

# *Software Product Line Engineering with SMarty*

**Prof. Dr. Edson Oliveira Jr**

Informatics Department

State University of Maringá, Brazil

[edson@din.uem.br](mailto:edson@din.uem.br)

# Who am I?

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



## ▶ Dr. Edson Oliveira Jr

- ▶ 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

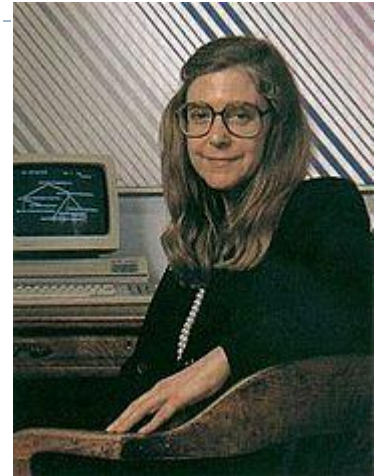


# Engenharia de Software

---

- ▶ 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 (I)?
  - ▶ “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 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

Copyright ©1990 by

The Institute of Electrical and Electronics Engineers  
345 East 47th Street, New York, NY 10017, USA

*No part of this document may be reproduced in any form,  
in an electronic retrieval system or otherwise,  
without the prior written permission of the publisher.*

# Qualidade de Software

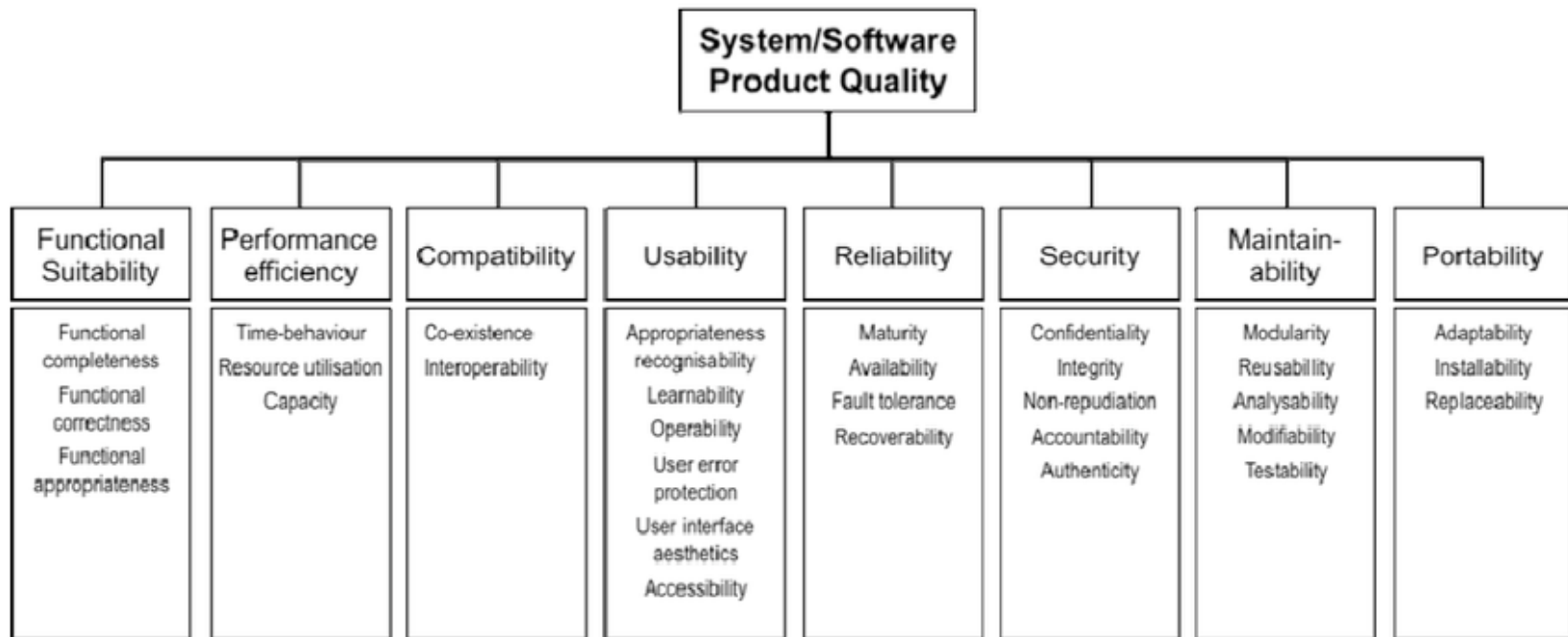
---

- ▶ **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

---

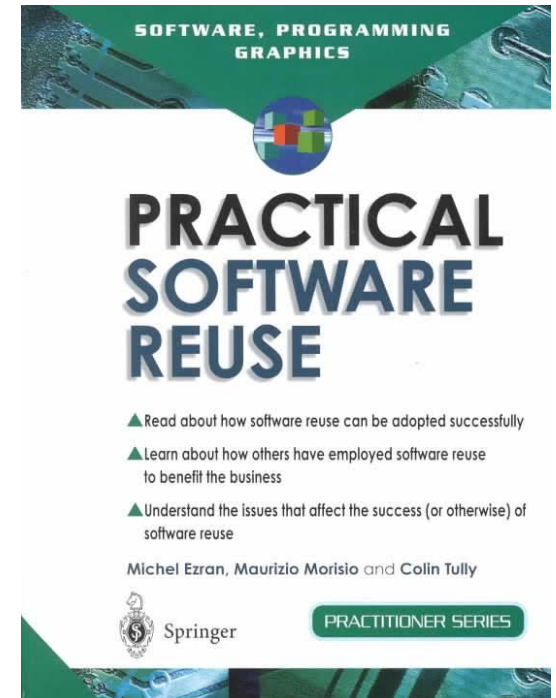
- ▶ Como conseguir qualidade em algo abstrato como software?
  - ▶ Não existe uma receita de bolo para isso!!
  - ▶ Uma possibilidade é a combinação de várias técnicas....
  - ▶ Dentre elas....
- ▶ Reuso de Software!!!



# 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

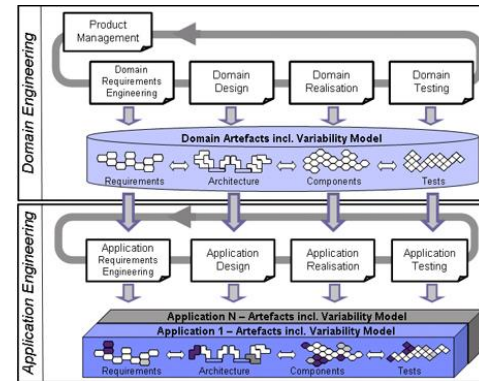
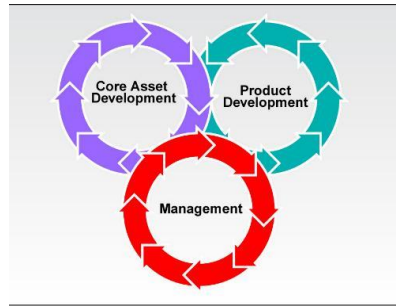
[www.ic.unicamp.br/~eliane/Cursos/Transparencias/Manutencao/reuso.ppt](http://www.ic.unicamp.br/~eliane/Cursos/Transparencias/Manutencao/reuso.ppt)

[www.les.inf.puc-rio.br/wiki/images/2/28/Aula06\\_software\\_product\\_lines\\_v2.ppt](http://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 **Engenharia** 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

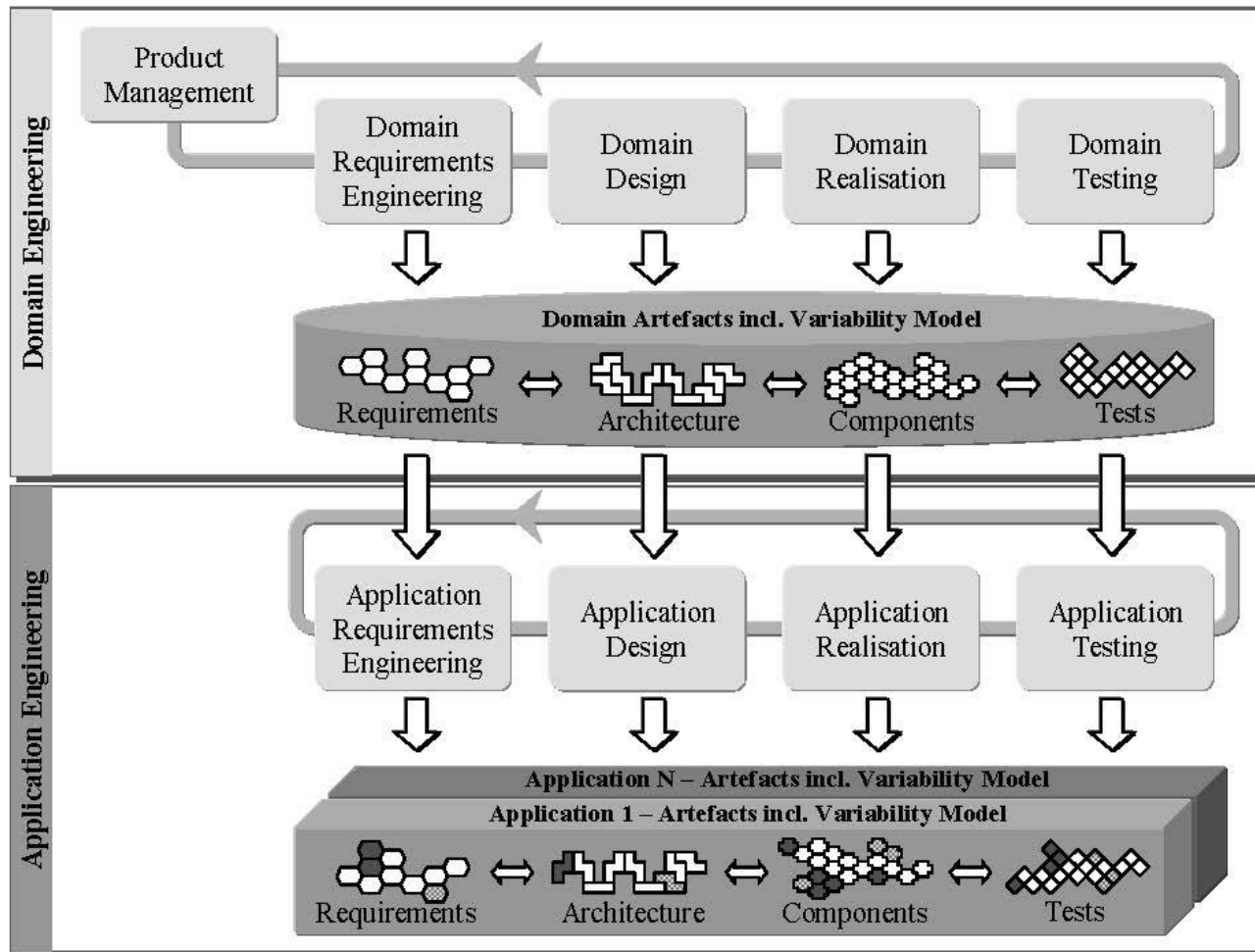
---

## ▶ 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

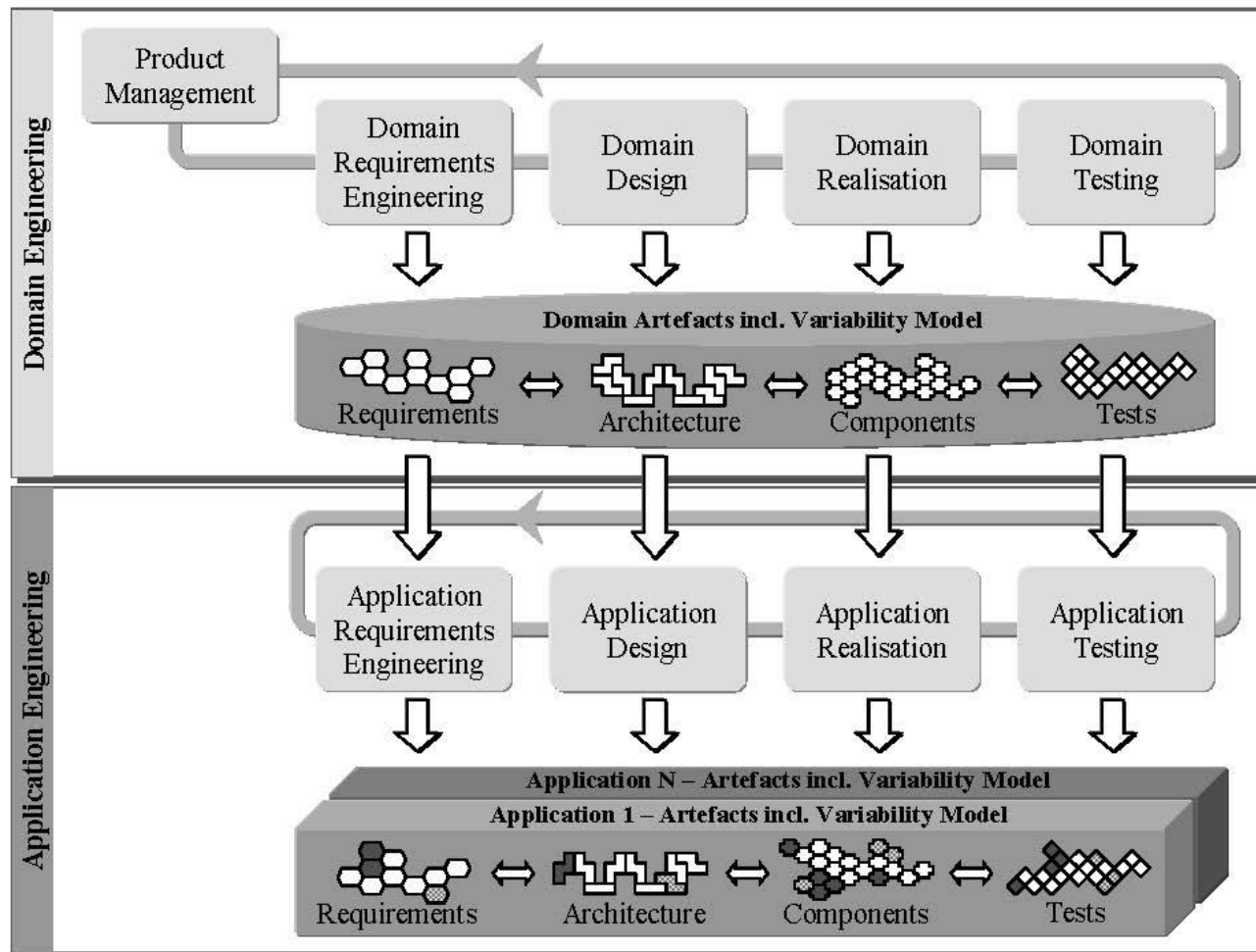
---

## ▶ 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

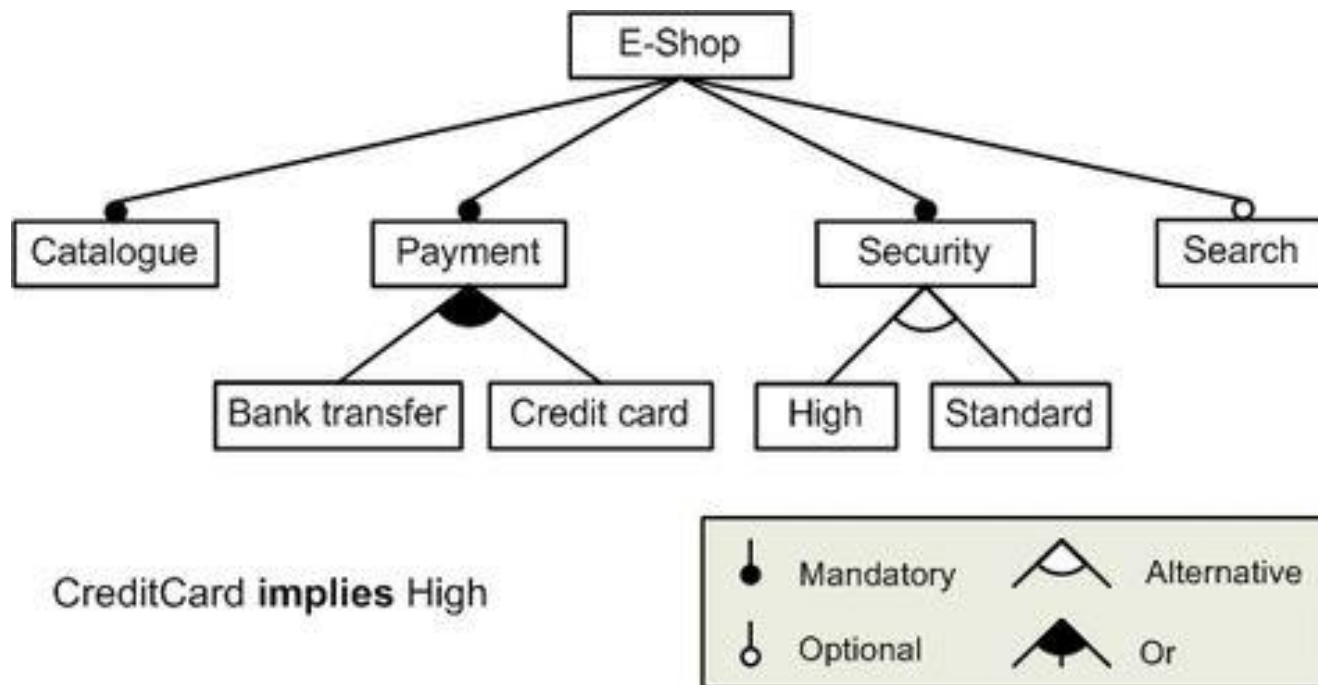
---

- ▶ 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

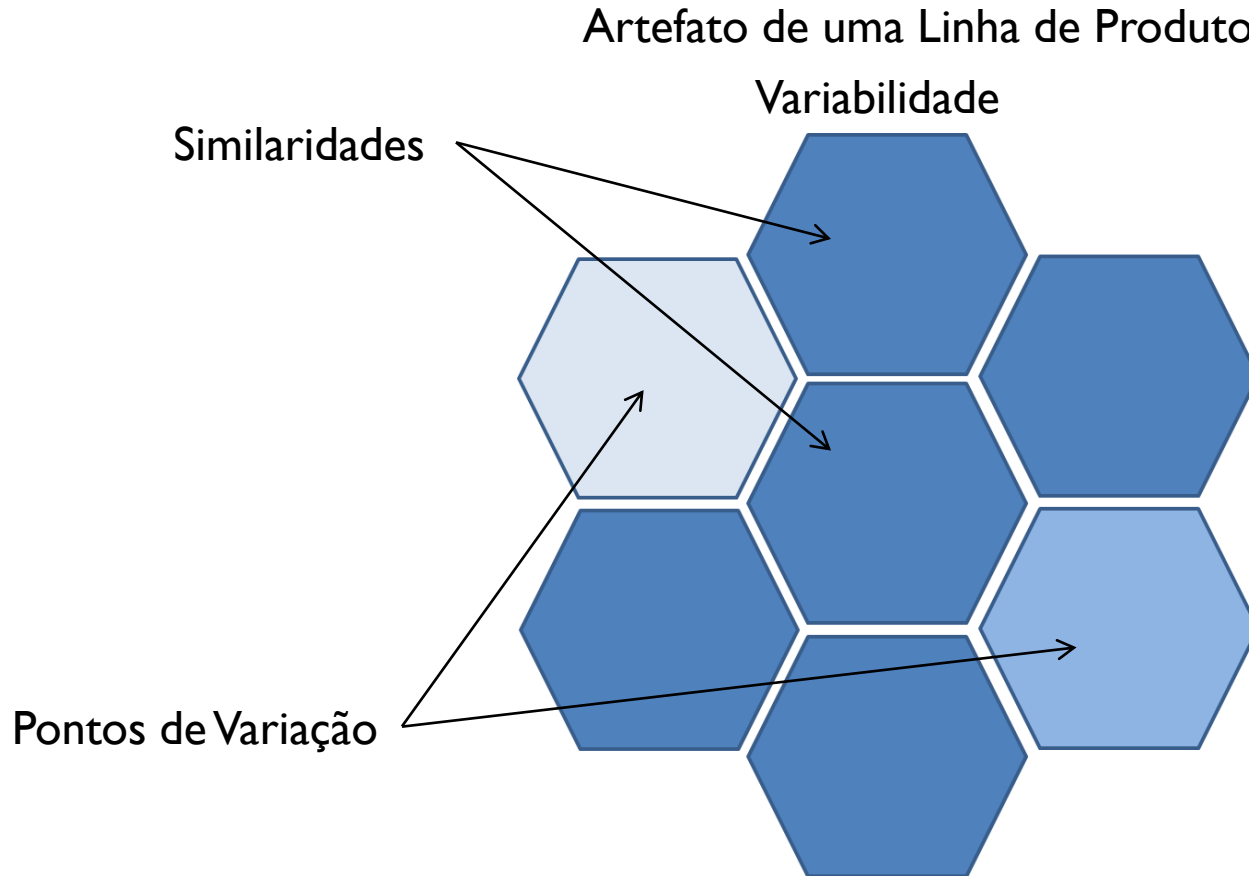
---

- ▶ 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

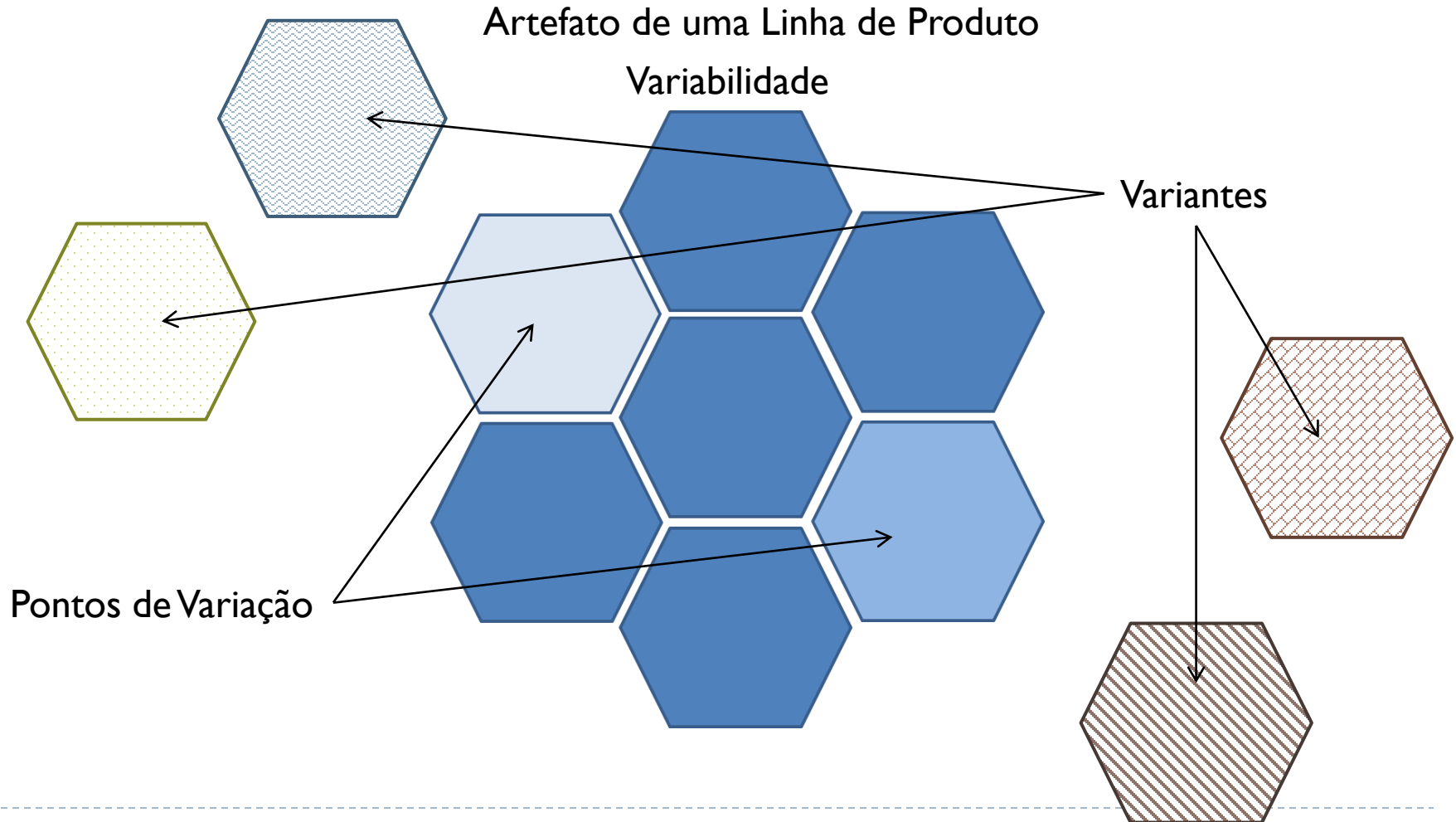


# Variabilidade em LPS

---

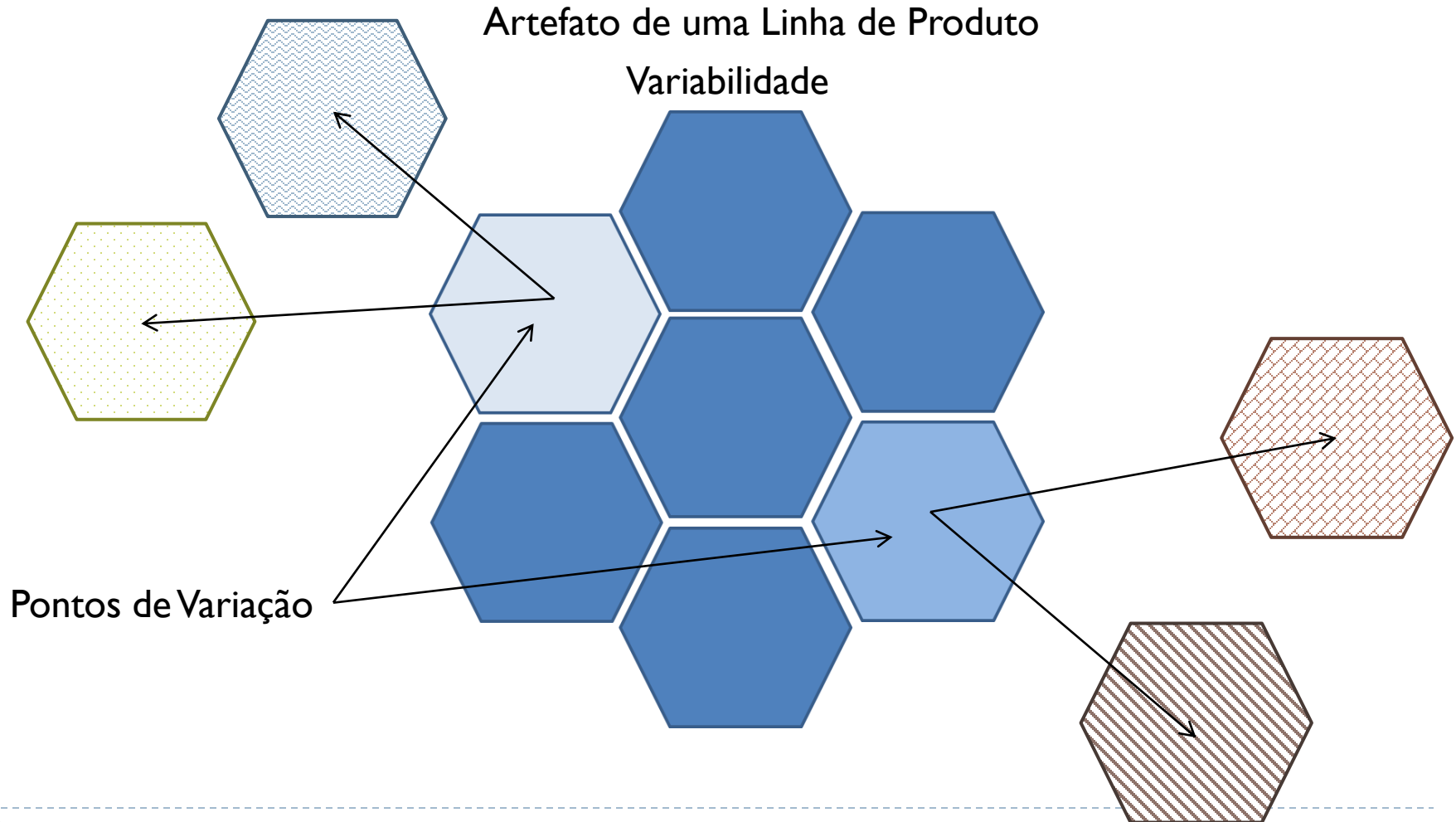


# Variabilidade em LPS

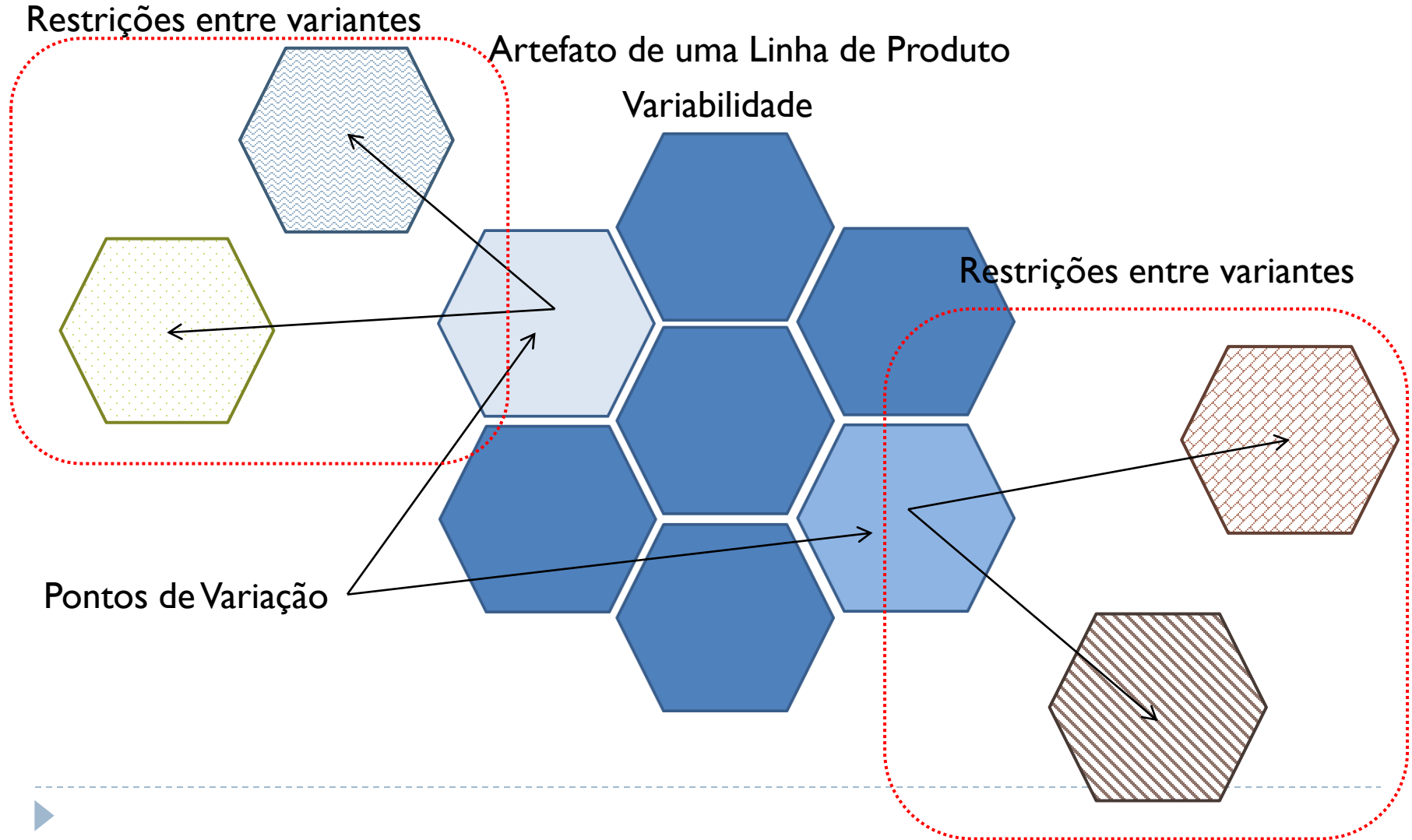


# Variabilidade em LPS

---

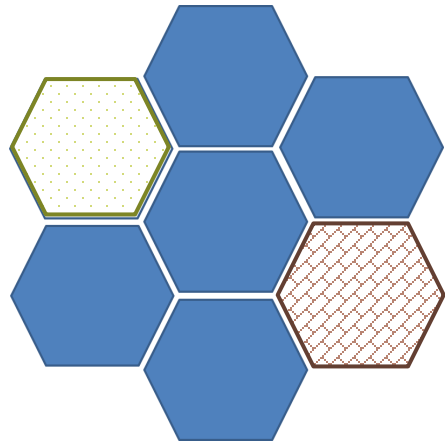


# Variabilidade em LPS

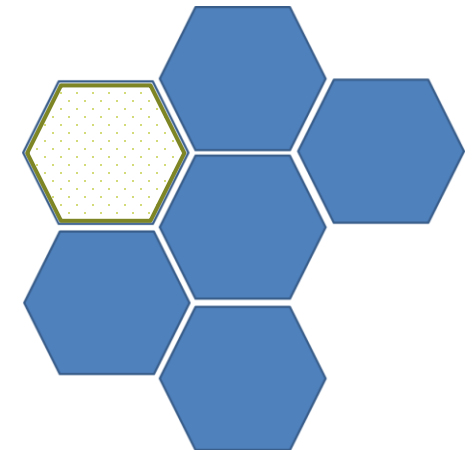
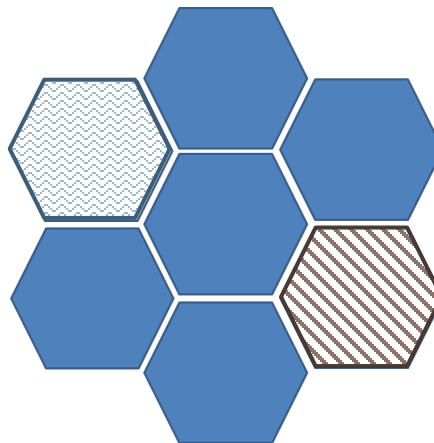
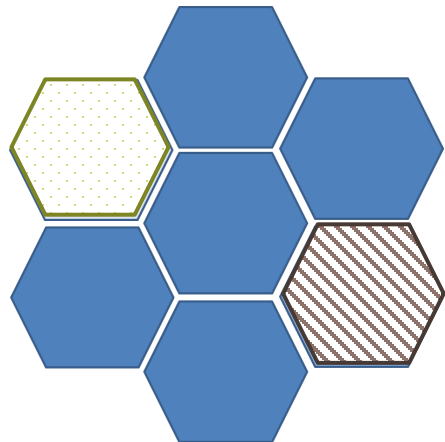
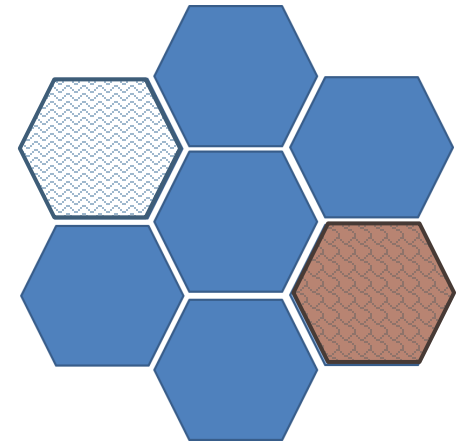


# Variabilidade em LPS

---



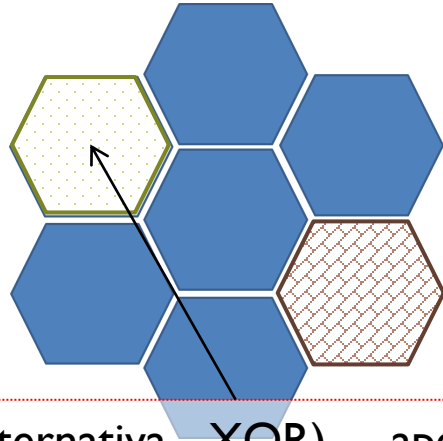
Possíveis produtos da LPS



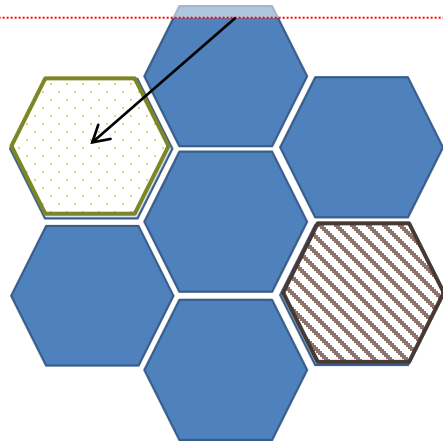


# Variabilidade em LPS

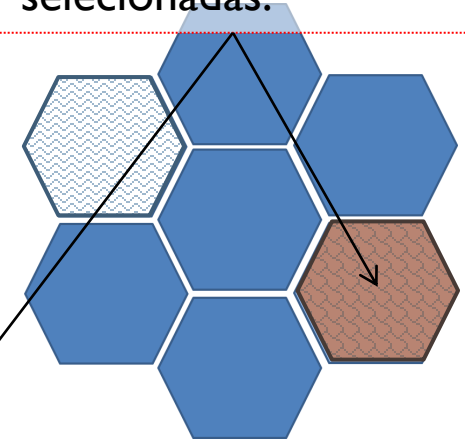
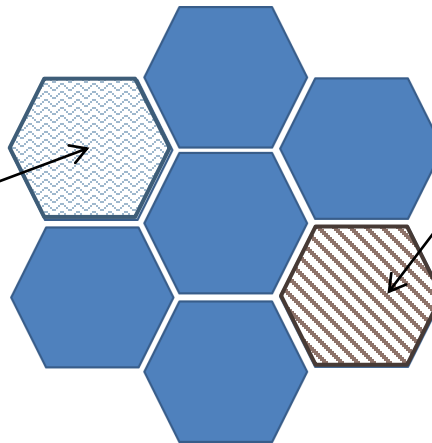
Ou – uma ou mais podem ser selecionadas.



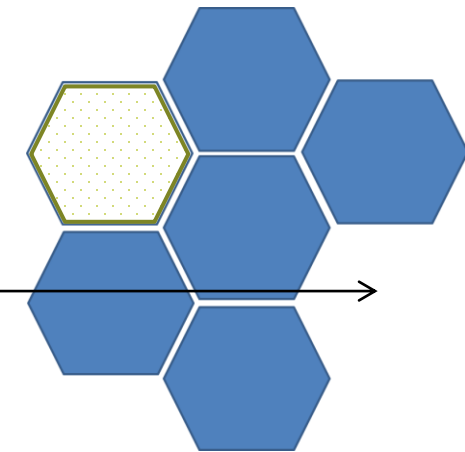
(alternativa - XOR) – apenas uma delas deve ser selecionada.



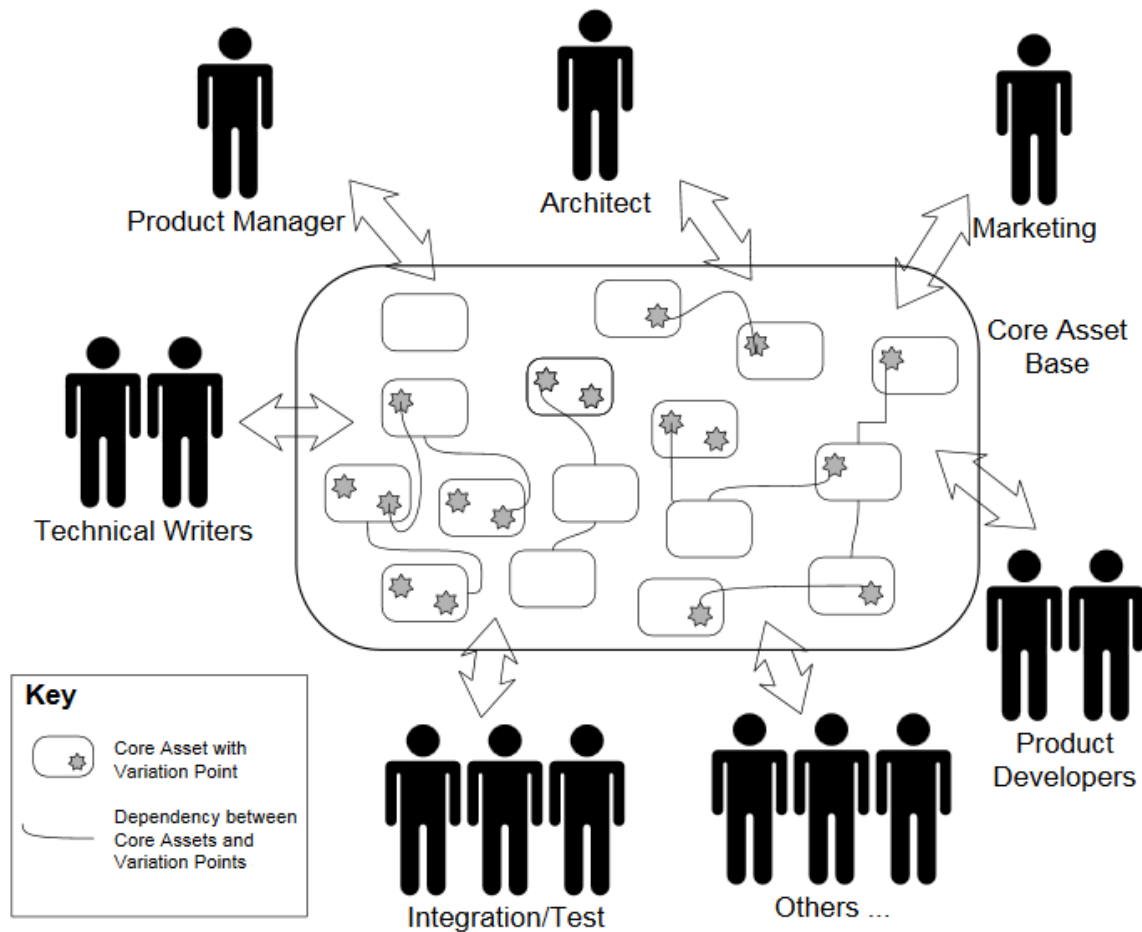
Possíveis produtos da LPS



Opcionais – podem ser selecionadas ou não.



# Variabilidade em LPS



[https://resources.sei.cmu.edu/asset\\_files/TechnicalReport/2005\\_005\\_001\\_14600.pdf](https://resources.sei.cmu.edu/asset_files/TechnicalReport/2005_005_001_14600.pdf)



# A Abordagem SMarty

# SMarty

---

- ▶ SMarty significa:
  - ▶ Stereotype-based Management of Variability
- ▶ 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

---

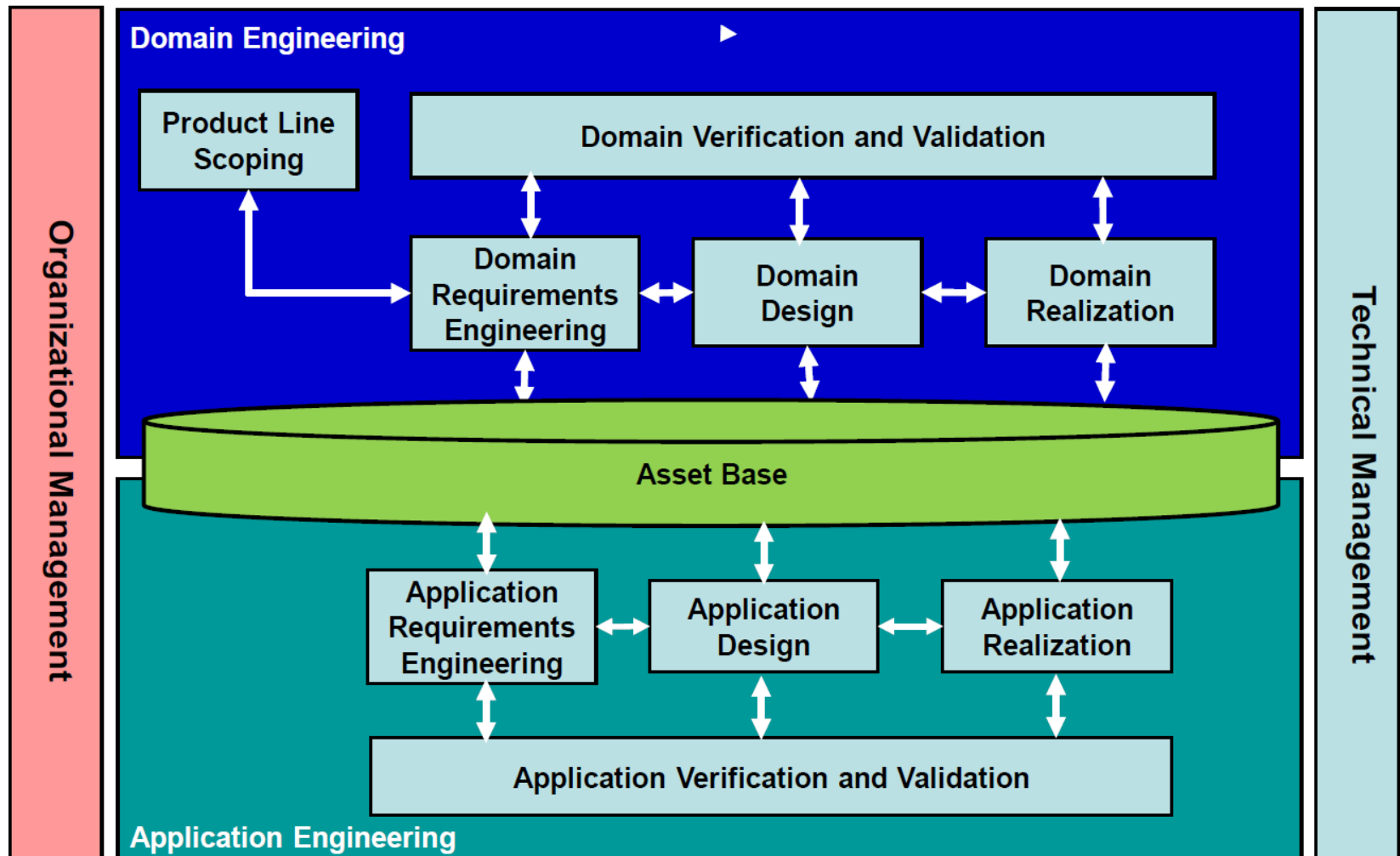
- ▶ **Princípios de SMarty:**

- ▶ 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



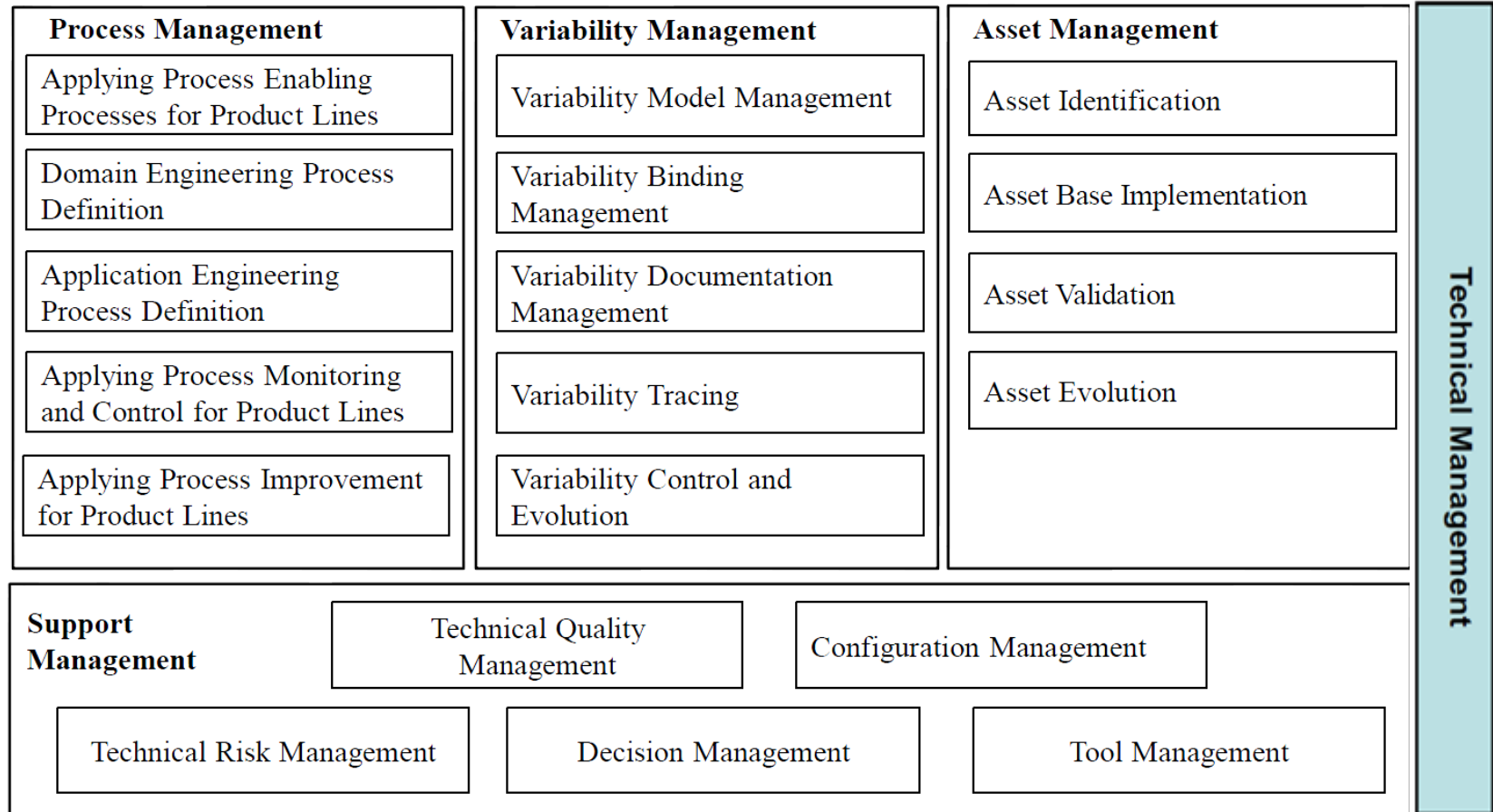
# SMarty

## 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.]

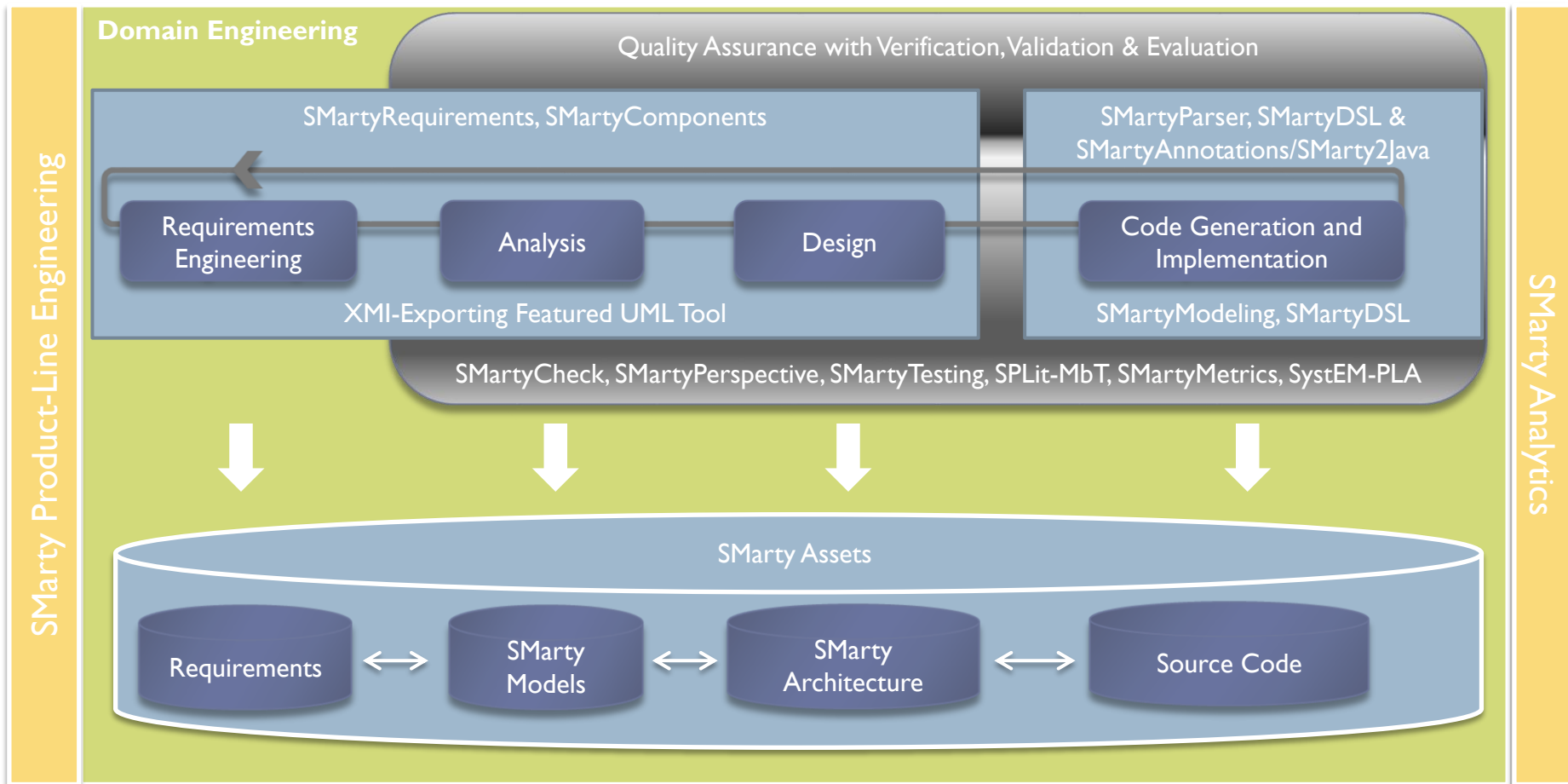
- ▶ SMartyRecommender, SMartyAnalytics

- ▶ 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

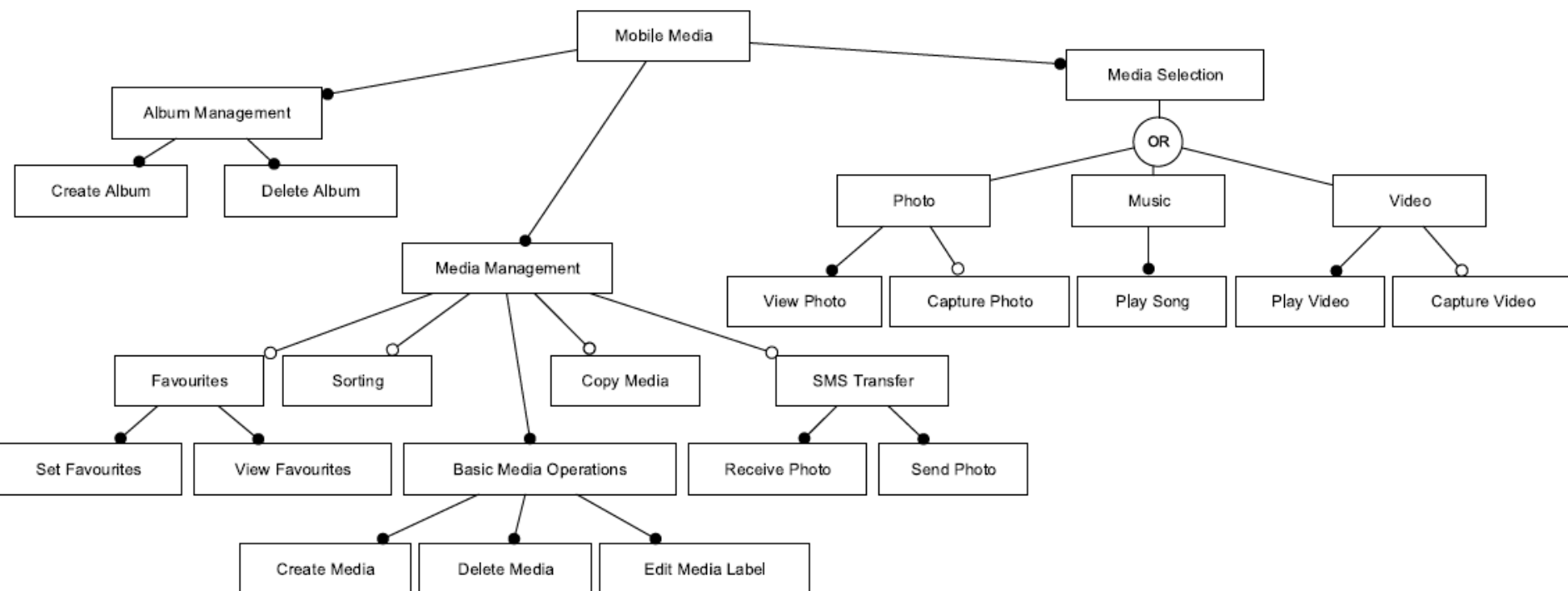
- ▶ SMartyModeling

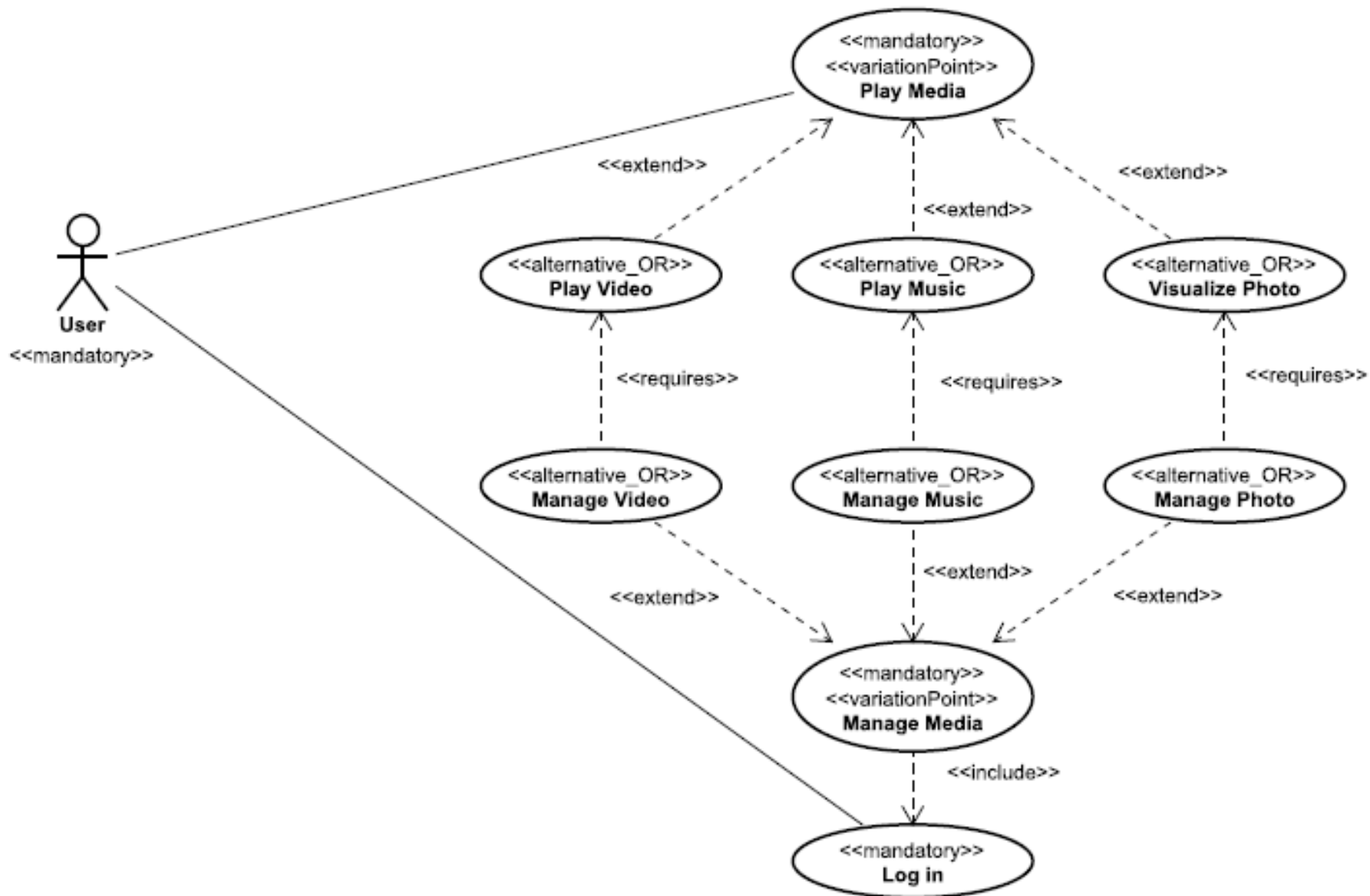
- ▶ Ferramenta principal

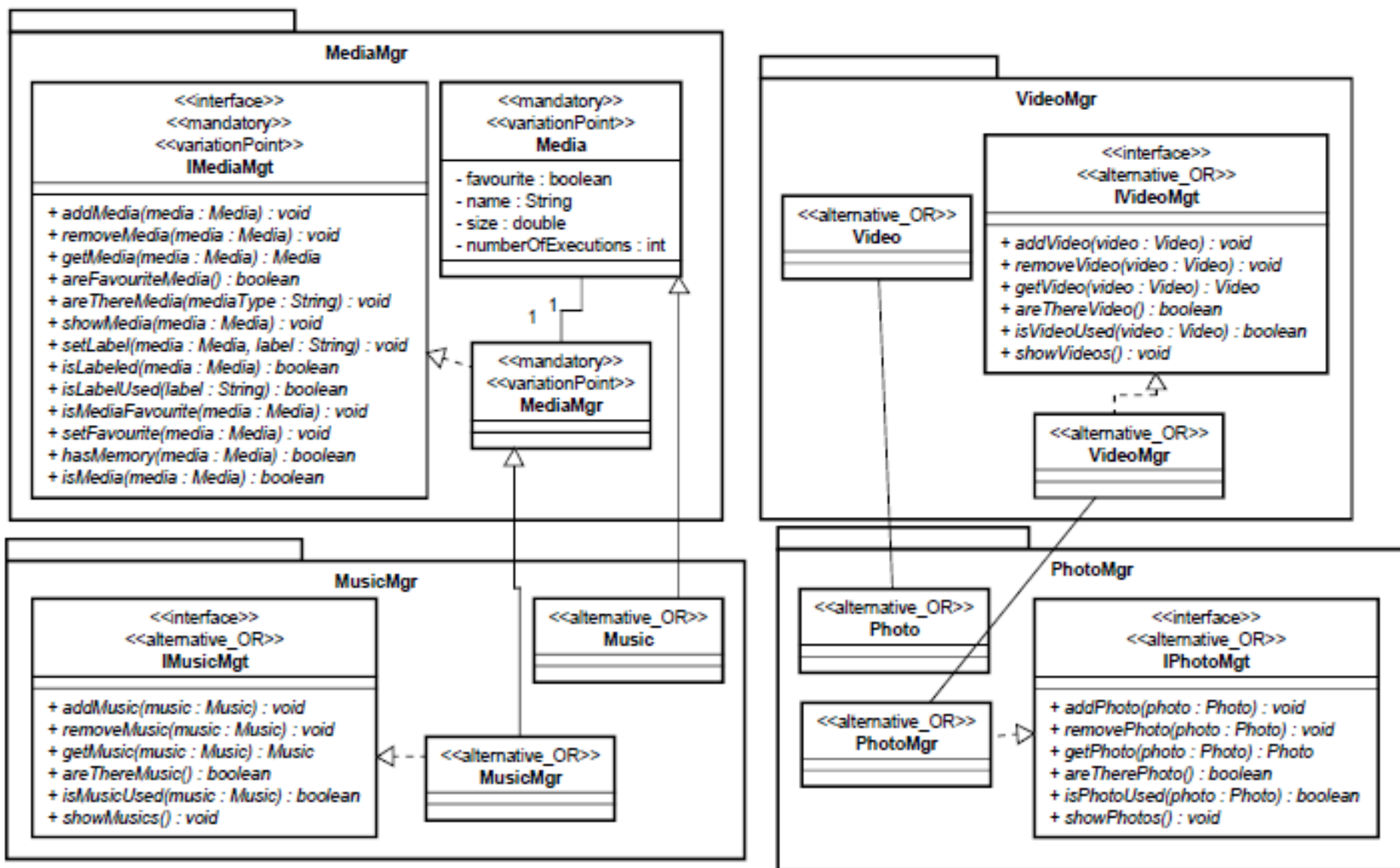


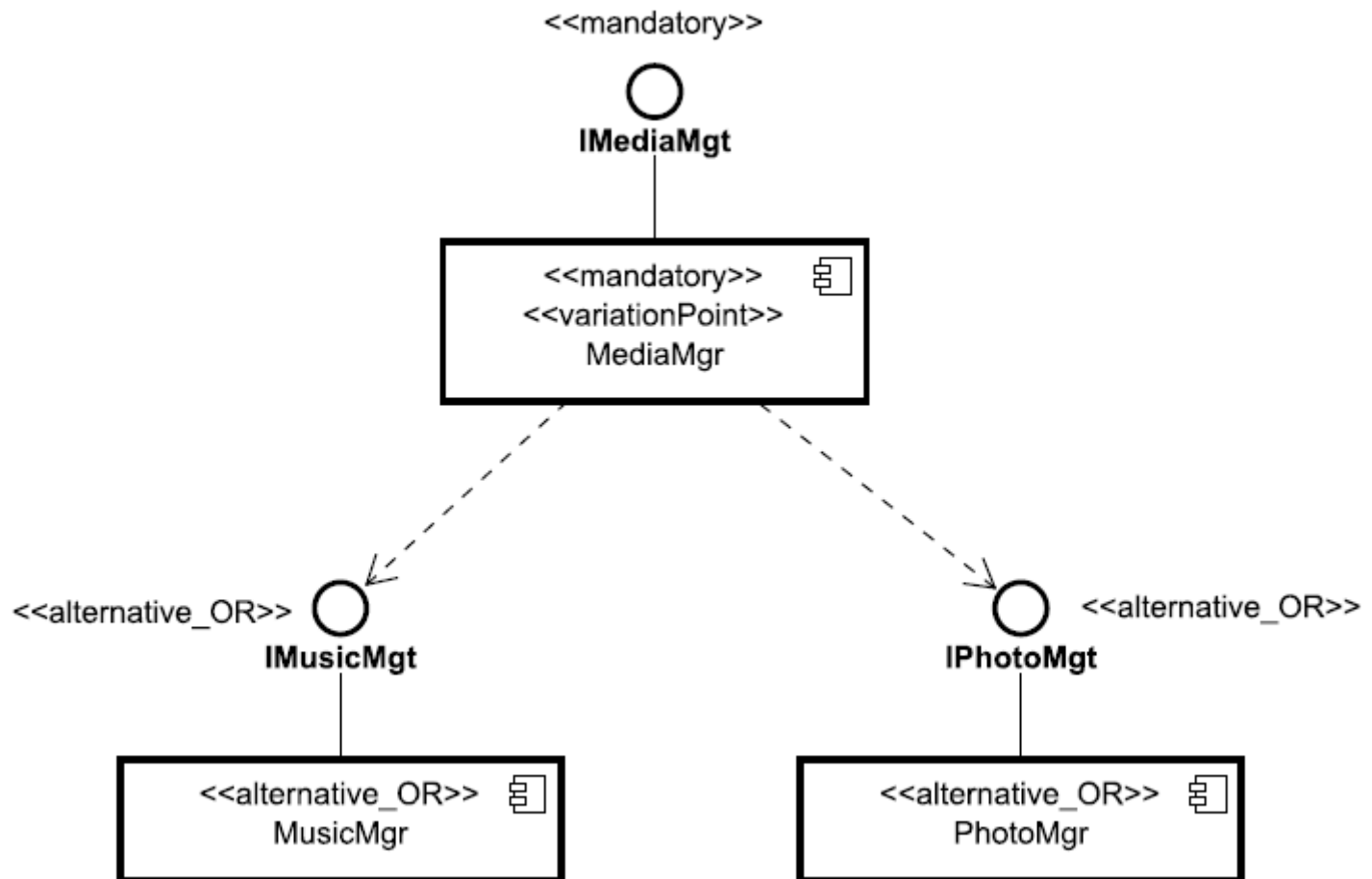
# Running SMartyModeling...



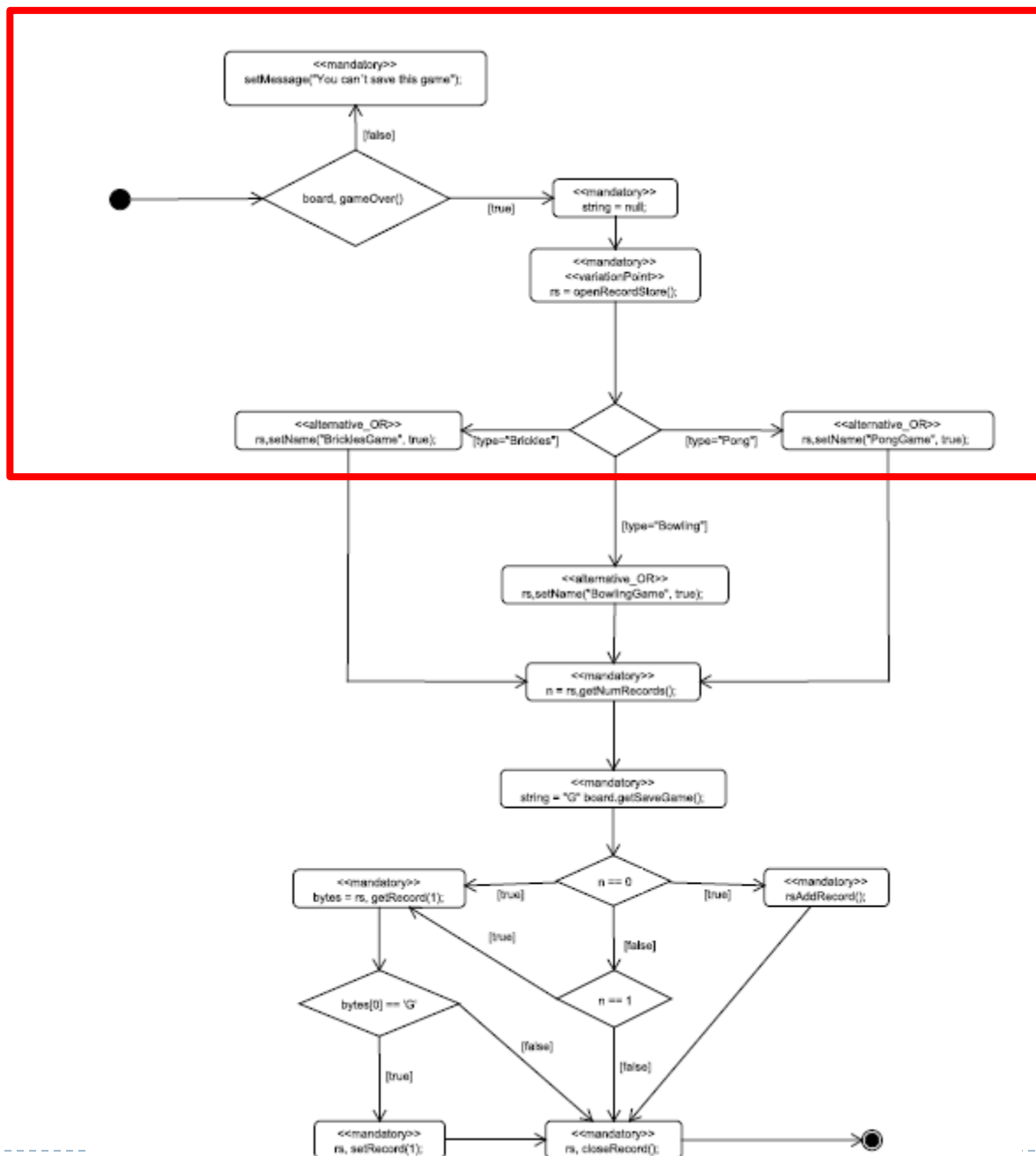


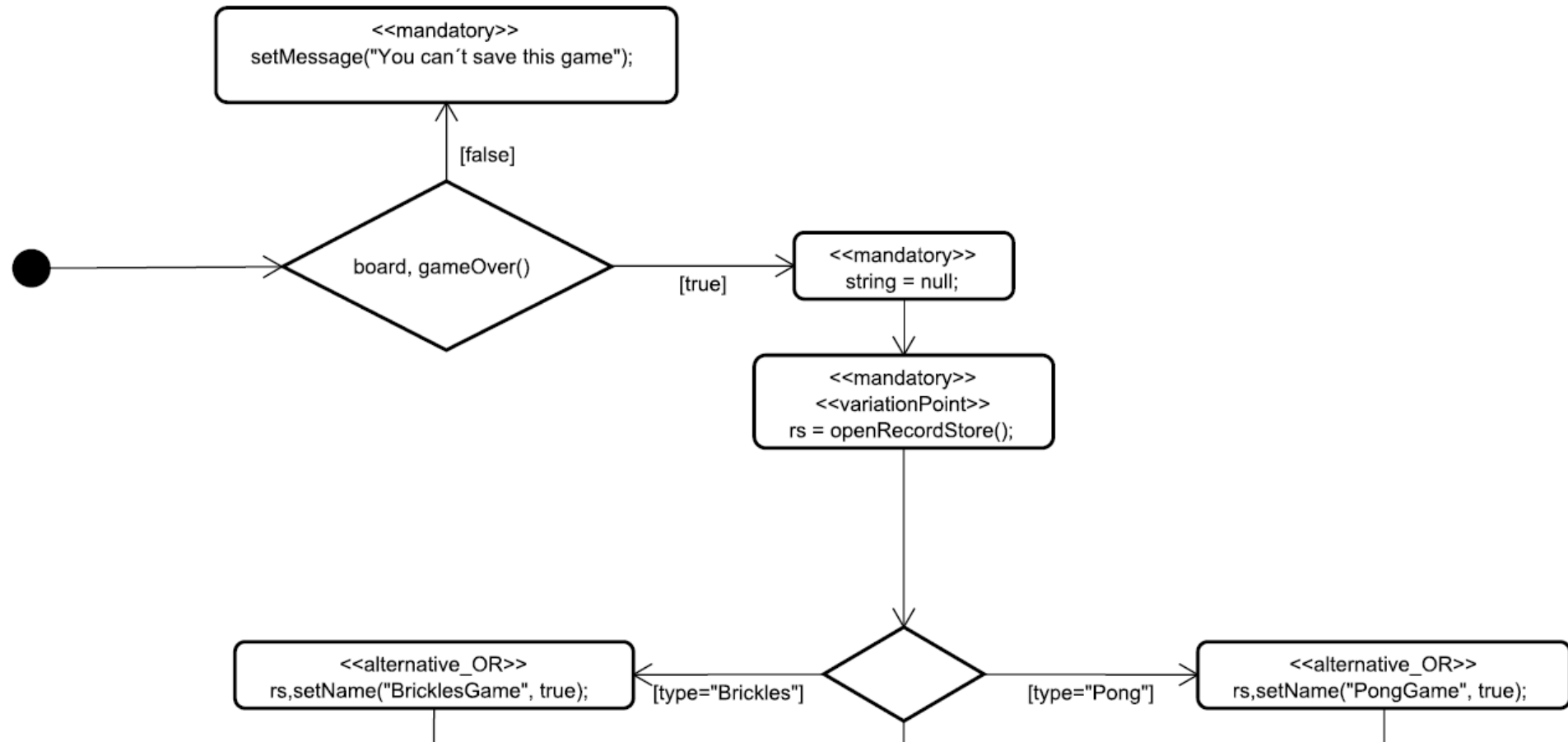


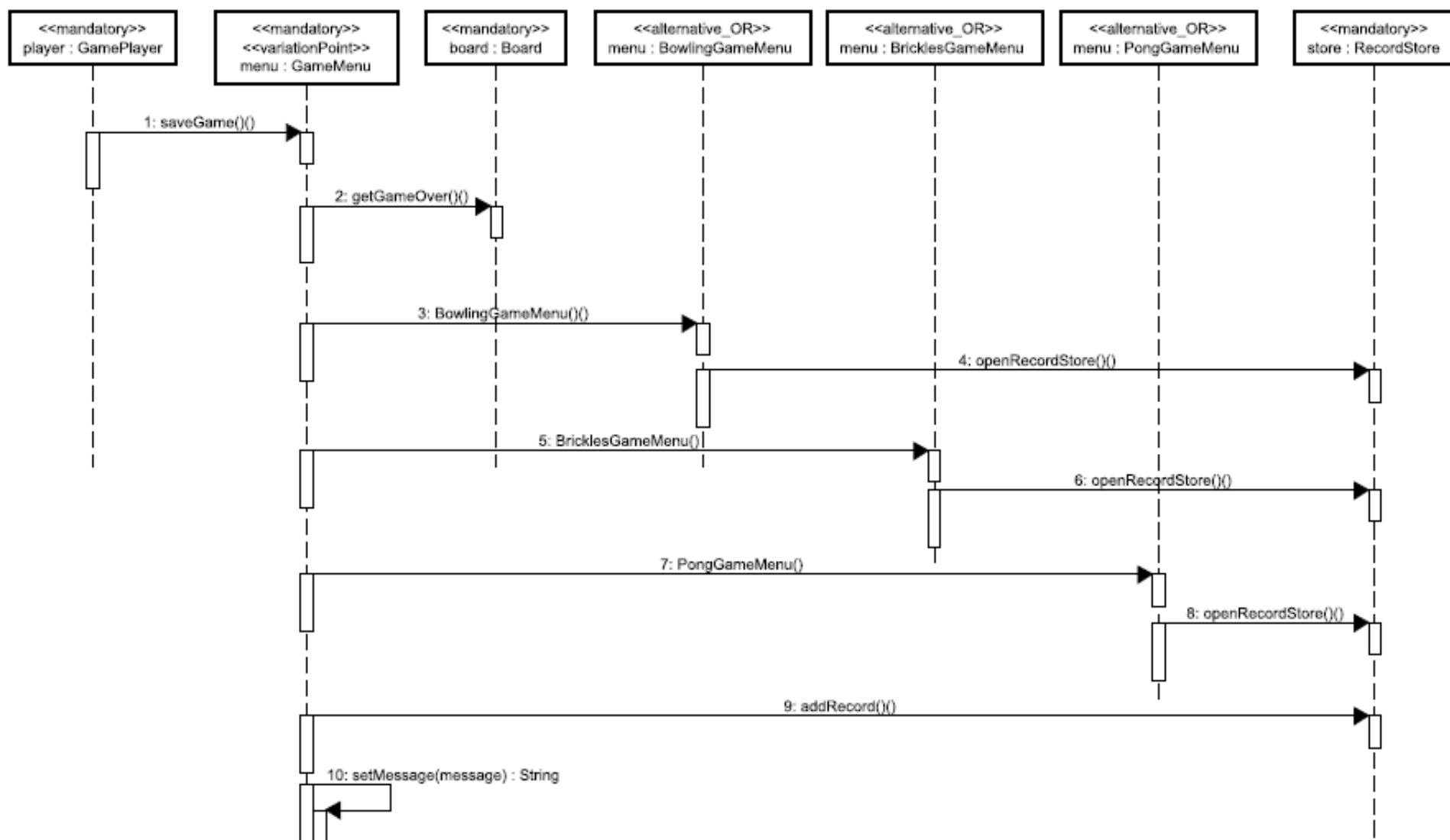


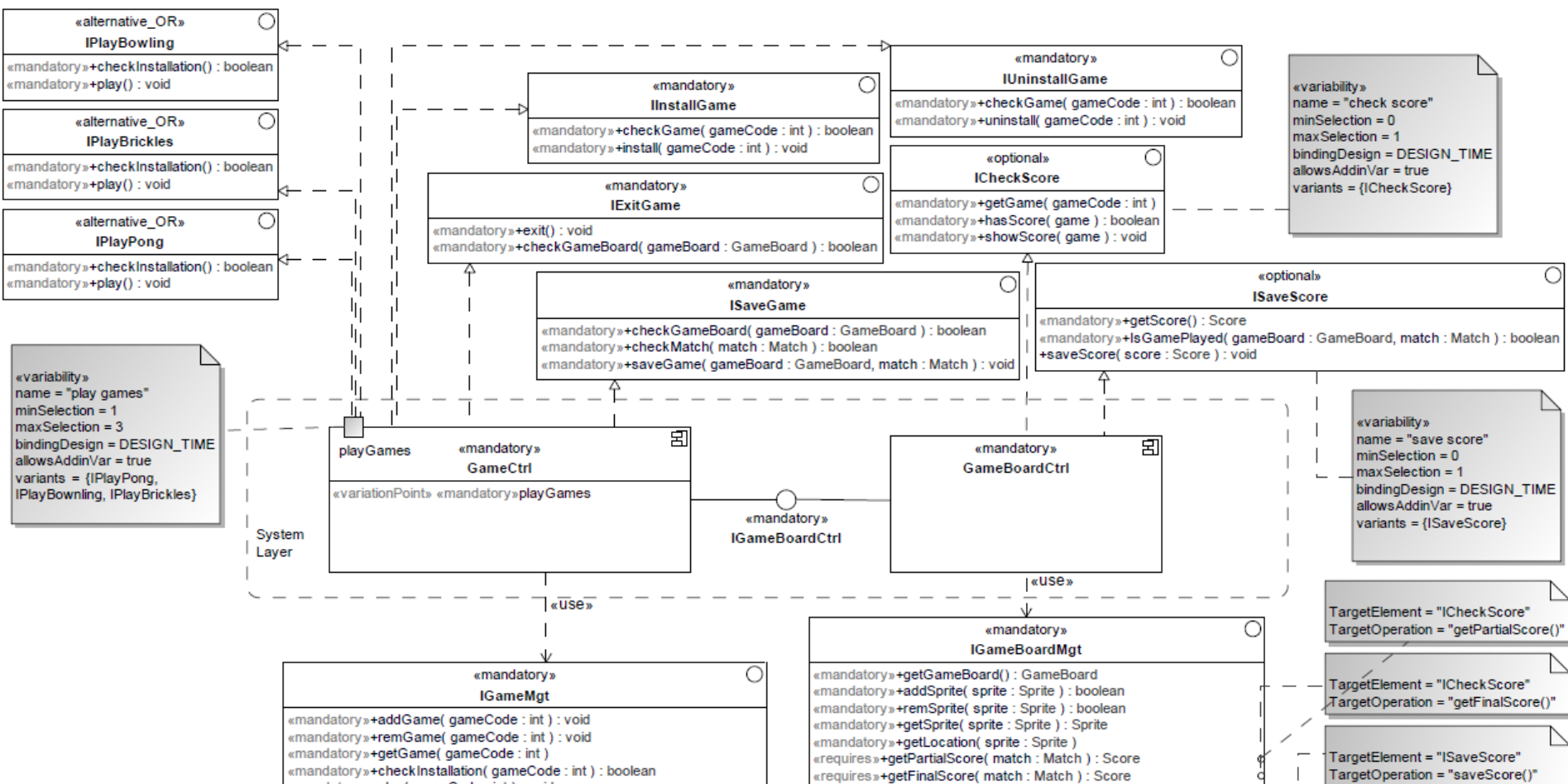












# E agora???

---

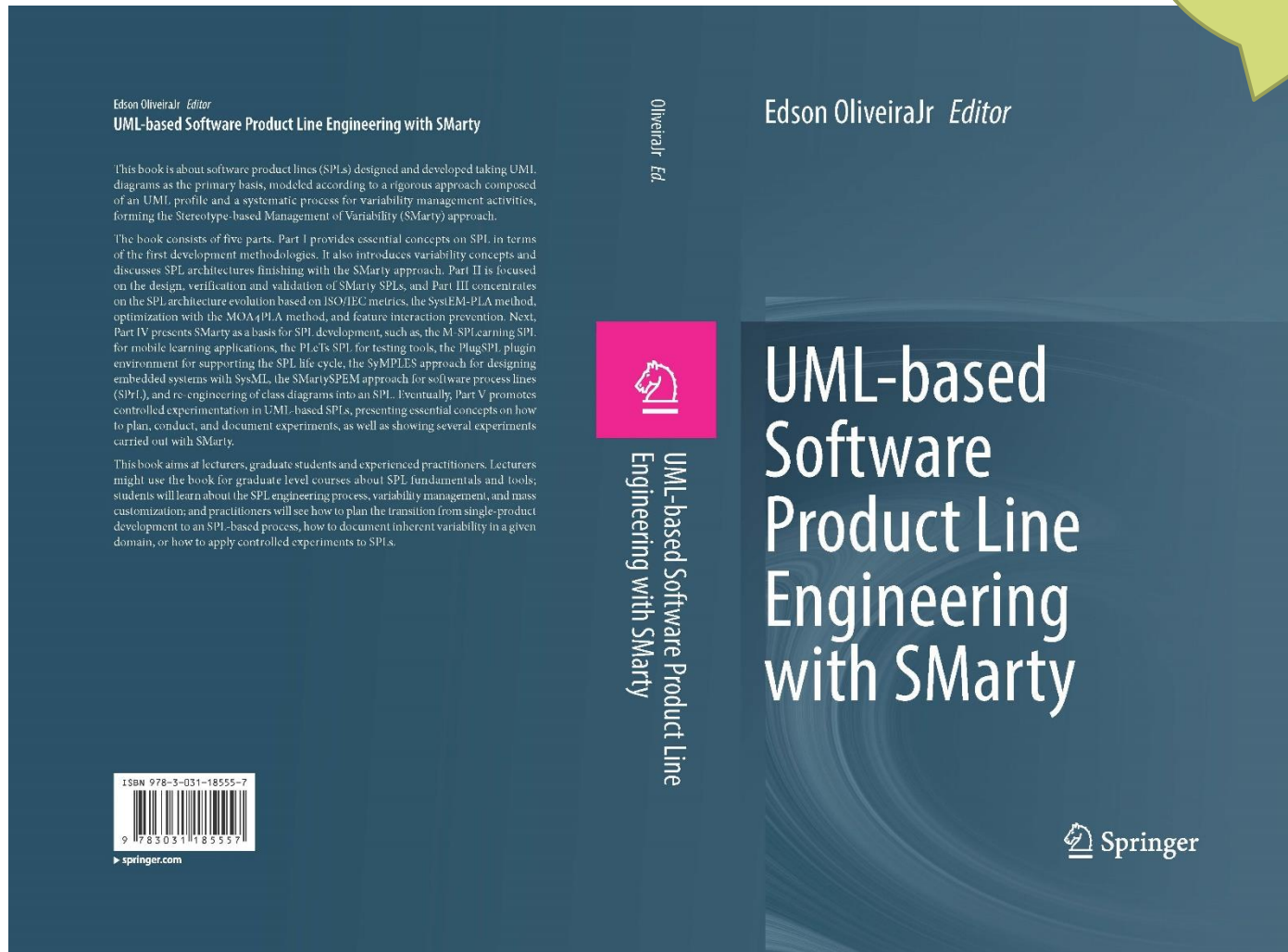
- ▶ Inspeção
  - ▶ SMartyCheck
  - ▶ SMartyPerspective
- ▶ Teste baseado em modelos
  - ▶ SMartyTesting
  - ▶ SPLiT-MBt
- ▶ Avaliação
  - ▶ SMartyMetrics – ISO 25010
  - ▶ SystEM-PLA
- ▶ Otimização
  - ▶ MOA4PLA – algoritmos genéticos e search-based
- ▶ Redução de *feature interaction*
  - ▶ Casos de uso, classes
- ▶ Import/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



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




Enter the following coupon code at checkout on [link.springer.com](https://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.

Edson Oliveira Jr *Editor*

# UML-based Software Product Line Engineering with SMarty

 Springer

E. Oliveira Jr

## UML-based Software Product Line Engineering with SMarty

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



## Part I Fundamentals of Software Product Lines and the SMarty Approach

<b>1 Principles of Software Product Lines</b>	3
Edson Oliveira Jr and David Benavides	
1.1 Characterizing Software Product Lines	3
1.2 SPL Terminology	6
1.3 SPL Engineering Methodologies and Reference Model	7
1.3.1 First Generation Methodologies	7
1.3.2 Second Generation Software Product Line Engineering (2CPL)	15
1.3.3 ISO/IEC Standards for SPL Engineering, Management and Tools	17
1.4 SPL Development Approaches	18
1.4.1 The Proactive Approach	18
1.4.2 The Extractive Approach	19
1.4.3 The Reactive Approach	21
1.4.4 Feature-Oriented SPL Development	22
1.5 Final Remarks	22
References	23
<b>2 Variability Implementation and UML-based Software Product Lines</b>	27
Ana Paula Allian, Eliza Yumi Nakagawa, Jabier Martinez, Wesley K. G. Assunção, and Edson Oliveira Jr	
2.1 Introduction	28
2.2 Implementing variability	29
2.2.1 Variability in the Problem Space	29
2.2.2 Variability in the Solution Space	30
2.2.3 SPL Variability Tools	34
2.3 Overview of UML-based SPL	35
2.4 Discussion	36
2.5 Final Remarks	37
References	37
<b>3 Software Product Line Architectures</b>	43
Crescencio Lima, Thelma Elita Colanzi, Matthias Galster, Ivan Machado, and Edson Oliveira Jr	
3.1 Software Architecture Foundations	44
3.1.1 What is Software Architecture	44
3.1.2 Quality Attributes and Software Architecture	45
3.1.3 Software Architecture Descriptions	45
3.1.4 Variability in Software Architecture	49
3.2 Software Product Line Architectures Foundation	49
3.2.1 Product Lines and Product Line Architectures	49
3.2.2 Product Line Design	50
3.2.3 Product Line Architecture Description	52
3.3 Product Line Architectures versus Reference Architectures	52
3.4 A Product Line Architecture Example	53
3.5 Final Remarks	55
References	56
<b>4 The SMarty Approach for UML-based Software Product Lines</b>	59
Edson Oliveira Jr, Iana M. S. Gimenes, and José C. Maldonado	
4.1 Overview of the SMarty Family	59
4.2 The SMarty Profile	63
4.3 The SMarty Process and Guidelines	68
4.4 Final Remarks	71
References	71

## Part II SMarty-based Software Product Lines: Design, Verification and Validation

<b>5 Designing, Tracing, and Configuring Software Product Lines with SMarty</b>	77
Edson Oliveira Jr, Leandro F. Silva, Anderson S. Marcolino, Thais S. Nepomuceno, André F. R. Cordeiro, and Rodrigo Pereira dos Santos	
5.1 Quick Start to SMarty Modeling	78
5.2 Designing SMarty Diagrams	79
5.2.1 Use Case Diagrams	80
5.2.2 Class Diagrams	83
5.2.3 Component Diagrams	85
5.2.4 Activity Diagrams	88
5.2.5 Sequence Diagrams	88
5.3 Traceability Among Designed Elements	90
5.4 Configuring Specific Products	93
5.5 Exporting and Importing SPLs	98
5.6 Final Remarks	99
References	100

<b>6 Product-Line Architecture Designing with SMarty Components</b>	101
Márcio H. G. Bera, Thelma Elita Colanzi, Edson Oliveira Jr, Nelson Tenório, Willian Marques Freire, and Aline M. M. Miotto Amaral	
6.1 The Role of SMarty in this Work	102
6.2 SMarty Components	102
6.3 Requirements Workflow	105
6.3.1 Activity: Requirements Definition	105
6.4 Specification Workflow	108
6.4.1 Activity: Component Identification	109
6.4.2 Activity: Component Interaction	114
6.4.3 Activity: Specify Components	116
6.5 Final Remarks	120
References	121
<b>7 Model-based Inspections of Software Product Lines</b>	123
Giovanna Bettin, Ricardo Thes Gerald, and Edson Oliveira Jr	
7.1 The Role of SMarty in this Work	123
7.2 Software Inspection Foundations	124
7.2.1 Checklist-Based Reading	125
7.2.2 Scenario-Based Reading	125
7.2.3 Perspective-Based Reading	126
7.3 SMartyCheck	126
7.3.1 SMartyCheck: Defect Types Taxonomy	127
7.3.2 SMartyCheck: Inspection of SMarty SPLs	131
7.3.3 SMartyCheck: Application Examples	131
7.4 SMartyPerspective	133
7.4.1 SMartyPerspective: Defect Types Taxonomy	134
7.4.2 SMartyPerspective Scenarios	135
7.4.3 The Product Manager Perspective	136
7.4.4 The Domain Requirements Engineer Perspective	138
7.4.5 The Domain Architect Perspective	142
7.4.6 The Domain Developer Perspective	147
7.4.7 The Domain Asset Manager Perspective	149
7.4.8 SMartyPerspective: Application Examples	151
7.5 Final Remarks	153
References	154

<b>8 Model-based Testing of Software Product Lines</b>	157
Kleber Lopes Petry, Edson Oliveira Jr, Leandro Teodoro Costa, Aline Zanin, and Avelino Francisco Zorzo	
8.1 The Role of SMarty in this Work	158
8.2 SPLIT-MBT: A Model-based Testing Method for Software Product Lines	158
8.2.1 SPLIT-MBT Characterization	159
8.2.2 SPLIT-MBT Phases	161
8.2.3 SPLIT-MBT Application Example	167
8.3 SMartyTesting: MBT on Use Case and Sequence Diagrams	175
8.3.1 SMartyTesting Characterization	176
8.3.2 Used Models	176
8.3.3 Converting Sequence Diagrams to Activity Diagrams	176
8.3.4 Used Tool	179
8.3.5 SMartyTesting Phases	180
8.3.6 Step 1 - Mapping Sequence Diagrams to Activity Diagrams	180
8.3.7 Step 2 - Generating Test Sequences	184
8.3.8 Variability Resolution	185
8.3.9 Limitations on the use of SPLIT-MBT	185
8.3.10 SMartyTesting Application Example	185
8.4 Final Remarks	188
References	189

## Part III Product-Line Architecture Evolution

<b>9 Maintainability Metrics for PLA Evaluation based on ISO/IEC 25010</b>	193
André F. R. Cordeiro, Leandro F. Silva, and Edson Oliveira Jr	
9.1 The Role of SMarty in this Work	194
9.2 SMartyMetrics Characterization	194
9.3 SMartyMetrics - Quality Attributes	194
9.4 SMartyMetrics - Metrics	196
9.4.1 Modularity	198
9.4.2 Reusability	200
9.4.3 Modifiability	204
9.4.4 Testability	204
9.5 SMartyMetrics Guidelines	206
9.6 Applying Metrics in SMartyModeling	210
9.7 Application Example	211
9.8 Final Remarks	218
References	219
<b>10 The System-PLA Evaluation Method</b>	221
Edson Oliveira Jr, André F. R. Cordeiro, Iana M. S. Gimenes, and José C. Maldonado	
10.1 The Role of SMarty in this Work	222
10.2 Characterization of System-PLA	222
10.3 Evaluation Metaprocess (EMP)	223
10.4 Evaluation Guidelines	226
10.5 Application Example	228
10.5.1 Product Line Architecture	229
10.5.2 Planning	230
10.5.3 Data Collection	237
10.5.4 Data Analysis and Documentation	242
10.6 Final Remarks	242
References	242

<b>11 Optimizing Product-Line Architectures with MOA4PLA</b>	245
Thelma Elita Colanzi, Mamoru Massago, and Silvia Regina Vergilio	
11.1 Introduction	245
11.2 Introduction to Multi-Objective Optimization	247
11.3 Search-Based SPL Optimization	249
11.4 The Role of SMarty in this Work	250
11.5 MOA4PLA	250
11.5.1 Product-line Architecture Representation	252
11.5.2 Evaluation Model	253
11.5.3 Search Operators	253
11.5.4 Implementation Aspects	259
11.6 Application Example	259
11.7 Final Remarks	262
References	264
<b>12 Preventing Feature Interaction with Optimization Algorithms</b>	267
Luciane Nicolodi Baldo, Aline M. M. Miotto Amaral, Edson Oliveira Jr and Thelma Elita Colanzi	
12.1 Introduction	268
12.2 Background	269
12.2.1 Approaches to Detect and Resolve Feature Interaction	269
12.2.2 PLA Design Optimization	270
12.2.3 The Role of SMarty in this Work	271
12.3 A Search-Based Approach to Prevent Feature Interaction	272
12.3.1 Potential-Feature Interaction Detection Patterns	272
12.3.2 Feature Interaction Preventive Actions	273
12.3.3 Implementation Aspects	276
12.4 Application Example	278
12.5 Limitations	283
12.6 Final Remarks	283
References	284

## Part IV SMarty-related Research

<b>13 M-SPLearning: a Software Product Line for Mobile Learning Applications</b>	289
Venilton Falvo Jr, Anderson S. Marcolino, Nemesio Duarte Filho, Edson Oliveira Jr, and Ellen F. Barbosa	
13.1 M-Learning Domain and the Role of SMarty in this Work	290
13.2 M-SPLearning Domain Engineering	293
13.2.1 Domain Analysis	294
13.2.2 Architecture Definition	297
13.2.3 Components Design	299
13.2.4 Production Plan	299
13.3 M-SPLearning Application Engineering	302
13.3.1 Products Generation	302
13.3.2 Products Evaluation	306
13.3.3 Related Work	309
13.4 Final Remarks	309
References	313
<b>14 PLTs: A Software Product Line for Testing Tools</b>	317
Elder M. Rodrigues, Avelino F. Zorzo, and Luciano Marchezan	
14.1 The Role of SMarty in this Work	318
14.2 Model Based Testing	318
14.3 PLTs Project	320
14.3.1 Requirements	322
14.3.2 Design Decisions, Process and Variability Control	323
14.3.3 Architecture and Implementation	327
14.4 Example of Use: Generating Performance MBT Tools	328
14.5 Example of Use: Generating Structural MBT Tools	331
14.6 Final Remarks	333
References	334
<b>15 PlugSPL: An Environment to Support SPL Life Cycle</b>	337
Elder M. Rodrigues and Avelino F. Zorzo	
15.1 The Role of SMarty in this Work	338
15.2 Introduction	338
15.3 Context	339
15.4 Background	340
15.5 Requirements	340
15.6 Design Decisions	342
15.7 PlugSPL Environment: Supporting Plugin-based Software Product Lines	343
15.7.1 SPL Design Activity	344
15.7.2 Component Management Activity	346
15.7.3 Product Configuration Activity	346
15.7.4 Product Generation Activity	348
15.8 Related Work	349
15.9 Final Remarks	350
References	350
<b>16 SyMPLES: Embedded Systems Design with SMarty</b>	353
Rogério F. da Silva, Alexandre A. Giron, and Iana M. S. Gimenes	
16.1 The Role of SMarty in this Work	353
16.2 The SyMPLES Approach	355
16.2.1 SyMPLES Profiles	355
16.2.2 SyMPLES Processes	355
16.2.3 SyMPLES Model Transformation	358
16.3 Application Example	360
16.3.1 Product Configuration	360
16.3.2 ATL Transformation	361

16.3.3 Generate Functional Blocks	362
16.4 Validation of MDE Transformations	362
16.4.1 Test case generation based on the Metamodel	364
16.4.2 Test case generation based on the SPL	365
16.5 Final Remarks	366
References	367
<b>17 Variability Representation in Software Process with the SMartySPEM Approach</b>	369
Maicon Pazin, Jaime Dias, Edson Oliveira Jr, Fellipe Araújo Azeiteiro, Uirá Kulesza, and Eldiane Nogueira Teixeira	
17.1 The Role of SMarty in this Work	370
17.2 Software Process Lines (SPL) Fundamentals	370
17.2.1 Software Process Lines Overview	371
17.2.2 SPL Variability Management	376
17.3 The SMartySPEM Approach	378
17.4 Application Example	381
17.5 Final Remarks	386
References	388
<b>18 Re-engineering UML Class Diagram Variants into a Product Line Architecture</b>	391
Wesley K. G. Assunção, Silvia R. Vergilio, and Roberto E. Lopez-Herrejon	
18.1 Introduction	392
18.2 Proposed Approach	393
18.2.1 Step 1: Search-based Model Merging	393
18.2.2 Step 2: Variability annotation	399
18.3 Evaluation	399
18.3.1 Implementation Aspects and Experimental Setup	400
18.3.2 Subject Systems	401
18.3.3 Results and Analysis	403
18.4 UML-based SPLs	408
18.5 Final Remarks	409
References	409
<b>Part V Software Product Line Experimentation</b>	
<b>19 Controlled Experimentation of Software Product Lines</b>	413
Viviane R. Furtado, Henrique Vignando, Carlos D. Luz, Igor F. Steinmacher, Marcos Kalinowski, and Edson Oliveira Jr	
19.1 Experimentation in Software Engineering	414
19.2 Quality of Experiments in Software Engineering	416
19.3 Software Product Line Experiments	417
19.4 Guidelines to Report SPL Experiments	418
19.4.1 Proposed Guidelines	419
19.4.2 Conceptual Model to Support Guidelines	426
19.5 An Ontology for SPL Experimentation	428
19.5.1 Software Engineering Ontologies	428
19.5.2 Building The OntoExper-SPL	428
19.6 Final Remarks	436
References	437
<b>20 Experimentally-based Evaluations of the SMarty Approach</b>	441
Anderson S. Marcolino, Thais S. Nepomuceno, Lilian P. Scatolon, and Edson Oliveira Jr	
20.1 Experiments on UML-based variability management approaches	442
20.1.1 The UML-based Variability Management Approaches	443
20.2 Experimental Evaluations of Effectiveness of Identification and Representation of Variabilities	447
20.2.1 Objectives (G.1)	447
20.2.2 Hypothesis Formulation (G.2)	448
20.2.3 Variables Definitions (G.3)	449
20.2.4 Sample Size (G.4)	451
20.2.5 Participants Definition and Selection (G.5)	451
20.2.6 Research Experiment Topic Definition (G.6)	455
20.2.7 Experimental Design Definition (G.7)	455
20.2.8 Experimental Materials Definition and Selection (G.8)	459
20.2.9 Experimental Material Validation (G.9)	459
20.2.10 Experimental Tasks Description (G.10)	461
20.2.11 Training Requirements Description (G.11)	462
20.2.12 Pilot Project Conduction (G.12)	463
20.2.13 Experimental Environment Conduction (G.13)	463
20.2.14 Experimental Date of Execution (G.14)	463
20.2.15 Experimental Execution (G.15)	463
20.2.16 Data Collected Description (GD.16)	464
20.2.17 Data Analysis Procedures (GD.17)	467
20.2.18 Mortality Rate (GD.18)	468
20.2.19 Statistical Data Analysis Tools (GD.19)	488
20.2.20 Effect Size (GD.20)	488
20.2.21 Results in the Point of View of Researchers and Practitioners (GD.21)	490
20.2.22 Implications of Developed Treatments (GD.22)	490
20.2.23 Threats to Validity Identified in the Experiment (GD.23)	491
20.2.24 Experimental Package Source (GD.24)	491
20.2.25 Experimental Template Used to Conduct Plan or Document the Experiment (GD.25)	492
20.3 SMarty Improvements based on The Experimental Evaluations	492
20.3.1 Evolution of the SMarty Approach for Identification and Representation of Variabilities	492
20.3.2 Evolution of the SMarty Approach for Configuration and Support for Traceability	493
20.4 Lessons Learned and SMarty Improvements	494
20.5 Final Remarks	495
References	499

*Obrigado!!!  
Thank you!!!*

**Prof. Dr. Edson Oliveira Jr**

Informatics Department

State University of Maringá, Brazil

[edson@din.uem.br](mailto:edson@din.uem.br)