Processo de Desenvolvimento de Software

Linhas de Produtos de Software

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

Departamento Acadêmico de Gestão e Tecnologia da Informação

Curso de Tecnologia em Análise e Desenvolvimento de Sistemas

Prof. Fellipe Aleixo (*fellipe.aleixo@ifrn.edu.br*)

Motivação

Hamburger

Double Hamburger

Cheeseburger

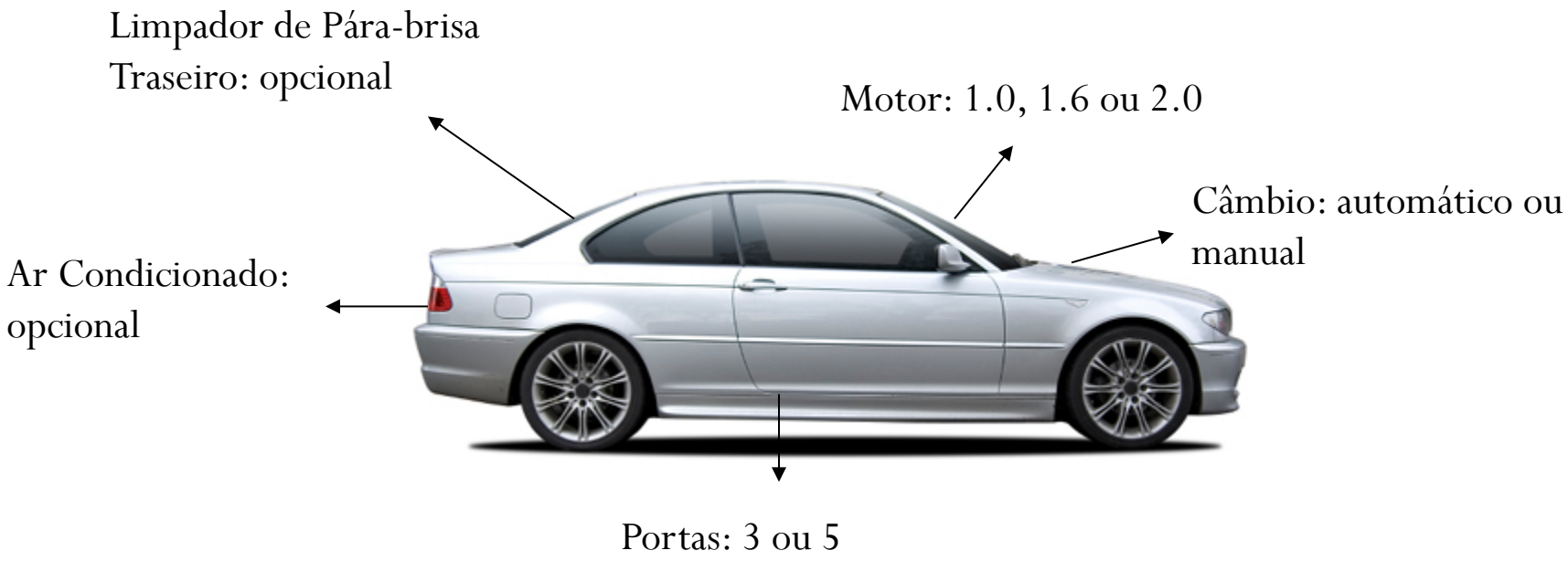
Double Cheeseburger

BK VEGGIE® Burger

Você vê os *componentes*, a *arquitetura* e o *reuso* nesses produtos?

Linhas de Produto

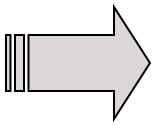
- Idéia de Linha de Produto não é nova
- Exemplos
  - História Antiga: pirâmides do Egito
  - Atualmente: linhas de produtos de carros



Linhas de Produto

- Família de Produtos
  - Características Comuns
  - Características Variáveis
- Customização em Massa
  - Produção em larga escala de bens adaptados de acordo com as necessidades individuais do usuário
- Plataforma
  - Qualquer base de tecnologias sobre a qual outras tecnologias ou processos são construídos

Desenvolvimento baseado em Plataformas
+
Customização em Massa

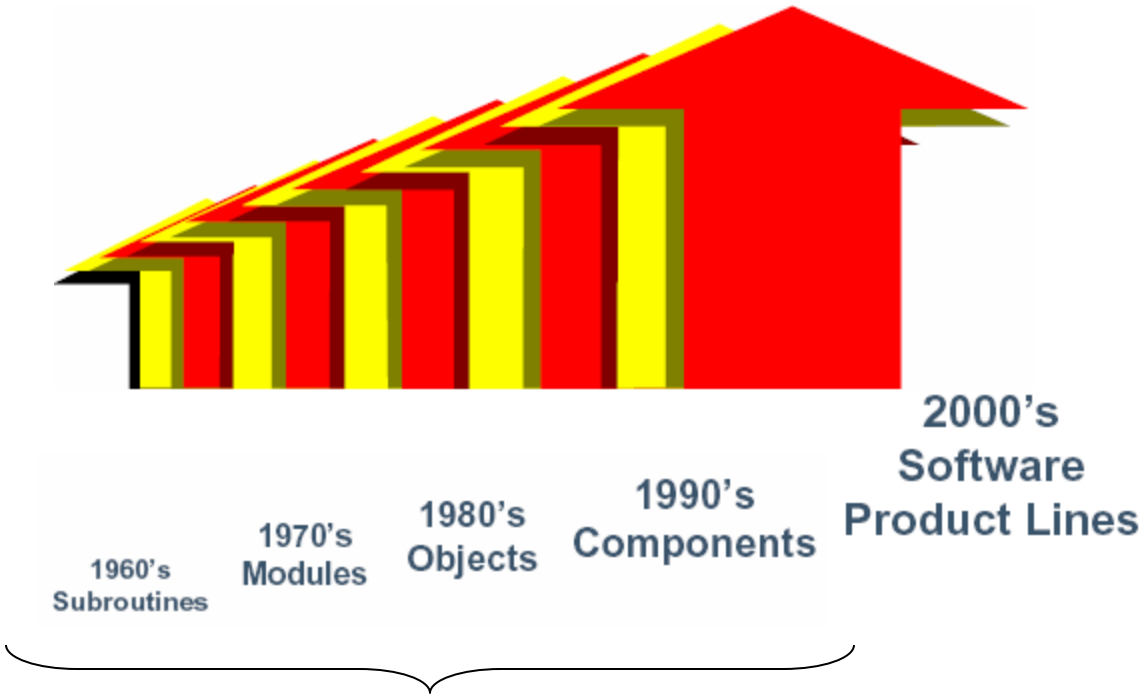


Reuso de uma base comum de tecnologia

Produtos (quase) de acordo com o desejo do usuário

Linhas de Produtos de Software

História do Reuso: do Ad-Hoc ao Sistemático



Reuso de baixa granularidade e oportunístico

Linhas de Produtos de Software

- ***Software Product Lines*** (SPL)
- Produtos → Sistemas de Software
  - Algumas funcionalidades comuns
  - Algumas funcionalidades variáveis
- Desenvolvimento de partes (*assets*) reusáveis
  - Reusados por diferentes membros da família
- Derivação de Produtos
  - Processo de construção de um produto a partir do conjunto de *assets* especificados ou implementados para uma SPL

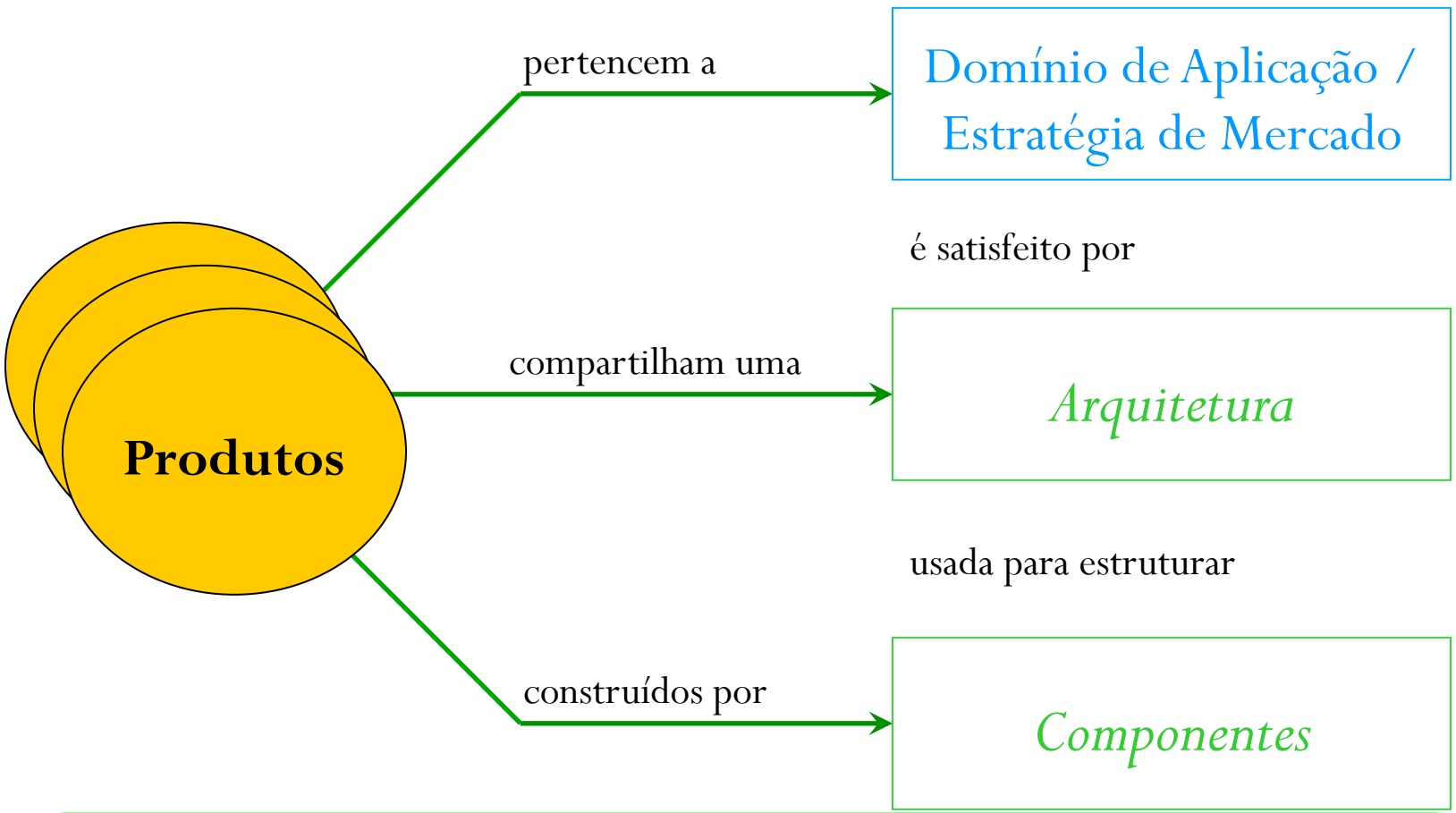
SPL – Algumas Definições

A **software product line** is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific need of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.

Software product line engineering is a paradigm to develop software applications (software-intensive systems and software products) using platforms and mass customisation.

A **software platform** is a set of software subsystems and interfaces that form a common structure from which a set of derivative products can be efficiently developed and produced.

Linhas de Produto de Software



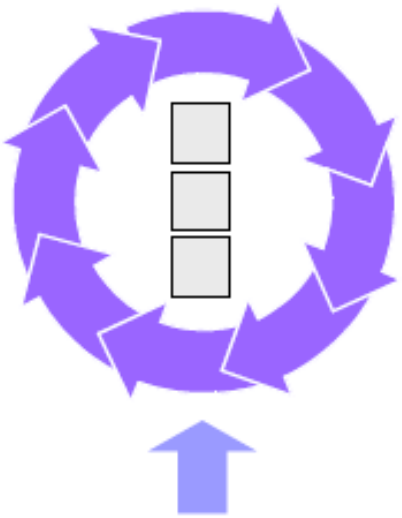
Linhas de Produtos:
Tiram vantagens econômicas sobre partes comuns (*commonality*)
Ligam *(bound)* a variabilidade

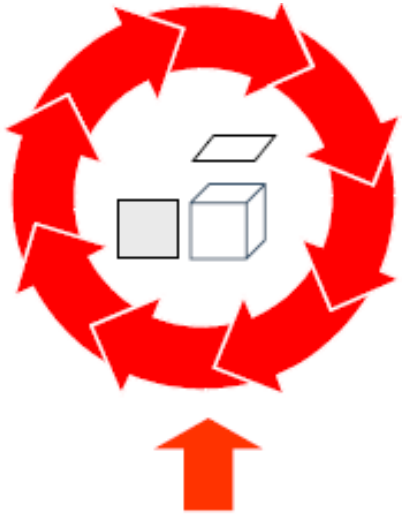
Linhas de Produto de Software

Uso de uma base comum de assets

para produção

de um conjunto de produtos relacionados





Arquitetura

Plano de Produção

Definição de Escopo

SPL - Vantagens

- Redução dos Custos de Desenvolvimento
  - Investimento *up-front*
  - *Pay-off* em torno de três sistemas
- Aumento da Qualidade
  - Qualidade melhorada através do reuso
- Redução do *time-to-market*
  - Ciclos de desenvolvimento mais curtos

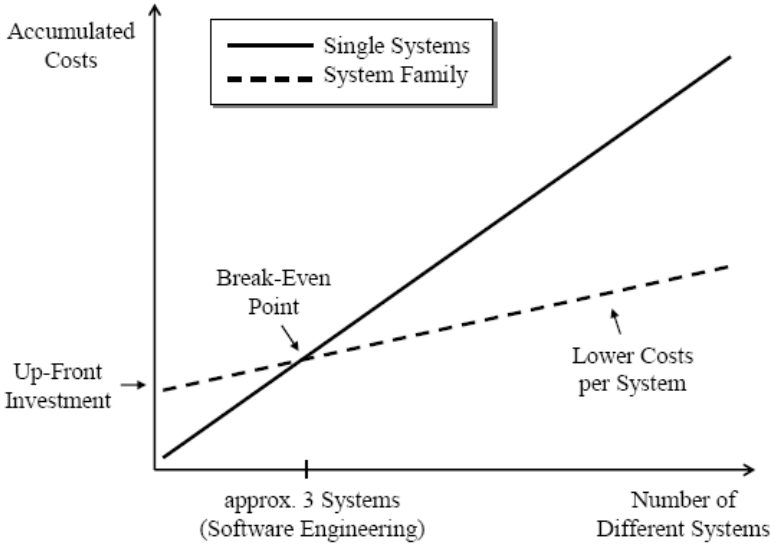
SPL - Vantagens

- Outras Vantagens
  - Redução dos Esforços de Manutenção
    - Propagação da Correção de Erros
  - Melhor estimação de custos
    - Plataforma simplifica estimação de custos
  - Benefícios para Consumidores
    - Maior qualidade, menor preço
    - *Look-and-Feel* comum

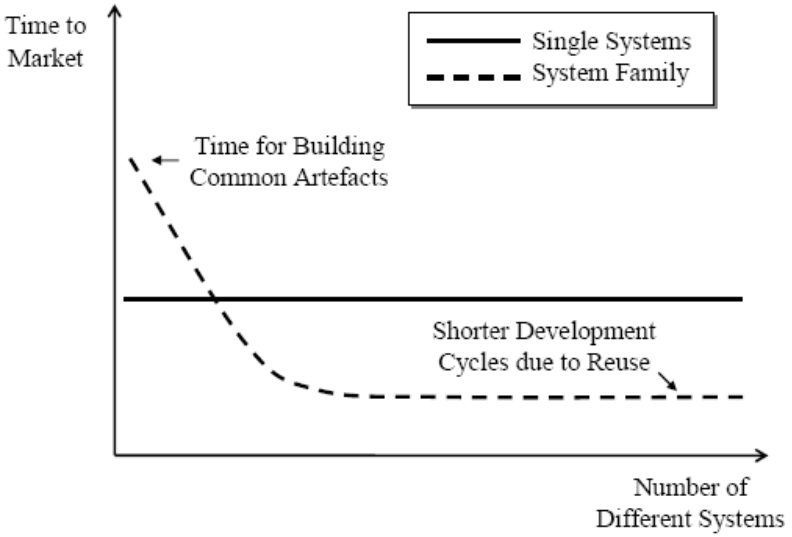
SPL - Vantagens

- Outras Vantagens
  - Copiando com Evolução
    - Evolução Organizada
  - Copiando com Complexidade
    - Windows NT 1.8 milhões de linhas de código
    - Windows XP 45 milhões de linhas de código
    - Complexidade
      - Pelo Aumento do Tamanho
      - Pela Migração de Software e Hardware
    - Plataforma reduz a complexidade

SPL - Vantagens



Menor custo de desenvolvimento



Redução do time-to-market

SPL - Riscos

- Maior Nível de Risco
  - Grande investimento inicial que pode se tornar inútil se importantes requisitos mudam
- Maior *time-to-market* para o primeiro produto baseado na arquitetura da SPL
- Requer uma Engenharia Experiente
- Gerenciamento Técnico e Organizacional

Features

- Característica do sistema **visível** pelo usuário final
- Permite expressar partes **comuns** e **variáveis** entre instâncias
- Nome conciso e descritivo
- Representam requisitos reusáveis e configuráveis
- Cada feature deve fazer diferença a alguém
- Podem ocorrem em qualquer nível
  - Requisitos de alto nível do sistema
  - Nível de arquitetura
  - Nível de subsistema e componentes
  - Nível de implementação-construção

Features

Expressar partes comuns e variáveis em termos de features é **natural** e **intuitivo**, pois clientes e engenheiros falam das características do produto em termos das features que o produto tem ou oferece.

- Podem ser
  - Obrigatórias: devem estar em todos os produtos
  - Opcionais: podem ou não estar em um produto
  - Alternativas: exatamente uma das *features* deve estar no produto
  - Or-features: um subconjunto das *features* deve estar no produto
  - Parametrizáveis: recebem algum parâmetro como entrada

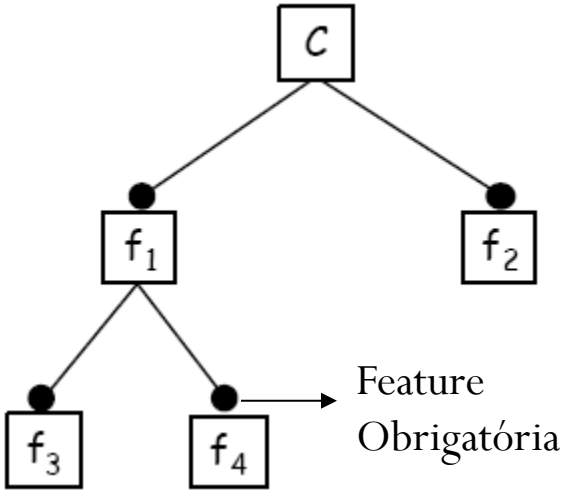
Obrigatórias ou Opcionais

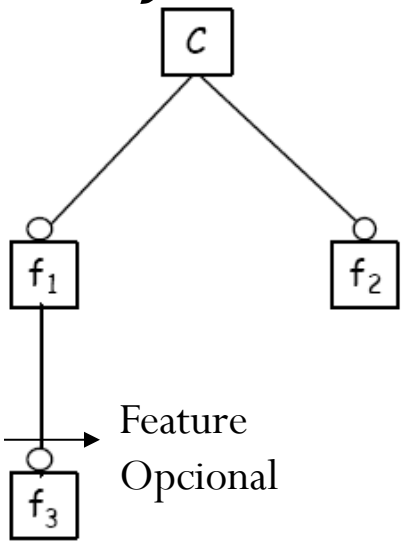
Modelo de Features

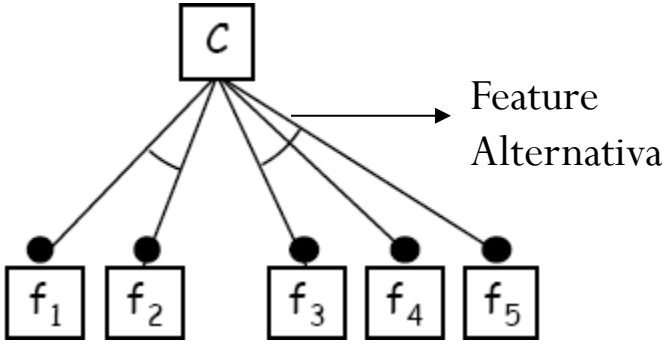
- Representa
  - *Features* comuns e variáveis aos produtos
  - Dependência entre as *features* variáveis
- Criado durante o *Feature Modeling*
  - Atividade de modelar propriedades comuns e variáveis dos produtos e suas interdependências, e organizando-as em um modelo coerente referido como um modelo de *features*
- Cada *feature* pode ter alguma informação adicional
  - Restrições (*Constraints*)
  - Regras de Dependências

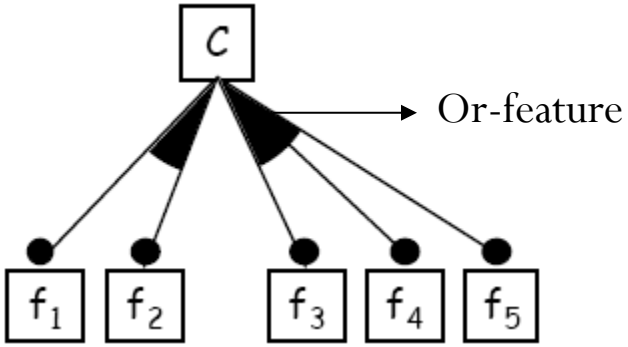
Modelo de Features

- Notação ***Feature Oriented Domain Analysis***





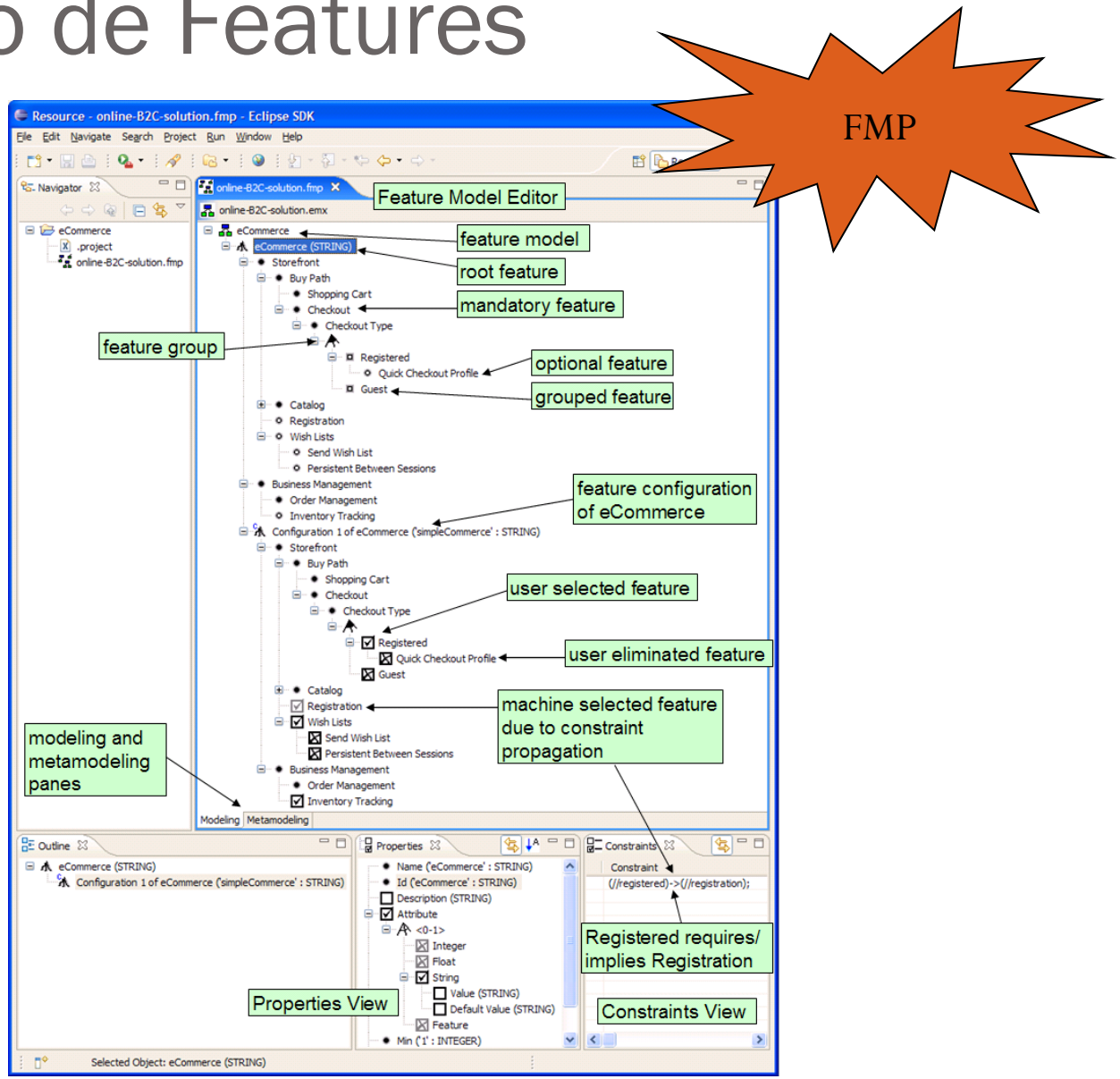




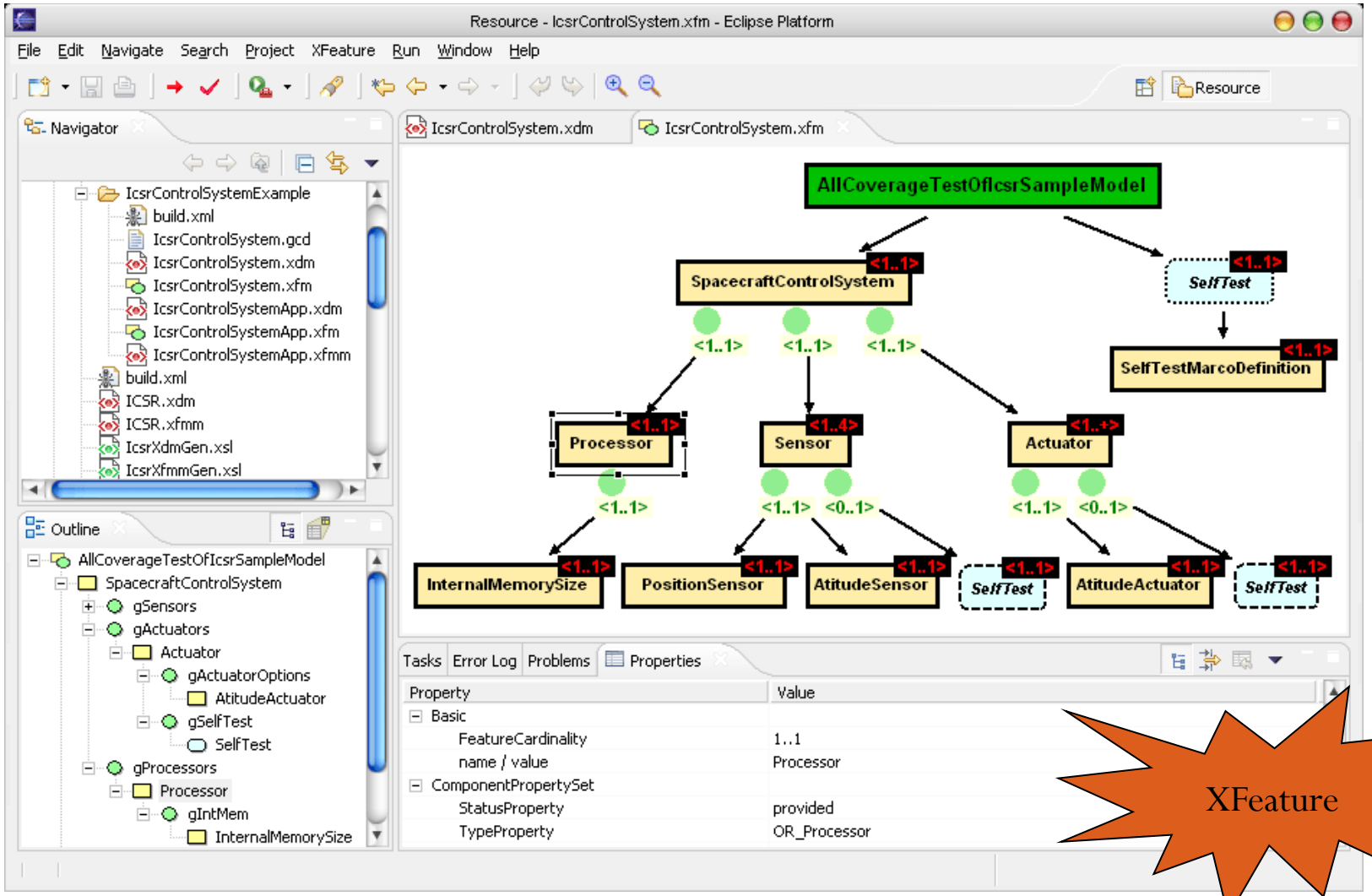
Modelo de Features

- Ferramentas
  - ***Feature Modeling Plug-in (fmp)***
    - Eclipse plug-in
    - Permite edição e configuração de modelos de feature
    - Download
      - http://gsd.uwaterloo.ca/projects/fmp-plugin/
  - ***XFeature***
    - Eclipse Plugin
    - Suporta a modelagem de famílias de produto e de aplicações instanciadas a partir delas
    - Permite que os usuários definam seu próprio meta-modelo
    - Download
      - http://www.pnp-software.com/XFeature/

Modelo de Features



Modelo de Features

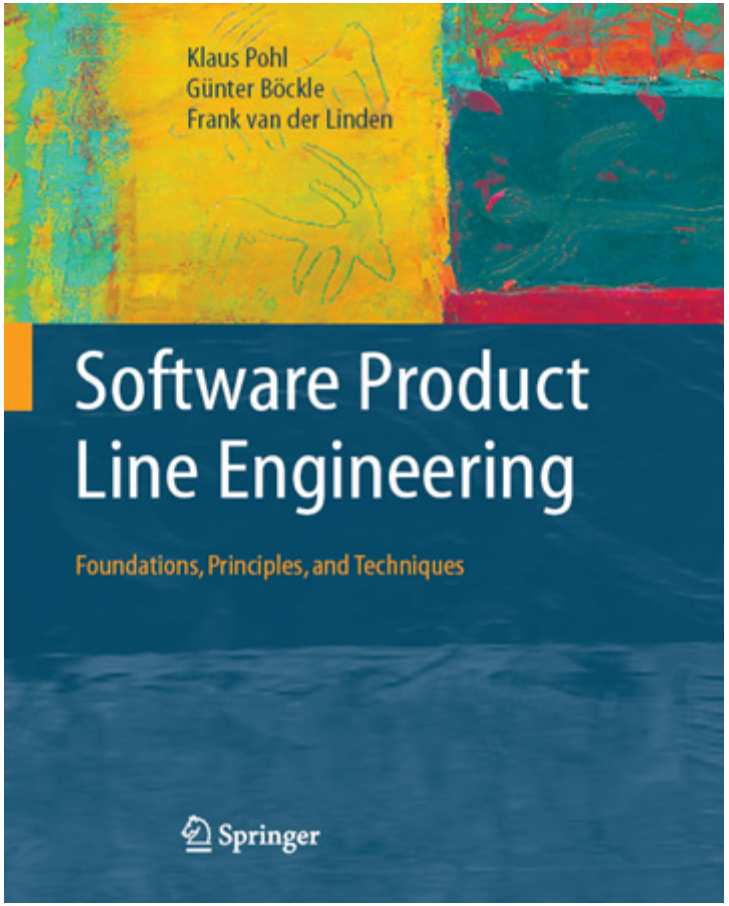


Princípios de Engenharia

- Separação de interesses
  - *Features* devem ser modularizadas do início ao fim do desenvolvimento da SPL
- Ocultamento de Informação
  - Característica que já vem da OO
  - Ocultar informação permite trocarmos classes, componentes, etc. com mínimo impacto
- Parametrização dos artefatos usando as *features*

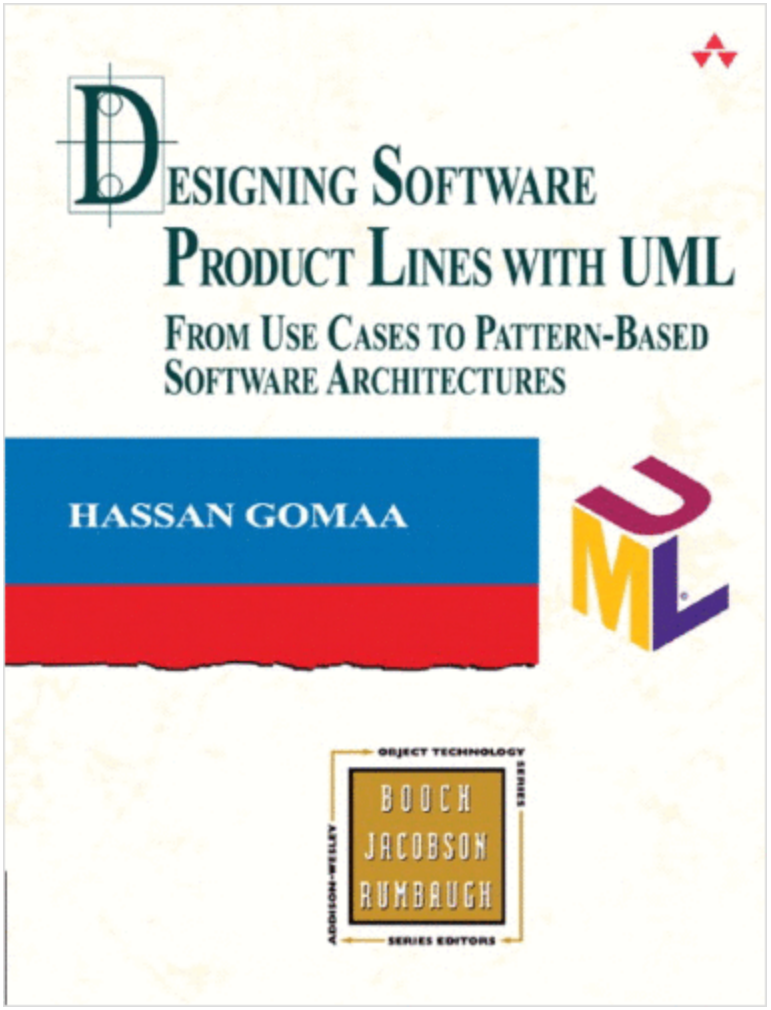
Referências

Referências



- ***Software Product Line Engineering – Foundations, Principles, and Techniques***
- Pohl, Böckle e van der Linden
- *Framework* para o desenvolvimento de SPLs, dividido em dois processos-chave
  - *Domain Engineering*
  - *Application Engineering*

Referências



- ***Designing Software Product Lines with UML – From Use Cases to Pattern-Based Software Architectures***
- Gomaa
- Apresenta o método PLUS
  - Extensão de métodos baseados em UML para o desenvolvimento de sistemas únicos para endereçar SPL

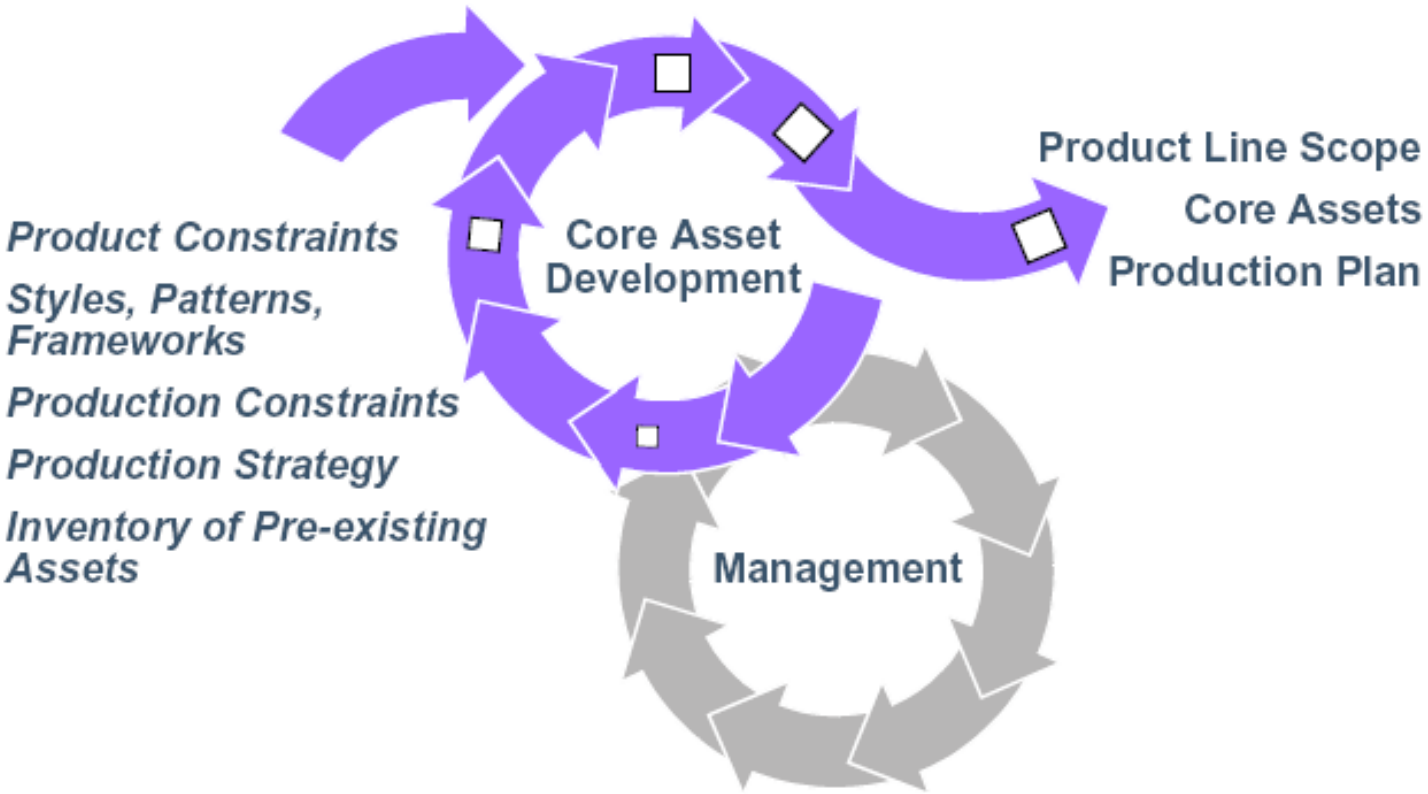
Desenvolvimento de SPL

As três atividades essenciais para SPL

Desenvolvimento de SPL

- Todas as três atividades
  - Estão inter-relacionadas
  - São altamente interativas
- Não existe "primeira" atividade
  - Em alguns contextos
    - Produtos existentes são a base para os core assets
  - Em outros
    - *Core assets* podem ser desenvolvidos e procurados para futuro reuso
- Forte *feedback* loop entre os *core assets* e os produtos
- Necessidade de forte gerenciamento entre múltiplos níveis
- Gerenciamento orquestra os processos para fazer as três atividades essenciais trabalharem juntas

Desenvolvimento de SPL



Desenvolvimento de SPL

***Core Assets* incluem:**

Requisitos e análise de requisitos

Modelo de domínio

Arquitetura de software

Engenharia de performance

Documentação

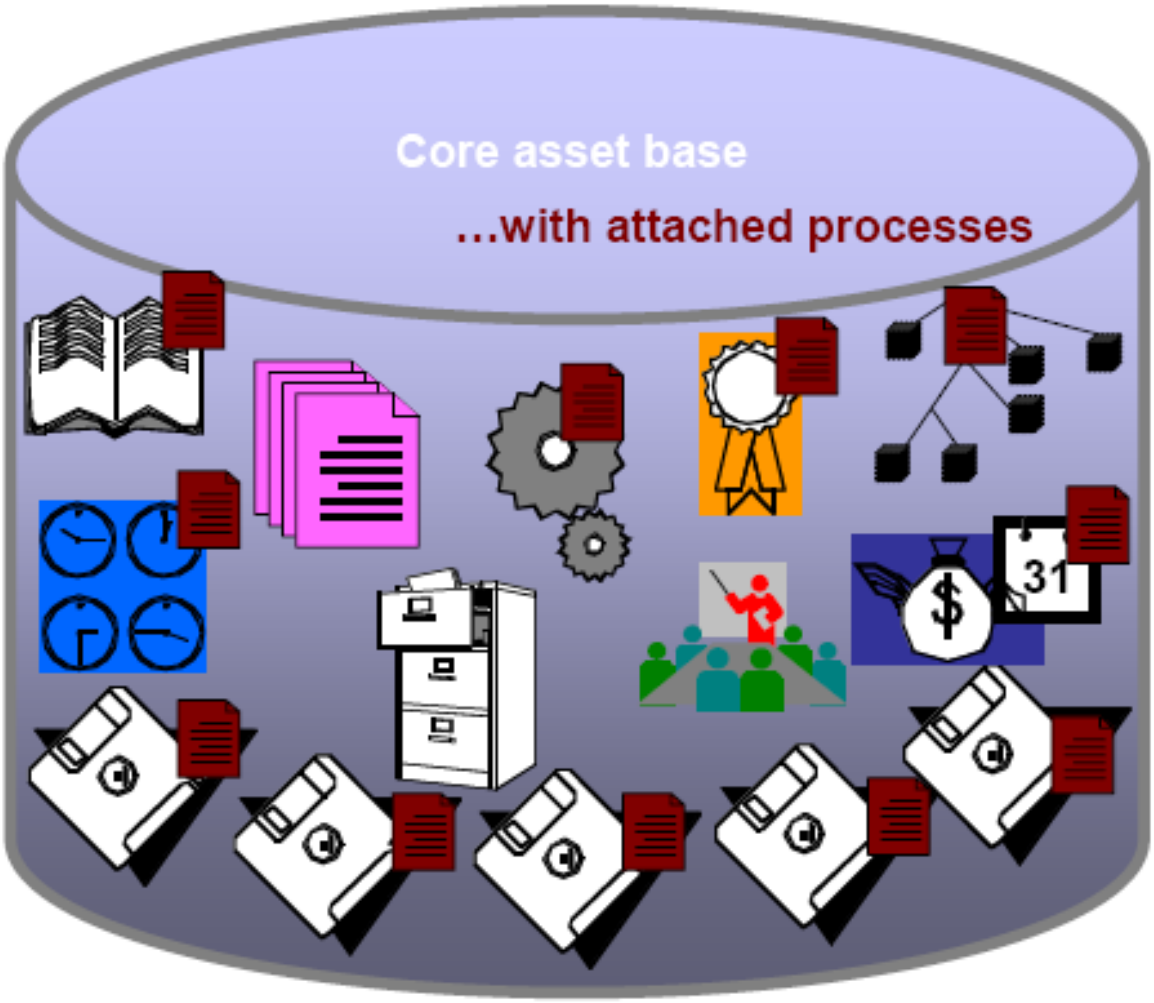
Planos de teste, casos de teste e dados

Conhecimento humano e habilidades

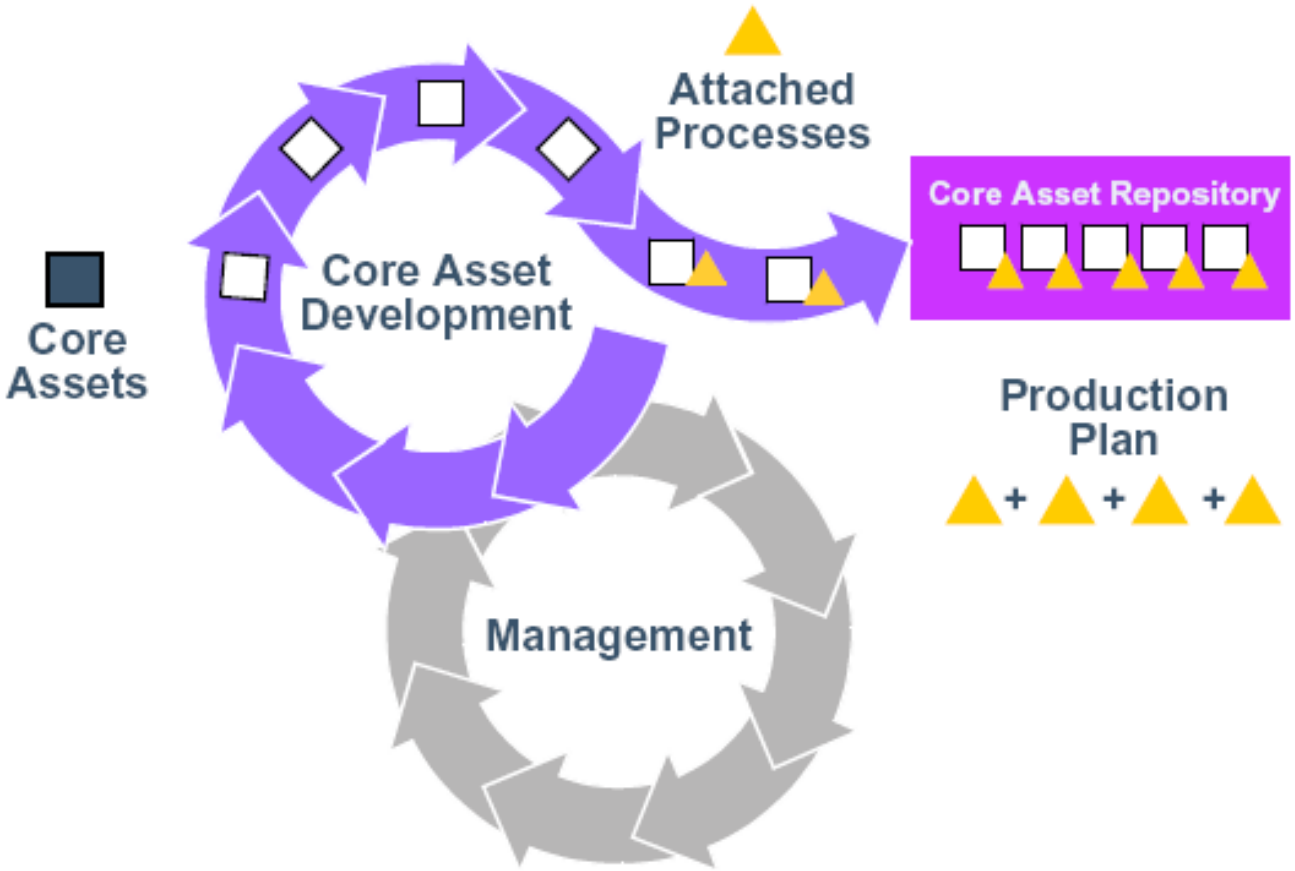
Processos, métodos e ferramentas

Despesas, cronogramas, planos de trabalho

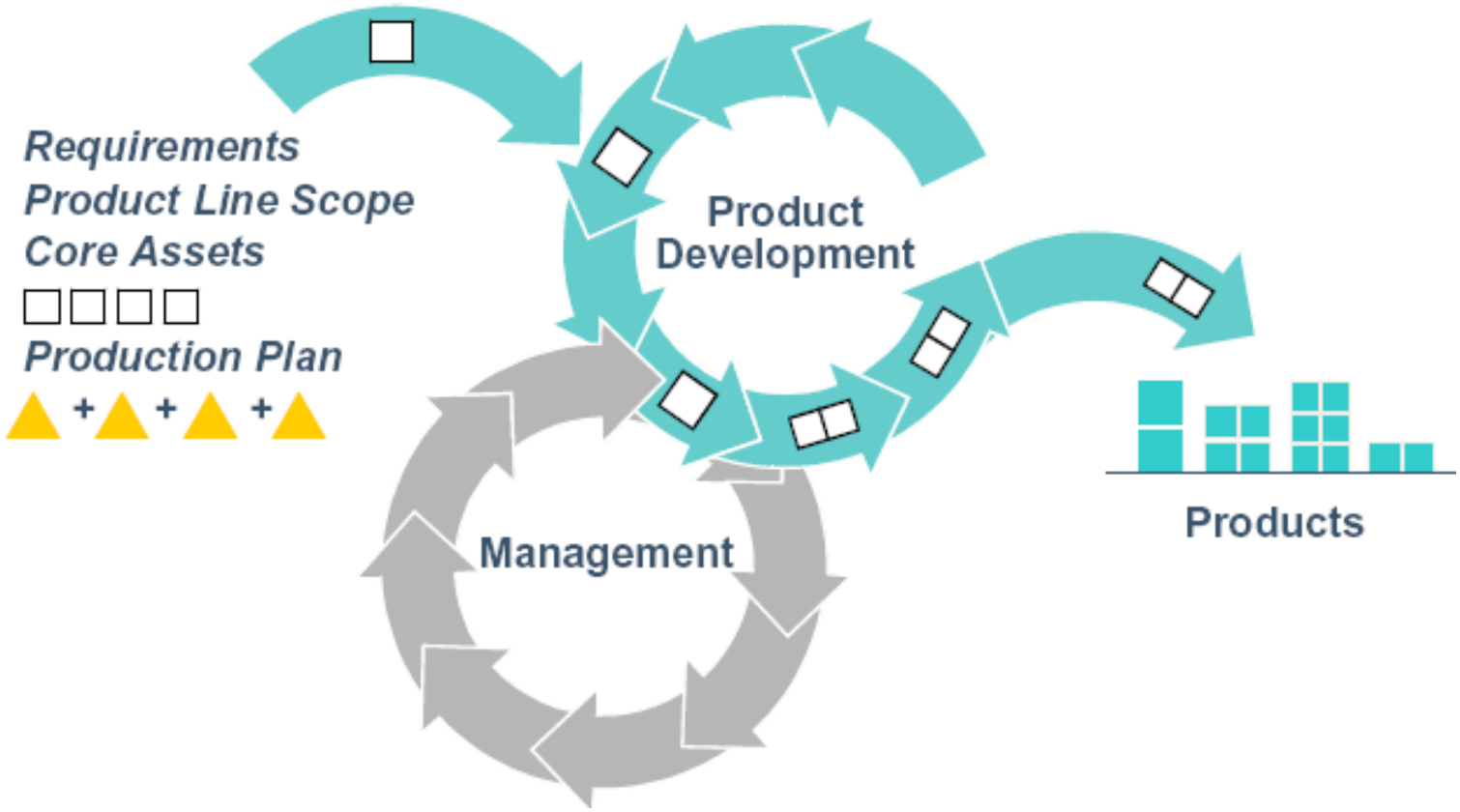
... e Software



Desenvolvimento de SPL



Desenvolvimento de SPL



Desenvolvimento de SPL

- Gerenciamento em múltiplos níveis tem um papel importante no sucesso da linha de produto
- Responsabilidades
  - Atingir a estrutura organizacional certa
  - Alocar recursos
  - Coordenar e supervisionar
  - Oferecer treinamento
  - Recompensar empregados apropriadamente
  - Desenvolver e comunicar uma estratégia de aquisição
  - Gerenciar interfaces externas
  - Criar e implementar um plano de adoção da linha de produto

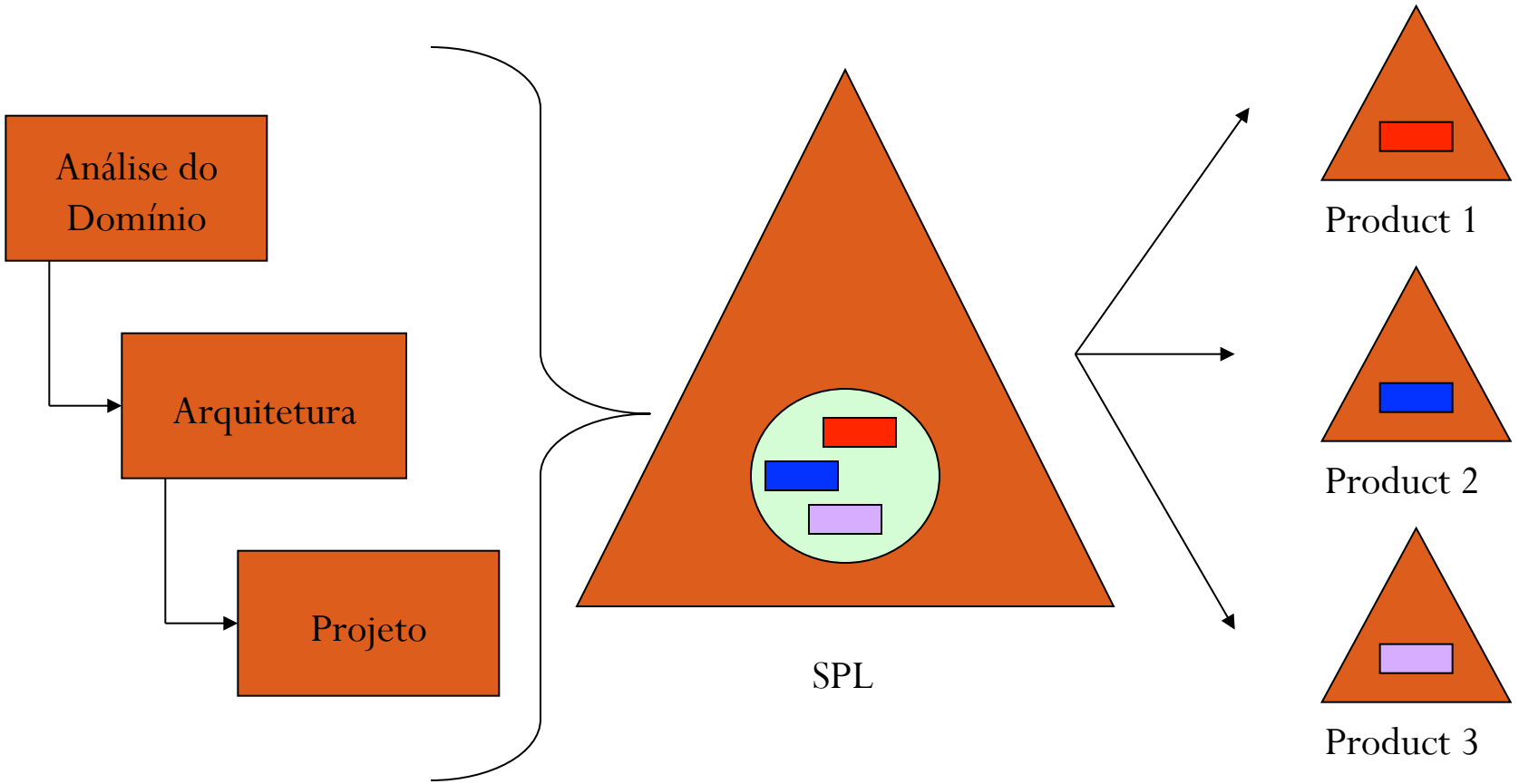
Desenvolvimento de SPL

- Terminologia Alternativa

Product Line	→	Product Family
Core Assets	→	Plataform
Business Unit	→	Product Line
Product	→	Customization
Core Asset Development	→	Domain Engineering
Product Development	→	Application Engineering

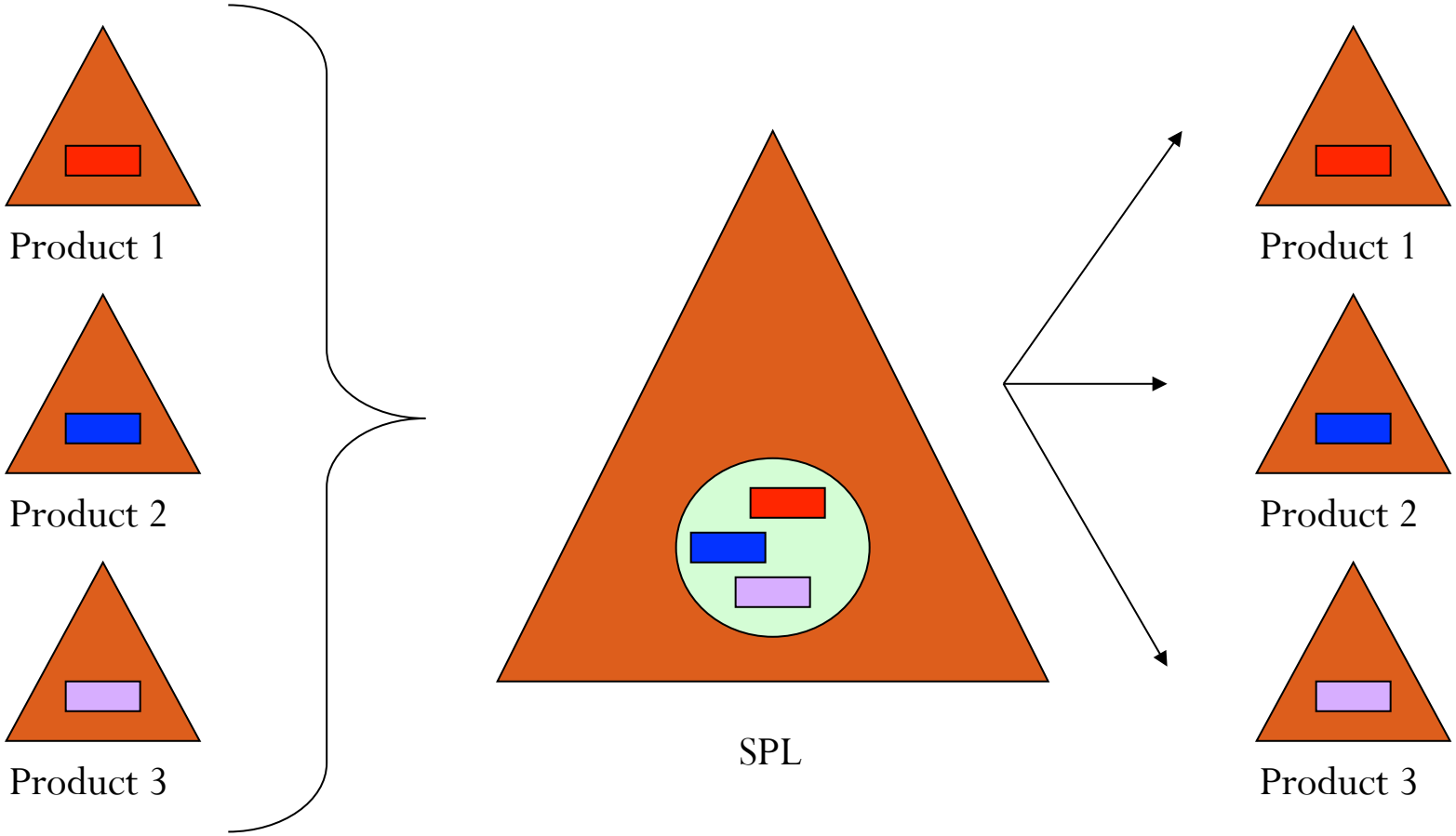
Abordagens para Construção de SPL

- Pró-ativa
  - Desenvolvimento de linhas de produto considerando todos os produtos a serem gerados previamente
  - Um conjuntos completo de artefatos é desenvolvido para a SPL

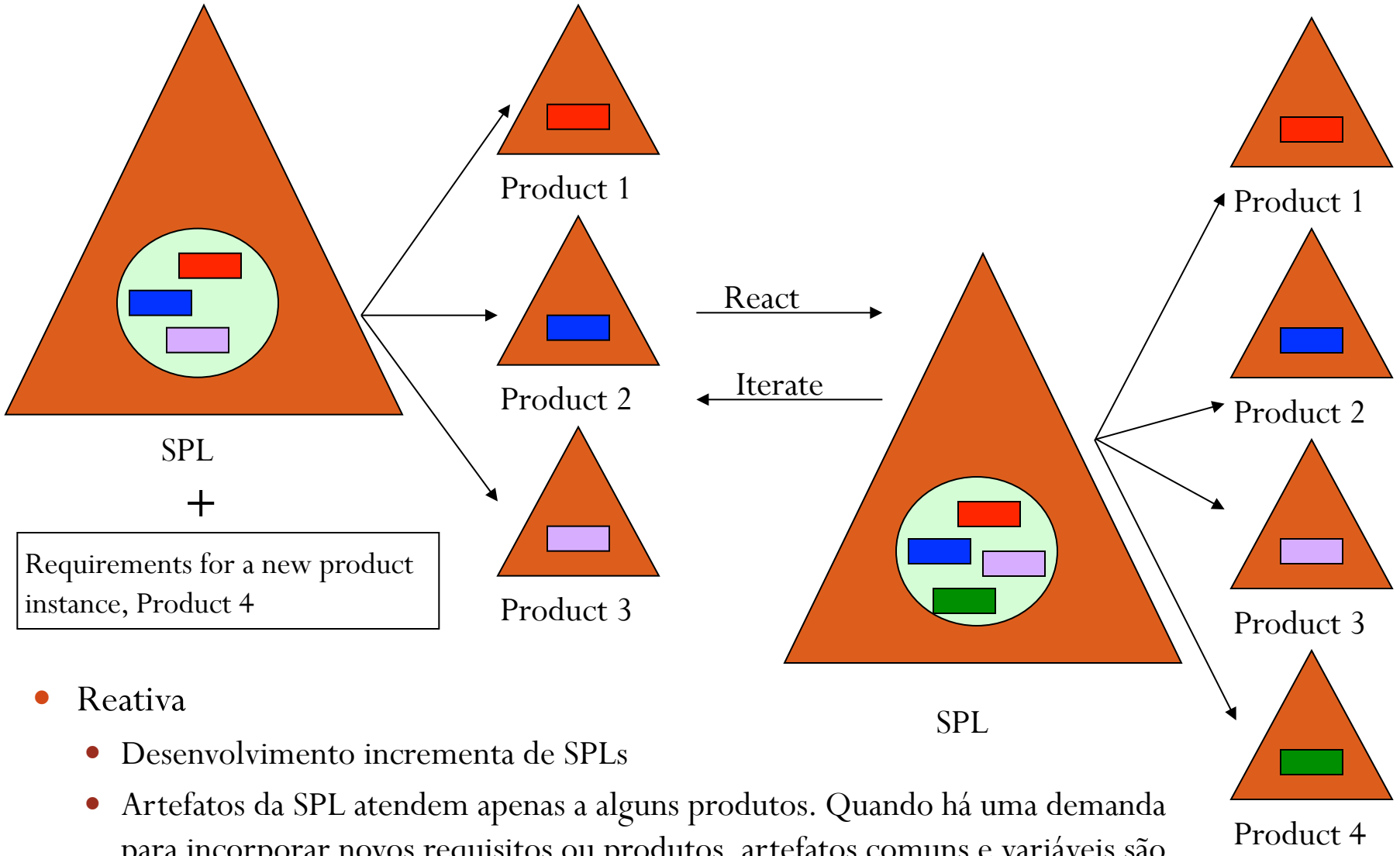


Abordagens para Construção de SPL

- Extrativa
  - A SPL é desenvolvida a partir de sistemas já existentes
  - *Features* variáveis e comuns são extraídas desses sistemas para derivar uma versão inicial da SPL



Abordagens para Construção de SPL



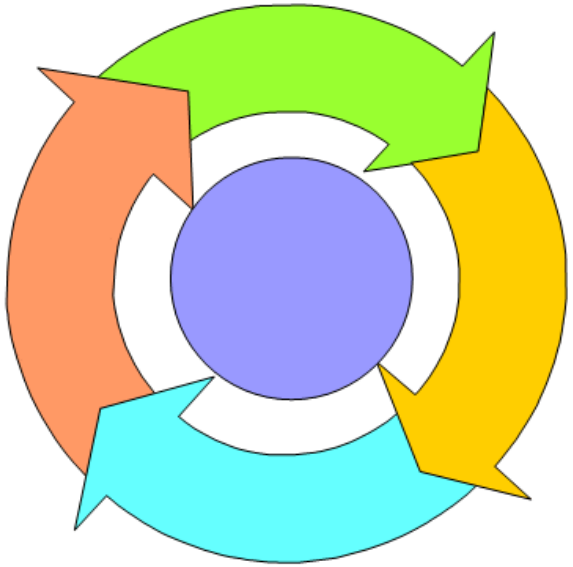
- Reativa
  - Desenvolvimento incrementa de SPLs
  - Artefatos da SPL atendem apenas a alguns produtos. Quando há uma demanda para incorporar novos requisitos ou produtos, artefatos comuns e variáveis são incrementalmente estendidos em reação a eles.

Implemtentação de SPL

- Orientação a objetos e polimorfismo
- Padrões de projeto
- Frameworks
- Programação orientada a features
- Variabilidade em tempo de *Deployment* e de Execução
- Transformação de Programas
- Compilação Condicional
- Programação orientada a Aspectos

Metodologias de SPL

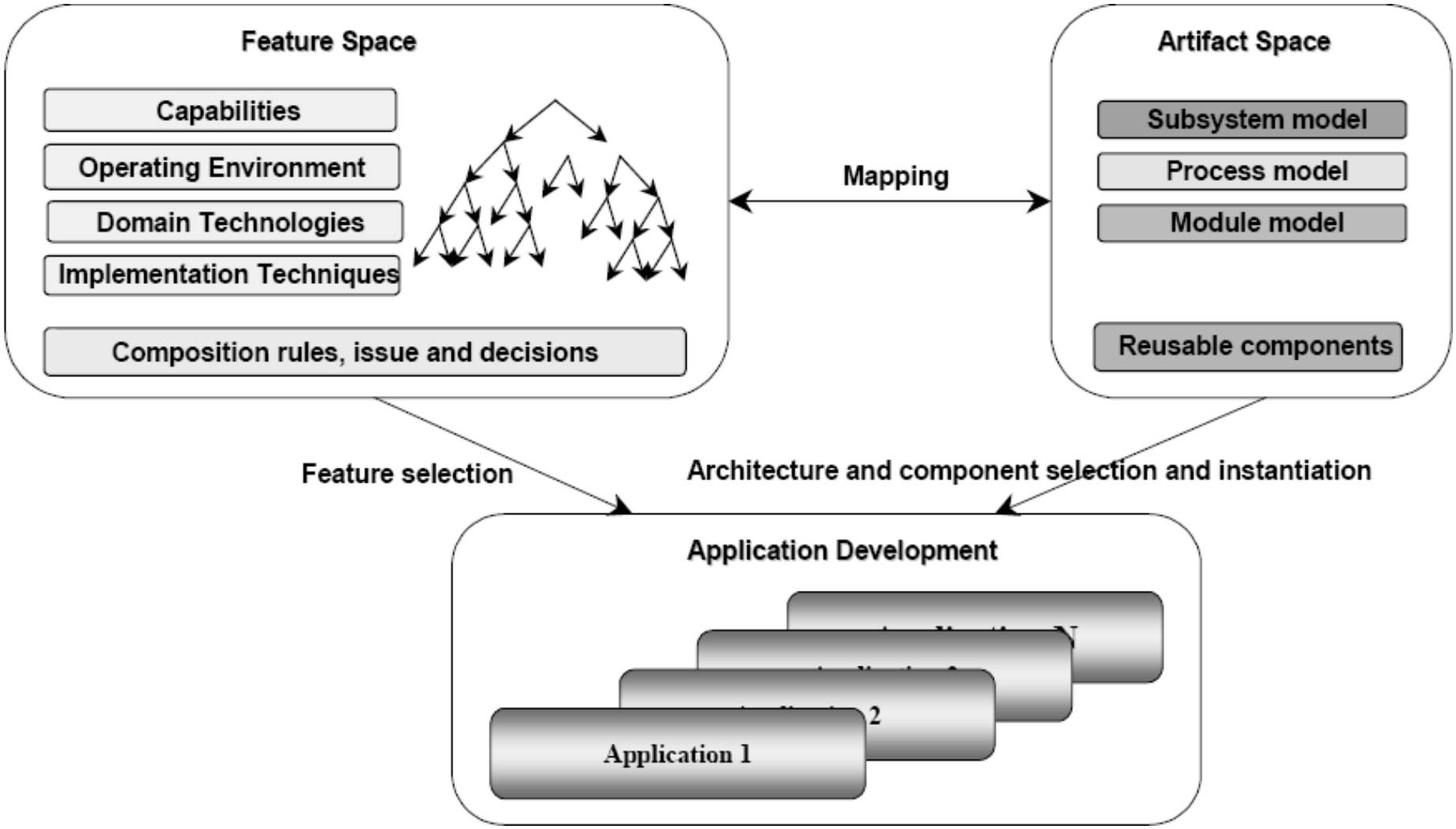
- Diversas Metodologias de SPL foram propostas
  - COPA
  - FORM
  - PuLSE
  - KobrA
  - FAST
  - PLUS
  - Framework proposto no livro de Pohl



FORM

- ***Feature Oriented Reuse Method***
- Extensão do FODA
  - *Feature Oriented Domain Analysis*
- Método Sistemático
  - Procura e captura partes comuns e diferenças em um domínio em termos de ***features***
  - Usa a análise dos resultados para desenvolver
    - Arquiteturas de Domínio
    - Componentes

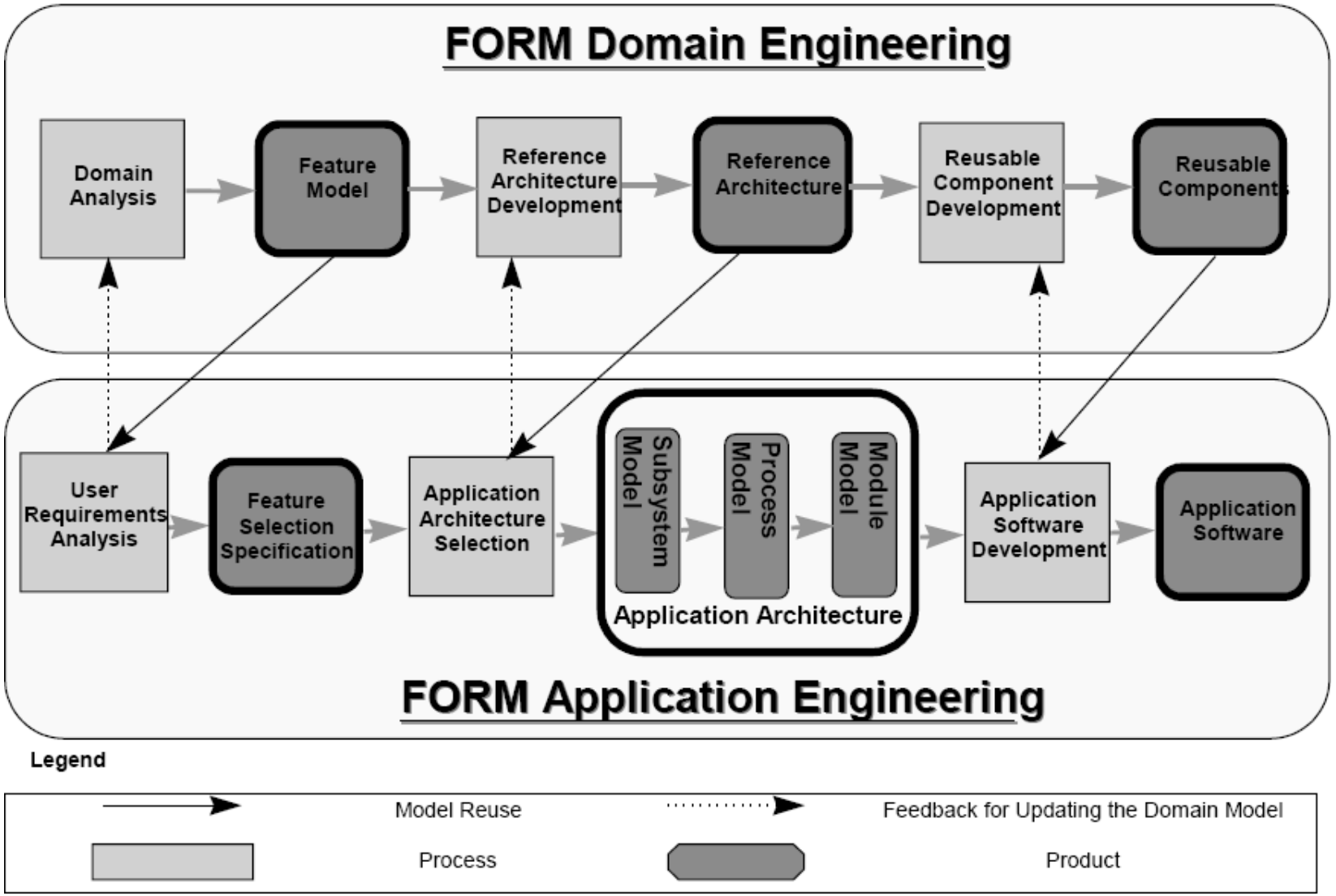
FORM


Feature Space
Capabilities
Operating Environment
Domain Technologies
Implementation Techniques
Composition rules, issue and decisions
Mapping
Artifact Space
Subsystem model
Process model
Module model
Reusable components
Feature selection
Architecture and component selection and instantiation
Application Development
Application 1

FORM

- Dois processos de engenharia
  - ***Domain Engineering***
    - Atividades para analisar os sistemas de um domínio
    - Criação de arquiteturas de referência e de componentes reusáveis como resultado da análise
  - ***Application Engineering***
    - Atividades para o desenvolvimento de aplicações usando artefatos criados na *Domain Engineering*
    - Trivial comparada com o desenvolvimento tradicional de aplicações

FORM


FORM Domain Engineering
Domain Analysis
Feature Model
Reference Architecture Development
Reference Architecture
Reusable Component Development
Reusable Components
User Requirements Analysis
Feature Selection Specification
Application Architecture Selection
Subsystem Model
Process Model
Module Model
Application Architecture
Application Software Development
Application Software
FORM Application Engineering
Legend
Model Reuse
Feedback for Updating the Domain Model
Process
Product

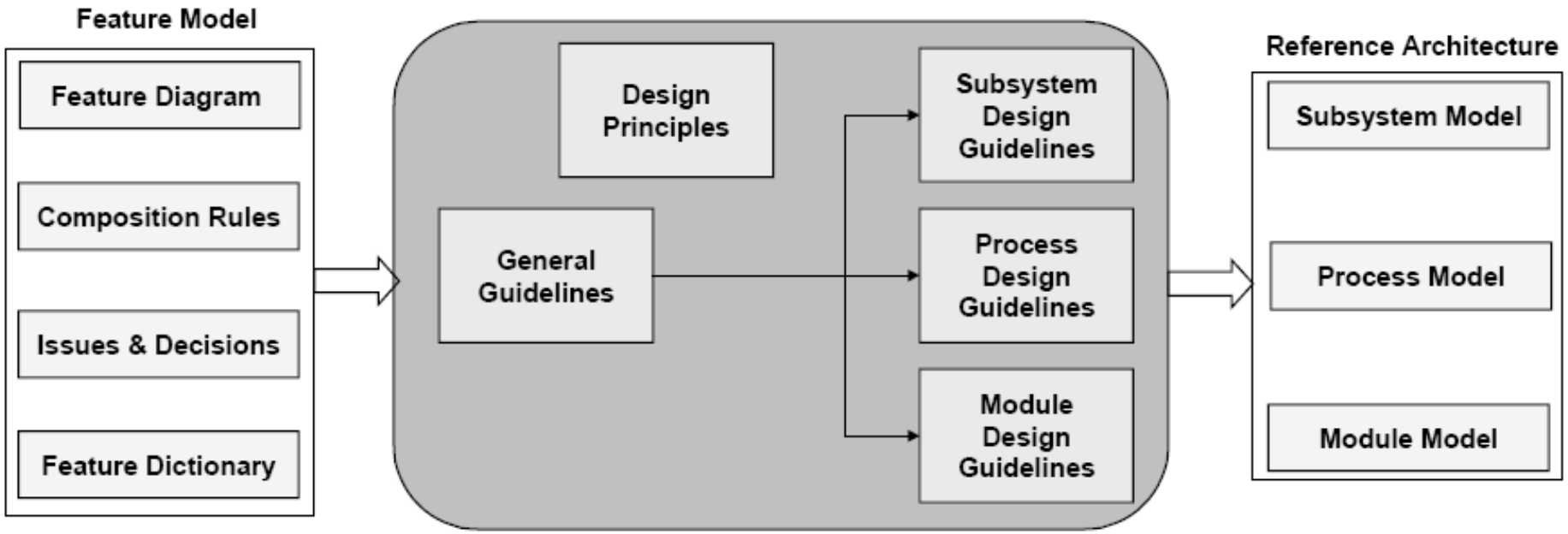
FORM: Domain Engineering

- Objetivo
  - Desenvolver artefatos do domínio
    - Podem ser usados no desenvolvimento de aplicações de um dado domínio
  - Estabelecer mapeamento entre o espaço de decisão e os espaço de artefatos
    - ***Feature Model*** → ***Architecture Model***
- Três Fases
  - *Context Analysis*
  - *Domain (ou features) Modeling*
  - *Architecture (e component) Modeling*

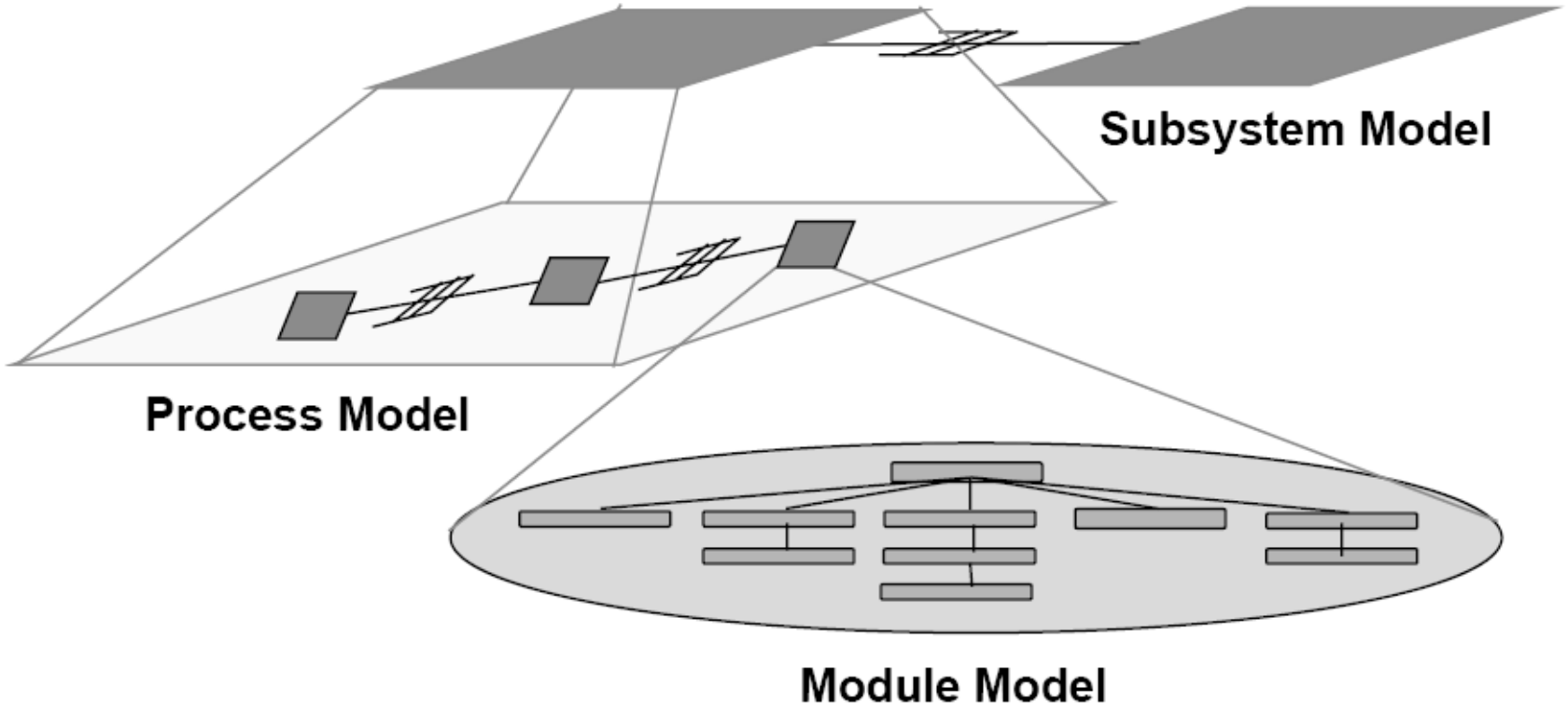
FORM: Domain Engineering

- ***Domain Analysis e Features Modeling***
  - Identificar partes comuns e diferenças nos sistemas de um domínio e representá-los de uma forma explorável
  - Três processos
    - *Planning*
    - *Feature Analysis*
    - *Validation Activities*
- ***Architecture Modeling and Component Development***
  - Arquitetura do domínio
    - Modelo para a criação de arquiteturas de diferentes sistemas
  - Definida em termos de um conjunto de modelos
    - *Subsystem Model*: arquitetura geral do sistema
    - *Process Model*: comportamento dinâmico do sistema
    - *Module Model*: modelo com componentes reusáveis

FORM: Domain Engineering



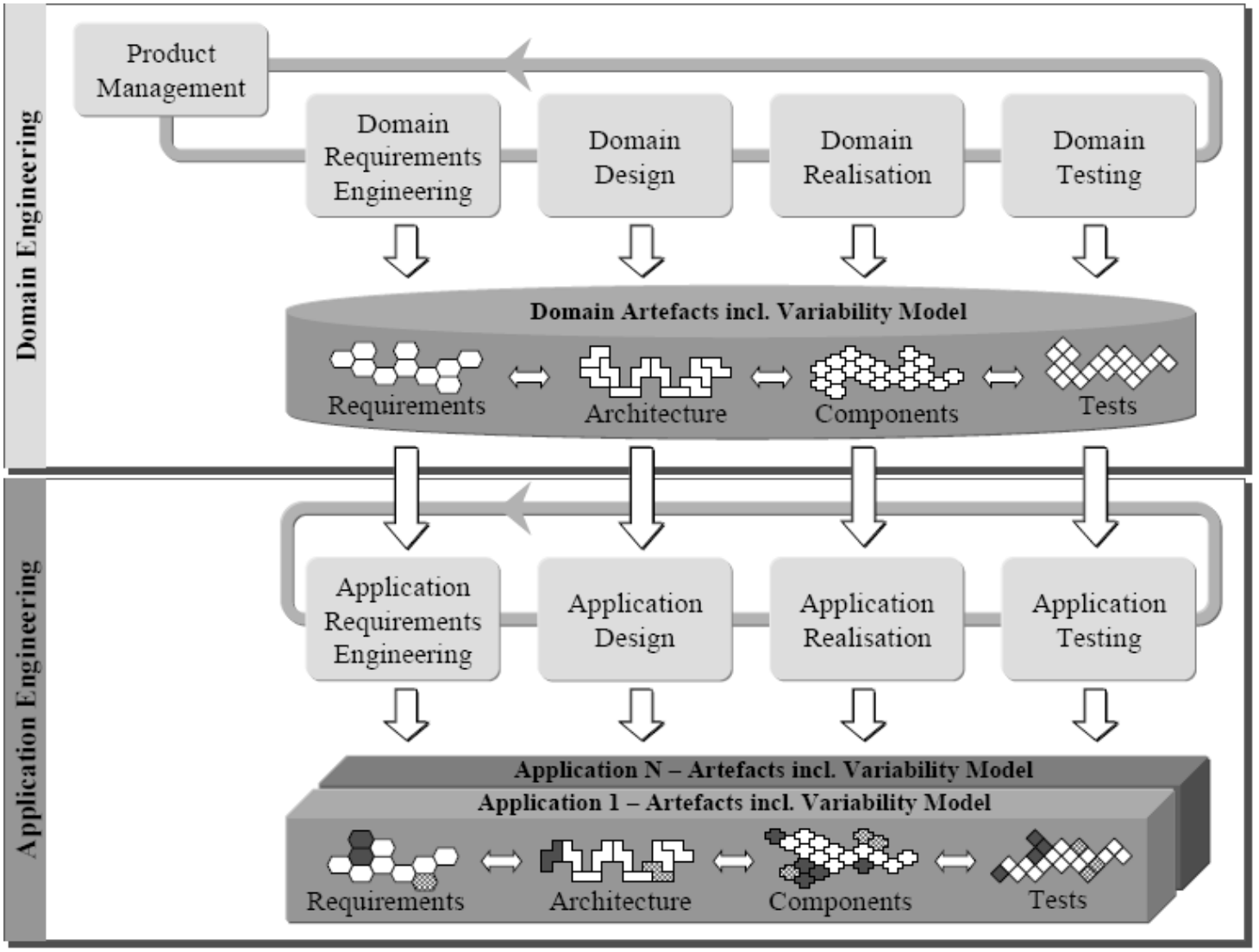
FORM: Domain Engineering



FORM: Application Engineering

- Processo de desenvolvimento de uma aplicação específica
- Faz uso do conhecimento obtido durante Domain Engineering
- Passos
  - Requirement Analysis e Feature Selection
    - Analisa requisitos do usuário
    - Seleciona as features apropriadas e válidas do domínio de um modelo de features
  - Architecture Model Selection e Application Development
    - Identifica modelo de referência correspondente
    - Completa o desenvolvimento da aplicação
      - Reuso de Componentes de um modo bottom-up

Klaus Pohl's Framework



Klaus Pohl's Framework

- *Domain Engineering*
  - Processo da engenharia de SPL no qual partes comuns e variáveis da linha de produto são definidas e realizadas
- *Application Engineering*
  - Processo da engenharia de SPL no qual as aplicações da linha de produto são construídas reusando artefatos do domínio e explorando a variabilidade da linha de produto

Klaus Pohl's Framework

- *Domain Engineering*
  - Define partes comuns e variáveis da SPL
  - Define para qual conjunto de aplicações que a SPL é planejada
    - Define o escopo da SPL
  - Define e constrói artefatos reusáveis que realizam a variabilidade desejada

Klaus Pohl's Framework

- Product Management
  - Lida com aspectos econômicos da SPL
  - Escopo da SPL
- Domain Requirements Engineering
  - Atividades para elicitar e documentar requisitos comuns e variáveis
- Domain Design
  - Atividades para definir a arquitetura de referência
- Domain Realization
  - Lida com o design detalhado
  - Implementação de componentes de software reusáveis
- Domain Testing
  - Responsável por validar e verificar os componentes reusáveis
  - Desenvolve artefatos de teste reusáveis

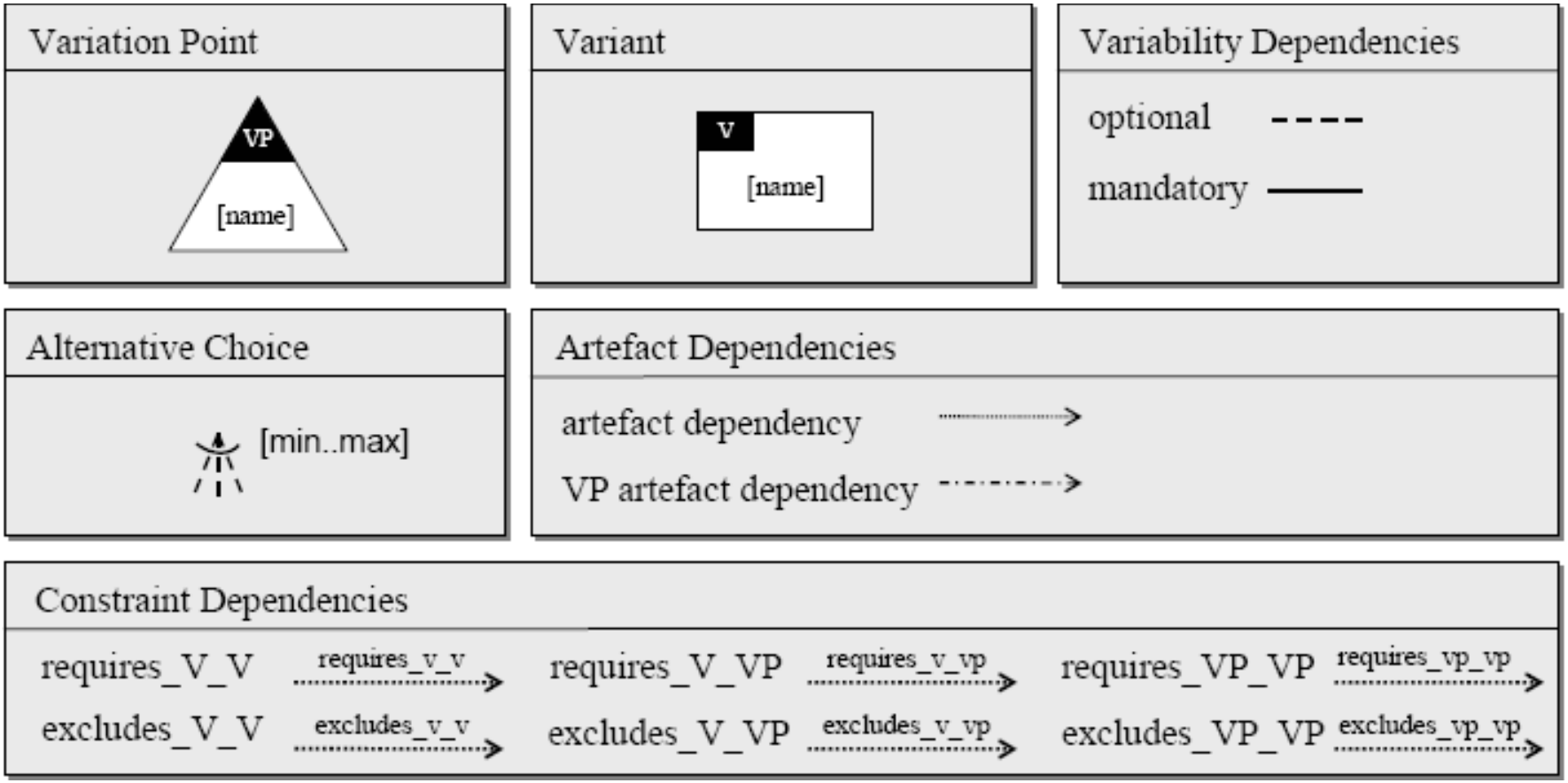
Klaus Pohl's Framework

- *Application Engineering*
  - Atinge o máximo de reuso possível dos assets do domínio na definição de desenvolvimento da aplicação da linha de produto
  - Explora as partes comuns e variaveis da SPL no desenvolvimento da aplicação da linha de produto
  - Documenta os artefatos da aplicação
  - Liga a variabilidade de acordo com as necessidades da aplicação a partir dos requisitos sobre arquitetura, componentes e casos de teste
  - Estima os impactos das diferenças entre os requisitos da aplicação e do domínio sobre arquitetura, componentes e testes

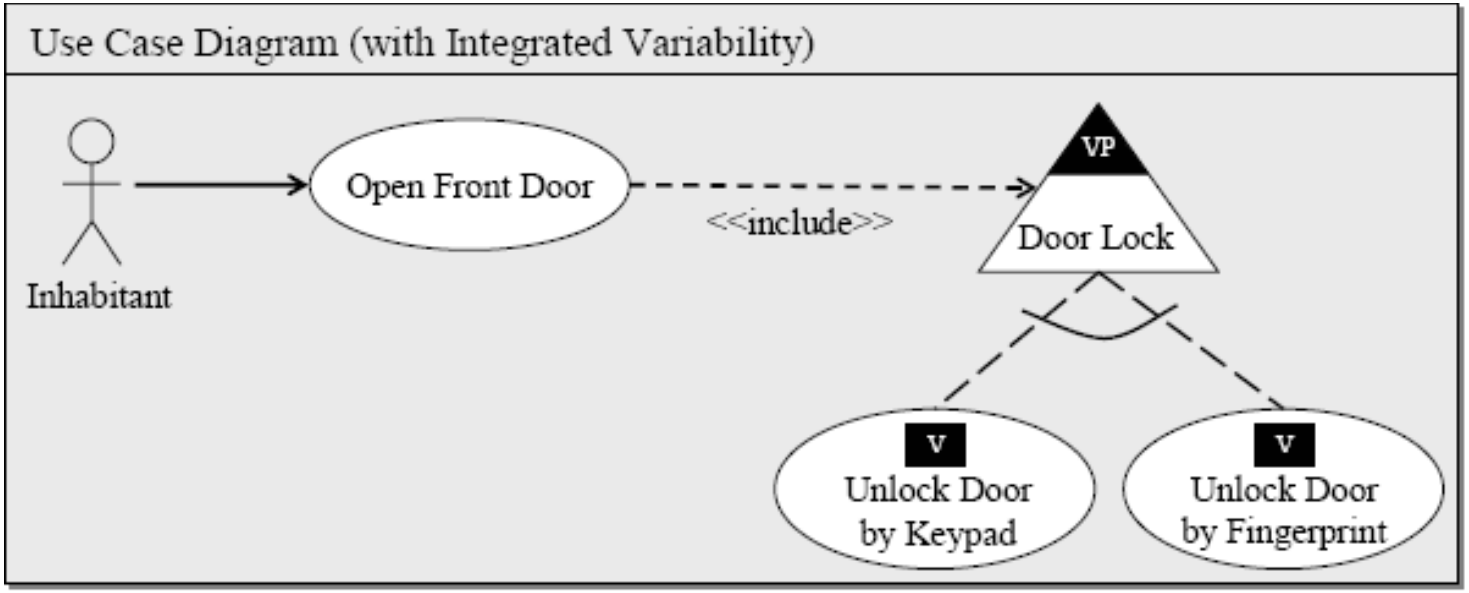
Klaus Pohl's Framework

- *Application Requirements Engineering*
  - Atividades para o desenvolvimento das especificações dos requisitos da aplicação
- *Application Design*
  - Atividades para a produção da arquitetura da aplicação
  - Usa a arquitetura de referência para instanciar a arquitetura da aplicação
  - Seleciona e configura as partes requeridas da arquitetura de referência
    - Incorpora adaptações específicas da aplicação
  - Variability bound
- *Application Realization*
  - Cria a aplicação considerada
  - Seleciona e configura componentes de software reutilizáveis
  - Realiza assets específicos da aplicação
- *Application Testing*
  - Atividades necessárias para validar e verificar a aplicação de acordo com sua especificação

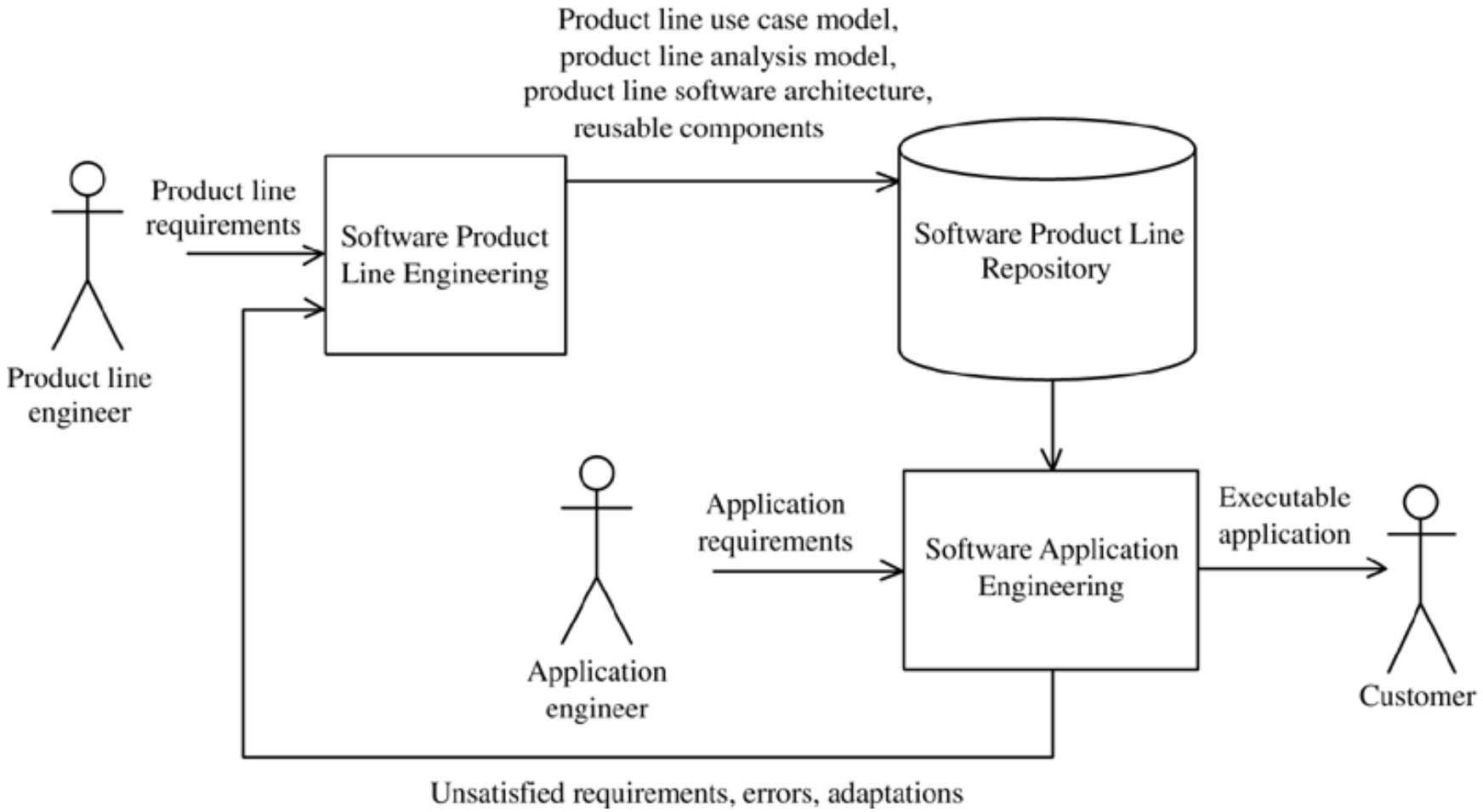
Klaus Pohl's Framework



Klaus Pohl's Framework



PLUS


Product line use case model,
product line analysis model,
product line software architecture,
reusable components
Product line
requirements
Software Product
Line Engineering
Software Product Line
Repository
Product line
engineer
Application
requirements
Software Application
Engineering
Executable
application
Application
engineer
Customer
Unsatisfied requirements, errors, adaptations

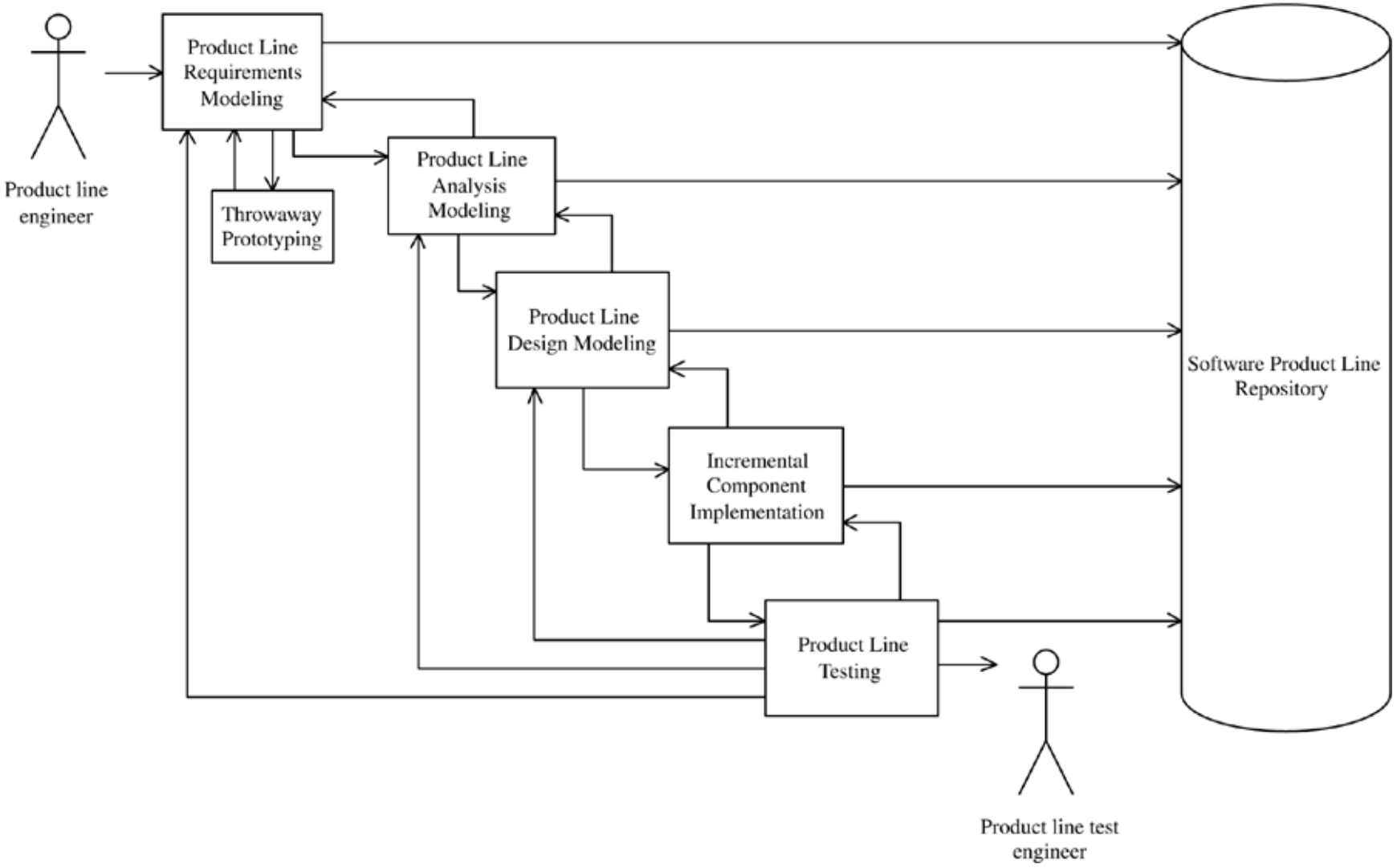
PLUS

- *Software product line engineering*
  - Partes comuns e variáveis são analisadas de acordo com os requisitos gerais da SPL
  - Desenvolvimento de
    - Product line use case model
    - Product line analysis model
    - Arquitetura da SPL
    - Componentes Reusáveis
  - Teste de
    - Componentes
    - Configurações da Aplicação
  - Artefatos são armazenados em um repositório da SPL

PLUS

- *Software application engineering*
  - Desenvolvimento da aplicação individual, membro da SPL
  - Desenvolvedores da aplicação fazem total uso de todos artefatos desenvolvidos durante o ciclo de SPL Engineering
  - Dados os requisitos gerais da aplicação individual
    - product line use case model deriva application use case model
    - product line analysis model deriva application analysis model
    - software product line architecture deriva architecture of the software application
  - Dada a arquitetura da aplicação e os componentes do repositório da SPL
    - Deploy da aplicação executável

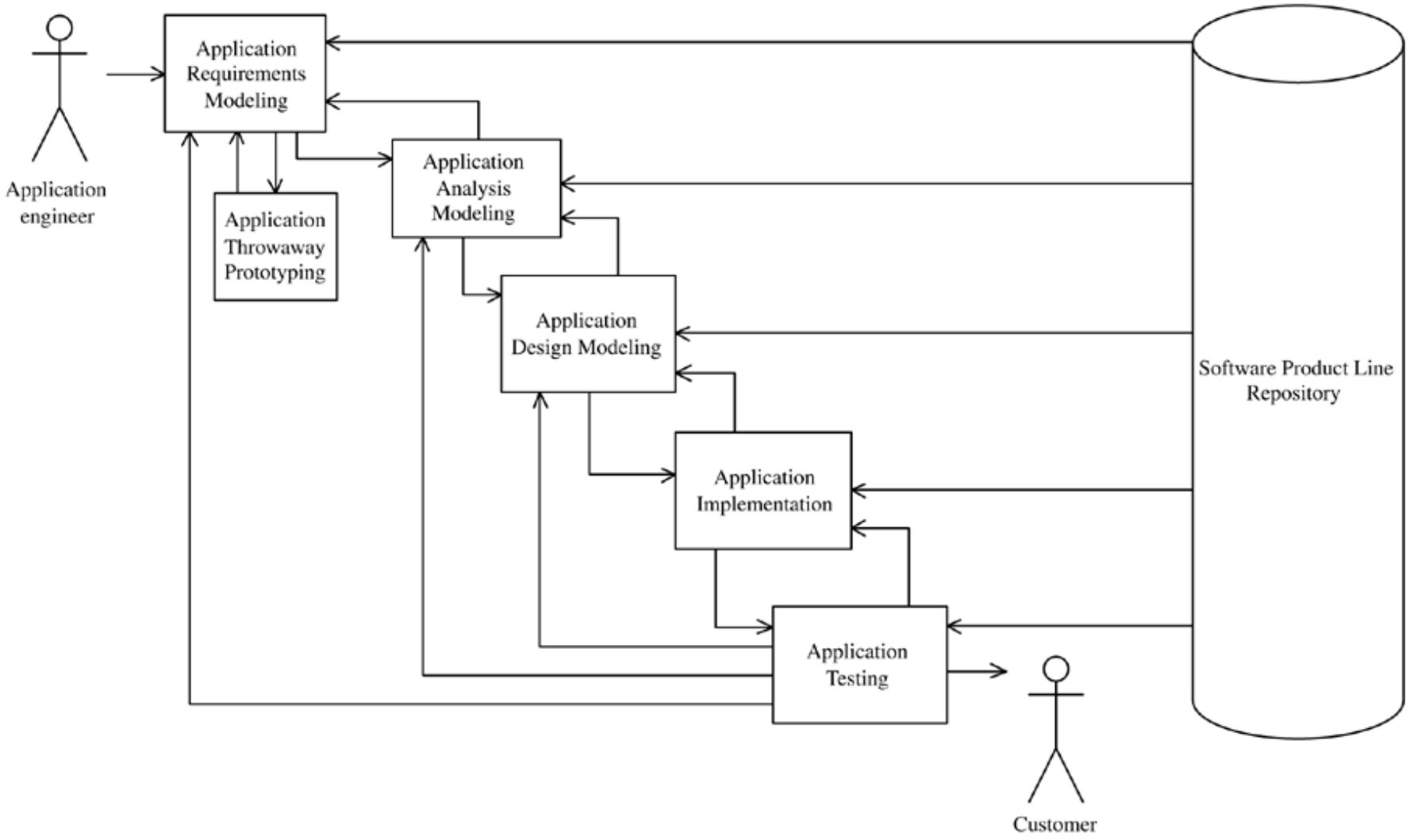
PLUS


Product line engineer
Product Line Requirements Modeling
Throwaway Prototyping
Product Line Analysis Modeling
Product Line Design Modeling
Incremental Component Implementation
Product Line Testing
Product line test engineer
Software Product Line Repository

PLUS

- *Software Product Line Requirements Modeling*
  - Desenvolvimento do modelo de requisitos consistindo de um modelo de casos de uso
  - Feature model
- *Software Product Line Analysis Modeling*
  - Desenvolvimento de modelos estáticos e dinâmicos
  - Desenvolvimento das dependências entre features e classes
- *Software Product Line Design Modeling*
  - Projeto e desenvolvimento de uma arquitetura de software baseada em componentes
  - Modelo de análise (problema do domínio) é mapeado em no modelo de projeto (domínio da solução)
- *Incremental Component Implementation*
- *Product Line Testing*
  - Testes funcionais e de integração

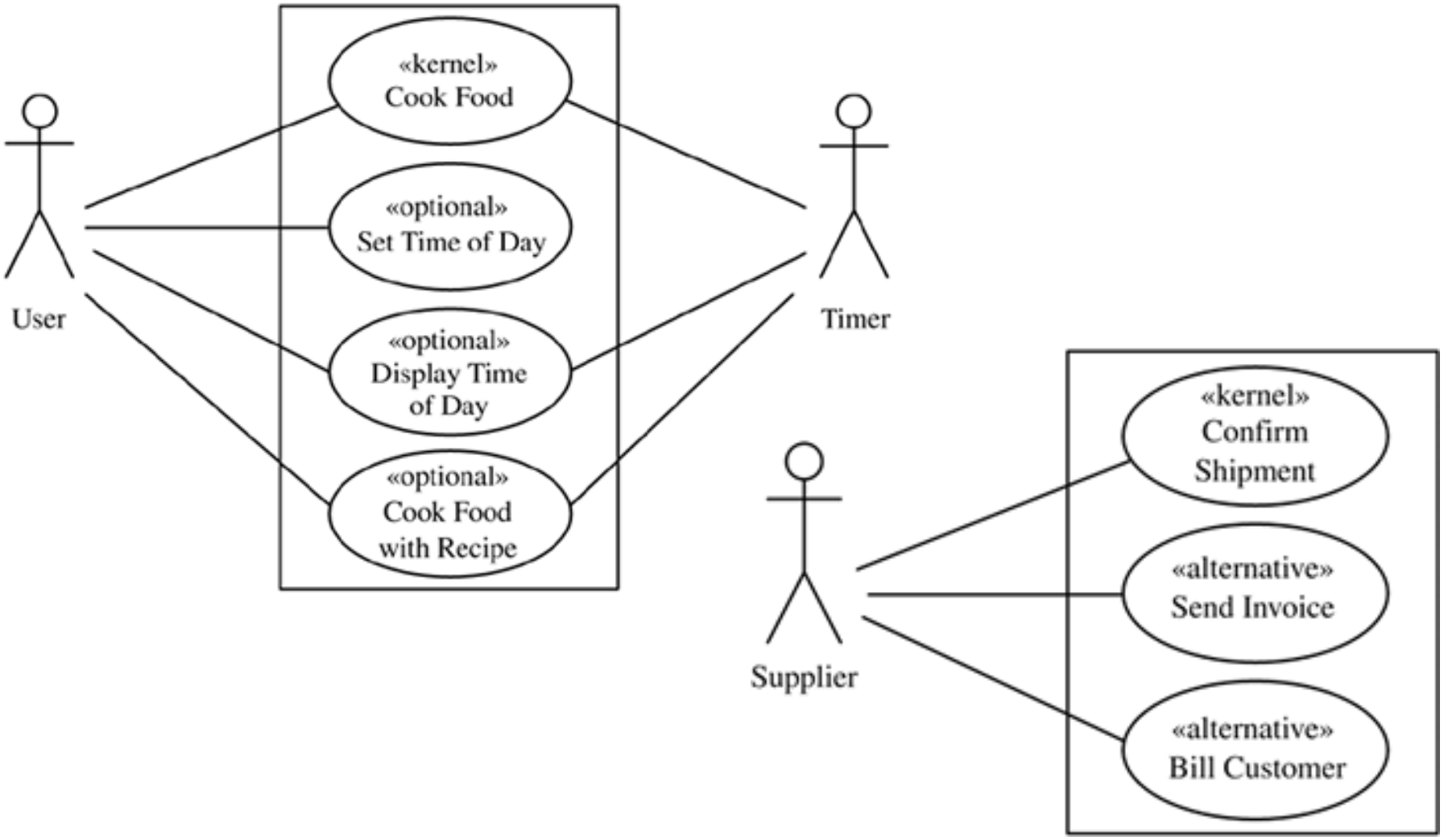
PLUS


Application engineer
Application Requirements Modeling
Application Throwaway Prototyping
Application Analysis Modeling
Application Design Modeling
Application Implementation
Application Testing
Customer
Software Product Line Repository

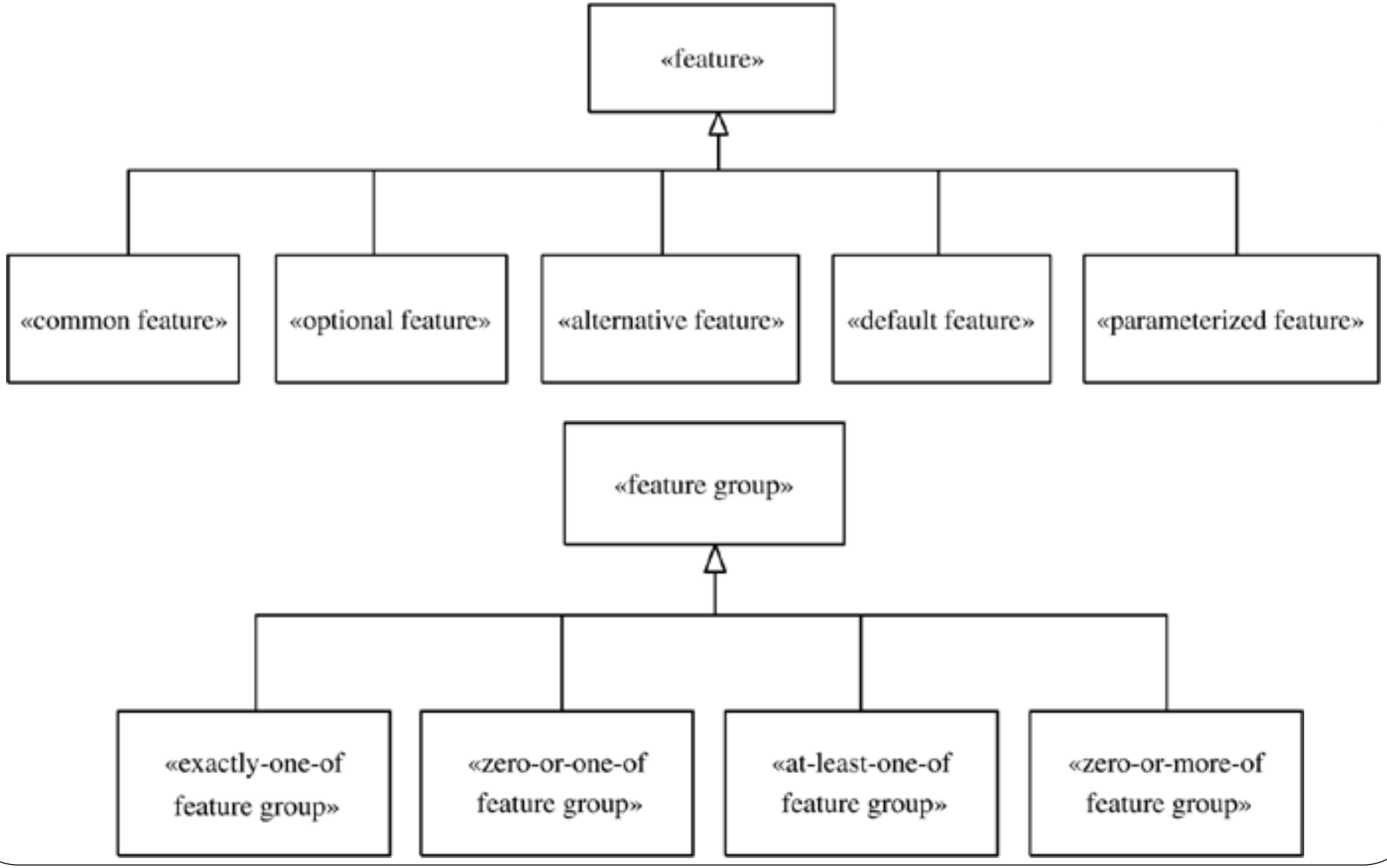
PLUS

- ***Application Requirements Modeling***
  - Desenvolvimento do modelo de requisitos
    - Comparados com feature model da SPL, determinando quais features fazem parte da aplicação
  - Aplicação típica = features do kernel + algumas opcionais e alternativas
  - Tabela de dependência feature/use case, indica
    - Casos de uso fazem parte da aplicação
    - Que variabilidade é inserida nos pontos de variação
- ***Application Analysis Modeling***
  - Modelos estáticos e dinâmicos
- ***Application Design Modeling***
  - Arquitetura de software da aplicação é adaptada a partir da arquitetura da SPL
  - Features da aplicação indicam quais componentes são selecionados
- ***Incremental Application Implementation***
- ***Application Testing***
  - Testes funcionais e de integração

PLUS

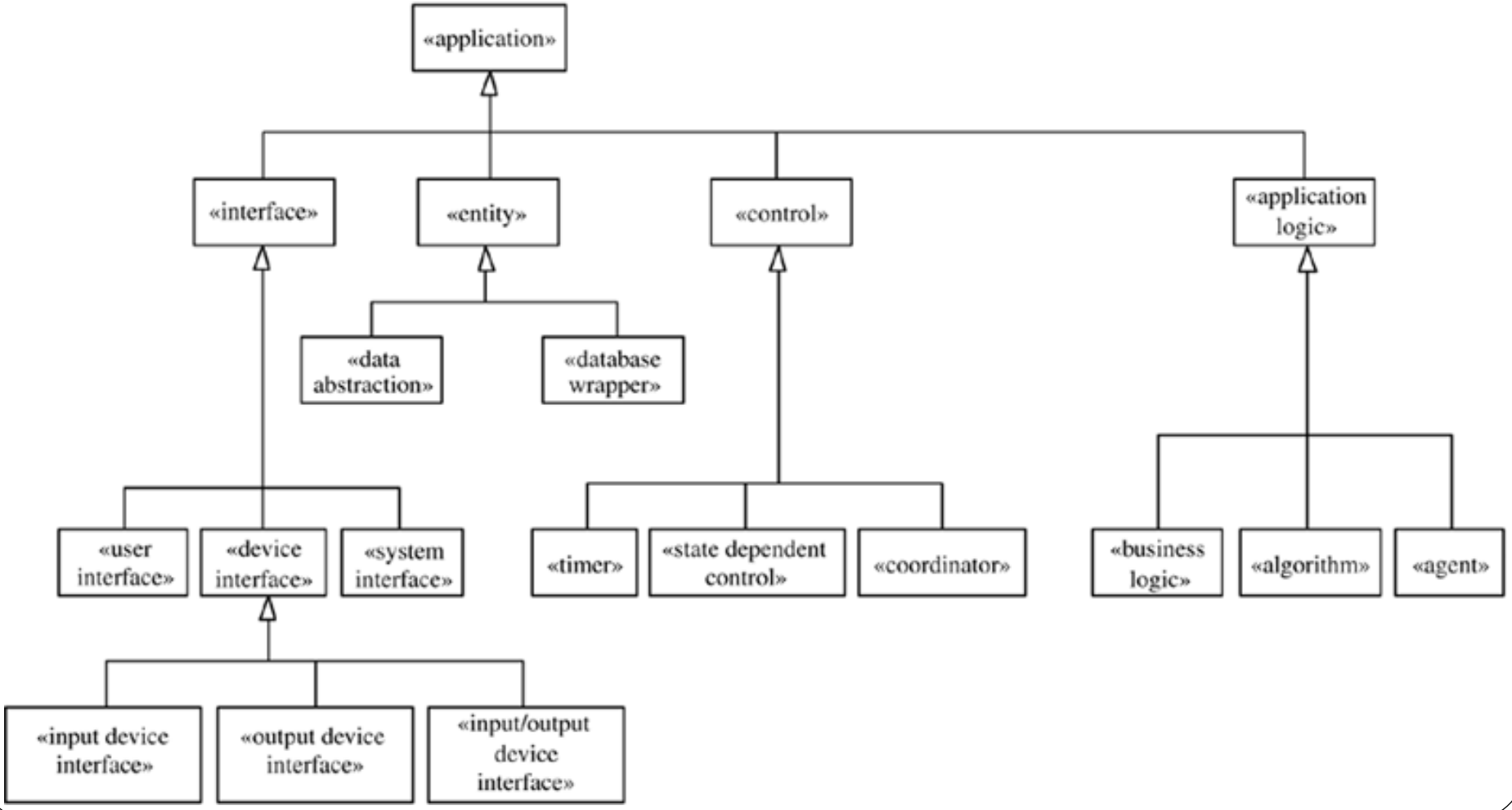

«kernel»
Cook Food
«optional»
Set Time of Day
«optional»
Display Time
of Day
«optional»
Cook Food
with Recipe
User
Timer
Supplier
«kernel»
Confirm
Shipment
«alternative»
Send Invoice
«alternative»
Bill Customer

PLUS


«feature»
«common feature»
«optional feature»
«alternative feature»
«default feature»
«parameterized feature»
«feature group»
«exactly-one-of feature group»
«zero-or-one-of feature group»
«at-least-one-of feature group»
«zero-or-more-of feature group»

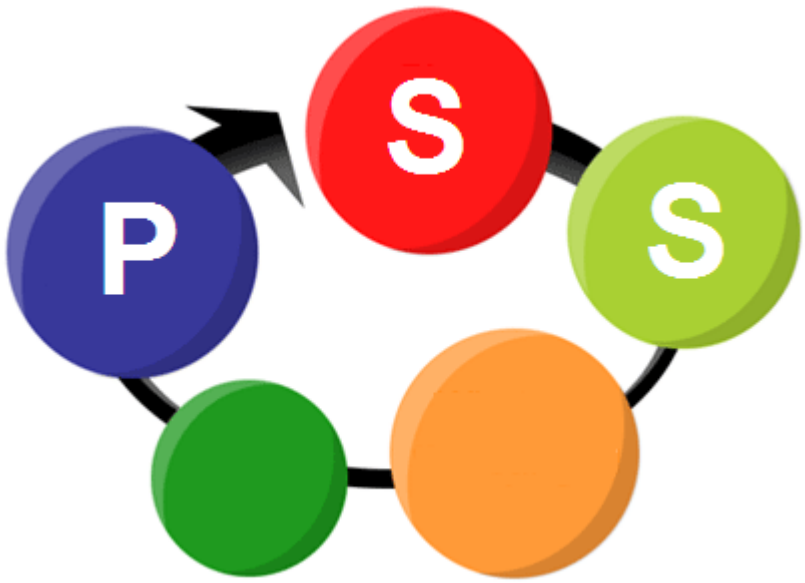
PLUS

- Classificação das classes da aplicação por estereótipos

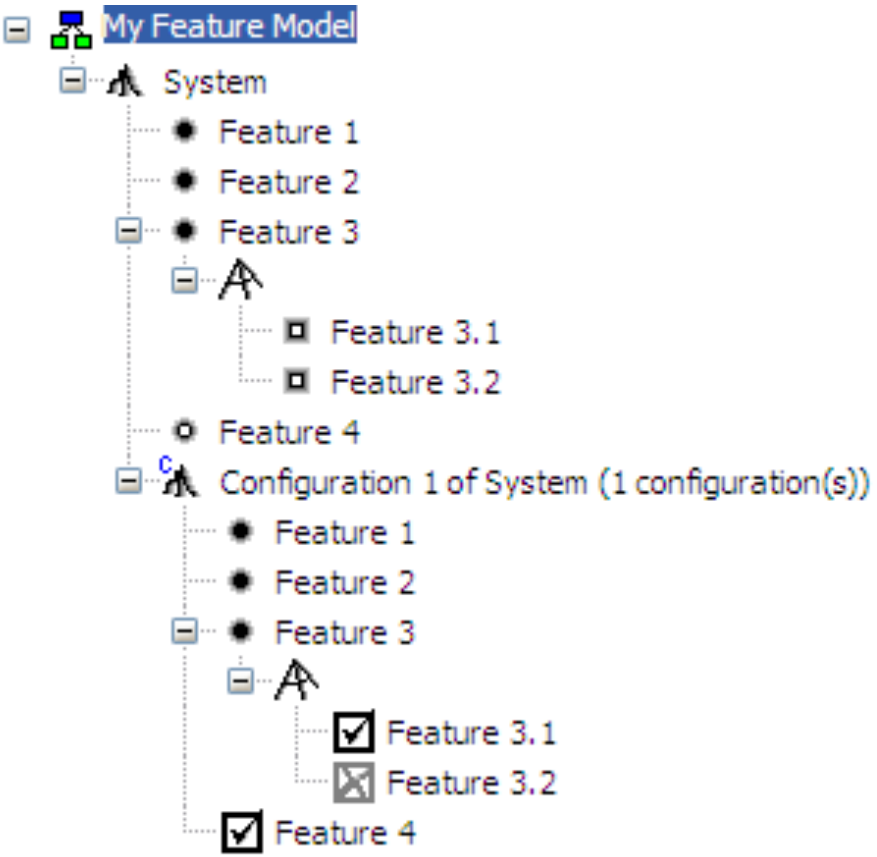


Metodologia Usada em PSS

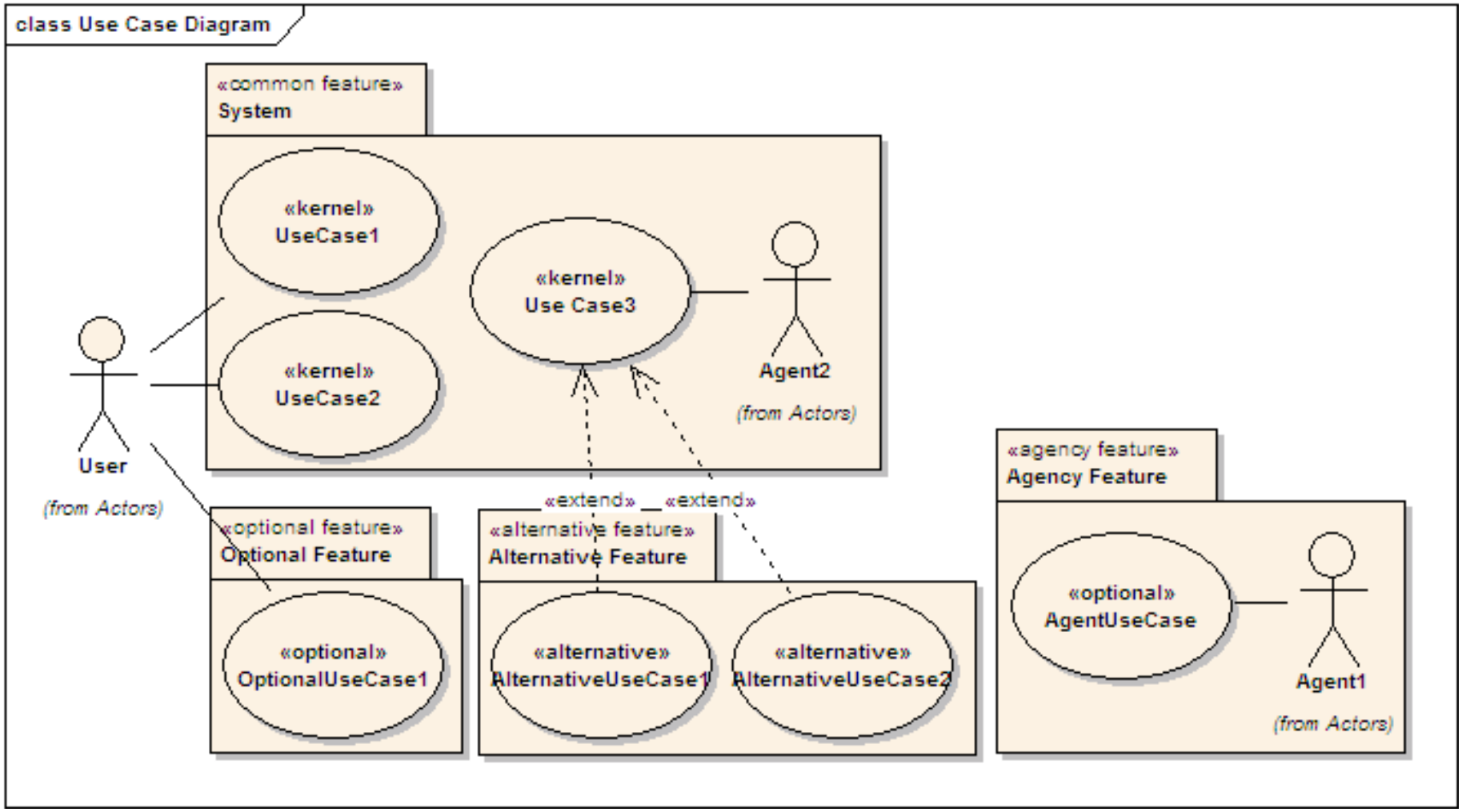
- Domain Engineering
  - Domain Analisys
    - Feature Model
    - Use Case Diagrams
    - Use Case Descriptions
  - Domain Design
    - Class Diagrams
    - Sequence Diagrams
  - Domain Realization
- Application Engineering



Metodologia Usada em PSS



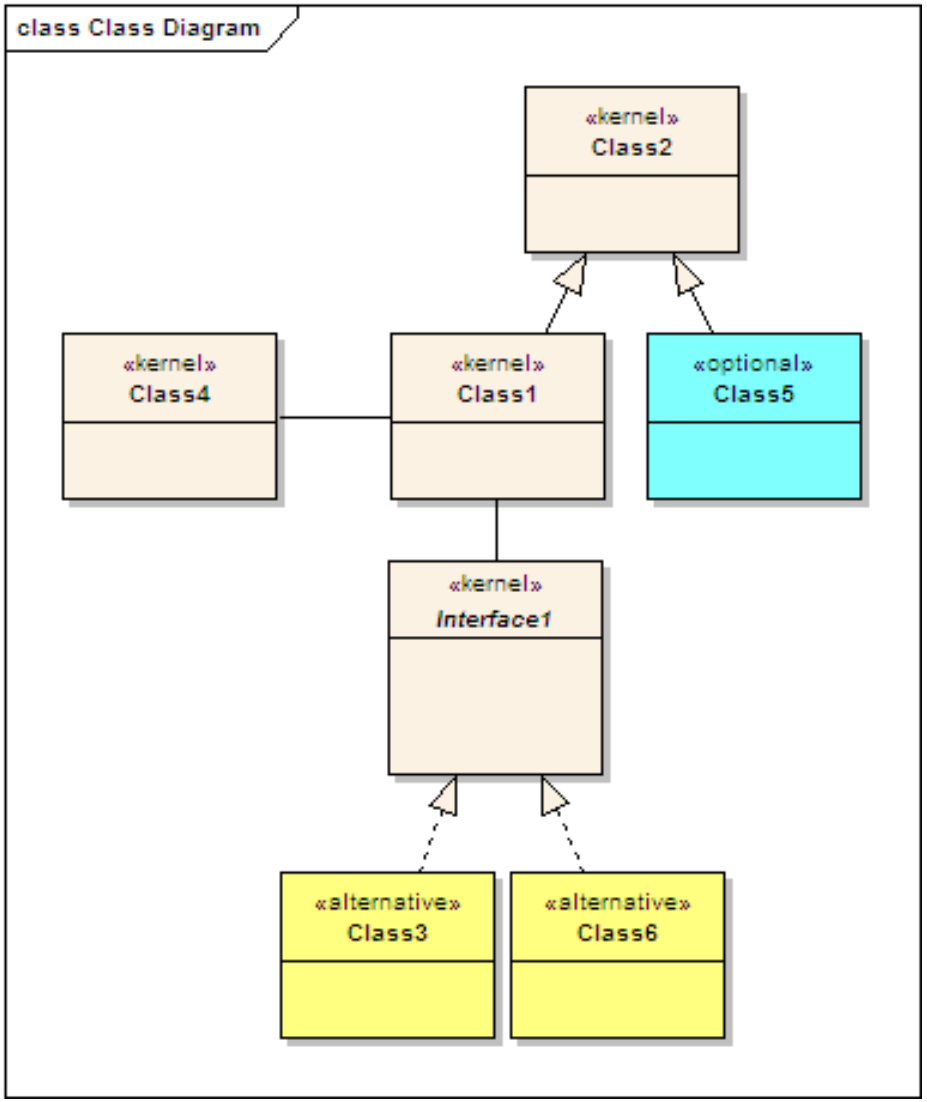
Metodologia Usada em PSS



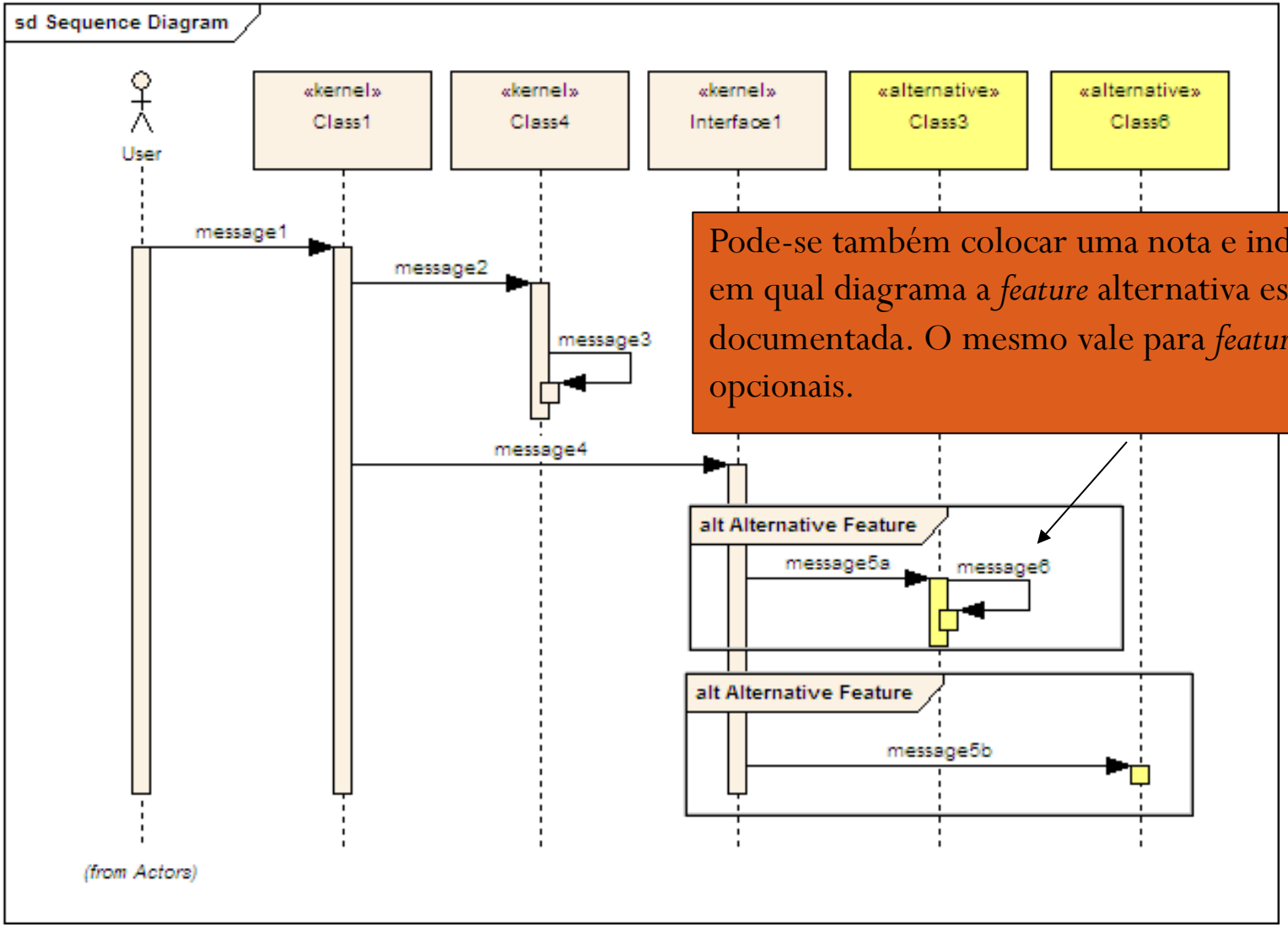
Metodologia Usada em PSS

- Descrições de Casos de Uso
  - Nome do Caso de Uso
  - **Categoria do Reuso**
    - **obrigatório, opcional, alternativo**
  - Descrição
  - Atores
  - **Dependências**
    - **Estende Caso de Uso X**
  - Pré-condições
  - Fluxo Principal
  - Fluxos Alternativos
  - Fluxos de Exceção
  - Estruturas de Dados
  - Regras de Negócio

Metodologia Usada em PSS



Metodologia Usada em PSS



Metodologia Usada em PSS

- ***Application Engineering***
  - Documento de como derivar produto
    - Instruções para Derivação
      - Indicar que artefatos (use cases, diagramas, classes) pertencem ao *kernel* e a cada *feature*
    - Arquivo de Configuração
      - Arquivo de Propriedades
      - Arquivo XML do Spring
    - Instanciação Automática
      - GenArch
  - Apresentar produto derivado

Referências

Atkinson, C., Bayer, J., Muthig, D.: Component-based product line development: The kobrA approach. In Donohoe, P., ed.: Proceedings of theFirstSoftware Product Line Conference. (2000) 289-309.

Cirilo, E., Kulesza, U., Lucena, C.: GenArch: A Model-Based Product Derivation Tool. In: Proceedings of the 1º Simpósio Brasileiro de Componentes, Arquiteturas e Reutilização de Software (SBCARS 2007), Campinas, Brazil (2007) 17-24.

Clements, P., Northrop, L.: Software Product Lines: Practices and Patterns. Addison-Wesley, Boston, MA, USA (2002).

Software Product Lines. http://www.sei.cmu.edu/productlines/

Czarnecki, K., Eisenecker, U.: Generative Programming: Methods, Tools, and Applications. Addison Wesley Longman (2000).

Gomaa H. Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2004.

Referências

Kang K. C., Kim S., Lee J., Kim K., Shin E., and Huh M.. Form: A feature-oriented reuse method with domain-specific reference architectures. Ann. Softw. Eng., 5:143–168, 1998.

Matinlassi M. Comparison of software product line architecture design methods: Copa, fast, form, kobra and qada. In ICSE '04: Proceedings of the 26th International Conference on Software Engineering, pages 127–136, Washington, DC, USA, 2004. IEEE Computer Society.

Nunes I., Nunes C., Kulesza U., and Lucena C. Developing and Evolving a Multi-Agent System Product Line: An Exploratory Study. In: 9th International Workshop on Agent-Oriented Software Engineering, 2008, Estoril. 9th International Workshop on Agent-Oriented Software Engineering, 2008. p. 177-188.

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

Weiss D. M., Lai C. T. R.: Software Product-line Engineering: A Family-Based Software Development Process, Addison-Wesley, 1999.