Software Product Line Engineering with SMarty

Prof. Dr. Edson OliveiraJr
Informatics Department
State University of Maringá, Brazil

edson@din.uem.br

Who am I?

[http://lattes.cnpq.br/8717980588591239]

- Dr. Edson OliveiraJr
  - Assistant Professor (2011-2019)
  - Associate Professor (2019-now)
  - Informatics Department
  - State University of Maringá (UEM), Brazil
- Education:
  - Bachelor in Informatics (UEM, 2003)
  - M.Sc. in Computer Science (UEM, 2005)
  - Ph.D. in Computer Science (ICMC/USP, 2010)
    - Visiting Scholar - Sandwich (University of Waterloo, Canada, feb-dec/2009)
  - Post-Doctorate Stage (PUCRS, 2018-2020)
  - Visiting Professor (PUCRS, 2022-now)
- Research Interests
  - Software Engineering
    - Software Reuse
    - Software Product Lines
    - Software Process Lines
    - Software Architectures and Quality Attributes
    - Metrics and Measures
    - Empirical Studies: RSL, MSL, Surveys, Qualitative Studies, Controlled Experiments
    - Education
  - Digital Forensics:
    - Controlled Experimentation
    - Tools Requirements
    - Ontologies and Conceptual Models
  - Open Science Practices:
    - Openness of Experiments and Secondary Studies

GREATER

Research Group on Systematic Software Reuse and Continuous Experimentation

Engenharia de Software

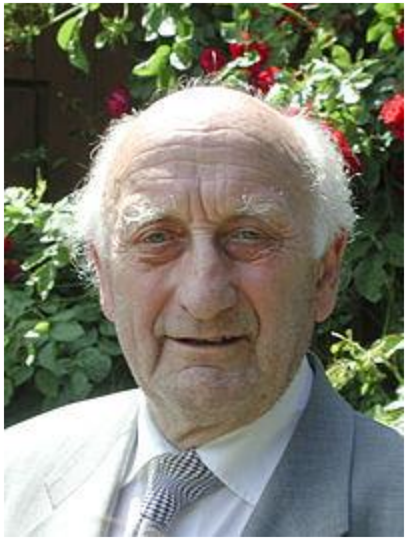
- Quem criou o termo?
  - Margaret Hamilton, MIT
  - Apollo 11 – 1ª. missão tripulada à Lua
    - Software de Navegação
    - Medalha Presidencial da Liberdade
- Quando surgiu?
  - Final da década de 60 (~1968)
  - Crise do Software (E. Dijkstra, 1972)
    - The Humble Programmer

http://www.cs.utexas.edu/users/EWD/ewd03xx/EWD340.PDF

Engenharia de Software

- Qual a definição (1)?
  - "Engenharia de Software é a criação e a utilização de *sólidos princípios de engenharia* a fim de obter *software de maneira econômica*, que seja *confiável* e que trabalhe em máquinas reais"

(Prof. Friedrich L. Bauer, Univ. de Munique, 1969)



Engenharia de Software

- Qual a definição (2)?
  - “The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the *application of engineering to software*.”

(IEEE Standard Glossary of Software Engineering Terminology, 1990, 2002)

http://ieeexplore.ieee.org/document/159342

IEEE
Std 610.12-1990
(Revision and redesignation of IEEE Std 792-1983)

IEEE Standard Glossary of Software Engineering Terminology

Sponsor

Standards Coordinating Committee of the Computer Society of the IEEE

Approved September 28, 1990

IEEE Standards Board

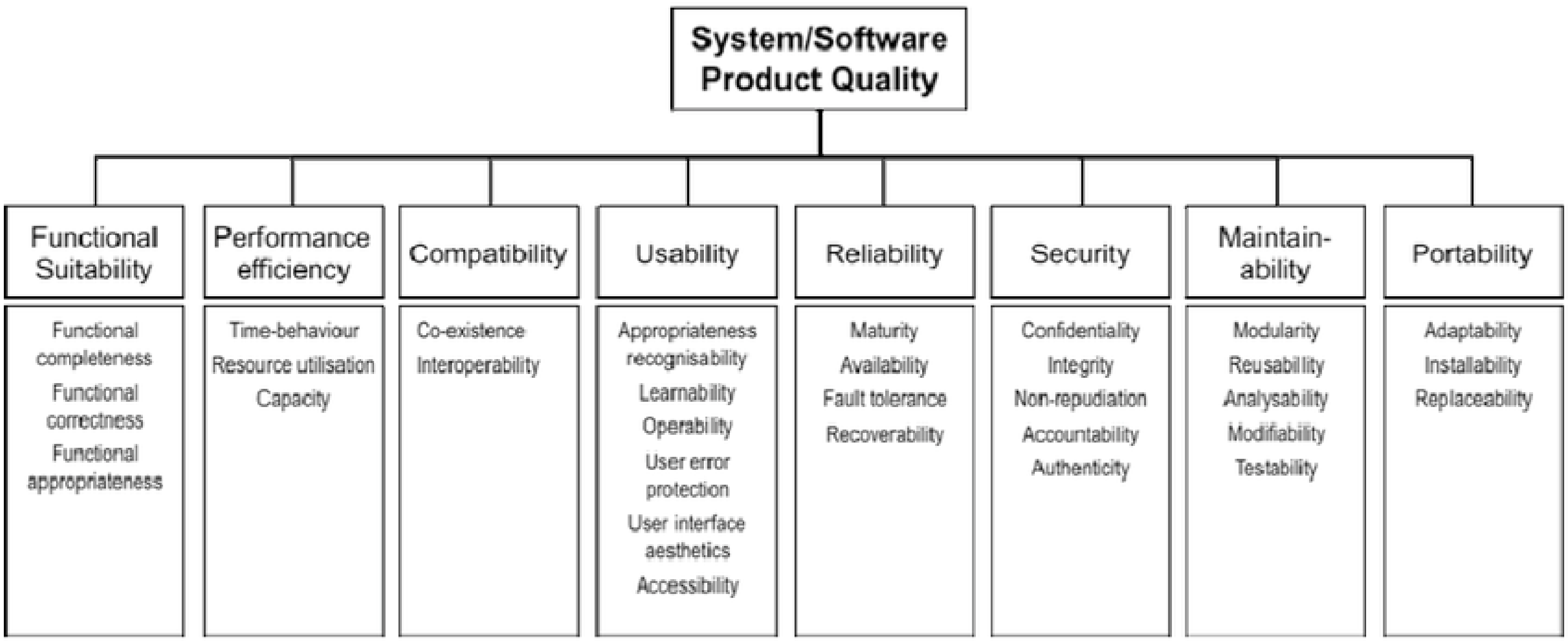
Abstract: IEEE Std 610.12-1990, *IEEE Standard Glossary of Software Engineering Terminology*, identifies terms currently in use in the field of Software Engineering. Standard definitions for those terms are established.
Keywords: Software engineering; glossary; terminology; definitions; dictionary

ISBN 1-55937-067-X

Qualidade de Software

- Qualidade de Software relaciona-se a:
  - Qualidade do produto
  - Qualidade do processo
- Qualidade do Produto:
  - ISO/IEC 9126 depois ISO/IEC 25010

Qualidade de Produto de Software ISO/IEC 25010



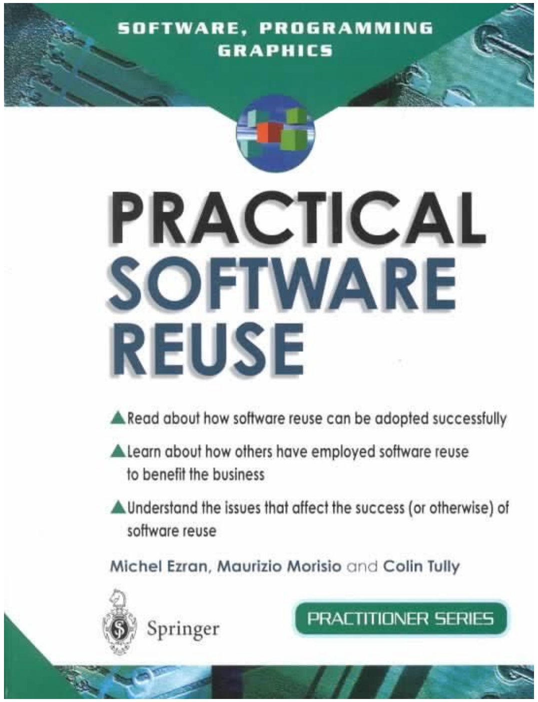
Qualidade de Software

- Como conseguir qualidade em algo abstrato como software?
  - Não existe uma receita de bolo para isso!!
  - Uma possibilidade é a combinação de várias técnicas....
  - Dentre elas....
- Reuso de Software!!!

Reuso de Software

- "A prática sistemática do desenvolvimento de software a partir de um conjunto de blocos, de forma que similaridades em termos de requisitos e/ou arquitetura entre aplicações possam ser exploradas para se alcançar substanciais benefícios em produtividade, qualidade e desempenho do negócio"

  (M. Ezran, M. Morisio, C. Tully 2002)

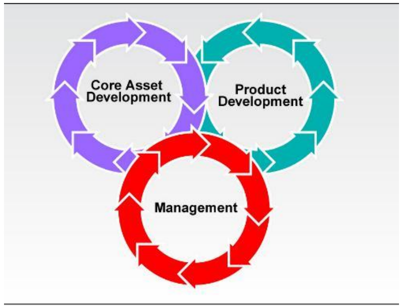


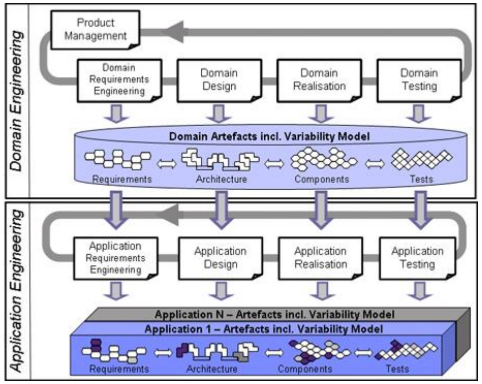
Reuso de Software

www.ic.unicamp.br/~eliane/Cursos/Transparencias/Manutencao/reuso.ppt
www.les.inf.puc-rio.br/wiki/images/2/28/Aula06_software_product_lines_v2.ppt

Histórico:

1960 Reutilização de **linhas de código** de um programa em outro
1970 Reutilização de **código comum** (subrotinas)
Reutilização de **funções genéricas** (bibliotecas de funções)
1980 OO: **herança, composição / delegação**
uso de **interfaces** (implementadas, em algumas linguagens, por classes abstratas
Polimorfismo e ligação dinâmica (*late binding*): qqr implementação da interface pode ser usada em tempo de execução
1990 **Padrões de software**: reutilização de várias classes e de suas colaborações. reutilização
não mais restrita ao código.
Frameworks: reutilização de análise, projeto, implementação e testes de domínios de aplicações.
Componentes: reutilização de código executável, configurável, adaptável.
2000 **Linhas de Produto de Software**
2004 **Serviço**: reutilização de unidade autônoma de execução (função de negócio).
2007 **Arquiteturas de Referência, Ecossistemas, SoS, etc...**
2015 **Micro-serviços**





Engenharia de Linha de Produto de Software

Engenharia de LPS

- Paradigma para desenvolver aplicações de software (*software-intensive systems*) com base em plataforma e customização em massa
- "Uma LPS representa um conjunto de sistemas que compartilham uma infraestrutura central comum e reutilizável para um determinado segmento de mercado ou missão" (Clements, Northrop, 2001)

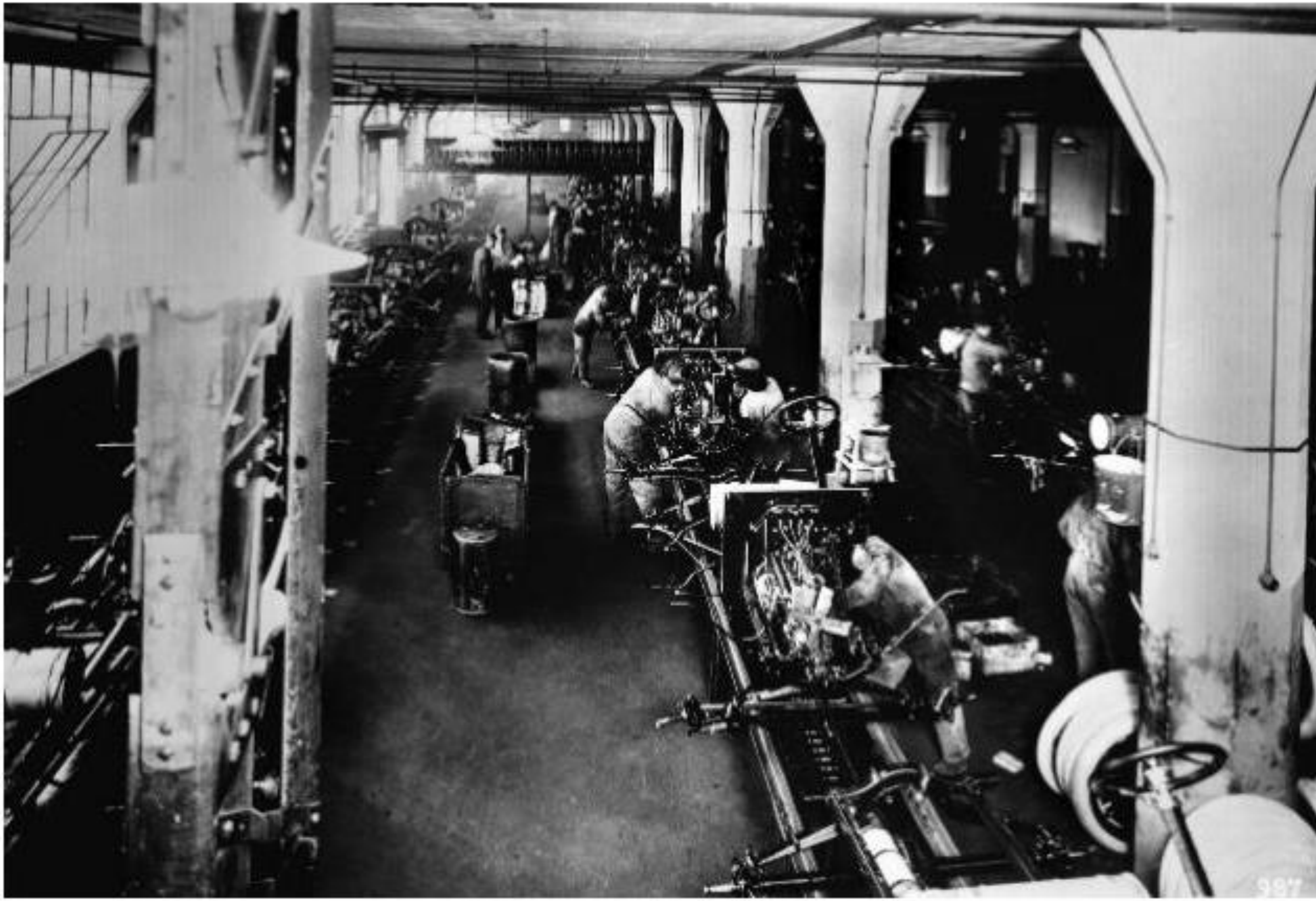


Fig. 1.1 Ford assembly line [48]

Linhas de Produto de Software

- Por que **<u>Engenharia</u>** de Linha de Produto de Software? Porque é:

  - sistemático

  - disciplinado (prescritivo)

  - quantificável

  - não-oportunístico!!

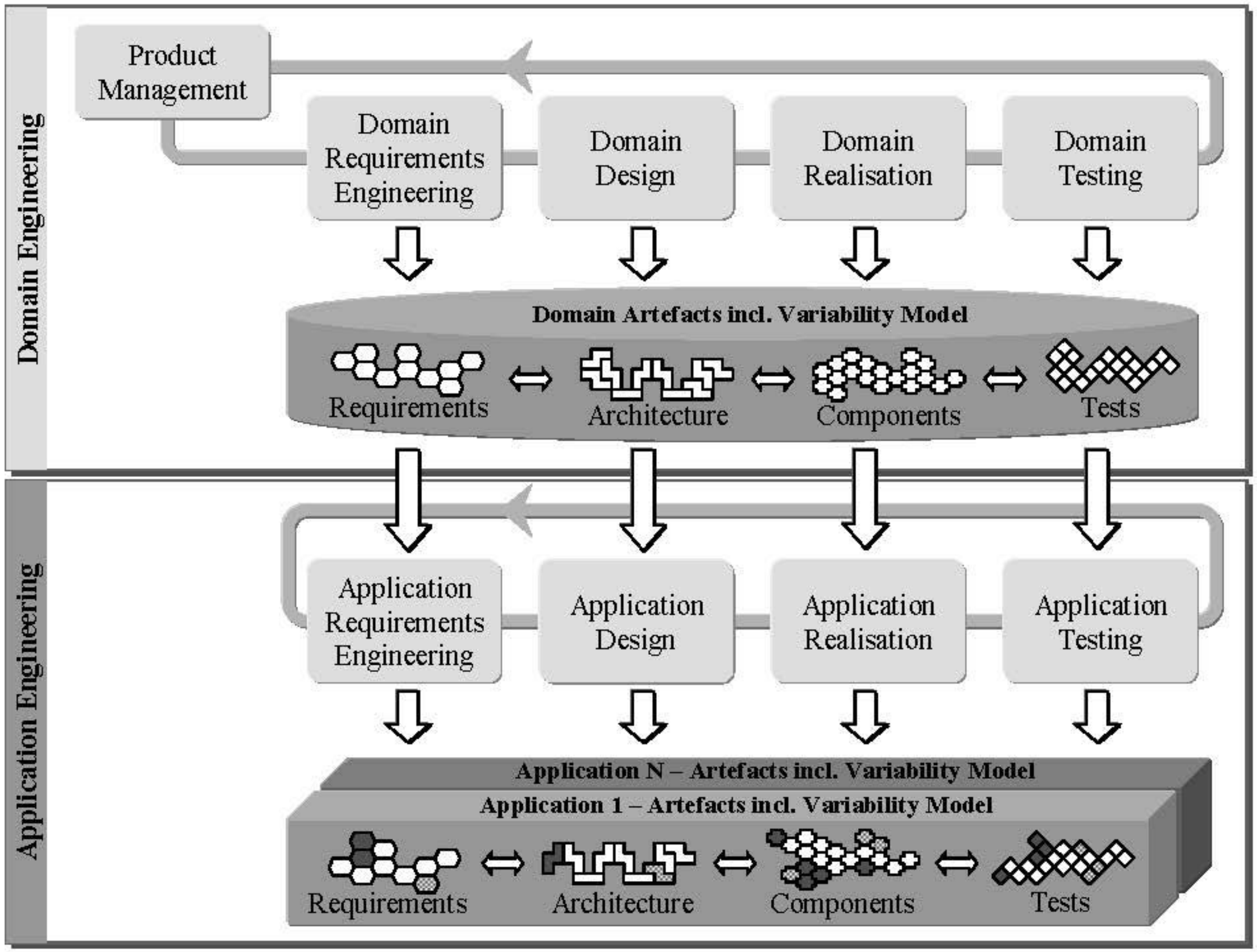
Linhas de Produto de Software

- A Engenharia de LPS se baseia essencialmente em três atividades conhecidas como:
  - Engenharia de Domínio
  - Engenharia de Aplicação
  - Gerenciamento da LPS

Linhas de Produto de Software

- Engenharia de Domínio
  - identifica e especifica as características similares e variáveis de uma LPS, além dos RF, RNF e de domínio
  - especifica uma arquitetura de LPS (*Product-Line Architecture* – PLA) → principal artefato da LPS
    - abstração da arquitetura de software de todos os possíveis/potenciais produtos de uma LPS
  - disponibiliza uma infraestrutura central conhecida como *Core Assets:*
    - artefatos do domínio: requisitos, PLA, componentes, testes, modelos de variabilidade, etc..
  - design, realização e inspeção/teste dos artefatos

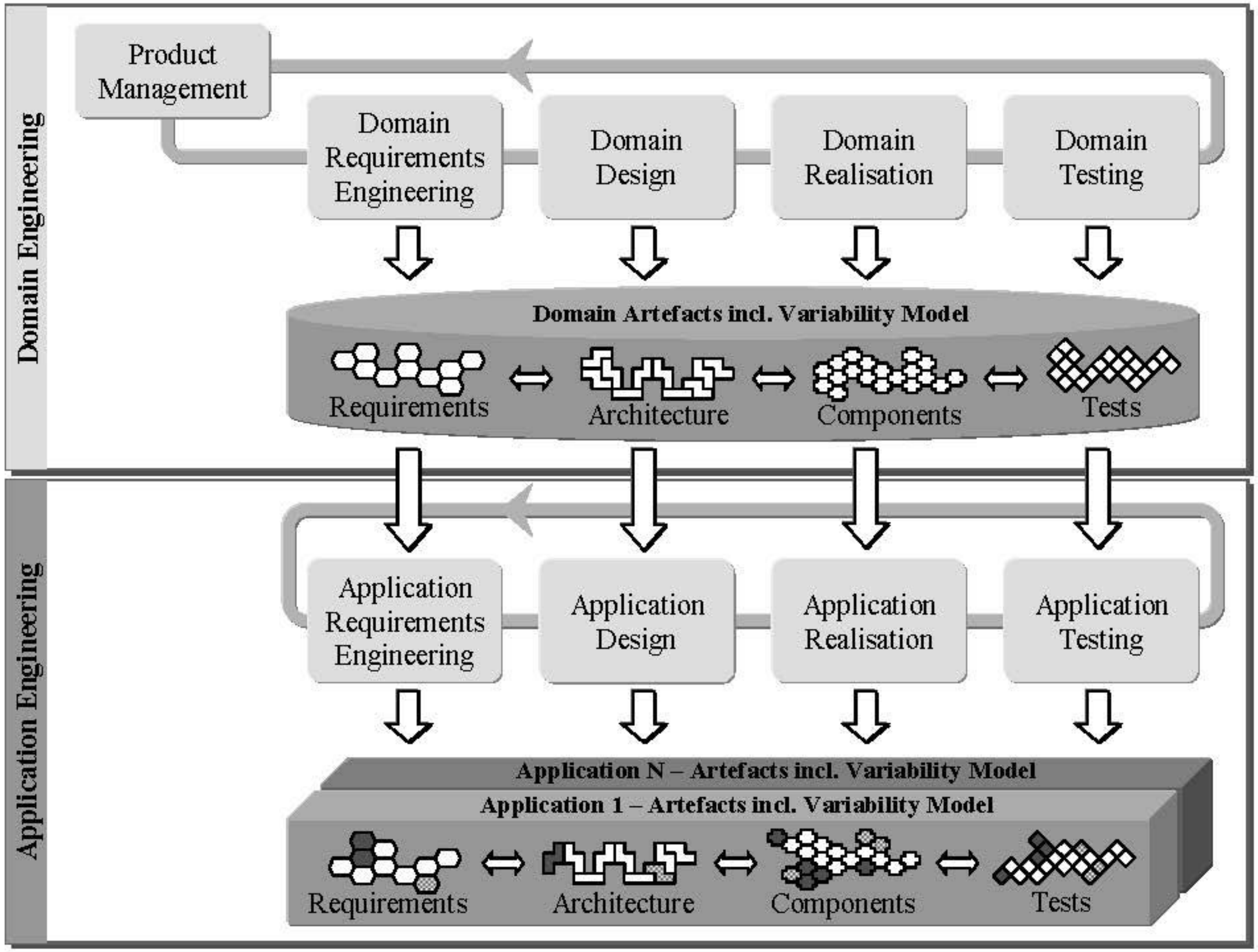
Framework de Engenharia de LPS (Pohl et al., 2005)



Linhas de Produto de Software

- Engenharia de Aplicação
  - produz produtos específicos com base:
    - em um plano de produção;
    - na instanciação da PLA;
    - na resolução das variabilidades de acordo com os requisitos do produto.
  - Gera um produto que contém:
    - RF e RNF (possivelmente requisitos de domínio) e restrições
    - uma arquitetura de software concreta e com componentes (reusados ou COTS)
    - sequência de teste, casos de teste, scripts de execução, etc

Framework de Engenharia de LPS (Pohl et al., 2005)



Linhas de Produto de Software

- Conceito de *Feature* (Característica):
  - conceito presente em praticamente todas as abordagens de LPS
  - forma de representação de variabilidades em LPS
  - origem no método *Feature-Oriented Domain Analysis* (FODA) (Kang, et al., 1990)
    - http://www.sei.cmu.edu/reports/90tr021.pdf

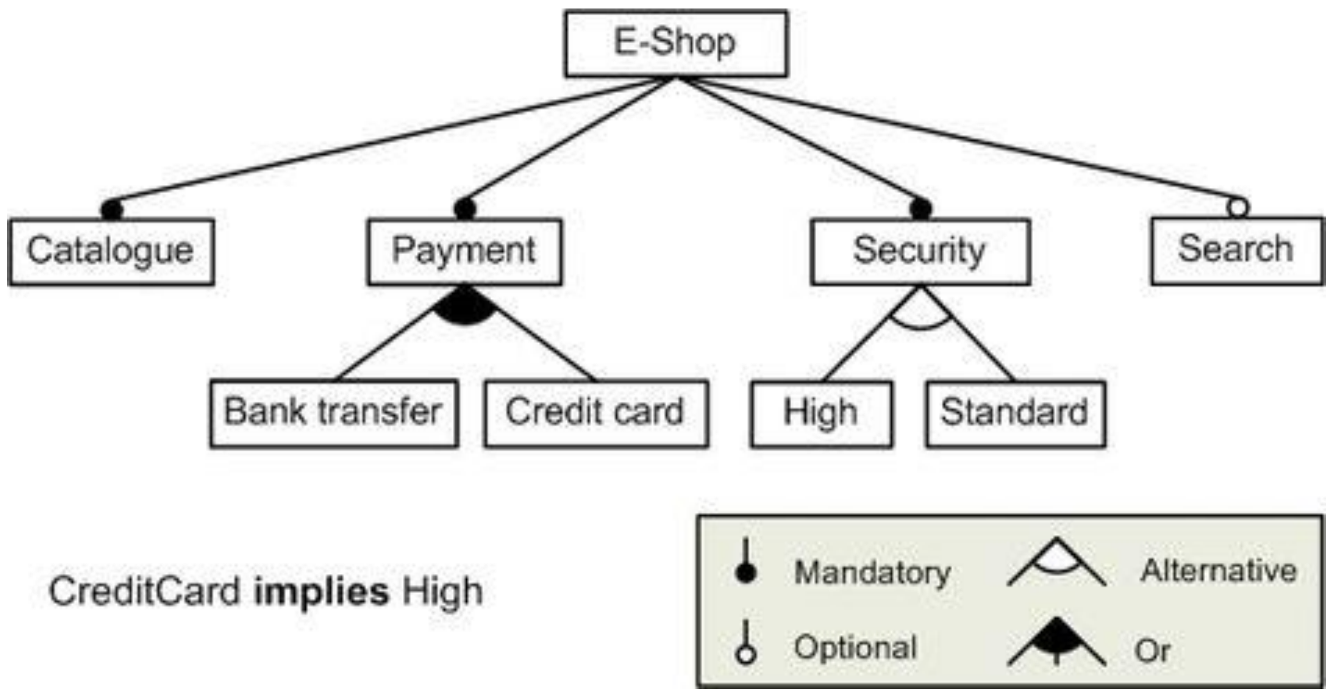
"Um aspecto, qualidade ou característica proeminente ou distintiva visível pelo usuário de um sistema ou sistema de software"

Linhas de Produto de Software

- Modelo de *Features* (*Feature Model*):
  - Representação compacta de todos os produtos de uma LPS em termos de suas características
  - *Features* são transversais aos artefatos do *Core Assets*
  - Contém:
    - Diagrama de *features*: notação visual hierárquica de um modelo de *features*
    - Configurações: conjunto de *features* selecionadas que descrevem um produto específico de uma LPS

Linhas de Produto de Software

- Exemplo de Diagramas de *Features*



Linhas de Produto de Software

- Ferramentas de apoio à modelagem de features:
  - Ahead Tool suite, BeTTy Framework, BeTTy Online Feature Model Generator, Clafer, Eclipse Modeling Framework Feature Model Project, FaMa Tool Suite, Feature Model Plug-in, Feature Modeling Tool, a plug-in for Visual Studio 2008, FeatureMapper, FAMILIAR, FeatureIDE, FLAME, Gears, Hydra, LieberLieber Feature Modeler, MOSKitt Feature Modeler, Pure::Variants, Requiline, S2T2 Configurator, SPLOT (Software Product Line Online Tools), ToolDAy - Tool for Domain Analysis, TouchCORE, XFeature, ZIPC Feature

Variabilidade em LPS

- Variabilidade é a capacidade de um artefato se adaptar ao uso em diferentes produtos no escopo de uma LPS.
- Em LPS variabilidade começa já no modelo de features e é aplicada em praticamente todos os artefatos do *core assets*, incluindo a PLA

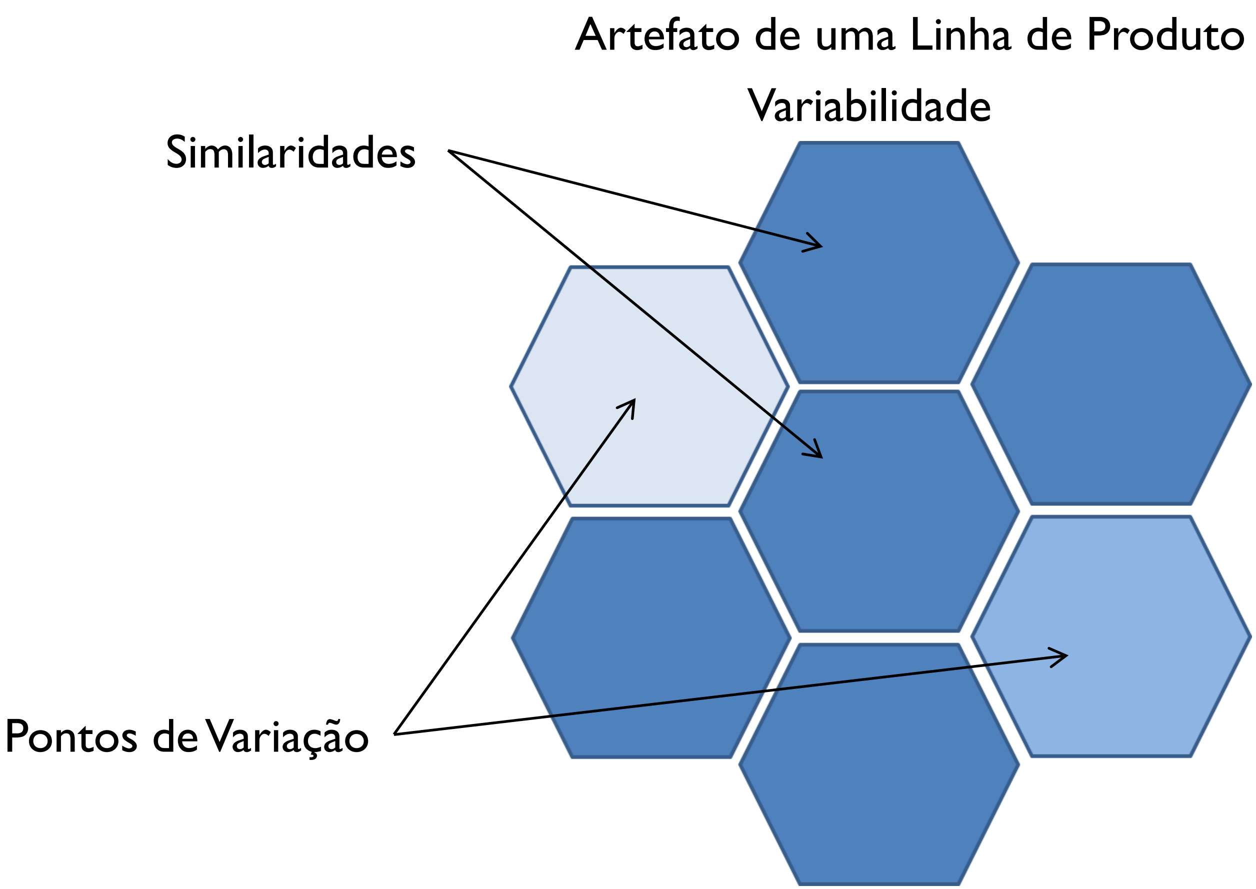
Variabilidade em LPS

- Variabilidade é normalmente descrita em termos de:
  - Ponto de Variação
    - Local específico em um artefato
  - Variante
    - Possível alternativa para resolução de um PV
  - Restrições entre Variantes
    - Permitem rastreabilidade
      - Requires
      - Mutex

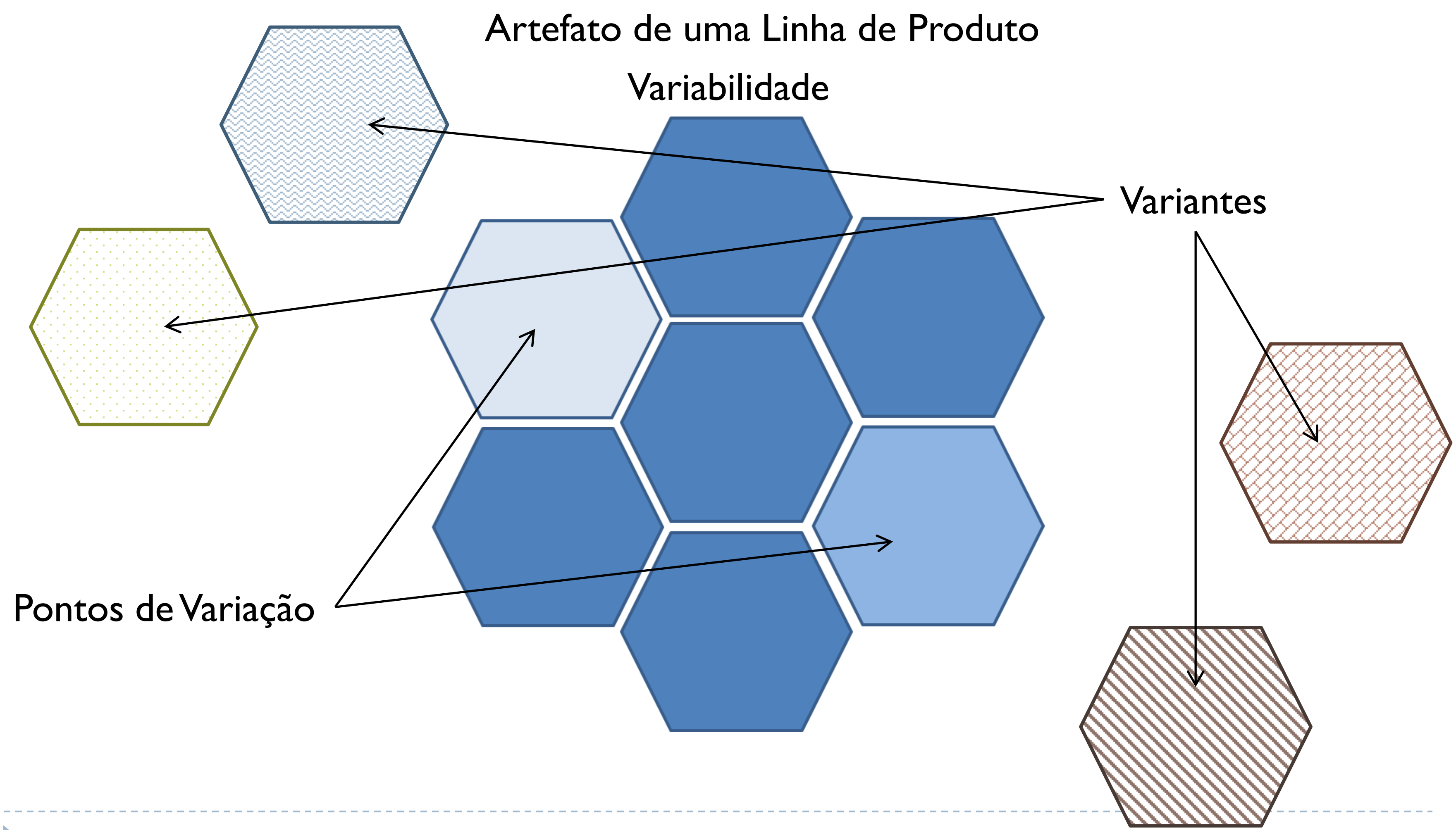
Variabilidade em LPS

- Pontos de variação e variantes podem ser:
  - Obrigatórias
    - Todos os produtos possuem tal artefato
  - Opcionais
    - Alguns produtos podem conter tal artefato
  - Inclusivas (OR)
    - Seleção de pelo menos um artefato dentre um conjunto possível
  - Alternativas (XOR)
    - Somente um artefato de um conjunto pode ser selecionado

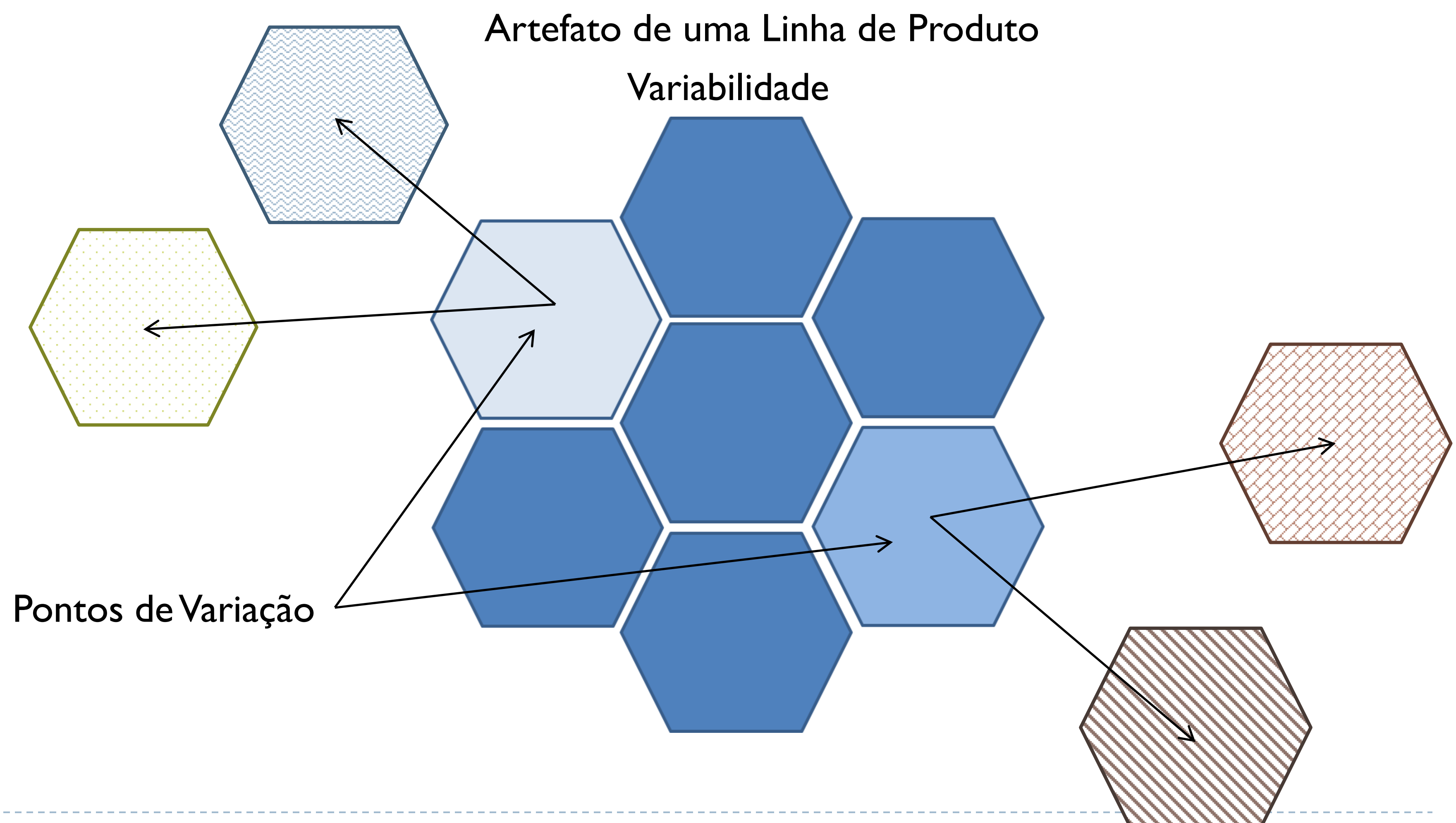
Variabilidade em LPS



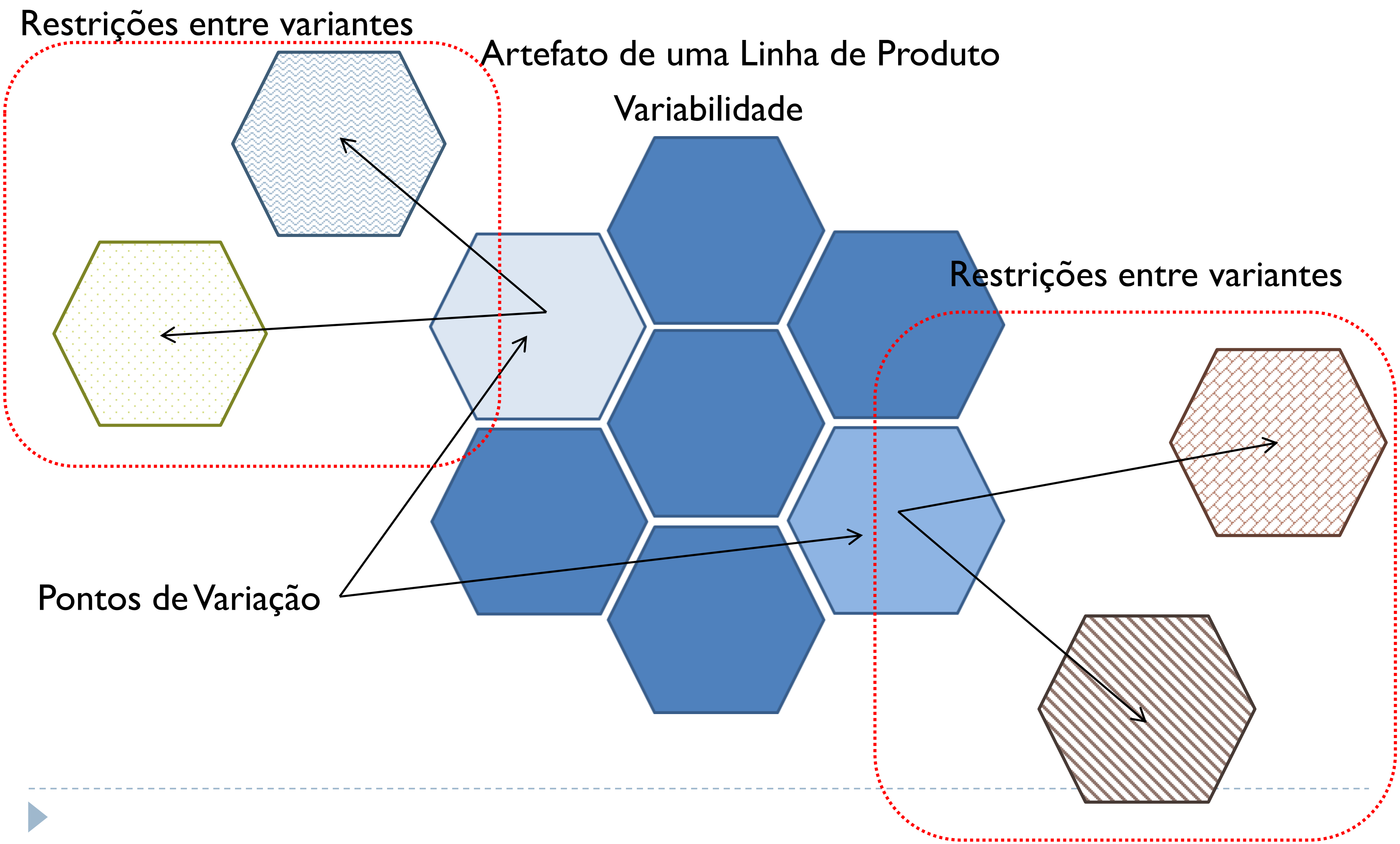
Variabilidade em LPS



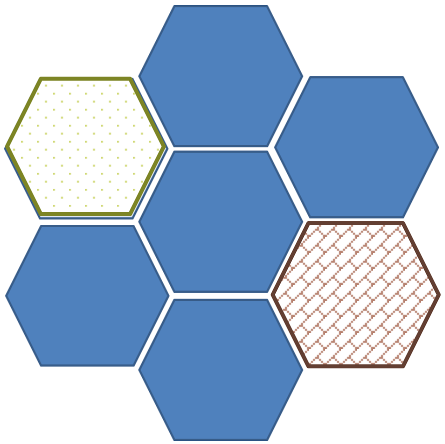
Variabilidade em LPS



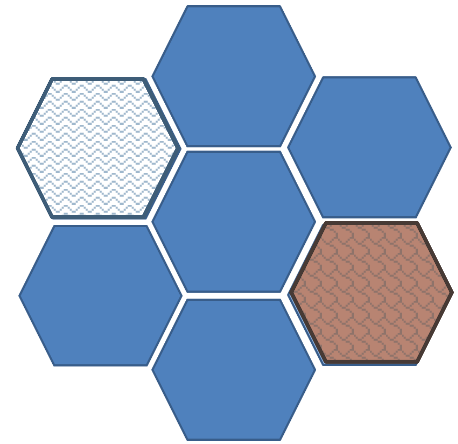
Variabilidade em LPS

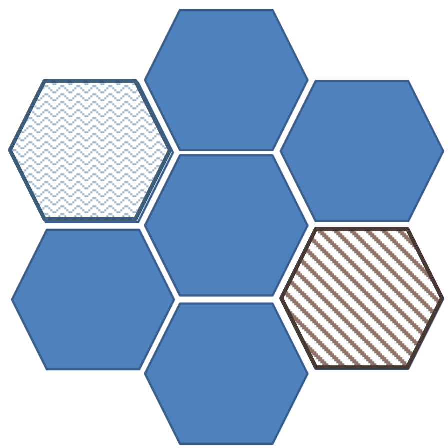


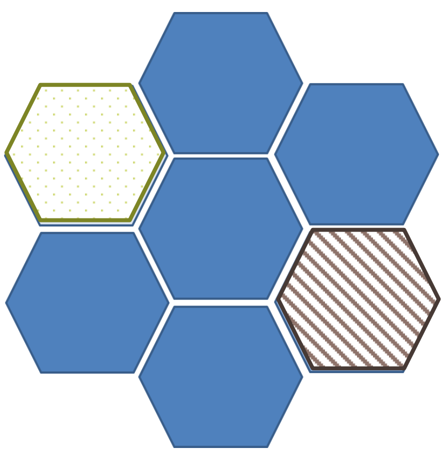
Variabilidade em LPS

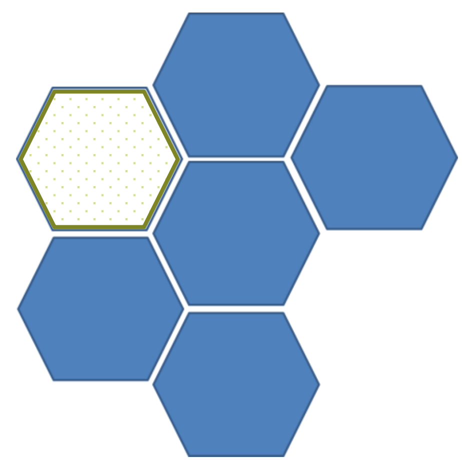


Possíveis produtos da LPS

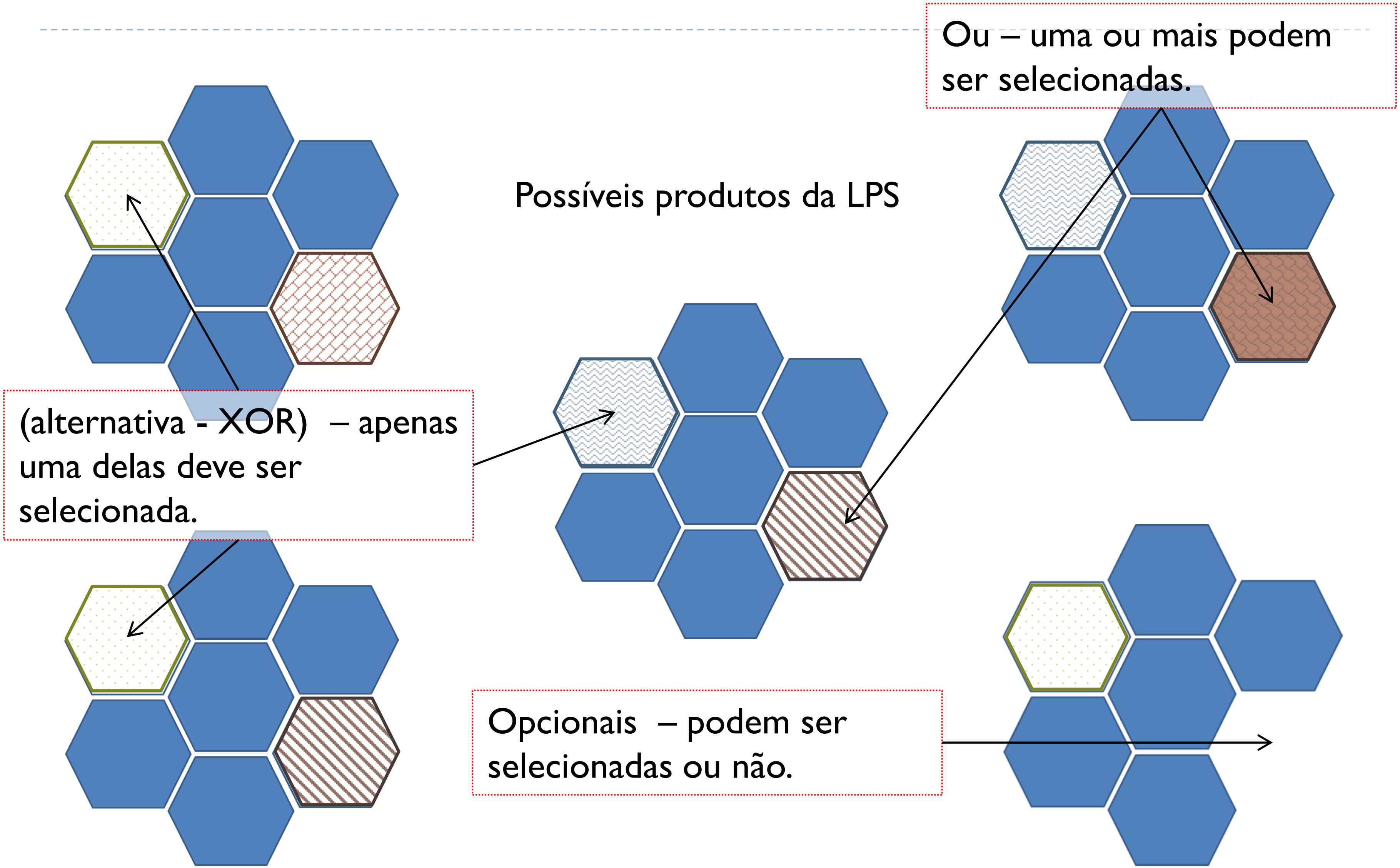




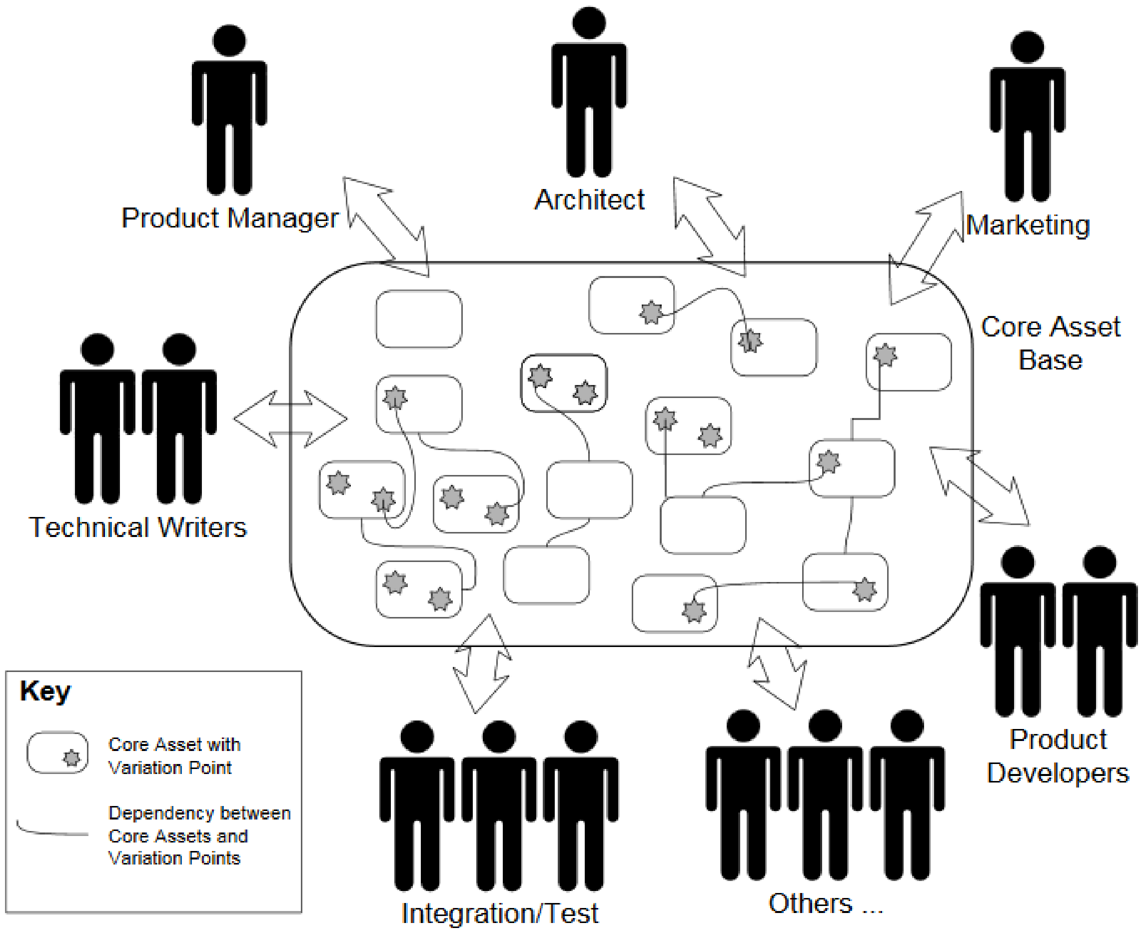




Variabilidade em LPS



Variabilidade em LPS



https://resources.sei.cmu.edu/asset_files/TechnicalReport/2005_005_001_14600.pdf

A Abordagem SMarty

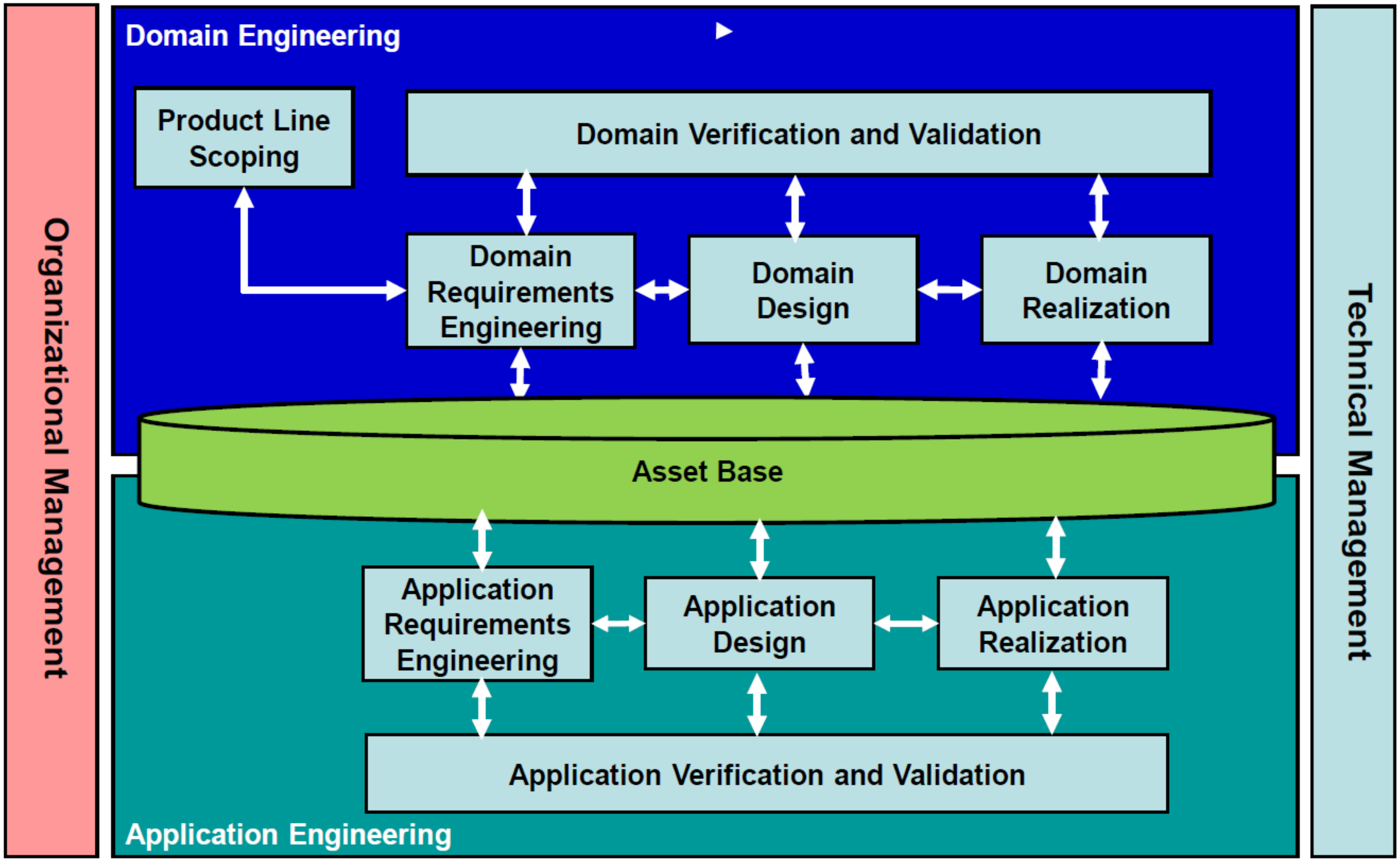
SMarty

- SMarty significa:
  - **S**tereotype-based **M**anagement of V**ar**iabili***ty***
- Começou como uma abordagem de gerenciamento de variabilidades
  - Com base no metamodelo da UML e profiling
- Hoje é uma abordagem [semi]completa para Engenharia de LPS baseada em UML

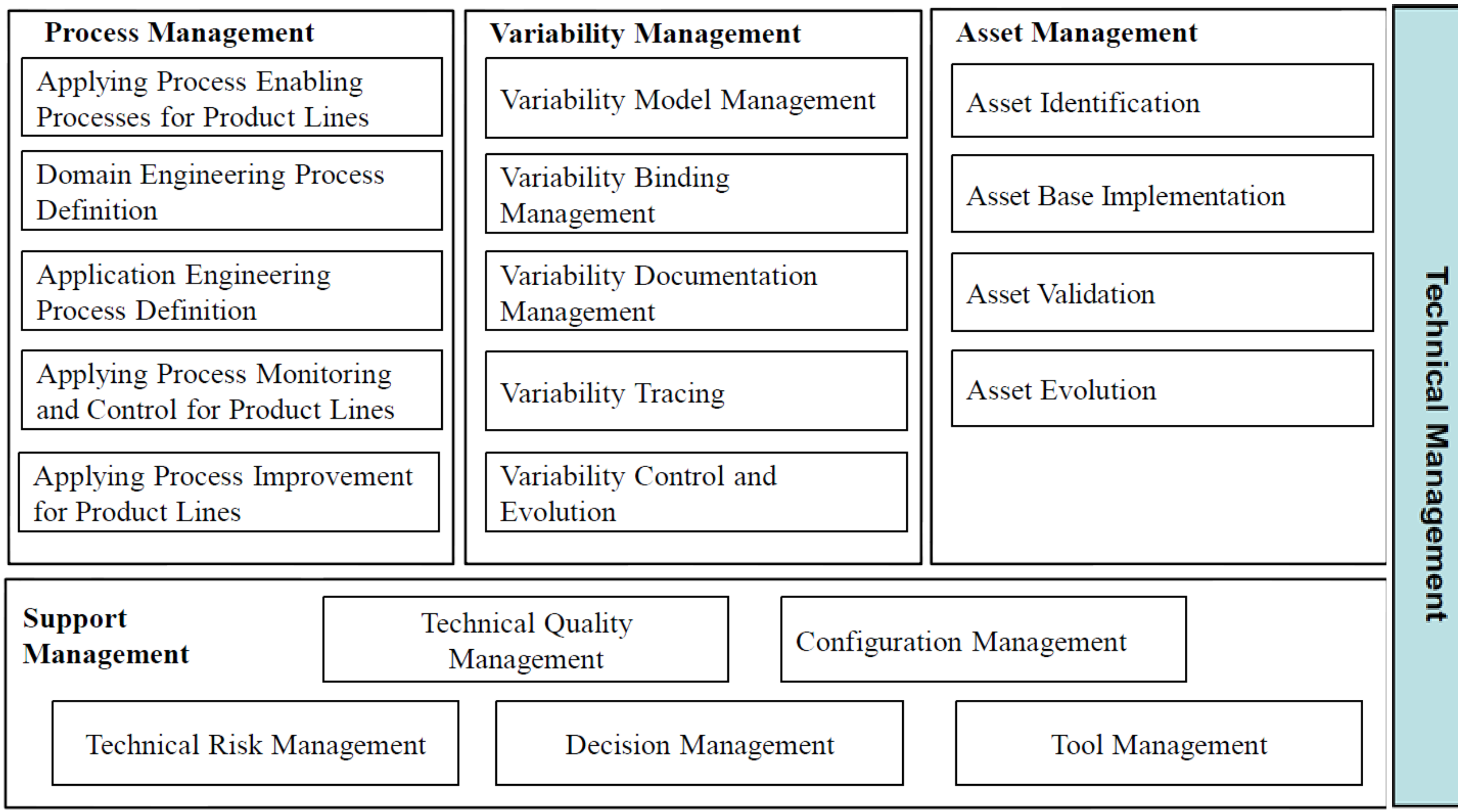
SMarty

- Princípios de SMarty:
  - Permitir realizar todas as atividades de Engenharia de Domínio em LPS de acordo com o Modelo de Referência da ISO/IEC 26550
    - Essencialmente Gerenciamento Técnico
  - Futuramente realizar as atividade de Engenharia de Aplicação
    - Hoje: parcialmente

SMarty
Modelo de Referência ISO/IEC 26550



SMarty
Gerenciamento Técnico ISO/IEC 26555



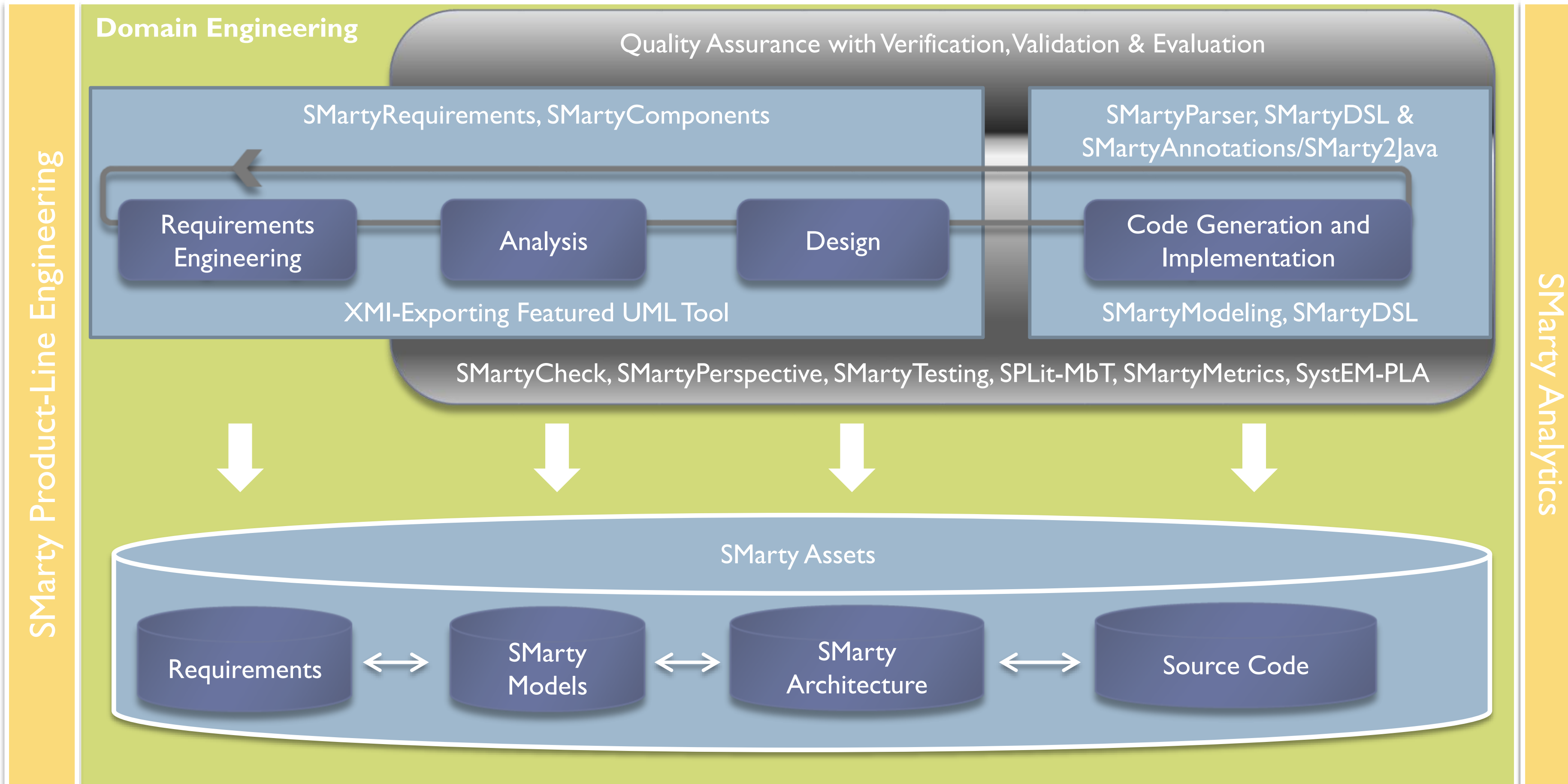
SMarty

- Gerenciamento Técnico:
  - Requisitos, Análise e Design:
    - SMartyRequirements [em desenv.]: engenharia de requisitos com auxílio de técnicas de PLN
    - SMartyProfile, SMartyProcess [+ Guidelines], SMartyComponents
  - Quality Assurance:
    - SMartyCheck, SMartyPerspective, SMartyTesting, SPLiT-MBT (casos de uso, atividades), SMartyMetrics (SMM/OMG, ISO/IEC 25010)

SMarty

- Gerenciamento Técnico:
  - Avaliação e Otimização de PLA
    - SystEM-PLA, MOA4PLA
  - Predictive SMarty [em desenv.]
    - SMartyRecommender, SMartyAnalytics
- SMarty Tools
  - SMartyModeling, SMartyDSL, SMartyAnalyzer, SMarty2Java, SMartyCheckTool

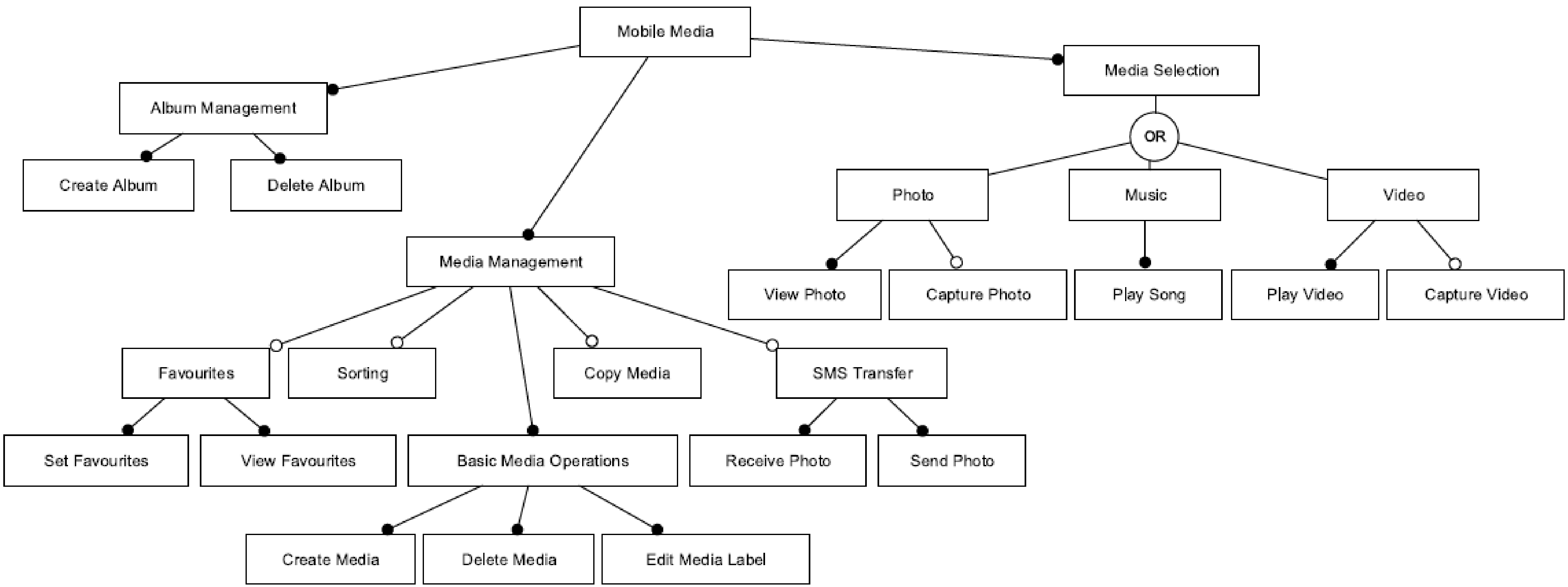
Engenharia de LPS com SMarty

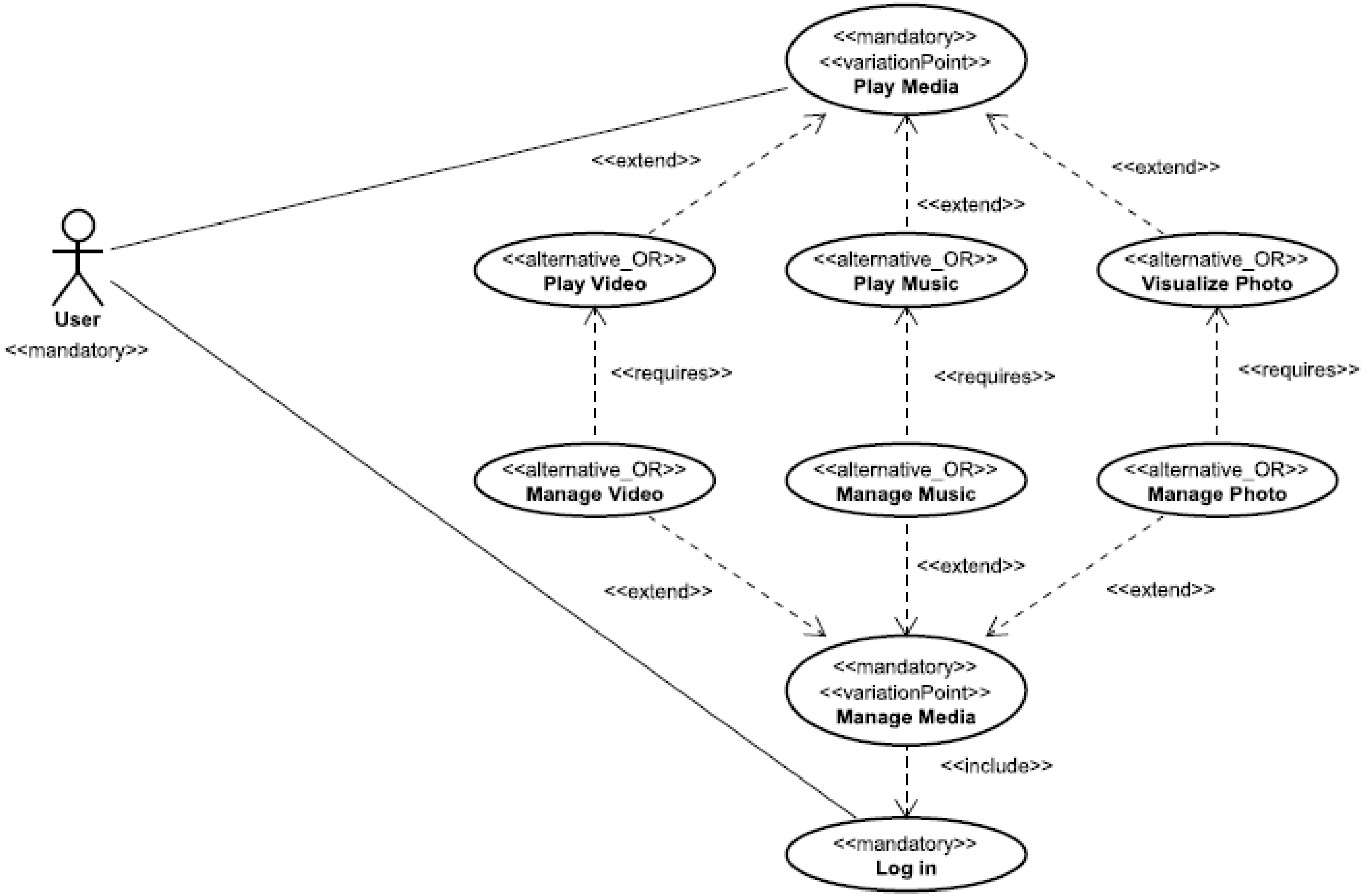


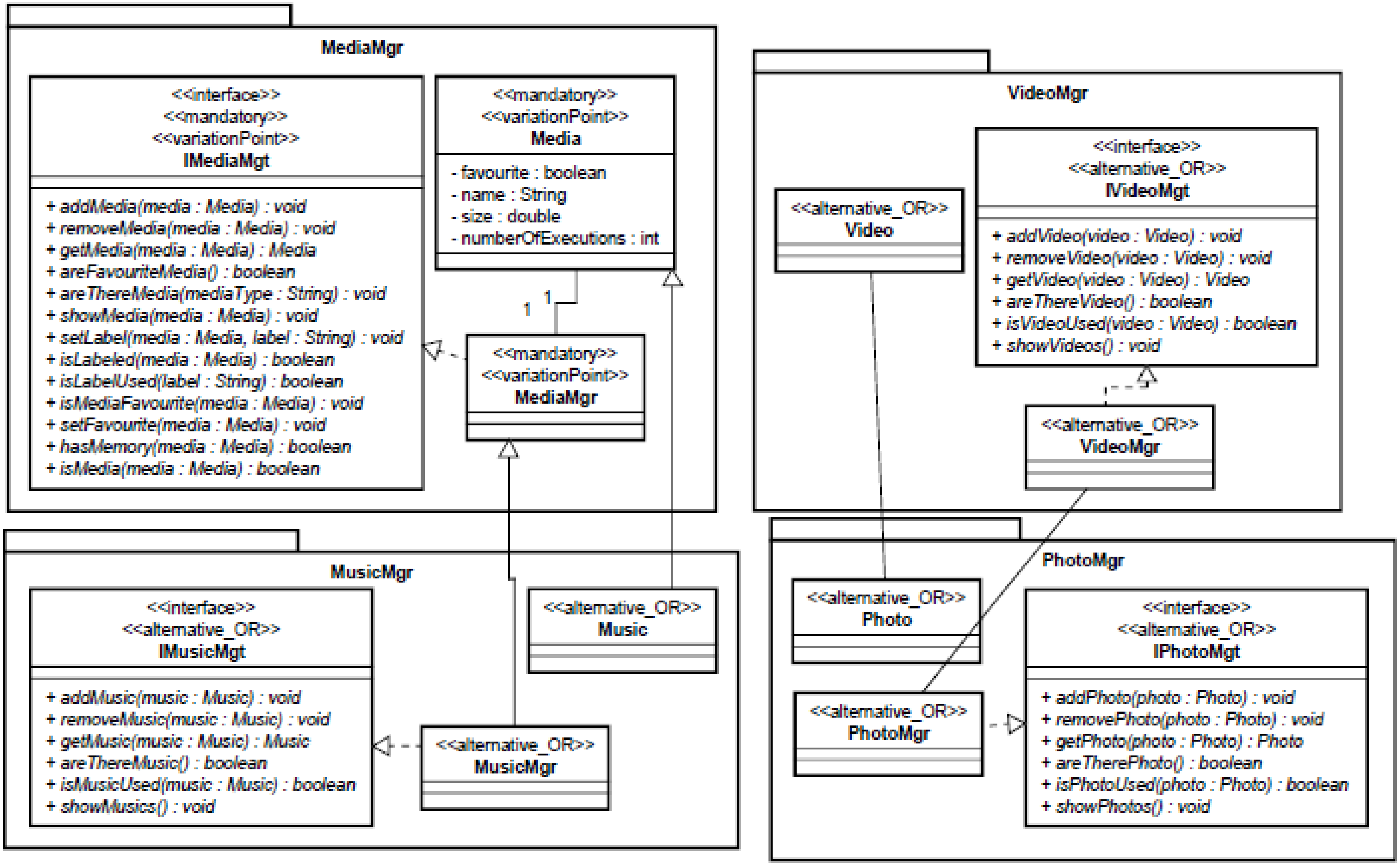
SMarty

- Análise e *Design*
  - SMartyProfile
    - Extensão do metamodelo da UML
    - Estereótipos próprios
    - Meta atributos
  - SMartyProcess [+ Guidelines]
    - Processo de gerenciamento de variabilidades
    - Diretrizes de identificação, representação e resolução de variabilidades
  - SMartyModeling
    - Ferramenta principal

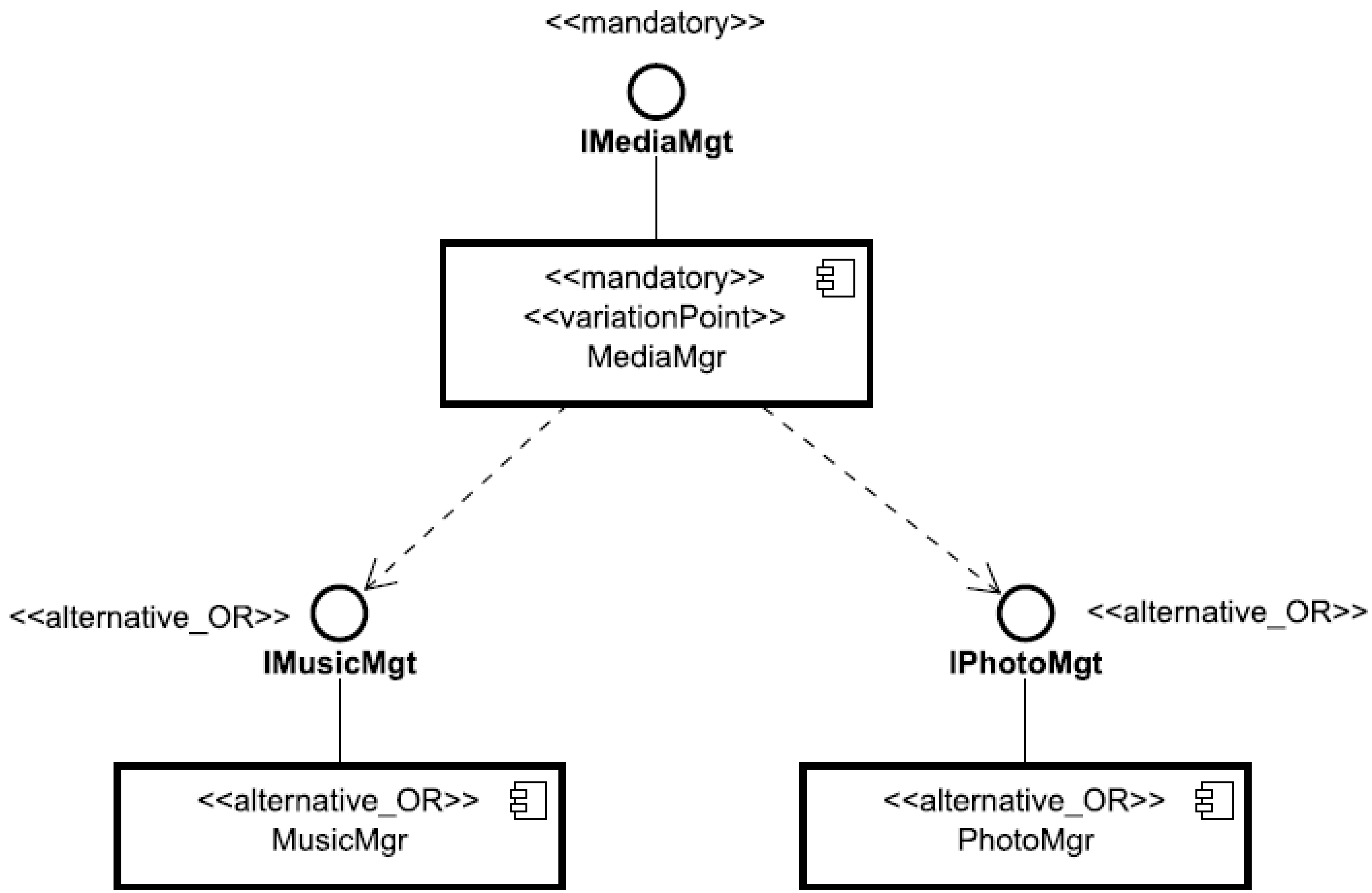
Running SMartyModeling...


Mobile Media
Album Management
Create Album
Delete Album
Media Management
Favourites
Sorting
Copy Media
SMS Transfer
Set Favourites
View Favourites
Basic Media Operations
Receive Photo
Send Photo
Create Media
Delete Media
Edit Media Label
Media Selection
OR
Photo
Music
Video
View Photo
Capture Photo
Play Song
Play Video
Capture Video

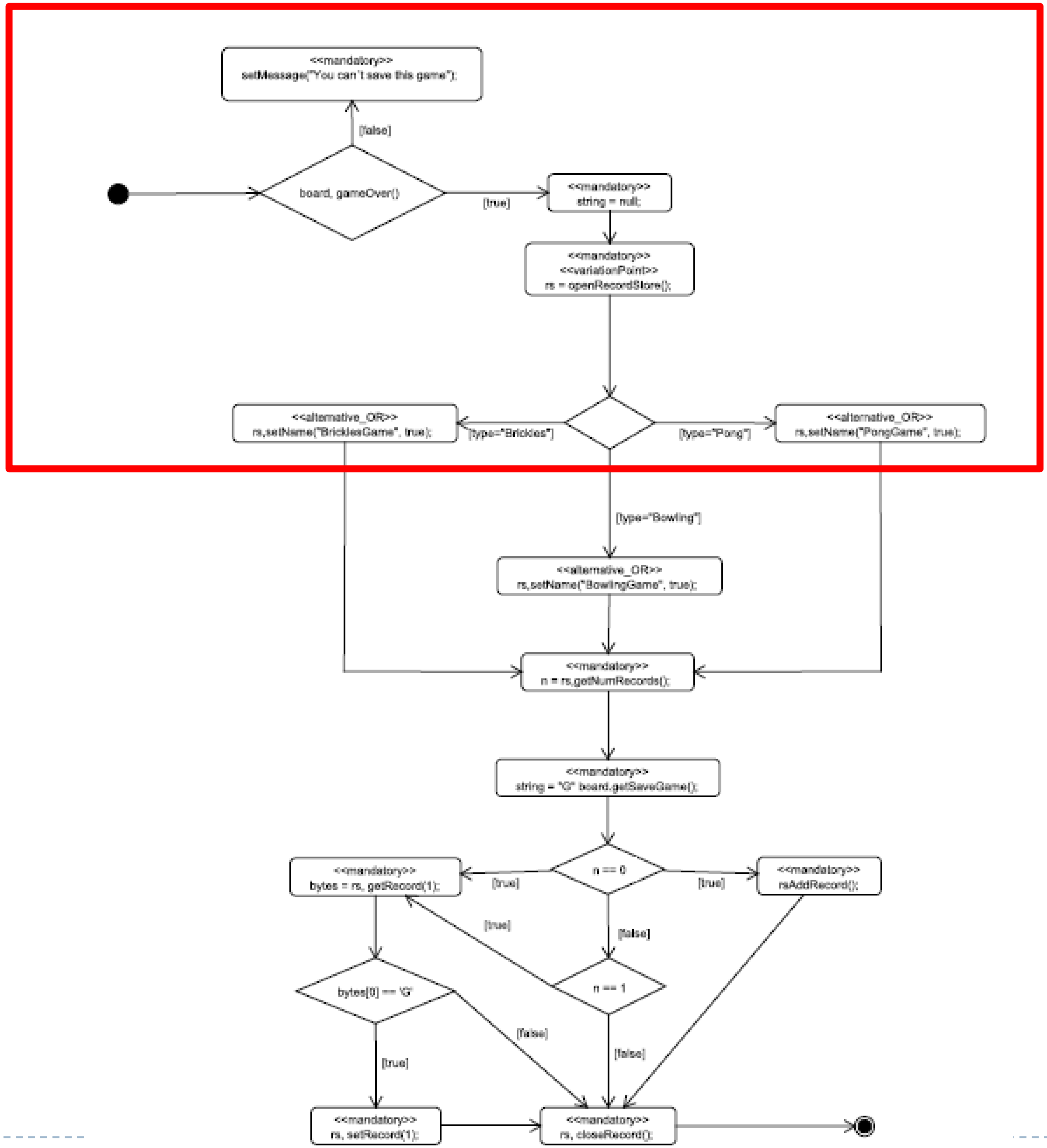

User
<<mandatory>>
<<mandatory>>
<<variationPoint>>
Play Media
<<extend>>
<<extend>>
<<extend>>
<<alternative_OR>>
Play Video
<<alternative_OR>>
Play Music
<<alternative_OR>>
Visualize Photo
<<requires>>
<<requires>>
<<requires>>
<<alternative_OR>>
Manage Video
<<alternative_OR>>
Manage Music
<<alternative_OR>>
Manage Photo
<<extend>>
<<extend>>
<<extend>>
<<mandatory>>
<<variationPoint>>
Manage Media
<<include>>
<<mandatory>>
Log in

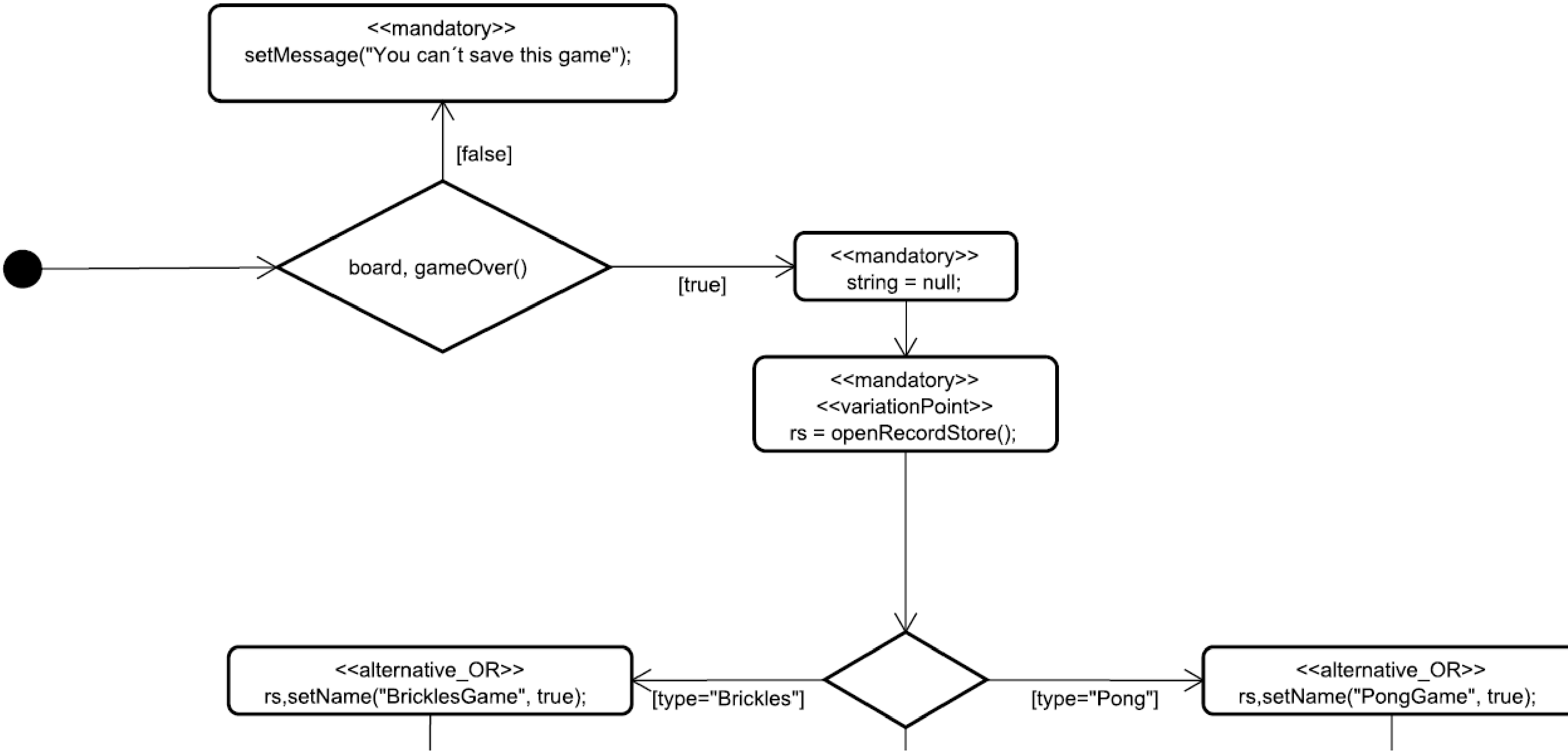


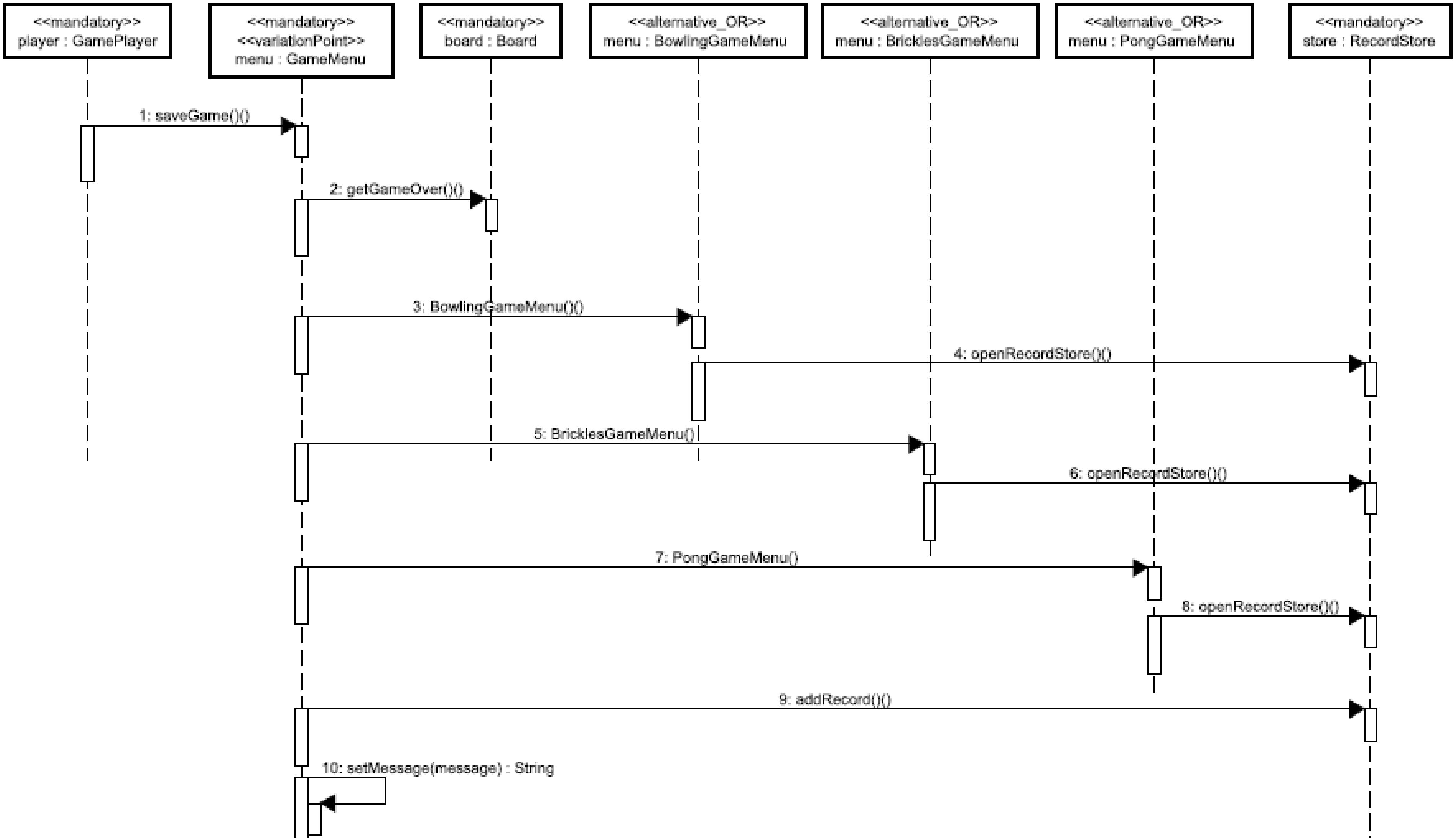
Prof. Edson OliveiraJr (State University of Maringá) - edson@din.uem.br



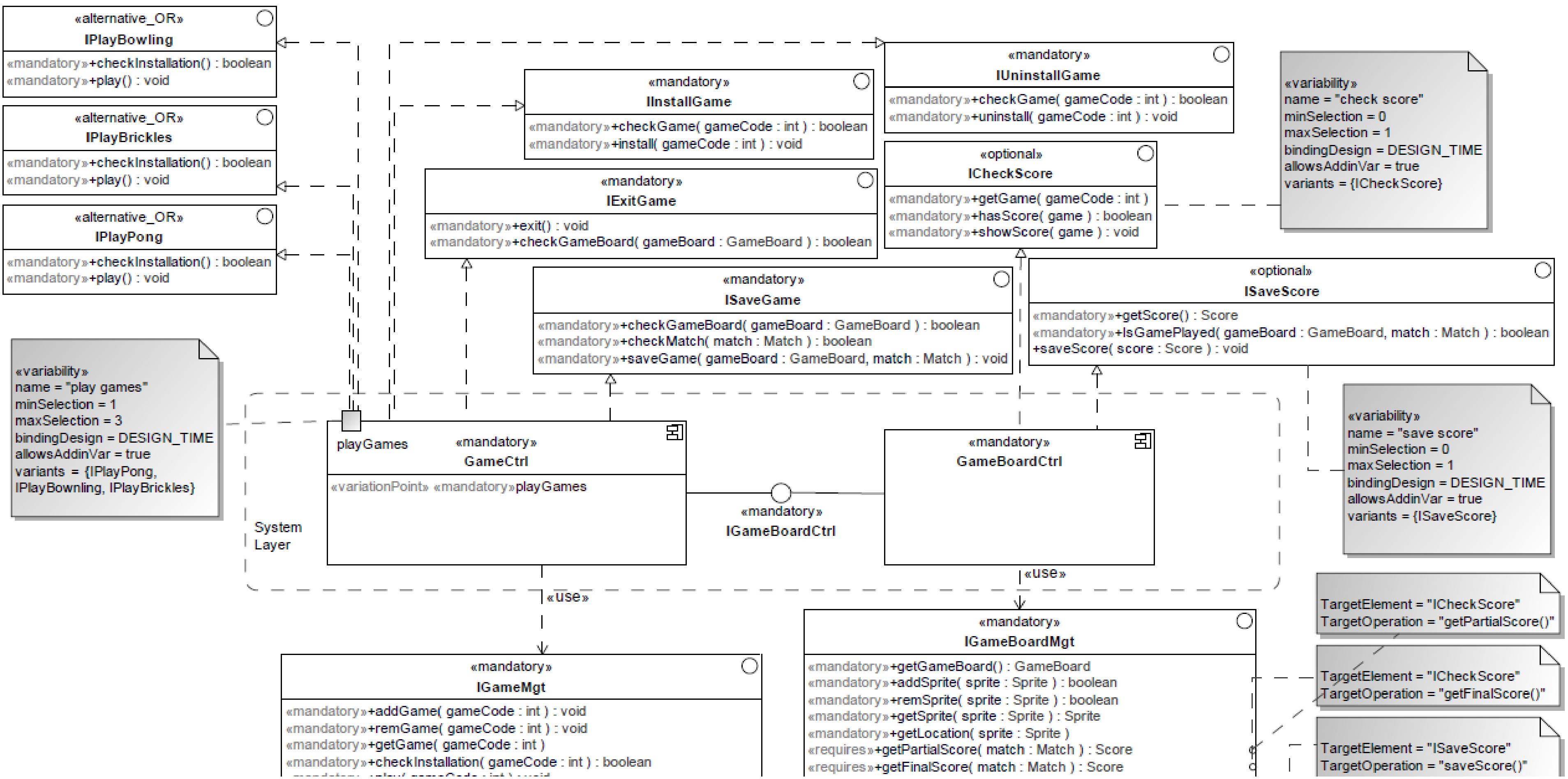
Prof. Edson OliveiraJr (State University of Maringá) - edson@din.uem.br


<<mandatory>>
setMessage("You can't save this game");
[false]
board, gameOver()
[true]
<<mandatory>>
string = null;
<<mandatory>>
<<variationPoint>>
rs = openRecordStore();
<<alternative_OR>>
rs.setName("BricklesGame", true);
[type="Brickles"]
[type="Pong"]
<<alternative_OR>>
rs.setName("PongGame", true);
[type="Bowling"]
<<alternative_OR>>
rs.setName("BowlingGame", true);
<<mandatory>>
n = rs.getNumRecords();
<<mandatory>>
string = "G" board.getSaveGame();
n == 0
[true]
[true]
<<mandatory>>
bytes = rs. getRecord(1);
<<mandatory>>
rsAddRecord();
[true]
[false]
n == 1
bytes[0] == 'G'
[false]
[false]
[true]
<<mandatory>>
rs, setRecord(1);
<<mandatory>>
rs, closeRecord();


<<mandatory>>
setMessage("You can´t save this game");
[false]
board, gameOver()
[true]
<<mandatory>>
string = null;
<<mandatory>>
<<variationPoint>>
rs = openRecordStore();
<<alternative_OR>>
rs,setName("BricklesGame", true);
[type="Brickles"]
[type="Pong"]
<<alternative_OR>>
rs,setName("PongGame", true);



<<mandatory>>
player : GamePlayer
<<mandatory>>
<<variationPoint>>
menu : GameMenu
<<mandatory>>
board : Board
<<alternative_OR>>
menu : BowlingGameMenu
<<alternative_OR>>
menu : BricklesGameMenu
<<alternative_OR>>
menu : PongGameMenu
<<mandatory>>
store : RecordStore
1: saveGame()()
2: getGameOver()()
3: BowlingGameMenu()()
4: openRecordStore()()
5: BricklesGameMenu()
6: openRecordStore()()
7: PongGameMenu()
8: openRecordStore()()
9: addRecord()()
10: setMessage(message) : String


«alternative_OR»
IPlayBowling
«mandatory»+checkInstallation() : boolean
«mandatory»+play() : void
«alternative_OR»
IPlayBrickles
«mandatory»+checkInstallation() : boolean
«mandatory»+play() : void
«alternative_OR»
IPlayPong
«mandatory»+checkInstallation() : boolean
«mandatory»+play() : void
«mandatory»
IInstallGame
«mandatory»+checkGame(gameCode : int) : boolean
«mandatory»+install(gameCode : int) : void
«mandatory»
IUninstallGame
«mandatory»+checkGame(gameCode : int) : boolean
«mandatory»+uninstall(gameCode : int) : void
«variability»
name = "check score"
minSelection = 0
maxSelection = 1
bindingDesign = DESIGN_TIME
allowsAddinVar = true
variants = {ICheckScore}
«mandatory»
IExitGame
«mandatory»+exit() : void
«mandatory»+checkGameBoard(gameBoard : GameBoard) : boolean
«optional»
ICheckScore
«mandatory»+getGame(gameCode : int)
«mandatory»+hasScore(game) : boolean
«mandatory»+showScore(game) : void
«mandatory»
ISaveGame
«mandatory»+checkGameBoard(gameBoard : GameBoard) : boolean
«mandatory»+checkMatch(match : Match) : boolean
«mandatory»+saveGame(gameBoard : GameBoard, match : Match) : void
«optional»
ISaveScore
«mandatory»+getScore() : Score
«mandatory»+IsGamePlayed(gameBoard : GameBoard, match : Match) : boolean
+saveScore(score : Score) : void
«variability»
name = "play games"
minSelection = 1
maxSelection = 3
bindingDesign = DESIGN_TIME
allowsAddinVar = true
variants = {IPlayPong,
IPlayBownling, IPlayBrickles}
playGames
«mandatory»
GameCtrl
«variationPoint» «mandatory»playGames
«mandatory»
GameBoardCtrl
«mandatory»
IGameBoardCtrl
System
Layer
«variability»
name = "save score"
minSelection = 0
maxSelection = 1
bindingDesign = DESIGN_TIME
allowsAddinVar = true
variants = {ISaveScore}
«use»
«use»
TargetElement = "ICheckScore"
TargetOperation = "getPartialScore()"
«mandatory»
IGameBoardMgt
«mandatory»+getGameBoard() : GameBoard
«mandatory»+addSprite(sprite : Sprite) : boolean
«mandatory»+remSprite(sprite : Sprite) : boolean
«mandatory»+getSprite(sprite : Sprite) : Sprite
«mandatory»+getLocation(sprite : Sprite)
«requires»+getPartialScore(match : Match) : Score
«requires»+getFinalScore(match : Match) : Score
TargetElement = "ICheckScore"
TargetOperation = "getFinalScore()"
«mandatory»
IGameMgt
«mandatory»+addGame(gameCode : int) : void
«mandatory»+remGame(gameCode : int) : void
«mandatory»+getGame(gameCode : int)
«mandatory»+checkInstallation(gameCode : int) : boolean
TargetElement = "ISaveScore"
TargetOperation = "saveScore()"

E agora???

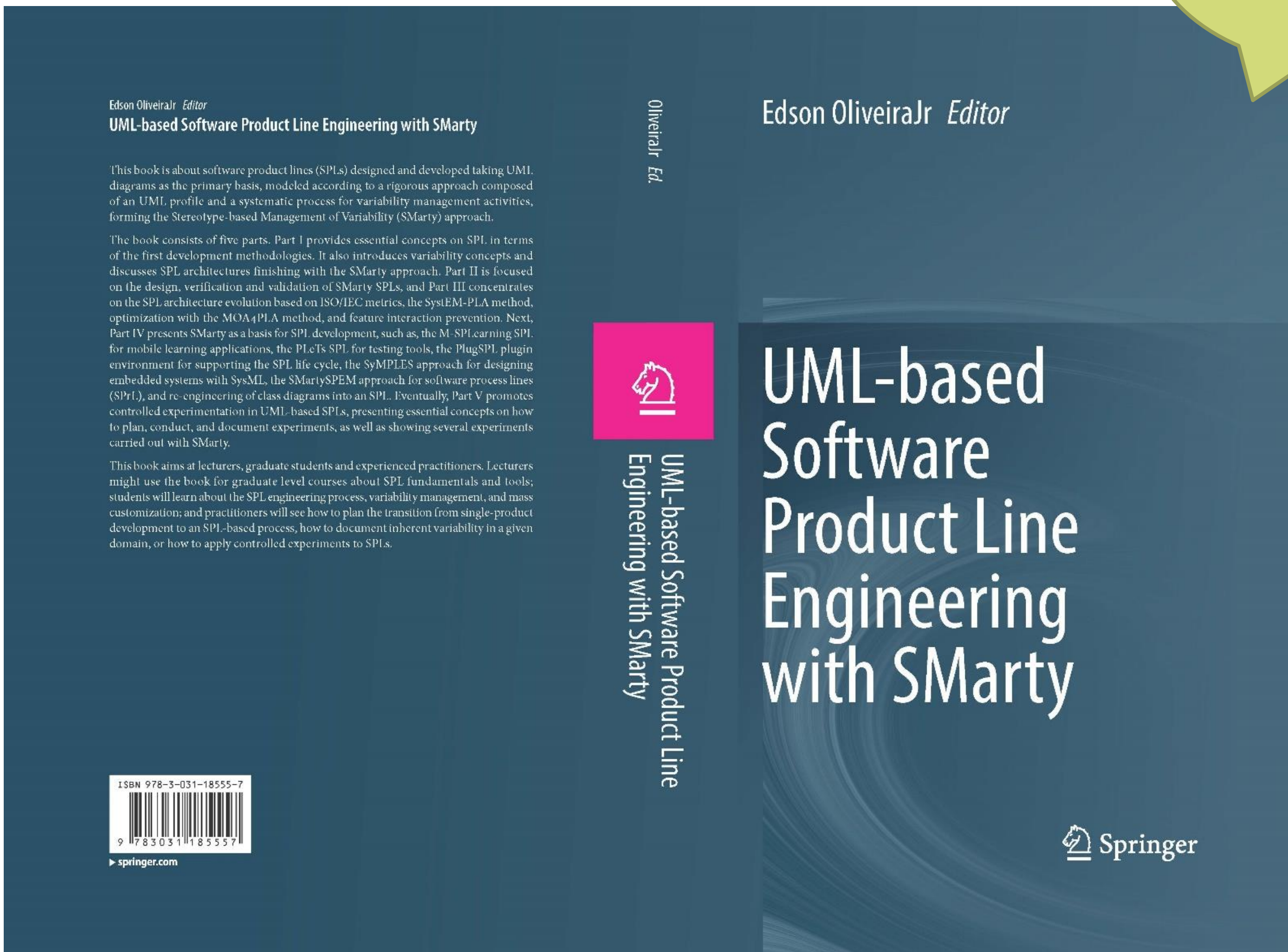
- Inspeção
  - SMartyCheck
  - SMartyPerspective
- Teste baseado em modelos
  - SMartyTesting
  - SPLiT-MBt
- Avaliação
  - SMartyMetrics – ISO 25010
  - SystEM-PLA
- Otimização
  - MOA4PLA – algoritmos genéticos e search-based
- Redução de *feature interaction*
  - Casos de uso, classes
- Impor/export
  - SMartyAnnotation
  - FeatureIDE
- Avaliação de features
  - DyMMER – UFC
- SMartyRefactoring
- SMartyGit
  - Project tracing
  - Round-trip SPL engineering
- SMartyEvidencing
  - recomendação/predição
- SMartyDSL???
- SMartyOpenness!!!
  - Open Science

Como aprender e usar “tudo” isso???

UML-based Software Product Line Engineering with SMarty

[https://link.springer.com/book/9783031185557]

Lançamento: Dez/2022



Prof. Edson OliveiraJr (State University of Maringá) - edson@din.uem.br

Springer

springer.com

Special offer / Get 20% off the printed book or eBook!

Enter the following coupon code at checkout on link.springer.com to apply discount

Y13xzwSS4MXM2F / Valid Dec 19, 2022 – Jan 16, 2023

The book is scheduled to be published in December, and can be ordered only then. This is why the token is only valid from Dec 19 onwards.

E. OliveiraJr

UML-based Software Product Line Engineering with SMarty

- Introduces SMarty, a UML-based systematic approach for developing software product lines

Prof. Edson OliveiraJr (State University of Maringá) - edson@din.uem.br

Research Group on Systematic Software Reuse and Continuous Experimentation

Computer Science Investigation in Digital Forensics

Obrigado!!!
Thank you!!!

Prof. Dr. Edson OliveiraJr

Informatics Department

State University of Maringá, Brazil

edson@din.uem.br

Universidade Estadual de Maringá

Apoio ao Desenvolvimento Científico e Tecnológico do Paraná