UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MARCIO HENRIQUE GIMENES BERA

SMartyComponents: um processo para especificação de arquiteturas de linha de produto de software componentizadas

Maringá
2015

MARCIO HENRIQUE GIMENES BERA

SMartyComponents: um processo para especificação de arquiteturas de linha de produto de software componentizadas

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Edson A. Oliveira Junior
Coorientadora: Profª. Drª. Thelma E. Colanzi

Maringá
2015

Dados Internacionais de Catalogação-na-Publicação (CIP)
(Biblioteca Central - UEM, Maringá – PR., Brasil)

B482s

Bera, Marcio Henrique Gimenes
SMartyComponents: um processo para especificação de arquiteturas de linha de produto de software componentizadas / Marcio Henrique Gimenes Bera. -- Maringá, 2015.
247 f. : il. col., figs., tabs.

Orientador: Prof. Dr. Edson Alves de Oliveira Junior.
Coorientadora: Profa. Dra. Thelma Elita Colanzi.
Dissertação (mestrado) - Universidade Estadual de Maringá, Centro de Tecnologia, Departamento de Informática, Programa de Pós-Graduação em Ciência da Computação, 2015.

1. Software - Arquitetura de linha de produto. 2. Software - Componentes. 3. *SMarty* - Gerenciamento de variabilidades. 4. UML (Linguagem de modelagem unificada). 5. *UML Components* - Processo de desenvolvimento. I. Oliveira Junior, Edson Alves, orient. II. Colanzi, Thelma Elita, coorient. Universidade Estadual de Maringá. Centro de Tecnologia. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação. IV. Título.

CDD 21.ed. 005.12

ECLS

FOLHA DE APROVAÇÃO

MARCIO HENRIQUE GIMENES BERA

SMartyComponents: um processo para especificação de arquiteturas de linha de produto de software componentizadas

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação pela Banca Examinadora composta pelos membros:

BANCA EXAMINADORA

Prof. Dr. Edson Alves de Oliveira Junior
Universidade Estadual de Maringá – DIN/UEM

Prof. Dr. Igor Fabio Steinmacher
Universidade Tecnológica Federal do Paraná – DACOM/UTFPR-CM

Profa. Dra. Patrícia Vilain
Universidade Federal de Santa Catarina – INE/UFSC

Aprovada em: 08 de dezembro de 2015.
Local da defesa: Sala 101, Bloco C56, *campus* da Universidade Estadual de Maringá.

DEDICATÓRIA

A Jesus Cristo, o Rei dos Reis.
A Ele toda Honra, toda Glória,
todo Domínio e todo o Poder.
Long Live The King.

AGRADECIMENTOS

Sou grato a DEUS pelo fôlego de vida, por me guiar e me inspirar nesta caminhada, e por ser minha âncora em meio às tempestades. Agradeço aos meus pais, **Orlando Gimenes Bera** e **M. Madalena Domingues Bera**, por me incentivar todos os dias a lutar pelos meus sonhos e pelas constantes orações; a minha irmã **Marcela Gimenes Bera Oshita** por me apoiar e me auxiliar quando precisei; e a minha irmã **Ana Gimenes Bera** por prestar auxílio sempre que preciso. Sou grato também a minha fiel companheira, minha esposa **Milena Roberta P. Bera**, que esteve ao meu lado desde o início, me apoiou em todas as circunstâncias e acreditou em mim. Te amo.

Sou grato aos amigos: **Guilherme Tomoike**, pelo incentivo dado desde o começo; à **Patricia Tiemi Tanaka Tomoike**, pelas contribuições nas revisões da dissertação; ao **Rafael Augusto Rodrigues**, pela sua sabedoria, incentivo e por me motivar; à **Cibele Andrade**, por me dar umas dicas de mestre e me incentivar quando me encontrava desanimado; Aos meus sogros **Roberto Carlos Pereira** e **Andrea Elaene Pereira** pela confiança e incentivo nesta caminhada; Aos meus pastores **Edvaldo O. Pardini**, **Vera M. R. de Carvalho Pardini**, **Filipe A. de Carvalho Pardini**, **Daniela P. Rodrigues Pardini** e **Rosangela M. da Conceição**, além dos meus líderes **Rodrigo Bechi** e **Sheder Chagas** e dos demais amigos e irmãos da **1ª IEQ de Maringá**, pelo apoio espiritual e pelo incentivo em alcancar meus objetivos.

Sou grato também aos meus colegas de pesquisa: **Anderson Marcolino**, **Ricardo Theis Geraldi**, **Jaime Willian Dias**, **André Felipe Cordeiro**, **Maicon Pazin** e **Ana Allian** pelos auxílios e colaborações prestadas; aos demais amigos de classe dos anos 2013, 2014 e 2015.

Agradeço a coorientadora **Profª. Drª. Thelma E. Colanzi** pela paciência e pelo grande auxílio prestado em todo o período em que esteve junto nas orientações. Agradeço em especial ao meu orientador **Prof. Dr. Edson Alves de Oliveira Júnior** por ter me ajudado desde o início, pela sua paciência, perseverança e todo auxílio prestado sempre que precisei. Agradeço aos professores que me auxiliaram durante todo o período das disciplinas; à secretária **Maria Inês Davanço** pela paciência, ajuda, amizade e disponibilidade em ajudar sempre.

Agradeço também a **Coordenação de Aperfeiçoamento de Pessoal de Nível Superior** (CAPES) pelo apoio financeiro concedido durante o desenvolvimento desta pesquisa.

Seria impossível me lembrar de todos aqueles que me ajudaram direta ou indiretamente, para que eu chegasse até aqui, mas sou imensamente grato por terem acreditado em mim. **MUITO OBRIGADO!**

SMartyComponents: um processo para especificação de arquiteturas de linha de produto de software componentizadas

RESUMO

Reduzir os recursos investidos no desenvolvimento de software é um tópico que tem sido constantemente investigado na academia e na indústria. Técnicas de reutilização de software efetivas veêm sendo exploradas por meio de *frameworks* e componentes. O Desenvolvimento Baseado em Componentes (DBC) tem como característica desenvolver software por meio de um conjunto estruturado de componentes, interfaces e contratos bem definidos. O DBC é apoiado por processos estabelecidos como é o caso do *UML Components*, que já nos estágios iniciais identifica os possíveis componentes de um sistema. Existem outras abordagens com foco em reúso, como por exemplo, Linha de Produto de Software (LPS), que representa um conjunto de sistemas com características similares e certas particularidades para um determinado domínio. Um dos artefatos mais importantes de uma LPS é a Arquitetura de LPS (ALPS), pois representa uma abstração de todas as possíveis arquiteturas de produtos específicos. Para tanto, atividades de Gerenciamento de Variabilidade (GV) devem ser muito bem definidas para que uma ALPS possa refletir as características de uma LPS. *Stereotype-based Management of Variability* (*SMarty*) é uma abordagem de GV que se destaca por permitir representar variabilidades em modelos UML, incluindo componentes de arquitetura lógica. A adoção de LPS junto ao DBC, pode permitir explorar os benefícios de ambas as abordagens visando minimizar os recursos necessários, além de permitir a customização em massa desses produtos. Esta dissertação tem por objetivo formalizar a proposta de um processo baseado na combinação do processo *UML Components* com a abordagem *SMarty*, denominado *SMartyComponents*, para especificar ALPSs componentizadas. Uma evolução da abordagem *SMarty* foi necessária, com objetivo de explorar a capacidade de representação de variabilidades em componentes, interfaces, operações e portas da UML 2.5. Tal evolução foi avaliada por meio de um estudo experimental, que apresentou indícios de sua efetividade. *SMartyComponents* foi avaliado por meio de um estudo empírico qualitativo, adotando procedimentos de *Grounded Theory*, do ponto de vista de especialistas em DBC e LPS. Os resultados obtidos de tal estudo forneceram indícios de viabilidade do processo e serviram como base para melhorias na proposta do *SMartyComponents*.

Palavras-chave: Arquitetura de Linha de Produto de Software. Componentes. Desenvolvimento Baseado em Componentes. SMarty. UML Components. Variabilidades.

SMartyComponentes: a process to specify componentized software product line architectures

ABSTRACT

Decrease the resources invested in software development is a topic which has been investigated in academy and industry. Effective software reuse techniques have been explored, such as frameworks and components. Component-Based Development (CBD) is an approach that is aimed at developing software by means of a structured set of components, interfaces and well-defined contracts. CBD is supported by well-established process, such as the *UML Components*, which focuses on identifying candidate systems components. There are other approaches focusing on reuse, such as *Software Product Line* (SPL), which represents a set of systems with similar characteristics and certain particularities for a given domain. One of the most important artifacts of an SPL is the SPL Architecture (SPLA), as it represents an abstraction of all possible architectures of specific products. Therefore, *Variability Management* (VM) activities should be well defined so that the SPLA can reflect the characteristics of an SPL. *Stereotype-based Management of Variability* (*SMarty*) is a VM approach, and it stands out for allowing representing variability in UML models, including components. Combining SPL and CBD, could allow exploiting the benefits of both approaches aiming at reducing costs and *time-to-market*, increasing quality in product development and enabling mass customization of such products. Thus, this work presents the proposal of a process based on the combination of the *UML Components* and *SMarty* named *SMartyComponents*, aimed at specifying component-based SLPAs. An evolution of *SMarty* was needed for exploring the capability of representing variability components, interfaces, operations and ports according to the *UML* 2.5. Such on evolution experimentally evaluated providing evidence of its effectiveness. *SMartyComponents* was empirically evalueted by means of a qualitative study, in which *Grounded Theory* procedures were adopted from the pont of view of experts on CBD and SPL. Obtained results provided evidence of its feasibility serving as a basis for *SMartyComponents* improvements.

Keywords: Components. Component-Based Development. SMarty. Software Product Line Architecture. UML Components. Variability.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

AGM: *Arcade Game Maker*
ALPS: Arquitetura de Linha de Produto de Software
BPMN: *Business Process Model and Notation*
C&C: *Component and Connector*
CBD: *Component Based Development*
DBC: Desenvolvimento Baseado em Componentes
FCA: *Formal Concept Analysis*
GQM: *Goal Question Metric*
GV: Gerenciamento de Variabilidade
LPS: Linha de Produto de Software
MM: *Mobile Media*
MOF: *Meta Object Facility*
MS: Mapeamento Sistemático
OVM: *Orthogonal Variability Model*
PLP: *Product Line Practice*
PLUS: *Product Line UML-based Software engineering*
SEI: *Software Engineering Institute*
SPEM: *Software and Systems Process Engineering Meta-Model*
ROI: *Return On Investment*
SMarty: *Stereotype-based Management of Variability*
SPL: *Software Product Line*
SPLA: *Software Product Line Architecture*
TCLE: Termo de Consentimento Livre e Esclarecido
OMG: *Object Management Group*
UML: *Unified Modeling Language*
VM: *Variability Management*
VU: *Virtual University*
XMI: *XML Metadata Interchange*

SUMÁRIO

1

Introdução

"Muitos homens devem a grandeza da sua vida aos obstáculos que tiveram que vencer."

C. H. Spurgeon (1834 - 1892), Pregador

1.1 Contextualização

Com a evolução da tecnologia nos últimos anos, a indústria de software busca meios para maximizar a produtividade no desenvolvimento de software (Jensen, 2015). Recursos são investidos na criação de produtos para um mesmo domínio sem que muitas vezes os artefatos destes produtos possam ser reutilizados. O tamanho e a complexidade dos softwares em conjunto com as exigências dos usuários por produtos customizáveis (Capilla *et al.*, 2013), faz com que a indústria busque alternativas para acelerar o desenvolvimento dos produtos. Várias ações têm sido propostas e aplicadas para reduzir os recursos empregados buscando promover a qualidade no desenvolvimento de software por meio de técnicas de reúso de software como, por exemplo, *frameworks* e componentes (Ahmaro *et al.*, 2014; Griss, 2015; Subedha *et al.*, 2015).

O Desenvolvimento Baseado em Componentes (DBC) (Gross, 2005; Pressman, 2010) tem como característica o desenvolvimento de software por meio de um conjunto estruturado de componentes que, além de promover o reúso de software, apresenta um impacto

positivo na qualidade, produtividade e custo em projetos de software (Abdellatief *et al.*, 2013; Tekumalla, 2012). O DBC é amplamente difundido na literatura, já com processos bem estabelecidos como, por exemplo, o *UML Components* (Cheesman e Daniels, 2001), que se destaca por permitir identificar os componentes logo nos primeiros *workflows* e especificar uma arquitetura de software componentizada, com o apoio da notação UML. Outras abordagens que visam o reúso podem ser utilizadas juntamente com o DBC.

A abordagem Linha de Produto de Software (LPS) utilizada em conjunto ao DBC pode proporcionar vantagens na redução de recursos e geração de qualidade no desenvolvimento de software. LPS tem como objetivo principal permitir a geração de produtos específicos baseados na reutilização de uma infraestrutura central, o núcleo de artefatos (*Core Assets*), que é formada, principalmente, por uma arquitetura de software e seus componentes (Linden *et al.*, 2007; Pohl *et al.*, 2005). Uma LPS pode gerar vários produtos de uma mesma família, o que indica a existência de várias arquiteturas diferentes, uma para cada produto específico. Uma abstração de todas essas arquiteturas é denominada Arquitetura de LPS (ALPS), que é considerada um dos artefatos mais importantes de uma LPS (Linden *et al.*, 2007). Tal representação só é possível com base em uma atividade de Gerenciamento de Variabilidades (GV) bem estabelecida (Galster *et al.*, 2013).

Uma variabilidade é identificada por meio de pontos de variação e variantes (Pohl *et al.*, 2005). Ponto de variação representa uma tomada de decisão em relação à um artefato, no qual podem ocorrer possíveis variações de um produto para outro, por meio de suas variantes. Tais variantes possuem algumas restrições, impactando em sua possível seleção para a geração de produtos específicos. Para permitir que as variabilidades de uma LPS possam ser gerenciadas, várias abordagens são apresentadas na literatura (Chen *et al.*, 2009; Galster *et al.*, 2013; Thurimella e Bruegge, 2012), com destaque para a abordagem *Stereotype-based Management of Variability* (*SMarty*) (Marcolino, 2014; Marcolino *et al.*, 2013b; OliveiraJr *et al.*, 2013a, 2010b). *SMarty* é apoiada por um perfil UML, o *SMartyProfile*, e possui um processo para GV, denominado *SMartyProcess*. *SMartyProfile* possui um conjunto de estereótipos para representar variabilidades, pontos de variação e variantes. Já o *SMartyProcess*, consiste em um conjunto de atividades e diretrizes que guiam o usuário em como identificar e aplicar sistematicamente cada estereótipo do *SMartyProfile*.

1.2 Motivação

Com base no exposto, acredita-se que seja possível explorar as vantagens da técnica de reúso por meio do processo *UML Components* e da abordagem de LPS com o objetivo

de especificar de forma explícita uma ALPS baseada em componentes, com o apoio da abordagem *SMarty*. Acredita-se que a combinação de *SMarty* e *UML Components*, denominada *SMartyComponents*, possa contribuir na definição de um processo sistemático para a especificação de ALPS com possíveis adaptações do processo em questão. Nos trabalhos de Contieri Junior (2010) e Correia (2010), ensaios incipientes sobre a combinação de *SMarty* e *UML Components* foram realizados. Arquiteturas bem definidas são essenciais para que uma LPS possa ser avaliada por meio de seus possíveis produtos. Métodos de avaliação de ALPS como, por exemplo, o *SystEM-PLA* (OliveiraJr *et al.*, 2013a), o *EATAM* (Kim *et al.*, 2008) e o *HoPLSAA* (Olumofin e Misic, 2005) podem se beneficiar deste tipo de arquitetura levando em conta análises de *trade-off* (OliveiraJr *et al.*, 2011) considerando, por exemplo, atributos de qualidade. Além disso, métricas específicas para ALPS (Contieri Junior *et al.*, 2011; Marcolino *et al.*, 2013a; OliveiraJr e Gimenes, 2014; OliveiraJr *et al.*, 2012, 2010c) podem ser coletadas, complementando tais avaliações de LPS.

Para tanto, um dos desafios que motivaram este trabalho é a identificação de mecanismos para representação de variabilidades em componentes UML, já que a abordagem *SMarty* 5.1 não fornece suporte suficiente para representação de variabilidades em tal modelo com base na UML 2.5, necessitando assim considerar a sua evolução. Assim, com base em um Mapeamento Sistemático (MS) realizado (Apêndice A), pôde-se identificar diferentes níveis de representação de variabilidades em componentes e as principais abordagens que vêm sendo utilizadas para tal, a fim de colaborar com a evolução da abordagem *SMarty*. Além disso, a união das abordagens DBC e LPS pôde ser explorada visando intensificar a aplicação dos conceitos de reúso de componentes em LPSs. Assim, foi necessário investigar como os *workflows* do *UML Components* poderiam apoiar a definição de variabilidades em seus artefatos, de forma a contribuir com a especificação de uma ALPS.

1.3 Objetivos

Este trabalho tem como objetivo principal formalizar o processo *SMartyComponents*, que é o resultado da união do processo *UML Components* e da abordagem *SMarty*, a fim de explorar as vantagens que as abordagens de DBC e LPS proporcionam com enfoque no reúso sistemático dos componentes e artefatos, visando a especificação de ALPSs componentizadas. Para tanto, foram definidos os seguintes objetivos específicos:

- identificar na literatura existente abordagens que buscam representar variabilidades em arquitetura de software e componentes;
- comparar as abordagens identificadas e propor a evolução do *SMarty* 5.1 para identificar e representar variabilidades em componentes UML e respectivos elementos, de acordo com a versão 2.5 da UML;
- avaliar experimentalmente a efetividade da evolução de *SMarty* para componentes UML;
- adaptar as atividades e artefatos do *UML Components* visando a integração de *SMarty* para componentes; e
- avaliar experimentalmente o processo *SMartyComponents*.

1.4 Metodologia de Desenvolvimento de Pesquisa

Para o desenvolvimento desta pesquisa foi adotado como metodologia de trabalho o estudo da fundamentação teórica, com base em revisão bibliográfica nos termos de um Mapeamento Sistemático (MS), com o objetivo de promover a evolução da abordagem *SMarty* para componentes e formalizar o processo *SMartyComponents*. O método empírico foi usado para avaliar a evolução de *SMarty* e a proposta do processo em questão.

A Figura 1.1 apresenta as atividades e etapas da metodologia utilizada neste trabalho:

- **Revisão Bibliográfica:** abrange o estudo dos conceitos dos seguintes temas: o processo *UML Components* e modelos de componentes da UML 2.5, Linha de Produto de Software (LPS) e Gerenciamento de Variabilidades (GV), com enfoque na abordagem *SMarty* e Arquitetura de LPS (ALPS);
- **Mapeamento Sistemático (MS):** consiste em uma busca por estudos em motores de busca e bases de dados científicas, por meio de uma *string* de busca. O objetivo é identificar estudos que apresentam abordagens que vêm sendo utilizadas para representar variabilidades em arquitetura de software e arquiteturas baseadas em componentes;
- **Evolução da Abordagem *SMarty***: evolução da abordagem *SMarty* 5.1 para componentes. Com base nos resultados do MS, os estudos relevantes identificados contribuíram para realizar tal evolução. Dessa forma, foi possível explorar níveis

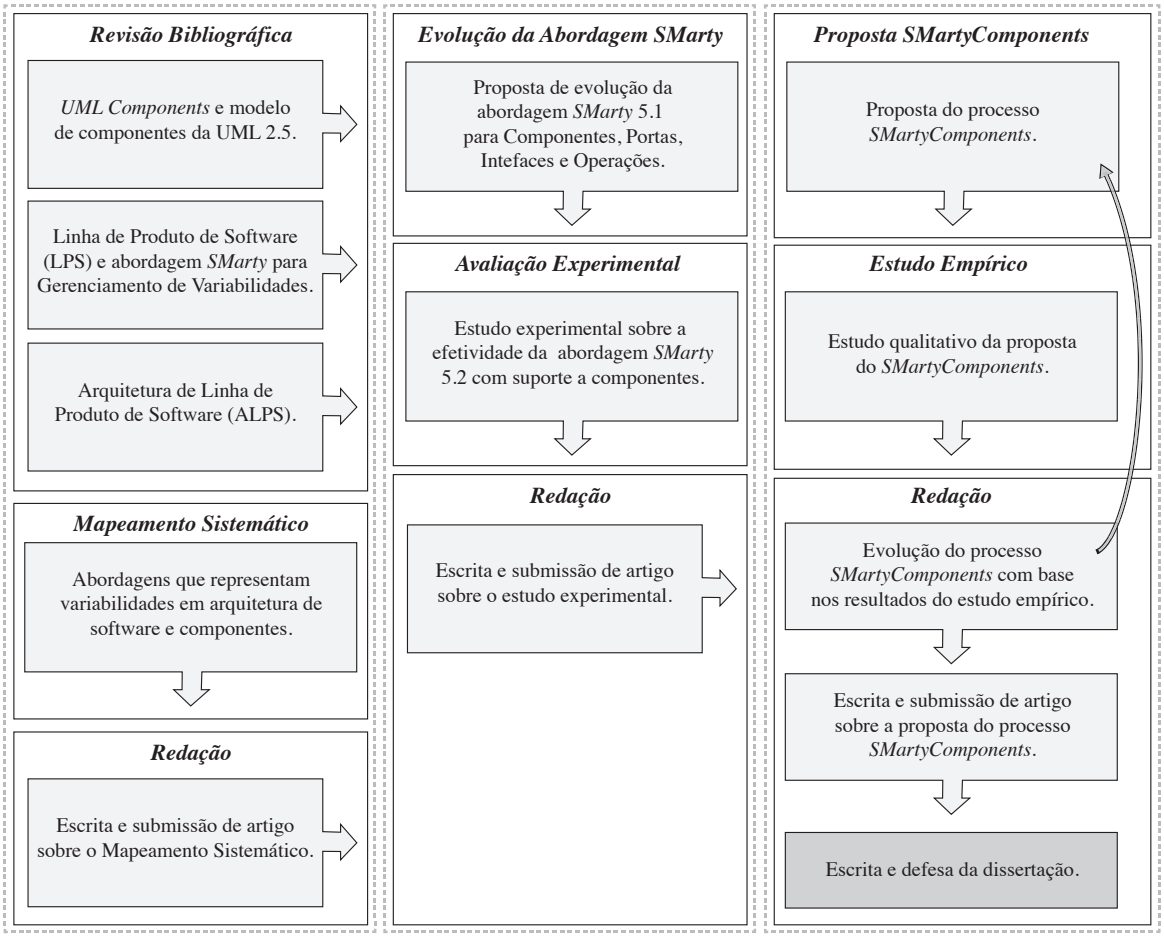

Figura 1.1: Etapas da Metodologia de Desenvolvimento de Pesquisa.

diferentes de representação de variabilidades em componentes de acordo com a UML 2.5;

- **Avaliação Experimental**: refere-se à realização da avaliação experimental, com o objetivo de analisar a efetividade da extensão realizada da abordagem *SMarty* para componentes;

- **Proposta do *SMartyComponents***: apresentação do processo *SMartyComponents* com base na integração *UML Components* e *SMarty*;

- **Estudo Empírico**: refere-se à realização de estudo empírico qualitativo com o objetivo de analisar o processo proposto, *SMartyComponents*, com base em procedimentos de *Grounded Theory* considerando a opinião de especialistas em DBC e LPS; e

- **Redação:** consiste na escrita e submissão de artigos: (i) da avaliação experimental conduzida; (ii) sobre a proposta do processo *SMartyComponents*; e (ii) sobre o MS realizado. Além disso, a respectiva escrita e defesa desta dissertação.

1.5 Organização do Trabalho

Este trabalho está organizado da seguinte forma: o Capítulo 2 apresenta uma revisão bibliográfica com a fundamentação teórica sobre o processo *UML Components*, LPS e variabilidade e os conceitos de Arquitetura de LPS; o Capítulo 3 apresenta a evolução da abordagem *SMarty* para componentes, bem como exemplos da sua aplicação, além da avaliação experimental realizada; o Capítulo 5 apresenta a proposta do *SMartyComponents*, além de exemplos de sua aplicação; o Capítulo 6 apresenta o estudo empírico qualitativo, referente à proposta do *SMartyComponents*, bem como as possíveis melhorias identificadas; e no Capítulo 7 são apresentadas as conclusões e as contribuições da pesquisa, as limitações percebidas, além de sugestões para trabalhos futuros.

2

Fundamentação Teórica

"Suba o primeiro degrau com fé. Não é necessário que você veja toda a escada. Apenas dê o primeiro passo."

Martin Luther King Jr.
(1929 - 1968),
Pastor e Ativista Político

2.1 Considerações Iniciais

Neste capítulo é apresentado a revisão de alguns conceitos básicos sobre o referencial teórico necessário para a compreensão desta pesquisa. Assim, na Seção 2.2 é apresentado o processo *UML Components*, com ênfase nos seus *workflows* de Requisitos e Especificação; na Seção 2.3 é apresentado os conceitos de Linha de Produto de Software (LPS) e gerenciamento de variabilidades; na Seção 2.4 é apresentado os conceitos sobre Arquitetura de LPS; e na Seção 2.5 é apresentado os conceitos básicos sobre a abordagem *SMarty*.

2.2 DBC e o Processo *UML Components*

Um componente é uma unidade de composição reutilizável com interfaces fornecidas e requeridas explicitamente, e atributos de qualidade, que denota uma única abstração e

pode ser composto sem modificação (Gross, 2005). De acordo com Szyperski (1998), componente é uma unidade de composição que possui interfaces como contratos especificados e com dependências somente de contexto, que podem ser implementados de forma independente e está sujeito a composição por terceiros. Um componente fornece acesso às suas funcionalidades através de suas interfaces (Brown, 2000). A Figura 2.1 ilustra o conceito de relacionamento entre componentes por meio dos contratos de suas interfaces. Neste exemplo, o componente B possui contratos com os componentes A e C. Uma interface fornecida possui os serviços que uma interface requerida necessita, formando, assim, o conceito de contratos. Ainda, pode-se perceber que os componentes A e B possuem portas, que são uma das maneiras de fornecer acesso do ambiente externo ao ambiente interno de um componente (OMG, 2015b).

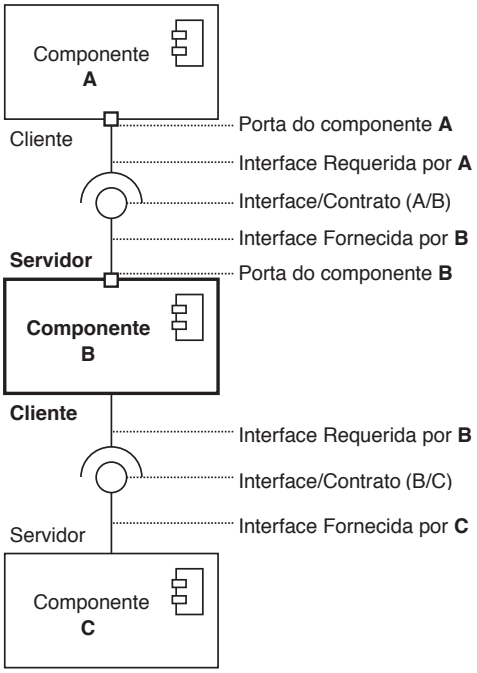

Figura 2.1: Representação de Componentes UML com Interfaces Fornecidas e Requeridas - Adaptado de Gross (2005).

De acordo com Gross (2005), componente é o bloco de construção fundamental no DBC. A principal função do DBC é a construção de novos produtos de software através da reutilização de peças facilmente disponíveis, ao invés de desenvolvê-las do início, para formar a solução de um sistema (Gross, 2005; Lewis *et al.*, 2009). Além disso, o DBC oferece benefícios como o potencial *time-to-market* mais rápido e reduzir os custos da manutenção e integração do sistema (Cesare *et al.*, 2006). Existem diversas abordagens

de DBC na literatura, porém, o *UML Components* (Cheesman e Daniels, 2001) se destaca por permitir a identificação dos componentes nos estágios iniciais.

O *UML Components* é um processo de DBC com o apoio da linguagem UML para a modelagem de seus artefatos de software. Seus conceitos de componentes e modelos de processos são extraídos de algumas abordagens como, por exemplo, o Processo Unificado (PU) (Jacobson *et al.*, 1999), e o *Catalysis* (D'Souza e Wills, 1999) (Cheesman e Daniels, 2001).

UML Components possui seis *workflows*, sendo que os *workflows* de Requisitos, Teste e Implantação, correspondem aos *workflows* do PU, porém o de Requisitos sofre algumas modificações. Os *workflows* de Especificação, Provisionamento e Montagem, substituem os *workflows* de Análise, Projeto e Implementação do PU (Cheesman e Daniels, 2001). O *workflow* de Especificação é considerado o mais importante, pois especifica a estrutura dos componentes e seus relacionamentos, além de preparar os componentes para os próximos *workflows*. A Figura 2.2 apresenta os *wokflows* do *UML Components*, as interações entre eles e seus artefatos de entrada e de saída.

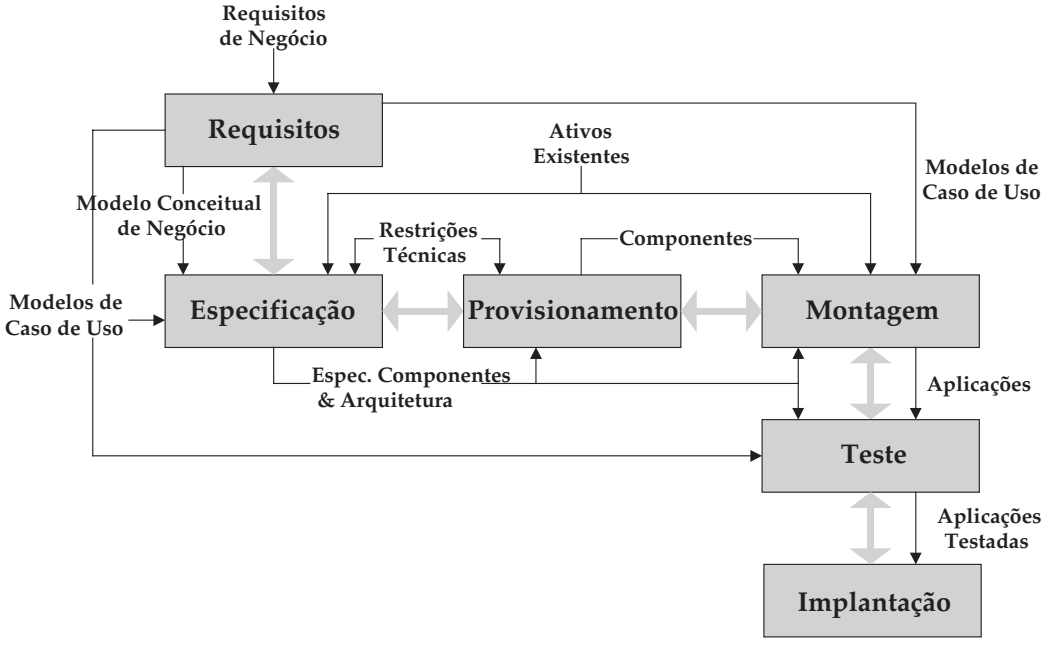

Figura 2.2: *Workflows* do UML Components - Traduzido de Cheesman e Daniels (2001).

Os artefatos que compõe o *UML Components* são:

- **Requisitos de Negócio (*Business Requirements*)**: as atividades de um determinado domínio que deverão estar contidas no produto final. Esse artefato é entrada para o *workflow* de Requisitos;

- **Modelo Conceitual de Negócio (*Business Concept Models*)**: cria um vocabulário comum entre os *stakeholders* do projeto. Esse artefato é produzido pelo *workflow* de Requisitos e é entrada para o *workflow* de Especificação;
- **Modelo de Casos de Uso (*Use Case Models*)**: descreve as possíveis interações entre os usuários e sistema. Esse artefato é produzido pelo *workflow* de Requisitos e é entrada para os *workflows* de Especificação, Montagem e Teste;
- **Restrições Técnicas (*Technical Constraints*)**: Modelo arquitetural ou ferramentas já definidas. Se disponível, esse artefato é entrada para os *workflows* de Especificação e Provisionamento;
- **Ativos Existentes (*Existing Assets*)**: recursos já existentes do domínio. Se disponível, esse artefato é entrada para os *workflows* de Especificação e Montagem;
- **Especificações de Componentes e Arquitetura (*Component Spec. & Architecture*)**: especifica as interfaces requeridas e fornecidas dos componentes e define as suas interações na arquitetura. Este artefato é produzido pelo *workflow* de Especificação e é entrada para os *workflows* de Provisionamento, Montagem e Teste;
- **Componentes (*Components*)**: determina quais componentes farão parte da solução, sendo eles desenvolvidos ou adquiridos de terceiros. Este artefato é produzido pelo *workflow* de Provisionamento e é entrada para o *workflow* de Montagem;
- **Aplicações (*Applications*)**: define a combinação dos componentes que formam as aplicações. Este artefato é produzido pelo *workflow* de Montagem, e em seguida é disponibilizado para o *workflow* de Teste; e
- **Aplicações Testadas (*Tested Applications*)**: As aplicações desenvolvidas são testadas e disponibilizadas para o *workflow* de Implantação.

Cada *workflow* do processo *UML Components* recebe artefatos de entrada e produz artefatos de saída. A seguir, tais *workflows* são apresentados:

- O primeiro *workflow* que inicia o processo de desenvolvimento é o de **Requisitos**, que recebe como entrada os `Requisitos de Negócio`, que é a descrição dos requisitos do sistema, e produz o `Modelo de Casos de Uso` e o `Modelo Conceitual de Negócio`;

- O segundo *workflow* é o de **Especificação**, que recebe como entrada o `Modelo de Casos de Uso`, o `Modelo Conceitual de Negócio`, os `Ativos Existentes` e as `Restrições Técnicas`, e produz um conjunto de `Especificações de Componentes e Arquitetura`;

- O terceiro *workflow* é o de **Provisionamento**, que recebe como entrada as `Restrições Técnicas` e as `Especificações de Componentes e Arquitetura`, e produz os `Componentes` que serão montados ou identifica quais `Componentes` de terceiros são necessários e quais serão reutilizados;

- O quarto *workflow* é o de **Montagem**, que recebe como entrada o `Modelo de Casos de Uso`, os `Ativos Existentes`, as `Especificações de Componentes e Arquitetura` e os `Componentes` gerados pelo *workflow* de **Provisionamento**, e produz as `Aplicações`;

- O quinto *workflow* é o de **Teste**, que recebe como entrada as `Aplicações` e as `Especificações de Componentes e Arquitetura`, e produz as `Aplicações Testadas` para serem implantadas; e

- O sexto e último *workflow* é o de **Implantação**, que recebe como entrada as `Aplicações Testadas` e realiza a implantação, finalizando assim o processo de desenvolvimento.

Este trabalho concentra-se nos *workflows* de Requisitos e Especificação, pois são os *workflows* essenciais para a especificação de arquiteturas de software componentizadas. Por isso, tais *workflows* e seus artefatos são descritos detalhadamente.

2.2.1 O *Workflow* de Requisitos

O *workflow* de Requisitos possui apenas uma atividade, Definição de Requisitos, e é formado por três tarefas: (i) `Desenvolver Modelo Conceitual de Negócio`; (ii) `Desenvolver Processos de Negócio`; e (iii) `Identificar Casos de Uso`. O *workflow* de Requisitos deve entregar ao *workflow* de Especificação um `Modelo de Conceito de Negócio`, que relaciona os conceitos importantes do problema do domínio mostrando as relações entre eles, e um conjunto de `Casos de Uso`, que esclarece os limites do software, identifica os atores que interagem com o sistema e descreve essas interações (Cheesman e Daniels, 2001). A Figura 2.3 apresenta uma visão geral do *workflow* de Requisitos, as tarefas que o compõe, e os respectivos artefatos de entrada e saída.

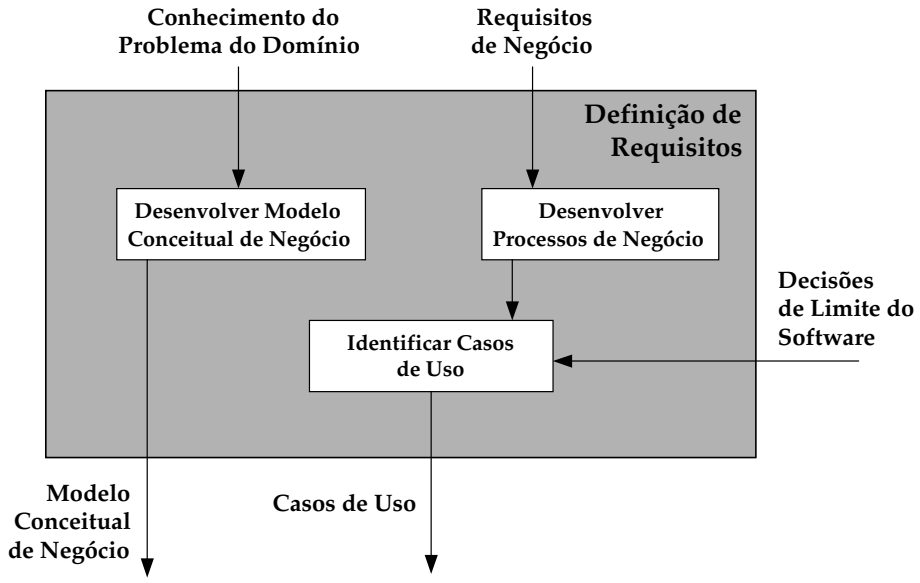

Figura 2.3: *Workflow* de Requisitos do UML Components - Traduzido de Cheesman e Daniels (2001).

2.2.2 O *Workflow* de Especificação

O *workflow* de Especificação é composto por três atividades: (i) Identificação de Componente; Interação de Componente; e Especificação de Componente. As diferentes atividades realizadas neste *workflow* levam à especificação da Arquitetura Componentizada a partir dos requisitos levantados, e detalhamento dos Componentes, Interfaces e Operações necessárias para atender às funcionalidades estabelecidas (Cheesman e Daniels, 2001). A Figura 2.4 ilustra as três atividades do *workflow* de Especificação. De acordo com Cheesman e Daniels (2001), é importante ressaltar que as atividades deste *workflow* são interativas, embora muitos artefatos apresentem dependências, o seu desenvolvimento é incremental, com inclusões e modificações a cada etapa.

A atividade Identificação de Componente é a primeira atividade do *workflow* de Especificação. A partir do *workflow* de Requisitos, ela toma como entrada os artefatos `Modelo Conceitual de Negócio` e os `Casos de Uso`. O objetivo dessa atividade é a criação de um conjunto inicial de interfaces e especificações de componentes, ligados entre si em uma arquitetura de componentes inicial (Cheesman e Daniels, 2001). Essa atividade possui quatro tarefas: (i) `Desenvolver Modelo de Tipo de Negócio`; (ii) `Identificar Interfaces de Negócio`; (iii) `Identificar Interfaces do Sistema & Operações`; e (iv) `Criar Especificação Inicial de Componente & Arquitetura`. Como resultado, é disponibilizado os artefatos `Interfaces de Negócio`, `Interfaces do`

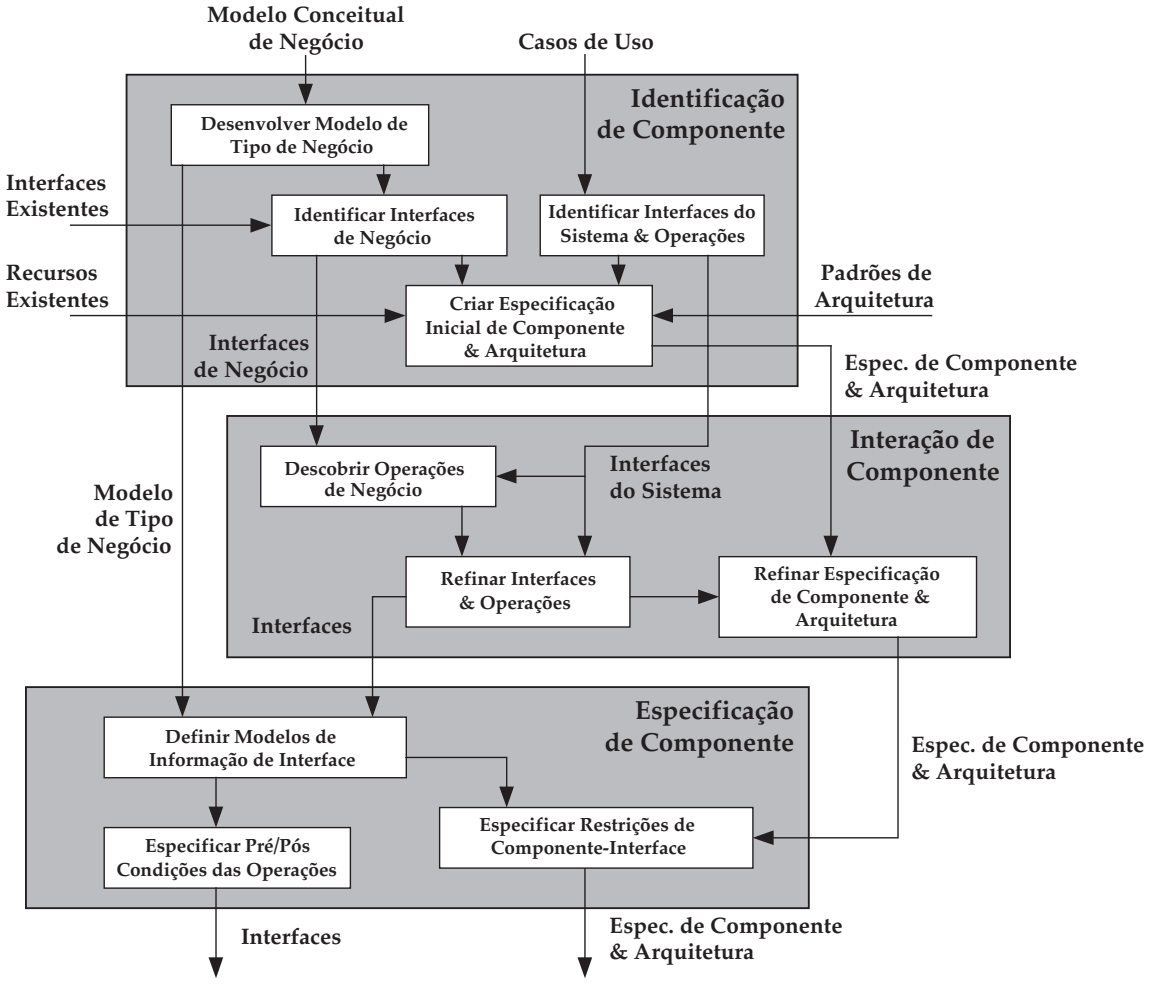

Figura 2.4: *Workflow* de Especificação do UML Components - Traduzido de Cheesman e Daniels (2001).

`Sistema` e `Especificação de Componente & Arquitetura`. Os artefatos disponibilizados pela atividade Identificação de Componente, são tomados de entrada pela atividade Interacão de Componente.

A atividade Interação de Componente realizará as interações a fim de fornecer as funcionalidades e comportamentos necessários para os componentes (Cheesman e Daniels, 2001). Nessa atividade, são realizadas três tarefas: (i) `Descobrir Operações de Negócio`; (ii) `Refinar Interfaces e Operações`; e (iii) `Refinar Especificação de Componente & Arquitetura`. Como resultado, é disponibilizado um conjunto de `Interfaces` e a `Especificação de Componente & Arquitetura` refinados.

A última atividade do `workflow` de Especificação é a Especificação de Componente, que explora o que compõe as interfaces e as examinas várias vezes nos mínimos detalhes

(Cheesman e Daniels, 2001). Essa atividade possui três tarefas: (i) `Definir Modelos de Informação de Interface`; (ii) `Especificar Pré/Pós Condições das Operações`; e (iii) `Especificar Restrições de Componente-Interface`. Por fim, essa atividade disponibiliza dois artefatos, sendo eles um conjunto de `Interfaces` e uma `Especificação de Componente & Arquitetura`.

2.3 Linha de Produto de Software e Variabilidade

Uma Linha de Produtos de Software (LPS) é um conjunto de produtos que endereçam a um determinado segmento de mercado ou missão particular (SEI, 2012). A abordagem LPS permite a geração de produtos específicos, com base na reutilização de uma infraestrutura central, formada por uma arquitetura de software e seus componentes, denominada Núcleo de Artefatos (*Core Assets*) (Linden *et al.*, 2007; Pohl *et al.*, 2005).

O *Software Engineering Institute* (SEI), por meio da abordagem *Product Line Practice* (PLP) (Clements e Northrop, 2002), estabelece atividades essenciais de LPS, sendo elas o Desenvolvimento do Núcleo de Artefatos (*Core Assets Development*), Desenvolvimento do Produto (*Product Development*) e o Gerenciamento de LPS (*Management*) (SEI, 2012).

A atividade de Desenvolvimento do Núcleo de Artefatos interage com a atividade de Desenvolvimento do Produto, gerando os produtos específicos de forma sistemática com suporte da atividade de Gerenciamento, que engloba a gerência técnica e organizacional (SEI, 2012). A atividade de Gerenciamento deve estabelecer como é feita tal interação, a fim de permitir a evolução da LPS e o gerenciamento das similaridades e variabilidades de cada artefato da LPS.

Uma das atividades mais importantes de LPS é o Gerenciamento de Variabilidades (GV), que define como os produtos se diferenciam entre si e como eles podem ser gerados. Uma variabilidade é identificada por meio de pontos de variação e variantes (Pohl *et al.*, 2005):

- **Ponto de variação**: é a resolução de variabilidades em artefatos genéricos de uma LPS. De acordo com Jacobson *et al.* (1997), "um ponto de variação identifica um ou mais locais onde a variação irá ocorrer";
- **Variante**: representa os possíveis elementos que compõem uma variabilidade, pelos quais um ponto de variação pode ser resolvido. Também pode representar uma maneira de resolver uma variabilidade diretamente.

Para ocorrer a resolução de um ponto de variação, uma ou mais variantes devem ser selecionadas, sendo que tais variantes são impostas por meio de restrições, que definem os seus relacionamentos (Linden *et al.*, 2007), em seu devido tempo de resolução.

O tempo de resolução de uma variabilidade pode ser tempo de projeto, compilação, de ligação, de execução e de atualização (Linden *et al.*, 2007). Neste trabalho, as variabilidades são resolvidas em tempo de projeto (*design*).

A literatura existente apresenta várias abordagens de GV, especialmente para modelos UML, como, por exemplo, *Stereotype-based Management of Variability* (*SMarty*) (Seção 2.5). *SMarty* é uma abordagem anotativa de GV em modelos UML, formada por um perfil denominado *SMartyProfile*, que é um conjunto de estereótipos para representar variabilidades, e um processo denominado *SMartyProcess*, onde os artefatos de saída de um determinado processo são tomados de entrada, e por meio de suas diretrizes, guia os *stakeholders* a identificar e representar as variabilidades.

Com o intuito de identificar como a variabilidade veem sendo modelada em arquiteturas de software, um Mapeamento Sistemático (MS) (Apêndice A) foi realizado. Para isso, alguns dos principais motores de busca foram utilizados, e por meio do processo de busca definido, os estudos recuperados foram analisados e organizados de modo com que as suas informações fossem classificadas de várias maneiras, por exemplo: (i) as bases de busca e quantidade de estudos recuperados classificados em um gráfico de setores; (ii) os autores com maior número de publicações encontradas com base nos filtros de busca, representados em gráficos; (iii) classificação em formato *Word Cloud* da distribuição dos estudos por locais de publicação; (iv) classificação da quantidade de estudos recuperados em relação ao ano de publicação; (v) classificação em formato *Bubble Plot* dos resultados obtidos em relação às Fontes de Busca x Classificação das Buscas x Contexto de Variabilidades; (vi) entre outras formas e formatos de classificação, além de uma discussão sobre o conjunto final de estudos mais relevantes.

Do conjunto final de estudos relevantes do MS (39), um subconjunto de 6 estudos foi escolhido para ser analisado e comparado à abordagem *SMarty* com relação à modelagem de variabilidades em componentes. Tais estudos foram selecionados por permitirem a representação de forma anotativa em diagrama de componentes e seus elementos, tornado explícita tal representação. A Tabela 2.1 mostra tais estudos e algumas das suas respectivas informações, tais como as ferramentas que utilizam em cada estudo, suporte à UML e sua versão, elementos que suportam variabilidade e se possui alguma forma de identificar a rastreabilidade entre os elementos que suportam variabilidade.

Diante dos dados de cada estudo, segue um resumo de acordo com a Tabela 2.1:

Tabela 2.1: Abordagens de GV que permitem Representar Variabilidades em Componentes.

Estudo	Abordagem	Suporte a UML	Versão da UML	Elementos que Modelam	Rastreabilidade
(Choi *et al.*, 2005)	Extensão da UML 2.0	Sim	2.0	Componentes e Conectores	Não
(Tekinerdogan e Sözer, 2012)	*Variability Viewpoint*	Não	–	Componentes e Pacotes	Não
(Satyananda *et al.*, 2007)	*C&C View* + FCA	Não	–	Componentes e Conectores	Sim
(Razavian e Khosravi, 2008)	*C&C View*	Sim	2	Componentes, Conectores e Interfaces	Não
(Gomaa, 2013)	PLUS	Sim	2.0	Componentes	Não
(Ryu *et al.*, 2012)	PLUS + OVM	Sim	2.0	Componentes	Não
`SMarty 5.2` (Marcolino e OliveiraJr, 2015)	Extensão da UML 2.0	Sim	2.4	Componentes	Sim

- Choi *et al.* (2005) representam variabilidades por meio de estereótipos em componentes e conectores da UML 2.0. Para isso, usa os seguintes estereótipos que representam variabilidades, sendo que: (i) ≪variation point≫ representa um ponto de variação; (ii) ≪variant≫ representa uma alternativa a ser escolhida, contidas em um determinado ponto de variação; (iii) ≪optional≫ representa um conector opcional; e (iv) ≪optional component≫ representa um componente opcional. Porém, esse estudo não permite representar variabilidades em portas e interfaces.

- Tekinerdogan e Sözer (2012) identificam e representam variabilidades utilizando estereótipos em grupos de elementos, que são identificados dentro de um quadrado com linha pontilhada, e utiliza elementos gráficos para representação de variabilidades nos relacionamentos dos elementos em si. Possui dois estereótipos, sendo eles: (i) ≪Alternative≫ que representa um conjunto de variantes alternativas; e (ii) ≪Optional≫ que representa um elemento opcional. Esse estudo, porém, não suporta UML.

- Satyananda *et al.* (2007) representam as variabilidades por meio de elementos gráficos, que representam componentes, conectores e variabilidades. Não possui nenhuma forma de representação compatível com a UML, nem permitem representar variabilidades em portas e interfaces.

- (Razavian e Khosravi, 2008) propõem uma solução para modelar variabilidade em componentes, conectores e interfaces. Para representar os pontos de variação nesses elementos, utiliza-se os seguintes estereótipos: (i) ≪alt vp≫ para os pontos de variação do tipo alternativo exclusivo; (ii) ≪opt vp≫ para os pontos de variação do tipo opcionais; (iii)≪optional≫ são elementos que podem ou não existir; (iv) ≪optv vp≫ para pontos de variação de variantes opcionais; (v) ≪altv vp≫ para pontos de variação de variantes alternativas; e (vi) ≪variant≫ representa as variantes associadas a um determinado ponto de variação. Esse estudo não permite a representação de variabilidade em portas lógicas, nem o rastreamento dos elementos modelados.

- Gomaa (2013) estende o método PLUS baseado em UML para resolver modelagem de caso de uso considerando características evolutivas e projeto arquitetural de software. Para a modelagem de arquiteturas baseadas em componentes, os seguintes estereótipos são utilizados: (i) ≪kernel≫ representa obrigatoriedade; (ii) ≪variant≫ representa variantes; e (iii) ≪optional≫ representa elementos opcionais. Esse estudo não apoia a modelagem de variabilidade em portas, interfaces e conectores UML.

- Ryu *et al.* (2012) utilizam o método PLUS + OVM (*Orthogonal Variability Model*) para representar variabilidades em componentes. Os estereótipos ≪kernel≫ e ≪optional≫ são utilizados para representar os componentes obrigatórios e opcionais, respectivamente. Os modelos gráficos OVM indicam um ponto de variação e relaciona quais componentes são as variantes. Esse estudo não apoia a modelagem de variabilidade em portas, interfaces e conectores UML.

Em resumo, a análise levantada comparando tais estudos, indica que o estudo de Razavian e Khosravi (2008) é considerado o mais adequado a esta pesquisa. Como critério de seleção, tal análise levou em consideração a quantidade de estereótipos, o suporte e a versão da UML, os níveis de representatividade das variabilidades nos elementos dos modelos de componentes e também o ano de publicação do estudo. Assim, tal escolha leva a acreditar que a forma como esse trabalho representa variabilidades nesses elementos em diagramas de componentes da UML, pode contribuir com a evolução da

abordagem *SMarty*, o que é essencial para esta pesquisa. Além disso, tal estudo permite representar variabilidade em vários elementos arquiteturais. A Figura 2.5 apresenta um exemplo de modelagem de variabilidade em arquitetura de acordo com a abordagem de Razavian e Khosravi. Neste exemplo, o conector `Media Stream Protocol` está anotado com o estereótipo ≪alt_vp≫ como sendo um ponto de variação que possui duas variantes mutualmente exclusivas, `High BW MS` e `Low BW MS`, anotadas com o estereótipo ≪variant≫. Note que o estereótipo ≪connector≫ é usado para representar um conector, mas não é uma prática formal da UML.

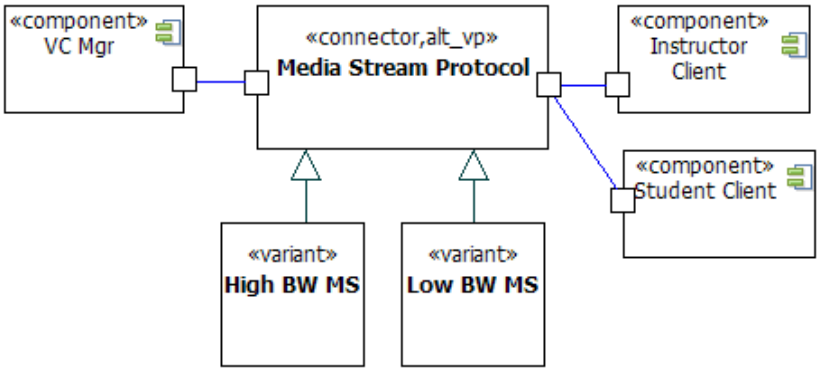

Figura 2.5: ALPS `Virtual University`: Representação de Variabilidade em *Bandwidth* (Razavian e Khosravi, 2008).

2.4 Arquitetura de LPS

A Arquitetura de LPS (ALPS) é um dos artefatos mais importantes de uma LPS (Linden *et al.*, 2007; OliveiraJr *et al.*, 2013a), pelo fato de representar a abstração de todas as possíveis arquiteturas de produtos específicos gerados de tal LPS. A ALPS serve como base para a geração de diferentes produtos de uma LPS (OliveiraJr *et al.*, 2013a).

Alguns requisitos importantes de uma ALPS são (Taylor *et al.*, 2009):

- manter-se estável durante a vida de LPSs, sofrendo o mínimo de alterações possíveis;
- fácil integração de novas características durante o ciclo de vida da arquitetura; e
- representar de forma explícita as variações de modo a proporcionar a reutilização.

A quantidade de diferenças ou de dependências entre os produtos de uma LPS refletirá diretamente na complexidade dessa arquitetura (Jazayeri *et al.*, 2000). Uma

arquitetura deve fornecer uma visão geral onde possam ser identificadas as variabilidades e as similaridades de uma LPS em termos de artefatos e suas configurações.

Um processo de software bem definido, com papéis, artefatos de entrada e saída, e tarefas, é importante para auxiliar a especificação coesa de uma ALPS (Linden *et al.*, 2007; Pohl *et al.*, 2005). A literatura existente apresenta várias abordagens de desenvolvimento de LPS, como, por exemplo, PLP (Clements e Northrop, 2002), PLUS (Gomaa, 2013), BAPO, ESAPS, CAFÉ, FAMILIES (Linden *et al.*, 2007), porém, estas não dizem explicitamente como obter uma ALPS com variabilidades bem definidas e explícitas. Uma vez que se tem uma ALPS bem definida, sua avaliação pode ser utilizada como parâmetro para avaliar a qualidade de uma LPS (OliveiraJr *et al.*, 2013a).

A importância de uma ALPS ser baseada em componentes se dá por definir um ambiente único para todos os componentes que serão utilizados nos produtos individuais. Isto garante que não haverá a necessidade de desenvolver vários componentes que abordam funcionalidades semelhantes, e diferem apenas no que diz respeito ao ambiente que trabalham (Linden *et al.*, 2007). Além disso, pode-se usar das técnicas já existentes em DBC para garantir a qualidade de uma LPS por meio de sua arquitetura.

Para garantir que a arquitetura de cada produto gerado por uma LPS seja viável, a ALPS deve ser avaliada. De acordo com OliveiraJr *et al.* (2013a), as instâncias da ALPS devem ser avaliadas para garantir que estão de acordo com os requisitos funcionais e de qualidade. Segundo Clements e Northrop (2002) e Taylor *et al.* (2009), a avaliação de LPS deve se concentrar nas variabilidades, garantindo assim, que: (i) sejam apropriadas; (ii) ofereçam flexibilidade suficiente para o escopo da LPS; (iii) possam ser exercidas derivando os produtos específicos; e (iv) não imponham custos de desempenho inaceitáveis em tempo de execução.

A avaliação deve ser aplicada ao núcleo de artefatos, principalmente à ALPS (OliveiraJr *et al.*, 2013a). ALPSs componentizadas podem ser utilizadas como forma de avaliar a LPS como um todo.

2.5 A Abordagem *SMarty* para Gerenciamento de Variabilidades

De acordo com os conceitos essenciais apresentados sobre LPS (Seção 2.3) e gerenciamento de variabilidades, a abordagem *Stereotype-based Management of Variability* (*SMarty*) foi proposta no estudo de OliveiraJr *et al.* (2010a), na qual possui suporte, em sua versão 4.0, a modelos UML de casos de uso, classes, atividades e componentes.

Tal abordagem (*SMarty*) tem o principal objetivo de permitir que as variabilidades de uma LPS possam ser gerenciadas de maneira efetiva em modelos UML. Para isto, a *SMarty* foi avaliada por meio de um conjunto de experimentos conduzidos e apresentados nos estudos realizados por Marcolino *et al.* (2013b), Marcolino *et al.* (2014a), (Marcolino *et al.*, 2014b), OliveiraJr *et al.* (2013a) e OliveiraJr *et al.* (2010a).

Adicionalmente, as avaliações experimentais conduzidas por Marcolino *et al.* (2014a, 2013b) permitiram que a *SMarty* fosse estendida de forma efetiva para suportar modelos UML de sequência em seu escopo, como proposto no estudo de Marcolino (2014). Deste modo, a *SMarty* evoluiu para a versão 5.1.

Assim, para cada modelo UML suportado pela *SMarty* uma versão (1.0) é acrescentada, ou seja, a *SMarty* suporta cinco modelos UML: casos de uso (1.0), classes (2.0), atividades (3.0), componentes (4.0) e sequência (5.0), os quais equivalem a versão 5.1 da *SMarty*.

A abordagem *SMarty* é composta de um Perfil UML, o *SMartyProfile*, e um Processo, o *SMartyProcess* (Fiori *et al.*, 2012; OliveiraJr *et al.*, 2010a, 2013a):

- ***SMartyProfile***: permite a representação gráfica de variabilidades por meio da UML e seu mecanismo de perfil, o qual é formado por um conjunto de estereótipos e meta-atributos (*tagged values*) utilizados para representar variabilidades em modelos UML de LPS; e

- ***SMartyProcess***: é um processo sistemático, no qual possui um conjunto de diretrizes associadas com o objetivo de guiar o usuário na identificação, representação, e rastreamento de variabilidades de uma LPS (OliveiraJr *et al.*, 2010a).

O Perfil *SMartyProfile*

O *SMartyProfile* baseia-se no inter-relacionamento dos principais conceitos de LPS, no qual engloba o gerenciamento de variabilidades (Seção 2.3). Com base na relação entre estes conceitos de gerenciamento de variabilidades e os modelos UML, é permitido que estes conceitos sejam utilizados aos elementos de interesse do metamodelo da UML.

Deste modo, o perfil UML *SMartyProfile* 5.1 possui um conjunto de estereótipos e meta-atributos (*tagged values*), os quais são apresentados na Figura 2.6, bem como são listados e descritos de forma sucinta a seguir:

- <<**variability**>> representa o conceito de variabilidade, no qual é uma extensão da metaclasse UML *Comment*. Este estereótipo é aplicado apenas em notas UML e é constituído dos seguintes meta-atributos:

  - `name`, nome de referência para uma variabilidade;
  - `minSelection`, denota a quantia mínima de variantes a serem selecionadas para resolver um ponto de variação ou variabilidade;
  - `maxSelection`, denota a quantia máxima de variantes a serem selecionadas para resolver um ponto de variação ou variabilidade;
  - `bindingTime`, delimita o instante no qual uma variabilidade será resolvida. Este tempo no qual acontecerá está representado pela classe de enumeração *BindingTime*;
  - `allowsAddingVar`, define se existe a possibilidade de inserção de novas variantes após uma variabilidade ser resolvida;
  - `variants`, apresenta a coleção de instâncias associada à variabilidade; e
  - `realizes`, representa a coleção de variabilidades de modelos de nível inferior, no qual é possível realizar a variabilidade.

- <<**variationPoint**>> representa o conceito de ponto de variação, no qual é uma extensão das metaclasses UML: *DecisionNode*, *Actor*, *UseCase*, *Interface*, e *Class*. Sendo assim, tal estereótipo é empregado somente a nós de decisão, atores, casos de uso, interfaces e classes. Este estereótipo é constituído dos seguintes meta-atributos:
  - `numberOfVariants`, indica o número de variantes associadas que podem ser selecionadas para resolver esse ponto de variação;
  - `bindingTime`, define o instante no qual um ponto de variação será resolvido. Este tempo no qual ocorrerá está representado pela classe de enumeração *BindingTime*;
  - `allowsAddingVar`, define se existe a possibilidade de inserção de novas variantes após uma variabilidade ser resolvida;
  - `variants`, representa a coleção de instâncias das variantes, as quais estão associadas a esse ponto de variação; e
  - `variabilities`, representa a coleção de variabilidades, nas quais esse ponto de variação está associado.

- <<**variant**>> engloba o conceito de variante, no qual é uma extensão abstrata das metaclasses UML: *Actor*, *UseCase*, *Interface*, *Class*, *Action*, *ActivityPartition*, *Component*, *Dependency* e *Comment*. Tal estereótipo é abstrato, o qual não pode ser aplicado em nenhum elemento UML. No entanto, suas especializações não

abstratas podem ser aplicadas em atores, casos de uso, interfaces, classes, ações, partição de atividade, componente, dependência, comentário, fragmento combinado, mensagem e pacotes. Portanto, suas especializações, não abstratas abrangem os seguintes estereótipos: <<mandatory>>, <<optional>>, <<alternative_OR>> e <<alternative_XOR>>. O estereótipo <<variant>> é constituído dos seguintes meta-atributos:

- `rootVP`, representa o ponto de variação ao qual está associado; e
- `variabilities`, coleção de variabilidades com as quais esta variante está associada.

- <<**mandatory**>> representa uma variante obrigatória que está presente em todos os produtos de uma LPS;
- <<**optional**>> representa uma variante que pode ser escolhida para resolver uma variabilidade ou um ponto de variação;
- <<**alternative_OR**>> representa uma variante que faz parte de um grupo de variantes inclusivas. Assim, diferentes combinações dessas variantes podem resolver pontos de variação de diferentes modos, gerando dessa forma, produtos distintos;
- <<**alternative_XOR**>> denota uma variante que pertence há um grupo de variantes exclusivas, na qual somente uma variante do grupo pode ser selecionada para resolver um ponto de variação;
- <<**mutex**>> significa o conceito de restrição "Exclusão Mútua", sendo um relacionameto mutuamente exclusivo entre variantes. Isto significa que para uma variante ser selecionada, a variante relacionada não pode ser selecionada;
- <<**requires**>> manifesta o conceito de restrição "Complemento", sendo um relacionamento entre variantes, no qual a variante escolhida requer a seleção da variante relacionada; e
- <<**variable**>> é uma extensão da metaclasse UML *Component*. Tal estereótipo aponta que um componente é formado por um conjunto de classes com variabilidades explícitas. Este estereótipo possui o meta-atributo `classSet`, no qual é a coleção de instâncias das classes variáveis que formam o componente.

Em adição, o perfil *SMartyProfile* possui características próprias para representar relações de cada modelo UML suportado entre pontos de variação e suas variantes,

nas quais, estas relações entre as variantes, são representadas por meio da utilização de meta-atributos. Além disso, a *SMarty* possibilita a extensão dos elementos que pertencem ao metamodelo da UML, diferentemente de outras abordagens, que não o fazem.

Dessa forma, o perfil *SMartyProfile* aplica os conceitos de pontos de extensão da UML em seus modelos, o que o torna compatível com os metamodelos da UML, bem como no uso de ferramentas automatizadas ou de apoio, nas quais manipulam arquivos XML *Metadata Interchange* (XMI), abrangendo modelos UML.

O Processo *SMartyProcess*

O *SMartyProcess* adota suas atividades comuns relacionadas às definidas no processo de desenvolvimento de LPS (Pohl *et al.*, 2005) (SEI, 2012). Este relacionamento é representado na Figura 2.7 por meio de um diagrama de atividades da UML, no qual mostra o processo genérico de Desenvolvimento de Linha de Produto, ilustrado pelas atividades alinhadas no lado esquerdo e o *SMartyProcess* simbolizado pelas atividades do retângulo à direita (OliveiraJr *et al.*, 2005).

No estudo de OliveiraJr *et al.* (2010b), definições formais e estereótipos foram propostos para a *SMarty* por meio do *SMartyProfile*. Com isso, de maneira orquestrada, a *SMarty* faz a combinação do perfil *SMartyProfile* e do processo *SMartyProcess*, formando uma abordagem guiada por diretrizes para gerenciar sistematicamente variabilidades de LPS.

O processo *SMartyProcess* é realizado pelo engenherio de LPS e é um processo iterativo e incremental. Assim, quando o *SMartyProcess* ocorre após a execução de cada atividade do desenvolvimento de LPS, o mesmo é iterativo e quando a quantidade de variabilidades tende a crescer, à medida que as atividades do *SMartyProcess* são executadas, o mesmo é incremental.

O *SMartyProcess* absorve os elementos do núcleo de artefatos de uma LPS, bem como o alimenta com outros. Assim, por meio de cada ciclo, entre o Desenvolvimento de LPS e o *SMartyProcess*, pode-se identificar progressivamente as variabilidades associadas para os modelos de casos de uso, classes, atividades, componentes e sequência, utilizando a atividade de identificação de variabilidades, na qual recebe como entrada cada um destes modelos. Entretanto, para realizar a identificação de variabilidades é necessário um conhecimento específico de analistas e gerentes de LPS, pois tal atividade que depende fortemente do domínio. Perante a isto, o *SMartyProcess* fornece um conjunto de diretrizes com o objetivo de apoiar e realizar com sucesso a atividade de gerenciamento de variabilidades (Fiori *et al.*, 2012; OliveiraJr *et al.*, 2010a, 2013a).

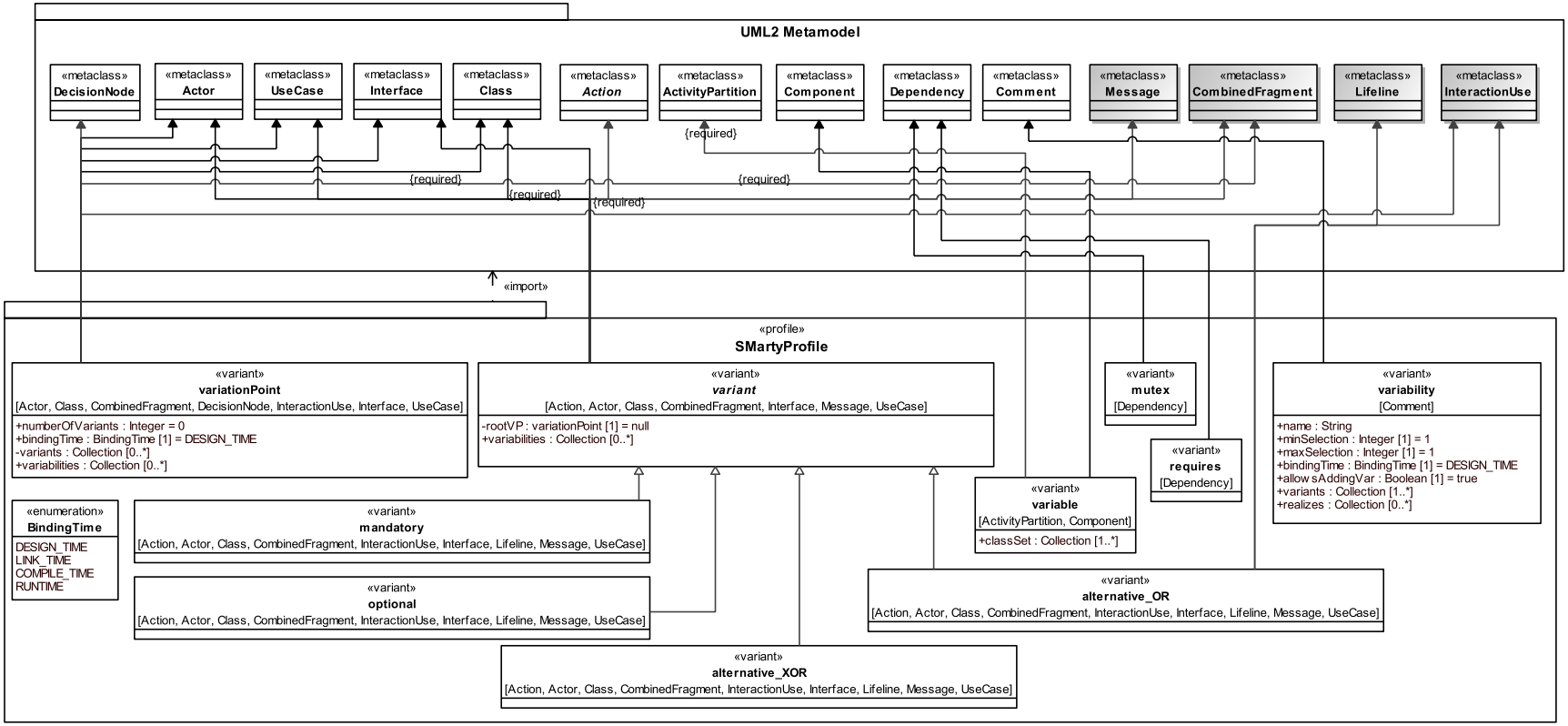

Figura 2.6: Estereótipos e Meta-Atributos do Perfil *SMartyProfile* 5.1 com Suporte a Modelos de Casos de Uso, Classes, Componentes, Atividades e Sequência (Fiori *et al.*, 2012; Marcolino, 2014; OliveiraJr *et al.*, 2010a, 2013a).

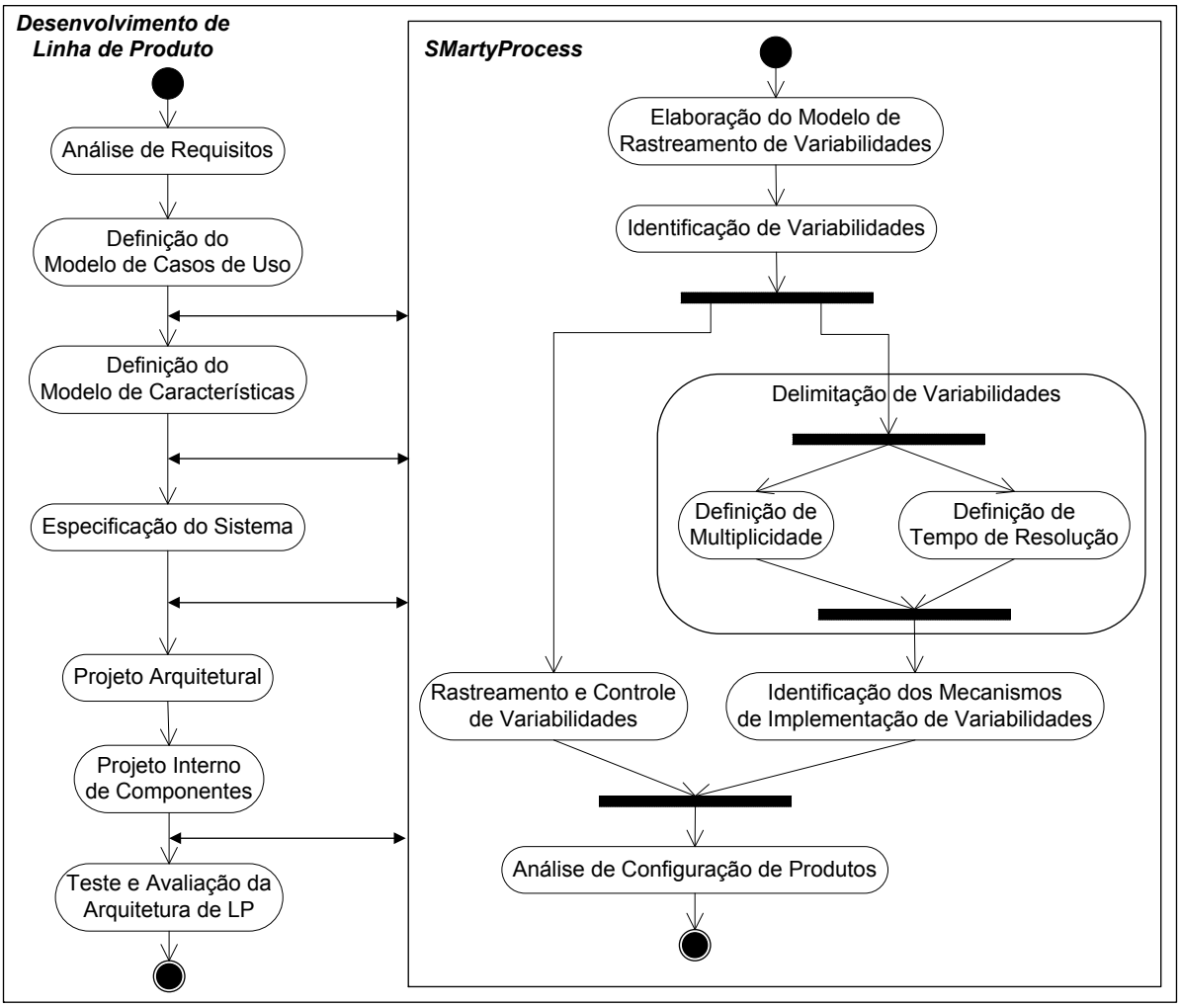

Figura 2.7: O Processo de Gerenciamento de Variabilidades *SMartyProcess* e sua Interação entre as Atividades com o Processo de Desenvolvimento de LPS, traduzido de (OliveiraJr *et al.*, 2005, 2010b).

Por conseguinte, as diretrizes gerais do *SMartyProcess* são apresentadas para a identificação e representação de variabilidades aplicando o *SMartyProfile*. Além disso, na sequência, diretrizes específicas são descritas para diferentes modelos UML *SMarty* de casos de uso, classes, atividades, componentes e sequência (Marcolino, 2014; OliveiraJr *et al.*, 2010a, 2013a, 2010b).

É importante mencionar, que todas as diretrizes foram reescritas na íntegra dos estudos de Marcolino (2014) e OliveiraJr *et al.* (2010b), com o objetivo de manter a descrição correta, bem como a organização e o padrão estabelecido para estas diretrizes.

- **Diretrizes para Identificação e Representação de Variabilidades (RV)**

RV.1 Variabilidades com variantes opcionais (<<optional>>) possuem multiplicidade `minSelection = 0` e `maxSelection = 1`;

RV.2 Variabilidades com variantes exclusivas (<<alternative_XOR>>) possuem multiplicidade `minSelection = maxSelection = 1`;

RV.3 Variabilidades com variantes inclusivas (<<alternative_OR>>) possuem multiplicidade `minSelection = 1` e `maxSelection = 1 = size(variants)` em que `size(x)` é uma função que retorna a quantidade de elementos da coleção `x`;

RV.4 O valor `bindingTime` deve ser definido escolhendo-se um dos valores da classe de enumeração *BindingTime*, que são: DESIGN_TIME, LINK_TIME, COMPILE_TIME e RUNTIME;

RV.5 O valor booleano do atributo `allowsAddingVar` deve ser analisado com base na possibilidade de manter o ponto de variação aberto (`true`) ou fechado (`false`); e

RV.6 O valor da coleção variantes (`variants`) é o conjunto formado pelas instâncias das variantes associadas ao ponto de variação ou variabilidade.

- **Diretrizes para Modelos de Casos de Uso (UC)**

UC.1 Elementos de modelos de casos de uso relacionados aos mecanismos de extensão e de pontos de extensão sugerem pontos de variação (<<variationPoint>>) com variantes associadas, as quais podem ser inclusivas (<<alternative_OR>>) ou exclusivas (<<alternative_XOR>>);

UC.2 Modelos de casos de uso com o relacionamento de inclusão (<<include>>) ou associados a atores sugerem variantes obrigatórias (<<mandatory>>) ou opcionais (<<optional>>);

UC.3 Variantes que, ao serem selecionadas para fazer parte de um produto, exigem a presença de outra(s) determinada(s) variante(s) devem ter seu relacionamentos de dependência marcados com o estereótipo <<requires>>;

UC.4 Variantes mutuamente exclusivas para um determinado produto devem ter seus relacionamentos de dependência marcados com o estereótipo <<mutex>>.

- **Diretrizes para Modelos de Classes (CL)**

CL.1 Em modelos de classes, pontos de variação e suas variantes são identificadas nos seguintes relacionamentos:

a) generalização, os classificadores mais gerais são os pontos de variação, enquanto os mais específicos são as variantes;

b) realização de interface, os "*suppliers*" (especificações) são os pontos de variação e as implementações (clientes) são as variantes;

c) agregação, as instâncias tipadas com losangos não preenchidos são os pontos de variação e as instâncias associadas são as variantes; e

d) composição, as instâncias tipadas com losangos preenchidos são os pontos de variação e as instâncias associadas são as variantes.

CL.2 Elementos de modelos de classes, relacionados a associações nas quais os seus atributos `aggregationKind` possuem valor *none*, ou seja, não representam, nem agregação, nem composição, sugerem variantes obrigatórias ou opcionais;

CL.3 Variantes em modelos de classes, que ao serem selecionadas para fazer parte de um produto, exigem a presença de outras determinada(s) variante(s) devem ter seus relacionamentos de dependência marcados com o estereótipo <<requires>>;

CL.4 Variantes em modelos de classes, mutuamente exclusivas para um determinado produto devem ter seus relacionamentos de dependência marcados com o estereótipo <<mutex>>.

- **Diretrizes para Modelos de Componentes (CP)**

CP.1 Componentes formados por classes com variabilidades são marcados com o estereótipo <<variable>>.

- **Diretrizes para Modelos de Atividades (AT)**

AT.1 Elementos de modelos de diagramas de atividades como *DecisionNode* sugerem pontos de variação marcados <<variationPoint>>, pois é um local formado explicitamente por possíveis caminhos para grupos de ações distintas;

AT.2 Elementos *Action* dos diagramas de atividades podem ser definidos como variantes obrigatórias ou opcionais;

AT.3 Elementos *Action* que representam fluxos alternativos de saída de um *DecisionNode* sugerem variantes alternativas inclusivas ou exclusivas;

AT.4 Elementos *ActivityPartition* que possuem elementos variáveis, *DecisionNode* como ponto de variação ou *Action* como variantes, devem ser marcados como <<variable>>, pois são compostos por elementos que sofrem algum tipo de variação.

- **Diretrizes para Modelos de Sequência (SQ)**

SQ.1 Elementos de diagramas de sequência como `CombinedFragment` que possuem do `interactionOperator` do tipo "alt" (*alternative*), indicam que apenas um fluxo do `CombinedFragment` será realizado, ou seja, sugerem variantes mutuamente exclusivas, nas quais os pontos de variação serão anotados como <<variationPoint>> e serão relacionados a um comentário da UML especificando a variabilidade (<<variability>>). As variantes correspondentes às mensagens devem ser estereotipadas como <<alternative_XOR>>;

SQ.2 Em diagramas de sequência, as duas possíveis ocorrências a seguir, sugerem variantes opcionais:

a) Elementos de diagramas de sequência como o `CombinedFragment` que possuem `interactionOperator` do tipo "opt" (*optional*) sugerem variantes opcionais, sendo estereotipados como <<optional>>, e são relacionados a um comentário da UML especificando a variabilidade (<<variability>>). As *lifelines* contidas neste `CombinedFragment` e, que fazem parte da variabilidade, deverão ser estereotipados também como <<optional>>;

b) Troca de mensagens entre dois objetos não obrigatórios, ou entre um objeto obrigatório e outro não, sugerem uma variante opcional, estereotipadas como <<optional>> e estarão relacionados a um comentário da UML especificando a variabilidade (<<variability>>). A(s) *lifeline(s)* correspondente(s) a essa variante será(ão) estereotipada(s) também como <<optional>>.

SQ.3 O elemento `interactionUse` "ref" sugere ponto de variação para variantes alternativas inclusivas, sendo estereotipado como <<variationPoint>> e relacionado a um comentário UML, que identifica os elementos da variabilidade (<<variability>>). Os diagramas de sequência referenciados pelo `interactionUse` "ref" correspondem às variantes do ponto de variação, são considerados portanto, alternativos inclusivos, podendo um ou mais serem selecionados, sendo estereotipadas como <<alternative_OR>>;

SQ.4 As mensagens (*messages*), nas quais são independentes dos fluxo contidos no `CombinedFragment` "alt", "opt", `interactionUse` "ref", ou não estejam relacionadas diretamente a uma variabilidade e seus elementos, são mantidas sem estereótipos e, consideradas assim, obrigatórias;

SQ.5 Variantes em diagramas de sequência que, ao serem selecionadas para fazer parte de um produto específico, exigirem a presença de outra(s) determinada(s) variante(s) devem ter seus relacionamentos de dependência marcados com o estereótipo <<requires>>;

SQ.6 Variantes mutuamente exclusivas de um diagrama de sequência, para um determinado produto devem ter seus relacionamentos de dependência marcados com o estereótipo <<mutex>>.

Em síntese, a abordagem *SMarty* 5.1 é ilustrada na Figura 2.8, na qual é possível observar os modelos UML suportados pela *SMarty*, um posicionamento de forma simplificada das diretrizes de cada modelo, além do processo interativo e incremental entre o perfil *SMartyProfile* e o processo *SMartyProcess*, representado pelas setas ao redor da Figura 2.8. Com isso, é possível abranger e permitir a evolução, bem como a identificação de novas variabilidades por meio dos modelos UML e suas respectivas diretrizes baseadas na *SMarty* (Marcolino, 2014).

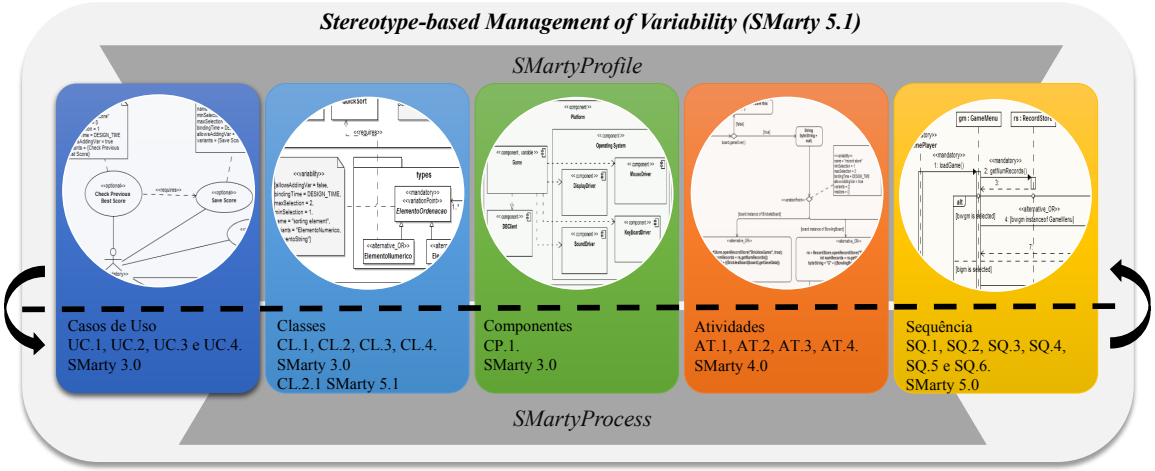

Figura 2.8: Visão Geral da *SMarty* 5.1, adaptado de (Marcolino, 2014).

2.6 Considerações Finais

Os conceitos apresentados neste capítulo são imprescindíveis para que se proponha um processo para especificar ALPS componentizadas, que é o objetivo deste estudo. Para isso acontecer, a abordagem *SMarty* precisa evoluir para dar suporte a elementos arquitetuais importantes, como os identificados por meio do MS (Apêndice A).

O *UML Components* é um processo sistemático para especificação de arquiteturas de software baseadas em componentes. Ele se destaca por auxiliar os *stakeholders* a identificar os componentes nos primeiros *workflows*. Ao final do segundo *workflow*, Especificação de Componentes, a especificação da arquitetura e dos componentes está definida, e servem de apoio para as estapas seguintes, de provisionamento, montagem, testes, e implantação.

A abordagem *SMarty* vem se destacando por permitir representar variabilidades em diferentes modelos da UML, além de possuir diretrizes bem definidas que apoiam os *stakeholders* na identificação e representação de variabilidades. *SMarty* vem sendo estabelecida (Marcolino e OliveiraJr, 2015; Marcolino *et al.*, 2014a,b) e constantemente evoluída para apoiar o GV nos diversos modelos UML disponíveis.

O próximo capítulo apresenta a evolução de *SMarty* 5.2 para representar variabilidades em diferentes níveis de abstração, como componentes, portas, interfaces e operações.

3

Evolução de *SMarty* 5.2 para Componentes, Portas, Interfaces e Operações

"O que eu faço, é uma gota no meio de um oceano. Mas sem ela, o oceano será menor."

Madre Teresa de Calcutá (1910 - 1997), Beata

3.1 Considerações Iniciais

O diagrama de componentes da UML 2.5 caracteriza-se pela representação das partes de um sistema em um alto nível de abstração, separando os módulos que podem ser substituíveis em componentes, e mantendo a representação de seus relacionamentos com os demais componentes envolvidos (OMG, 2015b). Os diagramas de classes são uma abstração em um nível mais baixo, e estão contidos, respectivamente, nos componentes abstraídos.

O diagrama de componentes é útil para proporcionar uma visão geral de como a arquitetura do sistema está logicamente organizada, permitindo identificar componentes

que necessitam ser substituídos, reutilizados, particionados (no caso de haver sobrecarga de informações) e até mesmo adquiridos de terceiros (OMG, 2015b).

O diagrama de componentes, na versão 2.5 da UML, fornece elementos (portas, interfaces e compartimentos) que são favoráveis à representação de variabilidades e que podem ser representados como artefatos, compondo o núcleo de uma ALPS.

Este capítulo apresenta a evolução da abordagem *SMarty* 5.1 por meio da evolução do seu suporte a componentes, gerando um *release* (Coupaye e Estublier, 2000) da versão 5.2. Tal evolução se refere à identificação e representação de variabilidades em componentes, portas, interfaces e operações segundo a versão 2.5 da UML. Exemplos de aplicação são apresentados em cada nível de identificação e representação de variabilidades com base na LPS *Mobile Media* (MM) (Young, 2005). Até então, a versão 5.1 de *SMarty* usava um único estereótipo para componentes, o «`variable`», indicando que um componente possui variabilidade em suas classes.

3.2 *SMarty 5.2*

A evolução de *SMarty* aborda mudanças em alguns elementos do diagrama de componentes da versão 2.5 da UML, como componentes e seus compartimentos para portas, interfaces e operações, visando a representação de variabilidades. Tais elementos foram analisados e agregados para compor a evolução. Assim, pode-se perceber que os compartimentos disponíveis nos classificadores (classes, componentes e interfaces) suportam a representação de estereótipos, o que torna-se uma vantagem para *SMarty* representar variabilidades de forma anotativa por meio dos estereótipos disponíveis no *SMartyProfile*. Portanto, esses elementos foram identificados como candidatos a ponto de variação e variantes.

Na versão da UML 2.5, os itens relacionados com os classificadores, podem ser listados em compartimentos. No caso do classificador de componente, suas portas, interfaces, atributos, operações, realizações e demais partes, podem ser referenciadas por meio dos compartimentos (OMG, 2015b). Sendo assim, em alguns casos, os estereótipos das portas e operações serão representados em seus respectivos compartimentos.

As interfaces são elementos que possuem operações, as quais fornecem ou requerem dados do ambiente externo (OMG, 2015b). Neste nível de relação, tais interfaces podem ser identificadas como sendo possíveis pontos de variação (3.5.1), sendo anotadas como «`variationPoint`», e suas respectivas operações, envolvidas neste relacionamento, podem ser caracterizadas como «`optional`», «`alternative_OR`», «`alternative_XOR`» ou «`mandatory`» de acordo com *SMarty* (Seção 2.5).

Porta é um elemento que fornece a interação do ambiente interno com o ambiente externo, e dispõe relacionamentos com interfaces fornecidas e requeridas. Em alguns casos, as portas podem ser identificadas como possíveis pontos de variação (Seção 3.5.2), em relação as suas interfaces, sendo que neste nível, tais interfaces são consideradas variantes, e neste relacionamento podendo ser caracterizadas como variantes «`optional`», «`alternative_OR`», «`alternative_XOR`» ou «`mandatory`».

O componente é um elemento que pode possuir relacionamento direto com suas portas e interfaces, dessa maneira, possui características como ponto de variação (Seção 3.5.3 e Seção 3.5.4), podendo ser anotado como «`variationPoint`», e as portas e interfaces relacionadas como suas variantes, neste relacionamento podendo ser caracterizadas como «`optional`», «`alternative_OR`», «`alternative_XOR`» ou «`mandatory`».

A rastreabilidade para identificação dos níveis de variabilidade pode ser identificada por meio do meta-atributo `realizes` contido na notação de comentários UML, com o estereótipo «`variability`». Para a identificação de rastreabilidade em níveis de maior abstração, o meta-atributo `realizes` contém um sinal de adição (+) adicionado nesta evolução de *SMarty*, e semelhantemente, para os níveis de menor abstração, o item `realizes` contém um sinal de subtração (-). Assim, a rastreabilidade das resoluções de variabilidade se torna bidirecional.

As seções 3.3 e 3.4 apresentam, respectivamente, o perfil UML de *SMarty* já com o suporte a componente, portas, interfaces e operações, além das diretrizes definidas como base na evolução de *SMarty*.

3.3 *SMartyProfile 5.2*

A Figura 3.1 apresenta o perfil *SMartyProfile*, destacado com as metaclasses que cobrem a identificação de variabilidades em nível de componentes, portas, interfaces e operações.

Para compor a evolução do *SMarty* 5.2, as metaclasses `Port` e `Operation` foram importadas do metamodelo da UML para o *SMartyProfile*. A metaclasse `Port` é estendida pelo estereótipo «`variationPoint`», e juntamente com a metaclasse `Operation`, são estendidas pelos estereótipos «`mandatory`», «`optional`», «`alternative_XOR`» e «`alternative_OR`». Na Figura 3.1 a metaclasse `Component` é estendida por «`variable`», porém, pelo fato de tal metaclasse ser uma subclasse de `Class`, as extensões aplicadas a `Class` são também aplicadas às suas subclasses.

Com base nos novos elementos suportados pela nova versão de *SMarty* (5.2), novas diretrizes para o *SMartyProcess* foram definidas visando apoiar o usuário na representação de variabilidades em nível de componentes UML.

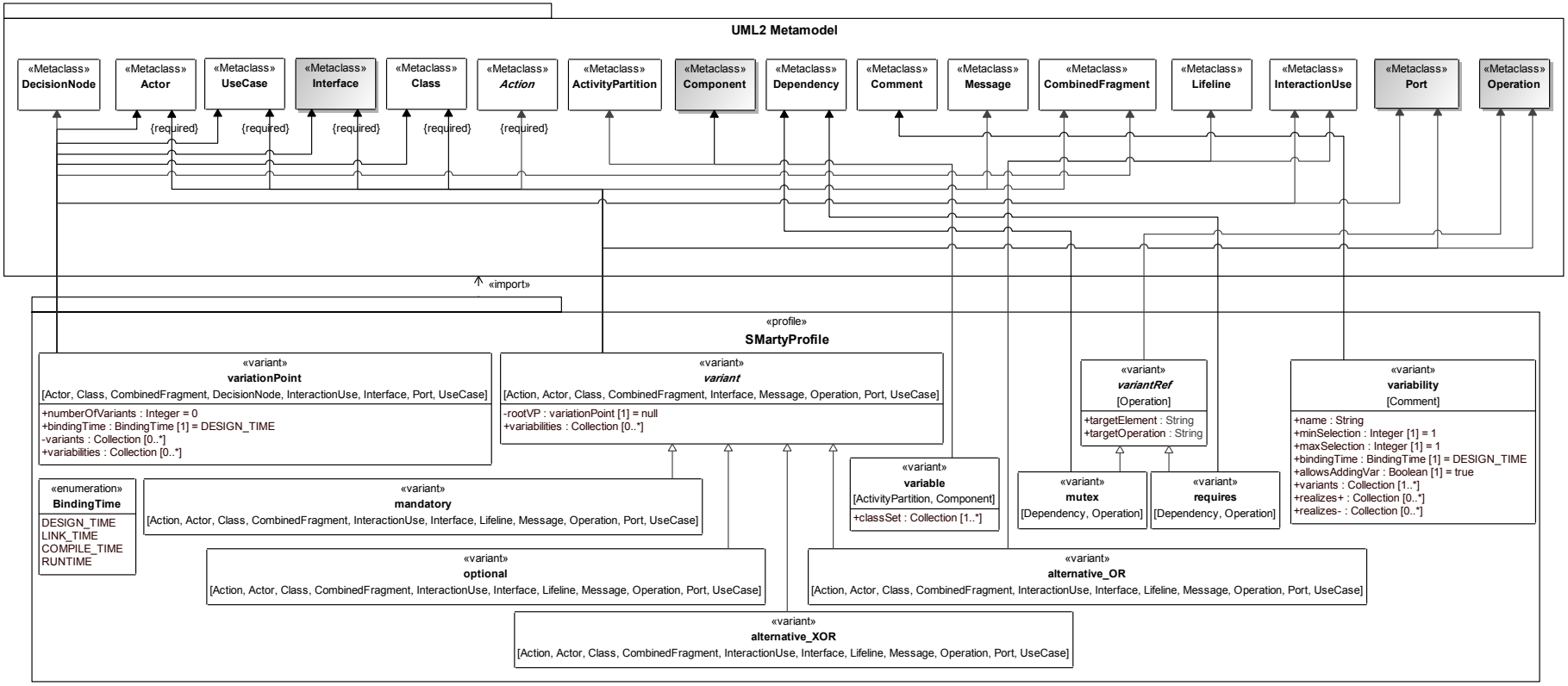

Figura 3.1: O Perfil *SMartyProfile* 5.2 com suporte a Componentes, Portas, Interfaces e Operações.

3.4 *SMartyProcess 5.2*

A partir da evolução dos novos elementos do metamodelo padrão da UML, seis diretrizes foram especificadas para o *SMartyProcess* visando apoiar a identificação e representação de variabilidades com base nos estereótipos do *SMartyProfile* 5.2. Tais diretrizes estão listadas abaixo:

CP.1 Componentes em que são identificados conjuntos de interfaces variantes do tipo inclusiva e/ou exclusiva, sugere-se o uso de portas identificadas com o estereótipo «`variationPoint`», agrupando as interfaces de acordo com os seus interesses.

CP.2 Para interfaces variantes opcionais, sugere-se a sua relação com o próprio componente, ou, em portas que são pontos de variação e que o seu valor `minSelection` seja maior ou igual a 1, evitando, assim, que existam portas sem interfaces selecionadas.

CP.3 Em portas e operações que possuem algum tipo de representação de variabilidade, sugere-se que o estereótipo referente a tal representação torne-se visível no compartimento do classificador, aumentando a compreensão da modelagem.

CP.4 Em interfaces que possuem operações contendo algum tipo de identificação de variabilidades, tal interface deve ser apresentada no formato de classificador, tornando visível as operações em seu compartimento e aumentando a compreensão da modelagem.

CP.5 Variantes que, ao serem selecionadas para fazer parte de um determinado produto, exigem a presença de outra(s) determinada(s) variante(s), devem ter seus relacionamentos de dependência identificados com o estereótipo «`requires`».

CP.6 Variantes que, ao serem selecionadas para fazer parte de um produto específico, exigem a ausência de outra(s) determinada(s) variante(s), devem ter seus relacionamentos de dependência identificados com o estereótipo «`mutex`».

A Figura 3.2 apresenta uma visão geral de *SMarty* e dos modelos UML suportados, bem como suas diretrizes.

A Seção 3.6 apresenta exemplos da identificação e representação de variabilidades de *SMarty* 5.2, apontando para as diretrizes referente à cada aplicação realizada.

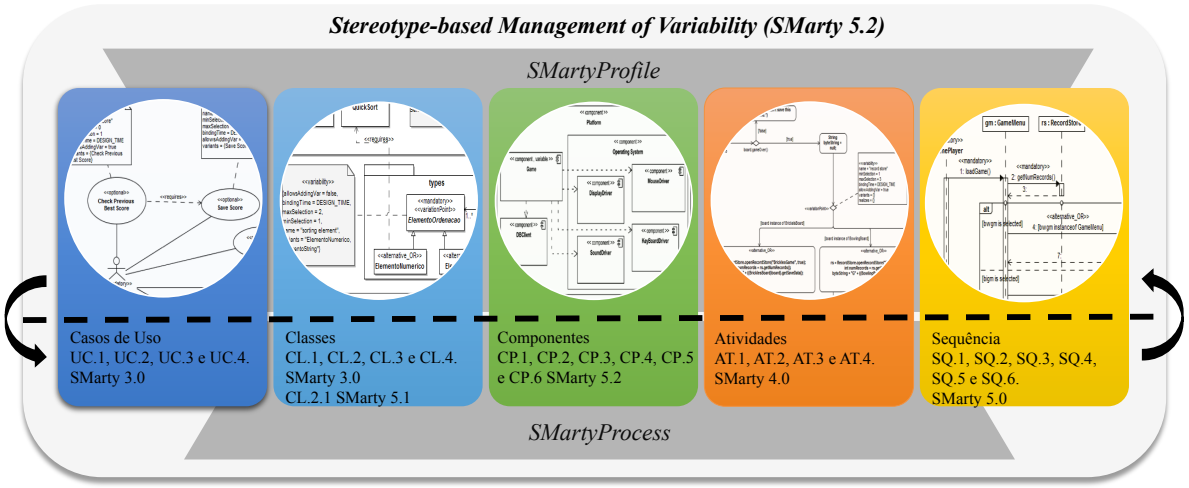

Figura 3.2: Visão Geral de *SMarty* 5.2

3.5 Caracterização do Suporte a Componentes

3.5.1 Relacionamento entre Interfaces e Operações

Nos casos em que interfaces são pontos de variação, as suas operações serão as suas variantes. Na Figura 3.3 há dois exemplos em que (a) é um modelo de representação de interface como um classificador (componente), e (b) é um modelo de estereótipo de interface. No modelo (a) é possível visualizar as operações contidas e seus estereótipos, por ser uma representação de classe e disponibilizar tais informações no compartimento de operações. Já no modelo (b) não é possível visualizar suas operações diretamente, mas é possível identificar suas variantes por meio da notação de variabilidade (comentário UML).

No modelo (a), a interface `IPlayMedia` possui três operações, que são respectivas às mídias de vídeo, música e foto. Assim, tal interface foi identificada como sendo um ponto de variação e as operações como suas respectivas variantes anotadas com o estereótipo «`alternative_OR`». Pela variabilidade (comentário UML) percebe-se que no mínimo uma e no máximo três das variantes devem ser selecionadas. Já no modelo (b) deste mesmo exemplo, as operações de `IPlayMedia` ficam ocultas, percebidas somente pelo meta-atributo `variants` da variabilidade associada à interface.

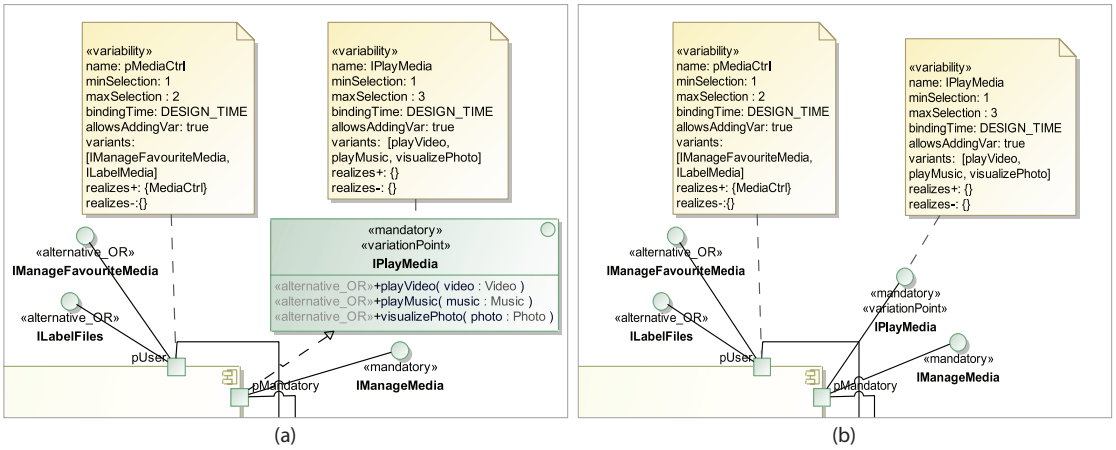

Figura 3.3: Representação de Variabilidade em Interfaces com Operações de acordo com *SMarty* 5.2.

3.5.2 Relacionamento entre Portas e Interfaces

Existem casos em que se sugere o uso de portas, como por exemplo, na Figura 3.4. Neste exemplo, o componente `GameCtrl` possui uma porta e também interfaces diretamente relacionadas. Este diagrama de componente apresenta as interfaces dos jogos disponíveis, ligados a uma determinada porta, e interfaces das operações obrigatórias do jogo, como por exemplo para instalação do jogo (`IInstallGame`), salvar o jogo (`ISaveGame`), desinstalar o jogo (`IUninstallGame`) e sair do jogo (`IExitGame`). Neste exemplo, foi sugerida a utilização de porta como ponto de variação, justamente para separar as interfaces que são de uma determinada característica, e que ocorrem algum tipo de variabilidade. Para a porta `pPlayGame`, foi atribuída a anotação do estereótipo «`variationPoint`» representando um ponto de variação, e tal estereótipo pode ser visualizado no compartimento do componente `GameCtrl`, antes do nome da respectiva porta. Um comentário UML representando a variabilidade associada está anotado com o respectivo estereótipo «`variability`», e indica quais interfaces são suas variantes e a quantidade mínima e máxima de seleção. As interfaces relacionadas à esta porta (`IPlayBowling`, `IPlayBrickles` e `IPlayPong`), são as respectivas variantes e estão estereotipadas como «`alternative_OR`».

Neste exemplo, a porta `pPlayGame` está relacionada às interfaces que operam a inicialização dos jogos, em que na versão 2 da LPS AGM proposta por Xavier (2011), o componente `GameCtrl` foi dividido em dois, justamente para evitar a sobrecarga sobre um componente específico, e tais interfaces foram atribuídas ao componente `PlayGameCtrl`, que será apresentado posteriormente na Figura 3.7.

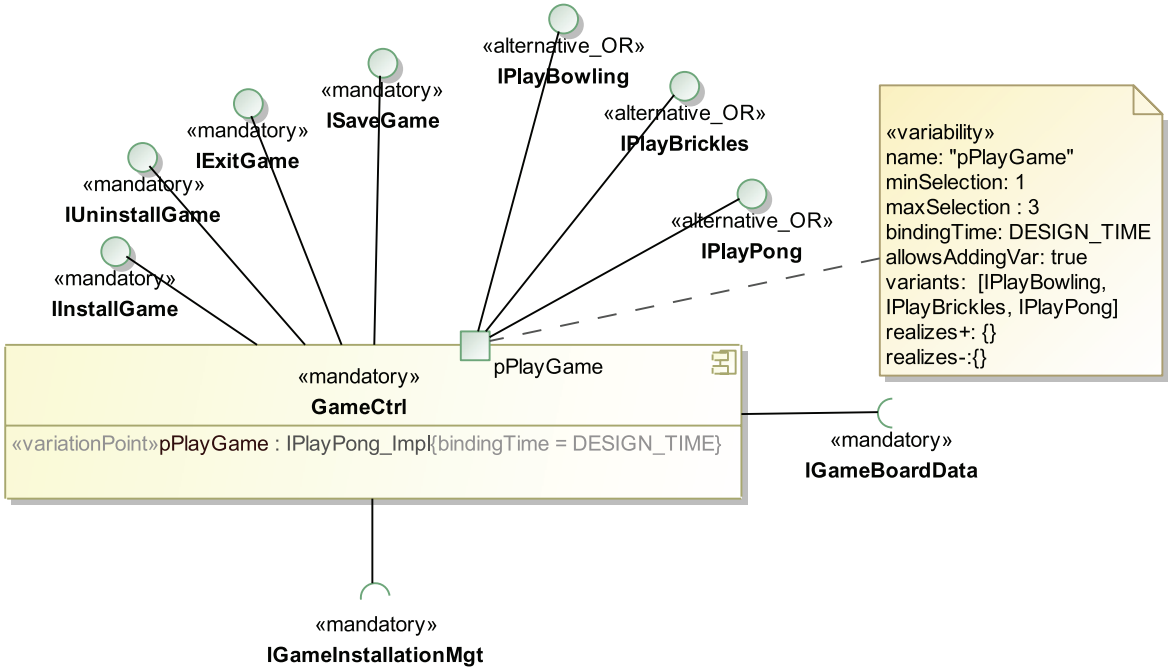

Figura 3.4: Relacionamento de porta com interfaces.

Nas seções a seguir, são apresentados exemplos dos relacionamentos definidos em nível de componentes, partindo dos relacionamentos em interfaces e operações rastreando os elementos até identificar os relacionamentos em níveis maiores de abstração, como por exemplo o nível de portas e componentes. Os exemplos baseiam-se na LPS MM e foram adaptadas de Correia (2010), Contieri Junior (2010), Xavier (2011) e Contieri Junior *et al.* (2011).

3.5.3 Relacionamento entre Componentes e Portas

A Figura 3.5 apresenta um exemplo de um componente de serviço `MediaMgr`, da LPS `MM` (Contieri Junior, 2010). Esse componente tem a função de fornecer serviços que a aplicação pode requerer, neste caso uma mídia, podendo ser um vídeo, uma música ou uma foto. Dessa forma, esse componente é um ponto de variação e está anotado como o estereótipo «`variationPoint`», e em seu compartimento suas portas estão listadas. Esse componente possui 4 portas, sendo a `pIMedia`, uma porta que fornece o serviço respectivo deste componente. As demais portas (`pIVideo`, `pIMusic` e `pIPhoto`) que fornecem os serviços específicos são, respectivamente, as variantes, e tais portas estão anotadas com o estereótipo «`alternative_OR`», que é visível em seu compartimento, indicando que pelo menos uma dessas variantes deve ser selecionada.

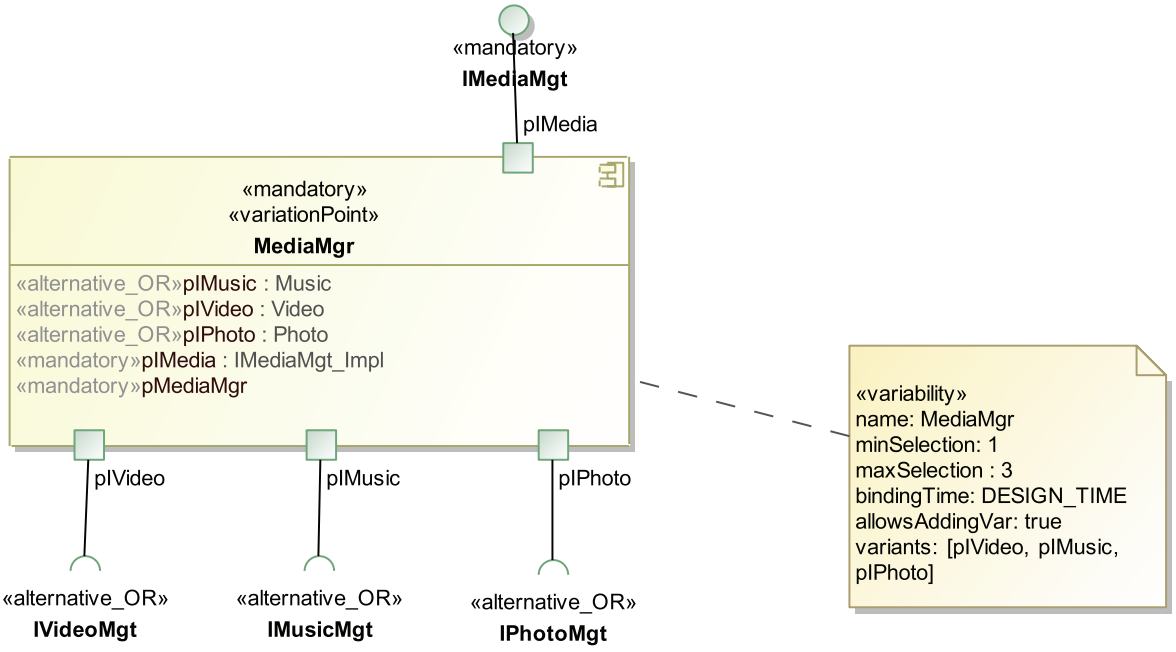

Figura 3.5: Representação de Variabilidade em Portas e Interfaces de acordo com *SMarty* 5.2.

O relacionamento de componentes com portas é opcional, porém, além de permitir uma melhor separação de interesses das interfaces, a utilização das portas pode trazer benefícios como reutilização de portas e interfaces. Neste exemplo, foram utilizadas portas sem interfaces relacionadas somente como uma forma de representar o relacionamento de componentes e portas. Na Figura 3.6 há um exemplo da utilização de portas com o intuito de separar as interfaces e melhorar seus relacionamentos, facilitando a identificação e representação das variabilidades.

Neste exemplo, o componente `MediaCtrl` representa variabilidades diretamente com as portas, sendo o próprio componente um ponto de variação, anotado com o estereótipo «`variationPoint`» e as portas `pMediaFiles` e `pMediaCtrl` como sendo suas variantes, anotadas com o estereótipo «`alternative_OR`». Além deste estereótipo, ambas as portas foram identificadas como pontos de variação e anotadas com o estereótipo «`variationPoint`» por possuírem interfaces que podem ser suas respectivas variantes. Em ambas as portas, a notação de variabilidade aponta as suas variantes e também por meio do meta-atributo `realizes` identifica as variabilidades que podem ser rastreadas em diferentes níveis de abstração. A porta `pMediaCtrl` possui três interfaces anotadas com o estereótipo «`alternative_OR`», e a variabilidade associada à esta porta indica que no mínimo uma e no máximo três das variantes podem ser selecionadas. A porta

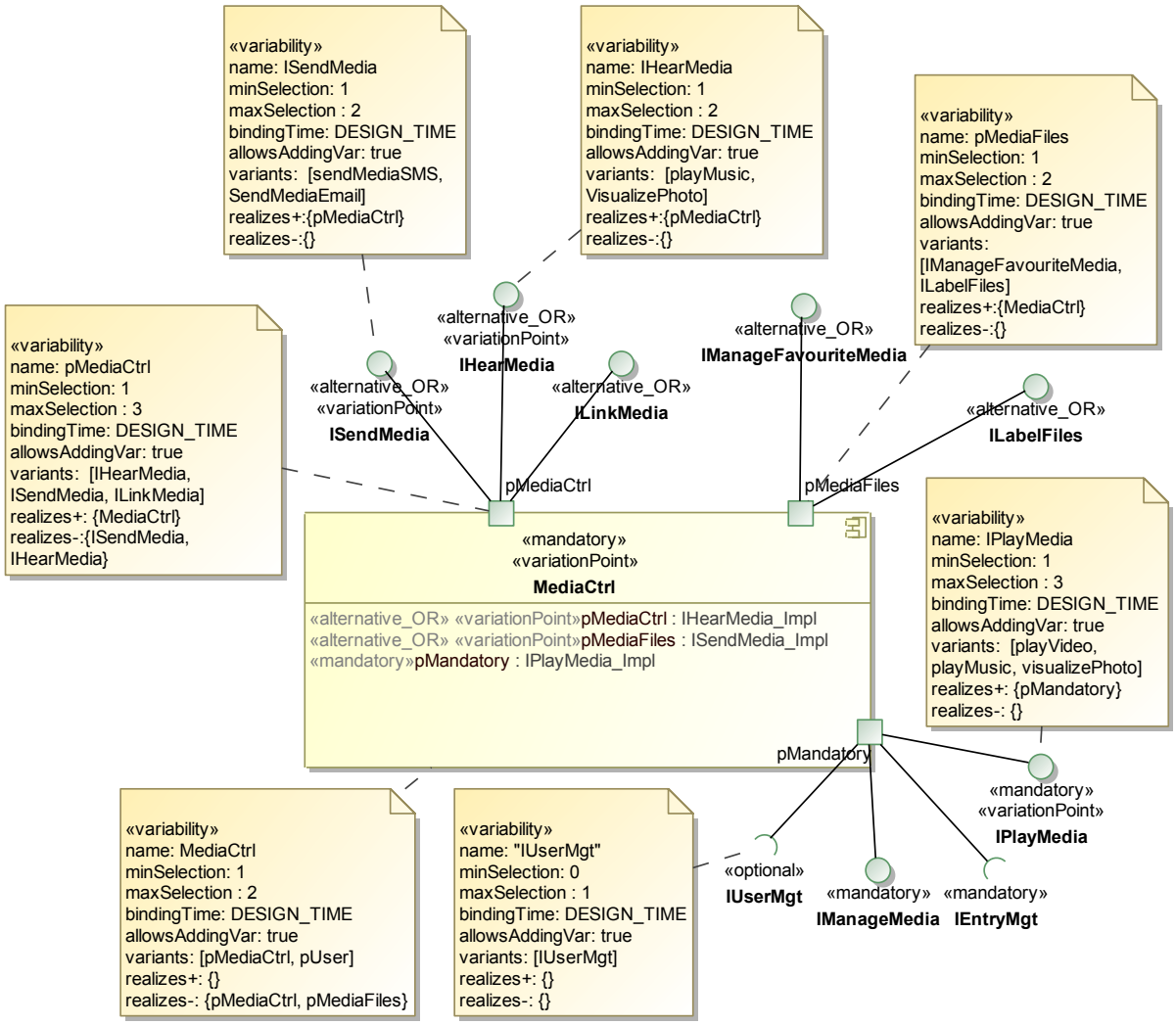

Figura 3.6: Representação de variabilidade de Componentes com Portas, e Portas com Interfaces de acordo com *SMarty* 5.2.

`pMediaFiles` foi identificada como sendo um ponto de variação, e possui três interfaces sendo que duas delas são alternativas inclusivas e outra opcional. As interfaces `IManageFavouriteMedia` e `ISendMedia` foram identificadas com variantes alternativas inclusivas e representadas com o estereótipo «`alternative_OR`», já a interface `IUserMgt` foi identificada como opcional e representada com o estereótipo «`optional`». Nesta interface, na variabilidade associada, o meta-atributo `realizes` identifica as variabilidades em um nível mais alto de abstração. Ainda neste componente, a porta `pMandatory` foi incluída contendo as interfaces que são obrigatórias, com a intenção de separar os interesses.

3.5.4 Relacionamento entre Componentes e Interfaces

Componentes podem se relacionar diretamente com interfaces. Dessa forma, a Figura 3.7 apresenta um exemplo onde há um componente com relacionamento direto com interfaces, sem utilização de portas. Este exemplo é um diagrama de componente de controle da versão 2 da LPS *Arcade Game Maker* (AGM) (Xavier, 2011), onde o componente `PlayGameCtrl` fornece os jogos disponíveis pela LPS. Por esse motivo, esse componente é determinado um ponto de variação, indicando que pelo menos um e no máximo três dos jogos devem ser selecionados para uma LPS. Assim, ele recebe o estereótipo «`variationPoint`». Tais jogos são disponibilizados como interfaces, consideradas variantes deste componente ponto de variação, sendo elas `IPlayBowling`, `IPlayBrickles` e `IPlayPong`. Tais interfaces, estão anotadas com o estereótipo «`alternative_OR`».

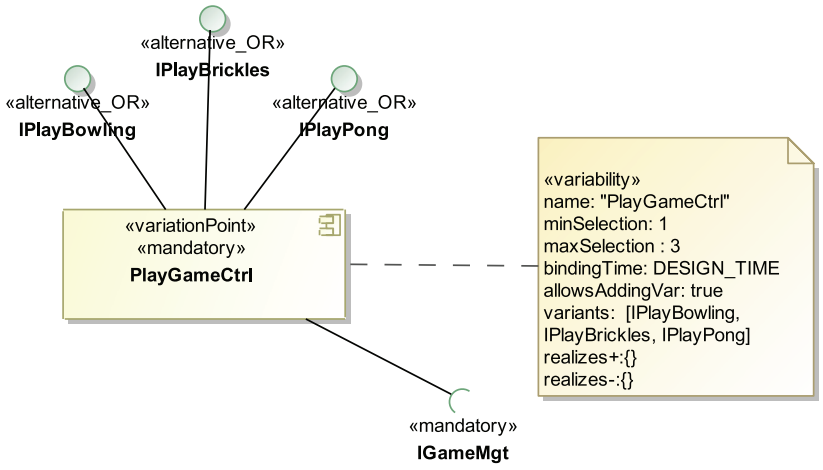

Figura 3.7: Representação de Variabilidade entre Componentes e Interfaces de acordo com *SMarty* 5.2.

3.6 Exemplo de Aplicação de *SMarty 5.2*

A Figura 3.8 apresenta a arquitetura da versão 8 da LPS MM de Contieri Junior (2010) e modelada de acordo com as diretrizes propostas para o *SMartyProcess* 5.2 bem como a aplicação de estereótipos do *SMartyProfile* 5.2.

Neste exemplo o componente `MediaCtrl` é o principal componente da LPS, tem o objetivo de executar as mídias, gerenciar as mídias favoritas, renomear arquivos, ouvir mídia, enviar mídia e gerenciar mídias e álbuns. `MediaCtrl` foi identificado como um ponto de variação e anotado com o estereótipo «`variationPoint`», e as portas `pMediaCtrl` e

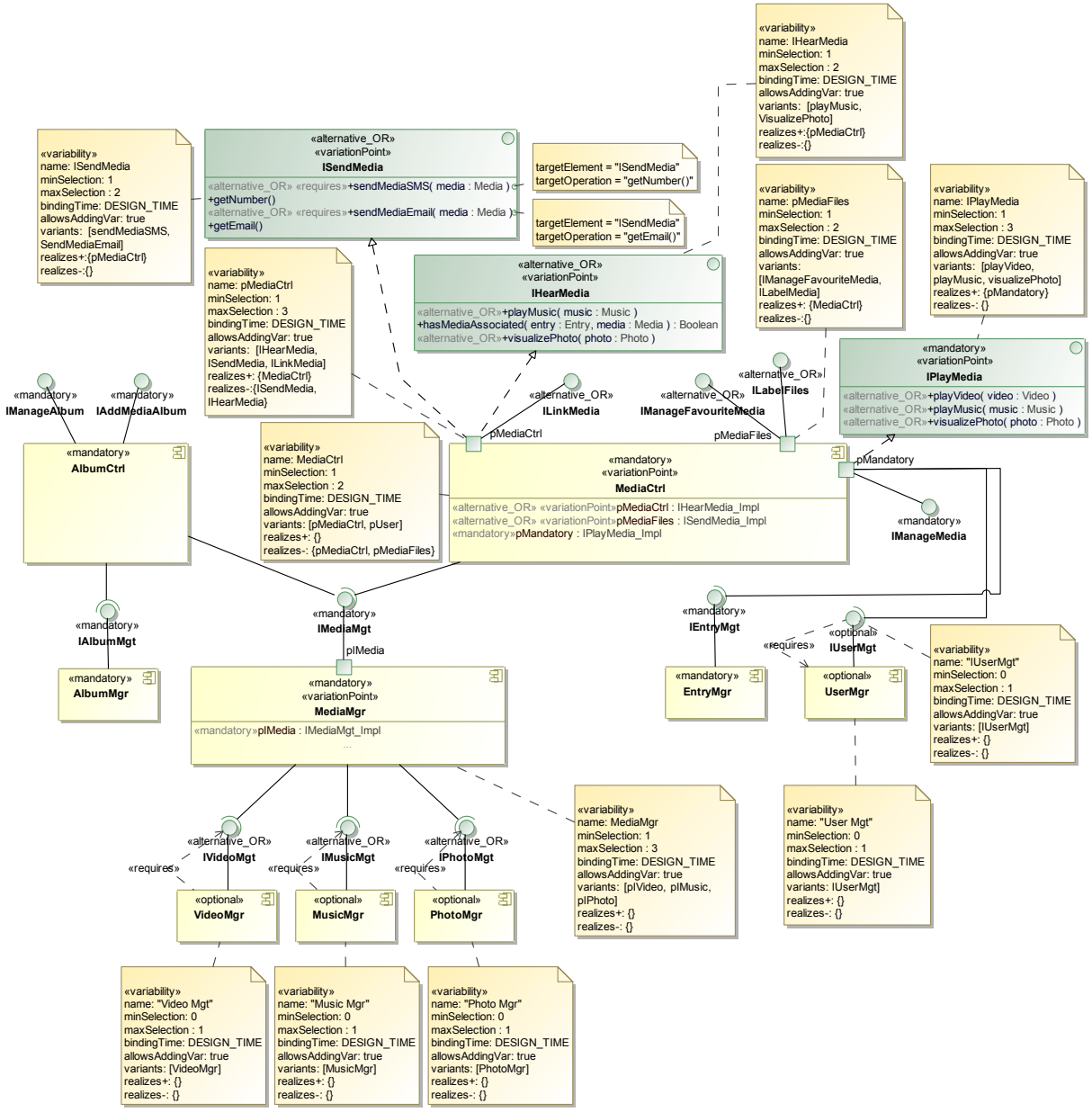

Figura 3.8: Aplicação de *SMarty* 5.2 na Versão 8 da LPS MM.

`pMediaFiles` foram identificadas com suas variantes alternativas inclusivas, anotadas com o estereótipo «`alternative_OR`». Para justificar a modelagem aplicada a estes elementos, foi aplicada a diretriz `CP.2`, onde sugere-se o uso de portas, quando identifica-se mais de um tipo de ponto de variação no mesmo componente, melhorando assim a representação de variabilidades. Também foi aplicada a diretriz `CP.3`, que sugere o uso de portas quando as interfaces identificadas como possíveis resoluções de variabilidades não são exclusivamente opcionais, de modo que, se tais interfaces fossem opcionais, a não seleção dessas interfaces

poderia considerar a possibilidade de existir uma porta sem interfaces relacionadas. No caso do exemplo, pelo fato da porta `pMandatory` conter interfaces obrigatórias, a interface opcional `IUserMgt` está relacionada à esta porta, assim, a não seleção dessa interface não compromete o uso dessa porta. Ainda, esse componente está anotado com o estereótipo «`mandatory`», que significa que ele é obrigatório em qualquer produto específico gerado.

As portas `pMediaCtrl` e `pMediaFiles`, além de serem variantes alternativas inclusivas, também foram identificadas como pontos de variação, anotadas com o estereótipo «`variationPoint`». A anotação de estereótipos referentes às portas pode ser percebida por meio do compartimento de portas no componente `MediaCtrl` de acordo com a diretriz `CP.4`. A porta `pMediaCtrl` possui três interfaces identificadas como variantes alternativas inclusivas, sendo elas `ILinkMedia`, `ISendMedia` e `IHearMedia`, anotadas com o estereótipo «`alternative_OR`». Já a porta `pMediaFiles` possui duas interfaces identificadas como variantes alternativas inclusivas, sendo elas `ILabelFiles` e `IManageFavouriteMedia`. Ambas as portas possuem ainda variabilidades associadas informando as propriedades referentes aos pontos de variação.

De acordo com a diretriz `CP.5`, as interfaces que possuem operações com algum tipo de variabilidade, devem ser modeladas no formato de classificador, sendo assim, as interfaces `ISendMedia`, `IHearMedia` e `IPlayMedia` apresentam em seus compartimentos as operações existentes. Ainda, de acordo com a diretriz `CP.4` tais operações devem ser anotadas com estereótipos que definem a sua representação de variabilidade. As demais interfaces que não possuem operações com algum tipo de identificação de variabilidades, opcionalmente podem permanecer com o estereótipo de interface fornecida ou requerida. Dessa forma, duas interfaces da porta `pMediaCtrl` foram identificadas como ponto de variação, e anotadas com o estereótipo «`variationPoint`», sendo elas `ISendMedia` e `IHearMedia`. A interface `ISendMedia` possui duas operações identificadas como variantes alternativas inclusivas, sendo elas `sendMediaSMS` e `sendMediaEmail`, anotadas com o estereótipo «`alternative_OR`». Já a interface `IHearMedia` possui duas operações identificadas como variantes alternativas inclusivas, sendo elas `playMusic` e `visualizePhoto`, anotadas com o estereótipo «`alternative_OR`».

O componente `MediaCtrl` ainda possui uma porta obrigatória `pMandatory` anotada com o estereótipo «`mandatory`», e possui duas interfaces `IManageMedia` e `IPlayMedia`. A interface `IManageMedia` foi identificada como obrigatória e anotada com o estereótipo «`mandatory`». Já a interface `IPlayMedia`, contém três operações que são referentes às mídias, dessa forma, tais operações podem ou não existir, de acordo com as mídias selecionadas para fazer parte de um produto específico. Assim, de acordo com a diretriz `CP.5`, tal interface foi identificada como um ponto de variação, anotada com o estereótipo

≪variationPoint≫ e suas operações foram identificadas como variantes alternativas inclusivas, anotadas com o estereótipo ≪alternative_OR≫.

O componente MediaMgr, é responsável por gerenciar as mídias disponíveis na LPS, sendo elas vídeo, música e fotos. Ele possui três interfaces, respectivas às mídias disponíveis na LPS. De acordo com a diretriz CP.2, esse componente não possui mais de um tipo de conjunto de interfaces, para que seja sugerida a utilização de portas. Dessa maneira, optou-se pela não utilização de portas. As interfaces deste componente foram relacionadas diretamente com o componente MediaMgr, e esse componente foi identificado com um ponto de variação, sendo anotado com o estereótipo ≪variationPoint≫. As interfaces IVideoMgt, IMusicMgt e IPhotoMgt, foram identificadas como sendo variantes alternativas inclusivas, e foram anotadas com o estereótipo ≪alternative_OR≫. O componente MediaMgr ainda possui uma interface IMediaMgt, que é obrigatória, e foi anotada com o estereótipo ≪mandatory≫. O componente MediaMgr possui uma variabilidade associada contendo as propriedades que se referem ao ponto de variação.

Ainda neste exemplo, o componente UserMgr é opcional, dessa forma a interface IUserMgt só pode existir se este componente estiver presente. O estereótipo ≪requires≫ indica essa restrição, de acordo com a diretriz CP.6.

Por fim, os componentes EntryMgr, AlbumCtrl e AlbumMgr, juntamente com suas respectivas interfaces, foram identificados como elementos obrigatórios em qualquer possível produto gerado dessa LPS. Dessa forma, tais elementos foram anotados com o estereótipo ≪mandatory≫.

3.7 Considerações Finais

A versão de *SMarty* 5.1 apresenta diversas opções para suportar diferentes modelos UML, porém identificou-se que os diagramas de componentes sofriam uma baixa representação, devido ao fato de possuir somente um estereótipo, que não conseguia expressar de forma significativa a representação de variabilidades.

Diante dessa preocupação, a evolução do diagrama de componentes para o *SMarty*, através da versão 2.5 da UML, pode atribuir novas diretrizes para identificação e representação de variabilidades em diagramas de componentes.

Foi conduzido um mapeamento sistemático (Apêndice A) visando identificar estudos que contemplassem a utilização de variabilidades em níveis de componentes e arquiteturas baseadas em componentes, dessa maneira, os principais elementos que foram identificados representando algum tipo de variabilidade, apoiaram a evolução do *SMarty*.

O próximo Capítulo apresenta a avaliação empírica realizada para avaliar a efetividade da evolução de *SMarty* 5.2.

4

Avaliação Experimental de *SMarty 5.2*

"Se queremos progredir, não devemos repetir a história, mas fazer uma história nova."

Mahatma Gandhi(1869 - 1948), Estadista

4.1 Considerações Iniciais

Avaliações experimentais são amplamente difundidas em várias áreas da ciência, e consequentemente, estendidas para a Ciência da Computação. A comunidade científica de Engenharia de Software constantemente propõe novas tecnologias, processos, procedimentos e ferramentas a fim de desenvolver projetos mais rápidos, com qualidade e retorno de investimento (Juristo e Moreno, 2010).

Na literatura existem várias abordagens para GV (Chen *et al.*, 2009) (Galster *et al.*, 2013), porém, tais estudos não se mostram suficientes para demonstrar a efetividade dessas, o que torna difícil a aceitação no meio industrial e acadêmico.

Assim, esta seção apresenta a avaliação experimental de *SMarty* 5.2, que tem por objetivo analisar a significância de sua efetividade em comparação com a abordagem de Razavian e Khosravi (2008), conforme apresentado na Seção 2.3.

4.2 Metodologia e Planejamento Experimental

Para avaliar a evolução da abordagem *SMarty* houve a necessidade de buscar por abordagens já existentes na literatura que representam variabilidades em componentes. Assim, o MS realizado (Apêndice A) recuperou alguns estudos relevantes, que apresentam maneiras de representar variabilidades em arquiteturas de software e arquiteturas baseadas em componentes. Um subconjunto desses estudos foi analisado e selecionado, conforme a Tabela 2.1 (Seção 2.3). Desses, o método deRazavian e Khosravi (2008) foi considerado o mais relevante, por possuir maior representatividade em componentes UML. Dessa forma, tal método foi selecionado para compor este estudo experimental. A abordagem de *Razavian e Khosravi*, conforme apresentada na Seção 2.3, permite explorar variabilidades em níveis de componentes, interfaces e conectores. Assim, foi planejado um experimento com a intenção de analisar a efetividade de *SMarty* 5.2 com relação à abordagem de *Razavian e Khosravi*.

Nessa pesquisa, a definição da efetividade se baseia em alguns trabalhos (Basili e Selby, 1987; Marcolino *et al.*, 2014a; Martinez-Ruiz *et al.*, 2011; Çöteli, 2013), nos quais é caracterizada em o quão bem uma abordagem permite que as variabilidades possam ser identificadas e representadas em modelos UML de LPSs. Para calcular a efetividade foram dispostas duas LPSs, nas quais os participantes deveriam identificar as variabilidades e elementos de acordo com a proposta de cada abordagem.

As etapas seguidas durante este estudo experimental estão representadas em um diagrama de atividades na Figura 4.1.

Os testes para verificar a diferença entre a efetividade de cada abordagem foram: o teste T (Wohlin *et al.*, 2012) (paramétrico) e o teste *Mann-Whitney-Wilcoxon* (Hole, 2006) (não paramétrico). Tais testes evidenciam a abordagem com maior efetividade. Após identificada tal abordagem, foi realizada uma correlação entre a efetividade obtida da abordagem com o nível de conhecimento em LPS e variabilidade informado previamente por cada participante em um questionário de caracterização (Seção 4.4.2).

O cálculo da correlação de *Spearman*, permite classificar o nível de correlação obtido entre o conhecimento que o participante tem sobre LPS e variabilidade com o resultado da modelagem da LPS gerada pelo usuário utilizando uma das abordagens. Dessa forma, uma análise das evidências é realizada para verificar se existe uma possível influência entre o nível de conhecimento prévio do usuário em LPS e variabilidade, e o resultado da modelagem obtido por este.

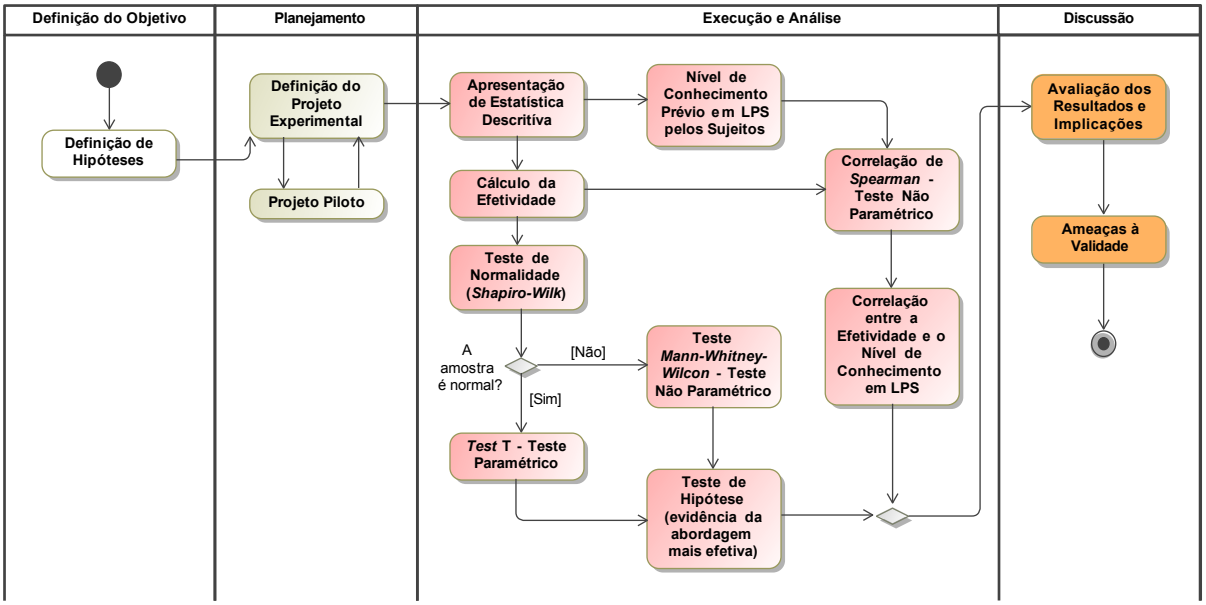

Figura 4.1: Etapas da Avaliação Experimental de *SMarty* 5.2.

As seções a seguir apresentam o estudo experimental realizado de acordo com Juristo e Moreno (2010) e Wohlin *et al.* (2012), além de seguir o template proposto por Jedlitschka *et al.* (2007).

4.3 Objetivos de Pesquisa

O objetivo deste estudo para diagrama de componentes foi **comparar** o método de *Razavian e Khosravi* e a abordagem *SMarty* 5.2, **com o propósito** de caracterizar a abordagem mais efetiva, **em relação à** capacidade de identificação e representação de variabilidades em diagramas de componentes UML de LPS, **do ponto de vista de** arquitetos de LPS, **no contexto de** estudantes de mestrado e doutorado da área de engenharia de software do Departamento de Informática (DIN) da Universidade Estadual de Maringá (UEM), do Instituto de Ciências e Matemáticas e de Computação de São Paulo (ICMC-USP) e da Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), para as LPSs *Virtual University* (VU) e *Mobile Media* (MM). A seleção dos participantes de diferentes localidades foi definida por meio da sugestão de evitar viés quanto ao conhecimento da abordagem *SMarty* pelos mesmos.

Neste estudo o método *Razavian e Khosravi* será tratado como abordagem, por convenção, e ambas as abordagens foram respectivamentes nomeadas como abordagem X

(Razavian e Khosravi) e abordagem Y (*SMarty* 5.2), para eliminar os possíveis viés entre os participantes.

4.4 Planejamento do Estudo Experimental

Nessa seção serão apresentados os itens que compõem o planejamento do estudo experimental.

4.4.1 Questões de Pesquisa

Neste estudo experimental a abordagem *Goal Question Metric* (GQM) (van Solingen *et al.*, 2002) foi seguida, e duas questões de pesquisa foram levantadas:

Q.P.1 Qual das abordagens, *Razavian e Khosravi* ou *SMarty*, é a mais efetiva para identificar e representar variabilidades em componentes UML?

Q.P.2 O nível de conhecimento dos participantes em LPS e variabilidade influencia a aplicação efetiva das abordagens em componentes UML?

4.4.2 Participantes

Os participantes foram selecionados mediante a seleção do grau de formação: mestres, mestrandos, doutores e doutorandos. O questionário de caracterização dos participantes foi preenchido previamente online, para promover a separação dos participantes em grupos distintos, sem que nenhum viés pudesse ser direcionado a alguma abordagem específica. Assim, os participantes com conhecimento prévio sobre uma das abordagens foram direcionados à outra abordagem. Os participantes que não tivessem conhecimento prévio sobre nenhuma das abordagens, seriam direcionados de forma aleatória. Já os participantes com conhecimento prévio sobre ambas as abordagens, seriam descartados.

Um total de 14 participantes, sendo eles mestrandos e doutorandos na área de engenharia de software, foram selecionados para este estudo. A separação dos grupos ocorreu com base nas características à respeito do conhecimento prévio dos participantes. Uma vez que os grupos foram equilibrados, os objetos de tratamento foram randomizados e atribuídos aos participantes, como sugerido por Wilkinson (1999) e discutidos por (Heinsman e Shadish, 1996).

Os participantes não obtiveram nenhum ônus de participação. A principal motivação para a participação no estudo foi aprender novos conceitos em relação a abordagens de

GV baseadas em UML. Além disso, a participação aconteceu com consentimento dos participantes, onde estes preencheram um Termo de Consentimento Livre e Esclarecido (TCLE). Tal termo também continha várias cláusulas a respeito da confidencialidade dos dados do estudo.

4.4.3 Objetos de Estudo

Cada participante recebeu dois documentos (Seção 4.7) contendo as LPSs MM e VU sem representação de variabilidade. O material referente a cada abordagem (X ou Y) foi modelado de acordo com suas particularidades, porém continham as mesmas quantidade de variabilidades e similaridades.

A modelagem da LPS MM apresentava 5 pontos de variação com 14 variantes inclusivas, com 5 variantes opcionais e 4 obrigatórias. Já a modelagem da LPS VU apresentava 4 pontos de variação, 8 variantes inclusivas, 1 variante opcional e 11 obrigatórias.

4.4.4 Instrumentação

Um questionário de caracterização foi preenchido por cada participante, onde tais participantes informaram o seu nível de educação (mestre, mestrando, doutor e doutorando), a área de atuação profissional (ambiente acadêmico ou industrial) e anos/meses de experiência na área de atuação. Além disso, foi investigado o conhecimento prévio em UML, LPS e variabilidade.

As seguintes perguntas referentes ao conhecimento em LPS estavam disponíveis no questionário:

- Eu nunca ouvi falar a respeito de LPS;
- Já li, de forma superficial, algo a respeito de LP;
- **Minha experiência com LPS é básica.** Eu conheço os seguintes conceitos da abordagem: ciclo de desenvolvimento de LPS e suas atividades (engenharia de domínio e engenharia de aplicação). Porém, não tenho experiência com identificação e representação de variabilidades;
- **Minha experiência com LPS é moderada.** Eu conheço os conceitos da opção anterior, e com relação ao gerenciamento de variabilidades, eu sei o conceito de pontos de variação, variantes e os relacionamentos entre variantes; e

- **Minha experiência com LPS é avançada.** Eu conheço os conceitos da opção anterior, além de alguns processos existentes de desenvolvimento de LP (FODA, PLP, PLUS, PuLSE, entre outros). Com relação ao gerenciamento de variabilidades, eu sei os conceitos da opção anterior, além de: modelos de resolução; abordagens existentes para o gerenciamento de variabilidades, e representação de variabilidades (usando a UML, modelos de características, entre outras);

Dessa forma, para as perguntas referente ao conhecimento prévio do participante em LPS e variabilidades foram atribuídos valores de 1 a 5, baseados na escala ordinal de *Likert*, que considera a ordem dos dados em relação ao nível de concordância com uma afirmação (Likert, 1932). Tais valores obtidos foram aplicados à correlação de *Spearman* (Brandalise, 2005; Marcolino, 2014; Pontes *et al.*, 2010).

Os participantes foram divididos em dois grupos. Um grupo focado na abordagem X (*Razavian e Khosravi*) e outro na abordagem Y (*SMarty*). Cada grupo recebeu, separadamente, um treinamento respectivo à abordagem direcionada para identificar e representar variabilidades, com o objetivo de sanar as possíveis dúvidas. Exercícios com a abordagem direcionada foram realizados por ambos os grupos, utilizando uma LPS alternativa, diferente das LPSs utilizadas na realização do experimento em si, buscando minimizar os possíveis viés em relação ao conhecimento prévio das LPSs e suas resoluções corretas. Todos resultados obtidos dos exercícios foram descartados.

Para o experimento foram distribuídos dois documentos, sendo eles a descrição das LPSs VU e MM, além de mais dois documentos contendo o diagrama de componentes UML de tais LPSs, de acordo com cada abordagem de GV.

4.4.5 Tarefas

Os participantes tiveram que identificar e representar variabilidades em diagrama de componentes UML das LPSs MM e VU de forma anotativa, por meio dos estereótipos disponíveis de acordo com a abordagem X e Y, respectivamente. A identificação e representação foi realizada sem nenhuma ferramenta de modelagem UML, apenas com os materiais de apoio referente à abordagem pré-definida.

4.4.6 Hipóteses, Parâmetros e Variáveis

As seguintes hipóteses foram formuladas para a **Q.P.1**:

- **Hipótese Nula** ($H_{0.QP1}$): não existe diferença significativa entre as abordagens X e Y para representar variabilidade em componentes de forma efetiva.

$H_{0.QP1}$: $\mu(efetividade(X)) = \mu(efetividade(Y))$;

- **Hipótese Alternativa** ($H_{1.QP1}$): a abordagem X é significantemente menos efetiva que a abordagem Y para representar variabilidade em componentes.

  $H_{1.QP1}$: $\mu(efetividade(X)) < \mu(efetividade(Y))$;

- **Hipótese Alternativa** ($H_{2.QP1}$): a abordagem X é significantemente mais efetiva que a abordagem Y para representar variabilidade em componentes.

  $H_{2.QP1}$: $\mu(efetividade(X)) > \mu(efetividade(Y))$.

As seguintes hipóteses foram formuladas para a **Q.P.2**:

- **Hipótese Nula** ($H_{0.QP2}$): a efetividade obtida das abordagens X e Y não é afetada pelo conhecimento prévio dos participantes.

  $H_{0.QP2}$: $Corr(conhecimento(X)\ e\ efetividade(X)) = Corr(conhecimento(Y)\ e\ efetividade(Y)) = 0$;

- **Hipótese Alternativa** ($H_{1.QP2}$): a efetividade obtida da abordagem X é menos afetada pelo conhecimento prévio dos participantes do que a efetividade obtida da abordagem Y.

  $H_{1.QP2}$: $Corr(conhecimento(X)\ e\ efetividade(X)) < Corr(conhecimento(Y)\ e\ efetividade(Y))$;

- **Hipótese Alternativa** ($H_{2.QP2}$): a efetividade obtida da abordagem X é mais afetada pelo conhecimento prévio dos participantes do que a efetividade obtida da abordagem Y.

  $H_{2.QP2}$: $\mu(conhecimento(X)\ e\ efetividade(X)) > \mu(conhecimento(Y)\ e\ efetividade(Y))$.

A Tabela 4.1 descreve as variáveis dependentes e independentes desse estudo experimental. A LPS fornecida pelos autores Razavian e Khosravi foi utilizada a fim de evitar possíveis viés.

4.4.7 Projeto Experimental

Foi utilizado um ensaio fatorial de 2x2, sendo dois fatores (LPSs e abordagens de GV) com dois tratamentos cada (MM e VU para as LPSs e *Razavian e Khosravi* e *SMarty*

Tabela 4.1: Descrição das Variáveis Dependentes e Independentes.

Nome da Variável	**Tipo de Variável** (dependente, independente, moderada)	**Abreviação**	**Classe** (produto, processo, recurso, método)	**Entidade** (Instância da Classe)	**Tipo de Atributo** (interno, externo, …)	**Tipo de Escala** (nominal, ordinal, …)	**Unidade**	**Alcance**	**Regra de Contagem**
Abordagem de GV	Independente	X; Y	método	GV baseado em UML	N.A.	nominal	N.A.	Razavian e Khosravi; SMarty 5.2	N.A
Diagramas para LPSs		LPS		Diagrama UML	N.A.	nominal	N.A.	Componentes	N.A
Efetividade	Dependente	efetividade	processo	Variabilidades Corretas e Incorretas	interno: Efetividade; externo: Qualidade	ordinal	Número de Variabilidades corretas menos o número de Incorretas	qualquer inteiro	Variabilidades Corretas conta +1; Incorretas conta - 1;
Correlação entre o Conhecimento Prévio dos Participantes e a Efetividade		*Corr* (conhecimento e efetividade)	processo	Conhecimento Prévio dos Participantes e a Efetividade da abordagem	externo	nominal	Escala de correlação de *Spearman*	[-1.0, +1.0]	Equação de *Spearman (p)*

5.2 para as abordagens de GV), com 14 participantes no total, dentre eles acadêmicos de mestrado e doutorado.

O projeto piloto foi executado para avaliar a instrumentação da avaliação experimental. Para o projeto piloto, foi selecionada uma docente com 25 anos de experiência em engenharia de software, que, criteriosamente avaliou, discutiu e sugeriu alterações necessárias para a melhoria da instrumentação. É importante deixar claro que o resultado do projeto piloto não foi considerado junto ao resultado obtido pelos participantes.

4.4.8 Procedimentos

O estudo foi realizado em dois dias, sendo que no primeiro dia os participantes receberam um treinamento sobre os conceitos essenciais sobre LPS e variabilidade em um mesmo ambiente. Após este treinamento os grupos foram divididos e cada grupo participou de um treinamento específico sobre os conceitos de variabilidades em diagramas de componentes UML de acordo com cada abordagem. Durante todo o contexto do estudo, os participantes não souberam o nome verdadeiro das abordagens X e Y, para se evitar qualquer possibilidade de viés.

O estudo ocorreu em dois ambientes: (i) acadêmico, com acadêmicos da Universidade Estadual de Maringá (UEM); e (ii) online, com acadêmicos do Instituto de Ciências e Matemáticas e de Computação de São Paulo (ICMC-USP) e da Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS). É importante destacar que os estudos aplicados online foram em momentos diferentes e individualmente a cada participante.

Os procedimentos padrões adodatos para cada participante foram os seguintes:

- um pacote com uma documentação é entregue a eles, sendo que em cada pacote contém:
  - um documento contendo a descrição da LPS VU, de acordo com a abordagem direcionada;
  - um documento contendo a descrição da LPS MM, de acordo com a abordagem direcionada;
  - um documento contendo o diagrama de componentes UML da LPS VU para a identificação e representação de variabilidades, de acordo com os conceitos da abordagem na qual o participante recebeu o treinamento;
  - um documento contendo o diagrama de componentes da LPS MM para a identificação e representação de variabilidades, de acordo com os conceitos da abordagem na qual o participante recebeu o treinamento;
  - o experimentador tira as possíveis dúvidas em relação aos conceitos de LPSs;
  - os participantes iniciam a avaliação empírica, ordenando o início do estudo com o documento da LPS que lhe convém; e
  - ao finalizar as atividades propostas, cada participante guarda suas amostras no pacote distribuído e, para cada pacote, é referenciado uma identificação única.
- a partir da devolução dos pacotes experimentais resolvidos, a avaliação empírica é finalizada para o respectivo participante.

4.4.9 Procedimentos de Análise

Após o término das sessões experimentais, os dados coletados devem ser preparados para calcular a efetividade de cada amostra da abordagem de GV e, em seguida, correlacionar tal efetividade em relação ao nível de conhecimento prévio de variabilidade para cada abordagem. A efetividade é utilizada em vários trabalhos (Abdelnabi *et al.*, 2004; Basili e Selby, 1987; Martinez-Ruiz *et al.*, 2011; Çöteli, 2013) como uma medida da presença ou ausência de algo na realização de uma determinada atividade, o que pode aumentar a confiabilidade (Farooq e Quadri, 2011). Neste trabalho, a efetividade está relacionada com o sucesso de produzir um resultado desejado, que é a identificação correta das variabilidades em LPSs por meio do GV em diagramas de componentes UML. A preparação de dados envolve a tabulação dos dados e o cálculo das estatísticas descritivas.

A efetividade é, então, calculada com base na seguinte equação (Basili e Selby, 1987):

$$\textbf{\textit{efetividade(z)}} = \Big\{ nVarC - nVarI$$

sendo que:

- z é a abordagem de GV;
- $nVarC$ é a quantidade de elementos de variabilidades identificados de forma correta, em relação à abordagem z; e
- $nVarI$ é a quantidade de elementos de variabilidades identificados de forma incorreta, em relação à abordagem z.

Após o resultado das amostras de efetividade, os testes de normalidade são, então, aplicados às tais amostras, com o intuito de verificar o teste de hipótese a ser aplicado. Em seguida, a correlação entre o conhecimento prévio dos participantes sobre LPS e variabilidade em relação a efetividade de cada estudo é obtida.

4.5 Análise Experimental

A análise do estudo experimental é apresentada nessa seção, como os dados coletados, estatísticas descritivas e testes de normalidade e de hipóteses.

4.5.1 Coleção de Dados e Estatística Descritiva

Os resultados extraídos das amostras colhidas referente ao método *Razavian e Khosravi* e a abordagem *SMarty* para as LPSs VU e MM foram submetidos às etapas que podem ser vistas a seguir:

- a Tabela 4.2 apresenta a análise e interpretação dos dados coletados para as abordagens X e Y, por meio do teste de *Shapiro-Wilk* e teste T; e
- análise e interpretação da correlação existente entre a efetividade das abordagens e o nível de conhecimento em LPS fornecido pelos participantes no questionário de caracterização, por meio da técnica de correlação de *Spearman*.

De acordo com os dados coletados e apresentados na Tabela 4.2 é possível evidenciar que a abordagem X obteve a média da efetividade ($\mu = 6,43$) menor que a abordagem Y ($\mu = 9,26$). Ainda, a Figura 4.2 apresenta a mediana obtida das amostras em relação à efetividade das abordagens X e Y. Dessa forma, pode-se perceber que a mediana das duas

Tabela 4.2: Resultados Coletados e Estatística Descritiva das Abordagens X e Y

Abordagem X (Razavian and Khosravi)				**Abordagem Y (SMarty 5.2)**			
Identificador	Elementos de Variabilidade Corretos	Elementos de Variabilidade Incorretos	Efetividade	Identificador	Elementos de Variabilidade Corretos	Elementos de Variabilidade Incorretos	Efetividade
1	15,90	6,10	9,80	1	18,60	3,40	15,20
2	10,20	11,80	-1,60	2	15,70	6,30	9,40
3	17,50	4,50	13,00	3	15,10	6,90	8,20
4	19,60	2,40	17,20	4	14,20	8,70	6,40
5	15,40	6,60	8,80	5	12,10	9,90	2,20
6	13,20	8,80	4,40	6	14,50	7,50	7,00
7	7,20	13,80	-6,60	7	19,20	2,80	16,40
Média	14,14	7,71	6,43	Média	15,63	6,50	9,26
Desvio Padrão	4,29	4,03	8,31	Desvio Padrão	2,50	2,61	5,01
Mediana	15,40	6,60	8,80	Mediana	15,10	6,90	8,20

abordagens está quase na mesma posição, porém, em relação a quantidade de acertos a abordagem X obteve uma quantidade menor de acertos do que a abordagem Y, o que corrobora com o valor identificado das médias na Tabela 4.2.

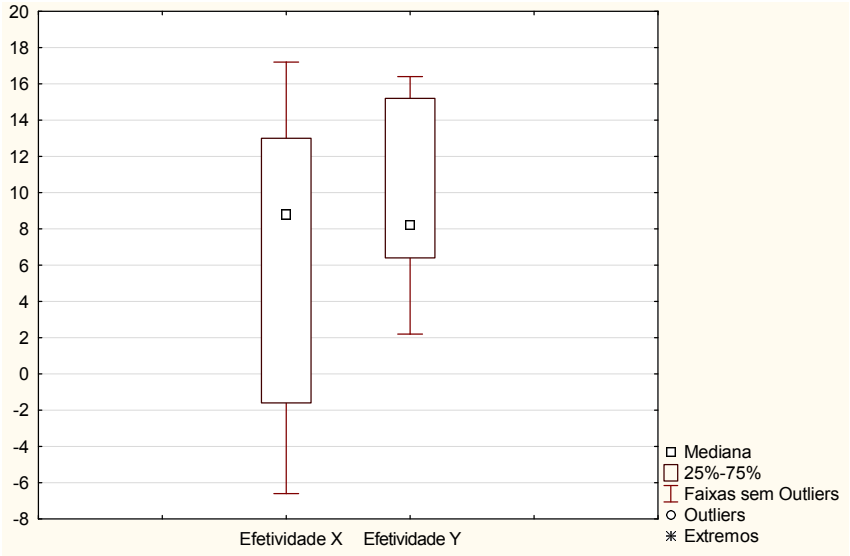

Figura 4.2: Box Plot da Efetividade para as abordagens X e Y

4.5.2 Teste de Normalidade e de Hipótese para Q.P.1

Esta seção apresenta os testes de normalidade e de hipótese deste estudo experimental em relação à Q.P.1.

- **Teste de Normalidade de Dados:** o teste de normalidade *Shapiro-Wilk* foi aplicado e a efetividade foi calculada para os modelos desenvolvidos com as LPSs

VU e MM, de cada participante, conforme apresentado nos histogramas da Figura 4.3, indicando os resultados conforme a seguir:

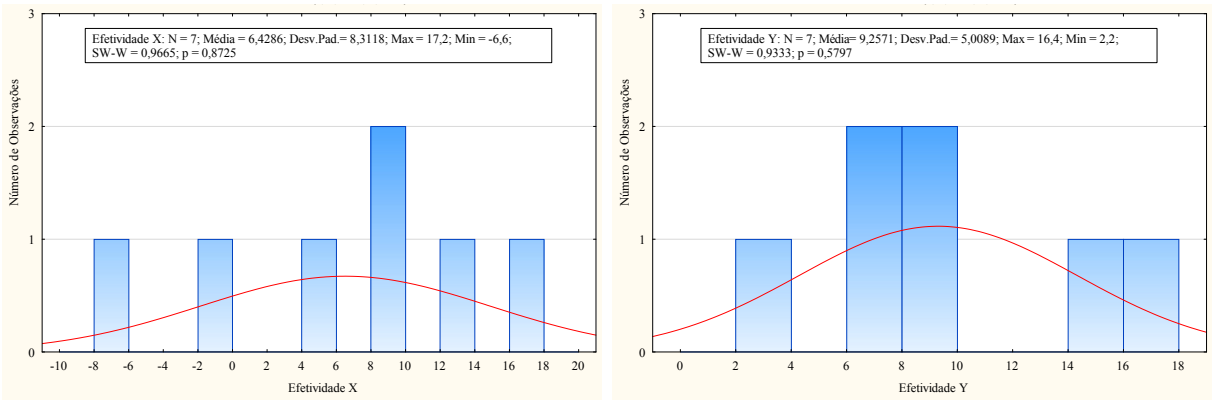

Figura 4.3: Histograma da Amostra da Efetividade para as abordagens X e Y

Abordagem X (N = 7):

- Efetividade para a LPS VU: para a média (μ) **3,8**, desvio padrão (σ) **7,02**, a efetividade para a aplicação da abordagem X na LPS VU resultou em p = **0,08**, ao ser aplicado ao teste de normalidade *Shapiro-Wilk*.

  O teste indicou, para uma amostra *(N)* de tamanho 7 com 95% de nível de significância (α = 0,05), p = **0,0847**, (0,0847 > 0,05) e o valor calculado de W = **0,8326** > W = **0,8030**, que a amostra é considerada normal;

- Efetividade para a LPS MM: para a média (μ) **2,63**, desvio padrão (σ) **3,30**, a efetividade para a aplicação da abordagem X na LPS MM resultou em p = **0,7024**, ao ser aplicado ao teste de normalidade *Shapiro-Wilk*.

  Assim, para (α = 0.05), p = **0,7024** (0,7024 > 0,05) e o valor calculado (W), W = **0,947** > W = **0,8030**, indica que a amostra é considerada normal;

- Efetividade Total: para a média (μ) **6,43**, desvio padrão (σ) **8,31**, a efetividade total para a aplicação da abordagem X resultou em p = **0,8725** para o teste de *Shapiro-Wilk*.

  Assim, para um nível de significância (α = 0.05), p = **0,8725** (**0,8725** > 0.05) e W = **0,9665** > W = **0,8030**, a amostra total é considerada normal.

Abordagem Y (N = 7):

- Efetividade para a LPS VU: para a média (μ) **7,97**, desvio padrão (σ) **1,93**, a efetividade para a aplicação da abordagem Y na LPS VU resultou em p = **0,6577**, ao ser aplicado ao teste de normalidade *Shapiro-Wilk*.

  O teste indicou, para uma amostra *(N)* de tamanho 7 com 95% de nível de significância (α = 0,05), p = **0,6577**, (0,6577 > 0,05) e o valor calculado de W = **0,9421** > W = **0,8030**, que a amostra é considerada normal;

- Efetividade para a LPS MM: para a média (μ) **1,23**, desvio padrão (σ) **3,84**, a efetividade para a aplicação da abordagem Y na LPS MM resultou em p = **0,8693**, ao ser aplicado ao teste de normalidade *Shapiro-Wilk*.

  Assim, para (α = 0.05), p = **0,8693** (0,8693 > 0,05) e o valor calculado (W), W = **0,9661** > W = **0,8030**, indica que a amostra é considerada normal;

- Efetividade Total: para a média (μ) **9,26**, desvio padrão (σ) **5,01**, a efetividade total para a aplicação da abordagem Y resultou em p = **0,5797** para o teste de *Shapiro-Wilk*.

  Assim, para um nível de significância (α = 0.05), p = **0,5797** (**0,5797** > 0.05) e W = **0,9333** > W = **0,8030**, a amostra total é considerada normal.

O Teste T foi aplicado à amostra de efetividade das abordagens X e Y. Nesse estudo, as amostras são independentes, mas o Teste T pode ser aplicado também à amostras dependentes ou grupos relacionados.

Utilizando o valor da média da amostra X (μ1 = 6,43) e da amostra Y (μ1 = 9,26), com desvio padrão de σ1 = 8,31 e σ2 = 3,84, e amostra de tamanho 7 (N = 7), foi obtido o valor de $t_{calculado} = 0,7711$.

Utilizando o tamanho da amostra (N = 7), é calculado o grau de liberdade (df), que combinado com o valor de t indica qual o valor de t crítico que deve ser selecionado para o teste de hipóteses.

Buscando pelo índice df = 12 e definindo o valor de t na tabela T (*student*), o valor crítico de t foi encontrado, que corresponde a 2,179 ($t_{critico} = 2,179$), para o nível de significância (α) de 0,05. Isto significa que a probabilidade de $t_{calculado}$ (0,7711) < $t_{critico}$ (2,179) é 95%. Existe diferença significativa entre as amostras de X e Y, o que indica que é possível rejeitar $H_{0.QP1}$ com α = 0,05. Portando, a hipótese $H_{1.QP1}$ deve ser aceita.

Tabela 4.3: Correlação de *Spearman* entre a Efetividade das Abordagens X e Y em relação ao nível de conhecimento dos participantes em LPS.

Abordagem X (Razavian and Khosravi)						
Identific.	Efetividade	r_{a1}	Nível de Conhecimento	r_{a2}	r_{a1}-r_{a2}	$d1^2$
1	9,80	3	4	4,5	-1,5	2,25
2	-1,60	6	4	4,5	1,5	2,25
3	13,00	2	5	1,5	0,5	0,25
4	17,20	1	5	1,5	-0,5	0,25
5	8,80	4	4	4,5	-0,5	0,25
6	4,40	5	4	4,5	0,5	0,25
7	-6,60	7	3	7	0	0

Abordagem Y (SMarty 5.2)						
Identific.	Efetividade	r_{b1}	Nível de Conhecimento	r_{b2}	r_{b1}-r_{b2}	$d2^2$
1	15,20	2	4	3,5	-1,5	2,25
2	9,40	3	4	3,5	-0,5	0,25
3	8,20	4	3	6	-2	4
4	6,40	6	5	1	5	25
5	2,20	7	2	7	0	0
6	7,00	5	4	3,5	1,5	2,25
7	16,40	1	4	3,5	-2,5	6,25

4.5.3 Análise de Correlação para Q.P.2

Para calcular a correlação entre a efetividade das abordagens e a influência do conhecimento prévio dos participantes foi utilizada a correlação de *Spearman*.

- **Correlação de *Spearman*:** Esta é uma técnica não paramétrica aplicada com o intuito de verificar a possível correlação entre dois conjuntos de valores, nesse caso, a efetividade das abordagens X (*Razaviand e Khosravi*) e Y (*SMarty* 5.2) e o nível de conhecimento dos participantes em LPS. A Fórmula 4.1 apresenta o cálculo de ρ da correlação de *Spearman*, onde n representa o tamanho da amostra:

$$\rho = 1 - \frac{6}{n(n^2-1)} \sum_{i=1}^{n} d_i^2 \tag{4.1}$$

  Os dados necessários para o cálculo da correlação de *Spearman* estão apresentados na Tabela 4.3, contendo a efetividade das abordagens X e Y e o nível de conhecimento dos participantes em LPS. Os valores r_a e r_b foram obtidos após a ordenação decrescente dos valores da efetividade e respectivamente do nível de conhecimento dos participantes em LPS. Em situação de empate entre os valores, as médias das posições foram calculadas e classificadas.

  As Equações abaixo apresentam o cálculo da correlação de *Spearman* para as abordagens X (4.2) e Y (4.3):

$$\rho(Corr.1) = 1 - \frac{6}{7(7^2-1)} * 5.5 = 1 - 0,10 = \mathbf{0,90} \tag{4.2}$$

$$\rho(Corr.2) = 1 - \frac{6}{7(7^2 - 1)} * 40 = 1 - 0,71 = \mathbf{0,29} \quad (4.3)$$

Dessa forma, os valores de ρ foram obtidos e analisados, com base na escala de classificação de *Spearman*, de acordo com a Figura 4.4:

- *Correlação da abordagem X, entre a sua efetividade e o nível de conhecimento dos participantes em LPS:* $\rho = 0{,}90$ = Correlação positiva forte; e
- *Correlação da abordagem Y, entre a sua efetividade e o nível de conhecimento dos participantes em LPS:* $\rho = 0{,}29$ = Correlação positiva fraca;

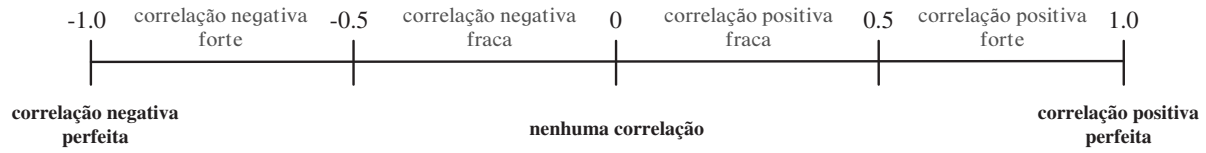

Figura 4.4: Escala de Correlação de *Spearman*

4.6 Discussão

Esta seção apresenta uma discussão em relação aos resultados obtidos e implicações, bem como as ameaças de validade para os experimentos realizados.

4.6.1 Avaliação dos Resultados e Implicações

Os resultados do teste de normalidade e de hipótese para Q.P.1 foram obtidos com base no resultado do teste T, apresentados na Seção 4.5.2. Portanto, a hipótese nula ($H_{0.QP1}$) deve ser rejeitada, e a hipótese alternativa ($H_{1.QP1}$) aceita, o que fornece evidências de que a abordagem Y (*SMarty*) é mais efetiva que a abordagem X (*Razavian e Khosravi*), para representar variabilidades em componentes UML, nessa avaliação experimental. Assim, a hipótese $H_{1.QP1}$ foi aceita.

Baseado nessa análise, podemos observar que *SMarty* 5.2 tem um resultado superior corroborado com o fato que: (i) *SMarty* evidência ser mais efetiva em modelar detalhes de variabilidades de baixa granularidade, tal como, números de variantes mínimos e máximos a serem selecionados e o conjunto de possíveis escolhas de variantes, enquanto a abordagem

de *Razavian e Khosravi* não possui tais detalhes; (ii) considerando esses detalhes, *SMarty* evidência possuir uma capacidade maior de derivar produtos específicos consistentes sem a intervenção de especialistas do domínio; e (iii) sem a intervenção de especialistas de domínio, o processo de derivação de produtos específicos consistentes pode ser totalmente automatizado no *SMarty*, (pelo fato de *SMarty* possuir meta-atributos que representam variabilidades. Assim, tal automatização poderia ser facilitada) enquanto a derivação de produtos por meio do diagrama de componentes de variabilidades de *Razavian e Khosravi* exige um estágio humano-supervisionado prospectivo, pois fornece informações insuficientes.

Em uma comparação entre a versão do *SMarty* 5.1 e a abordagem de *Razavian e Khosravi*, acredita-se que a abordagem de *Razaian e Khosravi* poderia fornecer uma enorme eficácia, pois *SMarty* 5.1 era composta por somente um estereótipo para componentes «`variable`» e não fornece mecanismos para derivar produtos específicos com base em tal modelagem. Assim, corrobora que *SMarty* necessitava evoluir para a versão 5.2.

A partir da análise da correlação entre a efetividade das abordagens em relação ao conhecimento prévio do participante em LPS e variabilidade, pode-se perceber que os resultados obtidos por meio da correlação de *Spearman* apresentam uma diferença considerável.

A *Corr.1* (positiva forte) fornece indícios de que o nível de conhecimento dos participantes pode ter influenciado a aplicação do método *Razavian e Khosravi* e seus estereótipos para identificar e representar variabilidades no diagrama de componentes. Enquanto isso, a *Corr.2* (positiva fraca) fornece indícios de que o nível de conhecimento dos participantes quase não influenciou, ou influenciou pouco, a aplicação da abordagem *SMarty* e seus estereótipos para a identificação e representação de variabilidades em componentes da UML.

Tais resultados nos levam a acreditar que as diretrizes que *SMarty* fornece podem ter minimizado os esforços dos partificipantes durante a aplicação de *SMarty*, de acordo com a correlação obtida sobre a influência do conhecimento prévio obtido por eles. Para afirmar tais indícios, outras avaliações futuras devem ser conduzidas.

4.6.2 Ameaças à Validade

Esta seção apresenta as ameaças à validade identificadas durante este estudo experimental. Para cada ameaça, foram descritas as ações tomadas para resolvê-las, de acordo com Neto e Conte (2013):

- **Ameaças à Validade Interna**: (i) em relação a distribuição dos participantes para as abordagens nesse estudo, o questionário de caracterização serviu de apoio para o devido balanceamento dos grupos, buscando assim minimizar as ameaças referente ao conhecimento prévio dos participantes em relação à abordagem a ser direcionada; (ii) os participantes selecionados para fazer o experimento com a abordagem X não poderiam ter acesso ao conteúdo da abordagem Y, e nem os participantes da abordagem Y ter acesso à abordagem X, a fim de evitar possíveis viés. Portanto, os participantes foram divididos em grupos equilibrados de acordo com o conhecimento prévio dos participantes, obtidos por meio do questionário de caracterização; (iii) os participantes não tiveram contato prévio com a abordagem selecionada, portanto, um treinamento foi realizado a fim de tirar as possíveis dúvidas com relação à tal abordagem; e (iv) Para cada abordagem, os participantes teriam que modelar duas LPSs, além de participar das sessões de treinamento. Assim, com o intuíto de evitar a fadiga dos participantes, as sessões de treinamento e o experimento em si foram realizados em dias diferentes, com uma média de 30 minutos cada;

- **Ameaças à Validade Externa**: Duas ameaças foram detectadas: (i) os diagramas de componentes das LPSs não são reais, assim estudos adicionais devem considerar LPSs reais; e (ii) apesar de acadêmicos de mestrado e doutorado serem selecionados ao invés de profissionais da indústria, eles não foram considerados um viés para este estudo como a importância do uso de alunos em estudos experimentais (Höst *et al.*, 2000);

- **Ameaças à Validade de *Construto***: a construção do experimento foi baseada nas características de cada LPS selecionada e na avaliação da instrumentação durante o projeto piloto. Assim, a fim de reduzir os possíveis viés em relação à seleção de uma LPS que favoreça uma determinada abordagem, foram selecionadas duas LPSs, sendo que essas foram previamente utilizadas por cada abordagem em outros contextos; e

- **Ameaças à Validade de Conclusão**: o tamanho da amostra (n = 14) é pequena, do ponto de vista estatístico. Ainda assim, o tamanho teve que ser reduzido pela necessidade de avaliar as duas abordagens separadamente. Uma vez que o número de participantes é reduzido, os dados extraídos a partir destes estudos só podem ser considerados indicadores e não conclusivos (Marcolino e OliveiraJr, 2015). No entanto, de acordo com Wohlin *et al.* (2012) pode não ser possível obter amostras homogêneas, nesse caso, as conclusões estatísticas podem ser obtidas com menos

significância. Portanto, acredita-se que o tamanho da amostra deve ser aumentada em replicações futuras.

4.7 Pacote Experimental da Avaliação

Toda a instrumentação utilizada nessa avaliação experimental é disponibilizada a fim de promover possíveis replicações futuras, além porporcionar a transferência do conhecimento tácito dos experimentadores (Mendonca, 2008; Shull, 2004).

Para realizar o *download* do pacote experimental da avaliação, acesse: `http://www.din.uem.br/~smarty/trab3.html`.

4.8 Considerações Finais

A evolução de *SMarty* 5.2 foi utilizada nessa avaliação empírica, de forma que as suas modificações puderam ser avaliadas, considerando a sua efetividade para componentes UML em relação ao método de *Razavian e Khosravi*.

Ainda, por meio da correlação de *Spearman*, foi possível evidenciar que houve uma baixa correlação entre a efetividade de *SMarty* 5.2 e o nível de conhecimento prévio dos participantes em LPS e variabilidades, o que indica que a modelagem de *SMarty* 5.2 não foi afetava pelo conhecimento prévio dos particiantes. Já em relação ao método de *Razavian e Khosravi*, a correlação obtida foi positiva forte, o que indica que a modelagem utilizando o método de *Razavian e Khosravi* possívelmente foi afetava pelo conhecimento prévio dos participantes.

O próximo capítulo apresenta o processo *SMartyComponents*. A evolução de *SMarty* 5.2 colaborou com a formalização de tal processo.

5

SMartyComponents um Processo para a Especificação de Arquitetura de LPS

"Existem coisas melhores adiante do que qualquer outra que deixamos para trás."

C.S. Lewis (1898 - 1963), Escritor

5.1 Considerações Iniciais

As vantagens que a abordagem de LPS possui combinadas com os benefícios que o DBC proporciona, podem guiar os *stakeholders* a especificarem ALPSs baseadas em componentes, resultando assim em uma abstração de todos os possíveis produtos que uma LPS possa gerar. Dessa forma, a combinação do método *UML Components* com a abordagem *SMarty* para gerenciar variabilidades, permite a especificação sistemática de ALPSs baseadas em componentes com representação explícita de suas variabilidades ao longo do ciclo de vida da LPS.

Para a proposta do processo *SMartyComponents* foi necessária a adaptação de vários artefatos provenientes do *UML Components*, bem como da respectiva evolução de *SMarty* para apoiar a representação de variabilidades em componentes de acordo com a UML 2.5.

Este capítulo apresenta o processo *SMartyComponents*. A modelagem do processo *SMartyComponents* foi feita com a notação *SPEM* (conceitos básicos no Apêndice B). Para ilustrar o *SMartyComponents* é utilizada a LPS *Arcade Game Maker* (AGM) (Anexo D).

5.2 Caracterização do Processo *SMartyComponents*

O *SMartyComponents* concentra-se nos *workflows* de Requisitos e Especificação por entender que são os principais estágios para a especificação de uma ALPS. Os artefatos de saída dos *workflows* de Requisitos e de Especificação do *UML Components* servem de entrada para o processo *SMartyProcess*, o qual identifica e representa variabilidades em modelos UML, permitindo a especificação de ALPS baseada em componentes. A Figura 5.1 apresenta os *workflows* do *UML Components* com destaque para os *workflows* de Requisitos e de Especificação, considerados pelo *SMartyComponents*. Os demais *workflows* do *UML Components* estão fora do escopo desta proposta.

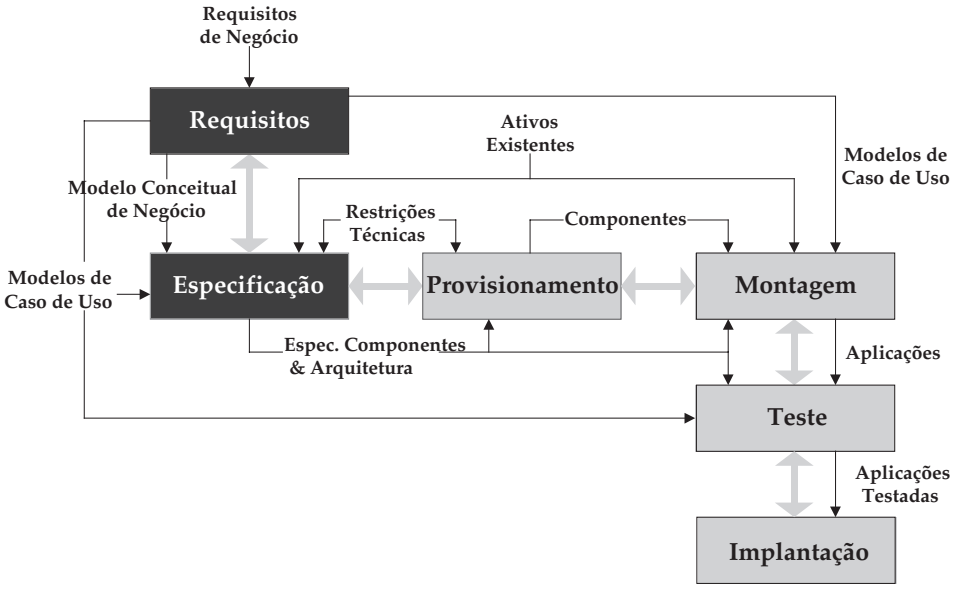

Figura 5.1: Workflows do *UML Components* para o *SMartyComponents* - Adaptada de Cheesman e Daniels (2001).

A Figura 5.2(a) apresenta as atividades dos *workflows* de Requisitos e de Especificação do *UML Components*. Para o *SMartyComponents*: (i) um *workflow* é composto por atividade(s); (ii) as atividades são compostas por um conjunto de tarefas, papéis e artefatos; e (iii) tarefas consomem e geram artefatos, e são realizadas por algum papel. A

Figura 5.2(b) apresenta as atividades que compõem os *workflows* do *SMartyComponents*. Note que todas as atividades produzem e consomem artefatos do *SMartyProcess*, uma vez que este é responsável por identificar e representar variabilidades em modelos UML.

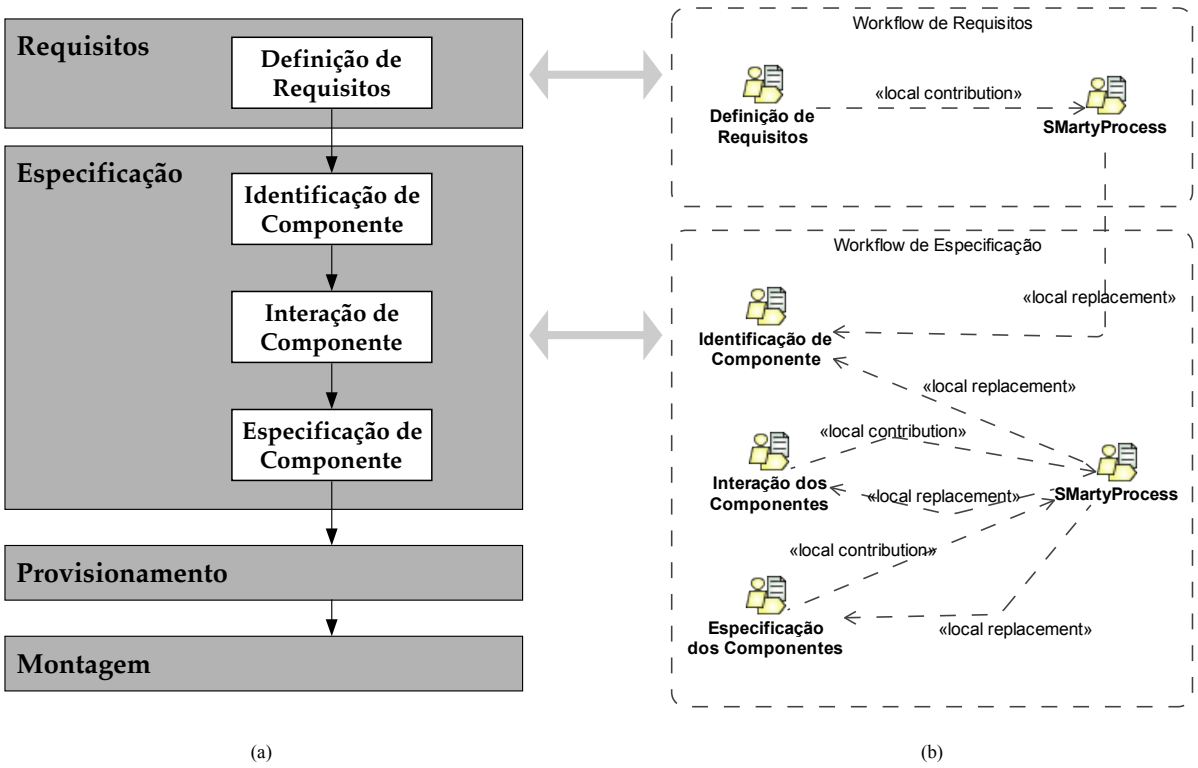

Figura 5.2: (a) Atividades dos *workflows* do *UML Components* - Traduzida de Cheesman e Daniels (2001); e (b) Modelagem em *SPEM* das atividades dos *workflows* do *SMartyComponents*.

A Figura 5.3 apresenta os *workflows* que o *SMartyComponents* suporta, bem como os papéis que estão envolvidos nas atividades e tarefas de cada *workflow*, e os artefatos de entrada e de saída de cada *workflow*.

Nas próximas seções será detalhado cada *workflow* do *SMartyComponents*, as atividades e tarefas que os compõem, bem como a representação dos papéis e suas responsabilidades em cada uma das tarefas.

5.3 O Workflow de Requisitos do SMartyComponents

De acordo com o *UML Components* (Cheesman e Daniels, 2001), o *workflow* de Requisitos não é considerado para a captura de requisitos em si. Neste *workflow*, assume-se que o usuário entenda o conceito do processo de negócio que se deseja apoiar. Tal processo

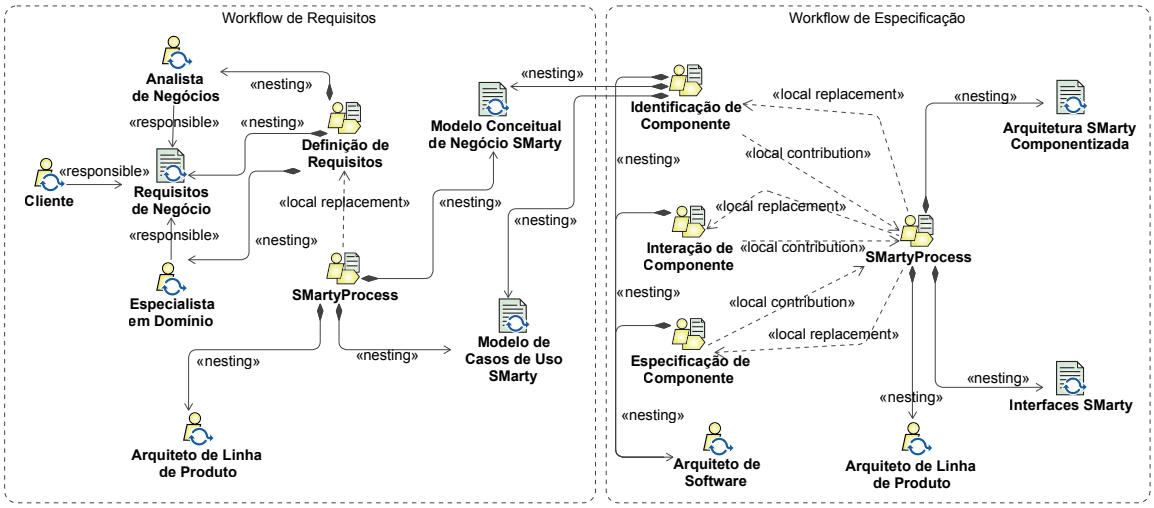

Figura 5.3: Workflows do *SMartyComponents* e suas Atividades, Artefatos e Papéis.

de negócio leva em consideração as etapas que o projeto deve possuir, bem como uma descrição do conceito do negócio em si, mesmo que todas as restrições do negócio ainda não tenham sido formuladas. A descrição do processo de negócio deve incluir os termos de domínio, no qual se deve ter clareza. Essa descrição do processo de negócio deve ser fornecida pelo cliente, e adaptada pelo Analista de Negócios e Especialista em Domínio.

5.3.1 Atividade: *Definição de Requisitos*

A atividade que compõe o *workflow* de requisitos é a `Definição de Requisitos` (Figura 5.4), que tem como entrada os artefatos `Conhecimento do domínio da LPS` e `Requisitos de Negócio`. O artefato `Conhecimento do domínio da LPS` e os processos de negócio identificados, são consumidos pela tarefa `Desenvolver Modelo Conceitual de Negócio`, realizada pelo Especialista em Domínio e/ou pelo Analista de Negócios. Já o artefato `Requisitos de Negócio` é consumido em duas tarefas realizadas pelo Analista de Negócios: (i) `Descrever Processos de Negócios`, que conta com a participação do Especialista em Domínio; e (ii) `Identificar Casos de Uso`. A Figura 5.4 apresenta a composição do *workflow* de requisitos.

Tarefa: *Desenvolver Modelo Conceitual de Negócio*

A partir do conhecimento do domínio da LPS e dos processos de negócio, o `Analista de Negócios` e/ou o *Especialista em Domínio* iniciam a tarefa de desenvolver o modelo conceitual de negócio. Chama-se de `Modelo Conceitual de Negócio` um mapeamento

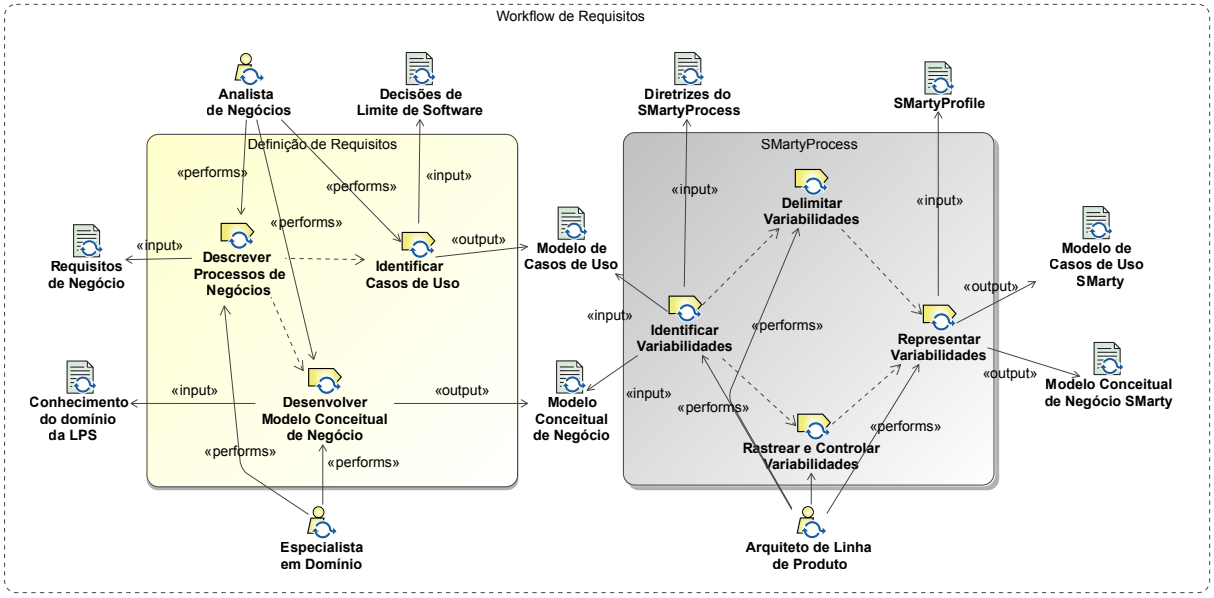

Figura 5.4: *Workflow* de Requisitos do *SMartyComponents*

que relaciona os termos contidos na descrição do processo de negócio e outros termos importantes, ainda sem nenhuma representação de variabilidades.

Este modelo é desenvolvido utilizando o diagrama de classes da UML, conforme sugerido em Cheesman e Daniels (2001). Neste momento o modelo conceitual de negócios não precisa ser bem detalhado, pois nesta tarefa não serão revelados quaisquer propriedades ou operações, mas pode-se mostrá-las caso elas surjam naturalmente durante a criação do modelo.

O `Conhecimento do Domínio da LPS` é um artefato que armazena o entendimento e conhecimento sobre como os envolvidos compreendem o domínio, além de captar as regras de negócio sobre o conhecimento da LPS. Para isso, pode-se utilizar técnicas como *storyboards* ou *mind-maps* (Beel e Gipp, 2010), com o objetivo de representar diferentes situações e cenários do domínio da LPS.

O `Especialista em Domínio`, por meio das informações do processo de negócio identificadas, desenvolve um modelo conceitual do negócio inicial, que tende a ser reavaliado no próximo *workflow*.

O artefato `Modelo Conceitual de Negócio` gerado alimenta o *SMartyProcess*, que realiza as tarefas de identificar e representar as variabilidades em tal modelo. A identificação e representação de variabilidades, que se apoia nas diretrizes para classes UML propostas pelo *SMartyProcess*, guiarão o Arquiteto de LPS. As variabilidades são representadas por meio do conjunto de estereótipos do *SMartyProfile*. Assim, o *SMartyProcess* gera

como saída o `Modelo Conceitual de Negócio SMarty`, que é uma evolução do `Modelo Conceitual de Negócio` com variabilidades representadas.

Exemplo: Artefato *Modelo Conceitual de Negócio*

O `Modelo Conceitual de Negócio` da LPS AGM se baseia nas seguintes descrições:

- um jogador (`Player`) pode jogar no mínimo um e no máximo três jogos (`Game`);
- para cada jogo, deve existir no mínimo uma e no máximo várias regras (`Rule`);
- cada regra pode envolver pontuação (`Score`);
- cada jogo possui uma mesa de jogo (`GameBoard`);
- cada mesa de jogo, possui no mínimo um obstáculo (`Sprite`);
- cada mesa de jogo pode conter no mínimo uma partida (`Match`); e
- cada partida pode conter uma pontuação (`Score`).

De acordo com tais descrições, o `Especialista em Domínio` realiza a atividade `Desenvolver Modelo Conceitual de Negócio`, e a partir dessa atividade o artefato `Modelo Conceitual de Negócio` é gerado. A Figura 5.5 apresenta o exemplo de tal artefato para a AGM.

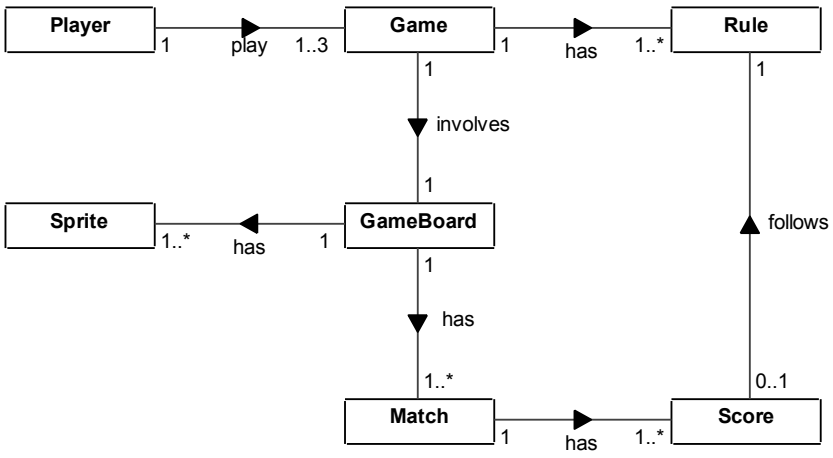

Figura 5.5: `Modelo Conceitual de Negócio` da AGM.

Após a sua criação, tal artefato serve como entrada para o *SMartyProcess*. O Arquiteto de LPS realiza as tarefas de identificar, delimitar e representar variabilidades, por meio

do *SMartyProfile*. Na tarefa `Identificar Variabilidades` o Arquiteto de LPS leva em consideração as `Diretrizes do SMartyProcess` para modelos de classes UML. A Figura 5.6 apresenta o artefato gerado pelo *SMartyProcess*, o `Modelo Conceitual de Negócio SMarty`.

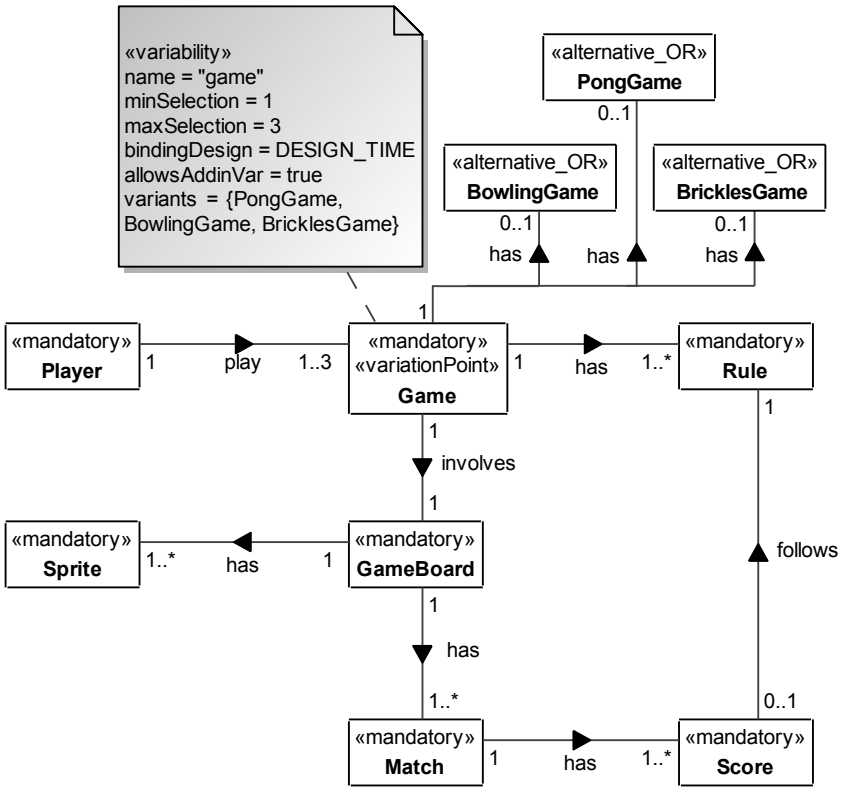

Figura 5.6: Artefato `Modelo Conceitual de Negócio SMarty` da AGM

Como o *SMartyProcess* é um processo iterativo e incremental é possível que na medida em que os artefatos das tarefas evoluem, novos pontos de variação e variantes sejam identificados e modelados.

Tarefa: *Descrever Processos de Negócios*

O objetivo dessa tarefa é compreender os processos de negócio, a fim de realizar a tarefa `Identificar Casos de Uso`.

Para descrever os processos de negócio é fundamental compreender as funcionalidades do negócio. Assim, o artefato `Requisitos de Negócio` é relevante para discernir os processos de negócio. Tal artefato contém informações que são identificadas em reuniões e *workshops* realizados com os clientes, e entende-se que tais informações já foram

discutidas entre os envolvidos no projeto, analistas, especialistas, clientes e usuários. Tais informações são relevantes para realizar a tarefa `Descrever Processos de Negócios`.

O processo de negócio pode ser modelado utilizando um diagrama de atividades UML, de maneira que torne mais clara a interpretação da lógica do negócio. Essa sugestão não faz parte do contexto do processo proposto, portanto, se os envolvidos preferirem, podem utilizar os modelos com os quais estão mais familiarizados como, por exemplo, modelos em BPMN (Cheesman e Daniels, 2001).

Tarefa: *Identificar Casos de Uso*

Esta tarefa tem por objetivo identificar os casos de uso do sistema. Os processos de negócio identificados, guiam o Analista de Negócios a identificar os casos de uso.

Os casos de uso são responsáveis por apresentar como o sistema cumprirá as suas responsabilidades, esclarecer quais são os seus limites, identificar os atores que interagem com o sistema e descrever essas interações. Os casos de uso são criados a partir dos eventos que desencadeiam uma série de etapas do processo, assim, todos os passos formam um único caso de uso.

Ao final dessa tarefa o `Modelo de Casos de Uso` alimenta o *SMartyProcess*, que inicia a tarefa de identificar as variabilidades. Quando se trata de diagramas de casos de uso, o Arquiteto da LPS toma como base as diretrizes para casos de uso do *SMartyProcess*. Após identificar as variabilidades contidas nos casos de uso, o Arquiteto da LPS delimita as possíveis variabilidades e em seguida, por meio do *SMartyProfile*, representa as variabilidades, gerando assim casos de uso com variabilidades representadas. Cada caso de uso gerado faz parte do `Modelo de Casos de Uso SMarty`.

Exemplo: Artefato *Modelo de Casos de Uso*

A LPS AGM possui alguns requisitos (Anexo D) necessários para compreender os processos de negócio e para a identificação dos casos de uso.

De acordo com os requisitos da LPS AGM:

- um usuário (`GamePlayer`) poderá selecionar um jogo para jogar (`Play Selected Game`), salvar o jogo (`Save Game`), sair do jogo (`Exit Game`), salvar pontuação (`Save Score`) e checar a melhor pontuação anterior (`Check Previous Best Score`);
- outro usuário (`GameInstaller`) será responsável por instalar o jogo (`Install Game`), desintalar o jogo (`Uninstall Game`), selecionar um jogo para jogar (`Play Selected Game`), sair do jogo (`Exit Game`) e salvar o jogo (`Save Game`);

- o usuário poderá selecionar um jogo do tipo *Brickles* (`Play Brickles`), *Pong* (`Play Pong`) ou *Bowling* (`Play Bowling`); e
- cada jogo necessita ser inicializado (`Initialization`) e também de animações (`Animation Loop`).

De acordo com os requisitos da LPS AGM, a Figura 5.7 apresenta o artefato gerado `Modelo de Casos de Uso`.

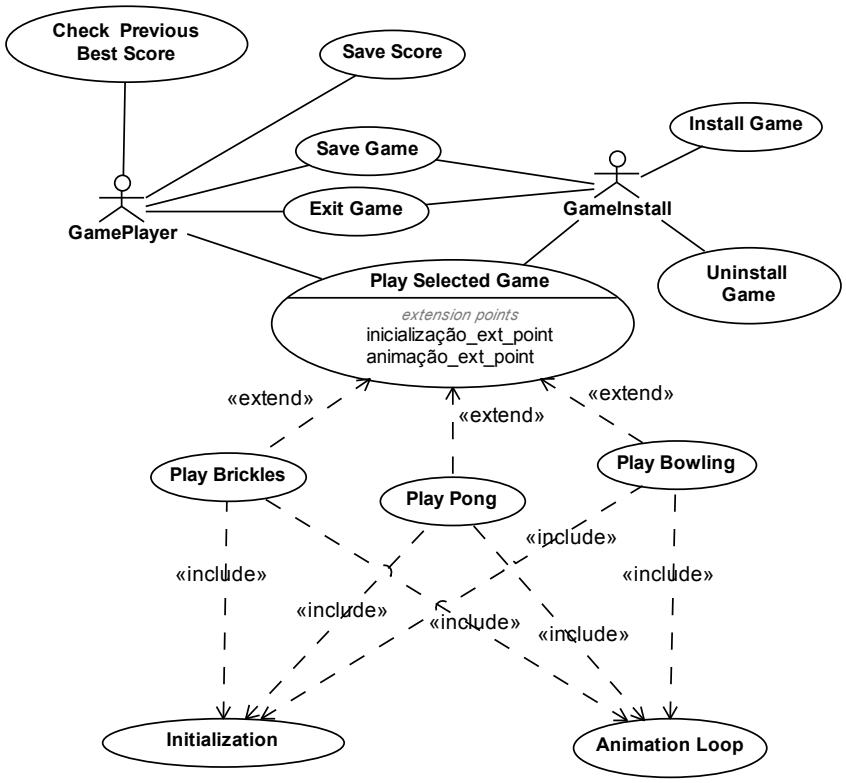

Figura 5.7: `Modelo de Casos de Uso` da AGM

Após a criação do `Modelo de Casos de Uso`, tal artefato serve como entrada para o *SMartyProcess*, em que o Arquiteto de LPS identifica, delimita e representa as variabilidades, de acordo com as diretrizes para modelos de casos de uso.

Seguindo o *SMartyProcess*, alguns dos casos de uso são obrigatórios para a existência da LPS AGM, como o `InstallGame`, referente à instalação do jogo; `UnistallGame`, referente à desinstalação do jogo; `Exit Game`, referente à opção de sair do jogo; `Save Game`, referente à opção de salvar o jogo; `Play Selected Game`, referente ao jogo que será selecionado; `Initialization` referente à inicialização do jogo selecionado; `Animation Loop`, referente à animação correspondente ao jogo selecionado; o ator `GamePlayer`,

referente ao jogador; e o ator `GameInstaller`, referente ao usuário que instala e desinstala jogos.

O caso de uso `Check Previous Best Score` é uma opção de jogo para mostrar os melhores resultados dos jogadores; e o caso de uso `Save Score` referente a salvar a pontuação. Esses casos podem ser identificados como opcionais, porém o `Save Score` só pode existir caso exista o `Check Previous Best Score`.

O caso de uso `Play Selected Game` é identificado como ponto de variação, pelo fato de possuir três outros casos de usos que são suas variantes.

De acordo com as diretrizes do *SMartyProcess* e os estereótipos do *SMartyProfile*, a Figura 5.8 apresenta o artefato `Modelo de Casos de Uso SMarty`.

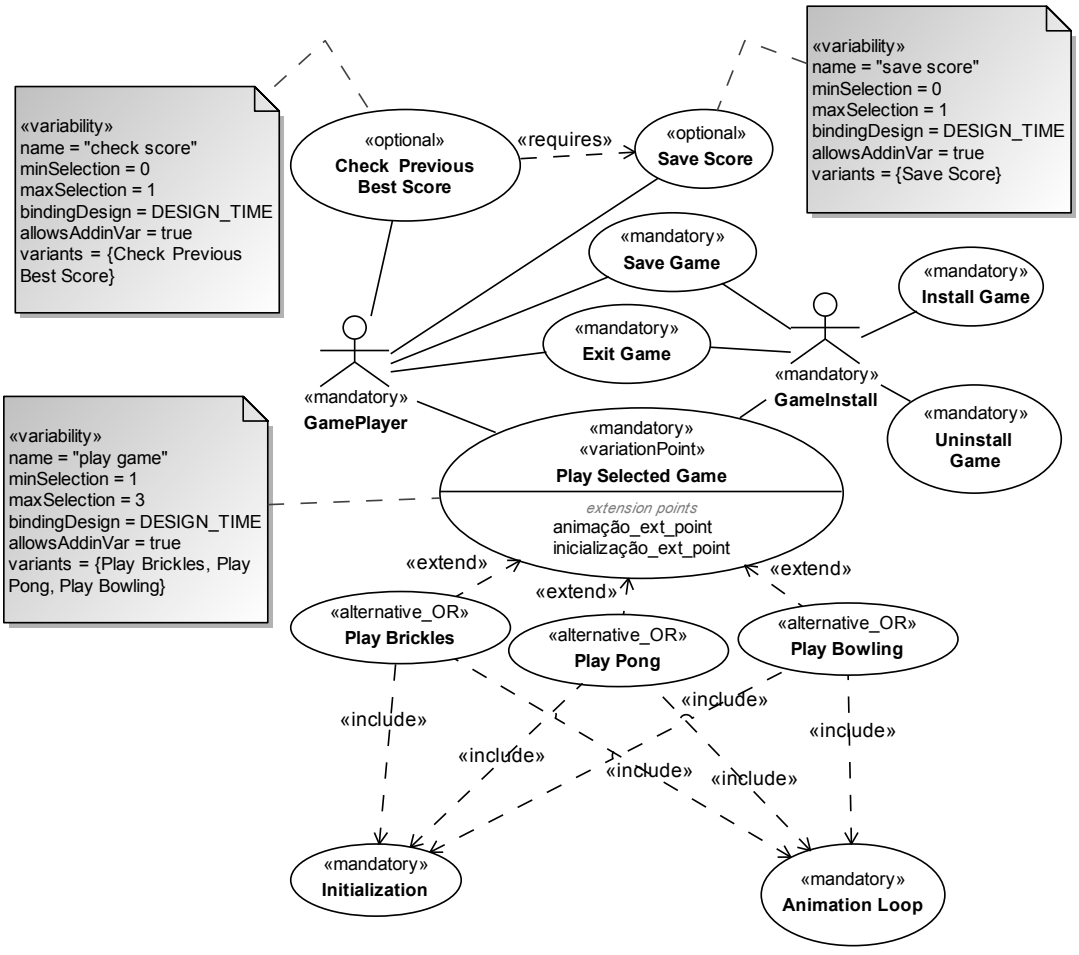

Figura 5.8: `Modelo de Casos de Uso SMarty` da AGM

5.4 O Workflow de Especificação do SMartyComponents

Neste *workflow* as informações referentes aos conceitos de negócio e aos casos de uso são consumidos a fim de gerar a especificação da ALPS componentizada e a especificação das interfaces.

Como visto na Figura 5.2 (b), este *workflow* é dividido em três atividades: (i) Identificação de Componente; (ii) Interação de Componente; e (iii) Especificação de Componente. A Figura 5.9 apresenta o fluxo das atividades deste *workflow*, mostrando as entradas para o *SMartyProcess* e as saídas geradas por ele ao final do *workflow*. Nas seções a seguir, cada atividade será apresentada de forma a detalhar suas tarefas, papéis e artefatos gerados.

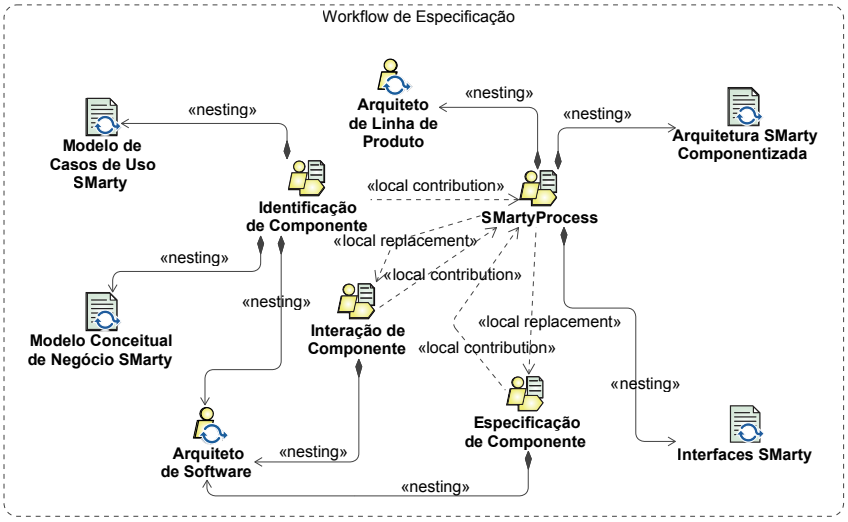

Figura 5.9: Atividades do *workflow* de Especificação do *SMartyComponents*

5.4.1 Atividade: *Identificação de Componente*

Nesta atividade é realizada a identificação dos componentes. O papel Arquiteto de Software é responsável por realizar as tarefas da atividade `Identificação de Componente`, enquanto o Arquiteto de LPS é responsável pelas tarefas do *SMartyProcess*. Com base no `Modelo de Casos de Uso SMarty`, inicia-se a identificação das interfaces do sistema e suas operações, e cria-se uma especificação inicial dos componentes e da arquitetura. Com base no `Modelo Conceitual de Negócio SMarty`, inicia-se o desenvolvimento do `Modelo de Tipos de Negócio` e a identificação das `Interfaces de Negócio`. A Figura 5.10 apresenta os artefatos, tarefas e papéis que compõem a atividade `Identificação de Componente`.

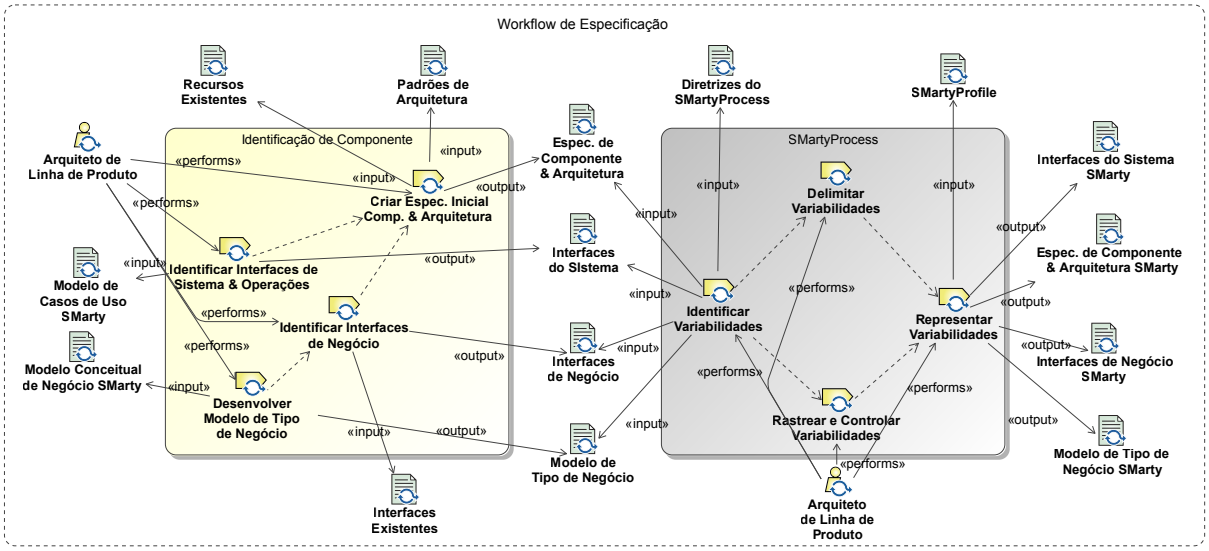

Figura 5.10: *Workflow* de Especificação: Identificação de Componente

Tarefa: *Desenvolver Modelo de Tipos de Negócio*

O objetivo dessa tarefa é gerar o `Modelo de Tipos de Negócio` a partir dos conceitos de negócios identificados no *workflow* de Requisitos. Essa tarefa é realizada pelo Arquiteto de Software.

O `Modelo de Tipos de Negócio` é representado por um modelo de classes UML, assim como o `Modelo Conceitual de Negócio`, mas sua proposta é diferente. Enquanto o `Modelo Conceitual de Negócio` é um simples mapeamento das informações de interesse do domínio da LPS, o `Modelo de Tipos de Negócio` contém as informações de negócios específicas que podem ajudar a guiar a especificação da ALPS.

A criação do `Modelo de Tipos de Negócio` se inicia com uma derivação do artefato `Modelo Conceitual de Negócio SMarty`. Assim, pode-se adicionar ou remover os elementos que fazem parte do escopo desejado. Os elementos restantes nesta etapa serão anotados com o estereótipo «`type`» do *UML Components*. Em seguida, deve-se definir os tipos de dados de cada atributo, um conjunto de tipos de dados para usar neste modelo e as restrições do modelo, por meio de relacionamentos e multiplicidades.

Nesta tarefa as restrições definidas não precisam ser formais. Na atividade de `Especificação de Componente` deste *workflow* é que serão produzidas as especificações formais. Essas restrições não formais identificadas podem ser incluídas em notas da UML, deixando claro quais são para futuras implementações.

Após alterações e inclusões de restrições no `Modelo de Tipos de Negócio`, deve-se decidir quais são os tipos que devem ser considerados núcleo (*core*). O objetivo é identificar

quais elementos dependem exclusivamente de outros e quais elementos são independentes. Esta informação é útil para alocar responsabilidades de informação para as interfaces. Para tanto, nos núcleos identificados o estereótipo «`type`» é substituído por «`core`».

Deve-se lembrar de manter as variabilidades que foram definidas no *workflow* de Requisitos. Tais variabilidades evoluirão, se necessário, no *SMartyProcess*.

Definido o modelo de tipo de negócio, tal artefato segue como entrada para o *SMartyProcess*, que refinará a identificação de variabilidades, por meio de suas diretrizes para diagrama de classes UML. Após tal identificação, o Arquiteto de LPS delimita e representa as variabilidades, gerando assim o `Modelo de Tipos de Negócio SMarty`.

Exemplo: Artefato *Modelo de Tipos de Negócio*

No caso da LPS AGM, de acordo com o artefato `Modelo Conceitual de Negócios SMarty` (Figura 5.6), os elementos `Player` e `Rule` foram removidos, e foram adicionados os elementos contidos no ponto de variação `Game`, e no elemento `Sprite`. Foram identificados dois tipos de `Sprite`: (i) `MovableSprite`, referente aos elementos que se movem durante os jogos; e (ii) `StationarySprite`, referente aos elementos que ficam parados durante os jogos.

Para a `Sprite` do tipo `MovableSprite`, foram identificados três tipos: (i) `Puck`, referente ao objeto que recebe uma colisão; `Paddle` referente ao objeto que se movimenta vertical ou horizontalmente; e `Velocity`, referente à velocidade dos elementos. Para a `Sprite` do tipo `StationarySprite` foram identificados 5 tipos: (i) `Ceiling`, referente ao teto do jogo; (ii) `Floor`, referente ao chão do jogo; (iii) `Wall`, referente às paredes do jogo; (iv) `BrickPile`, referente à pilha de tijolos; e (v) `Brick`, referente aos tijolos.

Os elementos que possuem algum tipo de dependência de outros elementos, ou que não existiriam sem a presença de outros foram anotados com o estereótipo «`type`» do *UML Components*. Já os elementos que não possuem dependência de outros, ou que existiriam sem a presença de outros, foram anotados com o estereótipo «`core`» do *UML Components*.

A Figura 5.11 apresenta o artefato `Modelo de Tipos de Negócio` da LPS AGM, de acordo com as especificações citadas. É importante observar que os estereótipos identificados anteriormente pelo *SMartyProcess*, permancem inalterados. As possíveis modificações necessárias serão realizadas na próxima atividade do *SMartyProcess*.

Após gerar o artefato `Modelo de Tipos de Negócio`, tal artefato é tomado como entrada pelo *SMartyProcess*, que identifica as variabilidades, por meio de suas diretrizes para classes. Após identificar, o Arquiteto de LPS delimita e representa as variabilidades

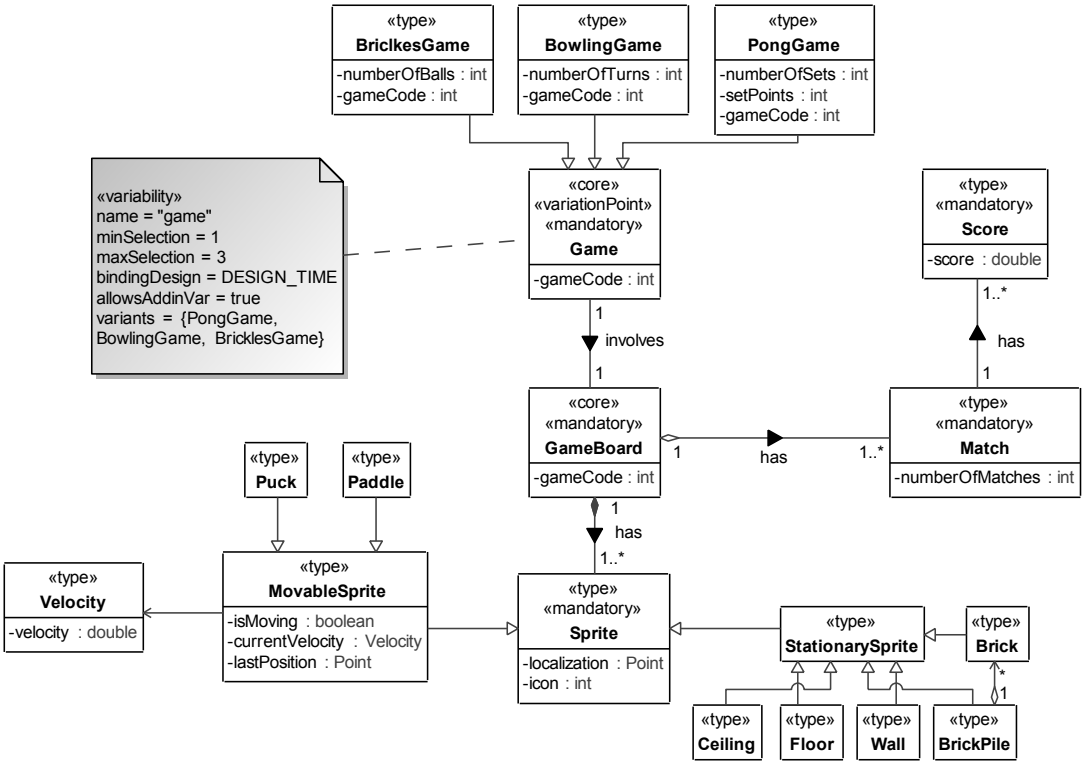

Figura 5.11: `Modelo de Tipo de Negócio` da AGM

por meio dos estereótipos do *SMartyProfile*. Os estereótipos do *UML Components* devem ser removidos. Por fim, o artefato `Modelo de Tipos de Negócio SMarty` (Figura 5.12) é gerado.

Tarefa: *Identificar Interfaces de Negócio*

As interfaces de negócio são abstrações de informações que devem ser gerenciadas pelo sistema (Cheesman e Daniels, 2001). Para identificar as interfaces de negócio é necessário ter o `Modelo de Tipos de Negócio`, refiná-lo e especificar quaisquer regras de negócio adicionais com restrições. Essa tarefa é realizada pelo Arquiteto de Software.

Um dos passos mais importantes para identificar as interfaces de negócio é identificar quais são os tipos de negócio considerados núcleo (*core*). A partir disso, pode-se criar as interfaces de negócio para estes tipos de núcleo e adicioná-las ao `Modelo de Tipos de Negócio`. Tais interfaces identificadas, devem ser anotadas pelo estereótipo «`interface_type`» do *UML Components*. Por fim, deve-se refinar o `Modelo de Tipos de Negócio` para indicar as responsabilidades das interfaces de negócio.

A identificação das interfaces de negócio não é restrita a essas condições. Pode ser que existam políticas primordiais do domínio definidas por uma arquitetura de componentes

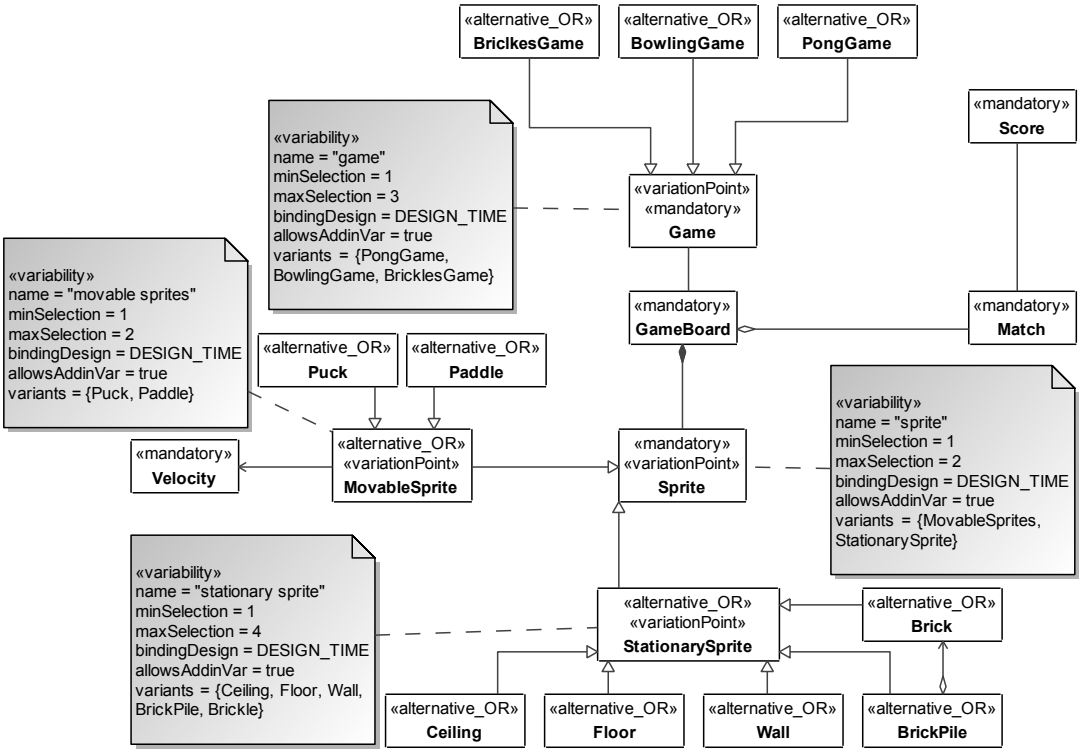

Figura 5.12: `Modelo de Tipo de Negócio SMarty` da AGM

corporativa. Tais interfaces ou restrições devem estar contidas no artefato `Recursos Existentes` e também devem ser consideradas nesta tarefa.

Cada interface de negócio deverá gerenciar as informações representadas por um tipo de negócio *core*. As interfaces identificadas devem ser nomeadas por um padrão do *UML Components*, sendo que o prefixo da nomenclatura deve conter a letra `I` de interface, e o término a sigla `Mgt` de *Management*, significando um contrato (por exemplo, "IxxxMgt") (Cheesman e Daniels, 2001).

Após identificadas as interfaces de negócio, tal artefato serve de entrada para o *SMartyProcess*.

Exemplo: Artefato *Interfaces de Negócio*

As interfaces de negócio são identificadas a partir dos elementos anotados com o estereótipo «`core`» do `Modelo de Tipos de Negócio`.

A Figura 5.13 apresenta duas interfaces de negócios identificadas. A primeira é a `IGameMgt`, identificada no elemento `Game` e anotada com o estereótipo «`interface_type`». A segunda interface é a `IGameBoardMgt`, identificada no elemento `GameBoard` e, da mesma

forma que a primeira, anotada com o estereótipo «`interface_type`». Possivelmente devem existir outras interfaces de negócio que serão descorbertas em futuras iterações.

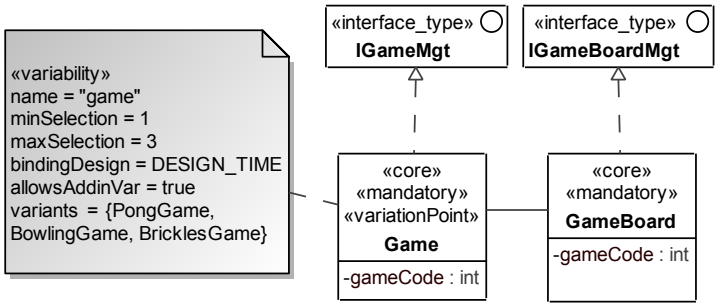

Figura 5.13: `Interfaces de Negócio` da AGM

Após descobertas as interfaces de negócio existentes, o artefato gerado `Interfaces de Negócio` é tomado como entrada para o *SMartyProcess*, em que o Arquiteto de LPS identifica, delimita e representa as variabilidades existentes, com base nas diretrizes de diagrama de classes. Por fim, o artefato `Interfaces de Negócio SMarty` é gerado. Nesta atividade, os estereótipos «`interface_type`» e «`core`» do *UML Components* são removidos e substituidos pelo estreótipo «`mandatory`» do *SMartyProfile*. A Figura 5.14 apresenta tal artefato com suas respectivas mudanças.

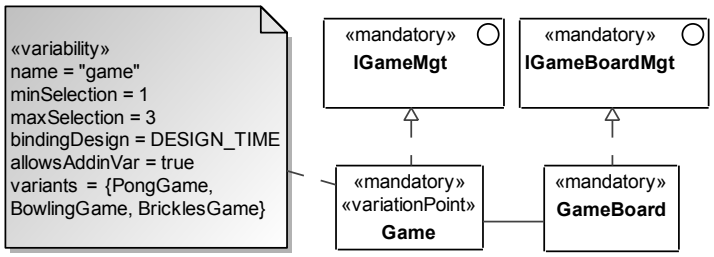

Figura 5.14: `Interfaces de Negócio SMarty` da AGM

Tarefa: *Identificar Interfaces do Sistema & Operações*

O objetivo dessa tarefa é identificar as possíveis interfaces e operações do sistema. Essa tarefa é realizada pelo Arquiteto de Software e tem como entrada o `Modelo de Casos de Uso SMarty`.

Para cada caso de uso, um tipo de diálogo e uma interface de sistema são identificados. Deve-se revisitar cada caso de uso considerando a possibilidade de existirem responsabilidades do sistema que devem ser modeladas. Se existir, tais responsabilidades devem ser representadas como operações da interface em si. Essas mudanças irão proporcionar um

conjunto inicial de interfaces e operações. Cada caso de uso com variabilidade identificada, convertido para uma interface, deve manter a variabilidade e o estereótipo. As possíveis mudanças referentes devem ser realizadas na próxima atividade, em que o *SMartyProcess* refina tais responsabilidades.

Na atividade de `Interação de Componente` serão reavaliadas se as interfaces devem assumir as responsabilidades para cada operação. Pode ser que algumas operações, identificadas incialmente nesta etapa, sejam convertidas em interfaces na próxima.

Após realizar essa tarefa, o artefato de saída *Interfaces do Sistema* é entrada para o *SMartyProcess*. Por fim, é gerado o artefato `Interfaces do Sistema SMarty`.

Exemplo: Artefato *Interfaces do Sistema*

A partir do `Modelo de Casos de Uso SMarty` foram identificadas as interfaces do sistema. Cada caso de uso se torna uma interface e um tipo de dado, representado por uma classe que depende da interface. Cada interface deve ser anotada com o estereótipo ≪`interface_type`≫ do *UML Components*, além de manter o estereótipo de variabilidade contido no caso de uso. Possíveis operações das interfaces do sistema também podem ser identificadas nessa etapa.

A Figura 5.15 apresenta todas as interfaces identificadas a partir dos casos de uso do `Modelo de Casos de Uso SMarty` para a LPS AGM.

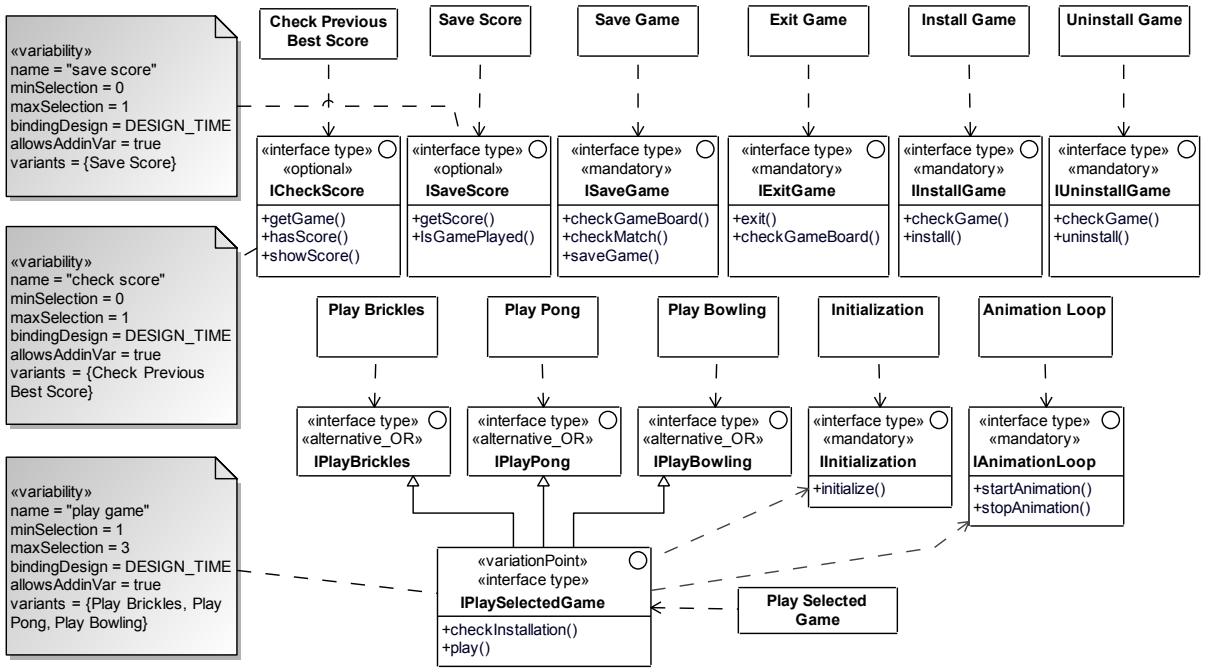

Figura 5.15: `Interfaces do Sistema` da AGM

Após identificadas as interfaces, o artefato `Interfaces do Sistema` é gerado. Tal artefato é tomado de entrada pelo *SMartyProcess*. Por fim, o artefato `Interfaces do Sistema SMarty` é gerado. A Figura 5.16 apresenta tal artefato, com as respectivas variabilidades identificadas e representadas.

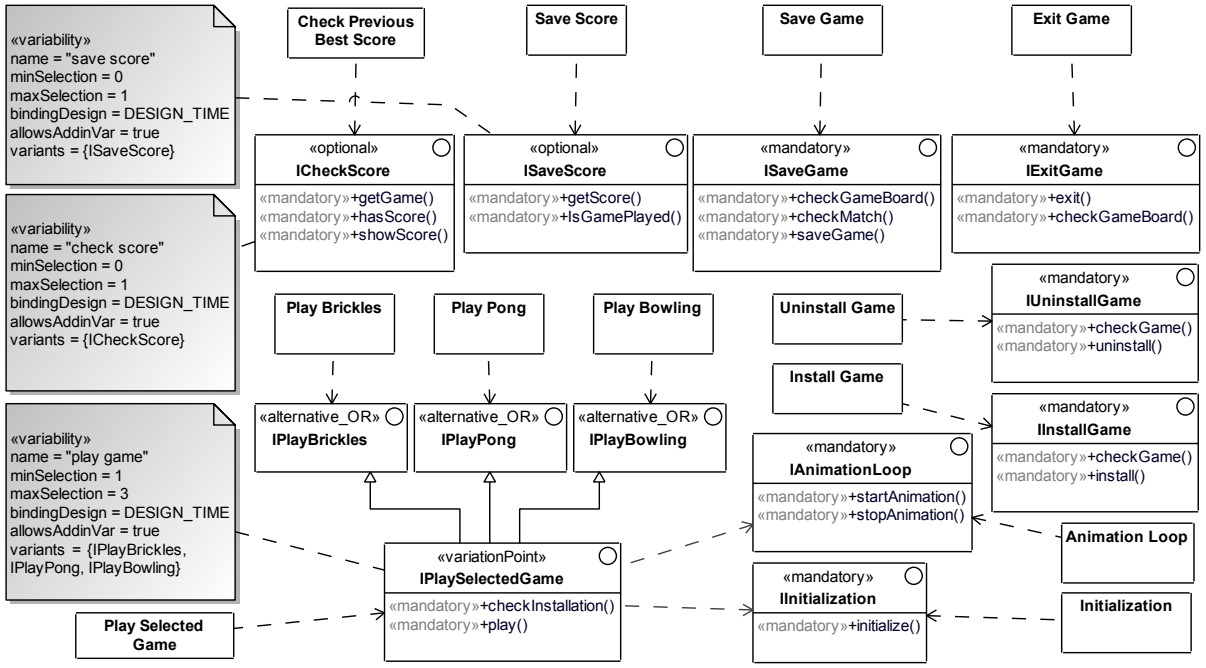

Figura 5.16: `Interfaces do Sistema SMarty` da AGM

Tarefa: *Criar Especificação Inicial de Componentes & Arquitetura*

Após realizar a tarefa `Identificar Interfaces do Sistema & Operações` e a tarefa `Identificar Interfaces de Negócio`, os arteatos gerados por esssas são tomados de entrada pela tarefa `Criar Especificação Inicial dos Componentes & Arquitetura`. O Arquiteto de Software é responsável por tal tarefa. Nela, um conjunto inicial das especificações dos componentes é criado, a fim de formar uma ideia de como eles podem interagir. Os componentes podem ser construídos ou adquiridos de terceiros, assim, deve-se escolher os componentes de forma que faça sentido à funcionalidade dessa unidade.

De acordo com Cheesman e Daniels (2001) algumas entradas possíveis para esta tarefa são:

- as interfaces de negócio e/ou de sistema;
- especificações de componentes existentes que visam a reutilização;

- uma possível arquitetura de especificação de componentes existente na qual deva-se adaptar; e
- a escolha de padrões de arquitetura de especificação de componentes.

As interações entre as interfaces são complexas, frequentes, ou envolvem grandes quantidades de dados. Em alguns casos terá que ser criada uma especificação de componentes separada para cada especificação de interfaces que forem identificadas, de acordo com Cheesman e Daniels (2001).

Duas especificações iniciais são realizadas nesta tarefa: (i) Especificação de Componentes de Sistema; e (ii) Especificação de Componentes de Negócio.

- **Especificação de Componentes de Sistema**: para esta especificação, as interfaces do sistema são derivadas a partir dos casos de uso. Pode-se optar por tê-las em uma mesma especificação de componente, mas deve-se pensar na possibilidade de reúso. Assim, se necessário for, tal interface pode apoiar outra especificação de componente. As interações serão melhor definidas na próxima etapa do *workflow*; e
- **Especificação de Componentes de Negócios**: para as interfaces de negócio, o ponto de partida é uma especificação de componente por interface. A inteface tem como prefixo a letra "I", e como terminação "Mgt". Os componentes de negócio possuem somente terminação "Mgr"de *Manager*. Estas especificações serão refinadas nas próximas etapas.

A partir dessas tarefas, tem-se um conjunto inicial de especificações de componentes, incluindo suas interfaces e suas dependências. Após removidas quaisquer redundâncias nas especificações, as interfaces requeridas podem ser conectadas às interfaces fornecidas. Dessa forma, obtem-se o artefato da `Especificação de Componentes e Arquitetura` inicial. Possivelmente, as interfaces já devem possuir alguma representação de variabilidades, vinda dos casos de uso. Assim, o artefato gerado é obtido como entrada para o *SMartyProcess*, que refinará a arquitetura de componentes, interfaces e operações, com base nas diretrizes para diagrama de componentes UML do *SMartyProcess*, de acordo com a Seção 3.4. Após a identificação, o Arquiteto de LPS delimita e representa as variabilidades, por meio do *SMartyProfile*, e finalmente é gerado o artefato `Especificação de Componentes & Arquitetura SMarty`.

Exemplo: Artefato *Especificação de Componente & Arquitetura*

Para a criação deste artefato tem-se como entrada as `Interfaces do Sistema` e `Interfaces de Negócio`, além de ativos e padrões de arquitetura, se estes existirem.

Duas especificações serão criadas: (i) a Especificação de Componentes do Sistema; e (ii) a Especificação de Componentes de Negócio.

Na Especificação de Componentes de Sistema, todas as interfaces do sistemas criadas a partir dos casos de uso são atribuídas a um componente, que no nosso caso chamamos de `ArcadeGameMaker`. Tal componente foi anotado com o estereótipo «`comp_spec`» do *UML Components*.

Foi percebido que duas interfaces provenientes dos casos de uso, `Inicialization` e `AnimationLoop`, não tinham relacionamentos diretos com os atores, e que elas dependiam dos jogos para existirem. Assim, tais interfaces foram atribuídas como sendo interfaces de negócio e seus tipos de dados convertidos em componentes de negócio.

Na Especificação de Componentes de Negócio estão contidas as interfaces de negócio, juntamente com as classes do tipo «`core`», mantidas anteriormente no artefato de `Interfaces de Negócio`. Tais classes são convertidas em componentes de negócio, e são atribuídas à nomenclatura com terminação "Mgr"de *Manager*. Todos os estereótipos provenientes dos artefatos de entrada foram mantidos, e serão refinados na próxima iteração do *SMartyProcess*.

A Figura 5.17 apresenta a arquitetura inicial identificada a partir dessas especificações de componentes.

Após gerar o artefato `Especificação de Componente & Arquitetura` inicial, tal artefato é tomado como entrada para o *SMartyProcess*, onde o Arquiteto de LPS irá identificar as variabilidades, delimitar e representar, de acordo com as diretrizes para diagrama de componentes. Os estereótipos «`core`», «`comp_spec`» e «`interface_type`» poderão ser removidos pelo *Arquiteto de LPS* nesta etapa, pois não será mais utilizado nas próximas interações. A Figura 5.18 apresenta o artefato gerado `Especificação de Componentes & Arquitetura SMarty`.

5.4.2 Atividade: *Interação de Componente*

Esta atividade do *workflow* de especificação está focada em como os componentes interagem entre si. O Arquiteto de Software é responsável por realizar as tarefas dessa atividade. De acordo com as `Interfaces do Sistema SMarty` e as `Interfaces de Negócio SMarty`, serão definidas as operações de negócio, e, ambas interfaces e operações identificadas serão refinadas, gerando assim o artefato `Interfaces SMarty`. Ainda, tais interfaces servirão de apoio para refinar a especificação de componentes e arquitetura. A seguir são apresentadas as tarefas da atividade `Interação de Componente`. A Figura 5.19 apresenta tais tarefas, os papéis envolvidos e os artefatos de entrada e saída.

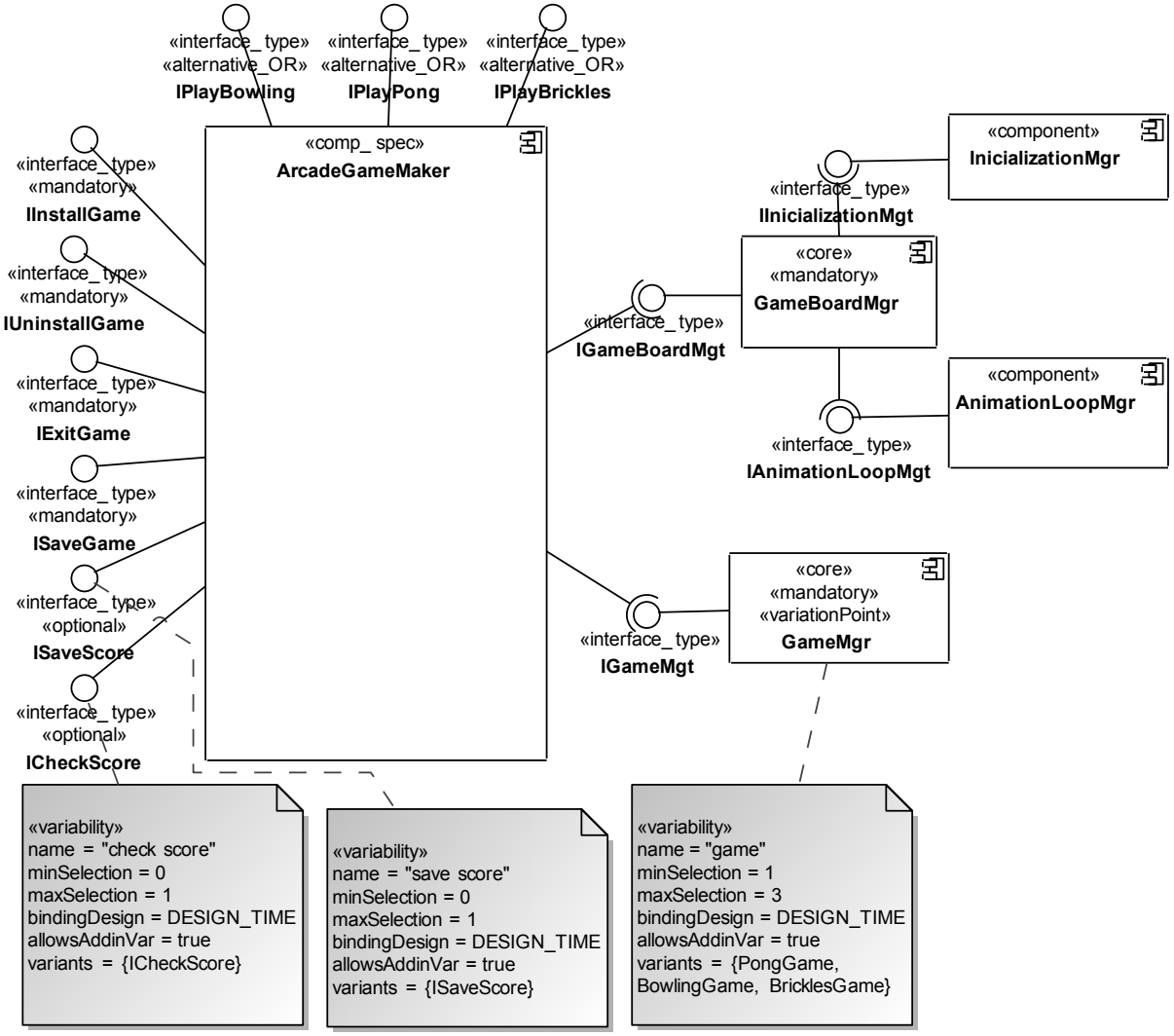

Figura 5.17: `Especificação de Componentes & Arquitetura` da AGM

Tarefa: *Descobrir Operações de Negócio*

O objetivo dessa tarefa é descobrir as operações contidas nas interfaces de negócio. Tal tarefa é realizada pelo Arquiteto de Software. Para apoiar esta tarefa, ela recebe como entrada os artefatos `Interfaces do Sistema SMarty` e `Interfaces de Negócio SMarty`.

Para descobrir as operações de interfaces de negócio é importante rever as interações que ocorrem nas operações das interfaces do sistema. Assim, um ou mais diagramas de comunicação podem ser desenhados, se necessário, rastreando as possíveis restrições existentes no fluxo de execução resultante da invocação de tais operações. Cada diagrama de comunicação poderá mostrar uma ou mais interações, em que cada interação mostra

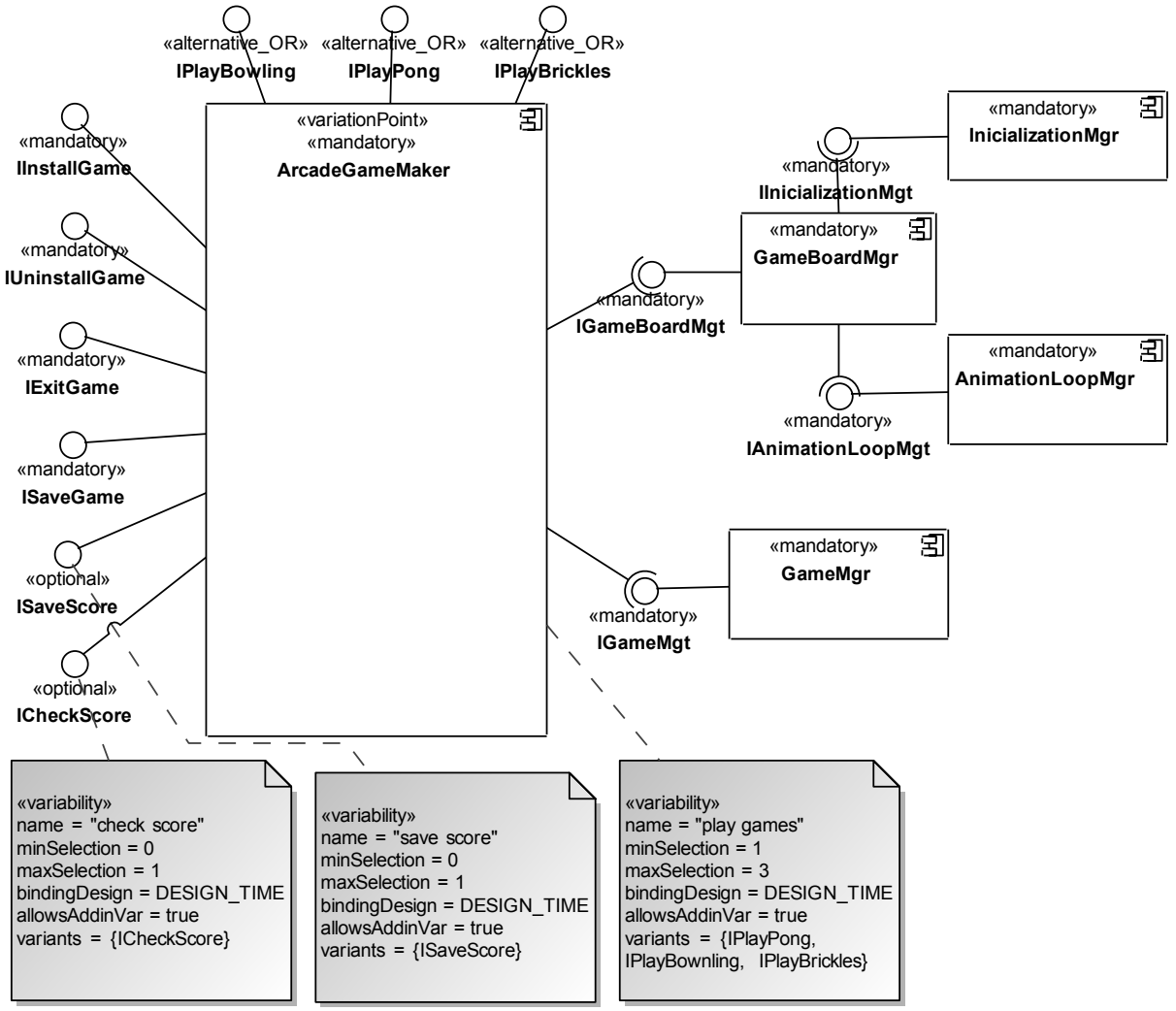

Figura 5.18: `Especificação de Componentes & Arquitetura SMarty` da AGM

um possível fluxo de execução. Assim, se houver vários fluxos alternativos importantes poderá ser necessário desenhar várias interações. Por meio das interações representadas nos diagramas de comunicação tornará mais claro quais operações serão necessárias de acordo com os eventos a serem realizados e objetos a serem gerenciados (Cheesman e Daniels, 2001).

Juntamente com a tarefa `Descobrir Operações de Negócio`, as tarefas de `Refinar Interfaces & Operações` e `Refinar Especificação de Componentes & Arquitetura` estão envolvidas em um fluxo evolutivo. O artefato gerado na tarefa Descobrir Operações de Negócio é entrada para a tarefa `Refinar Interfaces & Operações`, e o artefato gerado nesta tarefa, é entrada para a tarefa `Refinar Especificação de Componentes & Arquitetura`. Esse fluxo de tarefas pode ser observado na Figura 5.19.

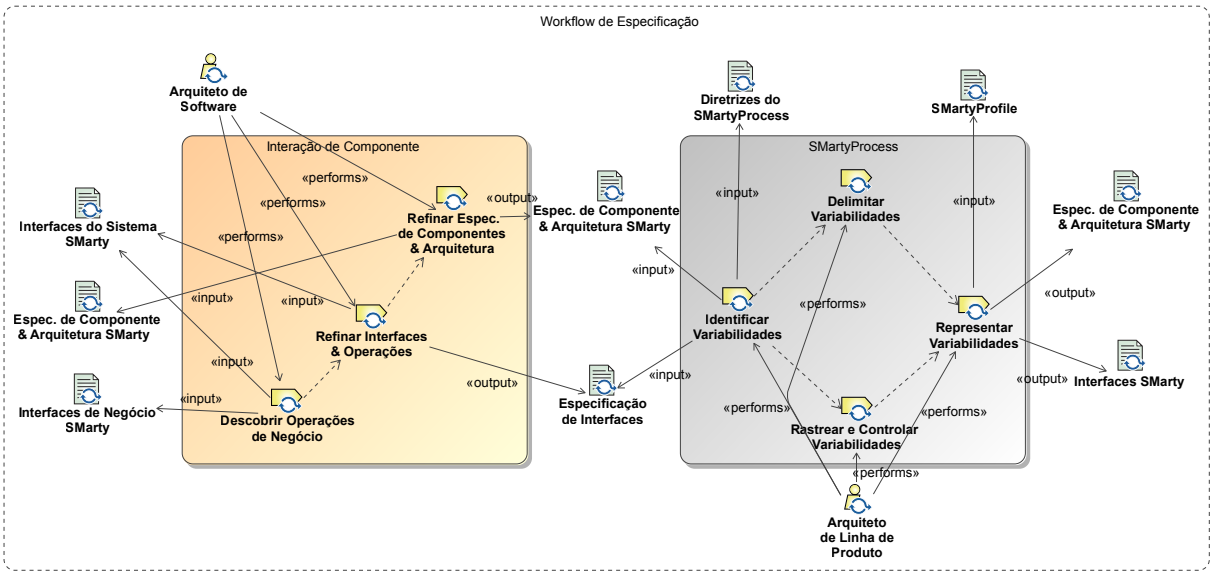

Figura 5.19: *Workflow* de Especificação: Interação de Componente

Tarefa: *Refinar Interfaces & Operações*

Nesta tarefa as responsabilidades atribuídas às interfaces devem ser analisadas, de modo que, se for necessário separar uma interface em duas com o objetivo de proporcionar o reúso ou dividir funcionalidades, é aqui que isso deve acontecer. O Arquiteto de Software é responsável por essa tarefa.

Por meio de diagramas de comunicação as interações das interfaces do sistema com as interfaces de negócio podem tornar mais explícitas as operações que estão faltando como também proporcionar uma visão do que pode ser melhorado, a fim de evitar duplicidade de responsabilidades.

Após refinar as interfaces e operações, o artefato resultante, `Especificação de Interfaces`, é tomado como entrada pelo *SMartyProcess*, que por meio das diretrizes para diagrama de componentes UML, como consta na Seção 3.4, identifica as variabilidades. Após a identificação, o Arquiteto de LPS delimita e representa as variabilidades encontradas, por meio do *SMartyProfile*. Por fim, é gerado o artefato `Interfaces SMarty`.

Exemplo: Artefato *Especificação de Interfaces*

Todas as interfaces existentes do artefato `Interfaces do Sistema SMarty` e todas as interfaces do artefato `Interfaces de Negócio SMarty`, com as possíveis operações identificadas na tarefa `Descobrir Operações de Negócio`, são abstraídas em um único artefato, com o objetivo de gerar uma especificação de todas as interfaces.

Tal especificação é útil para revisar as interfaces e operações, e ver se é necessário dividí-las a fim de proporcionar o reúso. A Figura 5.20 apresenta o artefato `Especificação de Interfaces` obtido neste exemplo.

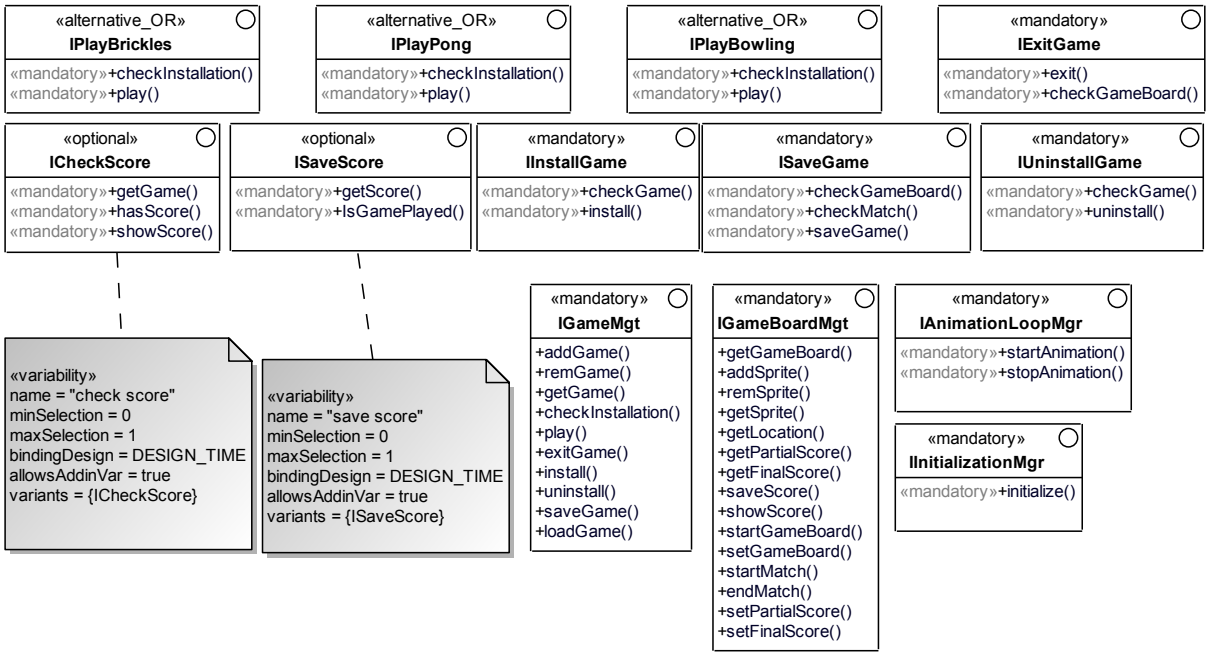

Figura 5.20: `Especificação de Interfaces` da AGM

Tal artefato é tomado como entrada pelo *SMartyProcess*. Assim, o artefato `Interfaces SMarty` é gerado. A Figura 5.21 é

Tarefa: *Refinar Especificação de Componentes & Arquitetura*

Essa tarefa recebe o artefato `Especificação de Componentes & Arquitetura SMarty` e `Especificação de Interfaces`, resultante da tarefa `Refinar Interfaces & Operações` que auxilia o refinamento da arquitetura e reorganização dos componentes e interfaces. O Arquiteto de Software é responsável por tal tarefa.

No momento em que as interfaces e operações vão sendo descobertas e refinadas, a especificação dos componentes e da arquitetura sofre as mesmas mudanças. Nessa tarefa, toda a especificação de arquitetura e componentes é refinada, visando proporcionar o reúso. Para isso, os componentes são reavaliados, de acordo com o artefato gerado pela tarefa anterior.

Após este refinamento, o artefato `Especificação de Componentes & Arquitetura SMarty` é tomado de entrada para o *SMartyProcess*. Por fim, é gerado o artefato `Especificação de Componentes & Arquitetura SMarty`.

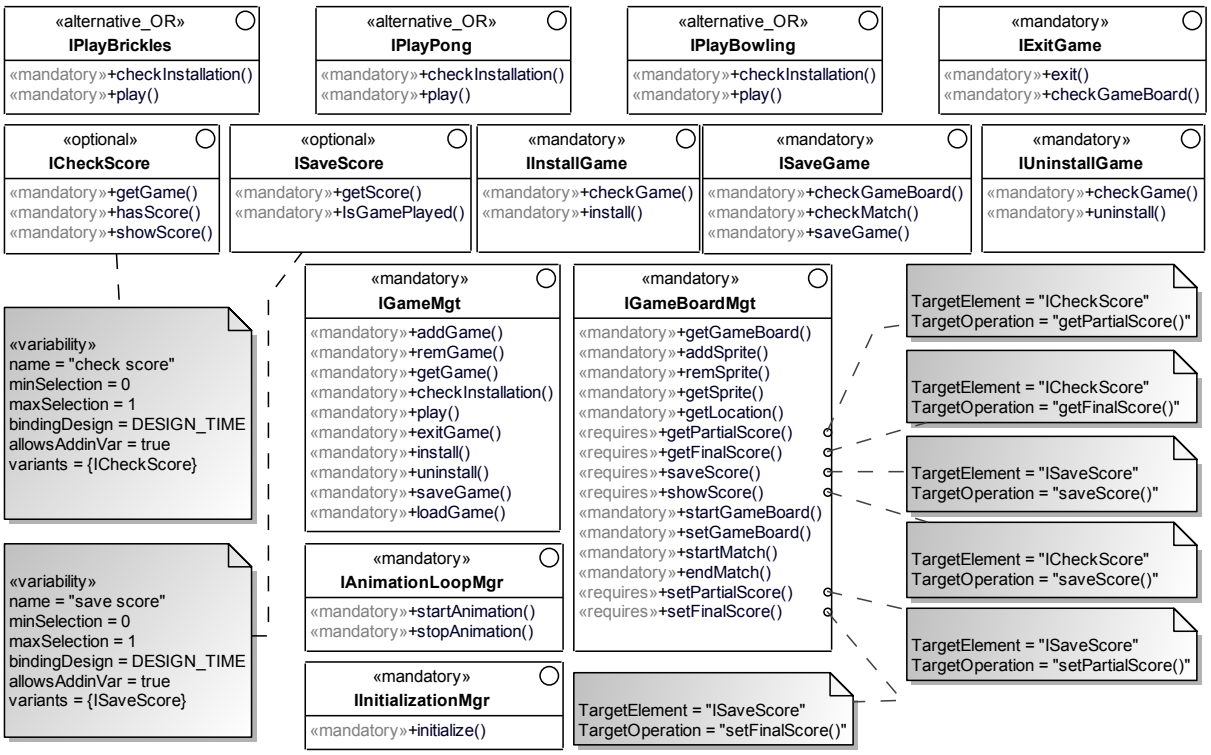

Figura 5.21: `Interfaces SMarty` da AGM

Exemplo: Artefato *Especificação de Componentes & Arquitetura SMarty*

O artefato `Especificação de Interfaces` auxilia a refinir o artefato `Especificação de Componentes & Arquitetura SMarty`. Todas as mudanças ocorridas nas interfaces serão atualizadas nesta especificação.

Para este exemplo, percebe-se que existem interfaces que dependem exclusivamente dos jogos, e interfaces que dependem mais dos resultados deles. As interfaces que visam os resultados são `ISaveScore` e `ICheckScore`, ambas com representação de variabilidades opcionais. Assim, o componente `ArcadeGameMaker` foi dividido em dois: (i) `GameBoardCtrl`, que será destinado a lidar com informações a respeito da mesa dos jogos exclusivamente; e `GameCtrl`, que irá lidar com os jogos em si. Mesmo separados, ambos necessitam de comunicação entre si. Assim, a interface `IGameBoardCtrl` foi criada.

A Figura 5.22 apresenta o artefato `Especificação de Componentes & Arquitetura SMarty` refinado, com tais modificações se comparado à Figura 5.18.

Após gerado, tal artefato é tomado como entrada pelo *SMartyProcess*. A Figura 5.23 apresenta o artefato `Especificação de Componentes & Arquitetura SMarty`. Uma porta *playGames* foi adicionada, a fim de proporcionar uma separação de interesses,

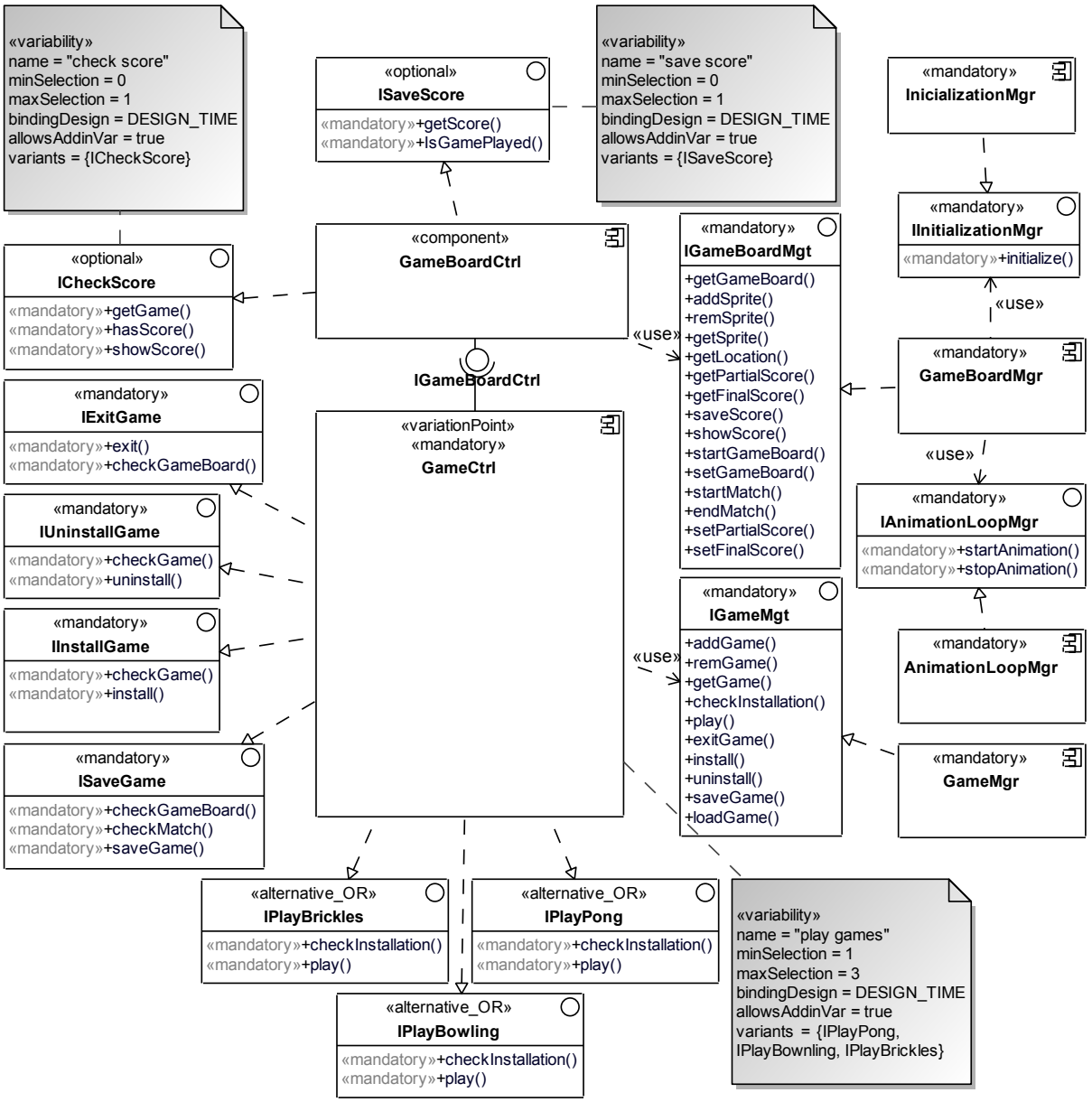

Figura 5.22: Artefato `Especificação de Componente & Arquitetura SMarty` da AGM

tornando mais explícita a ocorrência do ponto de variação, de acordo com a diretriz **CP.1** da Seção 3.4.

5.4.3 Atividade: *Especificação de Componente*

Esta atividade está focada em definir os contratos das interfaces e suas restrições. Para isso, esta atividade toma como entrada o artefato `Modelo de Tipos de Negócio SMarty`,

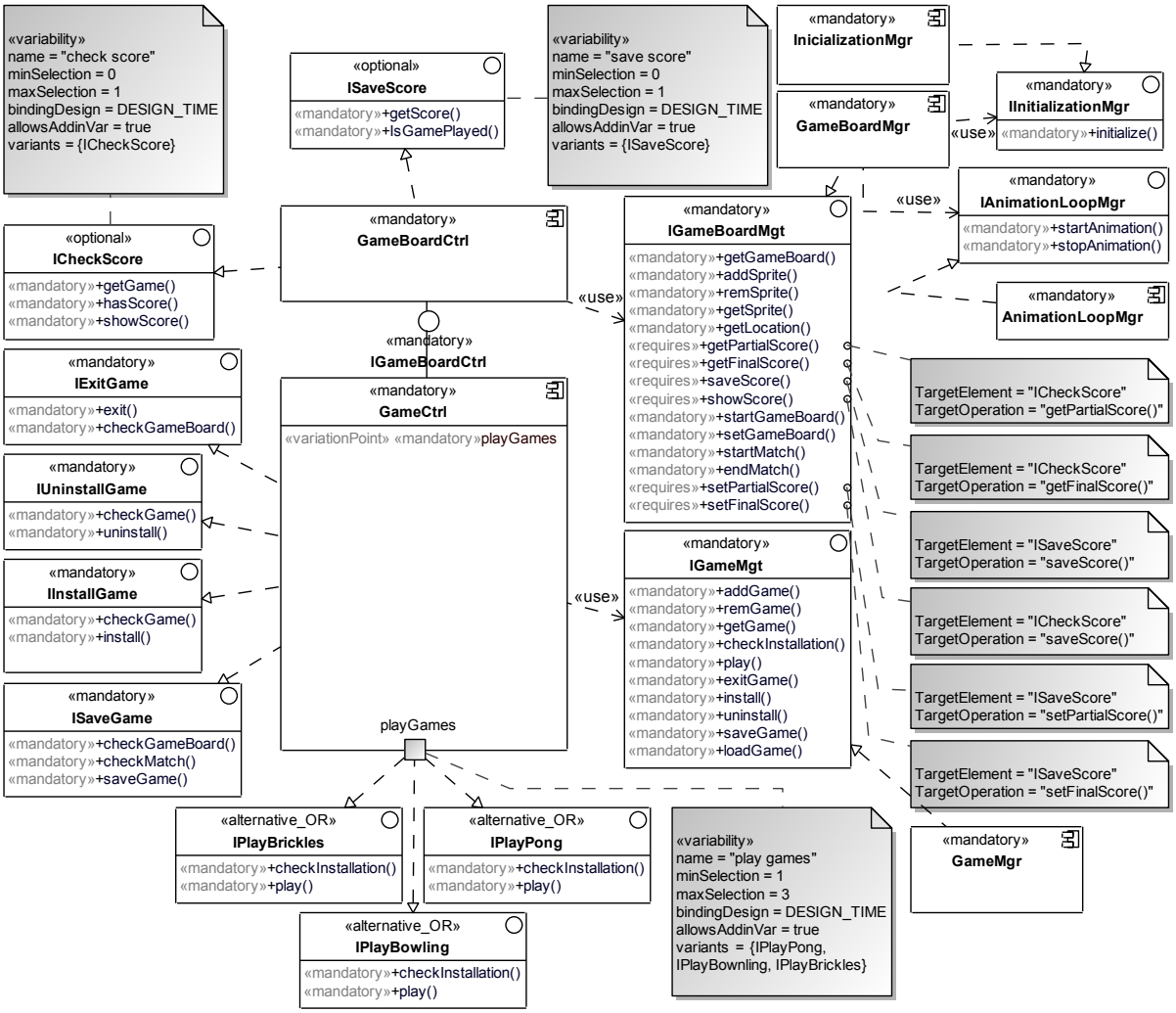

Figura 5.23: Artefato `Especificação de Componente & Arquitetura SMarty` da AGM

que foi concebido na atividade `Identificação de Componente`, que juntamente com os artefatos `Espec. de Componente & Arquitetura SMarty` e `Interfaces SMarty` auxiliam as tarefas `Definir Modelo de Informação de Interface` e `Especificar Restrições de Componente-Interface`. O Arquiteto de Software é responsável por realizar as tarefas da atividade `Especificação de Componente` enquanto o Arquiteto de LPS é responsável pelas tarefas do *SMartyProcess*. A Figura 5.24 apresenta as tarefas realizadas nesta atividade do *workflow* de Especificação, bem como os papéis envolvidos nas tarefas, e os artefatos de entrada e saída.

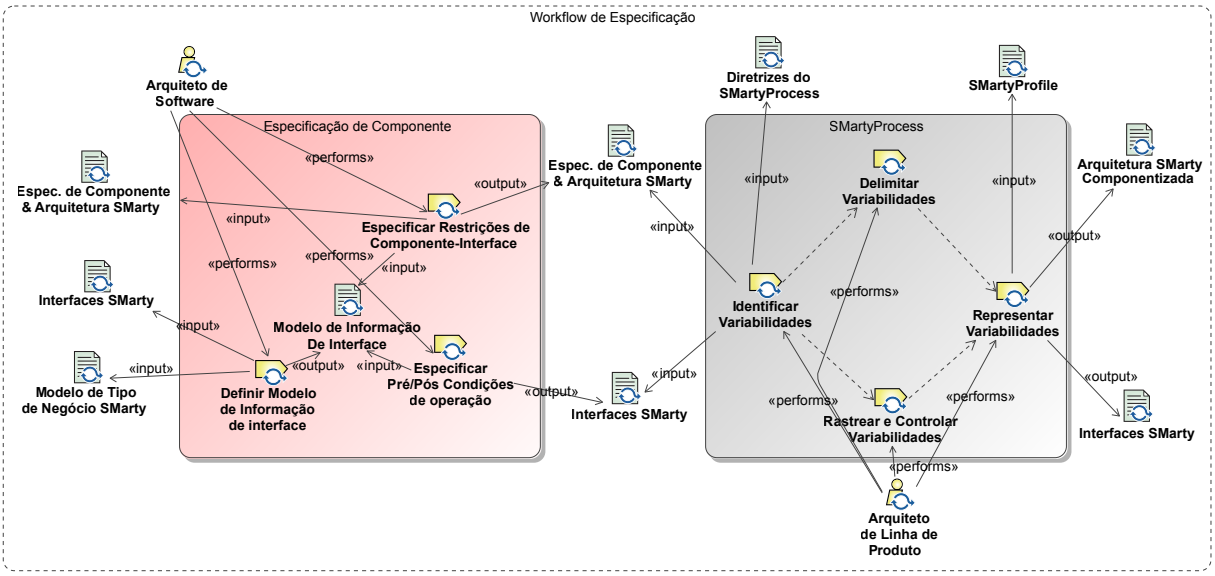

Figura 5.24: *Workflow* de Especificação: Especificação de Componente

Tarefa: *Definir Modelo de Informação de Interface*

O `Modelo de Informação de Interface` representa o estado do objeto do componente no qual a interface depende. Para cada interface deve existir um modelo de informação de interface. Este é um tipo de modelo desenhado no diagrama de especificação de interfaces, dos possíveis estados do objeto do componente para os quais as especificações de operações podem se referir. Todas as alterações ao estado do objeto do componente causadas por uma determinada operação podem ser descritas em termos dessa definição do modelo de informação (Cheesman e Daniels, 2001).

Esse `Modelo de Informação de Interface` pode ser derivado de uma forma simples a partir do `Modelo de Tipos de Negócio`. Quando um tipo de propriedade de uma interface refere-se a um tipo de propriedade de outra, o tipo referenciado aparece no `Modelo de Informação de Interface` de ambas as interfaces.

Para criar o `Modelo de Informação de Interface` para uma interface de negócio o arquiteto de LPS pode derivar o `Modelo de Tipos de Negócio`, e em seguida, eliminar os tipos, relacionamentos e atributos que não são necessários. Se preferir, também pode inserir dependências de rastreabilidade («`trace`») entre tipos de modelo de informações de interface e tipos de modelo de tipos de negócio, se desejar, e se a sua ferramenta apoia a manutenção desses (Cheesman e Daniels, 2001).

O modelo de informação de interface deve conter apenas o suficiente para permitir que as operações da interface possam ser especificadas. É possível construir o modelo de informações da interface de forma incremental à medida que cria as especificações

de operação, acrescentando os tipos, atributos e relacionamentos, conforme necessário (Cheesman e Daniels, 2001).

O artefato gerado por esta tarefa, serve de entrada para especificar as pré e pós condições de operações, como também para auxiliar a especificação de restrições de componentes e/ou interfaces.

Exemplo: Artefato *Modelo de Informação de Interface*

O artefato `Modelo de Informação de Interface` contém as informações referentes aos objetos dos componentes pelos quais tais interfaces dependem. Tal artefato auxilia a verificar se as operações estão corretas em relação aos objetos dos componentes relacionados. Todas as mudanças do estado do objeto do componente causadas por uma operação podem ser descritas em termos dessa definição de modelo de informação.

Para se criar o `Modelo de Informação de Interface` deve-se basear nas interfaces existentes, por meio do artefato `Interfaces SMarty` e do artefato `Modelo de Tipos de Negócio SMarty`. Esse modelo deve apresentar a interface em si e seu relacionamento, e um modelo de objeto extraído por meio de uma operação.

A Figura 5.25 apresenta um excerto das interfaces, juntamente com os objetos dos componentes (anotados com o estereótipo «`dataType`») em que as interfaces dependem, por meio de suas operações. Este artefato gerado auxilia a tarefa `Especificar Restrições de Componente-Interface` e `Especificar Pré/Pós Condições de Operação`.

Como exemplo da Figura 5.25, a interface `IPlayBrickles` foi extraída do artefato `Interfaces do Sistema SMarty` (Figura 5.15), e as operações em que ela depende da interface `IPlaySelectedGame` lhe foram atribuídas. O caso de uso que se tornou um tipo de dado no artefato `Interfaces do Sistema`, é representado na Figura 5.25 como o estereótipo «`dataType`» do *UML Components*. Assim, uma possível chamada das operações `checkInstallation()` ou `play()` na interface `IPlayBrickles` irão invocar o objeto `BriclkesGame`.

É importante deixar claro que este modelo é apenas em nível de especificação, representando os estados de um objeto que um componente pode ter. Assim, ele não descreve como tais estados devem ser implementados ou persistidos (Cheesman e Daniels, 2001).

Tarefa: *Especificar Pré/Pós Condições de Operação*

Após a tarefa de definir as informações de interfaces, o artefato gerado por ela auxilia a especificar as pré e pós condições das operações.

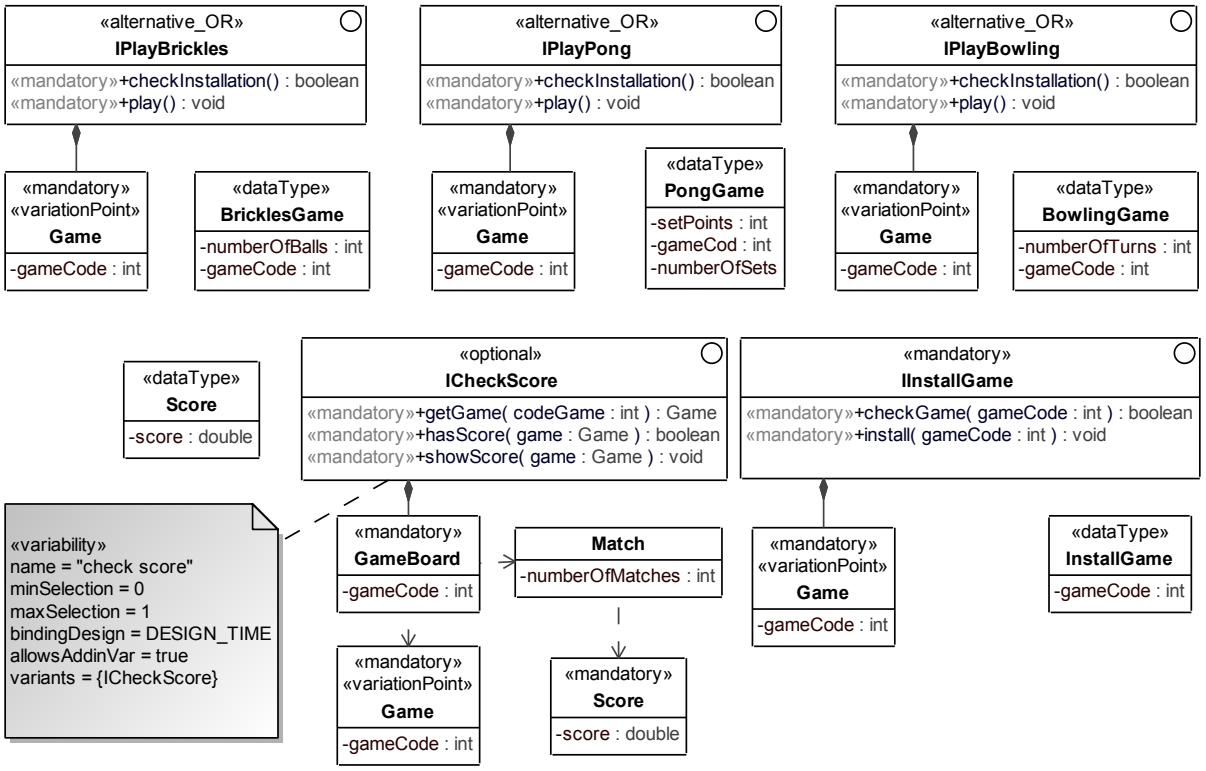

Figura 5.25: `Modelo de Informação de Interface` da AGM

A tarefa de especificar pré e pós condições de operação pode ser entendida com as pequenas letras de um contrato com um cliente. Especifica em detalhes o que as operações irão fazer, sempre como um par (pré e pós condições). A pós-condição especifica qual será o efeito da operação, desde que a pré-condição seja verdadeira. Se a pré-condição for falsa, então nenhum resultado deve ser retornado.

Em UML essas condições contratuais podem ser especificadas com precisão usando *Object Constraint Language* (OCL) (Cheesman e Daniels, 2001). OCL é uma linguagem declarativa que permite construir expressões lógicas, tais expressões garantem uma interpretação inequívoca, diferente de reivindicações feitas por linguagem natural (OMG, 2015a).

SMarty ainda não possui parâmetros definidos para tratamento de restrições OCL. Por ser assim, estas restrições não serão tratadas neste momento, neste trabalho. Em contrapartida, o *SMarty* possui estereótipos que suportam restrições entre variantes. Tais restrições podem ser aplicadas nas atividades do *SMartyProcess*.

Como resultado final, este artefato auxilia na evolução do artefato `Interfaces SMarty`. Tal artefato é tomado como entrada pelo *SMartyProcess*. Por fim, o artefato `Interfaces SMarty` é atualizado.

Exemplo: Artefato *Interfaces SMarty*

O artefato `Especificações de Interfaces SMarty` contém todas as interfaces identificadas e as retrições existentes entre as interfaces e/ou operações.

O artefato `Especificações de Interfaces SMarty` é tomado de entrada pelo *SMarty-Process*, onde o Arquiteto de LPS identifica, delimita e representa as variabilidades, de acordo com as diretrizes para diagrama de componentes. A Figura 5.26 apresenta tal artefato com as suas operações, restrições e definições de tipos de dados atualizadas em relação com a Figura 5.21.

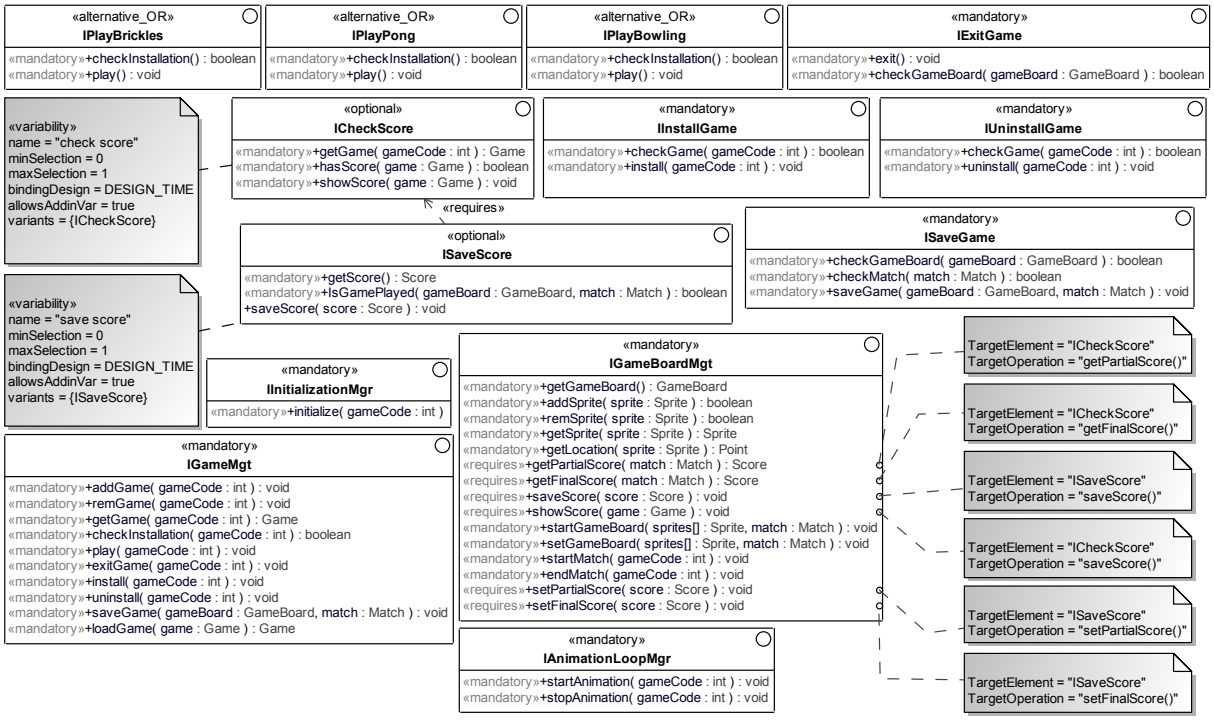

Figura 5.26: `Interfaces SMarty` da AGM

Tarefa: *Especificar Restrições de Componente-Interface*

As especificações de interface auxiliam a lidar com o uso de contrato entre um objeto componente e seus clientes. Nesta tarefa, deve-se considerar as informações adicionais da especificação no qual o implementador e o montador do componente precisam estar cientes, principalmente sobre as dependências de um componente em relação a outras interfaces, que precisam estar bem definidas (Cheesman e Daniels, 2001).

Para cada especificação de componente é necessário dizer quais interfaces suas realizações internas devem suportar. É necessário decompor o diagrama da arquitetura em

partes específicas para cada especificação de componente, a fim de que um pacote de especificação independente possa ser entregue ao implementador (Cheesman e Daniels, 2001).

Assim, essa tarefa gera o artefato `Espec. de Componente & Arquitetura SMarty`, que será refinado posteriormente por meio do *SMartyProcess*, onde o Arquiteto de LPS identifica, delimita e representa as variabilidades, por meio do *SMartyProfile*. Por fim, o artefato `Arquitetura SMarty Componentizada` é gerado.

A Figura 5.27 apresenta todas as atividades e tarefas que ocorrem no *workflow* de Especificação, bem como seus artefatos de entrada e saída. A disposição dos elementos foi alterada para uma melhor visualização das suas atividades.

Exemplo: Artefato *Espec. de Componente & Arquitetura*

Este artefato deve conter todas as informações a respeito da arquitetura. Ele recebe como entrada o artefato `Especificação de Componentes & Arquitetura SMarty` já refinado, e o `Modelo de Informações de Interfaces`.

A arquitetura é separada em camadas de acordo com o *UML Components*: (i) `System Layer`, que é a camada do sistema, onde os componentes e interfaces do sistema são adicionados; e (ii) `Business Layer`, que é a camada de negócios, onde os componentes e interfaces de negócio são adicionados (Cheesman e Daniels, 2001).

A Figura 5.28 apresenta o artefato `Arquitetura SMarty Componentizada`, com as suas operações de interfaces explícitas.

5.5 Considerações Finais

Existe uma lacuna na literatura quando se trata de variabilidade em arquiteturas de software. Tal afirmação foi percebida por meio dos estudos levantados no mapeamento sistemático (Apêndice A), em que vários estudos apontam que ainda não existe um padrão ou forma sistemática definida para preencher tal lacuna. Os estudos levantados criam hipóteses e até modelos para suportar tal dificuldade.

Diante disso, a proposta do *SMartyComponents* é fornecer um processo que apoia o desenvolvimento baseado em componentes juntamente com a representação de variabilidades em níveis de componentes e arquitetura com o intuito de explorar as vantagens da técnica de reúso de forma não oportunística.

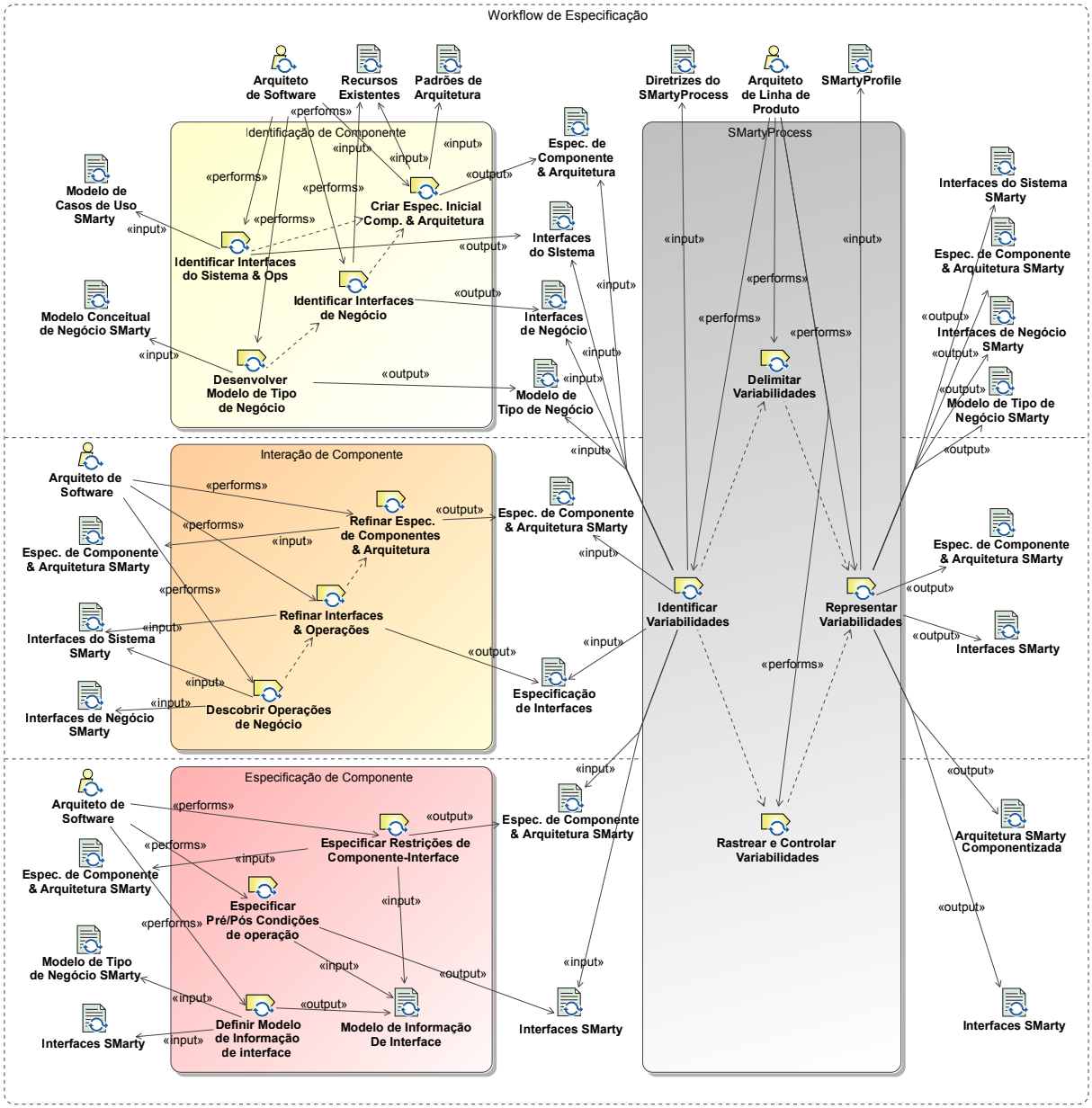

Figura 5.27: *Workflow* de Especificação do SMartyComponents com todas as suas Atividades e Respectivas Tarefas.

Ensaios sobre a integração do *UML Components* com a abordagem *SMarty* já tinham sido realizados em estudos anteriores e apresentavam resultados promissores. Tais resultados levaram à proposta do processo *SMartyComponents*.

Algumas melhorias precisam ser implementadas, como, por exemplo, o caso das tarefas `Especificar Pré/Pós Condições de Operação` e `Especificar Restrições de Componente-Interface`, pois elas exigem restrições em OCL. Por hora, *SMarty* não tem

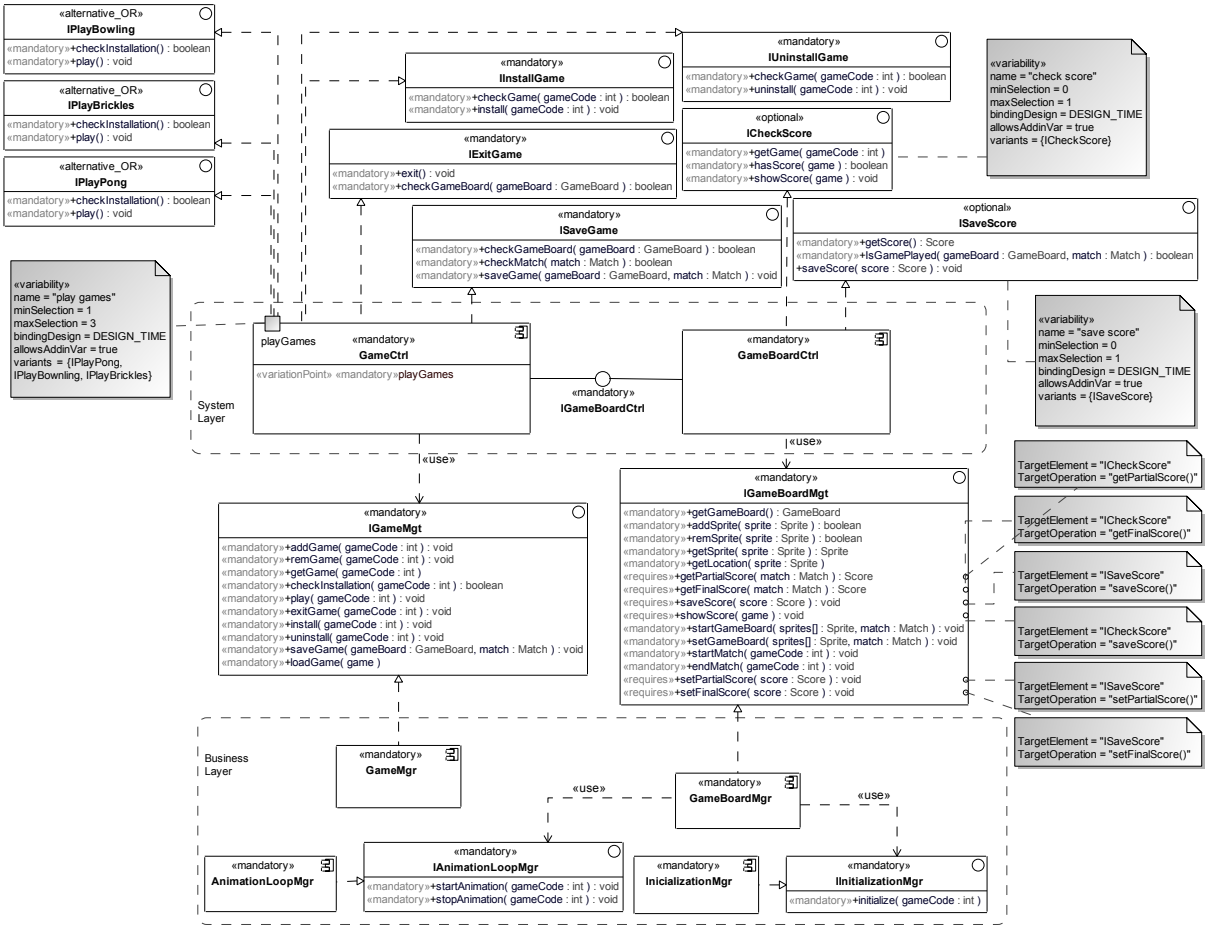

Figura 5.28: `Arquitetura SMarty Componentizada` da AGM.

suporte a OCL, mas em uma futura evolução do *SMarty* essa característica pode ser explorada.

O capítulo a seguir apresenta um estudo empírico qualitativo com o objetivo de analisar o processo proposto sob o ponto de vista de especialistas em DBC e LPS.

6

Estudo Empírico Qualitativo do *SMartyComponents*

"Todos querem o perfume das flores, mas poucos sujam as suas mãos para cultivá-las."

Augusto Cury (1958), Médico

6.1 Considerações Iniciais

De forma geral duas abordagens de investigação empírica se destacam: quantitativa e qualitativa. O objetivo da pesquisa quantitativa é obter uma relação númerica entre n variáveis ou alternativas em análise (Juristo e Moreno, 2010). Já a pesquisa qualitativa tem por objetivo produzir resultados não alcançados com procedimentos estatísticos ou de outros meios de quantificação (Corbin e Strauss, 2008).

Estudos qualitativos têm sido muito bem aceitos na literatura de Engenharia de Software conforme Seaman (1999). Um exemplo recente desse tipo de estudo é o de Geraldi *et al.* (2015) aplicado à técnicas de inspeção baseadas em *Checklist*, e de Steinmacher *et al.* (2014) sobre a entrada de novatos em projetos de software livre.

Essa seção apresenta uma avaliação empírica qualitativa, baseada em procedimentos da Teoria Fundamentada nos Dados (*Grounded Theory*), como Codificação (*Coding*)

proposta por Corbin e Strauss (2008), com o objetivo de analisar a proposta do processo *SMartyComponents*, bem como encontrar pontos fortes e fracos para melhorar tal proposta com base em especialistas em DBC e LPS.

A Figura 6.1 ilustra a metodologia de pesquisa utilizada neste estudo, apresentando os processos seguidos.

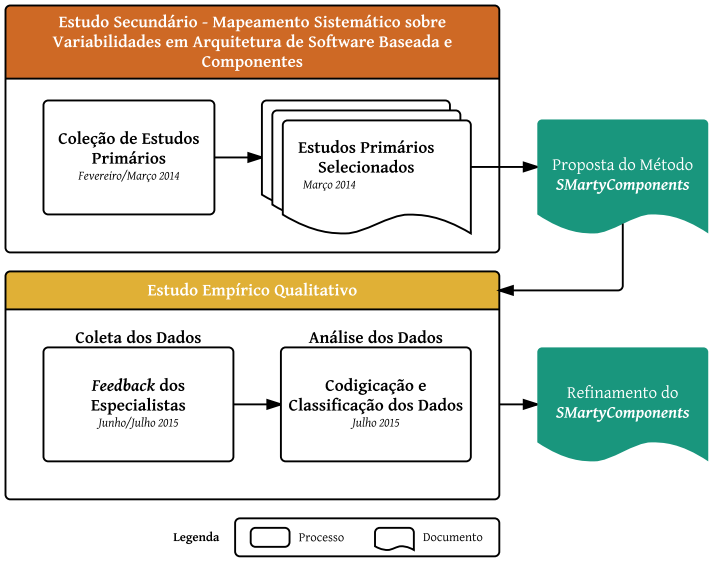

Figura 6.1: Metodologia de Pesquisa do Estudo Empírico Qualitativo

Existem três elementos básicos que são fundamentais na pesquisa qualitativa (Corbin e Strauss, 2008):

- **Dados**: que podem ser obtidos de diversas formas, como entrevistas, observações, documentos, registros e filmes. No caso deste estudo, por meio de formulários eletrônicos;
- **Procedimentos**: que são utilizados para a interpretação e organização dos dados; e
- **Relatórios**: que podem ser escritos e verbais, como artigos publicados em periódicos científicos, conferências ou em livros.

A **Codificação** (*Coding*) faz parte dos procedimentos e tem por objetivo conceitualizar e reduzir os dados, e elaborar categorias em termos de suas propriedades e dimensões, relacionando-os por meio de uma série de declarações preposicionais (Broberg, 2011).

Neste estudo, foram utilizados dois tipos de codificação: (i) **Codificação Aberta** (*Open Coding*); e (ii) **Codificação Axial** (*Axial Coding*).

A **Codificação Aberta** é a primeira parte do processo da análise dos dados. É realizada manualmente de acordo com a leitura dos dados recuperados e submetendo-os ao processo de codificação, podendo classificar palavras, linhas ou sentenças as quais expressam a essência do conteúdo recuperado.

A **Codificação Axial** é a segunda parte do processo da análise dos dados, em que, depois de realizada a codificação aberta, os códigos gerados são reagrupados em níveis de maior abstração. Assim, novas combinações são estabelecidas formando as subcategorias.

6.2 Definição do Estudo Empírico

O propósito principal deste estudo foi **analisar a proposta do processo *SMarty-Components* para especificar ALPSs componentizadas**. Baseado no modelo *Goal Question Metric* (GQM) (Basili *et al.*, 1994; van Solingen *et al.*, 2002), este estudo tem por objetivo:

> **Analisar** *a proposta do SMartyComponents*
>
> **Com o propósito de** *estabelecer a sua compreensibilidade*
>
> **Referente à** *capacidade de especificar ALPSs componentizadas*
>
> **Do ponto de vista de** *arquitetos de LPS*
>
> **No contexto de** *especialistas, mestres, doutores e acadêmicos de pós-graduação stricto sensu das seguintes instituições: Universidade Estadual de Maringá (UEM), Pontifícia Universidade Católica do Rio Grande do Sul (PUC-RS), Universidade Federal do Rio Grande do Sul (UFRGS), Instituto Federal do Rio Grande do Norte (IFRN) e Universidade Federal do Amazonas (UFAM).*

6.3 Planejamento do Estudo

A seguir são apresentados os itens que compõem o planejamento deste estudo:

- **Projeto Piloto**: com o intuito de avaliar a instrumentação proposta para o experimento, um projeto piloto foi realizado no mês de março de 2015, com três participantes, sendo eles doutorandos nas áreas de arquitetura de software, LPS e processo de software. Os dados obtidos dos participantes foram descartados, porém as considerações em relação a erros e melhorias na instrumentação proposta foram levadas em consideração;

- **Treinamento**: um treinamento foi realizado com os participantes, com o objetivo de nivelar o conhecimento de cada participante em relação à fundamentação teórica necessária para a realização do estudo, em relação aos conceitos do processo *UML Components*, conceitos de *SMarty* e conceitos de *SPEM*;

- **Especialistas**: para tal estudo os especialistas selecionados foram alunos e professores da área de engenharia de software, com conhecimento prévio em Processos de Software, Arquitetura de Software e LPS. Dos participantes, 2 são doutores (25%), 2 são alunos de doutorado (25%), 1 é mestre (12,5%) e 3 são alunos de mestrado (37,5%);

- **Instrumentação**: todos os especialistas receberam um conjunto de documentos, no qual contém: (i) uma cópia do Termo de Consentimento Livre e Esclarecido (TCLE); (ii) uma cópia do Questionário de Caracterização; (iii) uma cópia contendo os conceitos básicos de Processo de Software; (iv) uma cópia contendo os conceitos básicos de SPEM; (v) uma cópia contendo os conceitos básicos do método *UML Components*; e (vi) uma cópia contendo a proposta do processo *SMartyComponents*.

A instrumentação fornecida aos especialistas foram suficientes para responder os formulários eletrônicos, criados utilizando a plataforma do *Google* (*Google Docs*[1]). O estudo foi dividido em dois dias, sendo que no primeiro dia foi realizado o treinamento, além de questões referentes ao *Workflow* de Requisitos do *SMartyComponents*, e o segundo dia referente ao *Workflow* de Especificação. Tal divisão foi planejada com o intuito de evitar a fadiga dos especialistas.

6.4 Execução do Estudo

Essa seção apresenta as etapas seguidas durante a execução deste estudo.

Procedimentos de Participação: a participação de cada especialista no estudo ocorreu da seguinte forma:

1. o especialista recebe os documentos que compõem o estudo;

2. o especialista faz a leitura do TCLE, esclarece as possíveis dúvidas e assina o termo;

3. o especialista faz a leitura e preenche o Questionário de Caracterização de acordo com as suas especialidades;

[1]Disponível em *http://docs.google.com*

4. o pesquisador inicia o treinamento sobre *SMartyComponents*;
5. após o treinamento, o pesquisador esclarece as possíveis dúvidas do especialista;
6. o especialista recebe um link de acesso ao formulário do primeiro dia, contendo as questões referentes ao *Workflow* de Requisitos do *SMartyComponents*;
7. o pesquisador esclarece as possíveis dúvidas antes e durante o treinamento, referente a questões técnicas, de forma que não influencie nas respostas dos especialistas;
8. o especialista preenche o formulário e envia as questões. Em seguida, recebe um email com o link para o segundo dia;
9. no segundo dia o especialista tira as possíveis dúvidas com o pesquisador, e inicia a segunda parte do estudo; e
10. o especialista preenche o formulário e ao término envia as questões.

Os especialistas tiveram um prazo de até 14 dias para a finalização dos questionários, após realizarem treinamento e participação no primeiro dia. É importante destacar que 3 especialistas realizaram os treinamentos e preenchimento dos formulários presencialmente, e com os demais especialistas o treinamento e acompanhamento dos estudos foram feitos via Internet, pelo comunicador instantâneo *Skype*[2], de acordo com a disponibilidade de cada especialista.

6.5 Análise e Interpretação dos Resultados

Os dados coletados dos especialistas foram importados na aplicação *Dedoose*[3] para a análise com base nos conceitos de *Open Coding*. Diante disso, foi possível identificar e classificar 24 códigos nesse estudo:

1. Coerência na Atribuição dos Papéis;
2. Coerência na Atribuição das Tarefas;
3. Definição Correta do Fluxo das Tarefas;
4. Escopo Claro e Suficiente de Artefatos;

[2]Disponível em *http://www.skype.com*
[3]Disponível em *http://www.dedoose.com*

5. Escopo Claro e Suficiente de Atividades;
6. Atribuir Novos Artefatos;
7. Atribuir Novos Papéis;
8. Atribuir Novas Tarefas;
9. Inconsistência nas Tarefas;
10. Inconsistência nos Artefatos;
11. Inconsistência nos Papéis;
12. Reestruturação dos Papéis;
13. Reestruturação das Tarefas;
14. Reestruturação dos Artefatos;
15. Remover Artefatos;
16. Contribui na Organização das Tarefas;
17. Contribui na Definição de Arquiteturas de LPS;
18. Contribui no Entendimento e Previsibilidade do Processo;
19. Contribui na Especificação de Componentes com Variabilidades;
20. Complexidade nas Tarefas do *SMartyProcess*;
21. Complexidade da Atividade Especificação de Componente;
22. Complexidade da Atividade Definição de Requisitos;
23. Complexidade da Atividade Identificação de Componente; e
24. Complexidade da Atividade Interação de Componente.

Em seguida, tais categorias foram reagrupadas em categorias mais abstratas, de acordo com cada formulário específico para cada atividade do *SMartyComponents*, como seguem:

1. [CA] Compreensão da Atividade Definição de Requisitos;
2. [CA] Possíveis Melhorias na Atividade Definição de Requisitos;

3. [CA] Compreensão da Atividade Identificação de Componente;
4. [CA] Possíveis Melhorias na Atividade Identificação de Componente;
5. [CA] Compreensão da Atividade Interação de Componente;
6. [CA] Possíveis Melhorias na Atividade Interação de Componente;
7. [CA] Compreensão da Atividade Especificação de Componente;
8. [CA] Possíveis Melhorias na Atividade Especificação de Componente;
9. [CA] Possíveis Contribuições do *SMartyComponents*;
10. [CA] Maiores Complexidades do *SMartyComponents*;

Para um melhor entendimento das categorias, os códigos gerados e os especialistas são classificados da seguinte forma:

- **Categoria**: para cada categoria é utilizada a abreviatura CAn, tal que n é o número de identificação da lista de classificação (CA1 até CA10);
- **Código**: para cada código é utilizado um determinado número de identificação sequencial; e
- **Especialista**: Para cada especialista será utilizada a abreviatura E + ID, por exemplo: E1, E2... até E8.

A ordem de leitura dessa classificação é a seguinte: (i) Especialista; (ii) Categoria; e (iii) Código. Por exemplo, o *quote*:

- *"Dois pontos que podem contribuir para o workflow: - na definição de requisitos, o especialista de domínio poderia se envolver com "identificar casos de uso"* [E1.CA2.12];

Se refere ao Especialista Nº 1, à categoria CA2: "Possíveis Melhorias na Atividade Definição de Requisitos"; e o Código 12: "Reestruturação dos Papéis". As respostas dos especialistas foram classificadas de acordo com as categorias e *codings* gerados, e são apresentadas na integra no Apêndice C. A Tabela 6.1 mostra a relação dos *codings* com as categorias x especialistas.

Tabela 6.1: Classificação dos Códigos Gerados de Acordo com as Categorias x Especialistas

CA	E1	E2	E3	E4	E5	E6	E7	E8
CA1	1;3;4;5	1;3;4	1;3;4;5	1;2;3;4;5	1;3,4;5	2;4	3;4;5	1;3;4
CA2	8;9;11; 12	6;7;10; 12;14	9;20	11;12	9;10;12	9;10;11; 12;13	10;12	9;10
CA3	1;2;3;5	1;2;3;4	1;2;3;4;5	1;3;4	2;3	1;2;3	1;3;4;5	1;3;4;5
CA4	13	6;7;8; 13;14	7;20	10;12;13 14;23	7;9;10; 12;13;20	10;11;13; 14;15;23	–	7;9;10; 13;20
CA5	1;2;3; 5	1;2;3; 4	1;2;4; 5	1;2;4; 5	1;2;4	1;2;3; 4;5	1;2;3; 4;5	1;2;3; 4;5
CA6	10	7;9	7;12;20	24	7;9;12;20	24	7;10	10;13;20
CA7	1;2;4;5	1;2;3;4	1;2;4;5	1;2;4	1;2;4	1;2;3;4	1;2;3;4;5	2;3;4;5
CA8	9	7;13;14	7;12;20	14;21	7;9,20	13;14;21	7;10	7;13;14; 20
CA9	19	17;19	16;19	19	18	18;19	17	19
CA10	22	21	21	24	20	23	23	21

Com base nas respostas dos especialistas, os códigos gerados foram associados às categorias. As seguir tais associações serão representadas graficamente.

De acordo com as respostas dos especialistas, as categorias CA1 (Compreensão da Atividade Definição de Requisitos), CA3 (Compreensão da Atividade Identificação de Componente), CA5 (Compreensão da Atividade Interação de Componente) e CA7 (Compreensão da Atividade Especificação de Componente), apresentam associações com os seguintes códigos: (a) [1] "Coerência na Atribuição dos Papéis"; (b) [2] "Coerência na Atribuição das Tarefas"; (c) [3] "Definição Correta do Fluxo das Tarefas"; (d) [4] "Escopo Claro e Suficiente de Artefatos"; e (e) [5] "Escopo Claro e Suficiente de Atividades".

A Figura 6.2 apresenta a relação das categorias CA1, CA3, CA5 e CA7 com seus respectivos códigos, de acordo com as respostas dos especialistas.

A categoria CA2 (Possíveis Melhorias na Atividade Definição de Requisitos) apresenta associações com os seguintes códigos: (a) [6] "Atribuir Novos Artefatos"; (b) [7] "Atribuir Novos Papéis"; (c) [8] "Atribuir Novas Tarefas"; (d) [9] "Inconsistência nas Tarefas"; (e) [10] "Inconsistência nos Artefatos"; (f) [11] "Inconsistência nos Papéis"; (g) [12] "Rees-

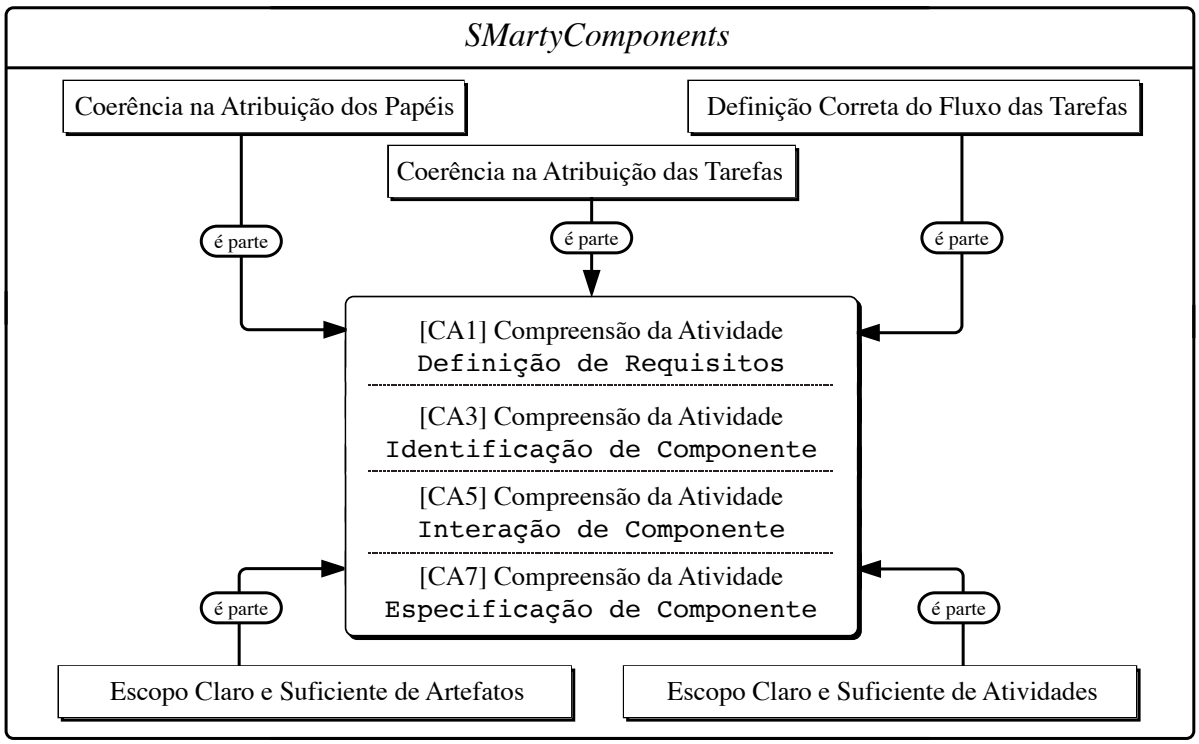

Figura 6.2: Representação Gráfica com as Associações dos Códigos Relacionados às Categorias CA1, CA3, CA5 e CA7.

truturação dos Papéis"; (h) [13] "Reestruturação das Tarefas"; (i) [14] "Reestruturação dos Artefatos"; e (j) [20] "Complexidade nas Tarefas do *SMartyProcess*".

A Figura 6.3 apresenta a relação da categoria CA2 com seus respectivos códigos, de acordo com as respostas dos especialistas.

A categoria CA4 (Possíveis Melhorias na Atividade Identificação de Componente) apresenta associações com os seguintes códigos: (a) [6] "Atribuir Novos Artefatos"; (b) [7] "Atribuir Novos Papéis"; (c) [8] "Atribuir Novas Tarefas"; (d) [9] "Inconsistência nas Tarefas"; (e) [10] "Inconsistência nos Artefatos"; (f) [11] "Inconsistência nos Papéis"; (g) [12] "Reestruturação dos Papéis"; (h) [13] "Reestruturação das Tarefas"; (i) [14] "Reestruturação dos Artefatos"; (j) [15] "Remover Artefatos"; (k) [20] "Complexidade nas Tarefas do SMartyProcess"; e (l) [23] "Complexidade da Atividade Identificação de Componente". A Figura 6.4 apresenta a relação da categoria CA4 com seus respectivos códigos, de acordo com as respostas dos especialistas.

A categoria CA6 (Possíveis Melhorias na Atividade Interação de Componente) apresenta associações com os seguintes códigos: (a) [7] "Atribuir Novos Papéis"; (b) [9] "Inconsistência nas Tarefas"; (c) [10] "Inconsistência nos Artefatos"; (d) [12] "Reestruturação

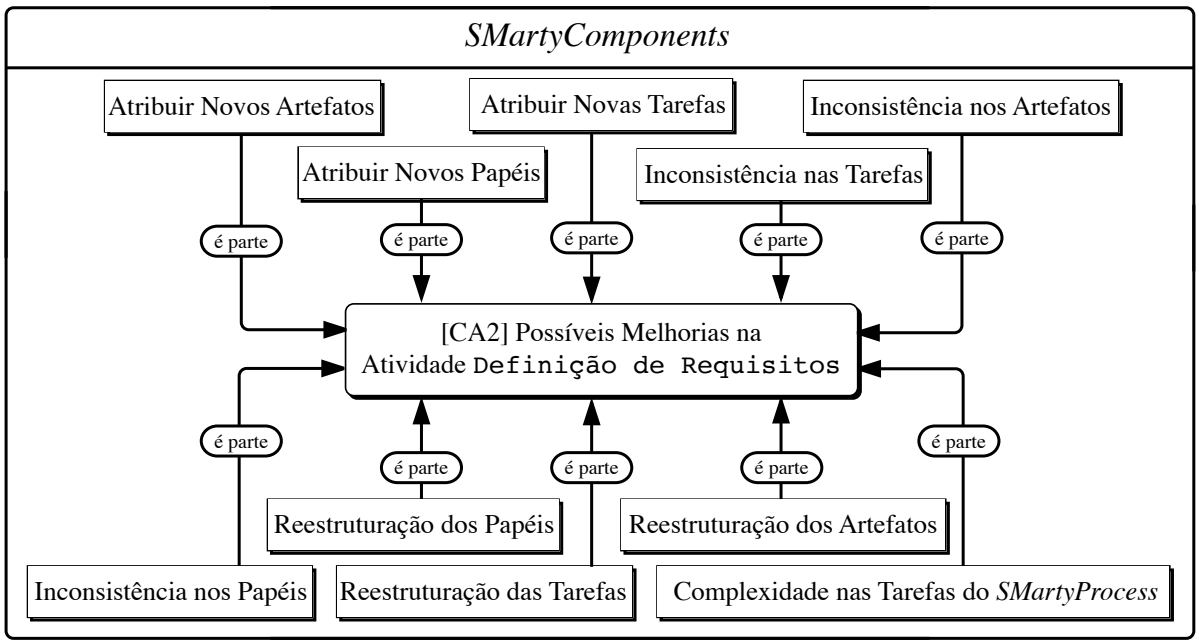

Figura 6.3: Representação Gráfica com as Associações dos Códigos Relacionados à Categoria CA2.

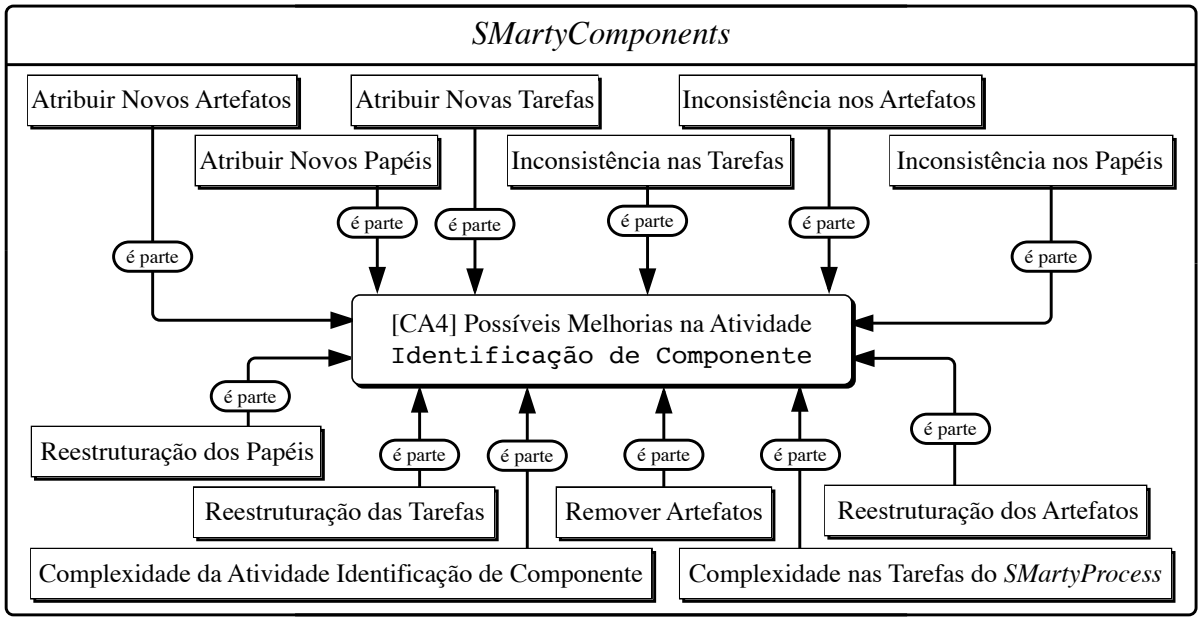

Figura 6.4: Representação Gráfica com as Associações dos Códigos Relacionados à Categoria CA4.

dos Papéis"; (e) [13] "Reestruturação das Tarefas"; (f) [20] "Complexidade nas Tarefas do SMartyProcess"; e (g) [24] "Complexidade da Atividade Interação de Componente".

A Figura 6.5 apresenta a relação da categoria CA6 com seus respectivos códigos, de acordo com as respostas dos especialistas.

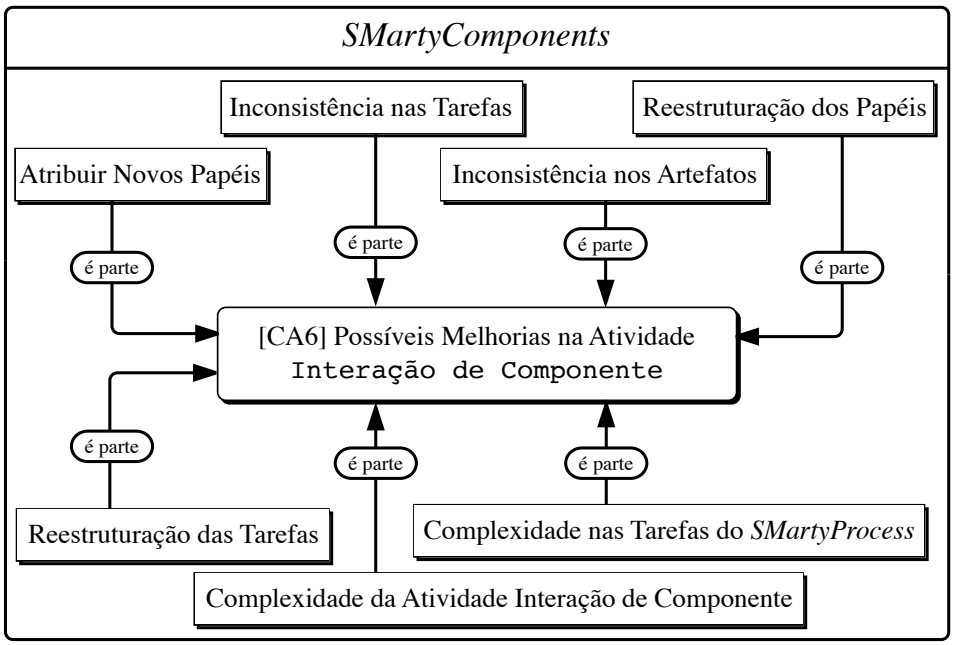

Figura 6.5: Representação Gráfica com as Associações dos Códigos Relacionados à Categoria CA6.

A categoria CA8 (Possíveis Melhorias na Atividade Especificação de Componente) apresenta associações com os seguintes códigos: (a) [7] "Atribuir Novos Papéis"; (b) [9] "Inconsistência nas Tarefas"; (c) [10] "Inconsistência nos Artefatos"; (d) [12] "Reestruturação dos Papéis"; (e) [13] "Reestruturação das Tarefas"; (f) [14] "Reestruturação dos Artefatos";(g) [20] "Complexidade nas Tarefas do SMartyProcess"; e (h) [21] "Complexidade da Atividade Especificação de Componente".

A Figura 6.6 apresenta a relação da categoria CA8 com seus respectivos códigos, de acordo com as respostas dos especialistas.

A categoria CA9 (Possíveis Contribuições do *SMartyComponents*) apresenta associações com os seguintes códigos: (a) [16] "Contribui na Organização das Tarefas"; (b) [17] "Contribui na Definição de Arquiteturas de LPS"; (c) [18] "Contribui no Entendimento e Previsibilidade do Processo"; e (d) [19] "Contribui na Especificação de Componentes com Variabilidades".

A Figura 6.7 apresenta a relação da categoria CA9 com seus respectivos códigos, de acordo com as respostas dos especialistas.

Por fim, a categoria CA10 (Maiores Complexidades do *SMartyComponents*) apresenta associações com os seguintes códigos: (a) [20] "Complexidade nas Tarefas do

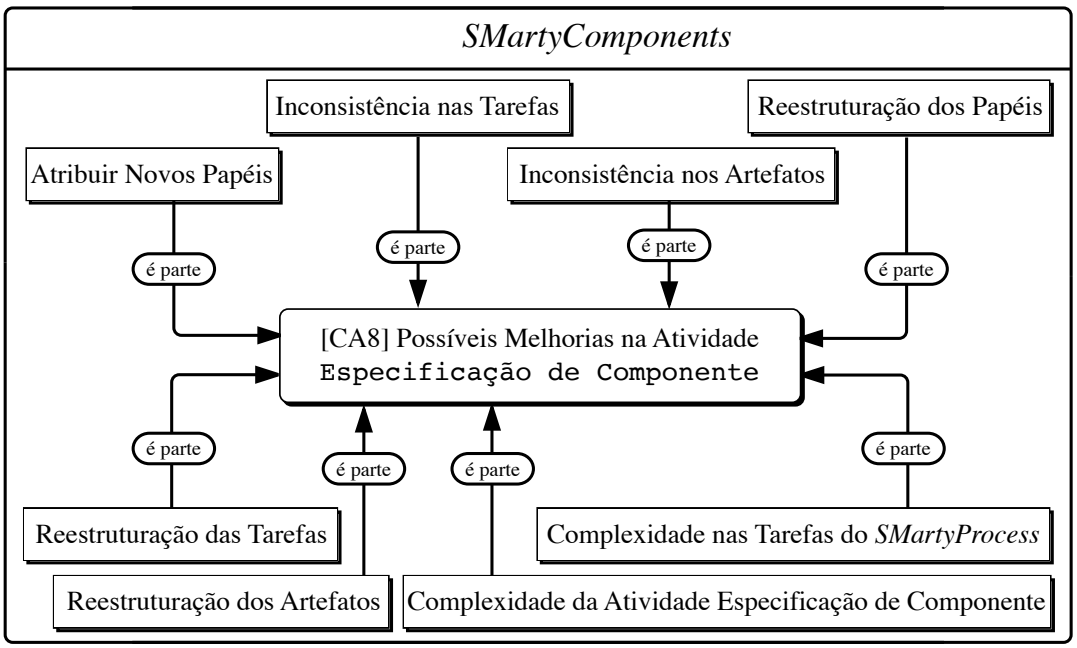

Figura 6.6: Representação Gráfica com as Associações dos Códigos Relacionados à Categoria CA8.

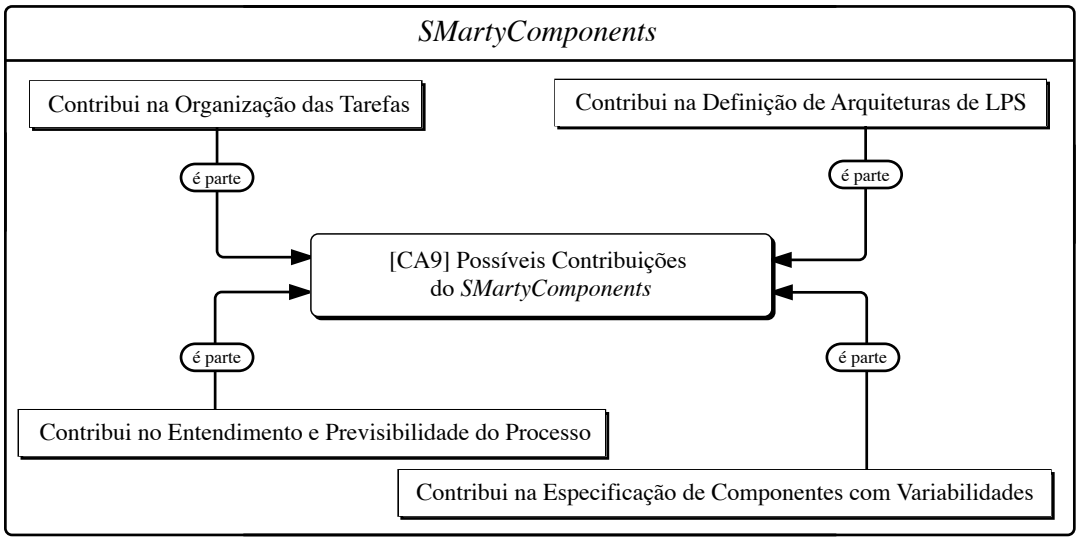

Figura 6.7: Representação Gráfica com as Associações dos Códigos Relacionados à Categoria CA9.

SMartyProcess"; (b) [21] "Complexidade da Atividade Especificação de Componente"; (c) [22] "Complexidade da Atividade Definição de Requisitos"; (d) [23] "Complexidade da Atividade Identificação de Componente"; e (f) [24] "Complexidade da Atividade Interação de Componente".

A Figura 6.8 apresenta a relação da categoria CA10 com seus respectivos códigos, de acordo com as respostas dos especialistas.

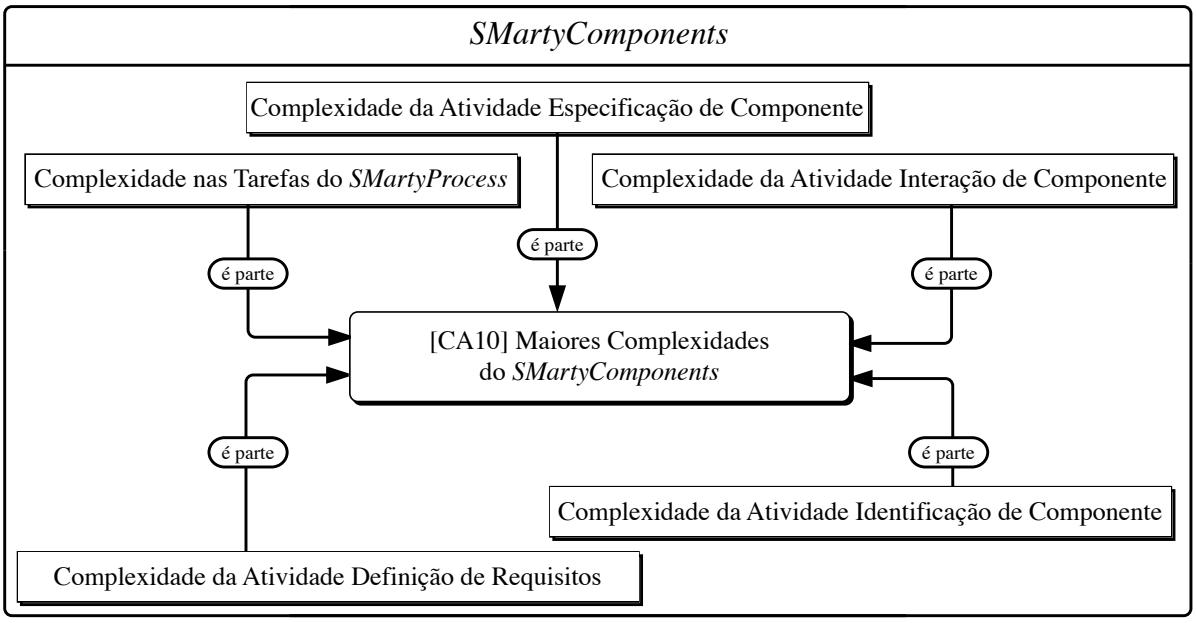

Figura 6.8: Representação Gráfica com as Associações dos Códigos Relacionados à Categoria CA10.

Percebe-se com base na codificação realizada que melhorias são necessárias para o processo proposto. Assim, a Seção 6.7 apresenta tais melhorias com base nas principais categorias identificadas com o *feedback* essencial dos especialistas.

6.6 Avaliação de Validade do Estudo

As ameaças consideradas e as ações tomadas para amenizar alguns dos problemas encontrados durante a execução do estudo são apresentadas nessa seção.

Ameaças à Validade de Conclusão

Uma das principais ameaças à validade de conclusão é o número de especialistas. Dos 15 especialistas previamente selecionados, somente 8 tiveram disponibilidade para contribuir com tal estudo. Porém, pelo fato do estudo qualitativo ser caracterizado pela qualidade dos especialistas, muitas vezes o baixo número de especialistas pode ser considerado. Apesar do número pequeno de especialistas, suas considerações foram significantes para colaborar com a melhoria da proposta.

Ameaças à Validade de *Constructo*

A instrumentação foi avaliada e adequada de acordo com as possíveis modificações sugeridas pelos participantes em estudo piloto. Os especialistas receberam treinamento sobre o *SMartyComponents* a fim de garantir a compreensão necessária para responder os formulários.

Ameaças à Validade Interna

Nesse estudo, as seguintes dificuldades foram percebidas:

- **Diferenças entre os especialistas**: devido ao pequeno tamanho da amostra, as variações entre as habilidades dos especialistas foram poucas, porém perceptíveis. Dessa forma, as tarefas designadas foram idênticas;
- **Acurácia das respostas dos participantes**: uma vez treinados nos conceitos do processo proposto, considerando o nível de conhecimento dos participantes em Arquitetura de Software e LPS, considera-se que as respostas fornecidas são válidas e significantes;
- **Efeito de Fadiga**: Com o intuito de reduzir os efeitos de fadiga dos participantes, o estudo foi dividido em dois dias. No primeiro dia, foi realizado um treinamento sobre a fundamentação teórica necessária para a compreensão do estudo e em seguida 25% das questões formuladas do estudo foram realizadas. O treinamento realizado teve 30 minutos de duração, já os primeiros formulários tiveram em média 50 minutos de duração. No segundo dia, as respostas dos formulários contendo 75% das questões restantes, tiveram em média 100 minutos de duração. Foi sugerido que os especialistas iniciassem as questões do segundo dia de acordo com a sua disponibilidade, com um prazo mínimo de 24 horas e máximo de 14 dias.
- **Outros fatores importantes**: a possível influência nas respostas entre os especialistas foi amenizada devido à supervisão do pesquisador durante a aplicação do estudo para os participantes que realizaram o estudo presencialmente. Em relação aos especialistas que realizaram o estudo via Internet, para estes, o estudo foi aplicado individualmente, tanto o treinamento, quanto os formulários. Assim, é provável que os especialistas tenham realizado o estudo distintamente, em locais, dias e horários diferentes sem o conhecimento prévio de outros especialistas.

Ameaças à Validade Externa

Uma ameaça foi identificada referente aos especialistas. O objetivo inicial do estudo era buscar profissionais, mestres e doutores com experiência em Arquitetura de Software e LPS. Porém, a obtenção destes participantes foi uma das grandes dificuldades deste estudo. Assim, estudantes de pós-graduação com conhecimento sobre tais assuntos foram convidados. Tal ameaça é reduzida com base no estudos de Höst *et al.* (2000).

6.7 Propostas de Melhoria para o *SMartyComponents*

Dada a identificação de propostas de melhorias encontradas por meio da análise das respostas dos especialistas, classificadas nas categorias CA2, CA4, CA6, CA8 e CA10, as sugestões de melhoria foram analisadas e algumas delas foram acatadas para refinar e melhorar a proposta do *SMartyComponents*, enquanto outras não foram acatadas e para tais algumas considerações foram justificadas. As propostas de melhorias estão identificadas numericamente em ordem crescente.

6.7.1 [CA2] Possíveis Melhorias na Atividade Definição de Requisitos

Sugestões Acatadas

Essa seção apresenta as possíveis melhorias sugeridas pelos especialistas, referentes à categoria CA2. Os formulários com as questões encontram-se no Apêndice C.

De acordo com a questão 1.2, as seguintes *quotes* foram levadas em consideração:

- **[E2.CA2.12]**: Entretanto, sugiro que o Especialista de Domínio também participe da atividade de Descrever Processos de Negócios. Isso é necessário, pois deve possibilitar uma discussão ampla e mais profunda com o objetivo de obter um melhor entendimento do que irá ou não conter o Modelo Conceitual de Negócio. Cada papel (Analista de Negócios e Especialista em Domínio) tem uma experiência, as quais devem ser trocadas de maneira bilateral e iterativa.

- **[E5.CA2.12]**: Embora haja uma leve sobrecarga de responsabilidades para o "analista de negócio", percebe-se a necessidade da sua participação em todas as tarefas da atividade de "definição de requisitos". Como forma de dividir melhor a carga, o "especialista de domínio", poderia auxilia-lo na tarefa de "descrever os processos de negócio".

De acordo com a questão 1.4, as seguintes *quotes* foram levadas em consideração:

- [**E7.CA2.12**]:

- Entretanto, vejo a necessidade de atribuir o papel Especialista de Domínio na tarefa de auxiliar a descrição dos processo de negócios em conjunto com o Analista de Negócios.

- [**E7.CA1.4**]: Minha compreensão em relação ao Workflow de Requisitos da SMarty Components, é que ele estende o Workflow de Requisitos do UML Components. Inicialmente utiliza-se os Requisitos de Negócio para Descrever os Processos de Negócio e Identificar os Casos de Uso, além de usar o Conhecimento do Domínio da LPS como entrada para Desenvolver o Modelo Conceitual de Negócio. Após gerados o Modelo de Casos de Uso e o Modelo Conceitual de Negócio, estes são entrada para Identificar as Variabilidades. Após isso são realizadas as tarefas: Delimitar Variabilidades, Rastrear e controlar Variabilidades e Representar variabilidades.

Portanto, a seguinte melhoria deve ser acatada:

1. Relacionar o `Especialista em Domínio` com a tarefa `Descrever os Processos de Negócios`.

O especialista de domínio auxilia a desenvolver o modelo conceitual de negócio, sendo assim, é relevante pensar que ter um conhecimento dos processos de negócio pode contribuir de forma relevante para desenvolver o modelo conceitual de negócio.

Como complemento dessa melhoria, de acordo com a questão 1.3, a seguinte *quote* foi levada em consideração:

- [**E6.CA2.9**]: Não, para "Desenvolver o Modelo Conceitual de Negócio"é necessário ter executado a tarefa "Descrever Processos de Negócio", não existe a seta pontilhada representando este fluxo, deixando a entender que poderiam ser executadas paralelamente. Meu entendimento é que isto não possa ocorrer.

Ainda, de acordo com a questão 2.2, a seguinte *quote* foi levada em consideração:

- [**E6.CA2.12**]: Minha sugestão seria que, para "Desenvolver o Modelo Conceitual de Negócio"é necessário ter executado a tarefa "Descrever Processos de Negócio", não existe a seta pontilhada representando este fluxo, deixando a entender que poderiam ser executadas paralelamente. Meu entendimento é que isto não possa ocorrer.

Portanto, levando em consideração as questões 1.3 e 2.2, juntamente com as respostas do especialista E6, a seguinte melhoria deve ser acatada:

2. A tarefa `Desenvolver Modelo Conceitual de Negócio` deve depender da tarefa `Descrever Processos de Negócio`.

A proposta do *SMartyComponents* é que o modelo conceitual de negócio seja criado a partir dos processos de negócio. Dessa forma, é relevante a sua dependência.

Portanto, de acordo com as melhorias sugeridas, o *Workflow* de Requisitos do *SMarty-Components* foi refinado e atualizado. A Figura 6.9 apresenta a versão anterior a essas melhorias, sendo que a Figura 5.4 apresenta a versão atualizada.

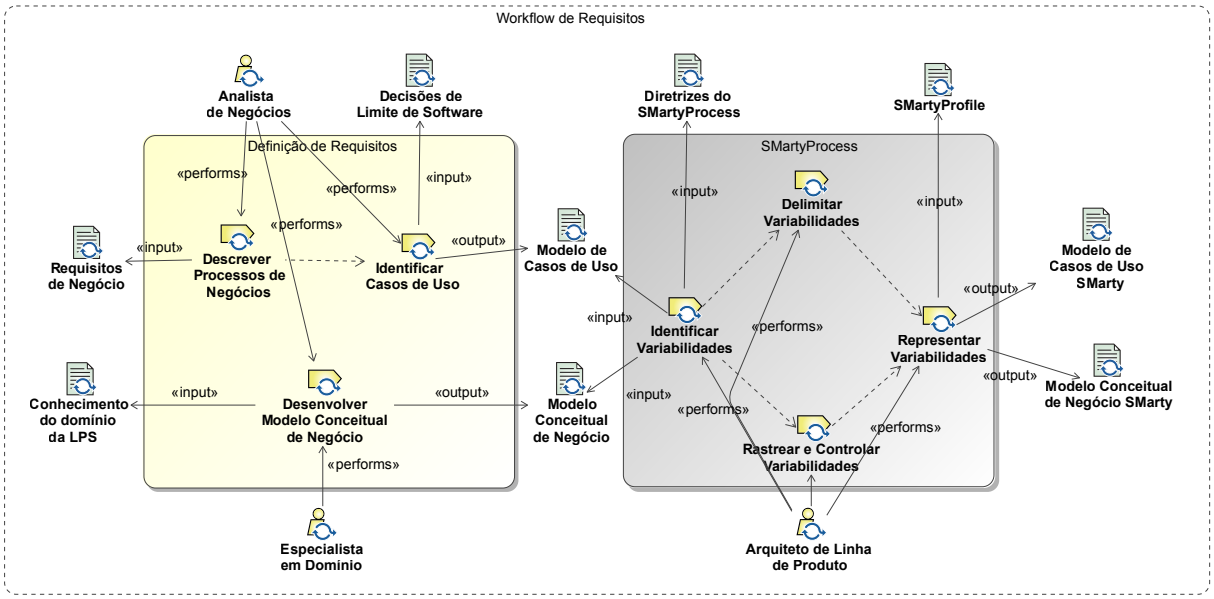

Figura 6.9: *Workflow* de Requisitos do *SMartyComponents* sem as Melhorias Sugeridas

Sugestões Não Acatadas

Nessa subseção serão discutidas algumas sugestões dos especialistas que não foram acatadas como possíveis melhorias.

- **[E1.CA2.12]**: Dois pontos que podem contribuir para o workflow: - na definição de requisitos, o especialista de domínio poderia se envolver com "identificar casos de uso";
- **[E1.CA2.8]**: com uma nova atividade que seria "validar casos de uso". Isso porque ele que deve bater o martelo sobre o que de fato é um caso de uso;
- **[E1.CA2.8]**: uma atividade de "validar modelo conceitual x casos de uso". Nos casos de uso acabamos definindo indiretamente os elementos do modelo conceitual. Então, fazer uma verificação se os dois modelos batem seria interessante.

- [**E1.C2.12**]: Respondi a 1.2 na 1.1. =) Envolver o especialista de domínio na construção dos casos de uso seria interessante.

Diante das *quotes* apresentadas, tais sugestões tornariam a atividade `Definição de Requisitos` do *UML Components* muito complexa e confusa. Primeiramente em relação à atribuição do papel Especialista de Domínio tomar as decisões sobre o que poderia ser considerado um caso de uso, e uma outra questão é sobre a sugestão da tarefa `Validar Modelo Conceitual x Casos de Uso`. Por meio das definições do *UML Components*, tais modelos não teriam uma relação próxima, o que torna inviável essa sugestão para esse processo.

- [**E2.CA2.6**]: No entanto, um novo artefato poderia ser gerado entre o Modelo de Casos de Uso e o Modelo Conceitual de Negócio para propiciar uma ponte entre os aspectos do negócio, possibilitando verificar o que realmente é um Caso de Uso adequado para uma determinada especificação.

No *UML Components*, tais artefatos são provenientes dos processos de negócio, porém eles tem funções distintas, o que torna inviável adaptar tais definições e propor um artefato que não é padrão do *UML Components*.

- [**E6.CA2.13**]: Acredito que o "Especialista em Domínio"poderia ser um ponto de variação, pois o "Analista de negócios"para um projeto baseado em DBC poderia possuir os conhecimentos necessários para os dois papéis.

Tal *quote* foi identificada como inviável para este processo, pois ela se refere à modelagem de linha de processo de software com variabilidades, sendo assim não é do escopo deste trabalho.

- [**E7.CA2.12**]: Acredito que o Especialista em Domínio também poderia apoiar a atividade "Identificar variabilidades", juntamente como Arquiteto de Linha de Produto.

As tarefas realizadas por tais papéis são distintas. O Arquiteto de LPS é responsável pelas tarefas que envolvem o *SMaryProcess*, enquanto o O Especialista em Domínio é responsável pelas tarefas que envolvem a organização e distribuição da arquitetura de software em si. Sendo assim, tal sugestão foi identificada como inviável.

- [**E2.CA2.7**]: Contudo, senti a falta de mais um papel para auxiliar o Arquiteto de Linha de Produto, como por exemplo, um outro tipo de Especialista em Linha de Produto.

O Arquiteto de LPS acaba não sendo responsável por realizar todas as tarefas do *SMartyProcess* de uma vez, mas sim as tarefas específicas para cada modelo. Entretanto, mais de um Arquiteto de LPS poderia auxiliar a realizar tais tarefas, pois são especialistas em LPSs. Assim, não há necessidade de modelar mais um papel para o *SMartyProcess* nesse momento.

- **[E2.CA2.12]**: O Analista de Negócios e o Especialista de Domínio poderiam ser unidos em um único papel, como por exemplo, Analista de Negócios/Domínio ou Arquiteto de Software. Assim, um padrão seria mantido como apresentado no SMartyProcess que contém apenas um papel denominado como Arquiteto de Linha de Produto.

Nesse primeiro *workflow* do *UML Components*, compreende-se que os dois papéis são importantes, de acordo com as sugestões dos autores. Assim, para as atividades posteriores, foram acatadas as sugestões de padronização de um papel para as atividades provenientes do *UML Components*.

- **[E2.CA2.12]**: Os Requisitos de Negócio e o Conhecimento do Domínio da LPS poderiam estar relacionados de maneira bilateral para possibilitar uma melhor compreensão do que realmente o cliente necessita considerando um possível novo projeto de software.

O *UML Components* não garante que exista um artefato referente ao conhecimento do domínio, seja da LPS ou de um software, portanto, se não existir préviamente, tal artefato pode ser criado com base nas informações levantadas em reuniões com os clientes. De todo modo, eles já estarão relacionados. Não identificamos a necessidade de adicionar nenhum outro elemento para deixar tal informação mais evidente.

- **[E5.CA2.9]**: A única preocupação que me surgiu ao analisar o processo proposto é que a identificação de variabilidades iniciou apenas em uma segunda etapa (tarefa), e tomou por base o resultado de uma primeira etapa (tarefa);

- **[E5.CA2.9, E5.CA2.10]**: Essa abordagem pode limitar a identificação de todas as possíveis variabilidades relativas ao domínio proposto. Pois, pode fazer com que o arquiteto da LPS se limite aos cenários que foram idealizados pelo analista de negócio e especialista do domínio; ou então a segunda tarefa vai gerar a necessidade de que a primeira tarefa seja executada novamente (para cobrir possíveis gaps que tenham sido deixados).

Não vemos tal preocupação, pois as atividades são iterativas e incrementais. Portanto, a identificação de variabilidades só ocorrerá após os artefatos terem sidos gerados.

- **[E2.CA2.14]**: Sugiro que os Produtos de Trabalho = Modelo de Casos de Uso e Modelo Conceitual de Negócio sejam unidos um único Produto de Trabalho para tornar a compreensão de que serão, após, modelados utilizando o SMartyProcess.

Tal sugestão é inviável pelo fato de que ambos os artefatos tem finalidades distintas durante o processo de desenvolvimento. Assim, para que tudo ocorra da maneira esperada, de acordo com o *UML Components*, devem existir ambos artefatos.

6.7.2 [CA4] Possíveis Melhorias na Atividade Identificação de Componente

Sugestões Acatadas

Essa seção apresenta as possíveis melhorias sugeridas pelos especialistas, referentes à categoria CA4.

De acordo com a questão 3.2, as seguintes *quotes* foram levadas em consideração:

- **[E3.CA4.7]**: Não caso este papel não tenha conhecimento sobre desenvolvimento baseado em componentes. Não consegui identificar se este arquiteto se encontra em alguma documentação sobre UML componentes, mas para realizar as tarefas ele deveria ter esse conhecimento sobre desenvolvimento baseado em componentes, porém a nomenclatura não aponta este fator.

- **[E5.CA4.7]**: Em parte. Seria interessante que um arquiteto de software participasse dessa tarefa.

- **[E5.CA4.7, E5.CA4.12]**: embora que a participação do arquiteto da LPS seja fundamental para contextualizar o trabalho. Embora o trabalho desses dois papéis tenha as suas similaridades, visam atender a objetivos claramente distintos.

De acordo com a questão 3.5, a seguinte *quote* foi levada em consideração:

- **[E3.CA4.7]**: Talvez o papel Arquiteto de Linha de Produto possa ser alterado por um Arquiteto de Componentes.

Portanto, com base nas questões 3.2 e 3.5 e as respostas dos especialistas E3 e E5, a seguinte melhoria deve ser acatada:

3. Atribuição de um papel voltado ao Desenvolvimento Baseado em Componentes.

Como complemento dessa melhoria, de acordo com as questões 3.2 juntamente com a resposta do especialista E5, a seguinte *quote* foi levada em consideração:

- **[E5.CA4.7]**: participação de um arquiteto de software, como responsável pela tarefa de "identificar interfaces de sistema e operações"e como auxiliar nas demais tarefas de "identificação de componentes".

Ainda, de acordo com a questão 3.3 juntamente com a resposta do especialista E2, a seguinte *quote* foi levada em consideração:

- **[E2.CA2.7]**: Contudo, senti a falta de mais um papel para auxiliar o Arquiteto de Linha de Produto, como por exemplo, um outro tipo de Especialista em Linha de Produto.

Portanto, a seguinte melhoria deve ser acatada:

4. Atribuição de um Papel para auxiliar o `Arquiteto de LPS` na atividade anterior ao *SMartyProcess*.

Dessa forma, é compreensível a sobrecarga de trabalho por um único papel, e a presença de um outro papel voltado ao desenvolvimento baseado em componentes pode ser muito relevante na atividade `Identificação de Componente`. Assim, um papel `Arquiteto de Software` ficará responsável pelas tarefas desempenhadas nessa atividade.

De acordo com as questõe 3.3, a seguinte *quote* foi levada em consideração:

- **[E6.CA4.11]**: Outro ponto é na verificação das interfaces existentes, esta tarefa deveria ser executada na etapa seguinte, provisionamento.

Ainda, de acordo com a questão 3.5, a seguinte *quote* foi levada em consideração:

- **[E6.CA4.15]**: Poderia remover a verificação das interfaces existentes.

Portanto, a seguinte melhoria deve ser acatada:

5. *Interfaces Existentes* não deveriam existir antes de executar a tarefa de `Identificar Interfaces de Negócios`.

As possíveis *Interfaces Existentes* podem apoiar na identificação das Interfaces de Negócio, porém, pode acabar sendo confundida com o fato das tarefas estarem identificando as interfaces, nesse caso elas ainda não teriam sido geradas. Toda via, pode acontecer de ter alguma interface existente. Dessa forma, para amenizar isso, as possíveis `Interfaces Existentes` estarão contidas no artefato `Recursos Existentes`, assim, existirá uma dependência da tarefa `Identificar Interfaces de Negócio` para este artefato, e na modelagem `Interfaces Existentes` deixará de existir.

Portanto, de acordo com as melhorias sugeridas, a atividade `Identificação de Componente` do *Workflow* de Especificação do *SMartyComponents* foi refinada e atualizada. A Figura 6.10 apresenta a versão anterior da atividade `Identificação de Componente`, enquanto a Figura 5.19 apresenta a versão atualizada.

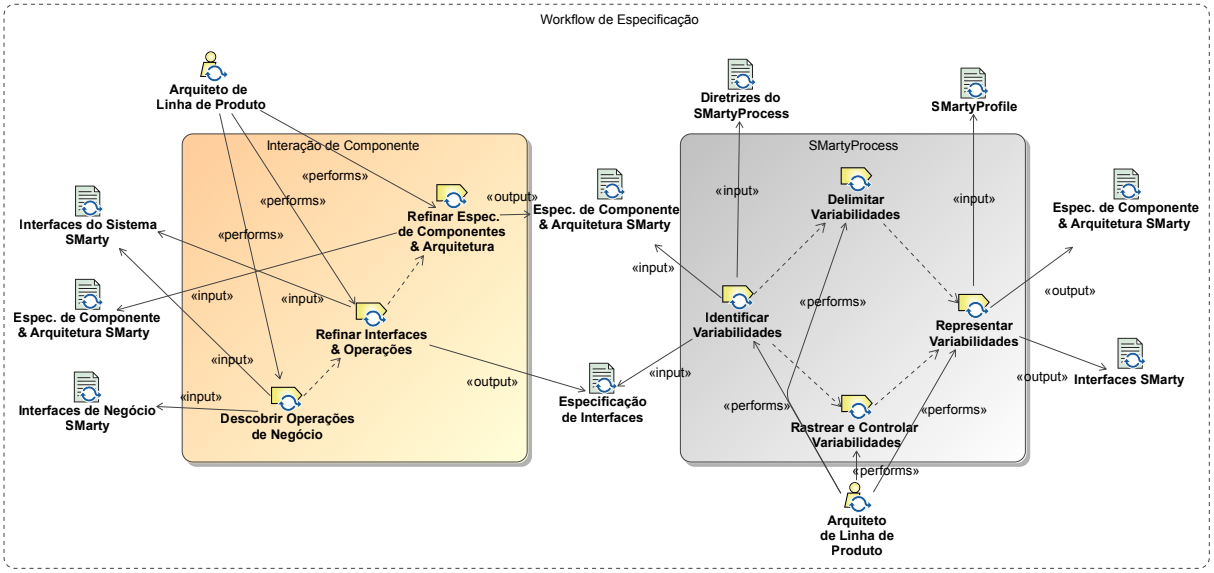

Figura 6.10: Atividade `Identificação de Componente` do *Workflow* de Especificação do *SMartyComponents* sem as Melhorias Sugeridas.

Sugestões Não Acatadas

Nessa subseção serão discutidas algumas sugestões dos especialistas que não foram acatadas como possíveis melhorias.

- **[E2.CA4.6, E2.CA4.14]**: No entanto, um novo artefato (atividade) poderia ser gerado entre o Modelo de Tipo de Negócio e as Interfaces de Negócio, ou então, poderiam ser unidos em um único artefato. Desse modo, poderia propiciar uma ponte entre os dois de forma a permitir verificar o que é realmente adequado para uma determinada especificação.

Os artefatos existentes já preenchem os requisitos para gerar uma arquitetura de software baseada em componentes, de acordo com o *UML Components*, unir os artefatos então é improvável pois eles tem funções distintas durante o ciclo de vida da LPS, e tal união poderia prejudicar alguma atividade que necessite de tal artefato para sua realização.

- [**E2.CA4.7**]: Contudo, sugiro que o Especialista de Domínio auxilie ou apoie o Arquiteto de Linha de Produto durante a tarefa Desenvolver Modelo de Tipo de Negócio. Acredito que isso possa colaborar com uma melhor modelagem da atividade de output referente a atividade Modelo de Tipo de Negócio.

- [**E2.CA4.7**]: Entretanto, vejo a necessidade de atribuir o papel Analista de Negócios na tarefa de auxiliar na descrição da tarefa Desenvolver Modelo de Tipo de Negócio.

- [**E8.CA4.7**]: entretanto creio que mesmo sem estar no método original da UML Components, o especialista do domínio poderia ter uma participação mesmo que para validação dos entregáveis nessa atividade.

O Especialista em Domínio e/ou Analista de Negócios não percente a essa atividade, suas considerações foram atribuídas no primeiro momento (*Workflow* de Requisitos) onde os processos de negócio foram definidos. A partir daí, o Arquiteto de Software tomará conta das responsabilidades da arquitetura de software em si. Tal decisão foi tomada com base nas sugestões acatadas para os papéis das atividades do *Workflow* de Especificação.

- [**E8.CA4.13**]: mas ainda assim pensaria em agrupar duas ou três tarefas mas fazê-las na linha de cada artefato, por exemplo, analisar e representar variabilidades de interfaces do sistema, analisar e representar variabilidades de espec. de comp. e assim por diante.

Tais tarefas são idependentes, por esse motivo são realizadas separadamente. Se fossem criados de forma unificada, o processo se tornaria muito mais complexo, pois para cada tarefa do *UML Components* todas as tarefas do *SMartyProcess* deveriam ser executadas. Por esse motivo, tal sugestão não é viável.

- Um refinamento nas atividades poderia ser interessante para de fato contribuir na subdivisão de tarefas para obter determinado output. Nos casos intermediários acontece isso (como no desenvolver-identificar-criar), mas o objetivo final dessas três atividades é a arquitetura. Refinando, principalmente a parte de criar arquitetura, seria interessante [**E1.CA4.13**];

Tal refinamento já acontece nas atividades posteriores do *Workflow* de Especificação.

- **[E2.CA4.8]**: Somente sugiro que seja criada uma outra atividade entre o Modelo de Tipo de Negócio e as Interfaces de Negócio;
- **[E2.CA4.8]**: bem como entre Modelo de Tipo de Negócio SMarty e as Interfaces de Negócio SMarty;
- **[E2.CA4.13]**: Poderiam estar relacionados de maneira bilateral para possibilitar uma melhor compreensão do que realmente o cliente necessita considerando um possível novo projeto de software.

Visto que cada artefato e tarefa foram propostos originalmente pelo *UML Components*, não há a necessidade implícita de realizar tal alteração, pois não foi identificado um propósito definido nesta sugestão. Os artefatos gerados pelas tarefas do *UML Components* possuem seus propósitos bem definidos, o que acaba justificando não acatar tal consideração.

- **[E4.CA4.10]**: Não ficou claro inicialmente a função do artefato "Modelo de Tipo de Negócio";
- **[E4.CA4.14]**: Logo, tal artefato poderia ter sua identificação (nome) alterada, de forma a apresentar de maneira mais intuitiva a sua função;

O artefato `Modelo de Tipo de Negócio` artefato é criado na primeira atividade do *workflow* de Especificação, porém só é utilizado na última atividade do *workflow*, Especificação de Componente. Tal artefato é utilizado como base para gerar o artefato `Modelo de Informação de Interface`.

6.7.3 [CA6] Possíveis Melhorias na Atividade Interação de Componente

Sugestões Acatadas

Essa seção apresenta as possíveis melhorias sugeridas pelos especialistas, referentes à categoria CA6.

De acordo com a questão 4.2, a seguinte *quote* foi levada em consideração:

- **[E2.CA6.7]**: Contudo, sugiro a especificação de outro papel, como um Engenheiro de Software, para auxiliar ou apoiar o Arquiteto de Linha de Produto durante as

tarefas em geral. Acredito que isso possa colaborar com uma melhor modelagem das atividade output: Espec. de Componente & Arquitetura SMarty e Especificação de Interfaces.

Ainda, de acordo com a questão 4.3, a seguinte *quote* foi levada em consideração:

- **[E2.CA6.7]**: No entanto, senti a falta de mais um papel para auxiliar o Arquiteto de Linha de Produto, como por exemplo, um outro tipo de Especialista em Linha de Produto.

Portanto, a seguinte melhoria deve ser acatada:

6. Atribuição de um Papel para auxiliar o `Arquiteto de LPS` na atividade anterior ao *SMartyProcess*.

Como complemento dessa melhoria, de acordo com a questão 4.2, as seguintes *quotes* foram levadas em consideração:

- **[E3.CA6.12]**: Não por conta da nomenclatura pois é necessário conhecimento sobre desenvolvimento baseado em componentes para realizar as tarefas definidas. Sim se o papel possuir este conhecimento.
- **[E5.CA6.12]**: Como na atividade anterior, creio que poderia participar também um arquiteto de software para trabalhar em conjunto e complementar a visão do arquiteto da LPS - mesma intenção, mas objetivos distintos.

Ainda, de acordo com a questão 4.5, as seguintes *quotes* foram levadas em consideração:

- **[E3.CA6.12]**: Novamente acredito que a nomenclatura do papel não leva em consideração o conhecimento sobre desenvolvimento baseado em componentes. Talvez um novo papel seja necessário.
- **[E5.CA6.7]**: A proposta seria a inclusão do papel de arquiteto de software, somando esforços com o arquiteto da LPS no desenvolvimento das tarefas relativas a essa atividade.

Portanto, a seguinte melhoria deve ser acatada:

7. Atribuição de um papel voltado ao Desenvolvimento Baseado em Componentes.

Dessa forma, novamente é compreensível a sobrecarga de trabalho por um único papel, e a presença de um outro papel voltado ao desenvolvimento baseado em componentes pode ser muito relevante na atividade `Interação de Componente`. Assim, um papel `Arquiteto de Software` ficará responsável pelas tarefas desempenhadas nessa atividade.

Portanto, de acordo com as melhorias sugeridas, a atividade `Interação de Componente` do *Workflow* de Especificação do *SMartyComponents* foi refinada e atualizada. A Figura 6.11 apresenta a versão anterior da atividade `Interação de Componente`, enquanto a Figura 5.19 apresenta a versão atualizada.

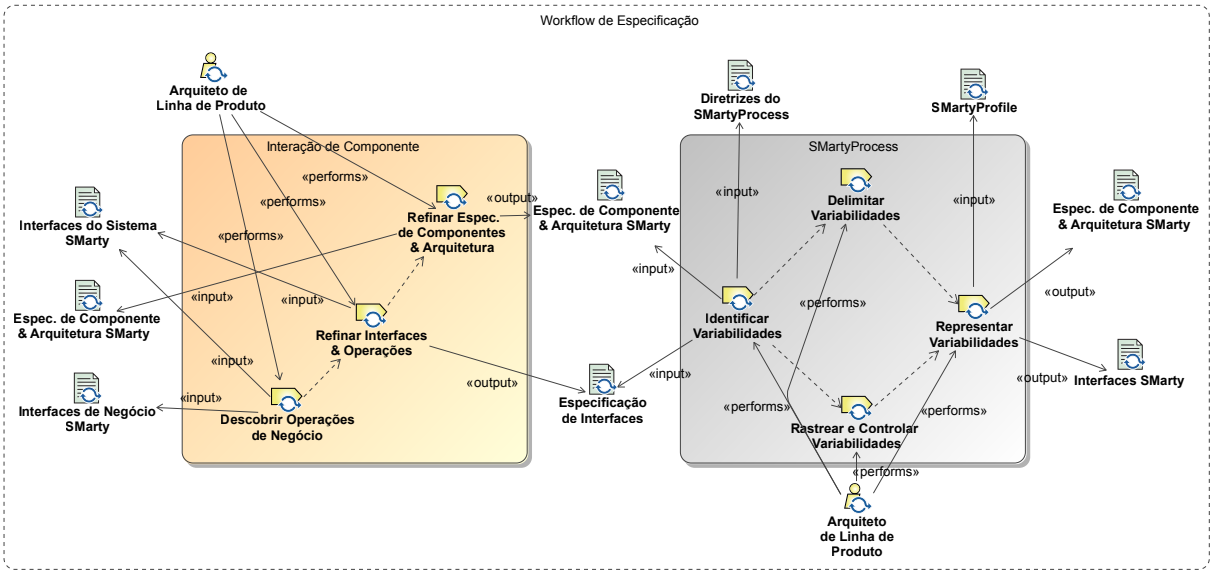

Figura 6.11: Atividade *Interação de Componente* do *Workflow* de Especificação do *SMartyComponents* sem as Melhorias Sugeridas

Sugestões Não Acatadas

Nessa subseção serão discutidas algumas sugestões dos especialistas que não foram acatadas como possíveis melhorias.

- [**E2.CA6.9**]: Entretanto existem exceções, pois não entendo por que a tarefa Descobrir Operações de Negócio está associada como um input para a atividade Interfaces do Sistema SMarty. Não está claro o objetivo disso. Sugiro que reveja esta associação.

- [**E8.CA6.10**]: Vale a ressalva que o output da atividade de interação de componentes antes do smartyProcess não deve ter o sufixo "Smarty"no artefato de saída.

Tal artefato de entrada, `Interfaces do Sistema SMarty` é proviniente da atividade anterior, Identificação de Componente, o que justifica a sua existência nesse momento. Os artefatos de saída de uma atividade são tomados de entrada pela próxima atividade, alimentando assim o fluxo entre as suas tarefas.

- **[E5.CA6.9]**: Em parte. Visto que a tarefa de "interação de componentes"só realiza um REFINAMENTO nas interfaces e componentes (produzidos na atividade anterior), não fica clara a necessidade dessa tarefa. Já a segunda tarefa repete as tarefas que já foram realizadas na atividade anterior. Creio que a necessidade dessa tarefa carece de uma melhor justificativa.

A atividade Interação de Componente tem por objetivo refinar as interfaces, componentes e suas interações. Nesse momento tal refinamento é feito, por isso justifica-se a sua existência.

- **[E7.CA6.7]**: Porém em relação a descobrir operações de negócio,o analista de negócios ou especialista no domínio não poderiam contribuir?.

- **[E2.CA6.7]**: Entretanto, vejo a necessidade de atribuir o papel Analista de Negócios na tarefa de auxiliar na descrição da tarefa Descobrir Operações de Negócio;

O Especialista em Domínio e/ou Analista de Negócios não percente a essa atividade, suas considerações foram atribuídas no primeiro momento (*Workflow* de Requisitos) onde os processos de negócio foram definidos. A partir daí, o Arquiteto de Software tomará conta das responsabilidades da arquitetura de software em si. Tal decisão foi tomada com base nas sugestões acatadas para os papéis das atividades do *Workflow* de Especificação.

- **[E2.CA6.9]**: Somente sugiro que seja criada uma outra atividade entre o Espec. de Componente & Arquitetura SMarty e Especificação de Interfaces, bem como entre Espec. de Componente & Arquitetura SMarty e as Interfaces SMarty. Poderiam estar relacionados de maneira bilateral para possibilitar uma melhor compreensão do que realmente o cliente necessita considerando um possível novo projeto de software.

Visto que cada artefato e tarefa propostos originalmente pelo *UML Components*, não há a necessidade implícita de realizar tal alteração, pois não foi identificado um propósito definido nesta sugestão. Os artefatos gerados pelas tarefas do *UML Components* possuem seus propósitos bem definidos, o que acaba justificando não acatar tal consideração.

6.7.4 [CA8] Possíveis Melhorias na Atividade Especificação de Componente

Sugestões Acatadas

Essa seção apresenta as possíveis melhorias sugeridas pelos especialistas, referentes à categoria CA8.

De acordo com com a questão 5.2, a seguinte *quote* foi levada em consideração:

- **[E2.C8.7]** Contudo, sugiro a especificação de outro papel, como um Engenheiro de Software, para auxiliar ou apoiar o Arquiteto de Linha de Produto durante as tarefas em geral.

Portanto, a seguinte melhoria deve ser considerada:

8. Atribuição de um Papel para auxiliar o `Arquiteto de LPS` na atividade anterior ao *SMartyProcess*.

Como complemento dessa melhoria, ainda de acordo com a questão 5.2, as seguintes *quotes* foram levadas em consideração:

- **[E3.C8.12]**: Novamente, a nomenclatura Arquiteto de Linha de Produto apenas me remete ao conhecimento de definição e gerenciamento de linhas de produto. Não sei se aborda também o conhecimento de desenvolvimento baseado em componentes.
- **[E5.C8.7]**: Sim. Embora creio que o auxílio de um arquiteto de software experiente seria muito útil no desenvolvimento de tarefas como aquela responsável por "especificar restrições de componentes - interface".
- **[E8.C8.7]**: Por mais que certamente o principal papel nesta atividade é desempenhado pelo arquiteto da LPS, eu considero que a participação de outros papéis como desenvolvedores ou especificadores de componentes poderiam subsidiar ou validar as atividades do arquiteto.

Ainda, de acordo com a questão 5.3, a seguinte *quote* foi levada em consideração:

- **[E2.C8.7]**: No entanto, senti a falta de mais um papel para auxiliar o Arquiteto de Linha de Produto, como por exemplo, um outro tipo de Especialista em Linha de Produto.

Portanto, a seguinte melhoria deve ser considerada:

9. Atribuição de um papel voltado ao Desenvolvimento Baseado em Componentes.

Dessa forma, semelhante às outras atividades desse *workflow*, é compreensível a sobrecarga de trabalho por um único papel, e a presença de um outro papel voltado ao desenvolvimento baseado em componentes pode ser muito relevante na atividade `Especificação de Componente`. Assim, um papel `Arquiteto de Software` ficará responsável pelas tarefas desempenhadas nessa atividade.

De acordo com a questão 5.5, a seguinte *quote* foi levada em consideração:

- [**E6.C8.14**]: O artefato "Especificação de Arquitetura"não é algo definido neste momento, os componentes devem respeitar uma arquitetura pré-definida.

Ainda, de acordo com a questão 6.2, as seguintes *quotes* foram levadas em consideração:

- [**E3.C7.2**]: Dois artefatos são gerados neste modelo: Especificação de Arquitetura e Especificação de Componentes. No UML Components original aparentemente esses dois artefatos são somente um, porém não afeta o modelo;
- [**E6.C8.14**]: A saída do artefato de "especificação de arquitetura".

Portanto, a seguinte melhoria deve ser considerada:

10. Os artefatos de saída não correspondem ao padrão do *UML Components*.

Na versão anterior de *SMartyComponents*, sem as devidas propostas de melhorias, o artefato `Especificação de Componentes & Arquitetura` havia sido decomposto e artefatos separados, sendo eles `Especificação de Arquitetura` e `Especificação de Componentes`. Analisando bem, há um retrabalho em identificar as variabilidades nos dois, além de não ficar no padrão do *UML Components*. Dessa forma, os artefatos foram unidos em um só `Espec. de Componentes & Arquitetura SMarty`, semelhante à nomenclatura do mesmo artefato de entrada. Após a última interação com o *SMarty-Process*, tem-se a arquitetura definida da LPS, por isso o artefato `Arquitetura SMarty Componentizada`.

Portanto, de acordo com as melhorias sugeridas, a atividade `Especificação de Componente` do *Workflow* de Especificação do *SMartyComponents* foi refinada e atualizada. A Figura 6.12 apresenta a versão anterior da atividade `Especificação de Componente`, enquanto a Figura 5.24 apresenta a versão atualizada.

As questões gerais sobre o *Workflow* de Especificação, 6.1 e 6.2, sugerem alterações gerais, que já foram previamente realizadas, como segue:

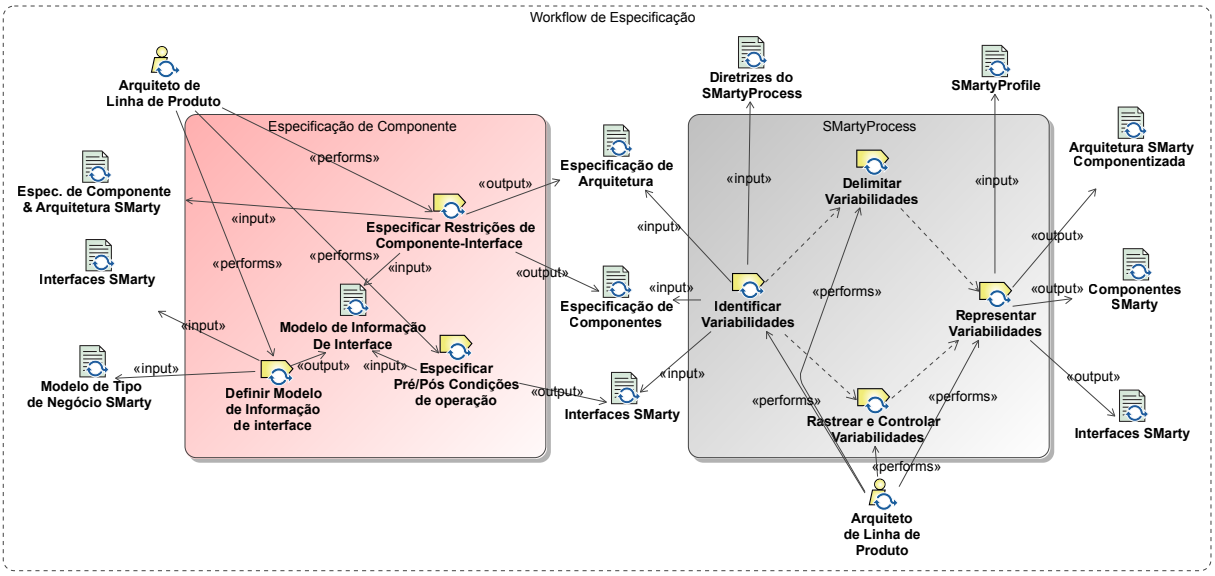

Figura 6.12: Atividade `Especificação de Componente` do *Workflow* de Especificação do *SMartyComponents* sem as Melhorias Sugeridas.

- Atribuição de um Papel para auxiliar o `Arquiteto de LPS` [CA4, CA6 e CA8];
- Atribuição de um papel voltado ao Desenvolvimento Baseado em Componentes [CA4, CA6 e CA8]; e
- Os artefatos de saída não correspondem ao padrão do *UML Components* [CA8].

Sugestões Não Acatadas

Nessa subseção serão discutidas algumas sugestões dos especialistas que não foram acatadas como possíveis melhorias.

- **[E8.C8.14]**: O output do UML Components está com sufixo "smarty"no artefato de interfaces, o que deveria ocorrer apenas após estressar o smartyProcess.

Os artefatos de saída da atividade anterior, são tomados de entrada pela próxima atividade. Por isso tal artefato tem o sufixo *SMarty*.

- **[E2.C8.13]**: Apenas sugiro que a atividade Modelo de Informação de Interface seja removido de dentro da Especificação de Componente e fique do lado externo (em torno) da atividade em geral.

Tal sugestão foi analisada no momento da formalização do processo, e a decisão de mantê-la dentro foi tomanda com base no fato de que tal artefato é gerado pela tarefa

`Definir Modelo de Informação de Interface`, e esse artefato alimenta outras duas tarefas, ou seja, ele não existe antes dessa tarefa gerá-lo. Por isso esta contido dentro do contexto dessa atividade.

6.7.5 [CA10] Maiores Complexidades do *SMartyComponents*

Essa seção apresenta as maiores complexidades encontradas sugeridas pelos especialistas, e a partir delas as possíveis melhorias, referentes à categoria CA10, com base na questão 7.2:

- De acordo com o especialista E1, a atividade `Definição de Requisitos` demanda mais esforço, devido ao envolvimento de mais papeis e mais decisões não objetivas:
  - [**E1.C10.24**]: Não me lembro o nome, mas é o primeiro diagrama dado o envolvimento de mais pessoas e mais decisões não objetivas.
- De acordo com os especialistas E2, E3 e E8, a atividade `Especificação de Componente` demanda mais esforços, pelo fato de existir atividades anteriores:
  - [**E2.C10.21**]: A Especificação de Componente. Devido ao fato de que compreende as duas atividades anteriores, sintetizando tudo o que já foi especificado. Isso gera uma certa carga cognitiva no Arquiteto de Linha de Produto em todo o processo até o final dessa fase.
  - [**E3.C10.21**]: A Identificação de Componente aparenta demandar maior esforço pelo número de tarefas e artefatos envolvidos. Além disso, demanda mais conhecimento técnico e experiência sobre desenvolvimento baseado em componentes para gerar os artefatos envolvidos.
  - [**E8.C10.21**]: Acredito que a Especificação de Componentes seja a que demanda maior esforço ou complexidade em função da granularidade. Para formar uma base suficiente para uma LPS, é necessário ter um grande número de artefatos e bastante robustos para efetivamente viabilizarem a reutilização e ganho em escala.
- De acordo com o especialista E4, a atividade `Interação de Componente` demanda mais esforços pelo fato dos artefatos gerados no *workflow* anterior, serem referentes ao conhecimento dos processos de negócios:

  - **[E4.C10.25]**: Acredito que a atividade de "Interação de Componente"seja a de maior esforço e complexidade, pois relaciona os artefatos já produzidos com a experiência do arquiteto na descoberta das operações de negócio.

- De acordo com o especialista E5, as tarefas do *SMartyProcess* demandaram maior esforço pelo fato de que a sua realização necessita estar relacionada com as tarefas anteriores (Definição de Requisitos, Identificação de Componente, Interação de Componente e Especificação de Componente):

  - **[E5.C10.20]**: Diria que a atividade recorrente, denominada de "SMarty-Process"demande o maior esforço, ao mesmo tempo que apresenta a maior complexidade, visto que a sua realização precisam ser "casadas"com as tarefas convencionais (proposta pelo UML Components). Digo isso, porque a ação de identificação e representação de variabilidades vai permear todo o ciclo de desenvolvimento (e não ser realizada depois das tarefas convencionais desse ciclo) - por esse fato, ressalto a importância do rastreamento das variabilidades.

- De acordo com os especialistas E6 e E7, a atividade `Identificação de Componente` demandou maior esforço pela quantidade de elementos e conceitos relacionados:

  - **[E6.C10.23]**: A atividade de "Identificação de Componentes"foi que demandou maior esforço pela quantidade de elementos e no entendimento dos conceitos e relacionamentos.
  - **[E7.C10.23]**: Provavelmente a identificação de componentes. Mas acho que isso vai depender do contexto e da qualidade de artefatos e conhecimento dos papeis envolvidos.

Por fim, levando em conta todas as melhorias consideradas, a Figura 6.13 apresenta a visão geral do *SMartyComponents* que posteriomente foi atualizada, enquanto a Figura 5.3 apresenta a visão geral atualizada.

6.8 Considerações Finais

Os resultados obtidos por meio da análise qualitativa com base nas respostas dos especialistas permitiram identificar o nível de compreensão obtido em relação ao processo proposto, e também possíveis melhorias para cobrir inconsistências e redundâncias percebidas no contexto da proposta. Assim, algumas melhorias em relação aos elementos que

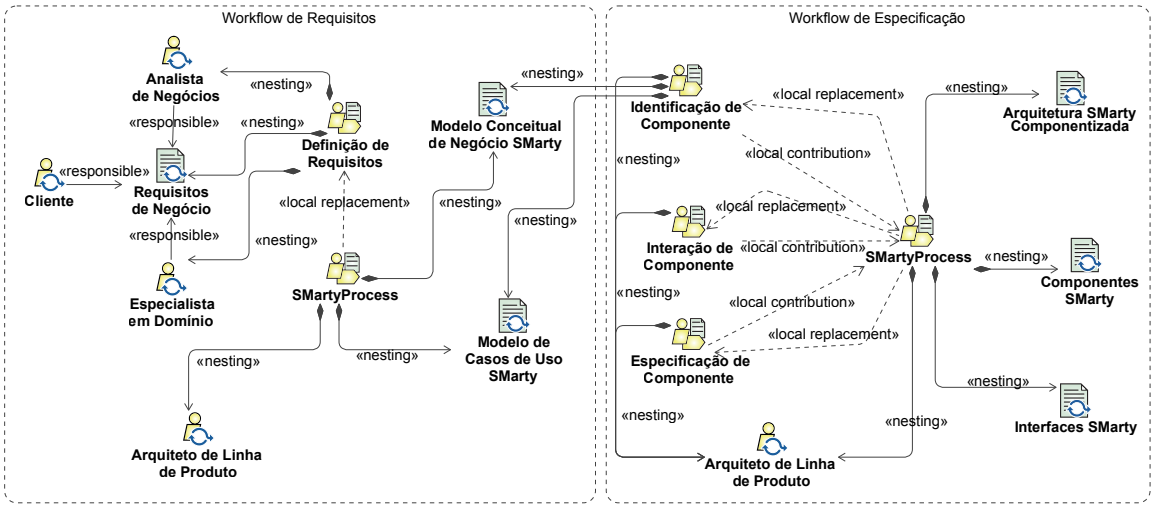

Figura 6.13: Visão Geral do *SMartyComponents* sem as Melhorias Consideradas.

compõem o *SMartyComponents* foram muito importantes para o refinamento e evolução do processo.

Foram sugeridas duas modificações na atividade `Definição de Requisitos` do *Workflow* de Requisitos, onde o Especialista em domínio interage com a tarefa Definição dos Processos de Negócio, e tal tarefa alimenta o fluxo da tarefa Desenvolver Modelo Conceitual de Negócio. Já em todas as atividades do *workflow* de Especificação, um papel Arquiteto de Sotware foi adicionado nas atividades do *UML Components*, e na atividade `Identificação de Componente` as interfaces existentes passaram a compor o artefato `Recursos Existentes`. Já na atividade `Especificação de Componente`, os artefatos `Especificação de Arquitetura` e `Especificação de Componente` foram unificados em um só: `Espec. de Componente & Arquitetura SMarty`.

Por fim, compreende-se que novos estudos devem ser realizados, com o objetivo de verificar os resultados gerados pelo processo proposto em ambientes acadêmicos e industriais. De acordo com os resultados obtidos nesse estudo qualitativo, pode-se pensar em estudos quantitativos, a fim de verificar a sua efetividade, eficiência e eficácia considerando LPSs reais, tomando como base o corpo de conhecimento inicial construído nesse trabalho.

7

Conclusão

"Aprendi que a coragem não é a ausência do medo, mas o triunfo sobre ele."

Nelson Mandela (1918 - 2013), Ex-Presidente da África do Sul

A reutilização de software é um paradigma que vem sendo explorado tanto no meio acadêmico quanto no meio industrial. Minimizar os esforços no desenvolvimento de software pode permitir reduzir os custos de projeto, além de gerar produtos customizáveis e maximizar a qualidade do produto final. Para tanto, algumas abordagens como LPS e DBC podem fornecer mecanismos que visam o reúso de componentes e artefatos.

A abordagem LPS permite que famílias de produto possam ser geradas, proporcionando a reutilização de artefatos e outros benefícios no processo de desenvolvimento, como produtividade no desenvolvimento no que tange à questão de tempo, diminuição de riscos e custos, e também pode proporcionar melhor *time-to-market* e retorno de investimento.

A abordagem DBC também colabora com a produtividade, proporcionando a reutilização dos componentes desenvolvidos, e também por permitir a utilização de componentes de terceiros, viabilizando a sua integração por meio das interfaces.

Portanto, as vantagens e benefícios que as abordagens de LPS e DBC porporcionam, puderam ser exploradas nesta pesquisa, a fim de propor um processo que englobe tais características, com o objetivo de especificar ALPSs componentizadas.

Para contribuir com a proposta desta pesquisa, uma evolução da abordagem *SMarty* foi realizada, com o objetivo de identificar diferentes níveis de representação de variabilidades

em modelos de componentes de acordo com a UML 2.5. Tal evolução foi avaliada experimentalmente, e os resultados obtidos justificam a sua efetividade.

O processo *SMartyComponents* foi definido e posteriormente analisado por meio de um estudo empírico qualitativo. De acordo com as melhorias identificadas, o processo foi revisado e evoluído. Dessa forma, evidências iniciais permitem acreditar que o *SMartyComponents* possa contribuir com a especificação de ALPSs componentizadas. Portanto, espera-se que esta proposta e os resultados dos estudos realizados possam proporcionar evidências incipientes da possibilidade de adoção do processo e futura transferência de tecnologia.

7.1 Contribuições

Esta pesquisa de mestrado teve como contribuição: (i) a evolução da abordagem *SMarty* para componentes, gerando assim a versão 5.2; (ii) um estudo experimental de *SMarty* 5.2 para avaliar a sua efetividade; (iii) a proposta de um processo proveniente da união do método *UML Components* com *SMarty* 5.2, nomeado *SMartyComponents*; e (iv) um estudo empírico qualitativo do *SMartyComponents* com o objetivo de analisar a sua compreensibilidade e a sua evolução. Os itens a seguir descrevem sobre tais contribuições.

Resultados e Contribuições quanto à Evolução da Abordagem *SMarty*:

- a evolução da abordagem *SMarty* 5.1 para modelos de componentes UML foi realizada com base na análise de um subconjunto dos resultados de um Mapeamento Sistemático (Apêndice A). Assim, os trabalhos relevantes foram estudados e as técnicas e representações de variabilidades foram analisadas e atribuídas aos elementos que compõem o modelo de componentes da UML 2.5;

- os elementos que compõem o modelo de componentes da UML 2.5 foram utilizados para a evolução da abordagem *SMarty* 5.2. Assim, os usuários podem se beneficiar por seguir os padrões da UML estabelecidos pela OMG, além de poder utilizar ferramentas distintas de modelagem UML e exportar os resultados em arquivos XMI (*XML Metadata Interchange*), que facilitam a troca de metadados entre ferramentas de modelagem;

- gerenciamento de variabilidades em níveis distindos entre os elementos que compõem o modelo de componentes da UML. Assim, facilita a identificação, representação e delimitação das variabilidades nos níveis de Componentes x Interfaces, Componentes x Portas, Portas x Interfaces e Interfaces x Operações;

- rastreabilidade entre diferentes níveis de elementos do modelo de componentes da UML. Assim, acredita-se que por meio dos meta-atributos *realizes+* e *realizes-*, possa ser possível identificar pontos de variação em diferentes níveis de abstração;

- a avaliação experimental da abordagem *SMarty* 5.2 foi planejada e conduzida. De acordo com os resultados da comparação entre as abordagens *SMarty* e *Ravavian e Khosravi*, *SMarty* foi considerada a princípio mais efetiva. Dessa forma, a evolução proposta para *SMarty* beneficiou os modelos gerados na avaliação, a ponto de apresentar indícios de que a utilização de *SMarty* em modelos de componentes UML é viável; e

- acredita-se que a evolução da abordagem *SMarty* para componentes possa ser utilizada juntamente com outros processos ou abordagens de desenvolvimento baseado em componentes, permitindo assim a identificação e representação de variabilidades em seus artefatos.

Resultados e Contribuições quanto ao Processo *SMartyComponents*:

- *SMartyComponents* foi proposta, gerando assim um processo sistemático para a especificação de ALPSs componentizadas. O *UML Components* foi devidamente analisado e modificações em seus artefatos foram necessárias para proporcionar o processo de especificação de ALPSs. *SMarty* 5.2 serviu de base para gerenciar variabilidades nos modelos UML gerados pelo *SMartyComponents*; e

- A avaliação empírica qualitativa do processo *SMartyComponents* foi planejada e conduzida. O objetivo do estudo foi analisar a compreensibilidade do processo com base em especialistas. Assim, pontos fortes e fracos foram identificados pelos especialistas, e pôde-se identificar possíveis melhorias. Tais melhorias foram levadas em consideração e proporicionaram a evolução do processo.

Publicações dos Resultados e Contribuições:

- BERA, M. H. G. ; OliveiraJr, E. *SMartyComponents*: uma Abordagem para Desenvolvimento de Arquiteturas de Linha de Produto de Software Componentizadas. In: VII Workshop de Teses e Dissertações em Sistemas de Informação, 2014, Londrina-PR. Anais do Simpósio Brasileiro de Sistemas de Informação, 2014. v. 5. p. 22-25.

- BERA, M. H. G. ; OliveiraJr, E. ; COLANZI, T. E. *Evidence-based SMarty Support for Variability Identification and Representation in Component Models.* In Proc.: International Conference on Enterprise Information Systems, 2015, Barcelona. Proceedings of the 17th International Conference on Enterprise Information Systems. Setúbal: SCITEPRESS Science and Technology Publications, 2015. v. 2. p. 295-302.

- BERA, M. H. G. ; OliveiraJr, E. ; COLANZI, T. E. *SMartyComponents*: um Processo para Especificação de Arquiteturas de Linhas de Produtos de Software Componentizadas. In: The Second Latin-American School on Software Engineering (ELA-ES), 2015, Porto Alegre. v. 1. p. 86-89.

- BERA, M. H. G. ; OliveiraJr, E. A Systematic Mapping on Variabilities in Software Architectures and Component-based Architectures. Submetido ao: International Journal on Software and Systems Modeling (SoSyM).

7.2 Limitações

Os itens a seguir apresentam as limitações percebidas nesta pesquisa:

- a pouca quantidade de participantes nos estudos realizados é uma limitação identificada. No estudo quantitativo, a falta de participantes bem qualificados acabou restringindo as amostras. Já no estudo qualitativo, os especialistas com qualificação adequada tinham tempo limitado para participar do estudo de forma efetiva. Assim, entende-se que para reduzir essas limitações, é necessário replicar os estudos com um maior número de participantes e especialistas com disponibilidade adequada;

- a dificuldade de encontrar LPSs reais acaba limitando os estudos à utilização de LPSs pedagógicas e acadêmicas. Assim, acredita-se que uma possível parceria entre a academia e a indústria possa fornecer LPSs reais para buscar corroborar os resultados dos estudos realizados; e

- a abordagem *SMarty* ainda não possui suporte a restrições em OCL, e essa limitação acaba interferindo em algumas tarefas propostas pelo *UML Components* e que fazem parte da atividade de *Especificação de Componente* do *Workflow* de Especificação do *SMartyComponents*. Assim, acredita-se que uma possível evolução de *SMarty* com suporte à OCL, possa contribuir para a definição de restrições mais formais no processo *SMartyComponents*.

7.3 Trabalhos Futuros

Algumas sugestões de trabalhos futuros:

- acredita-se que a evolução da abordagem *SMarty* 5.2 possa ser avaliada em relação a outras abordagens de gerenciamento de variabilidades em componentes, e com um número mais significativo de participantes, com o intuito de corroborar os resultados obtidos;

- o estudo experimental conduzido foi realizado com participantes da academia, mas deve ser considerada a participação de especialistas da indústria, com o objetivo de avaliar o desempenho de sua aplicação *in vivo*. Para tanto, vale a pena considerar a replicação do estudo com LPSs reais, se estas forem disponibilizadas;

- o processo *SMartyComponents* foi proposto e evoluído, porém, por conta de limitações em relação ao tempo da pesquisa, não foi avaliado de forma experimental. Assim, como trabalho futuro, pretende-se realizar estudos quantitativos do *SMartyComponents*, com o intuito de avaliar a sua efetividade em relação a abordagens similares de especificação de ALPSs;

- acredita-se que, após a evolução quantitativa de *SMartyComponents*, possa se pensar em alternativas para modelar arquiteturas de componentes em processos de construção reativo e/ou extrativo de LPS, em relação ao gerenciamento de novas características;

- acredita-se que possa se gerar a implementação de componente ou código, por meio de modelos XMI. Atualmente, há um trabalho em andamento onde algumas anotações são adicionadas ao XMI, chamadas *SMartyAnotations*, baseadas nos estereótipos de *SMarty*;

- acredita-se que de acordo com uma avaliação quantitativa de *SMartyComponents* possa se pensar em reduzir artefatos, tarefas e até mesmo papéis, visando deixar o processo menos complexo;

- o suporte de *SMarty* à OCL é uma das questões que podem ser consideradas, a fim de tornar o *SMartyComponents* mais formal. Acredita-se que dessa forma, possíveis lacunas existentes no *SMartyComponents* possam ser solucionadas; e

- com base no processo proposto fica evidente que algumas atividades podem ser automatizadas. Assim, pretende-se utilizar, por exemplo, transformações de modelos com apoio de linguagens específicas como ATL para automatizar parte das atividades do *SMartyComponents* que se referem à transformação de modelos do *UML Components* para modelos com variabilidades *SMarty*.

REFERÊNCIAS

Abdellatief, M.; Sultan, A. B. M.; Ghani, A. A. A.; Jabar, M. A. A Mapping Study to Investigate Component-based Software System Metrics. *Jornal of System and Software*, v. 86, n. 3, p. 587–603, 2013.

Abdelnabi, Z.; Cantone, G.; Ciolkowski, M.; Rombach, D. Comparing code reading techniques applied to object-oriented software frameworks with regard to effectiveness and defect detection rate. In: *International Symposium on Empirical Software Engineering, 2004.*, 2004, p. 239–248.

Ahmaro, I.; Bin Mohd Yusoff, M.; Mohd Abualkishik, A. The Current Practices of Software Reusability Approaches in Malaysia. In: *Software Engineering Conference (MySEC), 2014 8th Malaysian*, 2014, p. 172–176.

Bailey, J.; Budgen, D.; Turner, M.; Kitchenham, B.; Brereton, P.; Linkman, S. Evidence relating to Object-Oriented software design: A survey. In: *Empirical Software Engineering and Measurement*, 2007, p. 482–484.

Barney, S.; Petersen, K.; Svahnberg, M.; Aurum, A.; Barney, H. Software Quality Trade-Offs: a Systematic Map. *Information and Software Technology*, v. 54, n. 7, p. 651–662, 2012.

Basili, V.; Selby, R. Comparing the Effectiveness of Software Testing Strategies. *IEEE Transactions on Software Engineering*, v. 13, n. 12, p. 1278–1296, 1987.

Basili, V. R.; Caldiera, G.; Rombach, H. D. The Goal Question Metric Approach. In: *Encyclopedia of Software Engineering*, Wiley, 1994.

Bass, L.; Clements, P.; Kazman, R. *Software Architecture in Practice, Second Edition*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

Beel, J.; Gipp, B. Link Analysis in Mind Maps: A New Approach to Determining Document Relatedness. In: *Proceedings of the 4th International Conference on Uniquitous Information Management and Communication*, ICUIMC '10, New York, NY, USA: ACM, 2010, p. 38:1–38:5 (*ICUIMC '10*,).

Brandalise, L. T. Modelos de medição de percepção e comportamento: uma revisão. [*On-line*].
Disponível em `http://www.lgti.ufsc.br/brandalise.pdf`

Broberg, L. L. *A Grounded Theory Approach to Examining Design and Usability Guidelines for Four-year Tribal College Web Sites*. Tese de Doutoramento, aAI3460923, 2011.

Brown, A. W. *Large-Scale, Component Based Development*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000.

Capilla, R.; Bosch, J.; Kang, K.-C. *Systems and Software Variability Management: Concepts, Tools and Experiences*. Springer Berlin Heidelberg, 319 p., 2013.

Cesare, S.; Lycett, M.; Macredie, R. D. *Development of component-based information systems*. Armonk, New York: M.E. Sharpe, 251 p., 2006.

Cheesman, J.; Daniels, J. *UML Components: A Simple Process for Specifying Component-based Software*. 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001.

Chen, L.; Ali Babar, M.; Ali, N. Variability Management in Software Product Lines: A Systematic Review. In: *Proc. International Software Product Line Conference*, Pittsburgh, PA, USA: Carnegie Mellon University, 2009, p. 81–90.

Choi, Y.; Shin, G.; Yang, Y.; Park, C. An approach to extension of UML 2.0 for representing variabilities. In: *Computer and Information Science, 2005. Fourth Annual ACIS International Conference on*, 2005, p. 258–261.

Clements, P.; Northrop, L. *Software Product Lines: Practices and Patterns*. Boston, MA, USA: Addison-Wesley Longman Publishing, 2002.

Contieri Junior, A. C. *Aplicação de Métricas em Arquiteturas de Linhas de Produto de Software*. Trabalho de Conclusão de Curso, Departamento de Informática, Universidade Estadual de Maringá, 2010.

Contieri Junior, A. C.; Correia, G. G.; Colanzi, T. E.; Gimenes, I. M. S.; OliveiraJr, E.; Ferrari, S.; Masiero, P. C.; Garcia, A. F. Extending UML Components to Develop Software Product-Line Architectures: Lessons Learned. In: *Proc. European Conf. on Software Architecture*, Springer Berlin Heidelberg, 2011, p. 130–138 (*Lecture Notes in Computer Science*, v.6903).

Corbin, J. M.; Strauss, A. L. *Basics of qualitative research.* 2 ed. Sage Publ., 2008.

Correia, G. G. *Avaliação de Arquitetura de Linha de Produto de Software por meio de Métricas.* Trabalho de Conclusão de Curso, Universidade Estadual de Maringá, 2010.

Coupaye, T.; Estublier, J. Foundations of enterprise software deployment. In: *Proceedings of the Fourth European Software Maintenance and Reengineering, 2000.*, 2000, p. 65–73.

D'Souza, D. F.; Wills, A. C. *Objects, Components, and Frameworks with UML: The Catalysis Approach.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

Dybå, T.; Prikladnicki, R.; Rönkkö, K.; Seaman, C.; Sillito, J. Qualitative Research in Software Engineering. *Empirical Software Engineering*, v. 16, n. 4, p. 425–429, 2011.

Farooq, S. U.; Quadri, S. M. K. Evaluating effectiveness of software testing techniques with emphasis on enhancing software reliability. In: *Journal of Emerging Trends in Computing and Information Sciences*, 2011, p. 740–745.

Fiori, D. R.; Gimenes, I. M. S.; Maldonado, J. C.; OliveiraJr, E. Variability Management in Software Product Line Activity Diagrams. In: *Proc. International Conference on Distributed Multimedia Systems*, 2012, p. 89–94.

Galster, M.; Weyns, D.; Tofan, D.; Michalik, B.; Avgeriou, P. Variability in Software Systems - A Systematic Literature Review. *IEEE Transactions on Software Engineering*, v. 40, n. 3, p. 282–306, 2013.

Geraldi, R. T.; OliveiraJr, E.; Conte, T.; Steinmacher, I. Checklist-based Inspection of SMarty Variability Models - Proposal and Empirical Feasibility Study. In: *ICEIS 2015 - Proceedings of the 17th International Conference on Enterprise Information Systems, Volume 2, Barcelona, Spain, 27-30 April, 2015*, 2015, p. 268–276.

Gomaa, H. Evolving Software Requirements and Architectures Using Software Product Line Concepts. In: *2nd International Workshop on the Twin Peaks of Requirements and Architecture (TwinPeaks)*, 2013, p. 24–28.

Griss, M. Systematic Software Reuse - It isn't what it used to be! Presented as the 14th International Conference on Software Reuse, Miami, Florida, USA, 2015.

Gross, H.-G. *Component-Based Software Testing with UML.* Dordrecht: Springer, 2005.

Heinsman, D.; Shadish, W. Assignment methods in experimentation: When do nonrandomized experiments approximate answers from randomized experiments? *Psychological Methods*, v. 2, n. 1, p. 154–169, 1996.

Hole, G. J. *The Psychology of Driving.* 1st ed. Psychology Press, 2006.

Höst, M.; Regnell, B.; Wohlin, C. Using Students As Subjects&Mdash;A Comparative Study ofStudents and Professionals in Lead-Time Impact Assessment. *Empirical Softw. Engg.*, v. 5, n. 3, p. 201–214, 2000.

Jacobson, I.; Booch, G.; Rumbaugh, J. *The Unified Software Development Process.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

Jacobson, I.; Griss, M.; Jonsson, P. *Software Reuse: Architecture, Process and Organization for Business Success.* New York, NY, USA: ACM Press/Addison-Wesley Publishing, 1997.

Jazayeri, M.; Ran, A.; van der Linden, F. *Software Architecture for Product Families: Principles and Practice.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000.

Jedlitschka, A.; Ciolkowski, M.; Pfahl, D. Reporting Experiments in Software Engineering. 2007.

Jensen, R. W. *Improving Software Development Productivity: Effective Leadership and Quantitative Methods in Software Management.* Westford, Massachusetts, USA: Pearson Education, Inc., 2015.

Juristo, N.; Moreno, A. M. *Basics of Software Engineering Experimentation.* Madrid: Springer, 367 p., 2010.

Kim, T.; Ko, I. Y.; Kang, S. W.; Lee, D. H. Extending ATAM to assess product line architecture. In: *Computer and Information Technology, 2008. CIT 2008. 8th IEEE International Conference on*, 2008, p. 790–797.

Kitchenham, B. A. Guidelines for Performing Systematic Literature Reviews in Software Engineering. In: *Proc. International Conference on Software Engineering*, New York, USA: ACM Press, 2007, p. 10–51.

Kitchenham, B. A.; Budgen, D.; Pearl Brereton, O. Using Mapping Studies as the Basis for Further Research - a Participant-Observer Case Study. *Information and Software Technology*, v. 53, n. 6, p. 638–651, 2010.

Lewis, G. A.; Poernomo, I.; Hofmeister, C., eds. *Component-Based Software Engineering*, v. 5582 de *Lecture Notes in Computer Science*, East Stroudsburg, PA, USA: Springer, 2009.

Likert, R. A Technique for the Measurement of Attitudes. *Archives of Psychology*, v. 22, n. 140, p. 1–55, 1932.

Linden, F.; Schmid, K.; Rommes, E. *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering.* Springer, 340 p., 2007.

Marcolino, A. *Avaliação Experimental da Abordagem SMarty para Gerenciamento de Variabilidade em Linhas de Produto de Software Baseadas em UML.* Dissertação de mestrado, Universidade Estadual de Maringá (UEM), 2014.

Marcolino, A.; OliveiraJr, E. Avaliação Experimental da Abordagem SMarty para Gerenciamento de Variabilidade em Linhas de Produto de Software Baseadas em UML. In: *Concurso de Teses e Dissertações em Qualidade de Software (CTDQS) - Simpósio Brasileiro de Qualidade de Software (SBQS)*, Manaus, 2015, p. 339–353.

Marcolino, A.; OliveiraJr, E.; Gimenes, I. M. S. Towards the Effectiveness of the SMarty Approach for Variability Management at Sequence Diagram Level. In: *Proc. International Conference on Enterprise Information Systems (ICEIS)*, Lisboa, Portugal, 2014a, p. 249–256.

Marcolino, A.; OliveiraJr, E.; Gimenes, I. M. S.; Barbosa, E. Empirically Based Evolution of a Variability Management Approach at UML Class Level. In: *Proc. International Conference Computers, Software & Applications (COMPSAC)*, Vasteras, Sweden, 2014b, p. 354–363.

Marcolino, A.; OliveiraJr, E.; Gimenes, I. M. S.; Conte, T. U. Towards Validating Complexity-Based Metrics for Software Product Line Architectures. In: *Simpósio Brasileiro de Componentes, Arquiteturas e Reutilização de Software (SBCARS)*, IEEE, 2013a, p. 69–79.

Marcolino, A.; OliveiraJr, E.; Gimenes, I. M. S.; Maldonado, J. Towards the Effectiveness of a Variability Management Approach at Use Case Level. In: *Proc. International Conference on Software Engineering and Knowledge Engineering (SEKE)*, 2013b, p. 214–219.

Martinez-Ruiz, T.; Garcia, F.; Piattini, M.; Münch, J. Modelling Software Process Variability: An Empirical Study. *Software, IET*, v. 5, n. 2, p. 172–187, 2011.

Mendonca, M.G., M. J. O. M. C. J. F. S. S. F. T. G. H. H. E. B. V. A framework for software engineering experimental replications. In: *13th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS), 2008.*, 2008, p. 203–212.

Mohabbati, B.; Asadi, M.; Gašević, D.; Hatala, M.; Müller, H. a. Combining Service-Orientation and Software Product Line Engineering: a Systematic Mapping Study. *Information and Software Technology*, v. 55, n. 11, p. 1845–1859, 2013.

Mota Silveira Neto, P. A.; Carmo Machado, I.; McGregor, J. D.; Almeida, E. S.; Lemos Meira, S. R. A Systematic Mapping Study of Software Product Lines Testing. *Information and Software Technology*, v. 53, n. 5, p. 407–423, 2011.

Neto, A. A.; Conte, T. A conceptual model to address threats to validity in controlled experiments. In: *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*, New York, NY, USA: ACM, 2013, p. 82–85.

OliveiraJr, E.; Gimenes, I. M. S. Empirical Validation of Product-line Architecture Extensibility Metrics. In: *International Conference on Enterprise Information Systems, Volume 2, Lisbon, Portugal, 27-30 April, 2014*, 2014, p. 111–118.

OliveiraJr, E.; Gimenes, I. M. S.; Huzita, E.; Maldonado, J. A Variability Management Process for Software Product Lines. In: *Proc. International Conference of the Centre for Advanced Studies on Collaborative Research (CASCON)*, IBM Press, 2005, p. 225–241.

OliveiraJr, E.; Gimenes, I. M. S.; Maldonado, J. Systematic Management of Variability in UML-based Software Product Lines. *Journal of Universal Computer Science (JUCS)*, v. 16, n. 17, p. 2374–2393, 2010a.

OliveiraJr, E.; Gimenes, I. M. S.; Maldonado, J.; Masiero, P. Systematic Evaluation of Software Product Line Architectures. *Journal of Universal Computer Science (JUCS)*, v. 19, n. 1, p. 25–52, 2013a.

OliveiraJr, E.; Gimenes, I. M. S.; Maldonado, J. C. Systematic Management of Variability in UML-based Software Product Lines. v. 16, n. 17, p. 2374–2393, 2010b.

OliveiraJr, E.; Gimenes, I. M. S.; Maldonado, J. C. A Meta-Process to Support Trade-Off Analysis in Software Product Line Architecture. In: *Proc. International Conference on Software Engineering and Knowledge Engineering (SEKE)*, Knowledge Systems Institute Graduate School, 2011, p. 687–692.

OliveiraJr, E.; Gimenes, I. M. S.; Maldonado, J. C. Empirical Validation of Variability-based Complexity Metrics for Software Product Line Architecture. In: *Proc. International Conference on Software Engineering and Knowledge Engineering (SEKE)*, Knowledge Systems Institute Graduate School, 2012, p. 622–627.

OliveiraJr, E.; Maldonado, J.; Gimenes, I. M. S. Empirical Validation of Complexity and Extensibility Metrics for Software Product Line Architectures. *Proc. of the Brazilian Symposium on Software Components, Architectures and Reuse*, , n. 2, p. 31–40, 2010c.

OliveiraJr, E.; Pazin, M. G.; Gimenes, I. M. S.; Kulesza, U.; Araújo, A. F. SMartySPEM: A SPEM-Based Approach for Variability Management in Software Process Lines. In: *14th International Conference on Product-Focused Software Process Improvement, Paphos, Cyprus, June 12-14, 2013. Proceedings*, 2013b, p. 169–183.

Olumofin, F.; Misic, V. Extending the ATAM Architecture Evaluation to Product Line Architectures. In: *Software Architecture, 2005. WICSA 2005. 5th Working IEEE/IFIP Conference on*, 2005, p. 45–56.

OMG OMG Software & Systems Process Engineering Metamodel Specification (SPEM) - Version 2.0. `http://www.omg.org/spec/SPEM/2.0/`, 2008.

OMG Object Constraint Language - Version 2.4. `http://www.omg.org/spec/OCL/2.4/`, 2015a.

OMG OMG Unified Modeling Language (OMG UML) - Version 2.5. `http://www.omg.org/spec/UML/2.5/`, 2015b.

Petersen, K.; Feldt, R.; Mujtaba, S.; Mattsson, M. Systematic Mapping Studies in Software Engineering. In: *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, Swinton, UK: British Computer Society, 2008, p. 68–77.

Pohl, K.; Böckle, G.; Linden, F. *Software Product Line Engineering: Foundations, Principles, and Techniques.* Springer, 2005.

Pontes, A. C. F.; Pontes Junior, A. C. F.; Braga, A. S. Ensino de Correlação de Postos no Ensino Médio. *Proc. Simpósio Nacional de Probabilidade e Estatística*, 2010.

Pressman, R. S. *Software Engineering - A Practitioner's Approach.* 7th ed. McGraw-Hill, 930 p., 2010.

Razavian, M.; Khosravi, R. Modeling Variability in the Ccomponent and Connector View of Architecture Using UML. In: *IEEE/ACS International Conference on Computer Systems and Applications, 2008.*, 2008, p. 801–809.

Ryu, D.; Lee, D.; Baik, J. Designing an Architecture of SNS Platform by Applying a Product Line Engineering Approach. In: *IEEE/ACIS 11th International Conference on Computer and Information Science (ICIS), 2012*, 2012, p. 559–564.

Santos, J.; Moreira, A.; Araújo, J.; Amaral, V.; Alférez, M.; Kulesza, U. Generating Requirements Analysis Models from Textual Requirements. In: *Proc. International Workshop on Managing Requirements Knowledge*, IEEE, 2008, p. 32–41.

Satyananda, T. K.; Lee, D.; Kang, S.; Hashmi, S. I. Identifying Traceability Between Feature Model and Software Architecture in Software Product Line Using Formal Concept Analysis. In: *Proceedings of the The 2007 International Conference Computational Science and Its Applications*, Washington, DC, USA: IEEE Computer Society, 2007, p. 380–388.

Seaman, C. Qualitative Methods in Empirical Studies of Software Engineering. *IEEE Transactions on Software Engineering (TSE)*, v. 25, n. 4, p. 557–572, 1999.

SEI *Software Engineering Institute Arcade Game Maker (AGM) Pedagogical Product Line*. Carnegie Mellon University, 2009.
Disponível em `http://www.sei.cmu.edu/productlines/ppl/`

SEI *Software Engineering Institute A Framework for Software Product Line Practice - Version 5.0*. Carnegie Mellon University, 2012.
Disponível em `http://www.sei.cmu.edu/productlines/frame_report/index.html`

Shull, F., M. M. G. B. V. C. J. M. J. F. S. T. G. H. F. M. C. Knowledge-sharing issues in experimental software engineering. *Empirical Software Engineering*, v. 9, n. 1-2, p. 111–137, 2004.

van Solingen, R.; Basili, V.; Caldiera, G.; Rombach, H. D. *Goal Question Metric (GQM) Approach* John Wiley and Sons, Inc., 2002.

Steinmacher, I.; Wiese, I. S.; Conte, T.; Gerosa, M. A.; Redmiles, D. The hard life of open source software project newcomers. In: *Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering*, New York, NY, USA: ACM, 2014, p. 72–78.

Subedha, V.; Mohanaprakash, T. A.; Venkataraman; Brindhadevi, S. A decision support system through conceptual reference framework for reusable and functional verification environment. *Journal of Applied Sciences and Engineering Research*, v. 4, n. 2, p. 190–200, 2015.

Szyperski, C. *Component Software: Beyond Object-oriented Programming*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1998.

Taylor, R. N.; Medvidovic, N.; Dashofy, E. M. *Software Architecture: Foundations, Theory, and Practice*. Wiley Publishing, 2009.

Tekinerdogan, B.; Sözer, H. Variability Viewpoint for Introducing Variability in Software Architecture Viewpoints. In: *Proceedings of the WICSA/ECSA 2012*, New York, NY, USA: ACM, 2012, p. 163–166.

Tekumalla, B. *Status of Empirical Research in Component Based Software Engineering - A Systematic Literature Review of empirical studies*. Dissertação de Mestrado, University of Gothenburg, 2012.

Thurimella, A. K.; Bruegge, B. Issue-Based Variability Management. *Information and Software Technology (IST)*, v. 54, n. 9, p. 933–950, 2012.

Tofan, D.; Galster, M.; Avgeriou, P.; Schuitema, W. Past and Future of Software Architectural Decisions - A Systematic Mapping Study. *Information and Software Technology*, v. 56, n. 8, p. 850 – 872, 2014.

Wieringa, R.; Maiden, N.; Mead, N.; Rolland, C. Requirements Engineering Paper Classification and Evaluation Criteria: A Proposal and a Discussion. *Requirements Engineering*, v. 11, n. 1, p. 102–107, 2005.

Wilkinson, L. Statistical methods in psychology journals: guidelines and explanations. *American Psychologist*, v. 54, p. 594–604, 1999.

Wohlin, C.; Runeson, P.; Host, M.; Ohlsson, M. C.; Regnell, B.; Wesslén, A. *Experimentation in Software Engineering*, v. 44. Springer, 248 p., 2012.

Xavier, C. Q. *Análise de Estabilidade de Diferentes Versões de Arquitetura de Linha de Produto de Software*. Trabalho de Conclusão de Curso, Departamento de Informática, Universidade Estadual de Maringá, 2011.

Young, T. J. *Using AspectJ to Build a Software Product Line for Mobile Devices*. Dissertação de mestrado, University of British Columbia, 2005.

Çöteli, M. K. *Testing Effectiveness and Effort in Software Product Lines*. Dissertação de Mestrado, Middle East Technical University, 2013.

Apêndice A - Mapeamento Sistemático sobre Variabilidade em Arquiteturas de Software

A Arquitetura de Software de um programa ou sistema de computação é a estrutura ou estruturas do sistema, que incluem elementos de software, as propriedades externamente visíveis desses elementos e as relações entre eles. Propriedades "externamente visíveis"são suposições que um elemento pode fazer de outros elementos, como seus serviços prestados, características de desempenho, gerenciamento de falhas, o uso de recursos compartilhados, e assim por diante (Bass *et al.*, 2003).

Tais propriedades "externamente visíveis"são entendidas como unidades de software, que possuem uma identidade seja em tempo de implementação ou em tempo de execução, e são denominadas pelo termo componentes. Os componentes podem ser desenvolvidos ou adquiridos como uma unidade. Componentes possuem intencionalmente e explicitamente interfaces, que intermediam o comportamento dos elementos, como parte da arquitetura (SEI, 2012). A arquitetura de um software é o resultado de todas as decisões de projetos tomadas no desenvolvimento e em posteriores evoluções da arquitetura (Taylor *et al.*, 2009).

O Desenvolvimento Baseado em Componentes (DBC) proporciona várias vantagens, por exemplo, promover a reutilização de componentes semelhantes em projetos distintos, sem precisar gastar recursos refazendo o que já existe, diminuindo o retrabalho, reduzindo os custos com desenvolvimento e, consequentemente, minimizando custos com manutenção.

Um projeto que contém vários produtos, com partes similares e partes diferentes, permite unir processos de DBC com a abordagem de Linha de Produto de Software (LPS), assim, promovendo uma arquitetura bem definida e documentada, onde se tornam explicitas as variabilidades e similaridades contidas em cada produto a ser gerado.

Pelo fato dos componentes poderem ser reutilizados em outras aplicações, surge a necessidade de tornar explicitas as suas variabilidades. Entretanto, este é o maior desafio. Uma vez que os componentes encapsulam todas as classes contidas em si, o desafio surge em representar o tipo de variabilidades contida em tal componente, visto que se aplicam a algumas classes específicas. Quando representada diretamente nos componentes, sem levar em conta suas realizações internas, essas podem comprometer toda a aplicação. Desta forma, representar variabilidades em arquiteturas de software e componentes se torna um grande desafio.

Este apêndice, tem por objetivo apresentar a realização de uma busca exploratória por estudos e métodos, que vem sendo utilizadas tanto na indústria de software como também na academia, que tenham como o foco representar variabilidades em arquiteturas de software e arquiteturas baseadas em componentes. Como resultado final deste Mapeamento Sistemático (MS), 39 trabalhos foram selecionados contendo uma representatividade significativa quanto ao foco de busca estabelecido. Tais trabalhos são discutidos, e seus resutados são apresentados em gráficos para melhor compreensão de um todo. Na Seção A.1, está contida a metodologia de pesquisa utilizada no MS, bem como a questão de pesquisa e as categorias de classificação. A Seção A.2 apresenta o processo utilizado para a busca e seleção dos estudos, bem como a definição da questão de pesquisa; o processo de busca; a definição das bases de dados digitais; palavras-chaves e a *string* de busca; a definição dos critérios de inclusão e exclusão; e o esquema de classificação dos estudos recuperados. A Seção A.3 apresenta uma discussão referente aos resultados recuperados no MS, bem como representações gráficas dos dados extraídos. Por fim, a Seção A.4 apresenta as ameaças a validade identificadas neste estudo. As referências sobre cada artigo da seleção final deste MS, estão contidas na Tabela 1.4 da Seção A.3.3.

A.1 Background do Estudo

Um Mapeamento Sistemático (MS) se baseia em evidências de estudos secundários, que oferecem uma visão abrangente de uma área de pesquisa (Tofan *et al.*, 2014). O MS recupera informações a cerca dos trabalhos existentes em determinada área, e as classifica de diversas formas, por exemplo, por fontes de busca, locais de publicação dos trabalhos, classificação de trabalhos encontrados por ano, classificação dos trabalhos por foco de pesquisa, e assim por diante. Os estudos dirigidos a MS, oferecem vários benefícios como por exemplo, reduzir o esforço em retrabalho sobre temas que podem já terem sido discutidos anteriormente, tornando produtiva a pesquisa e oferecer apoio maior aos interessados pelo assunto abordado em si.

O método de busca utilizado neste MS foi aplicado em 5 fases, sendo que na primeira, foi identificado a problemática da pesquisa em si e quais seriam as possíveis soluções. Diante desta problemática, na segunda fase, foi realizada a busca por trabalhos, nas seguintes fontes de busca e bancos de dados científicos: *IEEE*, *ACM*, *ScienceDirect*, *Google Scholar* e *Springer*. Na terceira etapa, todos os artigos recuperados foram examinados, através da leitura do título e *abstract*, e assim os artigos interessantes foram separados para leitura na integra. Na quarta fase, os artigos foram lidos na integra, e separados pelos metodos aplicados nos critérios de busca. Por fim, na quinta e última fase, os artigos foram analisados e seus dados foram criteriosamente separados. Na Seção A.3, as análises serão apresentadas para cada estudo selecionado.

A.2 O Processo do Mapeamento Sistemático

O MS realizado neste estudo tem por objetivo identificar estudos na literatura que abordam a variabilidades em arquiteturas de software e componentes. MSs são pesquisas conduzidas por meio de um grande número de estudos primários.

Segundo Kitchenham (2007) o objetivo do MS é oferecer ao pesquisador uma visão ampla dos estudos primários de uma área de pesquisa como um todo. Sendo assim, uma pesquisa realizada no formato de MS trás uma grande quantidade de estudos retornados, pois não contém restrições com relação ao número de trabalhos obtidos, apenas se o pesquisador preferir aplicar uma filtragem dos estudos que acredita ser mais interessantes e relevantes em determinado momento em sua pesquisa. Desta forma, os critérios para apresentação dos resultados não necessitam ser precisos aos detalhes e elucidados, podendo ser divulgados de modo mais limitado e simples.

Entretanto, Petersen *et al.* (2008), com base na comparação de métodos utilizados em MS e revisões sistemáticas, estabeleceram um conjunto de *guidelines* para um MS. Assim, cinco estágios foram estabelecidos para a elaboração correta de um MS:

1. Definição do protocolo;
2. Definição das perguntas de pesquisa;
3. Condução da pesquisa para estudos primários;
4. Elaboração e aplicação de critérios de inclusão/exclusão nos estudos primários;
5. Classificação dos estudos; e
6. Estratégia de extração e agregação dos dados.

Os estágios da condunção deste MS estão de acordo com a Figura 1.1. Assim, esta Seção descreve como cada estágio foi planejado e conduzido, tal como: a definição da questão de pesquisa (Seção A.2.1); o processo de busca (Seção A.2.2); bases de dados digitais, palavras-chave e strings de busca (Seção A.2.3); critérios de inclusão e exclusão (Seção A.2.4); esquema de classificação (Seção A.2.5) e extração de dados e agregação (Seção A.2.6).

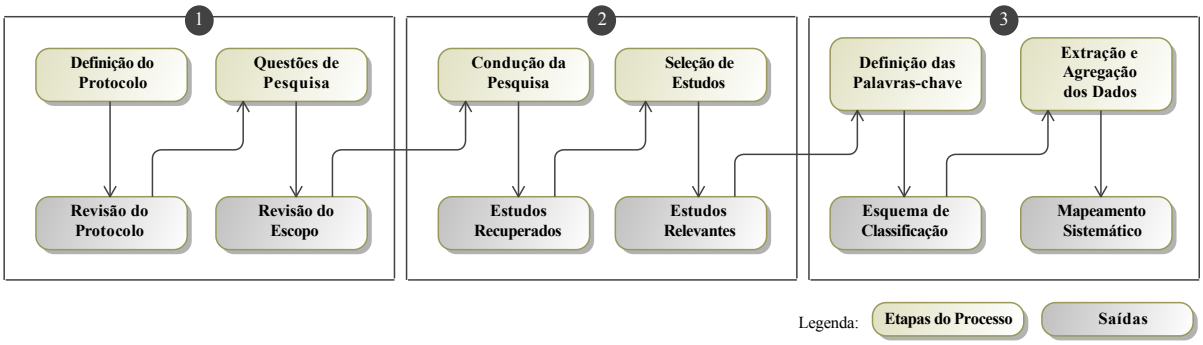

Figura 1.1: O Processo de Mapeamento Sistemático Seguido - Adaptado de Petersen *et al.* (2008)

A.2.1 Definição da Questão de Pesquisa

Com a intenção de buscar por técnicas e métodos que representam explicitamente variabilidade em arquitetura de software e arquitetura baseada em componentes, alguns objetivos de pesquisa foram utilizados como guia para a elaboração deste MS. Primeiramente, encontrar estudos que apresentam métodos que identificam, representam, ou rastream variabilidades em arquiteturas de softwares e componentes, permitindo assim identificar as técnicas utilizadas ou dificuldades que tem sido encontrada dentre os diversos pesquisadores da área. Em seguida, apresentar uma análise sobre os estudos recuperados, visando expor as soluções distintas de acordo com os níveis de representação de cada pesquisa. Diante destes objetivos, surge a formulação da questão de pesquisa a seguir:

- **[QP.1]:** Quais técnicas têm sido propostas e utilizadas para representar variabilidades em arquiteturas de software e componentes?

A importância de obter resultados diante desta questão de pesquisa, é preencher uma lacuna existente na engenharia de software, se tratando de representar variabilidades nos níveis de arquitetura e componentes. Identificar os trabalhos relacionados a este assunto, trarão uma gama de resultados possíveis, disponíveis para preencher essa lacuna

da forma que melhor se aplica ao domínio necessário. Estes trabalhos serão classificados em um contexto de variabilidade e também em nível de variabilidade em que tais técnicas encontradas se aplicam.

A.2.2 O Processo de Busca

O processo de busca deste trabalho está baseado nos critérios e diretrizes propostos por Kitchenham (2007), Kitchenham *et al.* (2010) e Petersen *et al.* (2008), com relação a performace do MSs. Sendo assim, a Figura Figura 1.2 apresenta as etapas do processo de busca, conforme segue:

- Seleção das bases de dados digitais mais relevantes e mecanismos de busca indexáveis (Seção A.2.3);
- Definição das palavras-chaves mais relevantes para compor a string de busca aplica às bases de dados digitais e mecanismos de busca para recuperar os estudos primários. As seguintes palavras chaves foram utilizadas: “variability”, “software architecture”, and “component-based architecture”(Seção A.2.3);
- Compor as palavras chaves e suas variações utilizando “AND”and “OR”(Seção A.2.3);
- Aplicação das strings de busca definidas nas bases de dados digitais e mecanismos de busca. Uma lista de estudos primários será recuperada. Os critérios de inclusão e exclusão serão aplicados para filtrar e selecionar os estudos mais relevantes para este MS. Estudos duplicados e com potenciais conflítos, em seguida, serão reanalisados para atualizar a lista (Seção A.2.4);
- Definição do esquema de classificação baseado nas categorias (Seção A.2.5); e
- Extração e agregação dos dados (Seção A.2.6) por meio das técnicas de visualizações graficas (*graphs*, *bubbles plots*, etc) de acordo com os presentes resultados obtidos. Assim, um *briefing* de discussões é projetado nos sujeitos relacionados deste MS (Seção A.3).

A.2.3 Definição de Bases de Dados Digitais, Palavras-chaves e Strings de Busca

Para a recuperação dos estudos primários foi definida uma estratégia de busca de acordo com as fontes de pesquisa, o idioma dos trabalhos, os tipos de documentos e o ano

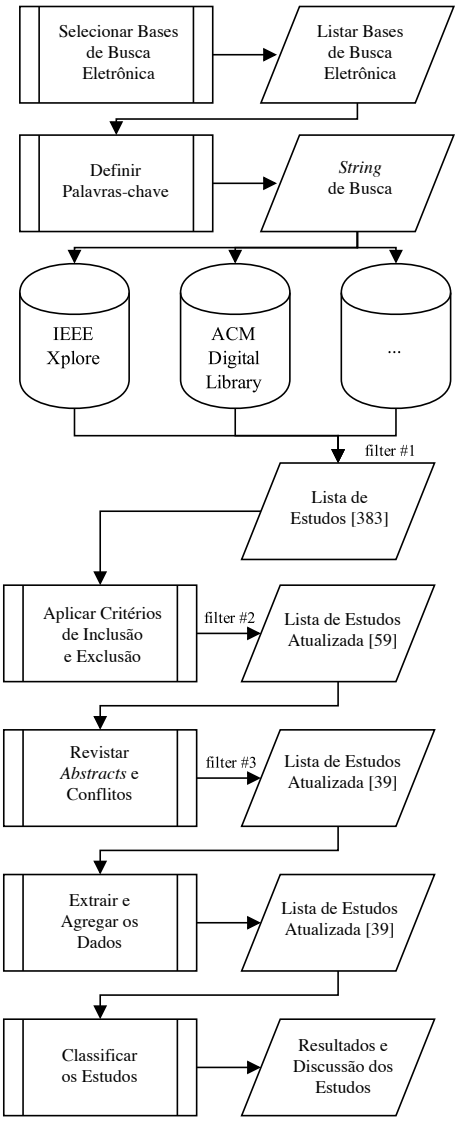

Figura 1.2: O Processo de Busca - Adaptado de Barney *et al.* (2012).

Tabela 1.1: Número de Estudos por Filtro e Bases de Busca.

Base de Busca	**Filtro #1**	**Filtro #2**	**Filtro #3**
ACM Digital Library	23	5	4
ELSEVIER ScienceDirect	9	4	4
Google Scholar	103	20	5
Springer	73	0	0
IEEE Xplore	175	30	26
Total	**383**	**59**	**39**

de publicação. Além disso, foram definidas as palavras-chave, a *string* de busca e as respectivas sequências de consulta. Para as fontes de pesquisa, bases de dados eletrônicas indexadas e os mecanismos de busca eletrônicos foram definidas, como seguem:

- **IEEE Xplore** - Conteúdos científicos e técnicas publicadas pelo Instituto de Engenheiros Elétricos e Eletrônicos (IEEE)e seus parceiros de publicação;
- **ACM Digital Library** - Conteúdos de científico de artigos publicados pela ACM e citações bibliográficas de grandes editoras em computação;
- **Elsevier - ScienceDirect** - Trabalhos científicos revisados por pares, técnicas e conteúdos medicos;
- **Google Scholar** - grande mecanisco de busca por trabalhos científicos indexados no google; e
- **Springer** - Trabalhos científicos publicados pela Springer e livros.

O próximo passo foi definir a string de busca, por meio das palavras chaves "*variability*", "*software architecture*", e "*component-based architecture*". A idéia central deste MS é de buscar trabalhos que apresentam as técnicas utilizadas pelos grandes pesquisadores sobre como eles lidam com variabilidades em níveis de arquitetura e componentes. Assim, de forma bem genérica a **string** de busca e sequência de consulta se basearam na combinação das palavras-chave utilizando os operadores lógicos "*AND*"e "*OR*"conforme apresentado na Tabela Tabela 1.2:

Tabela 1.2: *String* de busca e sequências de consulta.

String de Busca (SB)
(‘‘variability’’ AND (‘‘software architecture’’ OR ‘‘component-based architecture’’)

A.2.4 Definição de Critérios de Inclusão e Exclusão

Nesta etapa do MS foi definido os critérios de inclusão e exclusão dos estudos primários, a fim de refinar o MS e apresentar os trabalhos mais relevantes. Os critérios de inclusão e exclusão são apresentados como segue:

- ***critério de inclusão***. De acordo com a questão de pesquisa **QP.1**, foi definido a inclusão dos trabalho todos os estudos que apresentam conceitos e técnicas para definição de variabilidades em arquiteturas de software e componentes.

- ***critério de exclusão***. de acordo com a questão de pesquisa **QP.1**, foi definido como questão primária os estudos que não tratam de variabilidades em arquiteturas de software ou componentes. Além disso, independente da questão de pesquisa, deve-se considerar também os seguintes critérios de exclusão:

  i Estudos que não estejam publicados em língua inglesa;

  ii Estudos recuperados de meios eletrônicos que não estejam no formato PDF (*Portable Document Format*), DOC (Processador de Texto *Microsoft Word*) ou ODT (Processador de Texto do *Open Office*), sendo esses os meios mais comuns de divulgação de estudos;

  iii Estudos duplicados, encontrados anteriormente em outra(s) fonte(s);

  iv Estudos que não puderam ser recuperados (não disponíveis); e

  v Estudos com menos de 4 páginas.

A *string* de busca aplicada aos bancos de dados digitais retornaram os estudos primários (tal seleção foi classificada como sendo o *filtro #1*), conforme apresentado na Seção A.2.3. Em seguida, uma seleção preliminar foi realizada por meio da leitura do título e resumo dos estudos recuperados, bem como a aplicação dos critérios de inclusão e exclusão (tal seleção foi classificada com sendo o *filtro #2*). Por fim, os estudos selecionados foram lidos de forma integra e os mais relevantes foram selecionados (tal seleção foi classificada como sendo o *filtro #3*), e serão apresentados na Seção A.3. A Figura 1.3 apresenta esta sequência de atividades.

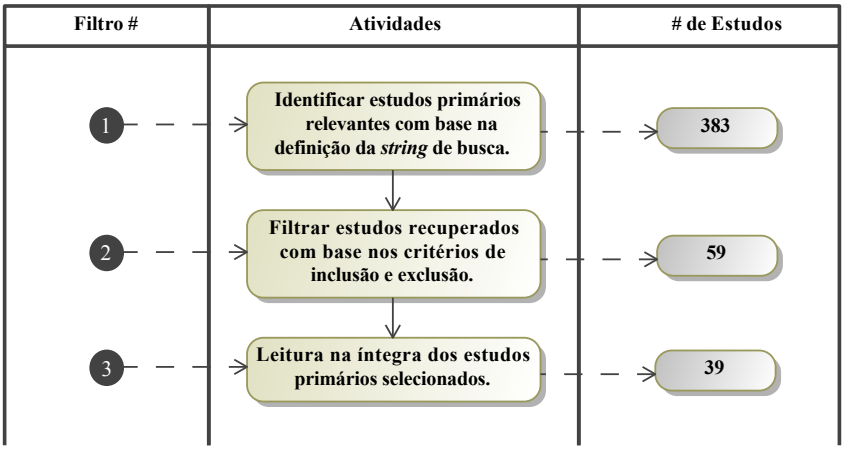

Figura 1.3: Estágios da Busca por Estudos Primários - Adaptado de Mohabbati *et al.* (2013); Mota Silveira Neto *et al.* (2011).

A.2.5 Esquema de Classificação

Para classificar o tipo de pesquisa recuperada em um MS, Kitchenham *et al.* (2010) recomendam a aplicação do método de classificação desenvolvido por Wieringa *et al.* (2005) por meio da utilização de seis categorias de classificação:

1. **Pesquisa de Validação:** tem o interesse de avaliar novas técnicas que não foram aplicadas na indústria. Normalmente realizada no meio acadêmico. Neste estágio, os métodos utilizados para validar a pesquisa são: experimentos, simulações, construção de protótipos, análises matemáticas, etc;

2. **Pesquisa de Avaliação:** tem o interesse de avaliar a pesquisa, o problema de pesquisa ou a implementação de alguma técnica na prática. Normalmente realizada na prática e na indústria;

3. **Proposta de Solução:** o interesse desta classificação está em discutir novas técnicas propostas ou revisadas para determinado problema de pesquisa;

4. **Estudos Filosóficos:** nesta classificação, a preocupação está em apresentar novos caminhos para pesquisas, um novo framework conceitual, etc;

5. **Estudos de Opinião:** contém a opinião do autor sobre algo, o qual deduz que pode estar certo ou errado;

6. **Estudos de Experiência:** discute sobre determinado assunto com base na pesquisa realizada pelo autor do estudo, no qual o autor expõe sua experiência pessoal do estudo realizado na prática, as lições aprendidas com a pesquisa, etc.

Enfim, a extração dos dados do MS visa classificar e categorizar os estudos de forma ampla, no intuito permitir a construção de um protocolo menor contendo todas as informações simplificadas de maneira abrangente, segundo Bailey *et al.* (2007).

Neste estágio, todos os estudos recuperados no MS foram classificados conforme o método de classificação proposto por Wieringa et al. (Wieringa *et al.*, 2005). Para isto, os resumos (*Abstracts*) de todos os estudos recuperados no MS foram lidos, no intuito de identificar a categoria na qual melhor se adequavam. Além disso, as palavras-chave, a conclusão e alguns trechos de cada estudo foram lidos brevemente, a fim de certificar a categoria de classificação estipulada para cada trabalho. Adicionalmente, foi atribuído para cada estudo, conforme o nome do tipo de publicação (conferências, *journals*, *workshops*, etc.). A Figura 1.4 apresenta tal esquema de classificação.

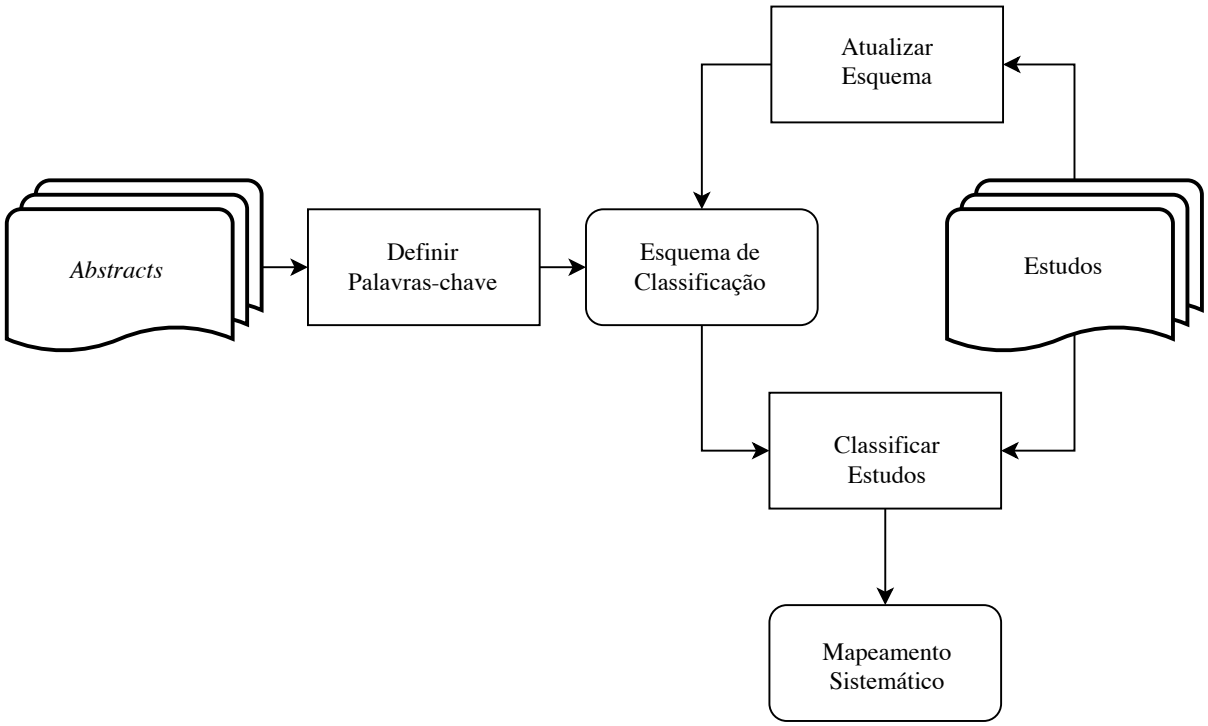

Figura 1.4: Esquema de Classificação - Adaptado de Petersen *et al.* (2008).

Finalmente, o contexto dos estudos finais selecionados para a leitura na íntegra foi escrito de forma resumida, com a principal finalidade de compreender o assunto de cada trabalho e a respectiva discussão dos mesmos, a fim de auxiliar no desenvolvimento desta pesquisa. A Seção A.3.4 apresenta o escopo de cada estudo lido integralmente.

A.2.6 Extração de Dados e Agregação

Durante a seleção dos estudos primários mais relevantes para a leitura na íntegra no processo de seleção final, algumas informações consideradas relevantes foram extraídas de cada um deles, sendo elas:

a **Fonte de Busca**: As bases de dados digitais *IEEE Xplore*, *ACM Digital Library*, *Elsevier ScienceDirect*, *Springer* e o mecanismo de busca eletrônica *Google Scholar*;

b **Título do Documento**: O título integralmente como apresentado nos trabalhos;

c **Autores**: A Figura 1.9, a Figura 1.10 e a Figura 1.11 apresenta os autores com mais publicações encontradas neste MS;

d **Ano de Publicação**: contempla todas publicações a partir do ano de 2005 à 2014;

e **Tipo de publicação**: Conferência, *Journal*, *Workshop*, *Book*, *Thecnical Report*, Dissertação de Mestrado, Tese de Doutorado;

f **Local de publicação**: A Figura 1.6 apresenta os locais de publicação encontrados neste MS;

g **Categorias de Classificação**: Estipuladas na Seção A.2.5;

h **Contexto de Variabilidade**: Apresenta o contexto em que suporta variabilidades, sendo eles: Implementação de Variabilidade; Variabilidade em Requisitos; Verificação e Validação; Variabilidade em Arquitetura; Rastreabilidade em Arquitetura; e Gestão de Variabilidade; e

i **Nível de Variabilidade**: Apresenta em que nível se encontra as variabilidades, sendo eles "Connectores, Portas e Interfaces"; "Modelo de Características"; "Componentes e Interfaces"; "Componentes, Conectores e Interfaces"; "Componentes"; "Código"; "Componentes e Conectores"; "Interfaces"; e "Indefinido"(Para os estudos onde não foram identificadas as definições de níveis de variabilidades).

A seleção dos estudos primários mais relevantes nos deram uma abstração de todos os trabalhos, permitindo a extração dos dados específicos para representarmos em formas de gráficos os dados extraidos, a fim de melhorar a compreensão sobre o resultado do processo de seleção final do MS.

A.3 Discussão dos Resultados do Mapeamento Sistemático

Esta Seção apresenta uma discussão sobre os resultados obtidos neste MS. Assim, a Seção A.3.1 apresenta uma visão geral dos estudos recuperados com base nos resultados da aplicação do filtro #1 (Figura 1.3), a Seção A.3.2 apresenta representações gráficas dos estudos com base no filtro #2; e as Seções A.3.3 e A.3.4 apresentam a seleção e discussão dos estudos mais relevantes com base no filtro #3.

A.3.1 Visão Geral do Mapeamento Sistemático

Este MS foi conduzido entre os meses Janeiro e Março de 2014. Um total de 383 estudos foram recuperados por meio da *string* de busca aplicada as bases de dados e mecanismos de busca. Na base de dados *ACM Digital Library*, 6% dos trabalhos foram recuperados, totalizando 23 documentos retornados. No mecanismo de busca *Google*

Scholar, aproximadamente 13.000 resultados foram retornados, desta forma, optamos por selecionar os estudos disponíveis até a pagina 10 dos resultados, totalizando assim 19% dos trabalhos totais recuperados, em um total de 73 documentos retornados. Na base de dados *IEEE Xplore*, 46% dos totais de trabalhos foram recuperados, ou seja, 175 documentos foram retornados. No mecanismo de busca *ScienceDirect*, 2% de todos os trabalhos foram recuperados, totalizando 9 documentos retornados. Por fim, a *Springer* retornou 27% de todos os estudos recuperados, totalizando 103 documentos retornados. A Figura 1.5 apresenta a quantidade total de estudos primários recuperados em porcentagem de acordo com o mecanismo de busca utilizado.

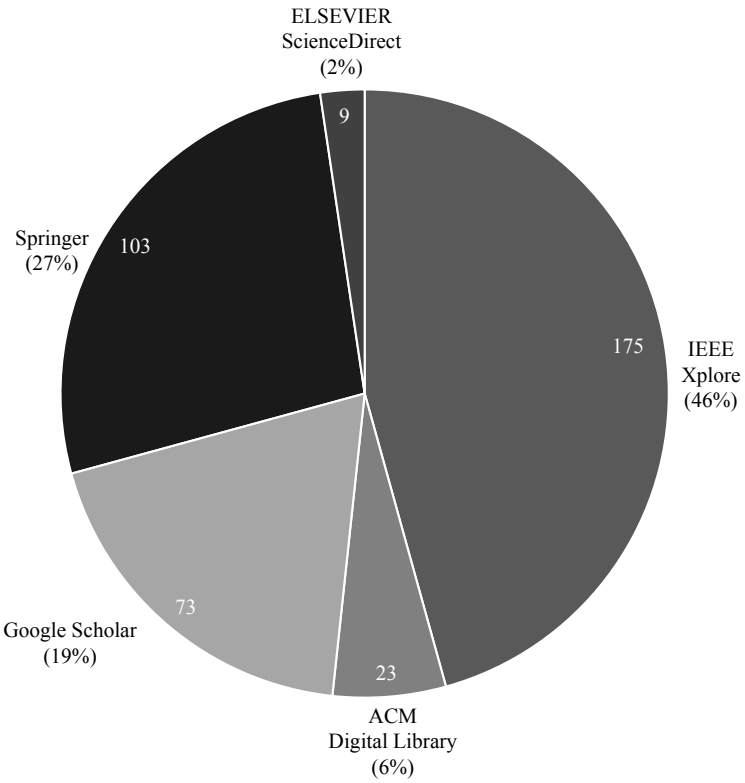

Figura 1.5: Estudos Primários Obtidos por Fonte de Busca.

A.3.2 Metadados dos Estudos Primários

Esta seção apresenta uma discussão dos resultados dos estudos obtidos baseados no filtro #2 para os 59 estudos selecionados.

Locais de Publicação

Os estudos recuperados neste MS apresentam soluções para gerenciar e representar variabilidades em arquiteturas de software e componentes, e foram publicados em 31 locais diferentes. A Figura 1.6 apresenta um gráfico *word cloud* de acordo com todos os locais de publicação recuperados por meio do filtro #2. Os locais com maiores publicações foram o *Working IEEE/IFIP Conference on Software Architecture* (WICSA) com 5 publicações encontradas, *European Conference on Software Architecture* (ECSA) com 4 publicações encontradas e o WICSA/ECSA com 6 publicações encontradas.

Figura 1.6: *Word Cloud* da Distribuição dos Estudos por Locais de Publicação

A Tabela 1.3 apresenta todos os locais de publicação obtidos por meio do filtro #2, com suas respectivas siglas, nomes, tipos e quantidade de estudos.

A Figura 1.7 apresenta os tipos de publicações encontrados, de acordo com a coluna *Type* da Tabela 1.3. Este gráfico apresenta 1 estudo publicado em livro, 41 estudos publicados em conferências, 4 estudos publicados em *Journals*, 4 estudos publicados em *Symposium* e 9 estudos publicados em *Workshops*.

Anos de Publicação e média de estudos dos autores

A Figura 1.8 apresenta os 59 estudos recuperados de acordo com o filtro #2, relacionados com os respectivos anos de publicação. Desta forma, gera uma extração das pesquisas realizadas desde o ano de 2005 até Fevereiro de 2014, na qual foi o mês em que ocorreu esta pesquisa exploratória.

Tabela 1.3: Locais de Publicação por meio do Filtro #2.

Sigla	Nome	Tipo	Fiiltro #2
AICCSA	The ACS/IEEE International Conference on Computer Systems and Applications	Conference	2
CIT	Crisis Intervention Team	Conference	3
ECSA/TDSA	Workshop on Traceability, Dependencies and Software Architecture (on ECSA Conference)	Workshop	1
SIGSOFT	Special Interest Group on Software Engeneering	Symposium	1
TwinPeaks	Twin Peaks of Requirements and Architecture	Workshop	1
HICSS	Hawaii International Conference on System Sciences	Conference	1
ICIS	International Conference on Information Systems	Conference	3
COMPSAC	Computer Software & Applications Conference	Conference	1
WEH	Workshop on Exception Handling	Workshop	3
WICSA/ECSA	Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture—	Conference	6
URKE	Uncertainty Reasoning and Knowledge Engineering	Conference	1
ACIS	Australasian Conference on Information Systems	Conference	1
VaMoS	International Workshop on Variability Modelling of Software-intensive Systems	Workshop	2
SCCC	Conference of the Chilean Society of Computer Science	Conference	1
WCRE	Working Conference on Reverse Engineering	Conference	1
IASTED	International Association of Science and Technology for Development	Conference	1
MOMPES	Model-based Methodologies for Pervasive and Embedded Software	Workshop	1
ICSEA	International Conference on Software Engineering Advances	Conference	2
ICCSA	International Conference on Computational Science and Its Applications	Conference	1
VARSA	Variability in Software Architecture	Workshop	1
JSS	The Journal of Systems and Software	Journal	3
WICSA	Working IEEE/IFIP Conference on Software Architecture	Conference	5
SPLC	International Software Product Line Conference	Conference	1
IST	Information and Software Technology	Journal	1
ISISE	Information Science and Engineering	Symposium	1
CiSE	Conference on Computational Intelligence and Software Engineering	Conference	1
PDCAT	Conference on Parallel and Distributed Computing, Applications and Technologies	Conference	1
ECSA	European Conference on Software Architecture	Conference	4
ICSE	International Conference on Software Engineering	Conference	1
ASA	Agile Software Architecture	Book	1
SPLINE	Software Product Line Conference	Conference	1
SERA	Software Engineering Research, Management and Applications	Conference	1

Considerando a Figura 1.8 é perceptível que as publicações referentes ao tema buscado neste MS oscilaram nos últimos anos. Teve uma crescente nos anos de 2007 e 2008, e nos seguintes anos 2009 e 2010 decairam, voltando a crescer novamente nos anos de 2011 e 2012, e nos próximos anos tem voltado a decair.

Mapeamos os autores mais influêntes por variabilidades em arquiteturas de software e componentes, de acordo com este MS, classificando-os em um gráfico com o *Top 30* dos autores com mais publicações. Os 30 autores foram extraídos dos 383 estudos recuperados por este MS, de acordo com o filtro #1. A Figura 1.9 apresenta esta lista dos 30

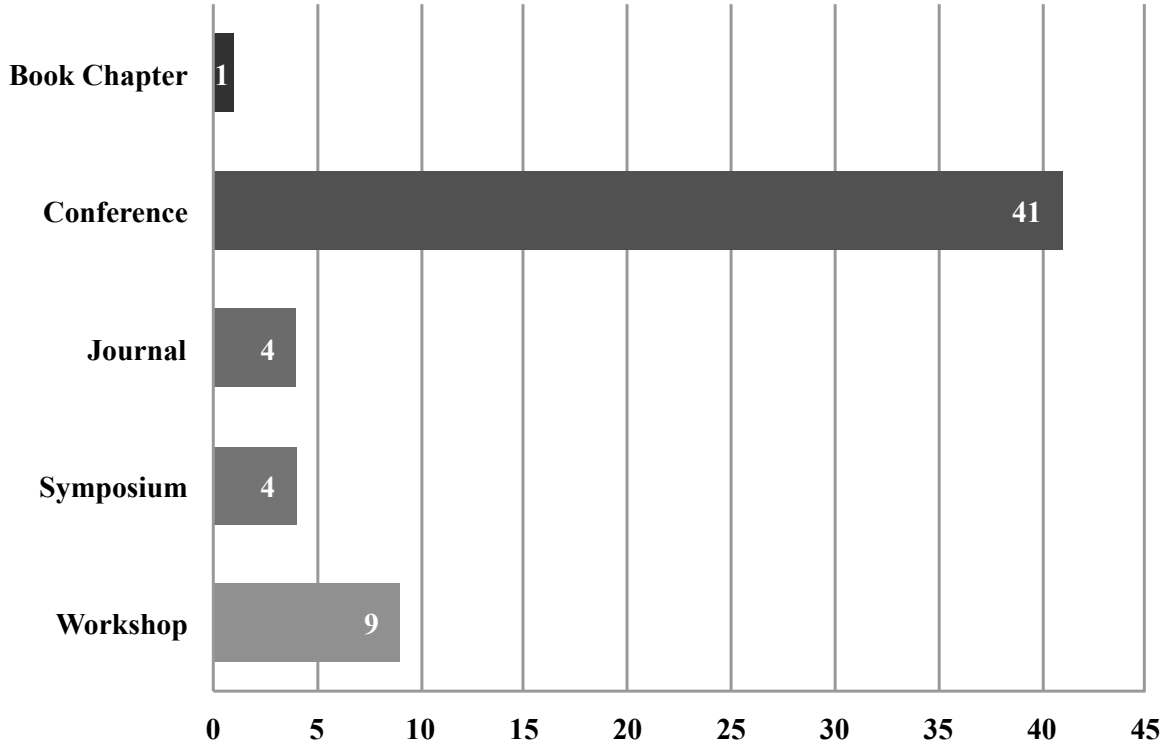

Figura 1.7: Número de Estudos por Tipo de Publicação.

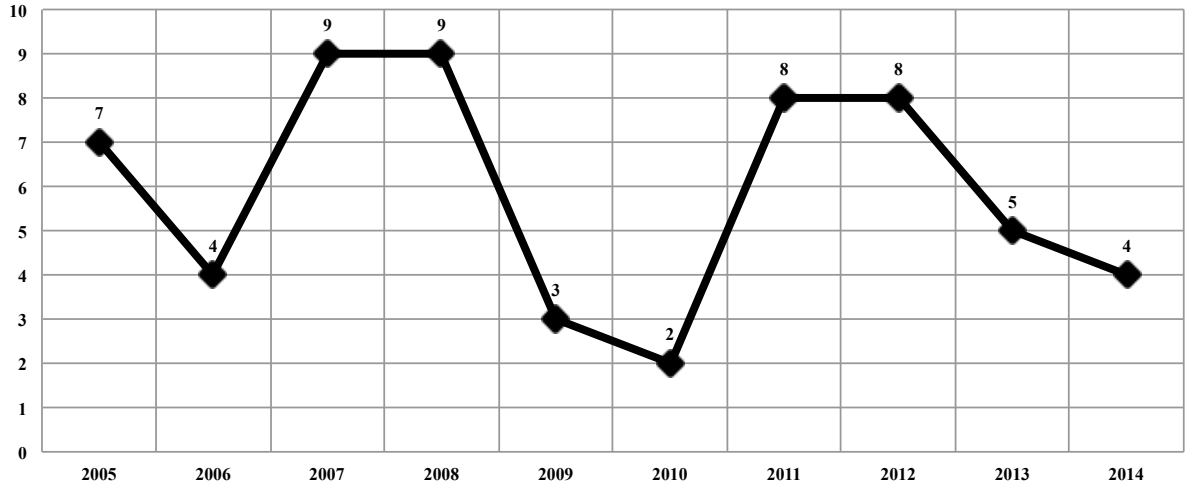

Figura 1.8: Trabalhos Recuperados a Partir de 2005 de Acordo com o Filtro #2.

autores mais influêntes. Destes, *Galster* está presente na maioria dos estudos recuperados, totalizando 13 estudos. Em seguida *Avgeriou*, com 12 estudos. Os autores *Dhugana* e *Grübacher* tiveram 9 estudos recuperados. *Kang* e *Kuleza* tiveram 7 estudos. Após esses, os números de publicações são bem próximos.

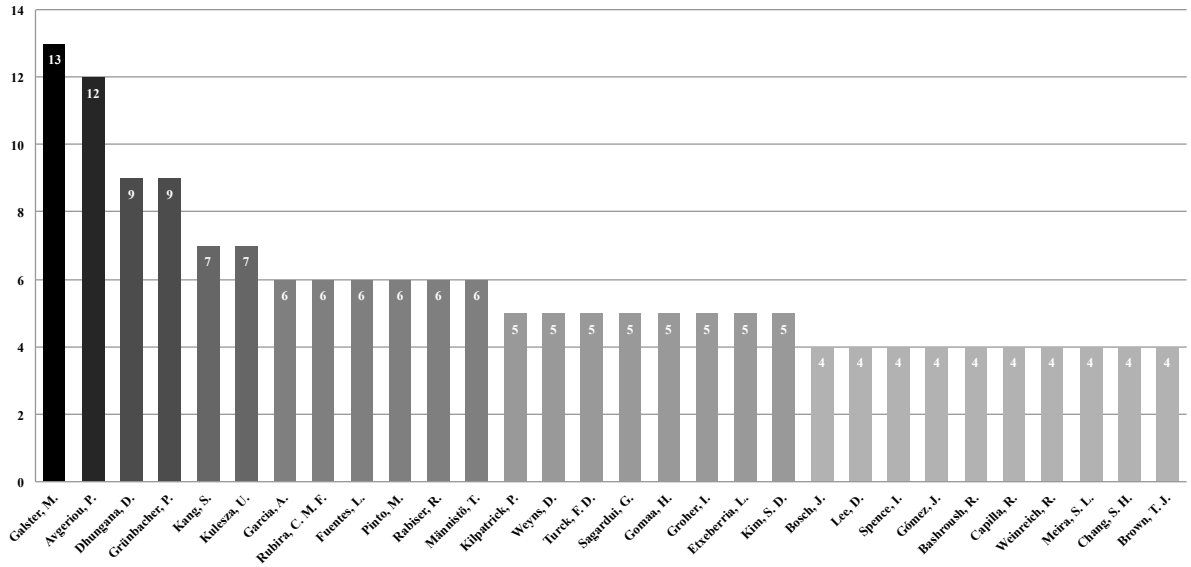

Figura 1.9: *Top 30* dos Autores mais Influêntes (filter #1).

Mapeamos também os 20 autores mais influêntes de acordo com o filtro #2, com 59 estudos recuperados. Destes *Galster* se encontra em 9 publicações e *Avgeriou* em 8 estudos. A Figura 1.10 apresenta a classificação dos *Top 20* autores mais influêntes.

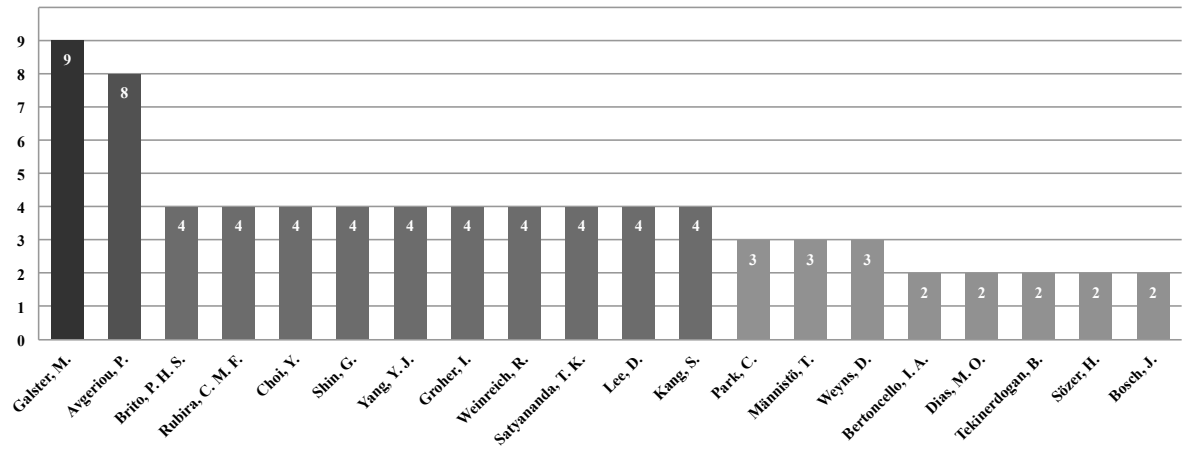

Figura 1.10: *Top 20* dos Autores Mais Influêntes (filter #2).

O filtro #3 retornou 39 estudos, e destes, mapeamos os 10 autores mais influêntes. A Figura 1.11 apresenta este mapeamento com o *Top 10* dos autores.

Ainda, as palavras-chave mais relevantes dos 383 estudos recuperados por meio do filtro #1 foram classificadas em um gráfico *Word Cloud* e são apresentadas na Figura 1.12. A palavra-chave "*Software Architecture*"esteve presente em 10 dos estudos, "*Variability*"em

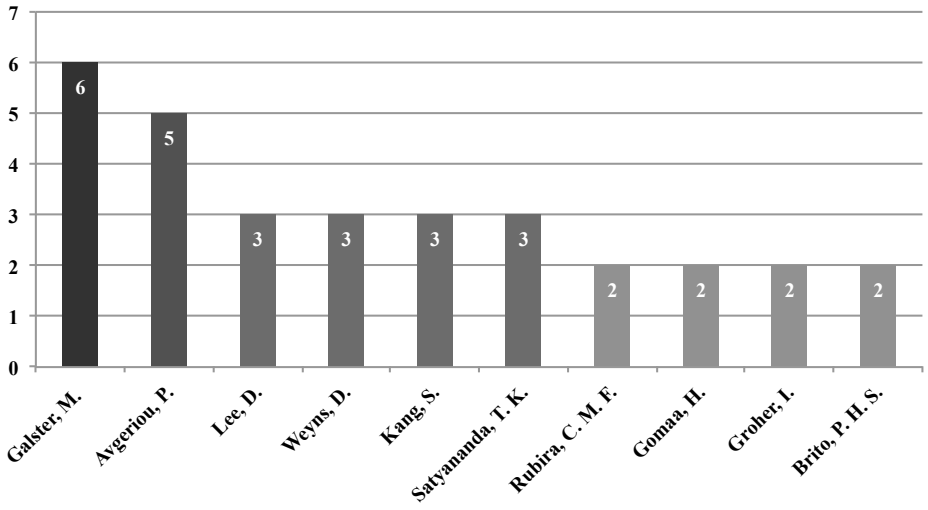

Figura 1.11: *Top 10* dos Autores Mais Influêntes (filter #3).

6 dos estudos e em seguida "*Software Product Line*"em 4 dos estudos. "*Domain Engeneering*"em 3 dos estudos, "*Traceability*", "*Variability Management*", "*Product Line Engineering*", "*Software Product Lines*"e "*Product Line Architecture*"estiveram presente em 2 dos estudos. As demais palavras-chave estiveram presente em apenas 1 estudo.

Figura 1.12: *Word Cloud* das Palavras-chave (filter #1).

A.3.3 Seleção dos Estudos Mais Relevantes

A questão de pesquisa deste MS será respondida de acordo com os 39 trabalhos recuperados por meio do filtro #3 do processo de busca. As próxima Seções apresentam os dados da análise dos estudos em termos de bases de dados digitais, tipos de pesquisa, questões de pesquisa e alguns modelos de contribuição. Tais modelos de contribuição, além dos já destacados foram:

1. **Contexto de Variabilidade**, apresentando em qual contexto variabilidade é apresentada no estudo recuperado. Tal contexto se aplica as seguintes classificações: (i) Gestão de Variabilidade; (ii) Implementação de Variabilidade; (iii) Rastreabilidade em Arquitetura; (iv) Variabilidade em Arquitetura; (v) Variabilidade em Requisitos; e (vi) Verificação e Validação;

2. *Tipo de Contribuição*, apresentando qual o tipo de contribuição apresentado no estudo recuperado, sendo estes tipos destacados como: (i) Abordagem; (ii) Método; (iii) Métrica; (iv) Modelo; e (v) Técnica;

3. *Nível de Variabilidade*, apresentando em qual nível foi identificado variabilidade nos trabalhos, sendo estes níveis destacados como: (i) Código; (ii) Componentes; (iii) Componentes e Conectores; (iv) Componentes e Interfaces; (v) Componentes, Conectores e Interfaces; (vi) Conectores, Portas e Interfaces; (vii) Interfaces; (viii) Modelo de Características; e para os trabalhos em que não foi possível identificar o nível de variabilidade, optamos por classificá-los como (ix) Indefinido;

4. *Nível de Componente*, nesta classificação o objetivo é indicar a que nível de componente se aplica a variabilidade, os itens para esta são: (i) Pacotes; (ii) UML; (iii) UML e Pacotes; (iv) XML; e para os níveis de componente não identificados no trabalho recuperado, definimos que seriam marcados com o item (v) Indefinido;

5. *Tipo de Arquitetura*, nesta classificação o objetivo é identificar qual o tipo de arquitetura que se aplica no estudo recuperado, os tipos de arquitetura foram definidos como: (i) Lógica; (ii) Física; e para os tipos não definidos optamos por classificar como (iii) Indefinido; e

6. *Tipo de Projeto*, nesta classificação o objetivo é identificar qual o tipo de projeto se enquadra o trabalho recuperado, sendo estes tipos classificados como: (i) Acadêmico; (ii) Indústrial;

A Tabela 1.4 apresenta os 39 estudos finais selecionados, os quais foram analisados e considerados relevantes para a leitura na íntegra nesta pesquisa. Finalmente, os respectivos estudos citados são discutidos na Seção A.3.4 deste documento, onde suas referências estão contidas na coluna *Ref.* da Tabela 1.4.

Fontes de Busca x Classificação das Buscas x Contexto de Variabilidade

Neste cenário, a Figura 1.13 apresenta um gráfico em formato *Bubble Plot* mostrando a relação entre a quantidade de estudos encontrados nas fontes de busca indexadas e a sua respectiva classificação das buscas, baseada nas seis categorias de classificação sugerida na Seção A.2.5 e no modelo de contribuição "Contexto de Variabilidade", apresentado nesta Seção. Desta forma, é possível identificar pela representação gráfica em quais buscadores os estudos com o contexto de variabilidade identificado pertencem ao tipo de classificação da busca.

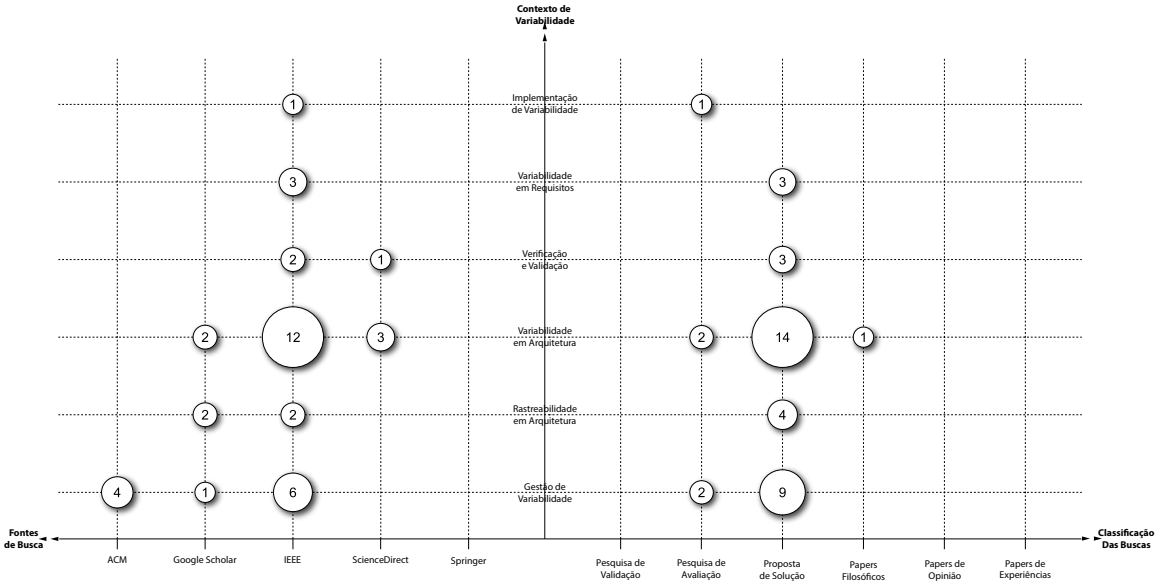

Figura 1.13: Relação dos Resultados Obtidos em relação a: Fontes de Busca x Classificação das Buscas x Contexto de Variabilidade.

Fontes de Busca x Nível de Variabilidade x Nível de componentes

Na Figura 1.14, um outro gráfico em formato *Bubble Plot* apresenta os resultados dos trabalhos recuperados, baseados nas classificações apresentadas na Seção A.2.5, e também os modelos de contribuição 3 e 4, destacados nesta Seção. O gráfico apresenta a representação

Tabela 1.4: Relação Final dos Estudos Considerados Relevantes para Esta Pesquisa.

Ref.	Autor	Título	Ano	Classificação	Fonte de Busca	Tipo de Publicação	Local de Publicação
1	Galster and Avgeriou [S1]	The Notion of Variability in Software Architecture - Results from a Preliminary Exploratory Study	2011	Pesquisa de Avaliação	ACM	Workshop	VaMoS
2	Bertoncello et al. [S2]	Explicit Exception Handling Variability in Component-based Product Line Architectures	2008	Proposta de Solução	ACM	Symposium	WEH
3	Choi et al. [S3]	An Approach to Extension of UML 2.0 for Representing Variabilities	2005	Proposta de Solução	ACM	Conference	ACIS
4	Tekinerdogan and Sözer [S4]	Variability Viewpoint for Introducing Variability in Software Architecture Viewpoints	2012	Proposta de Solução	ACM	Conference	WICSA/ECSA
5	Galvão et al. [S5]	A Model for Variability Design Rationale in SPL	2010	Proposta de Solução	Google Scholar	Conference	ECSA
6	Galster [S6]	Dependencies, Traceability and Consistency in Software Architecture: Towards a View-based Perspective	2010	Proposta de Solução	Google Scholar	Conference	ECSA/TDSA
7	Satyananda et al. [S7]	Identifying Traceability between Feature Model and Software Architecture in Software Product Line using Formal Concept Analysis	2007	Proposta de Solução	Google Scholar	Conference	ICCSA/FTRA
8	Son et al. [S8]	Method and apparatus for providing implicit variability rules for component model and architecture design	2013	Estudos Filosóficos	Google Scholar	Patente	United States Patent
9	Galster et al. [S9]	Variability in Software Architecture - Views and Beyond	2013	Proposta de Solução	Google Scholar	Workshop	VARSA
10	Groher and Weinreich [S10]	Integrating Variability Management and Software Architecture	2012	Proposta de Solução	IEEE	Conference	WICSA/ECSA
11	Galster and Avgeriou [S11]	Handling Variability in Software Architecture: Problems and Implications	2011	Pesquisa de Avaliação	IEEE	Conference	WICSA
12	Groher and Weinreich [S12]	Supporting Variability Management in Architecture Design and Implementation	2013	Proposta de Solução	IEEE	Conference	HICSS
13	Pérez et al. [S13]	Plastic Partial Components: A Solution to Support Variability in Architectural Components	2009	Proposta de Solução	IEEE	Conference	WICSA/ECSA
14	Satyananda et al. [S14]	Formal Verification of Consistency between Feature Model and Software Architecture in Software Product Line	2007	Proposta de Solução	IEEE	Conference	ICSEA
15	Duszynski et al. [S15]	Variant Comparison - A Technique for Visualizing Software Variants	2008	Proposta de Solução	IEEE	Conference	WCRE
16	Gomaa [S16]	Evolving Software Requirements and Architectures using Software Product Line Concepts	2013	Proposta de Solução	IEEE	Magazine	TwinPeaks
17	Bragança et al. [S17]	Adopting Computational Independent Models for Derivation of Architectural Requirements of Software Product Lines	2007	Proposta de Solução	IEEE	Workshop	MOMPES
18	Fant et al. [S18]	A Pattern-based Modeling Approach for Software Product Line Engineering	2013	Proposta de Solução	IEEE	Conference	HICSS
19	Satyananda et al. [S19]	A Formal Approach to Verify Mapping Relation in a Software Product Line	2007	Proposta de Solução	IEEE	Conference	CIT
20	Haber et al. [S20]	Hierarchical Variability Modeling for Software Architecture	2011	Proposta de Solução	IEEE	Conference	SPLC
21	Razavian and Khosravi [S21]	Modeling Variability in the Component and Connector View of Architecture Using UML	2008	Pesquisa de Avaliação	IEEE	Conference	ACS/IEEE
22	Reiser et al. [S22]	Compositional Variability - Concepts and Patterns	2009	Proposta de Solução	IEEE	Conference	HICSS
23	Moon et al. [S23]	A Metamodeling Approach to Tracing Variability between Requirements and Architecture in Software Product Lines	2007	Proposta de Solução	IEEE	Conference	CIT
24	Brito et al. [S24]	Verifying Architectural Variabilities in Software Fault Tolerance Techniques	2009	Proposta de Solução	IEEE	Conference	WICSA
25	Zhang et al. [S25]	Formally Composing Components in Product Line Context	2008	Proposta de Solução	IEEE	Symposium	ISISE
26	Murwantara [S26]	Initiating Layers Architecture Design for Software Product Line	2011	Proposta de Solução	IEEE	Conference	URKE
27	Lin et al. [S27]	An Approach for Modelling Software Product Line Architecture	2010	Proposta de Solução	IEEE	Conference	CiSE
28	Hendrickson and Hoek [S28]	Modeling Product Line Architectures through Change Sets and Relationships	2007	Proposta de Solução	IEEE	Conference	ICSE
29	Weyns et al. [S29]	An Architectural Approach to Support Online Updates of Software Product Lines	2011	Proposta de Solução	IEEE	Conference	WICSA
30	Ryu et al. [S30]	Designing an architecture of SNS platform by applying a product line engineering approach	2012	Pesquisa de Avaliação	IEEE	Conference	ICIS
31	Savolainen et al. [S31]	Transitioning from Product Line Requirements to Product Line Architecture	2005	Proposta de Solução	IEEE	Conference	COMPSAC
32	Kim et al. [S32]	Rule-based Component Development	2005	Proposta de Solução	IEEE	Conference	SERA
33	Kim et al. [S33]	Building Software Product Line from the Legacy Systems - Experience in the Digital Audio & Video Domain	2007	Proposta de Solução	IEEE	Conference	SPLINE
34	Bastarrica et al. [S34]	From a Single Product Architecture to a Product Line Architecture	2007	Proposta de Solução	IEEE	Conference	SCCC
35	Heider et al. [S35]	A Case Study on the Evolution of a Component-based Product Line	2012	Pesquisa de Avaliação	IEEE	Conference	WICSA/ECSA
36	Galster and Avgeriou [S36]	Chapter 6 - Supporting Variability Through Agility to Achieve Adaptable Architectures	2014	Proposta de Solução	ScienceDirect	Journal	ASA
37	Capilla et al. [S37]	An overview of Dynamic Software Product Line architectures and techniques: Observations from research and industry	2014	Proposta de Solução	ScienceDirect	Journal	JSS
38	Ahmed and Capretz [S38]	The software product line architecture: An empirical investigation of key process activities	2008	Proposta de Solução	ScienceDirect	Journal	IST
39	Galster et al. [S39]	Variability in Software Architecture - State of the Art	2014	Proposta de Solução	ScienceDirect	Journal	JSS

das Fontes de Busca, comparados com os Níveis de Variabilidade identificados e a sua respectiva classificação através do Nível de Representação de Componente identificado.

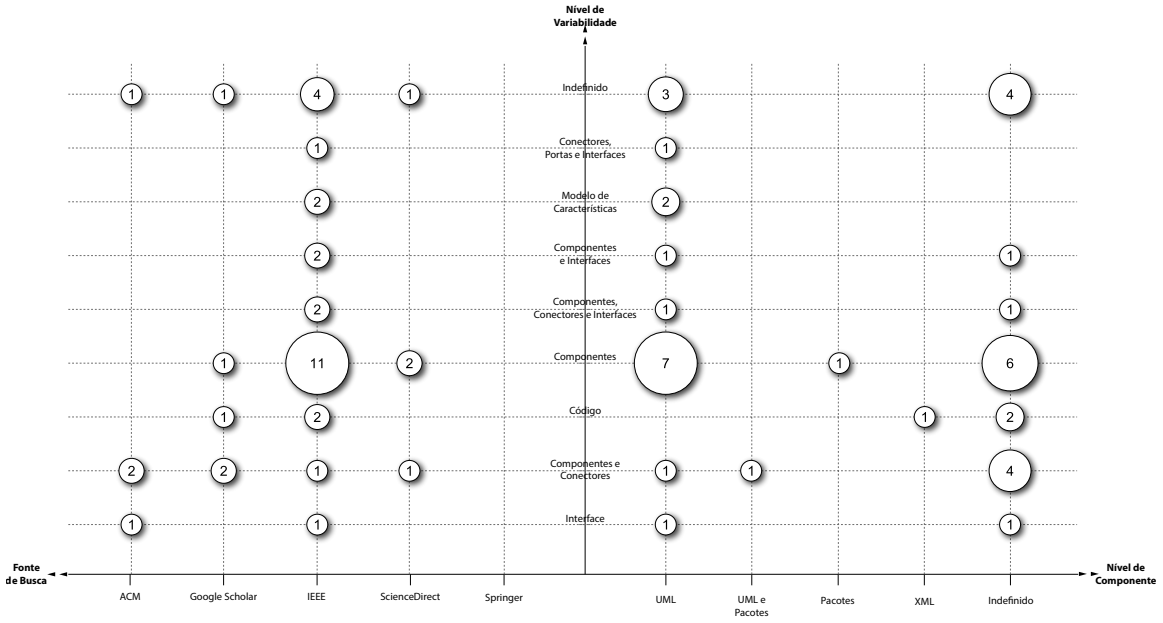

Figura 1.14: Representação dos resultados obtidos através das Fontes de Busca x Nível de Variabilidade x Nível de Componentes

A.3.4 Discussão dos Estudos Mais Relevantes

Para melhor compreensão, os 39 estudos foram divididos, classificados e ordenados respectivamente de acordo com a coluna *Ref.* da Tabela 1.4 apresentada na Seção A.3.3.

1. Este artigo é uma pesquisa exploratória realizada por *Galster* e *Avgeriou* [S1] na conferência ECSA em 2011, e foi dividida em duas partes, sendo uma pesquisa com 11 especialistas e um grupo de mini-foco com 4 participantes. Como resultado, observaram que parece não haver nenhuma compreensão comum de variabilidade no contexto de arquitetura de software. Para tanto, entende-se que este assunto pode exigir fundamentos mais teóricos;

2. Este artigo, de *Bertoncello et al.* [S2] apresenta um método para refatoração de linha de produtos de arquitetura orientada a objeto, com o objetivo de separar explicitamente o seu comportamento normal e excepcional em diferentes componentes de software;

3. *Choi et al.* [S3], propõem uma abordagem para representação de arquiteturas de linhas de produtos baseada em UML 2.0, sendo que novos recursos são adicionados na UML 2.0 a fim de modelar arquitetura de sistemas. Tal abordagem representa explicitamente variabilidades não só para os componentes, mas também para conectores;

4. Este artigo, proposto por *Tekinerdogan* e *Sözer* [S4] fornece uma visão geral de abordagens para lidar com a variabilidade no nível de projeto de arquitetura e, em seguida, apresenta o ponto de vista da variabilidade. O ponto de vista da variabilidade aborda as preocupações para a variabilidade e pode ser usado para introduzir variabilidade em pontos de vista de arquitetura de software;

5. Neste artigo, *Galvão et al.* [S5] apresentam um modelo para a especificação de variabilidade lógica de projeto e sua aplicação à modelagem de variabilidade de arquitetura em linha de produto de software;

6. *Galster* [S6] neste artigo, explora como os pontos de vista e visões de arquitetura de software podem ajudar a representar as dependências e facilitar a rastreabilidade e consistência na arquitetura de software. Além disso, com base em suas experiências, discute os desafios que ocorrem quanto a garantir rastreabilidade, consistência e gerenciamento de dependências entre os artefatos de desenvolvimento de software, com ênfase nos elementos de arquitetura de software;

7. Neste artigo, *Satyananda et al.* [S7] apresentam uma abordagem para identificação e rastreabilidade entre modelo de características e componentes, e visão do conector da arquitetura de software usando a técnica Análise de Conceito Formal (*Formal Concept Analysis* - FCA);

8. Este documento de *Son et al.* [S8], é uma patente que apresenta um método e aparelho para fornecer regras de variabilidade para o modelo de componentes e projeto de arquitetura;

9. O VARSA (*Workshop on Variability in Software Architecture*) é um *workshop* sobre variabilidade em arquitetura de software que foi realizado em conjunto com o WICSA e ECSA em 2012. Este artigo, é um relatório que resume os temas do *workshop* e apresenta os resultados das discussões dos grupos de trabalho;

10. Este artigo, de *Groher* e *Weinreich* [S10], mostra como é integrada a modelagem de variabilidade ortogonal e modelagem de recursos para a abordagem LISA, que é uma abordagem para gerenciamento de arquitetura e análise. Em LISA. a gestão de variabilidade não é uma atividade, mas uma parte integrada do ciclo de vida do desenvolvimento da arquitetura;

11. *Galster* e *Avgeriou* [S11] apresentam um estudo exploratório para identificar os problemas que ocorrem quando a execução de tarefas relacionadas com a variabilidade

durante o desenvolvimento da arquitetura do software. Em um estudo com 27 participantes, foram identificados 11 problemas vividos pelos sujeitos participantes do estudo. Além disso, o documento também apresenta implicações dos resultados para o campo de arquitetura de software;

12. Neste artigo, *Groher* e *Weinreich* [S12] apresentam uma extensão para a abordagem LISA, um modelo e um conjunto de ferramentas para gerenciamento de arquitetura e análise. O objetivo desta extensão, é a gestão integrada da variabilidade durante o projeto de arquitetura de software e implementação. Os arquitetos constantemente estarão cientes das variantes que estão trabalhando e suas implicações sobre o projeto de arquitetura e implementação;

13. *Pérez et al.* [S13], propõe neste artigo uma noção de componentes parciais de plástico (Plastic Partial Components) para suportar variações internas. A especificação desses componentes é realizada utilizando técnicas de composição de software invasiva e sem emaranhamento do núcleo e das arquiteturas de produto de uma LPS;

14. Neste artigo, *Satyananda et al.* [S14] apresentam uma abordagem formal para verificar a coerência entre modelo de características e de componentes, e visão do conector da arquitetura de software. Ao utilizar um sistema de verificação de protótipo (*Prototype Verification System* - PVS), apresentam um modelo de descrição de recursos e descrição da arquitetura, e ilustra a abordagem de verificação de consistência usando um exemplo de linha de produtos de relógio digitais (*Digital Watch*);

15. Neste artigo, *Duszynski et al.* [S15] propõe uma técnica que visualiza os pontos de variação no nível de arquitetura de software. Tal técnica, chamada de *variant comparison*, tem sido aplicada com êxito em um estudo interno e dois industriais. Este artigo resume suas experiências práticas na aplicação da técnica proposta, além de discutir lições aprendidas sobre a forma como a técnica pode permitir o gerenciamento de variabilidade explicita em uma organização de desenvolvimento;

16. *Gomaa* [S16] propõe neste artigo uma abordagem de desenvolvimento evolutivo que utiliza a LPS e conceitos de modelagem de características para evolução de requisitos e arquiteturas de software. Esta abordagem relaciona de perto a evolução da arquitetura de software com a evolução dos requisitos de software, além de fornecer rastreabilidade entre os requisitos e o arquiteto;

17. Neste trabalho, *Braganca et al.* [S17] apresenta uma evolução do método 4SRS (*Four Step Rule Set*) destinado a LPS. O 4SRS é um método dirigido por modelo baseado

em UML para o desenvolvimento de um sistema único que fornece suporte para o arquiteto de software nesta tarefa. Neste trabalho é descrito como esta extensão aborda a transformação de requisitos funcionais (casos de uso) em requisitos baseados em componentes para uma Arquitetura de LPS. Esta evolução, se baseia em uma extensão do perfil UML-F, que fornece extensões das notação UML necessárias para modelar variabilidade;

18. Neste artigo *Fant et al.* [S18] descrevem uma abordagem de engenharia de LPS, que é baseado em modelos e padrões. Esta abordagem, baseia a Arquitetura de LPS em padrões de arquitetura e relaciona esses padrões aos recursos de LPS. No artigo, é descrito como esta abordagem se aplica a uma LPS de voo espacial não tripulada;

19. *Satyananda et al.* [S19] neste artigo, por meio da FCA (*Formal Concept Analysis*) e PVS (*Prototype Verification System*) apresentam uma abordagem formal para a identificação de rastreabilidade e verificação de consistência entre o modelo de características e, componentes e visão de conector de arquitetura de software;

20. Neste artigo *Haber et al.* [S20] propõem uma modelagem de variabilidade hierárquica que permite especificar variabilidade em componente integrada com a hierarquia de componentes e localmente para os componentes. Apresentam também um modelo de meta para a modelagem de variabilidade hierárquica. para formalizar as ideias conceituais. Para se obter uma implementação da abordagem proposta juntamente com o apoio de ferramentas, estenderam a linguagem de descrição de arquitetura *MontiArc* com modelagem de variabilidade hierárquica. Para ilustrar a abordagem apresentada, usou-se um exemplo do domínio de sistemas automotivos;

21. *Razavian e Khosravi* [S21], neste artigo, propuseram um método de modelagem de variabilidade que é especificamente concebido para a exibição do componente e conector (C&C) da arquitetura. Usou-se a UML 2 como a linguagem de modelagem de arquitetura. Soluções de modelagem são propostos e classificados de acordo com o tipo de elemento variável e as técnicas utilizadas para realizar variabilidade. Inclui-se neste artigo também um estudo sobre as maneiras de evitar sobrecarregar a visão quando incluindo variabilidade. Um exemplo de caso é utilizado para clarificar os diferentes aspectos do método proposto;

22. Este artigo, proposto por *Reiser el al.* [S22], apresenta um modelo de gerenciamento rigoroso de variabilidade no contexto de uma hierarquia de tal composição, que estende de forma consistente o paradigma de projeto baseado em componentes para

o gerenciamento de variabilidade. Também apresentam vários padrões básicos de especificação de variabilidade quando se aplica este modelo na prática, além de mostrarem como tudo isso foi tecnicamente realizado em uma linguagem de descrição de arquitetura (EAST-ADL2) para o desenvolvimento de software automotivo, neste exemplo, mas tais conceitos se aplicam a outros domínios industriais que envolvem os sistemas intensivos de software;

23. *Moon et al.* [S23] propõem artigo uma abordagem de meta-modelagem para apoiar o rastreamento da variabilidade em requisitos e arquitetura. São propostos dois metamodelos que representam os requisitos de domínio e arquitetura de domínio com variabilidade. Baseado nos metamodelos propostos, é descrito as relações de rastreabilidade entre os requisitos e arquitetura em relação à variabilidade;

24. *Brito et al.* [S24] neste artigo consideram a representação de diferentes técnicas de tolerância a falhas de software como uma arquitetura de LPS (ALPS) para promover a reutilização de artefato de software. A ALPS proposta permite especificar uma série de aplicações arquitetônicas estreitamente relacionadas, que é obtido através da identificação de pontos de variação associados às decisões de projeto sobre tolerância a falhas de software. A abordagem proposta compreende também a formalização da ALPS, usando *B-Method* e CSP, para sistematizar a verificação dos sistemas de software tolerantes a falhas em nível de arquitetura;

25. Neste artigo, *Zhang et al.* [S25], apresentam métodos sistemáticos para composição de componentes com vADL. Técnicas de ligação de pontos de variabilidade, montagem de restrições e restringir condição de guarda são apresentados em detalhe, o que pode montar as variabilidades dos componentes. Técnica de montagem de comportamento também são descrita, utilizando mecanismo combinado paralelo de π-cálculo. Variabilidade na estrutura de montagem, montagem de portas e montagem comportamento também são discutidas. A fim de certificar exatidão e consistência, alguns métodos de análise para composição componentes também são fornecidos;

26. Este artigo, proposto por *Murwantara* [S26], apresenta algumas abordagens para resolver os problemas de projeto de arquitetura. Agrupamento de Características com preocupações específicas é o ponto de partida. Em seguida, os modelos de componentes e composição elemento de interação, com base em sua capacidade. Neste trabalho também é apresentado lógicas que ligam o elemento de interação e componente; Usando camadas, a arquitetura de linha de produtos pode ser estabelecida.

27. Neste artigo, Lin, Ye e Li [S27] propõe uma abordagem para transformar os modelos de recursos para modelos de arquitetura. Esta abordagem iterativa explicitamente modela a variabilidade apresentada no modelo de características em artefatos arquitetônicos e transfere as dependências de recursos para as interações entre os artefatos arquitetônicos no modelo de arquitetura. A abordagem melhora a rastreabilidade entre os modelos de recursos e modelos de arquitetura, assim, proporcionar uma melhor orientação para o desenvolvimento da arquitetura do produto membro da LPS;

28. Neste artigo, *Hendrickson e Hoek* [S28], propõem uma abordagem que utiliza conjuntos de alterações ao grupo relatando diferenças arquiteturais e relações para governar quais combinações de conjunto de mudança são validos quando composto em uma arquitetura de produto específica. O resultado eleva a modelagem da variabilidade para modelagem de estrutura arquitetônica, consolida pontos de variação relacionados e expressa, e separadamente administra suas compatibilidades;

29. *Weyns et al.* [S29], neste artigo, apresentam uma abordagem arquitetônica para atualizar produtos LPS que suporta múltiplas preocupações. A abordagem é composta por duas partes complementares: (1) Um ponto de vista de atualização que define as convenções para a construção e utilização de arquitetura vistas a lidar com vários problemas de atualização; e (2) um quadro de apoio que fornece uma infraestrutura de apoio extensível integradores de LPS. Ainda, avaliou-se a abordagem de um LPS industrial para sistemas de logística fornecendo evidência empírica para os seus benefícios e recomendações;

30. Este artigo de *Ruy et al.* [S30] mostra como gerenciar similaridades e variabilidades de uma SNS (*Social Networking Service*). Uma avaliação é feita sobre os pontos fortes e fracos da arquitetura do Facebook, utilizando o método ATAM (*Architecture Tradeoff Analysis Method*). Como resultado da aplicação de um *framework* para a pratica de LPS, a visão em camadas e baseada em componentes são ilustradas juntamente com a representação de variabilidades pelo método PLUS e OVM;

31. *Savolainen et al.* [S31] apresentam um conjunto de regras que mapeiam os valores de restrições de requisitos para a seleção dos valores de restrições de características que por sua vez mapeiam os valores de restrição de ativos da arquitetura;

32. Este artigo, proposto por *Kim et al.* [S32] apresenta uma abordagem intitulada ACM (*Adaptável Component Model*), cujo objetivo é separar as propriedades estáveis e básicas de propriedades variáveis. Também foi proposto uma nova arquitetura de

componentes para incluir as regras de componentes que definem a parte variável. Além disso, a reutilização de componentes é verificada por meio de regras *re-defining* para aplicação em um sistema de vendas de seguros;

33. *Kim et al.* [S33] apresentam neste artigo suas experiências em projetar ALPS como uma arquitetura de referencia comum no domínio AV Digital. Descrevem o seu processo de desenvolvimento, com aplicação em um projeto de caso, mostram também princípios e diretrizes para projetar e construir uma LPS. Como estudo de caso , foi descrito como encontrar , extrair e desenvolver os recursos do núcleo da família de produtos contra o processo e diretrizes que eles utilizam;

34. Neste artigo, *Bastarrica et al.* [S34] apresentam uma abordagem diferente, onde eles projetam uma ALPS depois de apenas um produto ser construído. MCC+ é uma extensão da abordagem MCC (*Model Consistency Checker*), desenvolvida por eles, onde dado um produto para a verificação de consistência de diferentes diagramas em modelos UML, identificam as variabilidades que devem gerenciar de forma útil para uma variedade de ferramentas de modelagem, obtendo assim uma linha de produto MCC-SPL. Consequentemente, atualiza-se a arquitetura original para uma ALPS;

35. *Heider et al.* [S35] neste artigo, apresentam resultados de um estudo de caso sobre análises de impacto e suporte da ferramenta desejada na evolução de LPS. Os resultados são baseados na observação de 30 pessoas/mês de desenvolvimento. Foram analisadas alterações feitas em uma LPS em cenários típicos de evolução, envolvendo os principais desenvolvedores. Dados empíricos foram utilizados sobre as atividades de desenvolvimento observadas e análise de impacto para derivar um modelo de informações de rastreamento mostrando as ligações de rastreamento frequentemente desejados;

36. Neste estudo, *Galster* e *Avgeriou* [S36], discutem o *background* relacionado a variabilidade e agilidade. Apresentam trabalhos relacionados na combinação de variabilidade e agilidade, abordam os desafios que ocorrem quando se combina variabilidade e agilidade, antes de elaborar sobre os argumentos para combinação de variabilidade e agilidade. Ainda, apresentam uma abordagem para a manipulação de variabilidade que utiliza conceitos de desenvolvimento ágil, e por fim, apresentam um exemplo industrial que é utilizado para ilustrar as etapas individuais da abordagem apresentada;

37. Nesta pesquisa, *Capilla et al.* [S37], fornecem uma visão geral das técnicas atuais que tendem a enfrentar os muitos desafios mecanismo de variabilidade de tempo de execução no contexto de LPSD (Linha de Produto de Software Dinâmica). Também

fornecem uma visão integrada dos desafios e soluções que são necessárias para apoiar os mecanismos de variabilidade de execução em modelos DSPL e arquiteturas de software;

38. As principais contribuições deste artigo, proposto por *Ahmed e Capretz* [S38], é aumentar a compreensão da influência das atividades chaves do processo da ALPS sobre o desempenho geral da LPS através da realização de uma investigação empírica abrangente, cobrindo uma ampla gama de organizações atualmente envolvido no negócio de LPS. Este é o primeiro estudo a investigar empiricamente e demonstrar as relações entre algumas das atividades de processo de ALPS e o desempenho LPS global de uma organização, baseada no conhecimento dos autores. Os resultados desta investigação fornecem evidências empíricas de que as atividades do processo de ALPS desempenham um papel significativo no desenvolvimento de sucesso e gestão de uma LPS; e

39. *Galster et al.* [S39] destacam a importância de lidar com variabilidade em nível de arquitetura na construção de sistemas de software complexos que atendem às necessidades das diferentes partes interessadas. Deste modo, este artigo vista proporcionar aos pesquisadores e profissionais da indústria com insights sobre o estado da arte de variabilidade em nível de arquitetura de software. De 20 trabalhos escritos, 4 foram aceitos como sendo de alta qualidade e são destacados neste artigo.

Através da analise dos 39 estudos recuperados pelo filtro #3, fica evidente que não há uma padronização de representação e especificação de variabilidades em arquiteturas de software e arquiteturas baseadas em componentes no período de pesquisa deste MS. Cada autor explora de forma diferente suas necessidades, porém são adaptáveis ao contexto em que cada pesquisa segue.

Percebe-se que existe uma grande preocupação em buscar como os arquitetos e pesquisaores tem representado variabilidades em nível de arquitetura de software e componentes, através de trabalhos como [S1], [S9], [S11], [S37] e [S39].

Algumas abordagens tem uma certa padronização de representação de variabilidades em componentes por meio de estereótipos UML, como se encontra nos trabalhos [S3],[S4],[S7],[S16],[S21] e [S30].

Em uma análise geral, é evidente que exista uma diversidade de tipos de estudos que envolvem variabilidade em arquiteturas de software e componentes. Os estudos recuperados neste MS indicam que apesar de ainda existir uma lacuna, os pesquisadores tem apresentado soluções distintas para isto, e tais soluções apresentam uma perspectiva motivadora. A quantidade de trabalhos que apresentam pesquisas, principalmente expe-

rimentais, dão indícios de que há uma necessidade conseguir resultados significativos a fim de sanar esta lacuna.

Assim, este MS é considerado essencial para direcionar os pesquisadores a compreender os estudos que vem sendo realizados no contexto de variabilidade em arquiteturas de software e componentes.

A.4 Ameaças à Validade

As seguintes maiores ameças a validade deste estudo serão discutidas a seguir:

- **Questão de Pesquisa**: O objetido deste MS foi simplesmente encontrar estudos publicados que abordam variabilidade em arquitetura de software e componentes, de acordo com esta condição só uma questão de pesquisa foi definida para este estudo.
- **String de Busca**: De acordo com a questão de pesquisa proposta, geramos uma única *string* de busca genérica, com a intenção de retornar estudos que abordam tal tema. Entretanto, estudos posteriores deste MS podem generalizar a *string* de busca, a fim de expandir as buscas e obter outros resultados.
- **Fonte de Dados**: Cinco fontes de dados foram selecionadas, dentre elas as consideradas essenciais para a comunidade científica de acordo com alguns mapeamentos sistemáticos e revisões de literatura. Entretanto, quanto mais fontes de dados, mais resultados são obtido e podem ser relevantes para a pesquisa. Estudos secundários deste MS devem considerar expandir a lista de fontes de busca.
- **Viés de Publicação**: Não podemos garantir que todos os estudos relacionados ao assunto pretendido deste MS foram recuperados. Isso pode acontecer pelo fato de que os motores de buscas não são tão efetivos como desajamos que fossem para processar e executar as consultas de acordo com as palavras-chaves definidas.

A.5 Considerações Finais

Existe várias propostas na academia visando melhorar o gerenciamento e desenvolvimento de softwares, explorando o reuso de artefatos e atividades, e buscando abordagens que permitam tais controles e aperfeiçoamento dos produtos no desenvolvimento final. Estas, motivam a buscar por técnicas que exploram de forma abrangente o reúso, partindo de níveis de abstração maiores, como o caso da arquitetura de software.

Neste MS, realizamos uma busca exploratória por trabalhos que propuseram a questão de variabilidades em arquiteturas baseadas em componentes e arquiteturas de software, com o intuito de analisa-las para entender quais padrões que vem sendo estudado e utilizado para propor estas melhorias no desenvolvimento dos softwares, através a utilização de variabilidade nos componentes.

Através da analise dos estudos obtidos, percebe-se que ainda há uma grande dificuldade em gerenciar variabilidades em níveis de arquitetura de software e componentes, pelo fato de, que atualmente, não existe uma abordagem que padronize esse tipo de atividade. Sendo assim, cada estudo propõe um tipo de sugestão diferente, ou baseada em estudos propostos anteriormente, mas que no geral pode-se obter como tem sido aplicado variabilidade neste contexto de arquiteturas de software e arquiteturas baseadas em componentess.

Possíveis trabalhos futuros, podem utilizar destes resultados para analisar os trabalhos recuperados, e através de suas metodologias propor uma abordagem que visa identificar e representar as variabilidades em um nível de abstração melhor e mais amplo, levando em consideração os indícios sugeridos pelos autores e colaborando com o desenvolvimento da literatura no contexto de variabilidades em níveis arquiteturais.

Apêndice B - SPEM 2.0

O ***S**oftware and Systems **P**rocess **E**ngineering **M**eta-Model* (SPEM) é um metamodelo de engenharia de processos, bem como estrutura conceitual, especificado pela *Object Management Group* (OMG). O metamodelo SPEM pode fornecer os conceitos necessários para a modelagem, documentação, apresentação, gestão, intercâmbio, e concretização dos métodos e processos de desenvolvimento (OMG, 2008).

A especificação formal da versão 2.0 do SPEM é dividida em duas partes (OliveiraJr *et al.*, 2013b):

- O metamodelo SPEM 2.0, que define todas as regras de estruturação, especificadas como um modelo MOF e reutiliza algumas classes fundamentais da UML 2. Também define a notação de diagramas de processo específicos;
- O perfil do SPEM 2.0, que define um conjunto de estereótipos da UML 2. Tal definição abrange apenas sua representação, tornando-se dependente do metamodelo SPEM 2.0 para as declarações semânticas e de restrições.

O SPEM 2.0 é utilizado para definir processos de software e desenvolvimento de sistemas e seus componentes. O SPEM fornece elementos necessários para definir qualquer processo de desenvolvimento de software e sistemas, sem a adição de características específicas para determinados domínios ou disciplinas de desenvolvimento (OMG, 2008). O objetivo é acomodar uma grande variedade de métodos e processos de diferentes estilos, origens culturais, níveis de formalismo, modelos de ciclo de vida, e as comunidades de desenvolvimento. No entanto, o foco do SPEM é projetos de desenvolvimento. O SPEM 2.0 não é uma linguagem de modelagem de processo genérico, nem mesmo fornecer seus próprios conceitos de modelagem de comportamento. SPEM 2.0 define a capacidade para o implementador escolher a abordagem de comportamento genérico de modelagem que melhor se adapta às suas necessidades: Atividades da UML 2.0 ou BPMN, ou outro método de descrição do comportamento.

B.0.1 Estrutura de Processo do SPEM 2.0

O pacote de metamodelos da estrutura de processo contém elementos estruturais básicos para definir processos de desenvolvimento (OMG, 2008). As características mais comum relacionadas a definição de processos de desenvolvimento é o sequenciamento de fases e marcos que expressam um ciclo de vida do produto em desenvolvimento. Processos definem também como um marco vai para o próximo por meio das definições de sequências de trabalho ou workflows, operações ou eventos que normalmente ocupam tempo, perícia ou outro recurso que produza algum resultado.

O meta-modelo do SPEM é capaz de representar "diferentes tipos de 'processos, tais como processos de cachoeira, bem como modelos de processos iterativos ou incrementais, modelando-os todos como estruturas de divisão, mas a aplicação de relações estruturais diferentes e atributos descritivos expressando sua especificidades ciclo de vida.

A Tabela 2.1 apresenta as definições de trabalho do SPEM 2.0 que envolvem os processos, juntamente com os respectivos ícones, que representam os estereótipos de cada definição de trabalho.

Tabela 2.1: Estereótipos do SPEM 2.0

Ícone	Estereótipo	Descrição
	*Activity*	Elemento Atividade, que representa um agrupamento de elementos, tais como, outras instâncias de Atividades (Activity), de Uso de Tarefas (Task Uses), de Uso de Papéis (Role Uses) e de Uso de Produtos de Trabalho (Work Product Uses).
	*TaskUse*	Elemento Uso da Tarefa (Task Uses), que representa uma Tarefa sendo realizada por um Papel no contexto de uma Atividade.
	*Step*	Elemento Passo (Step), que representa um dos passos necessários para realizar a Tarefa.
	*WorkProductUse*	Elemento Uso do Produto de Trabalho (Work Product Uses), que representa um Artefato consumido ou produzido no contexto de uma Atividade específica.
	*RoleUse*	Elemento Uso do Papel (Role Uses), que representa um Papel responsável por uma ou mais Tarefas específicas.

No SPEM 2.0 as associações definem algumas regras importantes nos relacionados dos elementos. A Figura 2.1 apresenta um exemplo de associações com a atividade *Define the System*. O estereótipo «neasting» simboliza que o elemento ligado a associação está contido dentro do elemento no qual o losango está ligado. Nesta figura, os *WorkProductUses* `Use Case Model` e `Vision`, e as *TaskUses* `System Analyst`, `Find Actors and Use Cases` estão aninhados na atividade *Define the System*. Ainda, os *WorkProductUse* `Use Case Model` e `Vision` são saídas das respectivas *TaskUses* `Find Actors and Use Cases` e `Develop Vision`. Portanto, tais associações estão estereótipadas com «output».

A Figura 2.2 apresenta um exemplo de reúso entre duas atividades. Neste exemplo a associação de dependência entre as atividades `Activity 2.2` e `Activity 1.2` está estereótipada com «local contribuition», que siginifica uma contribuição entre as atividades. Assim, a atividade no qual é o alvo (*target*) da seta de dependência recebe o conteúdo contido na outra atividade. Tal contribuição se assemelha a um comando *merge*, onde funde as duas atividades. Já as atividades *Activity 2.3* e *Activity 1.3* estão relacionadas

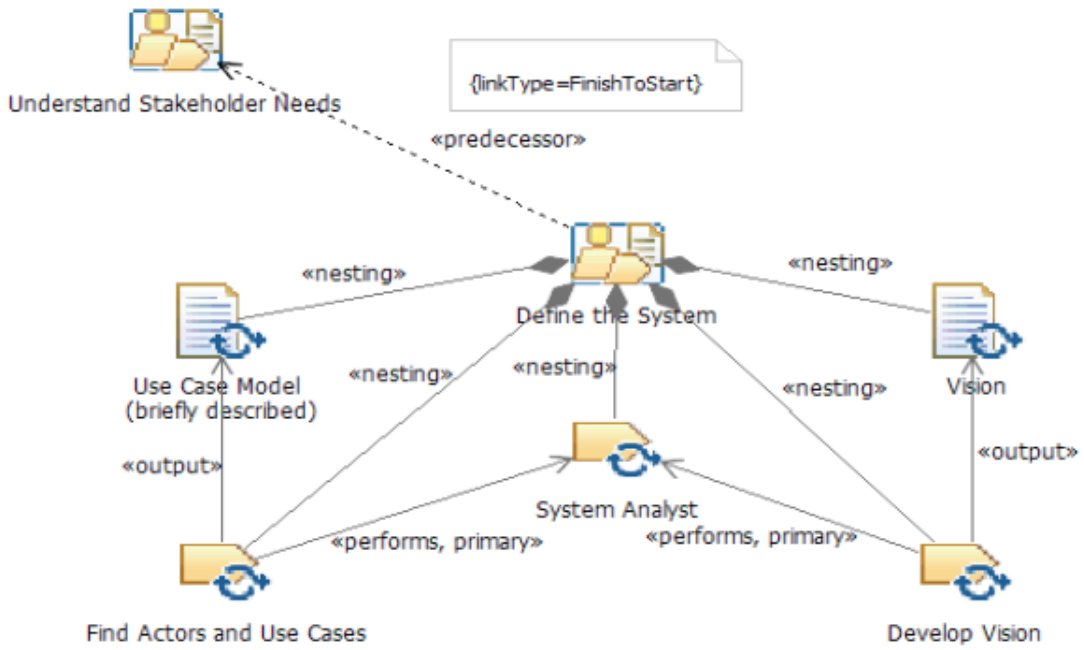

Figura 2.1: Exemplo de uma atividade com associações (OMG, 2008).

com uma seta de dependência contendo o estreótipo «local replacement», que simboliza a substituição de uma atividade por outra. Neste caso, a atividade da dependência *source* substitui a atividade apontada como *target*. A interpretação destes estereótipos pode ser observado na Figura 2.2.

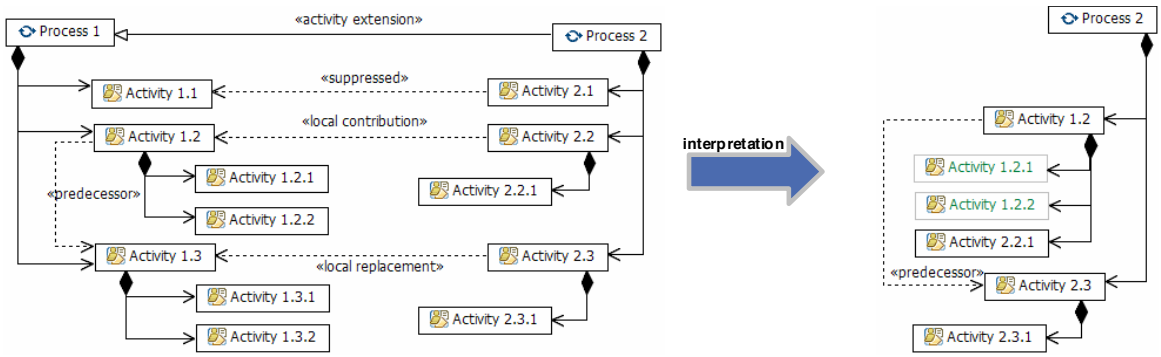

Figura 2.2: Exemplo de uma atividade reutilizando outra atividade (OMG, 2008).

O relacionamento do *RoleUse* com alguma tarefa ou artefato possui alguns estereótipos que representam o tipo de relacionamento. A Figura 2.3 apresenta um modelo onde o Papel `b_analyst` possui um relacionamento com o artefato `p_business_process` estereótipado com «responsible», que significa que o tal papel é responsável por tal artefato.

Já o relacionamento entre a tarefa e o papel possui um relacionamento estereotipado com «performs», onde significa que o papel executa a tarefa relacionada.

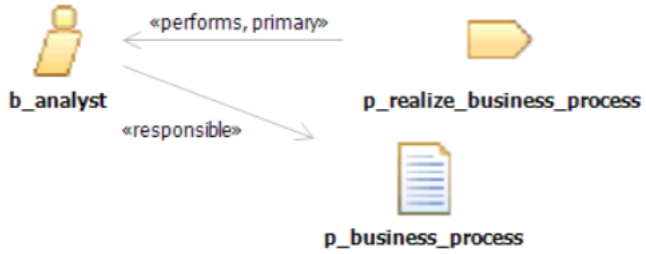

Figura 2.3: Exemplo de relacionamentos do Papel (OMG, 2008).

Apêndice C - Respostas dos Especilistas no Estudo Empírico Qualitativo do *SMartyComponents*

Este apêndice apresenta as respostas dos especialistas de forma integra, classificando-os com os elementos relacionados às categorias e códigos gerados, conforme apresentado na execução do estudo do Capítulo 6. Desta forma, especialistas, categorias e códigos (*codings)* serão apresentados separados por um ponto, como por exemplo, E1.CA1.1 referente ao especialista 1, a categoria 1 e o *coding* 1, de acordo com a Seção 6.5.

1. ***Workflow* de Requisitos - Atividade: Definição de Requisitos**

   1.1. Você considera a atividade `Definição de Requisitos` do *SMartyComponents* adequada em relação aos artefatos gerados (`Modelo de Casos de Uso` e `Modelo Conceitual de Negócio`) neste *workflow*? Justifique sua resposta.

   - **Resposta do Especialista 1**:
     - Acho que da forma que está bem sucinto e o workflow abrange o processo de forma geneeralizável [**E1.CA1.4**];
     - Dois pontos que podem contribuir para o workflow: - na definição de requisitos, o especialista de domínio poderia se envolver com"identificar casos de uso"[**E1.CA2.12**];
     - com uma nova atividade que seria"validar casos de uso". Isso porque ele que deve bater o martelo sobre o que de fato é um caso de uso [**E1.CA2.8**];
     - uma atividade de"validar modelo conceitual x casos de uso". Nos casos de uso acabamos definindo indiretamente os elementos do modelo conceitual. Então, fazer uma verificação se os dois modelos batem seria interessante [**E1.CA2.8**].

- **Resposta do Especialista 2**:
  - Sim, pois as tarefas (Descrever Processo de Negócios, Identificar Casos de Uso e Desenvolver Modelo Conceitual de Negócio) são claras e estão relacionadas de forma a contribuírem na corretude para a geração dos artefatos (Modelos de Casos de Uso e Modelo Conceitual de Negócio) [**E2.CA1.4**];
  - No entanto, um novo artefato poderia ser gerado entre o Modelo de Casos de Uso e o Modelo Conceitual de Negócio para propiciar uma ponte entre os aspectos do negócio, possibilitando verificar o que realmente é um Caso de Uso adequado para uma determinada especificação. [**E2.CA2.6**];
- **Resposta do Especialista 3**:
  - Sim, pois esta atividade é inteiramente baseada no workflow de requisitos do UML Components, o qual é baseado no Processo Unificado para desenvolvimento de software. Esses modelos já são amplamente utilizados e validados desde a definição dos mesmos, tanto no ambiente acadêmico quanto no ambiente industrial [**E3.CA1.4**].
- **Resposta do Especialista 4**:
  - Sim porque os artefatos gerados são dependentes dos requisitos, sendo portanto, adequados a esta atividade [**E4.CA1.4**].
- **Resposta do Especialista 5**:
  - Sim. As tarefas, específicas, que são responsáveis por produzirem os artefatos em questão tem os seus objetivos claramente definidos. A tarefa de"identificação de casos de uso"é antecedida pela tarefa de"descrever processos de negócio", a qual claramente é base para a definição das funcionalidades - casos de uso - propostas para a LPS [**E5.CA1.4**].
- **Resposta do Especialista 6**:
  - Sim, estes dois artefatos são suficientes para a definição de requisitos [**E6.CA1.4**].
- **Resposta do Especialista 7**:
  - Sim, considero adequados, uma vez que são saídas produzidas pelo Workflow de Requisitos [**E7.CA1.4**].
- **Resposta do Especilista 8**:

  - Considero adequada a atividade como um todo, [**E8.CA1.4**];
  - para o desenvolvimento do modelo conceitual o artefato "conhecimento do domínio da LPS" parece ser muito genérico [**E8.CA2.10**].

1.2. Você considera adequada a atribuição dos papéis (`Analista de Negócios` e `Especialista em Domínio`) envolvidos na atividade `Definição de Requisitos` do *SMartyComponents*? Justifique sua resposta.

- **Resposta do Especialista 1**:
  - Respondi a 1.2 na 1.1. =) Envolver o especialista de domínio na construção dos casos de uso seria interessante. [**E1.C2.12**].
- **Resposta do Especialista 2**:
  - Sim [E2.CA1.1];
  - Entretanto, sugiro que o Especialista de Domínio também participe da atividade de Descrever Processos de Negócios. Isso é necessário, pois deve possibilitar uma discussão ampla e mais profunda com o objetivo de obter um melhor entendimento do que irá ou não conter o Modelo Conceitual de Negócio. Cada papel (Analista de Negócios e Especialista em Domínio) tem uma experiência, as quais devem ser trocadas de maneira bilateral e iterativa [**E2.CA2.12**].
- **Resposta do Especialista 3**:
  - Sim, os papéis atribuídos na atividade são suficientes para realizar cada uma das tarefas definidas pois os artefatos gerados ao fim da atividade dependem basicamente desses papéis em um mundo real onde a atividade possa ser realizada [**E3.CA.1**].
- **Resposta do Especialista 4**:
  - Eu considero a adequada a atribuição desse papéis, desde que a interpretação de negócio e domínio estejam bem definidas. Na minha avaliação pareceu adequada por considerar que o negócio representa um conjunto de atividades inseridas em um domínio de atuação, logo estando contidas no domínio. Dessa forma é adequada a presença de dois profissionais especialistas em ramos diferentes [**E4.CA.1**].
  - Entretanto, se considerarmos negócio e domínio como sinônimos, a atribuição pode não ser adequada, a ponto de ser redundantes. Uma sugestão seria apresentar descrições que evidenciem a diferença entre negócio e domínio [**E4.CA2.11**].

- **Resposta do Especialista 5**:
  - Sim [E5.CA1.1];
  - Embora haja uma leve sobrecarga de responsabilidades para o"analista de negócio", percebe-se a necessidade da sua participação em todas as tarefas da atividade de"definição de requisitos". Como forma de dividir melhor a carga, o"especialista de domínio", poderia auxilia-lo na tarefa de"descrever os processos de negócio"[**E5.CA2.12**].
- **Resposta do Especialista 6**:
  - Acredito que o"Especialista em Domínio"poderia ser um ponto de variação, pois o"Analista de negócios"para um projeto baseado em DBC poderia possuir os conhecimentos necessários para os dois papéis [**E6.CA2.13**].
- **Resposta do Especialista 7**:
  - Acredito que o Especialista em Domínio também poderia apoiar a atividade"Identificar variabilidades", juntamente como Arquiteto de Linha de Produto [**E7.CA2.12**].
- **Resposta do Especialista 8**:
  - Considero adequada a separação, independente de como se chame cada papel pois o próprio analista de negócio poderia ter denominações diversas conforme a organização, às vezes sendo denominado analista desenvolvedor, analista de sistemas etc. mas o mais importante é desempenhar as atividades de analista de negócios efetivamente. Não vejo outros envolvidos nesta atividade para este processo [**E8.CA1.1**].

1.3. Você considera que o fluxo das tarefas do `Workflow de Requisitos` contribui de maneira objetiva e precisa para gerar os artefatos *SMarty* (`Modelo de Casos de Uso SMarty` e `Modelo Conceitual de Negócio SMarty`)? Justifique sua resposta.

- **Resposta do Especialista 1**:
  - Sim [**E1.CA1.3**];
  - Pode ter um refinamento para detalhar mais as atividades e papéis dos participantes [**E1.CA2.9, E1.CA2.11**];
  - mas está bem objetiva [**E1.CA1.3**].
- **Resposta do Especialista 2**:

  - Contribui de maneira relevante, considerando o apoio das diretrizes (SMartyProcess) e dos estereótipos (SMartyProfile) contidos na abordagem SMarty. Desse modo, é possível garantir uma modelagem mais adequada em todo o cenário do Workflow de Requisitos [**E2.CA1.3**];
  - Contudo, senti a falta de mais um papel para auxiliar o Arquiteto de Linha de Produto, como por exemplo, um outro tipo de Especialista em Linha de Produto [**E2.CA2.7**].

- **Resposta do Especialista 3**:
  - Sim, pois tanto a atividade Definição de Requisitos quanto a atividade SMartyProcess são baseadas é técnicas pré-definidas e validadas como no processo UML Components e na abordagem SMarty, respectivamente. O fluxo é simples e objetivo, utilizando recursos necessários para gerar os artefatos de saída do workflow [**E3.CA1.3**].

- **Resposta do Especialista 4**:
  - Sim porque o fluxo das tarefas do Workflow de Requisitos considera que uma tarefa ao final do fluxo precisa dos artefatos produzidos em tarefas anteriores deste mesmo fluxo. Logo é possível rastrear a geração dos artefatos de maneira objetiva e lógica. Por exemplo, para a geração do Modelo de Casos de Uso, é necessário que os casos de uso tenham sido identificados. Para a identificação dos casos de uso, é necessário que os processos de negócio tenham sido descritos. Para que tal descrição aconteça, é preciso que os requisitos do negócio estejam definidos.[**E4.CA1.4**].

- **Resposta do Especialista 5**:
  - Sim. As tarefas relativas a respectiva atividade, dados os seus objetivos, contribuem para a geração dos artefatos SMarty mencionados[**E5.CA1.4**];
  - Faria uma única observação, com relação a atividade de"rastrear e controlar variabilidades", visto que o objetivo da mesma não está tão claro como as demais, principalmente no que se trata a"controlar as variabilidades". As dependências dessa tarefa, ilustradas no diagrama, não me convenceram, visto que a"rastreabilidade"deve ser uma tarefa contínua e concomitante [**E5.C2.9**].

- **Resposta do Especialista 6**:

  - Não, para "Desenvolver o Modelo Conceitual de Negócio"é necessário ter executado a tarefa"Descrever Processos de Negócio", não existe a seta pontilhada representando este fluxo, deixando a entender que poderiam ser executadas paralelamente. Meu entendimento é que isto não possa ocorrer [**E6.CA2.9**];
  - Referente as tarefas do SMartyProcess, estas julgo de acordo [**E6.CA1.2**].
- **Resposta do Especialista 7**:
  - Sim, o fluxo das tarefas constrói adequadamente estes artefatos [**E7.CA1.3**].
- **Resposta do Especialsita 8**:
  - Acredito que sim [**E8.CA1.3**];
  - entretanto observei uma predominância nos dois artefatos de i) requisitos, aqui representados pelos casos de uso provenientes das regras de negócio, e ii) modelo conceitual. Por mais importantes que sejam, creio que alguns conceitos da engenharia do domínio talvez não estejam atendidos. Complemento na resposta 1.5 com feature model [**E8.CA2.10**].

1.4. Descreva a compreensão obtida em relação a execução das tarefas e geração dos artefatos no `Workflow de Requisitos`.

- **Resposta do Especialista 1**:
  - A ideia é bem direta. O analista usa os requisitos de software e requisitos de negócio para definir os casos de uso. O especialista de domínio junto do analista definem o modelo conceitual [**E1.CA1.1, E1.CA1.3, E1.CA1.4**].
- **Resposta do Especialista 2**:
  - Entendo que cada tarefa está relacionada com seu papel de maneira correta. O Analista de Negócios descreve os processos e identifica os casos de uso. A partir disso, é possível aplicar o SMartyProcess e melhorar a modelagem por meio da geração de novos artefatos com estereótipos da SMarty utilizando o SMartyProfile. [**E2.CA1.3, E2.CA1.4**];
  - Entretanto, vejo a necessidade de atribuir o papel Especialista de Domínio na tarefa de auxiliar a descrição dos processo de negócios em conjunto com o Analista de Negócios [**E7.CA2.12**];
  - Enfim, os artefatos gerados ao final são bem definidos e expressam o real objetivo deste workflow **E7.CA1.4**].

- **Resposta do Especialista 3**:
  - O workflow possui um fluxo de tarefas bem definido e que gera artefatos essenciais para um processo de desenvolvimento baseado em componentes [**E3.CA1.3**];
  - O fluxo se inicia ao realizar as tarefas Descrever Processos de Negócios (1) e Desenvolver Modelo Conceitual de Negócio (2), essas podendo serem executadas paralelamente. A primeira tarefa é seguida pela tarefa Identificar Casos de Uso (1.1), gerando assim o artefato Modelo de Casos de Uso. Ao mesmo tempo o artefato Modelo Conceitual de Negócio se torna a saída da segunda tarefa (2). Esses dois artefatos gerados inicia a atividade SMartyProcess ao serem utilizados como entrada da tarefa Identificar Variabilidades, que se baseia nas Diretrizes do SMartyProcess. Paralelamente as próximas tarefas são Delimitar Variabilidades e Rastrear e Controlar Variabilidades. Após a finalização dessas duas tarefas, uma nova tarefa Representar Variabilidades irá criar os artefatos Modelo de Casos de Uso SMarty e Modelo Conceitual de Negócios SMarty, com base nos estereótipos do SMartyProfile [**E3.CA1.3, E3.CA1.4**].
- **Resposta do Especialista 4**:
  - A compreensão obtida foi clara, no sentido de reconhecer a importância das atividades para a geração dos artefatos finais (Modelo de Casos de Uso SMarty e Modelo Conceitual de Negócio SMarty) [**E4.CA1.4**];
  - A execução das tarefas apresentadas permite reconhecer a importância de cada uma na geração dos artefatos [**E4.CA1.3**].
- **Resposta do Especialista 5**:
  - Num primeiro momento entram em cena o analista de negócio e o especialista de domínio e modelam o negócio e as funcionalidades propostas para a LPS. Num segundo momento, entra em cena o arquiteto da LPS com a função de identificar e modelar as variabilidades. Produzindo como resultado uma arquitetura de LPS, formalizada através dos artefatos SMarty [**E5.CA1.1, E5.CA1.3**].
- **Resposta do Especialista 6**:
  - Primeiro é identificado e descrito os processos de negócios de forma coloquial, a partir disto é identificado os casos de uso para implementação. Em paralelo é feito um modelo conceitual de negócio

descrevendo como os processos de negócios devem ser componentizados para atender uma LPS [**E6.CA1.4**].

- **Resposta do Especialista 7**:
  - Minha compreensão em relação ao Workflow de Requisitos da SMarty Components, é que ele estende o Workflow de Requisitos do UML Components. Inicialmente utiliza-se os Requisitos de Negócio para Descrever os Processos de Negócio e Identificar os Casos de Uso, além de usar o Conhecimento do Domínio da LPS como entrada para Desenvolver o Modelo Conceitual de Negócio. Após gerados o Modelo de Casos de Uso e o Modelo Conceitual de Negócio, estes são entrada para Identificar as Variabilidades. Após isso são realizadas as tarefas: Delimitar Variabilidades, Rastrear e controlar Variabilidades e Representar variabilidades [**E7.CA1.4**].
- **Resposta do Especialista 8**:
  - Se entendi corretamente a pergunta, entendo que as atividades visam receber os requisitos de alto nível (algo como casos de uso de negócio), agrupando-os com as diretrizes para constituição de uma linha de produto de software e incorporar variabilidades e similaridades para compor artefatos reutilizáveis de tal forma que pudessem ser incorporados em uma linha de produtos de software [**E8.CA1.4**].

1.5. Quais elementos (artefatos, papéis ou tarefas) você entende que poderiam ser alterados e/ou removidos do `Workflow de Requisitos` sem comprometer o objetivo final que é a especificação de uma arquitetura componentizada de LPS? Justifique sua resposta.

- **Resposta do Especialista 1**:
  - Acabei já respondendo a pergunta nas anteriores [**E1.CA2.8, E1.CA2.9, E1.CA2.11, E1.CA2.12**].
- **Resposta do Especialista 2**:
  - O Analista de Negócios e o Especialista de Domínio poderiam ser unidos em um único papel, como por exemplo, Analista de Negócios/Domínio ou Arquiteto de Software. Assim, um padrão seria mantido como apresentado no SMartyProcess que contém apenas um papel denominado como Arquiteto de Linha de Produto [**E2.CA2.12**];
  - Os Requisitos de Negócio e o Conhecimento do Domínio da LPS poderiam estar relacionados de maneira bilateral para possibilitar uma

melhor compreensão do que realmente o cliente necessita considerando um possível novo projeto de software [**E2.CA2.12**].

- **Resposta do Especialista 3**:
  - O conceito do modelo conceitual de negócio nem sempre é descrito no mundo real, entretanto acredito ser essencial para uma arquitetura como esta [**E3.CA1.4**];
  - Os demais elementos são essenciais.Acredito por fim que os elementos utilizados estão bem definidos e há necessidade de mudanças [**E3.CA1.4**].
- **Resposta do Especialista 4**:
  - Pela imagem apresentada e pela descrição dos papéis do SMartyComponents, penso que não ficou claro o papel do Especialista do Domínio [**E4.CA2.11**];
  - Pela descrição apresentada, tal especialista deve colaborar com o desenvolvimento do Modelo Conceitual de Negócios, de acordo com as informações apresentadas a respeito do conhecimento de domínio da LPS. A atividade do especialista é traduzir essas informações para o modelo conceitual simplesmente ou adaptá-las segundo o seu conhecimento ? Caso seja apenas a tradução, tal papel pode ser retirado e suas atribuições serem repassadas ao analista, visto que todo o conhecimento necessário está contido no documento [**E4.CA2.12**];
  - Caso a adaptação seja necessária, é justificado a presença deste especialista [**E4.CA2.11**].
- **Resposta do Especialista 5**:
  - A única preocupação que me surgiu ao analisar o processo proposto é que a identificação de variabilidades iniciou apenas em uma segunda etapa (tarefa), e tomou por base o resultado de uma primeira etapa (tarefa) [**E5.CA2.9**];
  - Essa abordagem pode limitar a identificação de todas as possíveis variabilidades relativas ao domínio proposto. Pois, pode fazer com que o arquiteto da LPS se limite aos cenários que foram idealizados pelo analista de negócio e especialista do domínio; ou então a segunda tarefa vai gerar a necessidade de que a primeira tarefa seja executada novamente (para cobrir possíveis gaps que tenham sido deixados) [**E5.CA2.9, E5.CA2.10**].

- **Resposta do Especialista 6**:
  - Minha sugestão seria que, para"Desenvolver o Modelo Conceitual de Negócio"é necessário ter executado a tarefa"Descrever Processos de Negócio", não existe a seta pontilhada representando este fluxo, deixando a entender que poderiam ser executadas paralelamente. Meu entendimento é que isto não possa ocorrer [**E6.CA2.12**].
- **Resposta do Especialista 7**:
  - Minha única sugestão de alteração é a que já mencionei na questão 1.2: o Especialista em Domínio também poderia apoiar a atividade"Identificar variabilidades", juntamente como Arquiteto de Linha de Produto [**E7.CA2.12**];
- **Resposta do Especialista 8**:
  - Eu não retiraria nenhum artefato ou papel [**E8.CA1.1, E8.CA1.4**];
  - mas acho que seria importante considerar, mesmo que parcialmente, uma representação como a de feature model [**E8.CA2.10**].

2. ***Workflow* de Requisitos**

   2.1. Qual(quais) a(s) maior(es) complexidade(s)/esforço percebidas no `Workflow de Requisitos`? Justifique sua resposta.

   - **Resposta do Especialista 1**:
     - Dentro do workflow são três tarefas, dois papéis e as depedências entre eles. Não vi algo com grande complexidade [**E1.CA1.5**].
   - **Resposta do Especialista 2**:
     - Compreender em que nível de abstração estão o primeiro Modelo de Casos de Uso e o Modelo Conceitual de Negócio e até onde é necessário eu aplicá-los utilizando o SMartyProcess.Na saída são específicados modelos ou diagramas [**E2.CA2.10**];
     - Sugiro que os Produtos de Trabalho = Modelo de Casos de Uso e Modelo Conceitual de Negócio sejam unidos um único Produto de Trabalho para tornar a compreensão de que serão, após, modelados utilizando o SMartyProcess [**E2.CA2.14**].
   - **Resposta do Especialista 3**:
     - Esse workflow é bastante simples e compreende atividades básicas para qualquer procedimento de desenvolvimento de software, independente de ser baseado em componentes [**E3.CA1.5**];

  - O maior esforço fica a cargo do conhecimento necessário sobre a atividade SMartyProcess. Essa atividade envolve diversos conceitos que precisam ser entendidos de forma correta para aplicar variabilidade em artefatos como o Modelo de Casos de Uso e o Modelo Conceitual de Negócio [**E3.CA2.20**].

- **Resposta do Especialista 4**:
  - A maior complexidade percebida foi quanto ao papel do Especialista de Domínio no desenvolvimento do Modelo Conceitual do Negócio, pois não ficou evidente a sua função, visto que existia também um documento contendo informações da LPS. Diante disso, surgiu o questionamento do tipo de trabalho realizado por esse especialista (tradução ou adaptação) [**E4.CA2.11**].
- **Resposta do Especialista 5**:
  - A tarefa de"identificação das variabilidades"acabou se tornando o coração do referido workflow. Dois motivos baseiam essa importância: (i) se tornar por base apenas o conteúdo dos artefatos produzidos, pode deixar de fora variabilidades relevantes para a LPS sendo modelada; e (ii) identificando possíveis variabilidades que não foram contempladas na primeira tarefa, ou ele faz o trabalho do analista de negócio e especialista do domínio, ou suscita que a primeira tarefa seja realizada [**E5.CA1.5**].
- **Resposta do Especialista 6**:
  - A não visualização do resultado com as variabilidades deixa um pouco confuso, mas entendo que este seria outro nível de abstração. O maior esforço foi na compreensão do propósito de cada papel, tarefa ou artefato [**E6.CA2.9, E6.CA2.10, E6.CA2.11**].
- **Resposta do Especialista 7**:
  - Dependendo da forma com que esteja descrito o Conhecimento do Domínio da LPS, a atividade Desenvolver Modelo Conceitual de Negócio vai ter uma complexidade maior [**E7.CA2.10**];
  - De maneira análoga, Descrever Processos de Negócio vai demandar grande esforço caso os Requisitos de Negócio não estejam bem descritos ou os processos de negócio não sejam bem conhecidos [**E7.CA2.10**].
- **Resposta do Especialista 8**:

  - A tarefa"Desenvolver Modelo Conceittual de Negócio"a partir do conhecimento do domínio da LPS creio que seja um desafio e precisa ser bastante explícito para quem for estressar o processo [**E8.CA2.9**];
  - e a tarefa"Identificar Variabilidades"pela extrema importância no sucesso do modelo conceitual resultante que, neste processo, servirá como base para a LPS [**E8.CA2.9**].

2.2. Quais elementos (artefatos, papéis ou tarefas) você removeria, adicionaria ou alteraria do `Workflow de Requisitos`? Justifique sua resposta.

- **Resposta do Especialista 1**:
  - Adicionaria uma tarefa de"validar modelo conceitual com modelo de casos de uso"[**E1.CA2.8**];
  - e faria o especialista de domínio participar da elaboração e validação dos casos de uso [**E1.CA2.12**].
- **Resposta do Especialista 2**:
  - O Analista de Negócios e o Especialista de Domínio poderiam ser unidos em um único papel, como por exemplo, Analista de Negócios/Domínio ou Arquiteto de Software. Assim, um padrão seria mantido como apresentado no SMartyProcess que contém apenas um papel denominado como Arquiteto de Linha de Produto [**E2.CA2.12**];
  - Os Requisitos de Negócio e o Conhecimento do Domínio da LPS poderiam estar relacionados de maneira bilateral para possibilitar uma melhor compreensão do que realmente o cliente necessita considerando um possível novo projeto de software [**E2.CA2.14**].
- **Resposta do Especialista 3**:
  - Pensando no Processo Unificado, no UML Components e no SMarty acredito que nenhum elemento deveria ser alterados, adicionado ou removido. Os elementos apresentados são suficientes para uma arquitetura genérica que possa ser aplicada a qualquer domínio [**E3.CA1.5**];
  - embora específicos domínios possam necessitar de artefatos ou tarefas específicas dependendo do projeto a ser realizado. Entretanto isso nem sempre é possível prever [**E3.CA2.9**].
- **Resposta do Especialista 4**:
  - Eu considero que os artefatos, papéis e tarefas estão coerentes com o Workflow de Requisitos e são suficientes para os artefatos gerados por este workflow [**E4.CA1.1, E4.CA1.2, E4.CA1.4, E4.CA1.5**].

- **Resposta do Especialista 5**:
  - Uma forma de minimizar os possíveis problema que mencionei em questões anteriores, seria a participação do arquiteto da LPS em alguma das tarefas da tarefa de"definição de requisitos", como a"identificação de casos de uso". Dessa forma as variabilidades seriam consideradas mais cedo, mesmo que só fossem modeladas em um segundo momento. Tudo o que seria necessário para a futura modelagem da variabilidade seria produzido antes [**E5.CA2.12**].
- **Resposta do Especialista 6**:
  - Minha sugestão seria que, para "Desenvolver o Modelo Conceitual de Negócio"é necessário ter executado a tarefa"Descrever Processos de Negócio", não existe a seta pontilhada representando este fluxo, deixando a entender que poderiam ser executadas paralelamente. Meu entendimento é que isto não possa ocorrer [**E6.CA2.9**].
- **Resposta do Especialista 7**:
  - Não removeria nenhum [**E7.CA1.5**].
- **Resposta do Especialista 8**:
  - Conforme anteriormente apresentado, creio que apenas o relacionamento com um feature model poderia ser interessante [**E8.CA2.9**].

3. ***Workflow* de Especificação - Atividade: Identificação de Componente**

   3.1. Você considera a atividade `Identificação de Componente` do *SMartyComponents* adequada em relação aos artefatos gerados (`Espec. de Componente & Arquitetura`, `Interfaces do Sistema`, `Interfaces do Negócio` e `Modelo de Tipo de Negócio`) neste *workflow*? Justifique sua resposta.

- **Resposta do Especialista 1**:
  - As atividades são gerais e adequadas. Só parece que as atividades são 1:1 com os outputs. Para cada output desejado, temos uma atividade no diagrama [**E1.CA3.2**].
- **Resposta do Especialista 2**:
  - Sim, pois as tarefas são claras e estão relacionadas de forma a contribuírem na corretude para a geração dos artefatos [**E2.CA3.2, E2.CA3.4**];

  - No entanto, um novo artefato (atividade) poderia ser gerado entre o Modelo de Tipo de Negócio e as Interfaces de Negócio, ou então, poderiam ser unidos em um único artefato. Desse modo, poderia propiciar uma ponte entre os dois de forma a permitir verificar o que é realmente adequado para uma determinada especificação [**E2.CA4.6, E2.CA4.14**].

- **Resposta do Especialista 3**:
  - Sim, pois segue o mesmo fluxo de tarefas definidas no UML Componentes, tornando-se assim um atividade já validada ao passar dos anos [**E3.CA3.4**].
- **Resposta do Especialista 4**:
  - Sim porque os artefatos gerados estão claramente relacionados aos componentes, contribuindo com a criação dos mesmos. Considerando esta relação, é adequado que tais artefatos sejam definidos justamente nesta atividade [**E4.CA3.4**].
- **Resposta do Especialista 5**:
  - Sim, as tarefas propostas, segundo os seus objetivos, têm condições de produzir os artefatos propostos para a atividade [**E5.CA3.2**];
  - Embora, precise haver uma clara distinção entre o que é definido como"interface de negócio"e como"interface de o sistema", para não haver confusão e mal entendidos [**E5.CA4.10**].
- **Resposta do Especialista 6**:
  - "Interface de Negócio"confunde um pouco com"Interface de Sistema"[**E6.CA4.10**
  - consigo distinguir a diferença, mas penso que as interfaces de negócio serviriam de entrada para a produção das interfaces do sistema [**E6.CA4.14**].
- **Resposta do Especialista 7**:
  - Sim, considero adequada [**E7.CA3.4**];
- **Resposta do Especialista 8**:
  - Considero adequada a atividade e os resultados mapeados integralmente da abordagem UML components para SmartyProcess, com o sufixo"Smarty"[**E8.CA3.4**];
  - entretanto sem a definição do processo detalhada é difícil afirmar integralmente a sua validade [**E8.CA4.10**].

3.2. Você considera adequada a atribuição do papel (`Arquiteto de Linha de Produto`) envolvido na atividade `Identificação de Componente` do *SMarty-Components*? Justifique sua resposta.

- **Resposta do Especialista 1**:
  - Sim. Ele que deve realizar as atividades mostradas no diagrama [**E1.CA3.1**];
- **Resposta do Especialista 2**:
  - Sim [**E2.CA3.1**];
  - Contudo, sugiro que o Especialista de Domínio auxilie ou apoie o Arquiteto de Linha de Produto durante a tarefa Desenvolver Modelo de Tipo de Negócio. Acredito que isso possa colaborar com uma melhor modelagem da atividade de output referente a atividade Modelo de Tipo de Negócio [**E2.CA4.7**];
- **Resposta do Especialista 3**:
  - Sim [**E3.CA3.1**];
  - Não caso este papel não tenha conhecimento sobre desenvolvimento baseado em componentes. Não consegui identificar se este arquiteto se encontra em alguma documentação sobre UML componentes, mas para realizar as tarefas ele deveria ter esse conhecimento sobre desenvolvimento baseado em componentes, porém a nomenclatura não aponta este fator [**E3.CA4.7**].
- **Resposta do Especialista 4**:
  - Sim porque o arquiteto pode organizar de forma adequada as entidades que formam um ou mais componentes, dada as suas informações. Essas informações são essenciais desde a especificação do negócio (Desenvolver Modelo de Tipo do Negócio) até a especificação da Arquitetura (Criar Espec. Inicial Comp & Arquitetura) [**E4.CA3.1**].
- **Resposta do Especialista 5**:
  - Em parte. Seria interessante que um arquiteto de software participasse dessa tarefa [**E5.CA4.7**];
  - embora que a participação do arquiteto da LPS seja fundamental para contextualizar o trabalho. Embora o trabalho desses dois papéis tenha as suas similaridades, visam atender a objetivos claramente distintos [**E5.CA4.7, E5.CA4.12**].

- **Resposta do Especialista 6**:
  - Sim, adequada e suficiente [**E6.CA3.1**].
- **Resposta do Especialista 7**:
  - Sim, considero adequada [**E7.CA3.1**].
- **Resposta do Especialista 8**:
  - Considero adequado, [**E8.CA3.1**];
  - entretanto creio que mesmo sem estar no método original da UML Components, o especialista do domínio poderia ter uma participação mesmo que para validação dos entregáveis nesta atividade [**E8.CA4.7**].

3.3. Você considera que o fluxo das tarefas da atividade `Identificação de Componente` do `Workflow de Especificação` contribui de maneira objetiva e precisa para gerar os artefatos *SMarty* (`Interfaces do Sistema SMarty`, `Espec. de Componente & Arquitetura SMarty`, `Interfaces de Negócio SMarty` e `Modelo e Tipo de Negócio SMarty`)? Justifique sua resposta.

- **Resposta do Especialista 1**:
  - Sim. O fluxo é claro e sucinto [**E1.CA3.3**].
- **Resposta do Especialista 2**:
  - Contribui de maneira relevante, considerando o apoio das diretrizes (SMartyProcess) e dos estereótipos (SMartyProfile) contidos na abordagem SMarty. Desse modo, é possível garantir uma modelagem mais adequada em todo o cenário de Identificação de Componente [**E2.CA3.3**];
  - Contudo, senti a falta de mais um papel para auxiliar o Arquiteto de Linha de Produto, como por exemplo, um outro tipo de Especialista em Linha de Produto [**E2.CA4.7**].
- **Resposta do Especialista 3**:
  - Sim, pois enquanto os artefatos de saída forem modelados com UML, provavelmente será possível aplicar o SMarty aos artefatos [**E3.CA3.3**].
- **Resposta do Especialista 4**:
  - Sim porque o fluxo de tarefas possibilita observar a sequência de atividades necessárias à geração dos artefatos [**E4.CA3.3**].
- **Resposta do Especialista 5**:

  - Em parte. Segundo o processo proposto, a modelagem de uma variabilidade como um componente fica prejudicada, pois a tarefa de identificação das variabilidades acontece após da identificação dos componentes e suas interfaces [**E5.CA4.13**].

- **Resposta do Especialista 6**:
  - Não, a identificação das interfaces de negócio deveria ser executada antes da definição das interfaces de sistema, e no modelo esta em paralelo [**E6.CA4.13**];
  - Outro ponto é na verificação das interfaces existentes, esta tarefa deveria ser executada na etapa seguinte, provisionamento [**E6.CA4.11**].
- **Resposta do Especialista 7**:
  - Sim, o fluxo das tarefas parece contribuir adequadamente [**E7.CA3.3**].
- **Resposta do Especialista 8**:
  - Considero que o fluxo está objetivo, [**E8.CA3.3**];
  - mas ainda assim pensaria em agrupar duas ou três tarefas mas fazê-las na linha de cada artefato, por exemplo, analisar e representar variabilidades de interfaces do sistema, analisar e representar variabilidades de espec. de comp. e assim por diante [**E8.CA4.13**].

3.4. Descreva a compreensão obtida em relação a execução das tarefas e geração dos artefatos na atividade `Identificação de Componente` do `Workflow de Especificação`.

- **Resposta do Especialista 1**:
  - O fluxo claro. Como escrevi na questão 2.1, a relação atividades/outputs está 1:1 praticamente [**E1.CA3.3**].
- **Resposta do Especialista 2**:
  - Entendo que cada tarefa está relacionada com seu papel de maneira correta [**E2.CA3.1, E2.CA3.2**];
  - O Arquiteto de Linha de Produto descreve os processos e identifica o que é necessário, como as interfaces de negócio e do sistema, além da especificação. A partir disso, é possível aplicar o SMartyProcess e melhorar a modelagem por meio da geração de novos artefatos com estereótipos da SMarty utilizando o SMartyProfile [**E2.CA3.3**];

  - Entretanto, vejo a necessidade de atribuir o papel Analista de Negócios na tarefa de auxiliar na descrição da tarefa Desenvolver Modelo de Tipo de Negócio [**E2.CA4.7**];
  - Enfim, os artefatos gerados ao final são bem definidos e expressam o real objetivo deste workflow [**E2.CA3.4**].

- **Resposta do Especialista 3**:
  - Inicialmente o papel Arquiteto de Linha de Produto realiza paralelamente as tarefas Identificar Interfaces de Sistema e Operações e Desenvolve Modelo de Tipo de Negócio. A primeira tarefa gera o artefato Interfaces do Sistema. A segunda tarefa gera o artefato Modelo de Tipo de Negócio é seguida pela tarefa Identificar Interfaces de Negócio que gera o artefato Interfaces do Negócio. Por fim é executada a tarefa Criar Espec. Inicial Comp. & Arquitetura que gera o artefato Espec. de Componente & Arquitetura [**E3.CA3.2, E3.CA3.3, E3.CA3.4**].
- **Resposta do Especialista 4**:
  - A compreensão obtida foi que a partir dos artefatos de entrada e após a realização de certas atividades, artefatos de saída são gerados [**E4.CA3.3, E4.CA3.4**].
- **Resposta do Especialista 5**:
  - O arquiteto da LPS, utilizando os artefatos gerados na atividade anterior, procede com a identificação das interfaces, que irão dar origem aos componentes - os quais irão compor a arquitetura da LPS [**E5.CA3.3**];
  - Num passo seguinte, são identificadas as variabilidades, as quais são delimitadas e representadas. Gerando por fim os artefatos que definem a arquitetura da LPS proposta [**E5.CA3.3**].
- **Resposta do Especialista 6**:
  - Primeiro é identificado e desenvolvido um modelo de tipos de negócios, depois a identificação das interfaces de negócio [**E6.CA3.2, E6.CA3.3**];
  - a seguir as interfaces de sistema e por fim a definição completa dos componentes junto com a arquitetura utilizada [**E6.CA3.3**].
- **Resposta do Especialista 7**:

  - Com base no Modelo de Casos de Uso SMarty, inicia-se a identificação das interfaces do sistema e suas operações, e cria-se uma especificação inicial dos componentes e da arquitetura [**E7.CA3.3, E7.CA3.4**];
  - Com base no Modelo Conceitual de Negócio SMarty, inicia-se o desenvolvimento do Modelo de Tipo de Negócio e a identificação das Interfaces de Negócio [**E7.CA3.3, E7.CA3.4**].
- **Resposta do Especialista 8**:
  - Incorporação de variabilidades e, possivelmente, consideração das similaridades, nos artefatos produzidos na primeira etapa de identificação de componentes do modelo do domínio, para subsidiar a criação de linhas de produto de software [**E8.CA3.4**].

3.5. Quais elementos (artefatos, papéis ou tarefas) você entende que poderiam ser alterados e/ou removidos da atividade `Identificação de Componente` do `Workflow de Especificação` sem comprometer o objetivo final que é a especificação de uma arquitetura componentizada de LPS? Justifique sua resposta.

- **Resposta do Especialista 1**:
  - Um refinamento nas atividades poderia ser interessante para de fato contribuir na subdivisão de tarefas para obter determinado output. Nos casos intermediários acontece isso (como no desenvolver-identificar-criar), mas o objetivo final dessas três atividades é a arquitetura. Refinando, principalmente a parte de criar arquitetura, seria interessante [**E1.CA4.13**];
- **Resposta do Especialista 2**:
  - Somente sugiro que seja criada uma outra atividade entre o Modelo de Tipo de Negócio e as Interfaces de Negócio [**E2.CA4.8**];
  - bem como entre Modelo de Tipo de Negócio SMarty e as Interfaces de Negócio SMarty [**E2.CA4.8**];
  - Poderiam estar relacionados de maneira bilateral para possibilitar uma melhor compreensão do que realmente o cliente necessita considerando um possível novo projeto de software [**E2.CA4.13**].
- **Resposta do Especialista 3**:
  - Os artefatos e tarefas seguem a UML Components e o SMarty [**E3.CA3.3**];
  - Talvez o papel Arquiteto de Linha de Produto possa ser alterado por um Arquiteto de Componentes [**E3.CA4.7**].

- **Resposta do Especialista 4**:
  - Não ficou claro inicialmente a função do artefato"Modelo de Tipo de Negócio"[**E4.CA4.10**];
  - Logo, tal artefato poderia ter sua identificação (nome) alterada, de forma a apresentar de maneira mais intuitiva a sua função [**E4.CA4.14**];
- **Resposta do Especialista 5**:
  - (1) participação de um arquiteto de software, como responsável pela tarefa de"identificar interfaces de sistema e operações"e como auxiliar nas demais tarefas de"identificação de componentes"[**E5.CA4.7**];
  - (2) Talvez caracterizar"interfaces"como externas (com outros sistemas) e internas (orientadas ao negócio) [**E5.CA4.10**];
  - (3) a tarefa de"rastrear e controlar variabilidades"aparenta não estar coerente com o fluxo proposto, talvez porque não esteja tão claro o seu objetivo - embora eu esteja convencido da necessidade do rastreamento das variabilidades [**E5.CA4.13**].
- **Resposta do Especialista 6**:
  - Poderia remover a verificação das interfaces existentes [**E6.CA4.15**];
  - e mudar o fluxo da identificação das interfaces de negócios para ser executado antes das interfaces de sistema [**E6.CA4.14**].
- **Resposta do Especialista 7**:
  - Não considero que nenhum elemento deva ser alterado [**E7.CA3.5**].
- **Resposta do Especialista 8**:
  - Não vejo necessidade de alterações, apenas apontei anteriormente uma abordagem que poderia ser diferente no smartyProcess [**E7.CA4.13**].

4. ***Workflow* de Especificação - Atividade: Interação de Componente**

   4.1. Você considera a atividade `Interação de Componente` do *SMartyComponents* adequada em relação aos artefatos gerados (`Espec. de Componente & Arquitetura SMarty` e `Especificação de Interfaces`) neste workflow? Justifique sua resposta.

- **Resposta do Especialista 1**:
  - Não bem do que é composta a arquitetura smarty [**E1.CA6.10**];
  - mas imagino que esteja adequada pelas entradas dadas [**E1.CA5.4**].

- **Resposta do Especialista 2**:
  - Sim, pois as tarefas estão organizadas e estão relacionadas de forma a contribuírem na corretude para a geração dos artefatos [**E2.CA5.2, E2.CA5.4**];
  - Entretanto existem exceções, pois não entendo por que a tarefa Descobrir Operações de Negócio está associada como um input para a atividade Interfaces do Sistema SMarty. Não está claro o objetivo disso. Sugiro que reveja esta associação [**E2.CA6.9**].
- **Resposta do Especialista 3**:
  - Sim, pois é totalmente baseado no UML Components [**E3.CA5.4**].
- **Resposta do Especialista 4**:
  - Sim porque os artefatos gerados apresentam características importantes para a interação entre componentes, como as interfaces e arquitetura. Considerando esta importância, é adequado que a geração destes artefatos aconteça nesta atividade [**E4.CA5.4**].
- **Resposta do Especialista 5**:
  - Em parte. Visto que a tarefa de"interação de componentes"só realiza um REFINAMENTO nas interfaces e componentes (produzidos na atividade anterior), não fica clara a necessidade desta tarefa. Já a segunda tarefa repete as tarefas que já foram realizadas na atividade anterior. Creio que a necessidade dessa tarefa carece de uma melhor justificativa [**E5.CA6.9**].
- **Resposta do Especialista 6**:
  - Sim, adequada [**E6.CA5.4**].
- **Resposta do Especialista 7**:
  - Sim, adequada [**E7.CA5.4**].
- **Resposta do Especialista 8**:
  - Idem a atividade anterior de identificação, considero adequada [**E8.CA5.4**].

4.2. Você considera adequada a atribuição do papel (`Arquiteto de Linha de Produto`) envolvido na atividade `Interação de Componente` do *SMartyComponents*? Justifique sua resposta.

- **Resposta do Especialista 1**:
  - Sim. Ele que deve desempenhar essas atividades [**E1.CA5.1**].

- **Resposta do Especialista 2**:
  - Sim [**E2.CA5.1**];
  - Contudo, sugiro a especificação de outro papel, como um Engenheiro de Software, para auxiliar ou apoiar o Arquiteto de Linha de Produto durante as tarefas em geral. Acredito que isso possa colaborar com uma melhor modelagem das atividade output: Espec. de Componente & Arquitetura SMarty e Especificação de Interfaces [**E2.CA6.7**].
- **Resposta do Especialista 3**:
  - Não por conta da nomenclatura pois é necessário conhecimento sobre desenvolvimento baseado em componentes para realizar as tarefas definidas. Sim se o papel possuir este conhecimento [**E3.CA6.12**].
- **Resposta do Especialista 4**:
  - Sim, com um anexo de que o papel do Arquiteto de Linha de Produto é essencial na minha opinião para o bom andamento desta atividade. O arquiteto pode contribuir com informações muitas vezes não explicitadas nos artefatos de entrada. Isso acontece porque situações de experiência no domínio e em LPS não são facilmente transmitidas em documentos, representado pelos artefatos de entrada [**E4.CA5.1**].
- **Resposta do Especialista 5**:
  - Como na atividade anterior, creio que poderia participar também um arquiteto de software para trabalhar em conjunto e complementar a visão do arquiteto da LPS - mesma intenção, mas objetivos distintos [**E5.CA6.12**].
- **Resposta do Especialista 6**:
  - Sim, adequada [**E6.CA5.1**].
- **Resposta do Especialista 7**:
  - Sim, o papel está adequado [**E7.CA5.1**];
  - Porém em relação a descobrir operações de negócio,o analista de negócios ou especialista no domínio não poderiam contribuir? [**E7.CA6.7**].
- **Resposta do Especialista 8**:
  - Considero adequada e não vejo necessidade de especialista do domínio nesta atividade [**E8.CA5.1**].

4.3. Você considera que o fluxo das tarefas da atividade `Interação de Componente` do `Workflow de Especificação` contribui de maneira objetiva e precisa para

gerar os artefatos *SMarty* (`Espec. de Componente & Arquitetura SMarty` e `Interfaces SMarty`)? Justifique sua resposta.

- **Resposta do Especialista 1**:
  - Sim [**E1.CA5.3**].
- **Resposta do Especialista 2**:
  - Contribui de maneira relevante, considerando o apoio das diretrizes (SMartyProcess) e dos estereótipos (SMartyProfile) contidos na abordagem SMarty. Desse modo, é possível garantir uma modelagem mais adequada em todo o cenário de Interação de Componente [**E2.CA5.3**];
  - No entanto, senti a falta de mais um papel para auxiliar o Arquiteto de Linha de Produto, como por exemplo, um outro tipo de Especialista em Linha de Produto [**E2.CA6.7**].
- **Resposta do Especialista 3**:
  - Sim, caso os artefatos gerados seja modelados com UML de forma que possa ser aplicado SMarty nesses artefatos [**E3.CA5.4**].
- **Resposta do Especialista 4**:
  - Sim, o fluxo de tarefas permite que cada tarefa tenha um único objetivo, sendo o resultado desta tarefa, entrada para outra tarefa [**E4.CA5.2**];
  - O fluxo de tarefas observado na atividade contribui para rastrear o processo. Essa rastreabilidade facilita o entendimento da geração dos artefatos, no que se refere a objetividade e precisão dos mesmos [**E4.CA5.2**].
- **Resposta do Especialista 5**:
  - Sim, de forma semelhante a atividade anterior, visto que não há uma distinção patente entre estas [**E5.CA5.2**].
- **Resposta do Especialista 6**:
  - Sim, clara e objetiva. [**E6.CA5.3**].
- **Resposta do Especialista 7**:
  - Sim, parece contribuir de maneira objetiva [**E7.CA5.3**].
  - No entanto, a tarefa Descobrir operações de negócio vai requerer apoio de outros artefatos ou outros conhecimentos não listados nos artefatos de entrada do processo [**E7.CA6.10**].

- **Resposta do Especialista 8**:
  - Considero adequado mas a consideração do item 3.3 é válido para esta atividade, assim como para a seguinte, provavelmente 5.3 [**E8.CA5.3, E8.CA6.13**].

4.4. Descreva a compreensão obtida em relação a execução das tarefas e geração dos artefatos na atividade `Interação de Componente` do `Workflow de Especificação`.

- **Resposta do Especialista 1**:
  - O fluxo é bem simples. O arquiteto realizar três atividades sequenciais e que dependem de três artefatos diferentes para serem realizadas para produzir os outputs [**E1.CA5.2**].
- **Resposta do Especialista 2**:
  - Entendo que cada tarefa está relacionada com seu papel de maneira correta [**E2.CA5.1, E2.CA5.2**];
  - O Arquiteto de Linha de Produto descreve os processos e identifica o que é necessário. A partir disso, é possível aplicar o SMartyProcess e melhorar a modelagem por meio da geração de novos artefatos com estereótipos da SMarty utilizando o SMartyProfile [**E2.CA5.4**];
  - Entretanto, vejo a necessidade de atribuir o papel Analista de Negócios na tarefa de auxiliar na descrição da tarefa Descobrir Operações de Negócio [**E2.CA6.7**];
  - Enfim, os artefatos gerados ao final são bem definidos e expressam o real objetivo deste workflow [**E2.CA5.4**].
- **Resposta do Especialista 3**:
  - Um sequência de tarefas em única via é realizada pelo papel Arquiteto de Linha de Produto. Iniciando pela tarefa Descobrir Operações de Negócio, passando pela tarefa Refinar Interfaces & Operações que gera o artefato Especificação de Interfaces, e por fim a tarefa Refinar Espec. de Componentes & Arquitetura é realizada, gerando o artefato Espec. de Componente & Arquitetura [**E3.CA5.1, E3.CA5.2, E3.CA5.4**].
- **Resposta do Especialista 4**:
  - A compreensão obtida foi que a execução das tarefas visava especificar o resultado de uma tarefa anterior, permitindo que cada tarefa pudesse ter um único objetivo [**E4.CA5.2, E4.CA5.4**];

  - Nesta sucessão de especificações, é possível compreender a importância do arquiteto para o refinamento dos resultados das atividades e geração do artefato de saída final [**E4.CA5.1**].

- **Resposta do Especialista 5**:
  - O arquiteto da LPS realiza a identificação das operações de negócio e o consequente refinamento nos componentes, suas interfaces e arquitetura. Num segundo momento são ajustadas as variabilidades modeladas com base nos componentes (realizada na atividade anterior) [**E5.CA5.1, E5.CA5.2, E5.CA5.4**].
- **Resposta do Especialista 6**:
  - A partir das interfaces de negócio e sistema, descobrir as operações de negócio, a partir disso refinar as interfaces & operações e depois os componentes & arquitetura [**E6.CA5.2, E6.CA5.3**].
- **Resposta do Especialista 7**:
  - Com base nas Interfaces de Sistema SMarty e as Interfaces de Negócio SMarty, são definidas as operações de negócio [**E7.CA5.4**];
  - As interfaces e operações identificadas são refinadas, criando um artefato Interfaces SMarty [**E7.CA5.2, E7.CA5.4**].
- **Resposta do Especialista 8**:
  - Assim como na resposta 2.4, estou fazendo referência apenas à complementação do SmartyProcess, que insere características de LPS no método originalmente proposto com os artefatos de especificação de componentes e arquitetura e especificação de interfaces [**E8.CA5.2, E8.CA5.4**];
  - Vale a ressalva que o output da atividade de interação de componentes antes do smartyProcess não deve ter o sufixo"Smarty"no artefato de saída [**E8.CA6.10**].

4.5. Quais elementos (artefatos, papéis ou tarefas) você entende que poderiam ser alterados e/ou removidos da atividade `Interação de Componente` do `Workflow de Especificação` sem comprometer o objetivo final que é a especificação de uma arquitetura componentizada de LPS? Justifique sua resposta.

- **Resposta do Especialista 1**:
  - Sem saber com detalhes o que é uma arquitetura Smarty é complicado dizer. o diagrama deste questionário parece mais refinado e melhor do que o anterior [**E1.CA5.5**].

- **Resposta do Especialista 2**:
  - Somente sugiro que seja criada uma outra atividade entre o Espec. de Componente & Arquitetura SMarty e Especificação de Interfaces, bem como entre Espec. de Componente & Arquitetura SMarty e as Interfaces SMarty. Poderiam estar relacionados de maneira bilateral para possibilitar uma melhor compreensão do que realmente o cliente necessita considerando um possível novo projeto de software [**E2.CA6.9**].
- **Resposta do Especialista 3**:
  - Novamente acredito que a nomenclatura do papel não leva em consideração o conhecimento sobre desenvolvimento baseado em componentes. Talvez um novo papel seja necessário [**E3.CA6.12**].
- **Resposta do Especialista 4**:
  - Não verifiquei artefatos que pudessem ser alterados ou removidos nesta atividade. Todos os artefatos, papéis e tarefas demonstraram intuitivamente suas funções. Sabendo disso, foi possível analisar o processo e não enxergar aparentemente, a necessidade de remoção ou alteração de algo [**E4.CA5.5**].
- **Resposta do Especialista 5**:
  - A proposta seria a inclusão do papel de arquiteto de software, somando esforços com o arquiteto da LPS no desenvolvimento das tarefas relativas a essa atividade [**E5.CA6.7**].
- **Resposta do Especialista 6**:
  - Nesta atividade acredito estarem todos de acordo [**E6.CA5.5**].
- **Resposta do Especialista 7**:
  - Não sugiro nenhuma alteração [**E7.CA5.5**].
- **Resposta do Especialista 8**:
  - Considero a abordagem adequada [**E8.CA5.5**].

5. **Workflow de Especificação - Atividade: Especificação de Componente**

   5.1. Você considera a atividade `Especificação de Componente` do *SMartyComponents* adequada em relação aos artefatos gerados (`Especificação de Arquitetura`, `Especificação de Componentes` e `Interfaces SMarty`) neste workflow? Justifique sua resposta.

- **Resposta do Especialista 1**:
  - Sim. Com a divisão de atividades e passos intermediários para obter outputs [**E1.C7.4**].
- **Resposta do Especialista 2**:
  - Sim, pois as tarefas estão organizadas e estão relacionadas de forma a contribuírem na corretude para a geração dos artefatos [**E2.C7.2, E2.C7.4**].
- **Resposta do Especialista 3**:
  - Sim, caso esteja seguindo a mesma sequência de tarefas e gerando os mesmos artefatos definidos na UML Components [**E3.C7.2, E3.C7.4**].
- **Resposta do Especialista 4**:
  - Sim porque os artefatos gerados correspondem aos possíveis resultados esperados da realização desta atividade [**E4.C7.4**].
- **Resposta do Especialista 5**:
  - Sim. Nesse caso, a evolução proposta pelas atividades listadas é bem capaz de produzir as informações necessárias para materializar os artefatos listados [**E5.C7.4**].
- **Resposta do Especialista 6**:
  - Sim, considero [**E6.C7.4**].
- **Resposta do Especialista 7**:
  - Parece estar adequada. Eu preferiria ver exemplos de certos artefatos instanciados para ter certeza do tipo de informação proposta (por exemplo no Modelo de Informação de Interface) [**E7.C7.4**].
- **Resposta do Especialista 8**:
  - Idem atividades anteriores, considero adequada [**E8.C7.4**].

5.2. Você considera adequada a atribuição do papel (`Arquiteto de Linha de Produto`) envolvido na atividade `Especificação de Componente` do *SMarty-Components*? Justifique sua resposta.

- **Resposta do Especialista 1**:
  - Sim [**E1.C7.1**].
- **Resposta do Especialista 2**:
  - Sim [**E2.C7.1**];

  - Contudo, sugiro a especificação de outro papel, como um Engenheiro de Software, para auxiliar ou apoiar o Arquiteto de Linha de Produto durante as tarefas em geral [**E2.C8.7**].

- **Resposta do Especialista 3**:
  - Novamente, a nomenclatura Arquiteto de Linha de Produto apenas me remete ao conhecimento de definição e gerenciamento de linhas de produto. Não sei se aborda também o conhecimento de desenvolvimento baseado em componentes [**E3.C8.12**].
- **Resposta do Especialista 4**:
  - A meu ver, atribuição do papel do Arquiteto de Linha de Produto é adequada e essencial nesta atividade pois o detalhamento dos componentes facilita as etapas posteriores do desenvolvimento da LPS [**E4.C7.1**];
  - Além disso, a experiência do arquiteto pode preencher lacunas não preenchidas pelos artefatos de entrada [**E4.C7.1**].
- **Resposta do Especialista 5**:
  - Sim. Embora creio que o auxílio de um arquiteto de software experiente seria muito útil no desenvolvimento de tarefas como aquela responsável por "especificar restrições de componentes - interface" [**E5.C8.7**].
- **Resposta do Especialista 6**:
  - Sim, considero [**E6.C7.1**].
- **Resposta do Especialista 7**:
  - Sim, papel adequado [**E7.C7.1**].
- **Resposta do Especialista 8**:
  - Por mais que certamente o principal papel nesta atividade é desempenhado pelo arquiteto da LPS, eu considero que a participação de outros papéis como desenvolvedores ou especificadores de componentes poderiam subsidiar ou validar as atividades do arquiteto [**E8.C8.7**].

5.3. Você considera que o fluxo das tarefas da atividade `Especificação de Componente` do `Workflow de Especificação` contribui de maneira objetiva e precisa para gerar os artefatos *SMarty* (`Arquitetura SMarty Componentizada`, `Componentes SMarty` e `Interfaces SMarty`)? Justifique sua resposta.

- **Resposta do Especialista 1**:

  - Aparentemente sim. O fluxo é entendível e direto [**E1.C7.2**].

- **Resposta do Especialista 2**:
  - Contribui de maneira relevante, considerando o apoio das diretrizes (SMartyProcess) e dos estereótipos (SMartyProfile) contidos na abordagem SMarty. Desse modo, é possível garantir uma modelagem mais adequada em todo o cenário de Especificação de Componente [**E2.C7.3**];
  - No entanto, senti a falta de mais um papel para auxiliar o Arquiteto de Linha de Produto, como por exemplo, um outro tipo de Especialista em Linha de Produto [**E2.C8.7**].
- **Resposta do Especialista 3**:
  - Sim, caso os artefatos gerados na atividade sejam modelados com a UML para ser possível aplicar o SMarty [**E3.C7.2**].
- **Resposta do Especialista 4**:
  - O Fluxo de tarefas da atividade contribuiu de maneira objetiva e precisa na geração dos componentes por permitir observar a utilidade de cada artefato, papel ou tarefa realizada [**E4.C7.2**].
- **Resposta do Especialista 5**:
  - Sim. Através das tarefas relativas a essa atividade são complementadas as informações necessárias aos artefatos produzidos anteriormente [**E5.C7.2**].
- **Resposta do Especialista 6**:
  - Sim, considero [**E6.C7.3**].
- **Resposta do Especialista 7**:
  - Sim, contribui [**E7.C7.3**].
- **Resposta do Especialista 8**:
  - Idem outras duas atividades, considero adequados [**E8.C7.3**];
  - mas faço considerações com relação ao uso das quatro tarefas da smartyProcess. Assim como nas outras atividades, a identificação exata dos limites de"delimitar variabilidades"e"rastrear e controlar variabilidades"pode dar entendimento difuso sem a especificação [**E8.C8.13**].

5.4. Descreva a compreensão obtida em relação a execução das tarefas e geração dos artefatos na atividade `Especificação de Componente` do `Workflow de Especificação`.

- **Resposta do Especialista 1**:
  - Similar ao anterior, o arquiteto desempenha três atividades conforme três documentos prévios. Uma das atividades é pré-requisito para as outras duas. Como produto final são obtidos a especificação de arquitetura, a especificação de componentes e interface smarty [**E1.C7.1, E1.C7.2, E1.C7.4**].
- **Resposta do Especialista 2**:
  - Entendo que cada tarefa está relacionada com seu papel de maneira correta [**E2.C7.2**];
  - O Arquiteto de Linha de Produto descreve os processos e identifica o que é necessário. A partir disso, é possível aplicar o SMartyProcess e melhorar a modelagem por meio da geração de novos artefatos com estereótipos da SMarty utilizando o SMartyProfile [**E2.C7.1, E2.C7.3**].
- **Resposta do Especialista 3**:
  - Inicialmente a tarefa Definir Modelo de Informação de Interface precisa ser realizada para gerar o artefato Modelo de Informação de Interface. Este artefato serve como entrada para realizar paralelamente duas tarefas. A tarefa Especificarr Pré/Pós Condições de Operação gera o artefato Interfaces SMarty e a tarefa Especificar Restrições de Componente-Interface gera como saídas os artefatos Especificação de Arquitetura e Especificação de Componentes [**E3.C7.1, E3.C7.2, E3.C7.4**].
- **Resposta do Especialista 4**:
  - A Compreensão obtida foi de que a execução das tarefas visava refinar com um nível maior de detalhamento, os componentes da LPS. Uma outra compreensão obtida foi a maior ênfase dada a interface nesta atividade, com a definição por exemplo, do tipo de informação que deve ser adicionada nas interfaces [**E4.C7.1, E4.C7.2, E4.C7.4**].
- **Resposta do Especialista 5**:
  - Num primeiro momento, o arquiteto da LPS se preocupa em definir as restrições dos componentes/interfaces, além de pré e pós condições para as operações identificadas para a LPS. Num segundo momento, é ajustada a modelagem das variabilidades, para contemplar as restrições identificadas [**E5.C7.1, E5.C7.2**].

- **Resposta do Especialista 6**:
  - São definidos as interações entre as interfaces, a partir disto é feito a especificação dos pré e pós condições para execução das interfaces, e por fim as restrições dos componentes [**E6.C7.2**].
- **Resposta do Especialista 7**:
  - As entradas para esta atividade são os artefatos Modelo de Tipo de Negócio SMarty, Especificação de Componentes SMarty e Interfaces SMarty. Estas entradas auxiliam as tarefas Definir Modelos de Informações de Interface e Especificar Restrições de Componente-Interface [**E7.C7.2, E7.C7.4**].
- **Resposta do Especialista 8**:
  - Considerando o smartyProcess, considero adequado e viável o processo [**E8.C7.2**];
  - O output do UML Components está com sufixo"smarty"no artefato de interfaces, o que deveria ocorrer apenas após estressar o smartyProcess [**E8.C8.14**].

5.5. Quais elementos (artefatos, papéis ou tarefas) você entende que poderiam ser alterados e/ou removidos da atividade `Especificação de Componente` do `Workflow de Especificação` sem comprometer o objetivo final que é a especificação de uma arquitetura componentizada de LPS? Justifique sua resposta.

- **Resposta do Especialista 1**:
  - Nada, mas fiquei na dúvida de o que seria o"especificar restrições de componente-interface"comparado ao"especificar pré/pós condições de operação"[**E1.C8.9**].
- **Resposta do Especialista 2**:
  - Apenas sugiro que a atividade Modelo de Informação de Interface seja removido de dentro da Especificação de Componente e fique do lado externo (em torno) da atividade em geral [**E2.C8.13**].
- **Resposta do Especialista 3**:
  - Dois artefatos são gerados neste modelo: Especificação de Arquitetura e Especificação de Componentes. No UML Components original aparentemente esses dois artefatos são somente um, porém não afeta o modelo [**E3.C7.2**];

  - O outro ponto fica novamente a questão do papel Arquiteto de Linha de Produto [**E3.C8.12**].

- **Resposta do Especialista 4**:
  - O artefato "Modelo de Informação de Interface"poderia ser alterado para um artefato de saída, dada a sua importância para as interfaces de um componente da LPS. É possível perceber tal importância pelas entradas que o mesmo recebe nesta atividade [**E4.C8.14**].
- **Resposta do Especialista 5**:
  - Acredito que o auxílio de um arquiteto de software experiente seria muito útil no desenvolvimento de tarefas da atividade de "especificação de componente", como por exemplo "especificar restrições de componentes-interface"[**E5.C8.7**].
- **Resposta do Especialista 6**:
  - O artefato "Especificação de Arquitetura"não é algo definido neste momento, os componentes devem respeitar uma arquitetura pré-definida [**E6.C8.14**].
- **Resposta do Especialista 7**:
  - Não acredito ser necessário remover ou alterar [**E7.C7.5**].
- **Resposta do Especialista 8**:
  - Não identifiquei sugestões [**E7.C7.5**].

6. **Workflow de Especificação**

   6.1. Qual(quais) a(s) maior(es) complexidade(s)/esforço percebidas no `Workflow de Especificação`? Justifique sua resposta.

- **Resposta do Especialista 1**:
  - As duas tarefas citadas na questão anterior [**E1.C8.9**].
- **Resposta do Especialista 2**:
  - De forma geral senti a necessidade de criação de mais papeis para auxiliar no Workflow e torná-lo mais organizado e prático [**E2.C8.7**];
  - Com relação a atividade Especificação de Componente fiquei em dúvida do que realmente poderia ser gerado nesse cenário. Além disso, as descrições (nomes) de cada tarefa estão muito extensos o que dificulta a leitura e o entendimento do nível de granularidade e abstração que se encontra [**E2.C8.14**].

- **Resposta do Especialista 3**:
  - O Workflow de Especificação é bastante simples e de fácil entendimento [**E3.C7.5**];
  - O único esforço para o utilizador seria compreender bem os conceitos envolvidos na técnica de linhas de produto e na abordagem SMarty [**E3.C4.20, E3.C6.20, E3.C8.20**].
- **Resposta do Especialista 4**:
  - A maior complexidade percebida no Workflow de Especificação foi entender a contribuição das atividades isoladas para o Workflow como um todo, visto os sucessivos refinamentos realizados no(s) componente(s) [**E4.C8.21, E4.C4.23, E4.C6.25**].
- **Resposta do Especialista 5**:
  - Creio que a maior complexidade está no casamento das macro-atividades de "especificar componentes"e "especificar variabilidades- visto que tecnicamente possível é possível que a variabilidade ocorra dentro de um componente (condicionando a sua forma de trabalhar) ou que a variabilidade seja modelada como um componente (incluindo, ou não, em uma instância da LPS). Em ambos contextos a modelagem das variabilidades, e principalmente os possíveis relacionamentos e restrições entre variabilidades irá apresentar os maiores desafios [**E5.C4.20, E5.C6.20, E5.C8.20**].
- **Resposta do Especialista 6**:
  - Para mim foi compreender os elementos do processo e distinguir suas prioridades e necessidades [**E6.C4.23, E6.C6.25, E6.C8.21**].
- **Resposta do Especialista 7**:
  - Descobrir operações de negócio vai requerer apoio de outros artefatos ou outros conhecimentos não listados nos artefatos de entrada do processo [**E7.C8.10**].
- **Resposta do Especialista 8**:
  - Na parte UML Components Criar Especificação Inicial de Componentes e Arquitetura [**E8.C4.9**];
  - e na parte smartyProcess Identificar Variabilidades [**E8.C4.20,E8.C6.20,E8.C8.**

6.2. Quais elementos (artefatos, papéis ou tarefas) você removeria, adicionaria ou alteraria do `Workflow de Especificação`? Justifique sua resposta.

- **Resposta do Especialista 1**:
  - Acho que nenhum. Talvez, eu precise de mais conhecimento sobre os artefatos smarty para argumentar aqui [**E1.C3.5, E1.C5.5, E1.C7.5**].
- **Resposta do Especialista 2**:
  - Mesma resposta das questões anteriores.
- **Resposta do Especialista 3**:
  - A única mudança que imagino seria incluir um novo papel focado no desenvolvimento baseado em componentes para executar as tarefas junto com o papel do Arquiteto de Linha de Produto [**E3.C4.7, E3.C6.7, E3.C8.7**].
- **Resposta do Especialista 4**:
  - De maneira geral, não considero necessária a remoção de algum artefato, papel ou tarefa [**E4.C3.5, E4.C5.5, E4.C7.5**];
  - Considero que alguns artefatos, papéis ou tarefas deveriam ser alterados no que se referem à sua localização (dada a sua importância para a atividade) e indicadores (nomes - para facilitar a identificação intuitiva de suas funções). Tais observações de alteração de artefatos, papéis e tarefas foram mencionadas em questões anteriores, ao longo das atividades do Workflow de Especificação [**E4.C4.12, E4.C4.13, E4.C4.14**].
- **Resposta do Especialista 5**:
  - Na atividade de "especificação de componentes"proporia a inclusão de um arquiteto de software para auxiliar o arquiteto da LPS - com funções distintas [**E5.C8.7**];
  - Na atividade "SmartyProcess"proporia apenas rever (mudar nome e objetivos ou mudar a sua posição no fluxo) o fluxo com relação a "rastrear e controlar variabilidades"[**E5.C4.9, E5.C6.9, E5.C8.9**].
- **Resposta do Especialista 6**:
  - A saída do artefato de "especificação de arquitetura"e a prioridade de execução das tarefas de identificação das interfaces de negócio com os e sistema [**E6.C8.13, E6.C8.14**].
- **Resposta do Especialista 7**:

  - A principio, nao removeria nenhum. Talvez adicionaria o papel de analista de negócios para apoiar a identificação de operação de negócios [**E7.C8.7**].

- **Resposta do Especialista 8**:
  - Sem complementações sugeridas [**E8.C3.5, E8.C5.5, E8.C7.5**].

7. ***SMartyComponents***

7.1. Na sua opinião, qual a contribuição do *SMartyComponents* para refinar os artefatos *SMarty*? Justifique sua resposta.

- **Resposta do Especialista 1**:
  - Contribuem para um refinamento e definição melhor dos artefatos necessários para o smarty [**E1.C9.19**].
- **Resposta do Especialista 2**:
  - As principais contribuições estão relacionadas e especificação de estereótipos e diretrizes por meio da utilização da abordagem SMarty que colabora no panorama de modelagem e análise do workflow. Além disso, acredito que novas LPrS podem ser geradas com menor esforço e com maior organização em todo o processo [**E1.C9.19**];
  - Por fim, as três principais atividades propiciam a qualidade das gerações de LPrS por meio do SMartyComponents [**E2.C9.17**].
- **Resposta do Especialista 3**:
  - O SMartyComponents contribui na organização das tarefas baseadas no UML Components em comunhão com a abordagem SMarty facilitando o entendimento do processo a ser seguido [**E3.C9.16**];
  - Pensando em linhas de produto e os componentes que podem ser gerenciados com essa técnica, o SMartyComponents pode ser um ponto inicial para melhor entendimento e organização o processo envolvido [**E3.C9.19**].
- **Resposta do Especialista 4**:
  - Vejo que a contribuição do SMartyComponents está em especializar componentes segundo a abordagem SMarty, facilitando desta forma, a sua aplicação. A definição de uma arquitetura baseada em componentes é um exemplo de aplicação do SMartyComponents [**E4.C9.19**].

- **Resposta do Especialista 5**:
  - Toda sistematização de ações, como a proposta pelo SMartyComponents, trás o benefício do entendimento e previsibilidade do processo, para dessa forma poder mensura-lo, aplicar métricas, para então realizar os ajustes e as melhorias necessárias ao mesmo [**E5.C9.18**].
- **Resposta do Especialista 6**:
  - É difícil mensurar sem conseguir visualizar o resultado da variabilidade, mas é possível visualizar a aplicação do SMartyProcess nos artefatos e a reutilização destes artefatos nas atividades seguintes [**E6.C9.18, E6.C9.19**].
- **Resposta do Especialista 7**:
  - Vai apoiar de forma mais detalhada a especificação da Arquitetura e dos Componentes da Linha de Produtos [**E7.C9.17**].
- **Resposta do Especialista 8**:
  - Creio que não tenha compreendido corretamente a pergunta mas creio que a contribuição está em mapear um método sedimentado de identificação e especificação de componentes com a abordagem de linhas de produto de software. Quando finalizado o processo, os artefatos com sufixo "smarty"deverão ter pontos de variabilidade de tal forma que sejam instanciáveis conforme o contexto de uso requerido, dentro de uma linha de produtos de software [**E8.C9.19**].

7.2. Qual atividade de todas do *SMartyComponents* demanda maior esforço ou complexidade? Justifique sua resposta.

- **Resposta do Especialista 1**:
  - Não me lembro o nome, mas é o primeiro diagrama dado o envolvimento de mais pessoas e mais decisões não objetivas [**E1.C10.24**].
- **Resposta do Especialista 2**:
  - A Especificação de Componente. Devido ao fato de que compreende as duas atividades anteriores, sintetizando tudo o que já foi especificado. Isso gera uma certa carga cognitiva no Arquiteto de Linha de Produto em todo o processo até o final desta fase [**E2.C10.21**].
- **Resposta do Especialista 3**:
  - A Identificação de Componente aparenta demandar maior esforço pelo número de tarefas e artefatos envolvidos. Além disso, demanda mais

conhecimento técnico e experiência sobre desenvolvimento baseado em componentes para gerar os artefatos envolvidos [**E3.C10.21**].

- **Resposta do Especialista 4**:
  - Acredito que a atividade de "Interação de Componente" seja a de maior esforço e complexidade, pois relaciona os artefatos já produzidos com a experiência do arquiteto na descoberta das operações de negócio [**E4.C10.25**].
  - Um arquiteto com dificuldades em explicitar o seu conhecimento para aplicá-lo na busca e seleção das operações do componente pode comprometer todas as atividades futuras do workflow [**E4.C10.25**].
- **Resposta do Especialista 5**:
  - Diria que a atividade recorrente, denominada de "SMartyProcess" demande o maior esforço, ao mesmo tempo que apresenta a maior complexidade, visto que a sua realização precisam ser "casadas" com as tarefas convencionais (proposta pelo UML Components). Digo isso, porque a ação de identificação e representação de variabilidades vai permear todo o ciclo de desenvolvimento (e não ser realizada depois das tarefas convencionais desse ciclo) - por esse fato, ressalto a importância do rastreamento das variabilidades [**E5.C10.20**].
- **Resposta do Especialista 6**:
  - A atividade de "Identificação de Componentes" foi que demandou maior esforço pela quantidade de elementos e no entendimento dos conceitos e relacionamentos [**E6.C10.23**].
- **Resposta do Especialista 7**:
  - Provavelmente a identificação de componentes. Mas acho que isso vai depender do contexto e da qualidade de artefatos e conhecimento dos papeis envolvidos [**E7.C10.23**].
- **Resposta do Especialista 8**:
  - Acredito que a Especificação de Componentes seja a que demanda maior esforço ou complexidade em função da granularidade. Para formar uma base suficiente para uma LPS, é necessário ter um grande número de artefatos e bastante robustos para efetivamente viabilizarem a reutilização e ganho em escala [**E8.C10.21**].

Anexo B - A Linha de Produto de Software *Arcade Game Maker*

D.1 Introdução

A *Arcade Game Maker* (AGM) é uma LPS pedagógica para jogos, criada pelo *Software Engineering Institute* (SEI) (SEI, 2009) para apoiar o aprendizado e a experimentação de conceitos de LPS. Possui um conjunto completo de documentos e modelos UML, bem como um conjunto de classes testadas e código-fonte para três jogos diferentes, os quais são: *Pong*, *Bowling* e *Brickles*. Apesar de não ser uma LPS comercial, a AGM tem sido utilizada para ilustrar os conceitos de várias abordagens diferentes de LPS, estudos de caso e avaliação de arquiteturas de LPS.

Os artefatos essenciais da AGM são: o modelo de características, o modelo de casos de uso, o modelo de classes e a arquitetura lógica de componentes. Neste apêndice, são apresentados o modelo de casos de uso e o modelo de classes da LPS AGM, pois são relevantes com relação aos objetivos deste trabalho.

D.2 Similaridades e Variabilidades

A seguir são apresentadas as principais similaridades contidas na LPS AGM:

- todo jogo possui um conjunto de **Sprites**, nas quais são os elementos do jogo que os jogadores vêem e com os quais eles interagem;

- todo jogo possui um conjunto de **Rules**, nas quais são as regras que regem as ações dos jogos. Por exemplo, um jogo pode ter uma regra em que um objeto em movimento ao colidir com um objeto estático deve obedecer às leis da física; e

- todos os jogos envolvem movimentação.

Além das similaridades a LPS AGM possui as seguintes variabilidades:

- **Tipos de Regras**: é a maior diferença entre os jogos. Algumas regras estão relacionadas às leis da física (gravidade, colisões, etc.) e podem ser aplicáveis a múltiplos jogos. Outras regras estão especificamente relacionadas a um jogo e podem ser usadas em todas as implementações do jogo, mas não se aplicam a outros jogos; e

- **Tipos de Movimentação**: em alguns jogos a movimentação é inerente à operação do jogo. Isto acontece periodicamente e é orientada pelo tempo. Em outros jogos, o jogador escolhe e inicia a movimentação, sendo ações dirigidas pelo ator.

D.3 Atores e Casos de Uso

Os itens a seguir apresentam os atores e casos de uso da LPS AGM e as suas principais ações. A Figura 4.1 apresenta o modelo de casos de uso da LPS AGM.

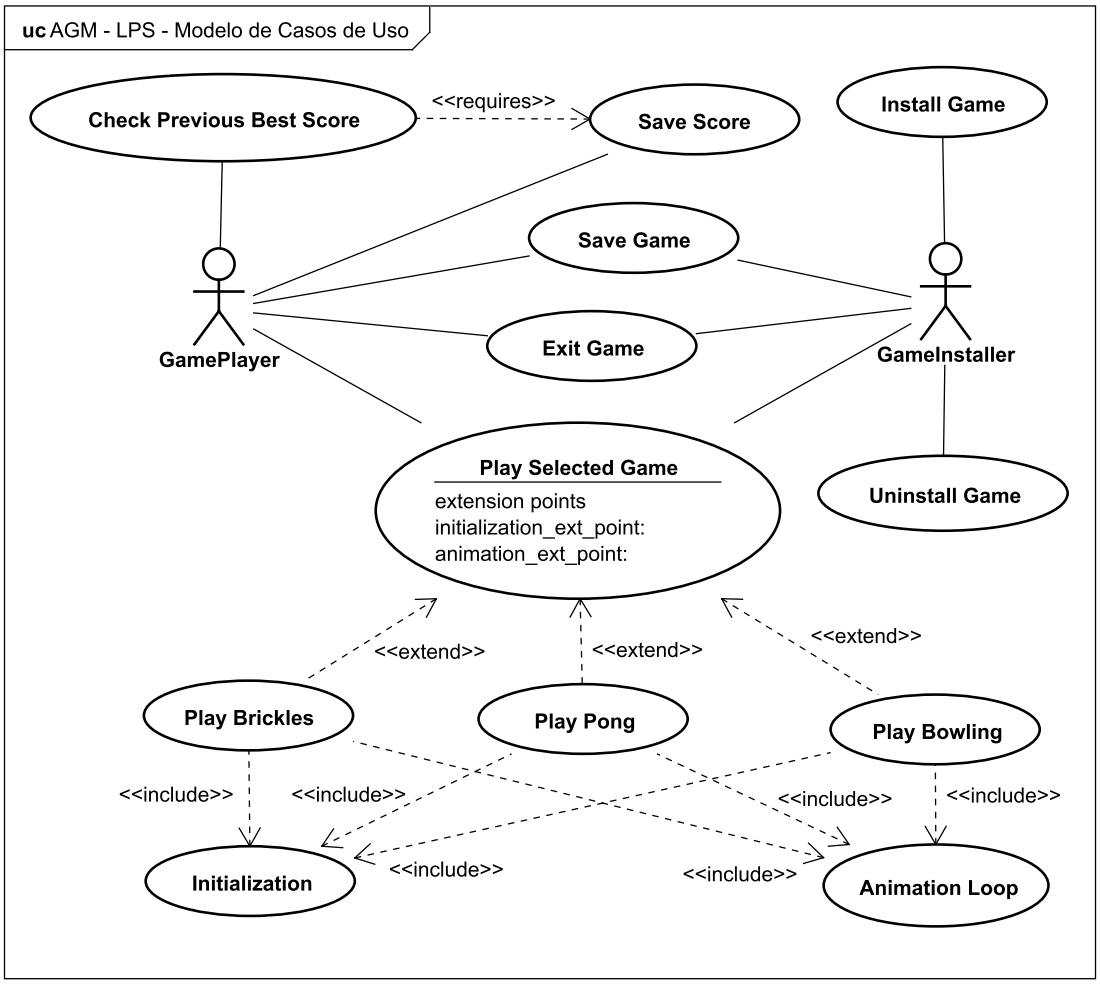

Figura 4.1: Modelo de Casos de Uso da LPS AGM. Adaptado de (OliveiraJr *et al.*, 2010b).

1. ***GamePlayer***: ator responsável por executar as principais ações de um jogo produzido pela AGM.

2. ***GameInstaller***: ator responsável por ações de configuração dos jogos (instalação e desinstalação) produzidos pela AGM.

3. ***Check Previous Best Score***: verifica e apresenta a melhor pontuação registrada anteriormente.

   (a) Ator seleciona a opção *CHECK PREVIOUS BEST SCORE* do menu do sistema. Sistema pede para fornecer o nome do arquivo, lê o arquivo e retorna o placar (*score*) à caixa de diálogo.

(b) Ator seleciona *OK* na caixa de diálogo para continuar. Sistema retorna ao estado anterior.

4. ***Save Score***: salva a pontuação corrente do jogador.

   (a) Ator seleciona *SAVE SCORE* no menu do sistema. Sistema pede um nome de arquivo (cria um novo se não existe), escreve o score no arquivo e retorna ao estado pré-salvo do jogo.

5. ***Save Game***: salva o jogo em andamento.

   (a) Ator seleciona a opção *SAVE GAME* no menu do sistema. Sistema permite ao ator especificar um nome de arquivo e, em seguida, escreve os dados do jogo no arquivo especificado, retornando ao estado pré-salvo do jogo.

6. ***Install Game***: instala o jogo escolhido.

   (a) Ator escolhe o instalador do jogo para ser executado. Sistema apresenta uma caixa de diálogo para escolher o diretório em que serão armazenados os arquivos do jogo.

   (b) Ator escolhe o diretório. Sistema armazena os arquivos no diretório escolhido.

7. ***Exit Game***: encerra o jogo em andamento.

   (a) Ator seleciona *EXIT* no Menu do sistema. Sistema apresenta a caixa de diálogo para salvar ou sair do jogo.

   (b) Ator salva o jogo. Sistema salva e sai ou cancela saída do jogo.

   (c) Sistema retorna à ação suspensa.

8. ***Uninstall Game***: remove o jogo selecionado.

   (a) Ator escolhe a opção *UNINSTALL* do menu do sistema. Sistema apresenta uma caixa de diálogo ao ator.

   (b) Ator seleciona o diretório do jogo a ser removido. Sistema exclui os arquivos do diretório e apresenta uma caixa de diálogo de remoção concluída.

   (c) Ator seleciona a opção OK da caixa de diálogo. Sistema fecha a caixa de diálogo.

9. ***Play Selected Game***: um ator seleciona o jogo e inicia a sua execução.

   (a) Ator seleciona a opção *PLAY* a partir do menu. Sistema inicializa o jogo e apresenta o *GameBoard*.

(b) Ator clica com o botão esquerdo e inicia o jogo. Sistema inicia a ação do jogo.

(c) Ator clica com o botão esquerdo ou usa o teclado para enviar comandos. Sistema responde aos comandos.

(d) Ator responde à caixa de diálogo *Won/Lost/Even* clicando com o botão esquerdo. Sistema retorna o *GameBoard* para ser inicializado.

(e) A qualquer instante o ator pode selecionar *EXIT*, via menu.

10. ***Play Bowling***: inicia o jogo *Bowling*.

(a) Ator seleciona *PLAY* do Menu do sistema. Sistema inicializa o jogo e apresenta o *GameBoard*.

(b) Ator clica com o botão esquerdo para jogar. Sistema inicia a ação do jogo.

(c) Ator repete as ações seguintes dez (10) vezes mais um lance de bônus (opcional)

(d) Ator posiciona o mouse e clica com o botão esquerdo para lançar a *Bowling-Ball* pela *Alley* (pista). Sistema move a *BowlingBall* pela *Alley* usando um algoritmo randômico. Se há colisões com os *BowlingPins*, o sistema move os *BowlingPins*, segundo as leis físicas de colisão. Sistema conta o número de pinos derrubados. Sistema computa o *score*.

11. ***Play Brickles***: inicia o jogo *Brickles*.

(a) Ator seleciona *PLAY* do menu do sistema. Sistema inicializa o jogo e apresenta o *GameBoard*.

(b) Ator clica com o botão esquerdo para iniciar o jogo. Sistema inicia a ação do jogo.

(c) Ator usa o botão esquerdo ou o teclado para enviar comandos ao jogo. Sistema move o *Paddle* horizontalmente, seguindo o rastro do mouse. A cada movimento do *Puck*, o sistema verifica se houve colisão com outro objeto. Se o *Puck* colide com o *Ceiling* (teto) ou com uma *Wall*, o *Puck* volta para a área de jogo. Se o *Puck* colide com o *Floor*, deixa de existir. Se o número máximo de *Pucks* não foi alcançado, um novo é criado, caso contrário a caixa de diálogo *Lost* é apresentada. Se o *Puck* colide com um *Brick*, a ação a ser tomada depende do *Brick*. Se for o último *Brick*, a caixa de diálogo *Won* é apresentada.

(d) Ator responde à caixa de diálogo *Won/Lost* clicando com o botão esquerdo. Sistema retorna o *GameBoard* ao estado inicializado e pronto para jogar.

12. ***Play Pong***: inicia o jogo *Pong*.

    (a) Ator seleciona *PLAY* do menu do sistema. Sistema inicializa o jogo e apresenta o *GameBoard*.

    (b) Ator clica com o botão esquerdo para iniciar o jogo. Sistema inicia a ação do jogo.

13. ***Animation Loop***: executa as ações de animação dos jogos.

    (a) Sistema gera periodicamente sinais e os envia para o jogo.

    (b) Sistema move todos os objetos passo a passo de acordo com os seus algoritmos de movimentação.

    (c) Sistema verifica se há colisões e executa os algoritmos de colisão dos objetos.

14. ***Initialization***: inicializa o jogo selecionado e apresenta o *GameBoard*.

    (a) Sistema cria as instâncias-padrão para as classes requeridas.

    (b) Sistema entra no estado *READY*.

D.4 Classes

A Tabela 4.1 apresenta as classes da LPS AGM e uma breve descrição sobre cada uma delas. A Figura 4.2 apresenta o modelo de classes da LPS AGM.

Tabela 4.1: Pacotes e Descrição das Classes da LPS AGM

Pacote	**Classe**	**Descrição**
coreAssets	*Board*	Borda de um jogo
	GameMenu	Menu com as opções de um determinado jogo
	GameSprite	Elementos dos jogos com os quais o jogador interage
	Menu	Menu com as principais opções dos jogos
	MovableSprite	Elementos que se movem em um jogo
	Paddle	Elemento utilizado em um jogo para colocar um Puck em movimento
	Point	Determinado ponto em um retângulo
	Puck	Representa o principal elemento de um jogo como, por exemplo, a bola que derruba os BowlingPins no jogo Bowling, a bolinha que destrói os BrickPile no jogo Brickles, etc.
	Rectangle	Um retângulo que demarca uma área em um jogo
	Size	Tamanho de um retângulo
	SpritePair	Par de elementos de um jogo que reagem à uma ação
	StationarySprite	Elementos que não se movem em um jogo
	Velocity	Velocidade de um MovableSprite
	Wall	Representa as paredes de um jogo
bowl	*Bowling*	Classe com a inicialização do jogo
	BowlingBall	Bola de boliche
	BowlingBoard	Borda do jogo Bowling
	BowlingGameMenu	Menu com as opções específicas do jogo
	BowlingPin	Pino do jogo de boliche
	Edge	Limites esquerdo e direito da canaleta de boliche
	EndOfAlley	Fim da pista de boliche
	Gutter	Canaleta da pista de boliche
	Lane	Pista de boliche
	RackOfPins	Local onde os pinos são posicionados
pong	*BottomPaddle*	Elemento que movimenta a Puck do jogo, localizado na parte inferior da PongBoard
	Ceiling	Teto do jogo
	DividingLine	Linha divisória dos Paddles
	Floor	Chão do jogo
	LeftWall	Parede à esquerda do jogo
	Pong	Classe com a inicialização do jogo
	PongBoard	Borda do jogo Pong
	PongGameMenu	Menu com as opções específicas do jogo
	RightWall	Parede à direita
	ScoreBoard	Placar do jogo
	TopPaddle	Elemento que movimenta a Puck do jogo, localizado na parte superior da PongBoard
brickles	*Brick*	Tijolo a ser quebrado pelo elemento Puck
	Brickles	Classe com a inicialização do jogo
	BricklesBoard	Borda do jogo Pong
	BricklesGameMenu	Menu com as opções específicas do jogo
	BrickPile	Pilha de tijolos
	Ceiling	Teto do jogo
	Floor	Chão do jogo
	LeftWall	Parede à esquerda
	PuckSupply	Quantidade de Pucks que o jogador tem direito em um jogo
	RightWall	Parede à direita

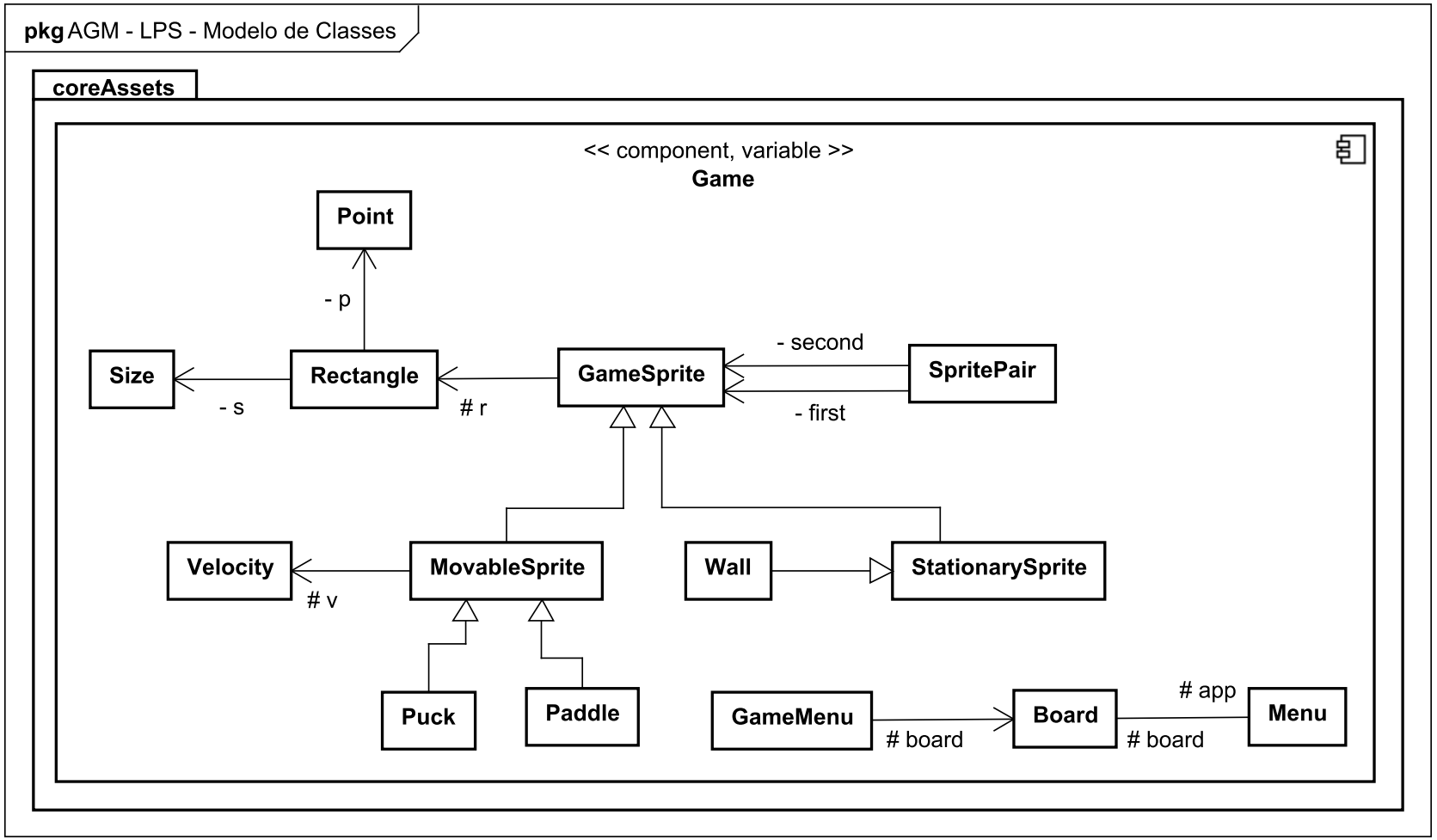

Figura 4.2: Modelo de Classes da LPS AGM. Adaptado de (OliveiraJr *et al.*, 2010b).

D.5 Telas dos Jogos

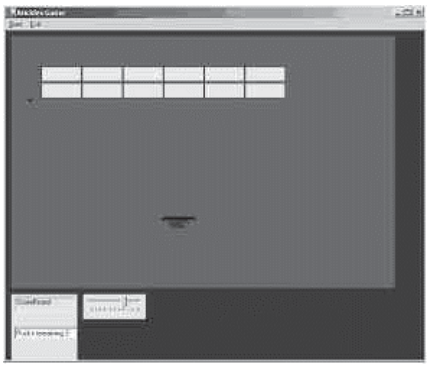

Figura 4.3: Tela do Jogo Brickles

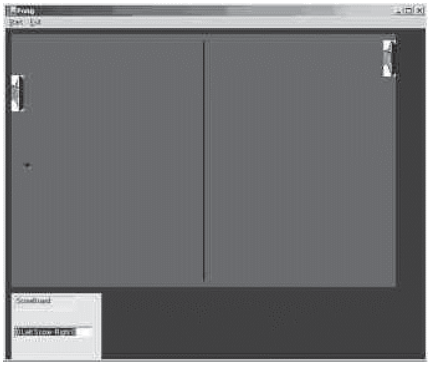

Figura 4.4: Tela do Jogo Pong

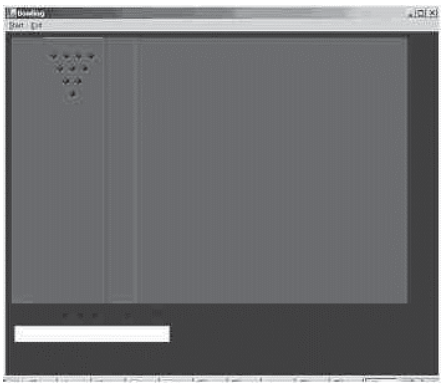

Figura 4.5: Tela do Jogo Bowling

Anexo C - A Linha de Produto de Software *Mobile Media*

A *Mobile Media* é uma LPS composta por aplicações (produtos) que manipulam músicas, vídeos e fotos para dispositivos móveis, como *smartphones* e *tablets*. Provê suporte para gerenciar (criar, excluir, visualizar, executar e enviar) diferentes tipos de mídia (Young, 2005).

A *Mobile Media* surgiu da extensão de um LPS já existente denominada *Mobile Photo* (Young, 2005) por meio da inserção de novas propriedades multimídia, como manipulação de vídeos e músicas, que somente podem ser realizados em alguns tipos de aparelhos.

De certa forma, pode-se dizer que a inserção das características opcionais e alternativas a determinados aparelhos caracterizou o surgimento da *Mobile Media*. A Figura 5.1 apresenta o Modelo de Casos de Uso da *Mobile Media*.

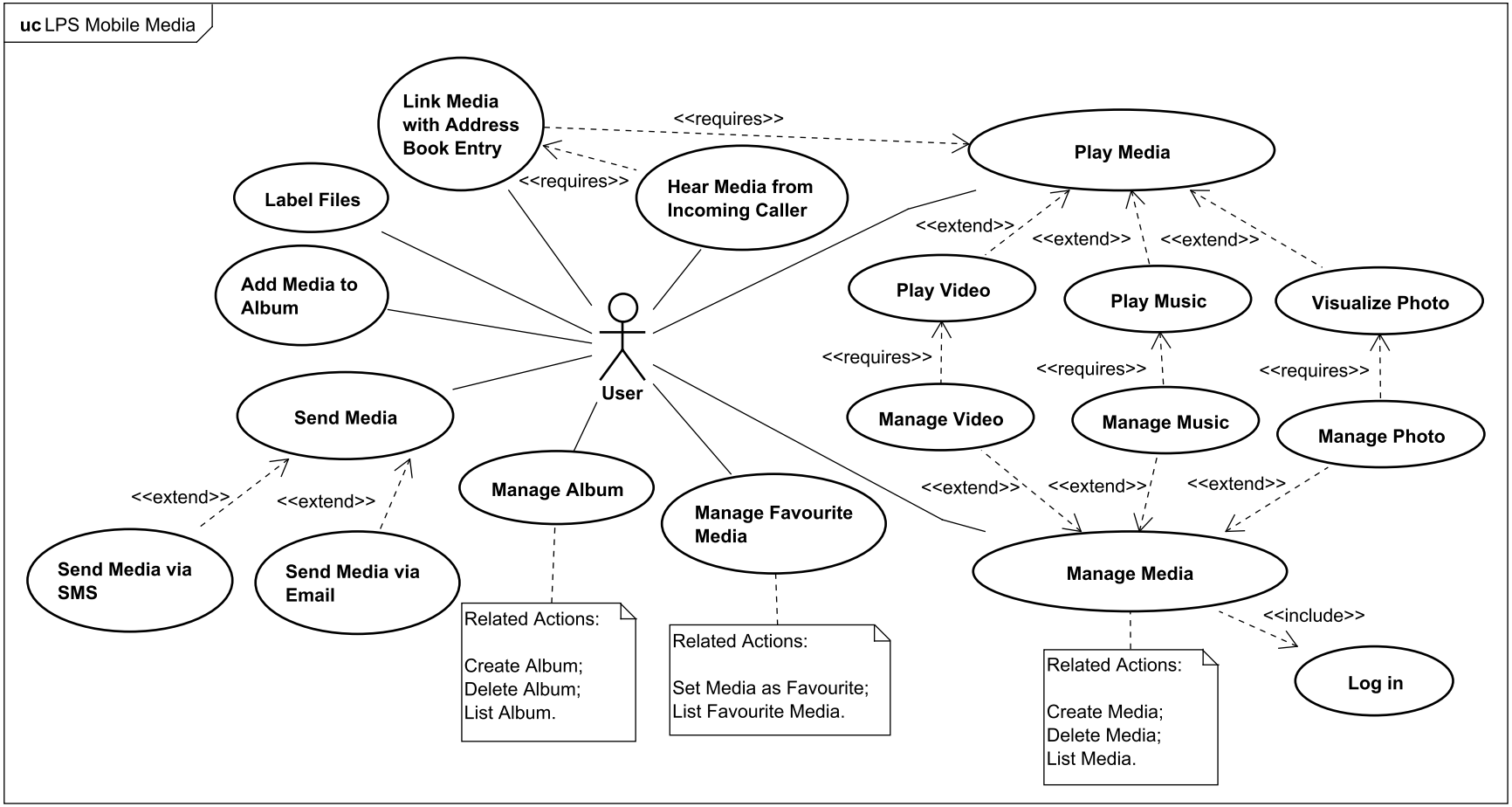

Figura 5.1: Modelo de Casos de Uso da LPS *Mobile Media*. Adaptado de (Santos *et al.*, 2008).

Os itens a seguir apresentam as características básicas e as características variáveis da *Mobile Photo*, agora denominada *Mobile Media*:

1. **Criar Álbum de Fotos**: Permite ao usuário definir novos álbuns de fotos para armazenar categorias de fotos no dispositivo. A persistência da informação do álbum é realizada utilizando RMS (J2ME *Record Management System*).

2. **Armazenar Foto**: Gerenciar a conversão e persistência dos arquivos de foto para o sistema de arquivos do dispositivo utilizando RMS.

3. **Adicionar/Deletar Foto**: Permite ao usuário excluir fotos permanentemente do dispositivo ou adicionar novas fotos a álbuns definidos.

4. **Rotular Foto**: Permite ao usuário determinar um texto para uma foto. Os rótulos aparecerão na lista de exibição e podem ser utilizados para uma futura funcionalidade relacionada à busca.

5. **Visualizar Foto**: Mostra uma foto selecionada na tela do dispositivo.

6. **Enviar Foto via SMS**: Permite a um usuário enviar uma foto para outro via *Short Messaging Service*.

7. **Enviar Foto via Email**: Permite a um usuário enviar uma foto para outro via Email.

8. **Relacionar Foto com Registro na Agenda**: Permite ao usuário associar um registro na sua lista de contatos com a foto do álbum.

9. **Mostrar Foto nas Chamadas Recebidas**: Intercepta chamadas recebidas e mostra a foto associada ao contato.

10. **Tocar Melodia nas Chamadas Recebidas**: Intercepta chamadas recebidas e toca uma melodia personalizada para aquele contato.

De acordo com a Figura 5.1 da LPS *Mobile Media*, a Figura 5.2 apresenta um Modelo de Casos de Uso da LPS *Mobile Media* com base na abordagem *SMarty*.

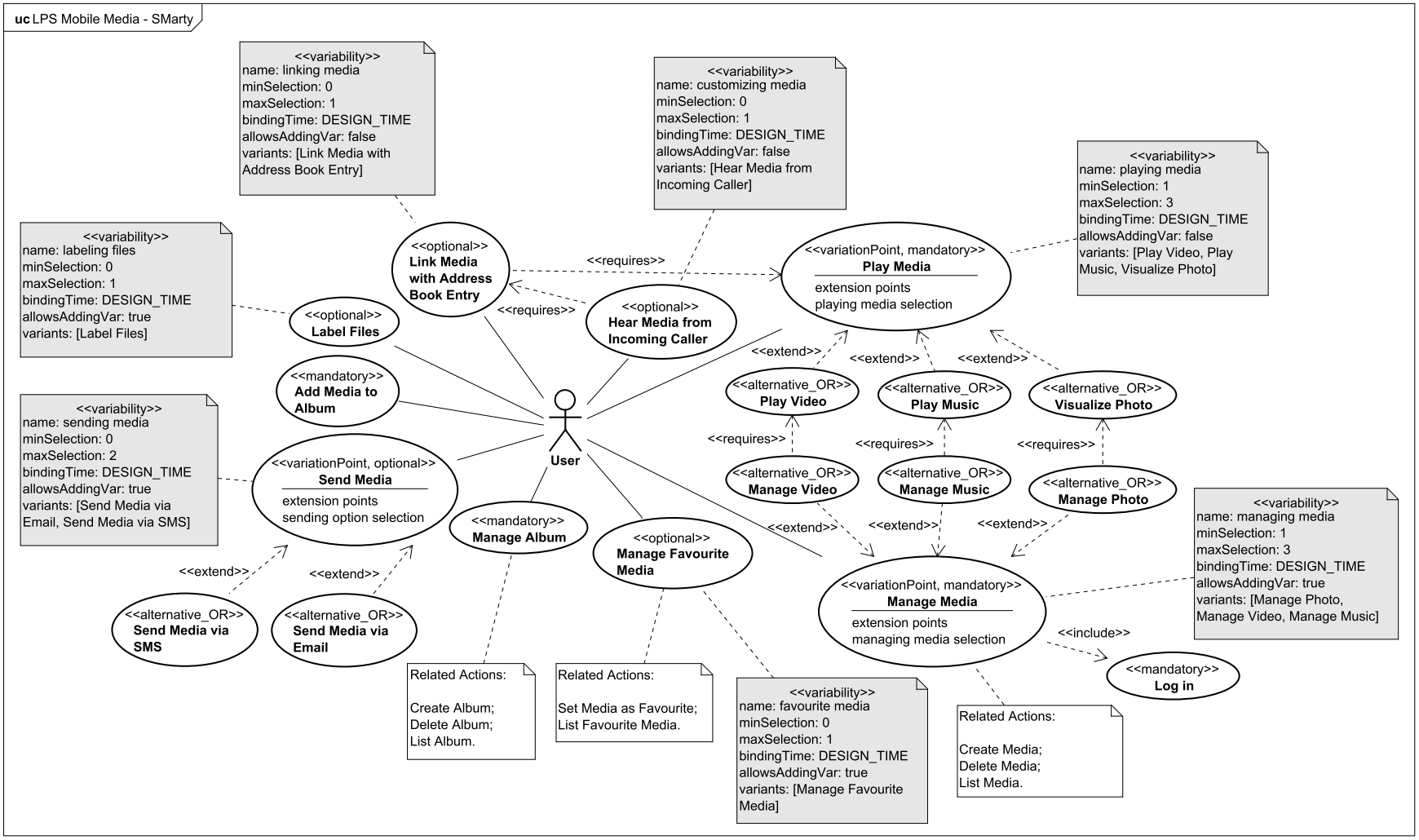

Figura 5.2: Modelo de Casos de Uso da LPS *Mobile Media* com base na *SMarty*. Adaptado de (Contieri Junior, 2010).