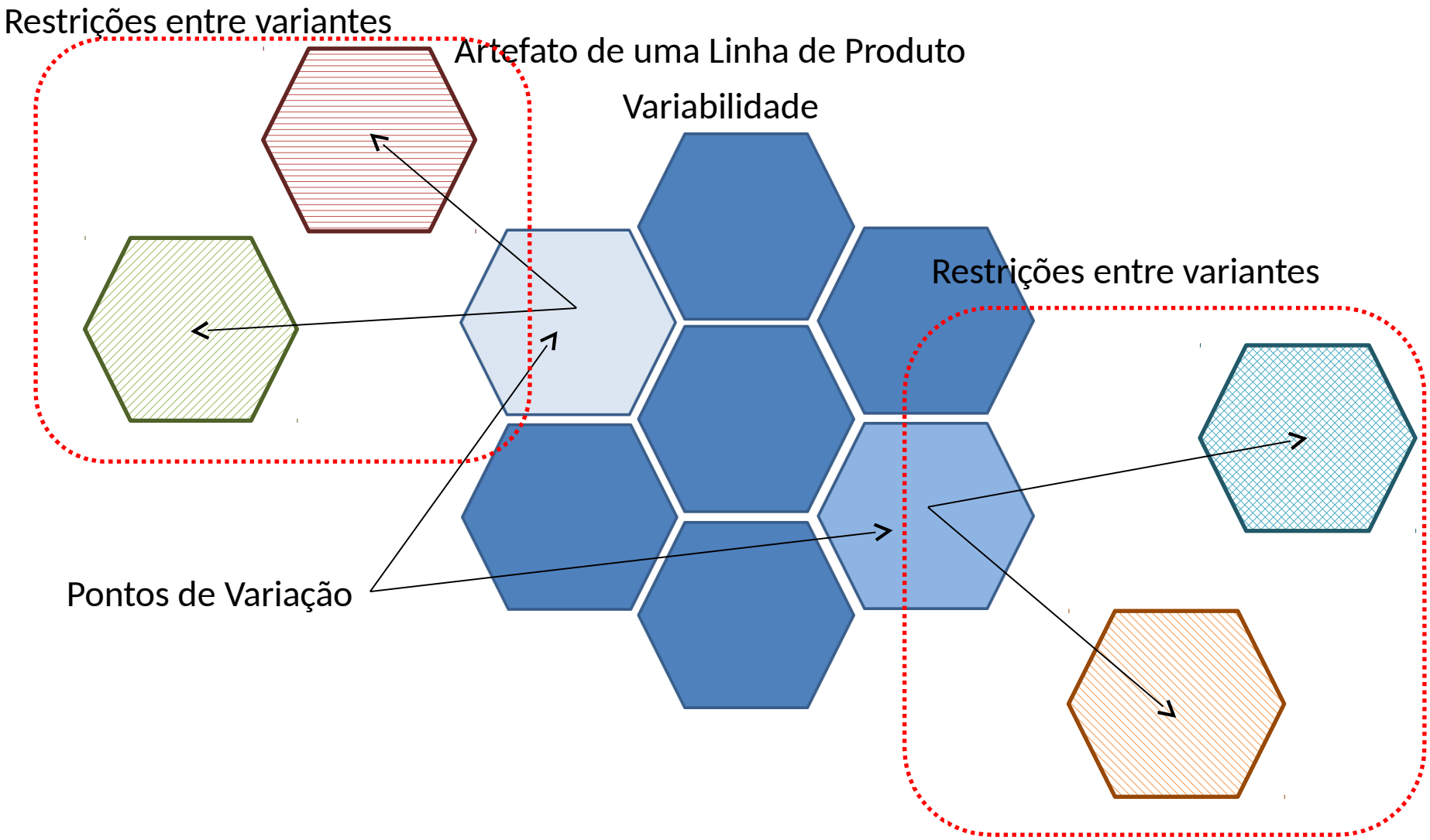
Gerenciamento de Variabilidade



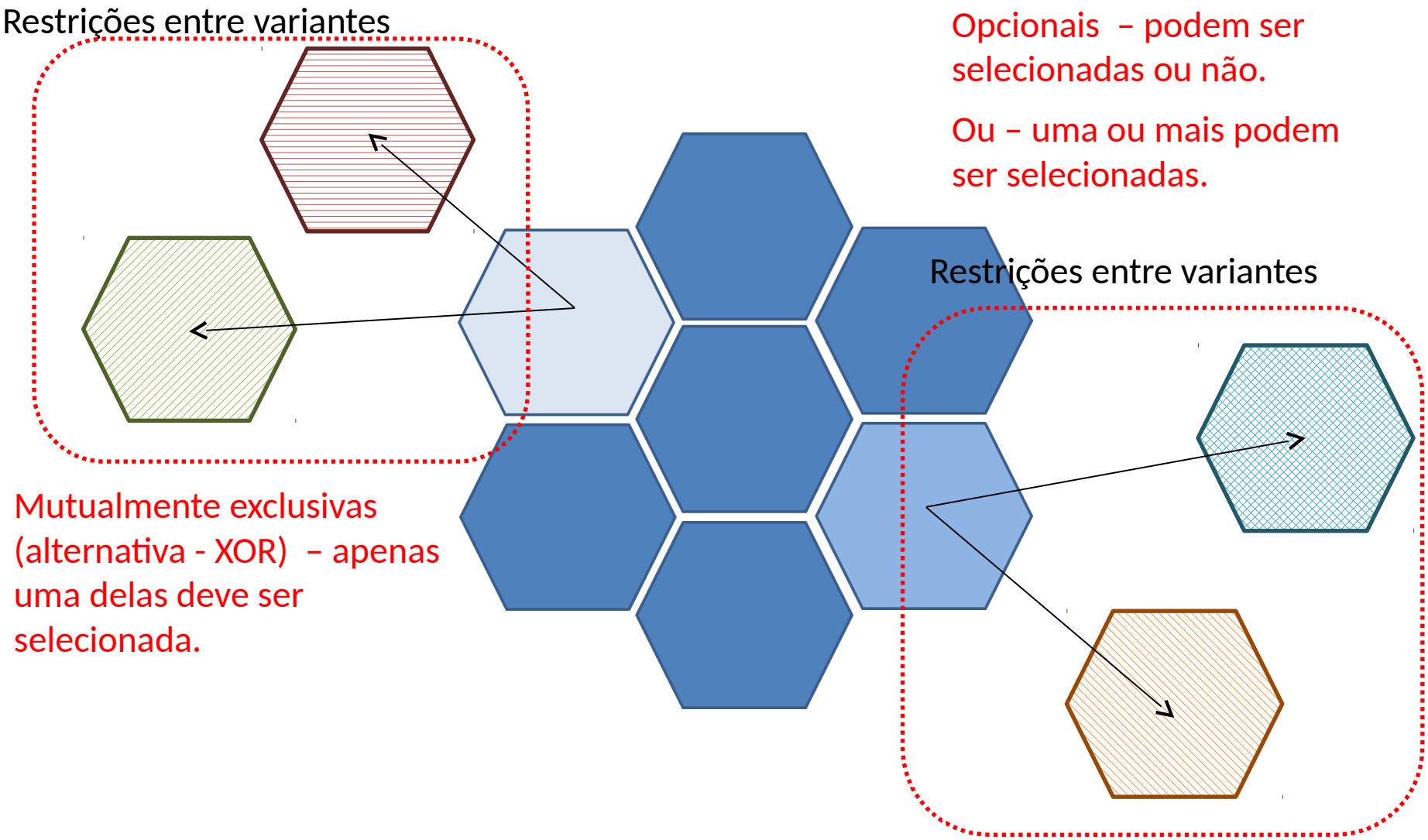
Gerenciamento de Variabilidade



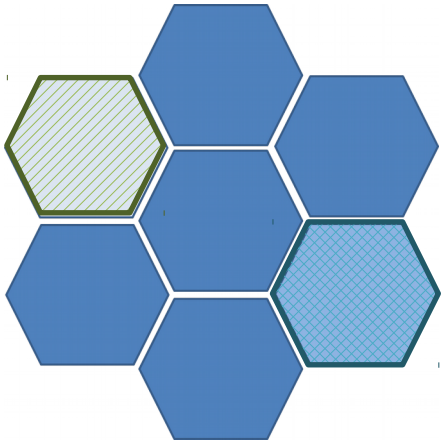
Gerenciamento de Variabilidade



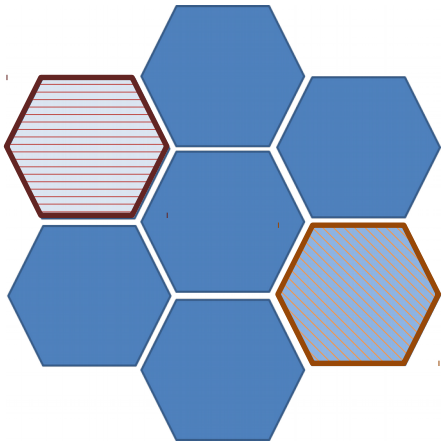
Gerenciamento de Variabilidade

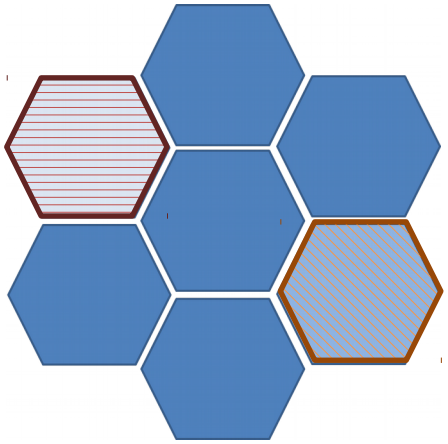


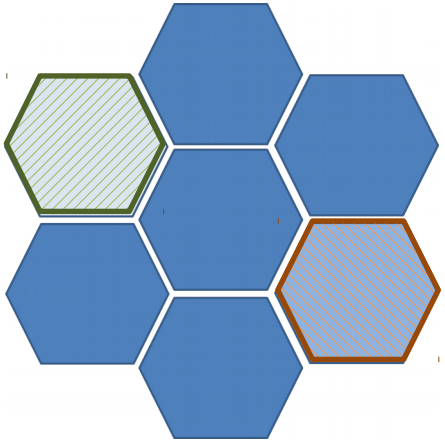
Gerenciamento de Variabilidade

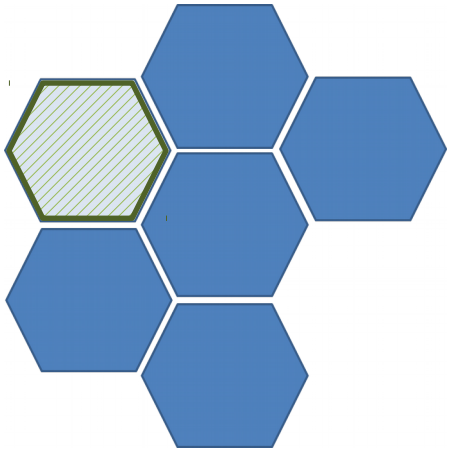


Possíveis produtos da LPS





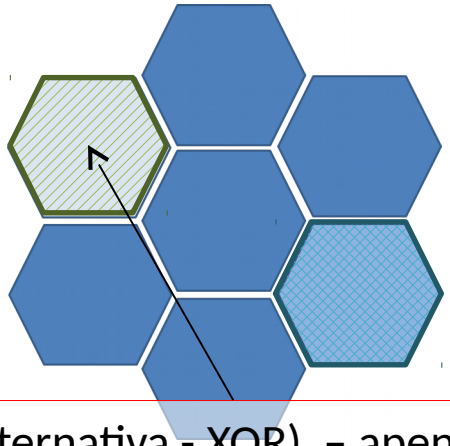


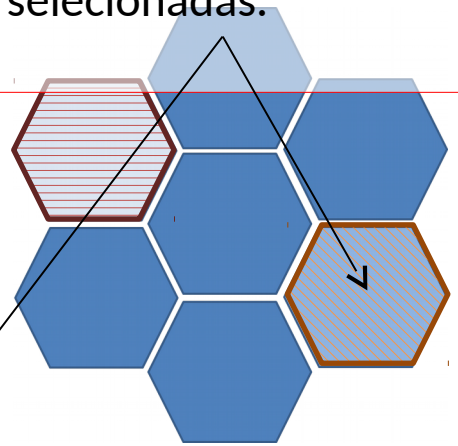


Gerenciamento de Variabilidade

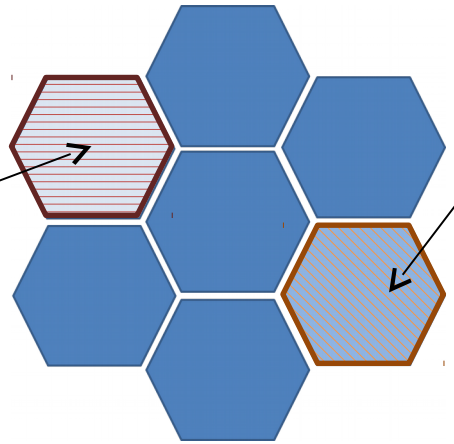
Ou – uma ou mais podem ser selecionadas.

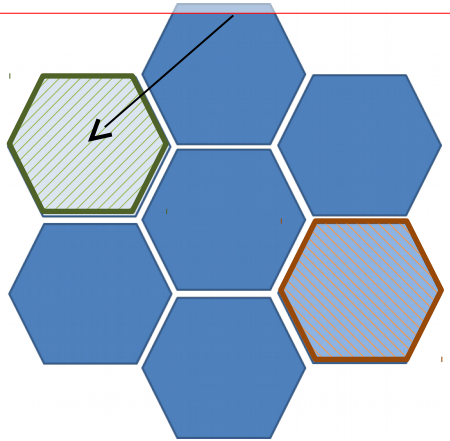
Possíveis produtos da LPS



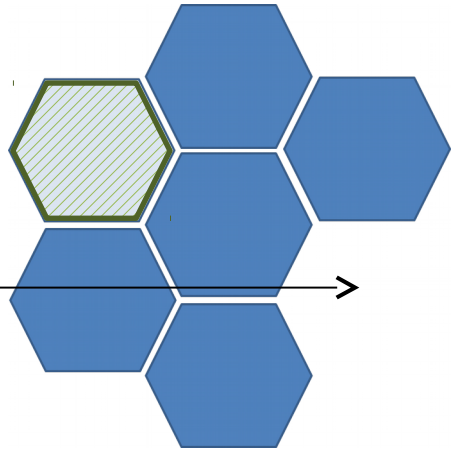


(alternativa - XOR) – apenas uma delas deve ser selecionada.





Opcionais – podem ser selecionadas ou não.



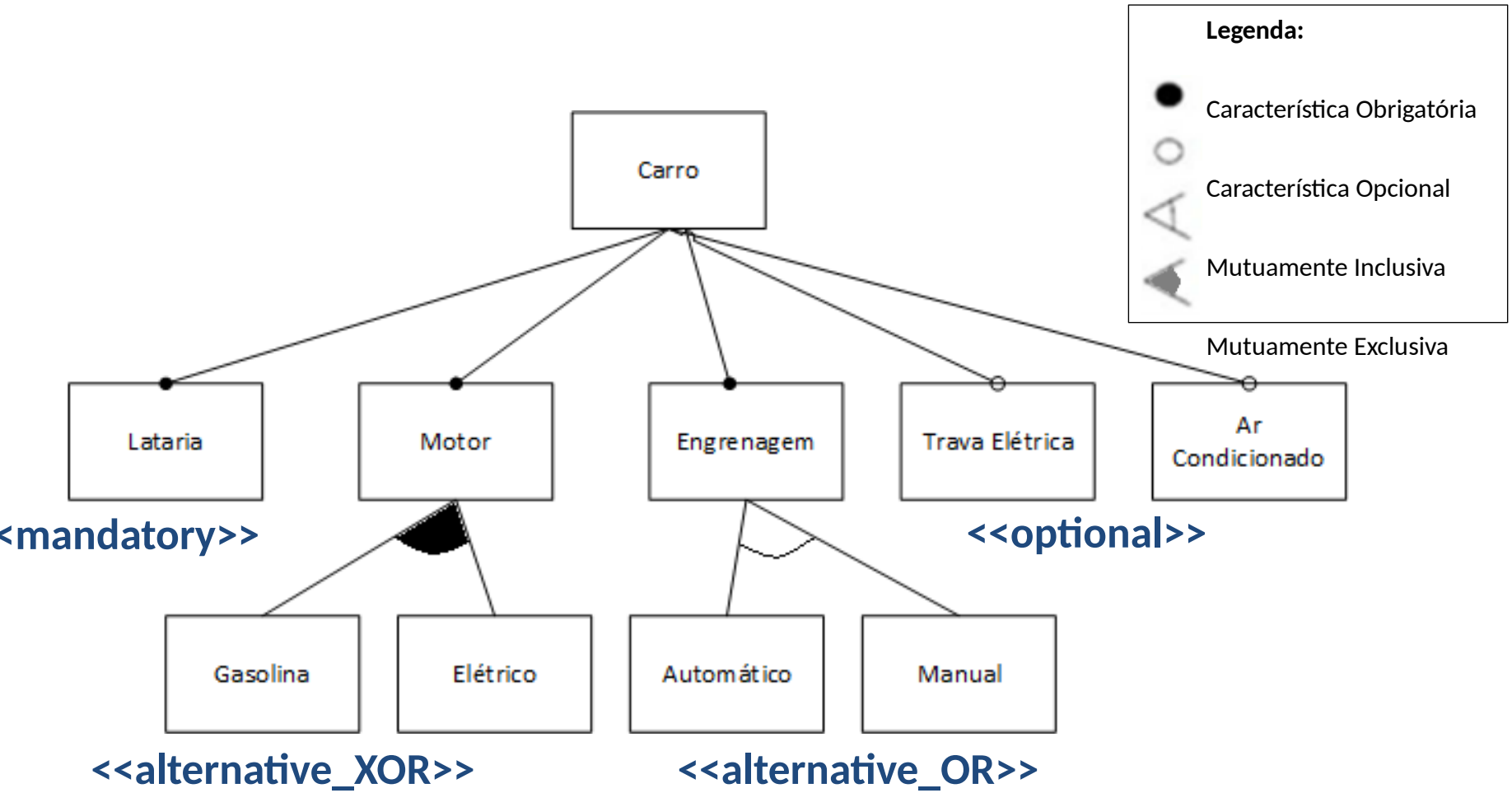
Abordagem SMarty

- Desenvolvida para auxiliar no gerenciamento de variabilidades;
- Possui como base a UML – *Unified Modeling Language*;
- Utiliza diversos conceitos da UML como o uso de estereótipos e diagramas.

Abordagem SMarty

Estereótipos Abordagem SMarty	
Estereótipo	**Utilização**
<<variationPoint>>	Representa o local em que ocorre uma variabilidade. Um ponto de variação está sempre associado a uma ou mais variantes.
<<mandatory>>	A variante estará obrigatoriamente presente na configuração de qualquer produto da linha de produto.
<<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.
<<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.
<<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.
<<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.
<<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.

Abordagem SMarty

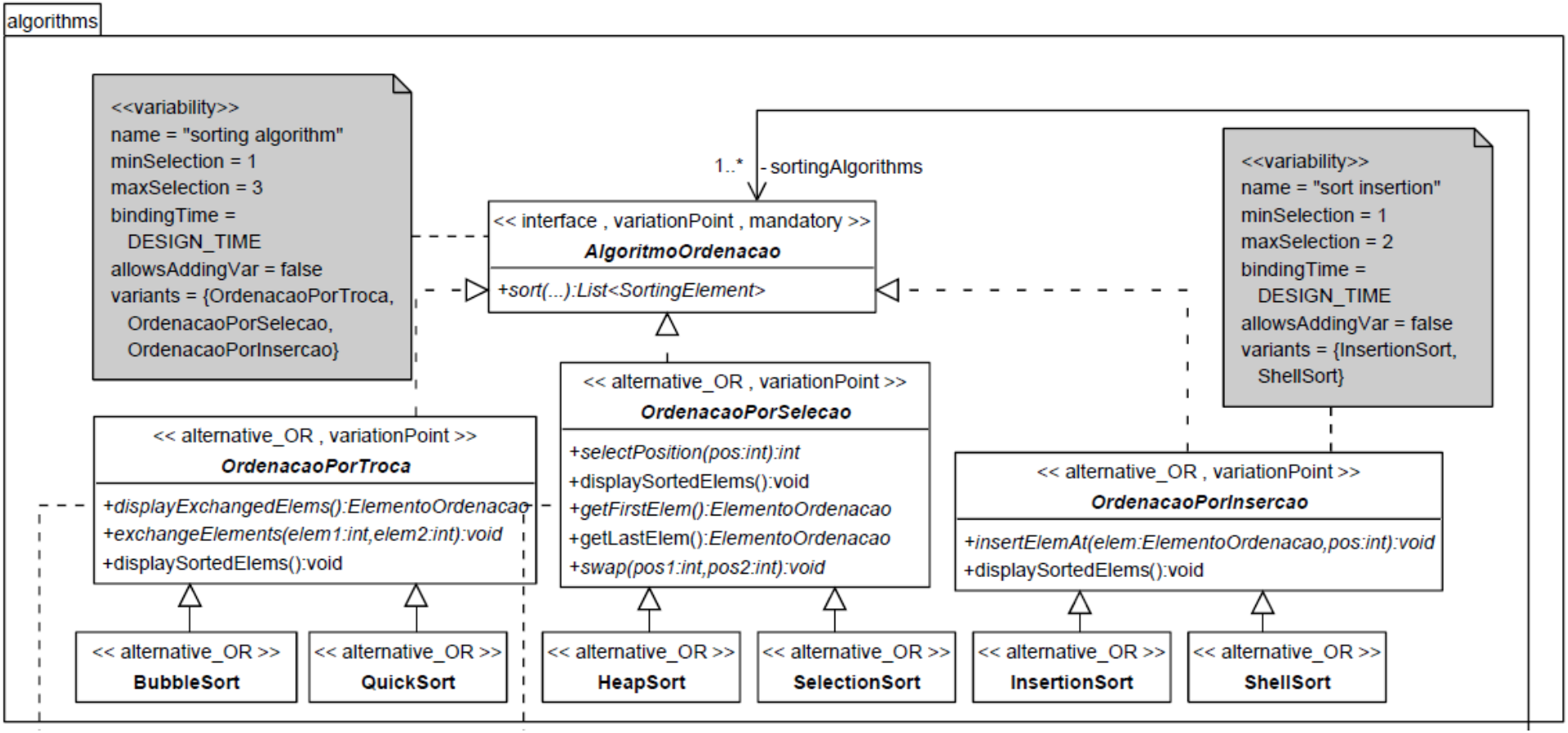


Abordagem SMarty

- As variabilidades são identificadas por meio do comentário UML, estereotipada com <<*variability*>> contendo os seguintes meta atributos:
  - ***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.

Abordagem SMarty



Exemplo Parcial de Diagrama de Classes com Representação de Variabilidades

Métricas Arquiteturais

Tem como objetivo avaliar algumas qualidades do software, como:

- Acoplamento entre componentes;
- Coesão de um componente;
- Tamanho de uma interface;
- Modularização de características;

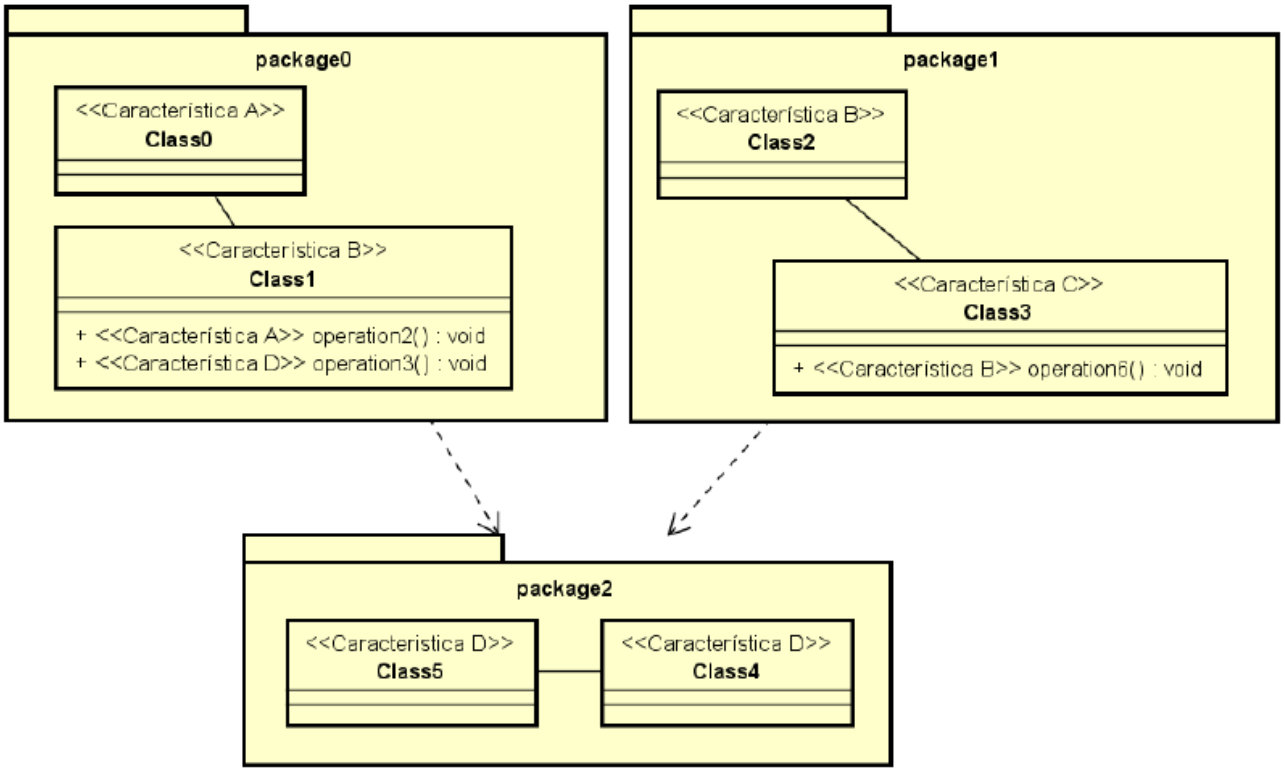
Métricas Convencionais

- **Coesão relacional**: número médio de relacionamentos internos entre as classes e interfaces de um pacote;
- **Acoplamento**: relacionamento entre classes ou entre classes e interfaces ou ainda entre pacotes;
- **Tamanho**: número de operações de uma interface.

Métricas Dirigidas a Características

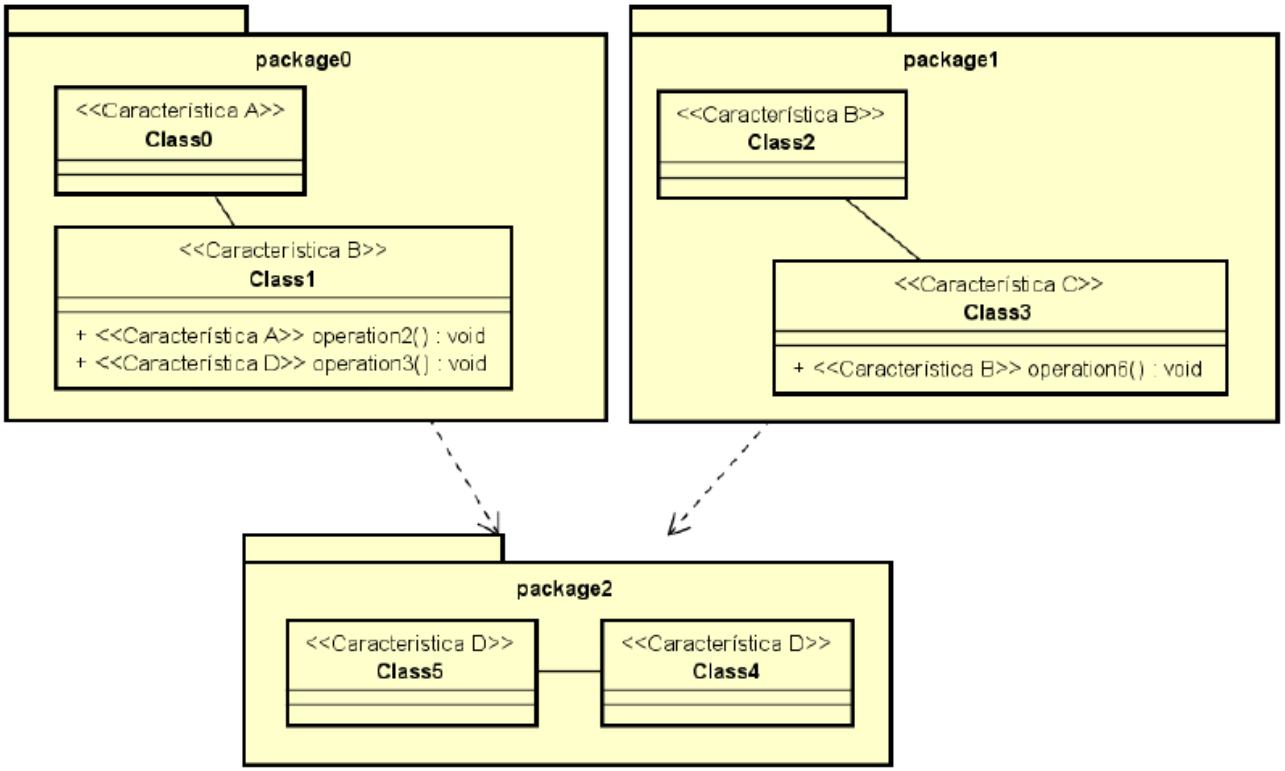
- Utilizada para identificar e avaliar modularização de características;
- Propriedades arquiteturais:
  - **Difusão de características:** conta o número de elementos arquiteturais associados a cada característica;
  - **Interação entre características:** conta o número de características com os quais uma dada característica compartilha ao menos um elemento arquitetural;
  - **Coesão baseada em características:** conta o número de características para cada pacote.

Métricas Dirigidas a Características



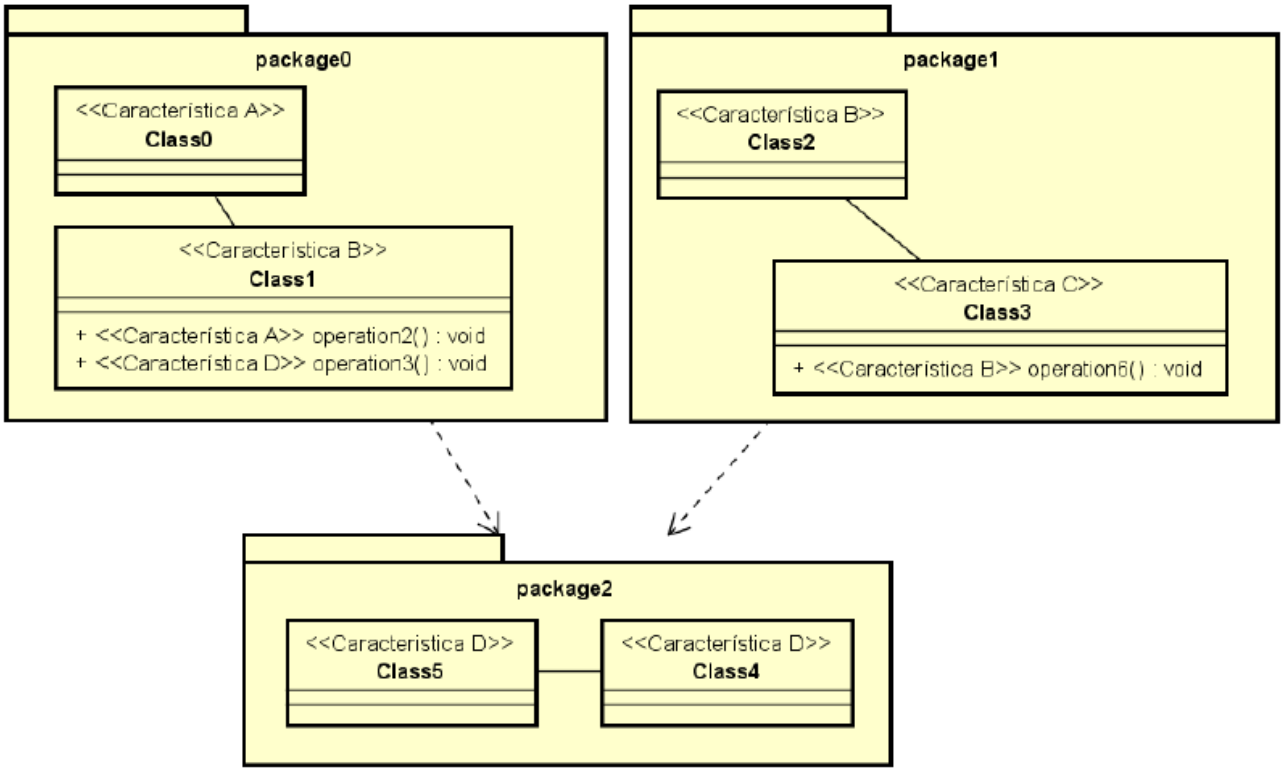
Difusão de características: a característica A está difusa em Class0 e operation2 (de Class1), a característica B está difusa em Class1, Class2 e operation6 (de Class3). Quanto menor a difusão, melhor porque aumenta a modularização das características e isso facilita manutenção e reuso.

Métricas Dirigidas a Características



Entrelaçamento de características: A característica B está entrelaçada com as características A e D na Class1 (e vice-versa). A característica C está entrelaçada com a características B em Class3 (e vice-versa). Quanto menor o entrelaçamento melhor porque aumenta a modularização das características facilitando a manutenção e o reuso.

Métricas Dirigidas a Características



Coesão baseada em características: o pacote package2 realiza somente a característica D, por isso ele é altamente coeso. O package0 é pouco coeso já que está associado as características A, B e D. Bons projetos devem ter o maior número de pacotes altamente coesos possível.

ANOTAÇÕES

- Uma das formas de se implementar características de LPS é tratando cada uma delas como um **interesse** presente na família.

- Interesses correspondem a requisitos, funcionais ou não, que devem estar presentes no sistema.

- Alguns são naturalmente transversais e, por isso, se espalham em vários pontos dentro do software.

- Quanto mais modularizado um interesse estiver dentro do projeto do software mais reusável será o seu projeto.

Os interesses podem ser mapeados em artefatos da LPS usando estereótipos UML.

ANOTAÇÕES

- Métricas convencionais são avaliadas pela função **CM**

- **CM(pla):** seu objetivo é alcançar soluções com alta coesão, baixo acoplamento e alta reusabilidade. A função é constituída pela soma de várias métricas convencionais (Colanzi, 2014)

ANOTAÇÕES

- Métricas dirigidas a características são avaliadas pela função **FM**

- **FM(pla):** seu objetivo é alcançar um projeto com alta modularização de características, é formada pela soma de métricas dirigidas a características (Colanzi, 2014)

ANOTAÇÕES

- Com o refinamento da função **CM** surgiu as funções **COE** e **ACLASS**

- **COE(pla):** seu objetivo é medir a coesão do projeto de PLA em termos de relacionamento interno entre classes de componentes mais o número de características associadas a cada componente (Santos et al., 2015)

ANOTAÇÕES

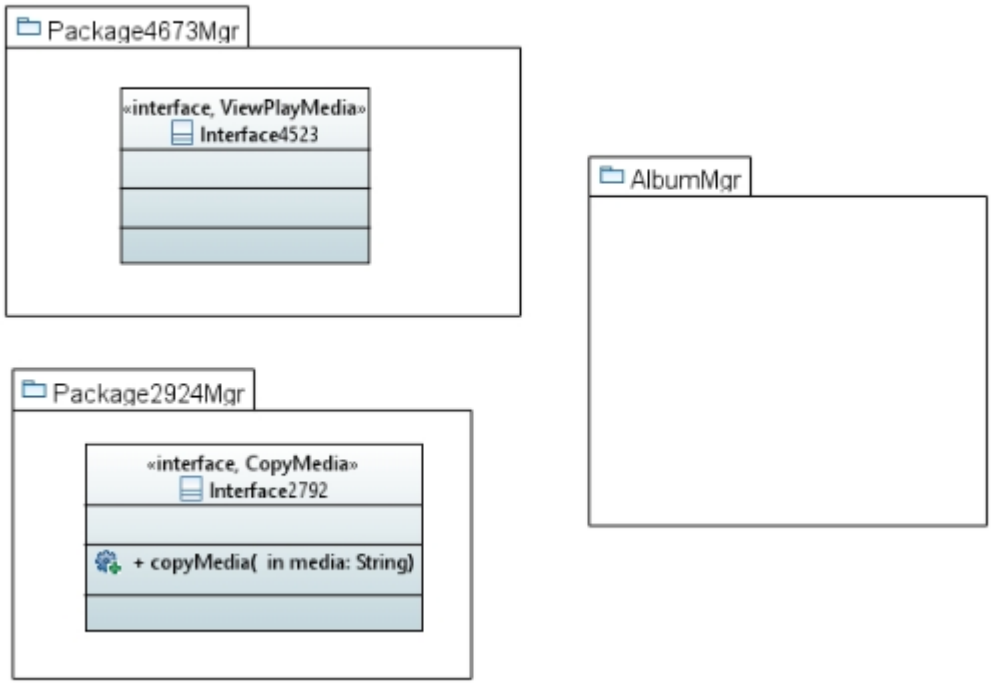
- Com o refinamento da função **CM** surgiu as funções **COE** e **ACLASS**

- **ACLASS(pla):** seu objetivo é medir o número de elementos arquiteturais que tem dependência de outras classes do projeto mais o número de elementos dos quais cada classe depende (Santos et al., 2015)

Considerações Finais

- Alguns problemas nas PLAs já foram identificados na avaliação de outros trabalhos realizados, tais como: **interfaces vazias, classes vazias e pacotes isolados;**
-
- Esses problemas devem ser desconsiderados na avaliação desse trabalho porque estão em fase de correção.

Considerações Finais



Exemplos de interface vazia, classe vazia e pacote isolado.

Experimento

Doc. 1 – Termo de Adesão (papel)

Doc. 2 – Questionário de Caracterização (pacote experimental – online)

Doc. 3 – Conceitos de LPS e Smarty (pacote experimental)

Doc. 4 – Descrição Geral da LPS (pacote experimental)

Doc. 5 – Formulário Experimento (pacote experimental – online)

Doc. 6 – Características da PLA (pacote experimental)