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

GIOVANNA CRISTINA DE SOUZA BETTIN

SMartyPerspective: uma técnica de inspeção baseada em perspectiva para modelos SMarty de linha de produto de software

Maringá
2021

GIOVANNA CRISTINA DE SOUZA BETTIN

SMartyPerspective: uma técnica de inspeção baseada em perspectiva para modelos SMarty de linha de produto de software

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

Maringá
2021

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

B565s

Bettin, Giovanna Cristina de Souza

SMartyPerspective : uma técnica de inspeção baseada em perspectiva para modelos SMarty de linha de produto de software / Giovanna Cristina de Souza Bettin. -- Maringá, PR, 2021.

278 f.: il. color., figs., tabs.

Orientador: Prof. Dr. Edson Alves de Oliveira Junior.

Dissertação (Mestrado) - Universidade Estadual de Maringá, Centro de Tecnologia, Departamento de Informática, Programa de Pós-Graduação em Ciência da Computação, 2021.

1. Inspeção de software. 2. Linha de produto de software. 3. SMarty - Gerenciamento de variabilidade. 4. SMartyPerspective. 5. UML (Unified modeling language). I. Oliveira Junior, Edson Alves de, orient. II. Universidade Estadual de Maringá. Centro de Tecnologia. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDD 23.ed. 005.12

Marinalva Aparecida Spolon Almeida - 9/1094

FOLHA DE APROVAÇÃO

GIOVANNA CRISTINA DE SOUZA BETTIN

SMartyPerspective: uma técnica de inspeção baseada em perspectiva para modelos SMarty de linha de produto de software

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

Profa. Dra. Aline Maria Malachini Miotto Amaral
Universidade Estadual de Maringá – DIN/UEM

Profa. Dra. Simone do Rocio Senger de Souza
Universidade de São Paulo – ICMC/USP

Aprovada em: 30 de março de 2021.
Local da defesa: Sala virtual
meet.google.com/fmd-wvue-ibm.

Dedico este trabalho as minhas maiores fontes de amor, força, coragem, trabalho, humildade e perseverança: Cláudia e Pedro Bettin. Vocês me inspiram! Não teria chego até aqui sem a dedicação, confiança, orações e o amor de vocês.

AGRADECIMENTOS

Agradeço primeiramente a **Deus** e ao meu **São Miguel Arcanjo** por iluminarem meus caminhos e darem a força necessária para completar a dissertação frente aos problemas de saúde que surgiram no período do mestrado.

Aos meus pais, **Cláudia** e **Pedro** que nunca mediram esforços para minha formação acadêmica e de caráter. Vocês são minha maior inspiração! São anjos que Deus colocou em minha vida! Obrigada por serem minha força e vivenciarem as alegrias e dificuldades durante o período do mestrado. E principalmente por sempre terem acreditado que era possível melhorar e concluir mais essa etapa.

Ao meu irmão **Pietro**, meu melhor amigo e alegria da casa! Obrigada por me inspirar a nunca desistir dos meus sonhos, a focar nos objetivos e sempre acreditar em mim. Me orgulho muito de você, do seu talento, dedicação e companheirismo.

As minhas avós, **Cecília** e **Sebastiana**, por cuidarem e rezarem por mim.

Ao meu orientador **Prof. Dr. Edson**, por me guiar e mostrar os melhores caminhos a serem tomados na pesquisa. Mas, principalmente por todo o cuidado que teve comigo durante este período e ter entendido e respeitado as limitações que a doença me trouxe. Obrigada por tudo! Sua paciência e confiança no meu trabalho foram importantes e motivadores para mim. Espero podermos compartilhar ainda muitas pesquisas!

Aos meus padrinhos **Helena** e **João Marcelo** que zelam sempre por mim. Obrigada por todas as orações, por serem presentes e pelas lindas rosas de Santa Rita.

Aos amigos que me deram a alegria e motivação nos momentos que o cansaço e a tristeza batiam: **Amanda, Aline, Leila, Catarina** e **Diego**. Em especial, a **Thais**, que compartilha a vida acadêmica comigo desde a graduação. Com você a caminhada ficou mais leve, cheia de alegria e comidas gostosas. Obrigada por todo o companheirismo, amizade e carinho.

A **Inês**, a secretária maravilhosa do PCC! Você foi um grande presente do mestrado. Obrigada pela amizade, preocupação, conselhos e sempre estar disponível para me ajudar.

A todos que contribuíram direta e indiretamente para a realização deste trabalho: aos professores que passaram ao longo da minha vida e foram fonte de conhecimento e aos que participaram da avaliação da técnica. Em especial, ao **Ricardo** que sempre esteve disponível e me auxiliou em relação aos conceitos principais deste trabalho, e as profas. dras. **Aline** e **Simone** pelas valiosas contribuições para refinamento deste trabalho.

Aos doutores que literalmente me colocaram de pé e não descansaram até me ver bem. A alegria que me recebem no consultório a cada vitória no tratamento me renova! **Dr. Marcus Petruco** e **Dra. Naiene**, vocês são um exemplo de trabalho sério, compromisso e amizade. Obrigada por não terem medido esforços para meu tratamento.

SMartyPerspective: uma técnica de inspeção baseada em perspectiva para modelos SMarty de linha de produto de software

RESUMO

Linha de Produto de Software (LPS) é um paradigma que reutiliza artefatos de software por meio de um conjunto de sistemas que têm características comuns e variáveis para atender um domínio específico. Para garantir a qualidade dos produtos gerados de uma LPS são necessárias atividades de verificação e validação em seus artefatos para que defeitos não se propaguem pelos produtos da família. A inspeção de software é um método econômico e eficiente de revisão que detecta antecipadamente defeitos desde os primeiros estágios de desenvolvimento com o suporte, por exemplo, das técnicas de leitura. A falta de técnicas de leitura para atender às características específicas de reúso em LPS, em especial para o gerenciamento de variabilidades, fez surgir a necessidade de propor técnicas próprias para esta atividade. A Leitura Baseada em Perspectiva (*Perspective-Based Reading - PBR*) tem se mostrado uma alternativa viável para inspeção, pois considera diferentes cenários e perspectivas dos revisores dos artefatos de software sob uma finalidade específica para que possam se concentrar nas informações que são importantes de acordo com sua especialidade durante a detecção de defeitos. Assim, o objetivo desta dissertação é especificar e avaliar a *SMartyPerspective*, uma técnica PBR para inspecionar diagramas *SMarty*, uma abordagem que permite o gerenciamento de variabilidades de LPS em diferentes diagramas UML e diagramas de *features*. A *SMartyPerspective* é composta de cinco perspectivas da Engenharia de Domínio de LPS: Gerente de Produto, Engenheiro de Requisitos de Domínio, Arquiteto de Domínio, Desenvolvedor de Domínio e Gerente de Ativos de Domínio. Para avaliar a viabilidade da técnica foi realizado um estudo qualitativo utilizando para análise dos dados o modelo *Technology Acceptance Model* (TAM) e procedimentos de *Grounded Theory*, como a Codificação. Os resultados obtidos indicam que a técnica é viável para inspeção dos diagramas, mas, revelam que devem ser realizadas melhorias para tornar a técnica menos cansativa, mais objetiva e simples para que haja um nível maior de concordância quanto à sua eficiência e aceitação de uso.

Palavras-chave: Gerenciamento de Variabilidades. Inspeção de Software. Leitura Baseada em Perspectiva. Linha de Produto de Software. SMarty. SMartyPerspective. UML.

SMartyPerspective: a perspective-based inspection technique for SMarty software product line models

ABSTRACT

Software Product Line (SPL) is a paradigm that reuses software artifacts through a set of systems that have common and variable characteristics to serve a specific domain. In order to guarantee the quality of the products generated from an SPL, verification and validation activities in its artifacts are necessary so that defects are not propagated by the family products. Software inspection is an economical and efficient method of revision that detects defects in advance from the early stages of development with the support, for example, of reading techniques. The lack of reading techniques to meet the specific characteristics of reuse in SPL, especially for the management of variability, led to the need to propose techniques specific to this activity. Perspective-Based Reading (PBR) has proved to be a viable alternative for inspection, as it considers different scenarios and perspectives of the reviewers of software artifacts under a specific purpose so that they can focus on the information that is important according to their specialty during the defect detection. Thus, the objective of this master thesis is to specify and evaluate SMartyPerspective, a PBR technique to inspect SMarty diagrams, an approach that allows the management of SPL variability in different UML and feature diagrams. SMartyPerspective is composed of five techniques (or scenarios) that understand the perspectives of SPL Domain Engineering: Product Manager, Domain Requirements Engineer, Domain Architect, Domain Developer and Domain Asset Manager. To assess the feasibility of the technique, a qualitative study was performed using the TAM and Grounded Theory method for data analysis. The results obtained indicate that the technique is feasible for inspection of the diagrams, but, they reveal that improvements must be made to make the technique less tiring, more direct and simple so that there is a greater level of agreement of its efficiency and acceptance of use.

Keywords: SMarty. SMartyPerspective. Software Product Line. Software Inspection Perspective-Based Reading. UML.Variability Management.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

AQD: Arquiteto de Domínio
CBR: *Checklist-Based Reading*
DSD: Desenvolvedor de Domínio
ERD: Engenheiro de Requisitos de Domínio
IEEE: *Institute of Electrical and Electronics Engineers*
GAD: Gerente de Ativos de Domínio
GRP: Gerente de Produto
LPS: Linha de Produto de Software
PBR: *Perspective-Based Reading*
SBR: *Scenario-Based Reading*
SEI: *Software Engineering Institute*
SMarty: *Stereotype-based Management of Variability*
UML: *Unified Modeling Language*
V&V: Verificação e Validação

SUMÁRIO

1 Introdução

1.1 Contextualização

O reúso de software tem como objetivo aproveitar partes comuns de softwares na criação de novos sistemas (Krueger, 1992). Esse processo surgiu como alternativa para colaborar com as necessidades da indústria de software para a construção e manutenção de sistemas com menor custo, confiáveis, eficientes e que atuassem sob fortes restrições (Soares, 2016).

Para apoiar o reúso foram elaboradas diversas abordagens que definem as ações e estratégias a serem tomadas para garantir a conformidade com padrões e normas (Krueger, 1992), como por exemplo: Linha de Produto de Software (LPS), *framework* de aplicações, padrões de projeto e biblioteca de programas (Szyperski e Murer, 2002). Entre elas, a LPS representa uma família de sistemas com artefatos que são comuns a todos os produtos (Clements e Northrop, 2002). Destaca-se das demais pelas melhorias de *time-to-market*, custo, flexibilidade, customização em massa, viabilidade, entre outras (de Almeida, 2019; Institute, 2017; Van der Linden *et al.*, 2007).

Além dos artefatos que são comuns, os produtos de uma LPS possuem diferenças entre si, que são denominadas variabilidades. Essa característica faz com que os produtos se diferenciem e sejam únicos em sua família. Destacar os produtos um dos outros e personalizá-los permite o sucesso de uma empresa no mercado. Logo, a gerência de variabilidades é uma das atividades mais importantes em uma LPS, por determinar a capacidade da organização em administrar as variabilidades dos produtos (Chen *et al.*, 2009; Pohl *et al.*, 2005).

Os artefatos similares ou diferentes de uma LPS podem ser qualquer ativo gerado durante o processo de desenvolvimento, como por exemplo: documento de requisitos,

projeto de arquitetura, componentes de software, modelos UML e planos de testes (Pohl *et al.*, 2005).

Como na abordagem tradicional de desenvolvimento de software, os produtos de uma LPS devem atingir alto grau de qualidade, por isso, a organização deve se prevenir de defeitos em seu núcleo de artefatos, já que seus ativos podem ser derivados em outros produtos e, assim, defeitos poderão ser espalhados por toda a família.

Entre as diversas abordagens para garantir a qualidade de uma LPS, a inspeção de software, uma atividade de verificação estática, se destaca por identificar defeitos antes que se propaguem pelo ciclo de vida do desenvolvimento do produto (Geraldi e OliveiraJr, 2017a). A inspeção é apoiada pelas técnicas de leitura que dão suporte a esta atividade.

A verificação quando realizada "de forma estruturada e com objetivos bem definidos, período estipulado e com a participação de pessoas com diferentes visões do problema tende ser mais efetiva que um simples processo de revisão casual" (Bartié, 2002).

Comparada às técnicas de leitura tradicionais (Leitura Baseada em *Checklist* (CBR) e *Ad hoc*), a PBR (Leitura Baseada em Perspectiva) considera as diferentes necessidades dos papéis da Engenharia de Software durante o ciclo de desenvolvimento para construir cenários que são usados para detectar defeitos sob diferentes óticas do projeto, visando tornar a inspeção mais efetiva e eficiente. Em diversos estudos relacionados em Mafra e Travassos (2005) a PBR se mostrou mais eficiente e efetiva quando comparada com as técnicas *Ad hoc* e Leitura Baseada em *Checklist* para artefatos do desenvolvimento tradicional de software.

1.2 Motivação e Justificativa

A qualidade de um sistema inclui o comportamento do software enquanto ele está em execução, o processo de desenvolvimento, estrutura e a organização dos componentes e a documentação associada (Lee, 2014). Logo, um engenheiro de software deve se preocupar com os artefatos gerados durante o processo de desenvolvimento, para que juntos, possam atingir a alta qualidade do produto final esperada por um cliente.

No contexto de LPS, em que um mesmo artefato é configurado e utilizado para atender as variabilidades de diversos produtos de uma mesma família, a qualidade pode se tornar ainda mais complexa, pois, um defeito incorporado em um artefato poderá ser propagado por toda a família de sistemas, produzindo não somente um, mas vários produtos que não satisfazem determinados atributos de qualidade (Geraldi *et al.*, 2015).

A inspeção de software é dos métodos mais efetivos e eficientes para garantir a qualidade dos artefatos que são gerados durante o desenvolvimento de software, verificando

as propriedades de qualidade dos artefatos por meio da detecção e remoção de defeitos o mais cedo possível, inclusive logo depois de ser criado, para que não custe até 10 vezes mais à organização por passar os defeitos para as próximas fases ou ao usuário (Basili *et al.*, 1996a; Fagan, 1986; Laitenberger *et al.*, 2000; Sabaliauskaite *et al.*, 2003; Travassos *et al.*, 1999; Zhu, 2016).

As técnicas de leitura contribuem para o processo de inspeção de software por meio da verificação de artefatos para detectar defeitos, mas, as técnicas usadas no desenvolvimento tradicional de software não são suficientes para atender as características de reusabilidade de uma LPS. Por isso, é importante o estudo e desenvolvimento de técnicas próprias para uma família de produtos. (Cunha *et al.*, 2012; Denger e Kolb, 2006; Geraldi *et al.*, 2015; de Mello *et al.*, 2014). Porém, até o momento há um número reduzido de técnicas de inspeção específicas para LPS: SMartyCheck (Geraldi *et al.*, 2015), SPLIT (Cunha *et al.*, 2012) e FMCheck (de Mello *et al.*, 2014).

Em seu trabalho, Mafra e Travassos (2005) reuniram 14 estudos para comparar a efetividade e eficiência das técnicas PBR com CBR e Ad hoc no processo tradicional de software. A PBR mostrou-se superior em 57% dos estudos, falhou em mostrar a superioridade em 29% e 14% não apresentaram diferenças significativas do ponto de vista estatístico.

Além deste trabalho, outros estudos evidenciaram a eficiência (Sabaliauskaite *et al.*, 2002), efetividade, eficácia e baixo custo por defeito identificado (Laitenberger *et al.*, 2000) da PBR em comparação com a CBR para apoiar a revisão de software em artefatos UML.

Logo, supõe-se que técnicas PBR sejam fortes candidatas para garantir o máximo possível de qualidade no gerenciamento de variabilidades nos diagramas por meio da detecção de defeitos em artefatos de LPS.

Além da importância da revisão de software para o apoio à qualidade de artefatos de LPS, outras motivações para este trabalho se referem a:

- pequena quantidade de técnicas de leitura para inspeção de diagramas UML, em especial para os que permitem o gerenciamento de variabilidades, como a abordagem *SMarty* (OliveiraJr *et al.*, 2010a); e
- carência de estudos que investigam o uso da PBR para LPS.

As técnicas apresentadas nos trabalhos relacionados que são próprias para LPS (Seção 2.4), exceto a *SMartyCheck*, são para inspeção em diagramas de *features* e se referem ao espaço do problema.

A abordagem *SMarty* refere-se ao espaço da solução, em que há um sistema concreto e pode ser representado nos modelos UML, que são mais baixo nível que os de *features*.

Surge então, a necessidade de mais atenção da área de garantia de qualidade de software em LPS baseadas em UML para modelos no espaço da solução, por representarem a realização do sistema.

Foi escolhida a abordagem *SMarty* para este trabalho, pois, entre as abordagens existentes para gerenciamento de variabilidades, ela apoia o maior número de diagramas (caso de uso, classe, componente, sequência e atividade) e se mostrou efetiva e eficiente em diversos estudos já realizados (Marcolino *et al.*, 2014a,b; Marcolino e OliveiraJr, 2017; Marcolino *et al.*, 2017; Nepomuceno e OliveiraJr, 2018; Nepomuceno *et al.*, 2020). Além disso, ela permite por meio de seus estereótipos e meta-atributos a rastreabilidade entre as variabilidades, que é útil para inspeção entre tipos diferentes de diagramas.

Assim, a especificação de uma técnica com uma nova abordagem para inspecionar artefatos de LPS com base em PBR, em especial diagramas UML *SMarty*, poderá contribuir para o avanço da área de revisão de software e para a qualidade de LPS baseadas em UML na atividade de Engenharia de Domínio.

1.3 Objetivos

Este trabalho tem como objetivo principal especificar e avaliar qualitativamente a *SMarty-Perspective*, uma família de técnicas de leitura baseada em perspectiva, para inspecionar diagramas *SMarty* para a detecção de defeitos.

Para alcançar o objetivo geral deste trabalho, os objetivos específicos necessários são:

- estudar as técnicas de leitura baseada em perspectiva existentes na literatura, bem como as classes de defeitos e papéis escolhidos;
- especificar a técnica *SMartyPerspective*; e
- avaliar empiricamente a técnica *SMartyPerspective* por meio de um estudo qualitativo.

1.4 Metodologia

As atividades realizadas durante o desenvolvimento deste trabalho podem ser divididas em três fases: Concepção, Desenvolvimento e Disseminação (Figura 1.1). Elas são descritas nesta seção com as metodologias utilizadas para alcançar os objetivos de cada uma das atividades.

A fase de **Concepção** engloba duas pesquisas na literatura que deram suporte às demais atividades para este trabalho: uma revisão bibliográfica e um estudo secundário. Na revisão bibliográfica foi realizada uma consulta de artigos e demais trabalhos científicos disponíveis em bibliotecas digitais, anais de eventos e periódicos em diferentes meios de publicação científica sobre Linha de Produto de Software, inspeção, diagramas *SMarty* e atributos de qualidade esperados após a realização de inspeção.

A **revisão bibliográfica** foi dividida em duas partes. A primeira é uma pesquisa sobre os temas: Linha de Produto de Software, abordagem *SMarty* e técnicas de inspeção, em especial as técnicas baseada em perspectiva e a *SMartyCheck*, uma técnica de leitura baseada em *checklist* para diagramas *SMarty* de caso de uso, classe e componente (Bettin *et al.*, 2018; Geraldi *et al.*, 2015).

Já na segunda parte da **revisão bibliográfica**, foi identificadas as perspectivas que podem ser relacionadas especificamente à Engenharia de Domínio considerando os diagramas UML com variabilidades, pois, na Engenharia de Aplicação, o produto já foi configurado e não há mais notações de variabilidades para serem gerenciadas. Logo, na Engenharia de Aplicação, podem ser utilizadas técnicas de inspeção de software para o desenvolvimento tradicional de software.

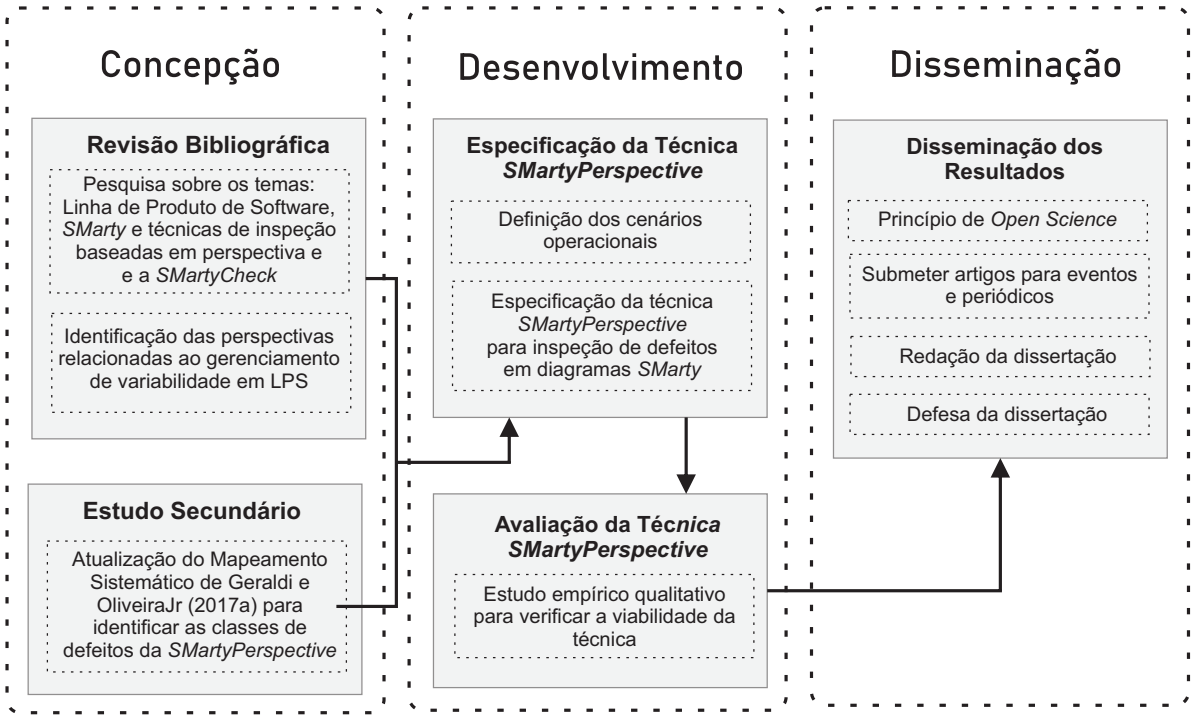
Na construção de um cenário operacional PBR deve ser considerado o ambiente no qual o processo de inspeção será aplicado, os atributos de qualidade esperados (taxonomia de defeitos) e os papéis relacionados aos artefatos que serão inspecionados (Ciolkowski *et al.*, 1997).

Logo, além das perspectivas, devem ser definidas as classes de defeitos da técnica *SMartyPerspective* que irão englobar os atributos de qualidade esperados nos diagramas *SMarty*. Portanto, o estudo secundário teve como objetivo atualizar o mapeamento sistemático de Geraldi e OliveiraJr (2017a) para identificar as classificações de defeitos para artefatos de software nos últimos 3 anos, no período de maio/2017 a maio/2020 para definição das questões dos cenários e, assim, atender os objetivos específicos deste trabalho.

Com base nas informações obtidas dos estudos realizados na fase de **Concepção**, a especificação e avaliação da técnica PBR para a inspeção de defeitos em diagramas *SMarty*, a *SMartyPerspective*, foi realizada na fase de **Desenvolvimento** deste trabalho.

Foram definidos primeiramente os cenários operacionais, descritos por meio de *template* do Laitenberger e Kohler (2001) para introdução, instrução e questões. Para tal, foram definidos a classificação de defeitos e as perspectivas com base no estudo secundário da fase anterior e da busca na literatura pelas perspectivas de Engenharia de Domínio, respectivamente, e como se relacionam com os diagramas *SMarty* e diagrama de *features*.

Figura 1.1: Etapas da Metodologia de Desenvolvimento de Pesquisa



Fonte: o autor

Com os cenários operacionais que constituem a *SMartyPerspective* definidos, o próximo passo a ser executado nesta pesquisa foi a avaliação da técnica, por meio de um estudo empíricos qualitativo que foi realizado *online* devido às condições de saúde pública.

O estudo qualitativo foi realizado com pesquisadores da área de Engenharia de Software familiarizados com a abordagem *SMarty* ou inspeção de software, com o objetivo de avaliar a viabilidade do uso da *SMartyPerspective* para detecção de defeitos em diagramas *SMarty* e do refinamento dos cenários para cada uma das perspectivas. A avaliação da técnica se deu por meio de um questionário de acordo com o modelo TAM (Modelo de Aceitação da Tecnologia) (Davis, 1989) sob as dimensões de facilidade uso, utilidade, atitude para uso e intenção comportamental. Além de perguntas abertas que foram analisadas utilizando *coding* do método qualitativo *Grounded Theory* (Corbin e Strauss, 2014; Strauss, 1987).

Para a injeção dos defeitos nos diagramas UML foi realizado uma análise dos estudos de Bettin *et al.* (2018); Chren *et al.* (2019); Geraldi e OliveiraJr (2017a); Ma (2017), para identificar quais classes de defeitos há uma maior dificuldade em serem encontrados pelos

participantes deste estudo, para que a injeção de defeitos não se torne uma ameaça à validade do estudo.

Por fim, a fase de **Disseminação** iniciou com a redação desta dissertação e continuará com a defesa, e escrita e submissão de artigos sobre a especificação da *SMartyPerspective* e do estudo empírico conduzidos na fase de **Desenvolvimento** com o apoio dos princípios de *Open Science*.

1.5 Organização do Texto

Esta dissertação está estruturada da seguinte forma: O Capítulo 2 apresenta o levantamento bibliográfico sobre os principais conceitos necessários para a definição da *SMartyPerspecive*, como LPS, diagramas UML *SMarty*, inspeção de software e técnicas de inspeção; o Capítulo 3 apresenta o processo de desenvolvimento da família de técnicas *SMartyPerspecive*;a documentação final da técnica com os cenários e explicações de cada uma das perspectivas é apresentada no Capítulo 4; exemplos de inspeção utilizando os cenários da técnica são apresentadas no Capítulo 5; o Capítulo 6 apresenta o estudo qualitativo desenvolvido; e o Capítulo 7 apresenta as contribuições, limitações e trabalhos futuros.

2

Fundamentação Teórica

2.1 Considerações Iniciais

O processo de desenvolvimento de software precisou mudar e avançar conforme o mercado competitivo exigia das empresas softwares eficientes, confiáveis, com custo reduzido de esforços e tempo (Soares, 2016).

Para tal, o reúso surgiu como uma solução para reutilizar inicialmente apenas sub-rotinas do sistema, mas com o tempo, mas as necessidades competitivas das organizações em produzir sistemas diferentes e personalizados fez com que a reutilização expandisse para além do código, englobando todas as características (*assets*) do sistema por meio da abordagem de Linha de Produto de Software (LPS) (Santos *et al.*, 2019).

Neste capítulo é apresentada uma revisão teórica necessária ao desenvolvimento do trabalho. A Seção 2.2 inicia com os conceitos de LPS e a abordagem *SMarty* para gerenciamento de variabilidades. Em seguida, na Seção 2.3 é descrita a atividade de inspeção de software e as principais técnicas de leitura utilizadas por revisores de software para colaborar com esse processo. Na Seção 2.3.3 é apresentada a Leitura Baseada em Perspectiva, que é a técnica base para alcançar o objetivo deste trabalho.

A Seção 2.4 descreve os trabalhos relacionados a esta dissertação. Em especial, a técnica de inspeção *SMartyCheck* que pode ser utilizada para colaborar com a qualidade de diagramas *SMarty* para casos de uso, classe e componente na detecção de defeitos.

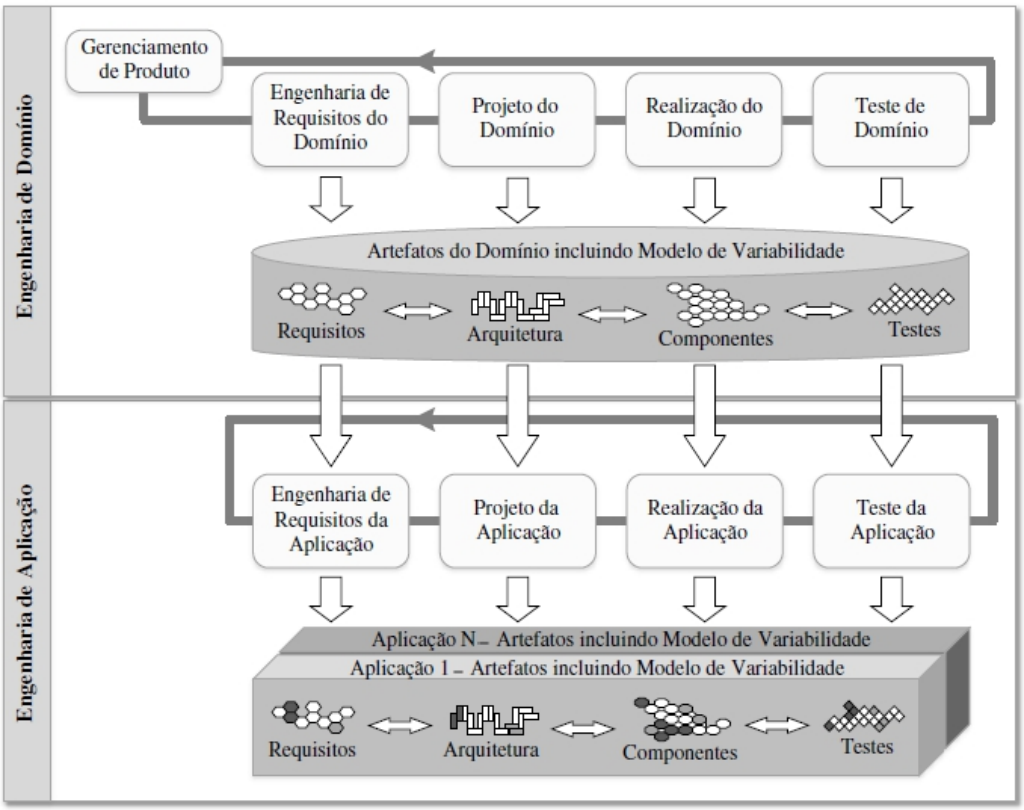
2.2 Linha de Produto de Software

Uma LPS é definida por Clements e Northrop (2002) como um conjunto de sistemas de software que compartilham características comuns e gerenciáveis, que satisfazem as necessidades ou missões de um segmento particular de mercado e são desenvolvidos a partir de artefatos comuns e essenciais, como: documentação, arquitetura, componentes reutilizáveis, etc.

Diferente de outras abordagens de reúso de software, uma LPS não considera um sistema como sendo único, mas, como pertencente à uma determinada família. Para Gomaa (2006), uma família possui recursos que são comuns, exigidos por todos os membros, e variáveis, que permitem diferenciar produtos dentro da sua família. Tais recursos formam o núcleo de artefatos, que é à base de uma LPS.

Os recursos variáveis definem a variabilidade de software, que é a capacidade ou habilidade de um sistema ou artefato ser eficientemente estendido, customizado ou configurado para ser usado em um determinado contexto (Van Gurp *et al.*, 2001).

Figura 2.1: Framework de Engenharia de LPS



Fonte: Geraldi *et al.* (2015) *apud* Pohl *et al.* (2005)

A Engenharia de Linha de Produto de Software estabelece processos e metodologias para o uso de LPS no desenvolvimento de software (Pohl *et al.*, 2005). Possui atividades essenciais que são interligadas e iterativas. Essas atividades são: Engenharia de Domínio e Engenharia de Aplicação (Figura 2.1).

O desenvolvimento do núcleo de artefatos, também conhecido como Engenharia de Domínio é um processo em que é criado o núcleo de artefatos, nele está incluso os artefatos reutilizáveis e os que são variáveis para facilitar o processo de reúso (Pohl *et al.*, 2005).

O núcleo de artefatos evolui enquanto novos produtos são desenvolvidos, revisados ou incluídos. Ele inclui tudo que é produzido durante o desenvolvimento de software, como a arquitetura de LPS, componentes reutilizáveis, modelos de domínios, requisitos da LPS, planos de teste e modelos de características e de variabilidades (OliveiraJr *et al.*, 2010b).

O desenvolvimento de produtos específicos, também conhecido como Engenharia de Aplicação, é um processo em que acontece a configuração de um produto individual, reutilizando artefatos de domínio e explorando a variabilidade de LPS. As novas características produzidas do produto serão mais tarde anexadas ao domínio da LPS para serem reutilizáveis (Pohl *et al.*, 2005).

A atividade de gerenciamento permite a evolução e manutenção da LPS com o controle da coordenação e distribuição dos recursos para as atividades. Portanto, ela está ligada em todo o ciclo de vida de desenvolvimento da LPS.

O gerenciamento é dividido em gerenciamento organizacional, que mitiga os riscos que ameaçam o sucesso da LPS com a identificação das restrições de produção, controle das atividades já desenvolvidas e o planejamento de novas funcionalidades, e gerenciamento técnico, que supervisiona o desenvolvimento de ativos e produtos garantindo que todos os envolvidos estejam trabalhando nas atividades necessárias (Northrop, 2002).

Na medida em que uma LPS evolui, espera-se uma ampla garantia de qualidade e produtos confiáveis, pois, um artefato pode ser diversas vezes reutilizado, o que aumenta a chance de serem detectados e corrigidos defeitos durante as atividades essenciais (Pohl *et al.*, 2005). Entretanto, um processo mal planejado e gerenciado poderá fazer com que artefatos desenvolvidos com defeitos durante a Engenharia de Aplicação sejam reportados para a plataforma na Engenharia de Domínio que poderão serão reutilizados na LPS afetando todos os produtos.

Para atingir a qualidade de software deve ser garantido que requisitos específicos de LPS, como o suporte a variabilidade, flexibilidade (capacidade de prover facilmente mudanças), evolutividade (qualidade de evolução da arquitetura de acordo com as mudanças) e capacidade de manutenção sejam atendidos (Pohl *et al.*, 2005).

Portanto, devem ser estabelecidas técnicas de garantia de qualidade para verificação e validação dos artefatos que são gerados, como teste e revisões minuciosas do projeto durante os processos da Engenharia de Domínio e Engenharia de Aplicação (Pohl *et al.*, 2005).

O gerenciamento de variabilidades é uma das atividades mais importantes no controle de uma LPS, já que inclui as atividades de identificar variabilidade, representá-la explicitamente em artefatos de software durante todo o ciclo de vida e rastrear as dependências entre variabilidades (Pohl *et al.*, 2005). Logo, o sucesso de uma LPS está relacionado à capacidade da organização em administrar as variabilidades e produzir sistemas customizados e que se destaquem no mercado.

2.2.1 A Abordagem *SMarty*

A abordagem *SMarty* (*Stereotype-based Management of Variability*) foi proposta para o gerenciamento de variabilidades de LPS baseadas em modelos UML (OliveiraJr *et al.*, 2010b). Encontra-se na versão 5.2 com suporte aos diagramas de: caso de uso, classe, atividade, componente e sequência.

SMarty permite o gerenciamento das variabilidades de uma LPS de maneira sistemática por meio do perfil (*SMartyProfile*), que representa de maneira gráfica as variabilidades por meio da UML e um processo (*SMartyProcess*), que define diretrizes para guiar o usuário na identificação, representação e no rastreamento de variabilidades (OliveiraJr *et al.*, 2010b).

Uma das vantagens dessa abordagem está relacionada com a extensão dos elementos que fazem parte do metamodelo da notação UML, a possibilidade de utilizar ferramentas que manipulam estes modelos, além da representação das relações entre variabilidade e suas variantes de maneira compatível com estes metamodelos.

O *SMartyProfile* é um conjunto de estereótipos e meta-atributos que representam os principais conceitos de gerenciamento de variabilidade: variabilidade, ponto de variação, variante e restrições variantes em modelos UML para LPS.

Os estereótipos e meta-atributos da *SMartyProfile* são:

- «`variability`» este estereótipo representa a variabilidade em uma LPS, mostra em forma de comentário que em determinado elemento do modelo há uma variabilidade. É composta pelos meta-atributos:
  - `name`: nome que referencia uma variabilidade;

  - `minSelection`: quantidade mínima de variantes a serem selecionadas para resolver um ponto de variação ou variabilidade;
  - `maxSelection`: quantidade máxima de variantes a serem selecionadas para resolver um ponto de variação ou variabilidade;
  - `bindingTime`:define o momento no qual uma variabilidade será resolvida;
  - `allowsAddingVar`: apresenta se há a possibilidade de inclusão de novas variantes após uma variabilidade ser resolvida;
  - `variants`: coleção de instâncias associada à variabilidade;
  - `realizes-`: coleção de variabilidades de modelos de nível inferior que realiza a variabilidade; e
  - `realizes+`: coleção de variabilidades de modelos de nível superior que realiza a variabilidade.

- `«variationPoint»` representa o conceito de ponto de variação. Define o local em que se encontra um ponto de decisão na LPS. Associado a este estereótipo tem os meta-atributos: `numberOfVariants` (número de variantes), `bindingTime` (momento que o ponto de variação deverá ser resolvido), `allowsAddingVar` (define se há a possibilidade de inserir uma nova variante após resolução da variabilidade naquele ponto), `variants` (coleção de variantes associadas) e `variabilities` (coleção de variabilidades associadas);

- `«variant»` define o conceito de variante e é uma extensão abstrata da UML, por isso não pode ser aplicado a nenhum elemento de um modelo UML;

- `«mandatory»`: uma variante com este estereótipo é obrigatória, por isso deve estar presente em qualquer configuração da LPS a qual pertence;

- `«optional»` uma variante opcional pode ou não estar em uma determinada configuração de uma LPS;

- `«OR»` este estereótipo é relacionado a resolução de um ponto de variação de maneira inclusiva. Para isso, uma das variantes associadas deve ser selecionada para resolver o ponto de variação;

- `«XOR»` é relacionado a resolução de um ponto de variação de maneira exclusiva. Para isso, apenas uma das variantes associadas deve ser selecionada para resolver o ponto de variação;

- «mutex» representa o conceito de restrição entre variantes. Se uma variante for selecionada, obrigatoriamente a variante dependente não poderá estar na configuração da LPS;

- «requires» representa o conceito de restrição entre variantes. Uma variante só será selecionada para uma configuração de LPS se a variante dependente estiver presente na configuração; e

- «variable» é uma extensão da metaclasse UML Component que indica que um componente tem um conjunto de classes com variabilidades explícitas.

O *SMartyProcess* acontece em paralelo com o desenvolvimento de uma LPS adotando as atividades comuns de um processo de uma LPS. É interativo, pois ocorre ao final de cada atividade executada durante o desenvolvimento de LPS e incremental, pois o número de variabilidades tende a crescer na medida em que as atividades são executadas e o núcleo de artefatos é alimentado (OliveiraJr *et al.*, 2010b).

As diretrizes para dar suporte à identificação e representação de variabilidade são tanto gerais para aplicação do *SMartyProfile*, quanto específicas para cada um dos diagramas UML suportados pela abordagem *SMarty*.

A Figura 2 ilustra a modelagem do diagrama de casos de uso com base na abordagem *SMarty* para Arcade Game Maker (AGM) (Institute, 2017), uma LPS pedagógica para três jogos diferentes: Pong, Bowling e Brickles, com regras e movimentações específicas para cada um.

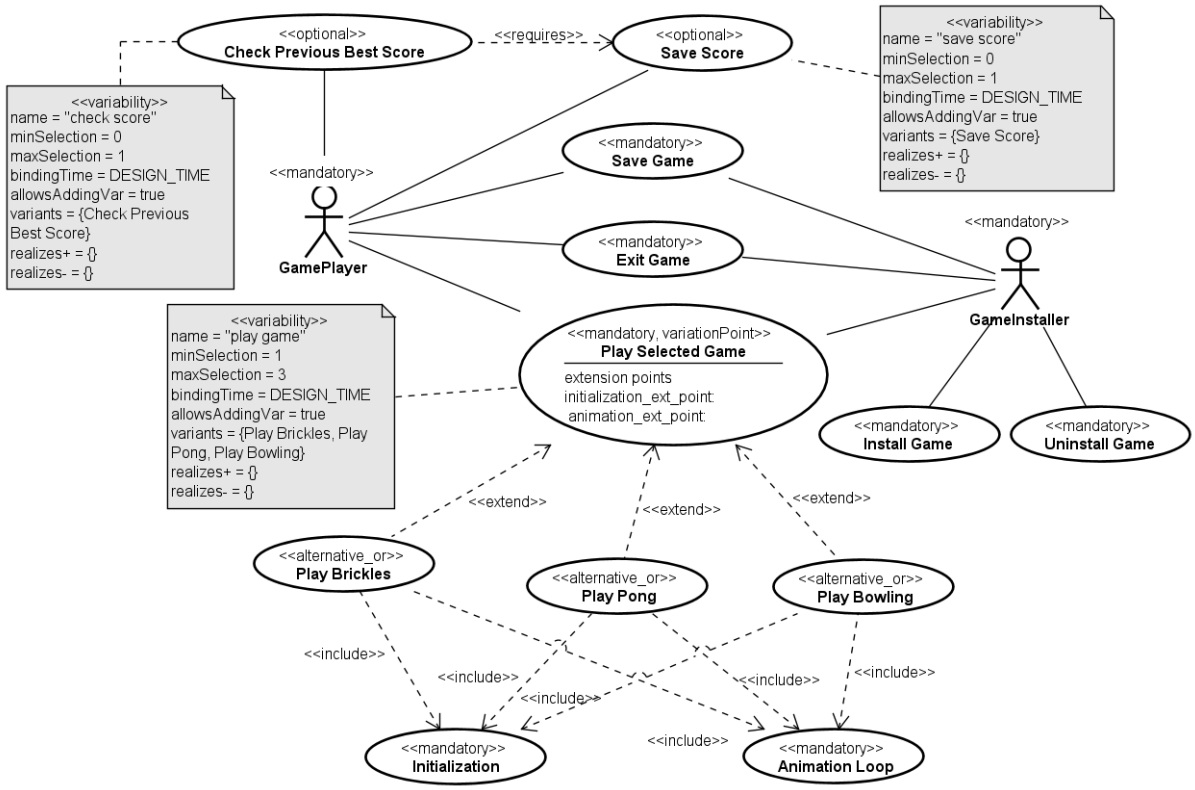
Neste modelo é possível observar além dos casos de uso que variam, os jogos que devem ser escolhido pelo jogador, os obrigatórios, como o InstallGame, Exit Game, UninstalGame, que instala, fecha e desinstala o jogo respectivamente.

2.2.2 Papéis em LPS

Coallier e Champagne (2005) acreditam que no gerenciamento e desenvolvimento de LPS os envolvidos devem ter uma visão que transcenda a entrega dos projetos atuais e seja alinhada aos objetivos do negócio e não apenas para solucionar o propósito do *stakeholder* atual, desenvolvendo assim, uma arquitetura que permita uma visão de longo prazo para que integre junto a ela componentes para serem reutilizados.

Baseado nas habilidades para solucionar as atividades essenciais de LPS, o SEI (*Software Engineering Institute* definiu em seu *framework* PLP (*Product Line Practice*), 29 áreas de atuação que compreendem uma coleção de atividades para a organização obter

Figura 2.2: Diagrama de caso de uso *SMarty* da AGM



Fonte: Bera *et al.* (2015)

sucesso, sendo elas divididas em três categorias: Engenharia de Software, Gerenciamento Técnico e Gerenciamento Organizacional (Northrop *et al.*, 2007).

A prática de Engenharia de Software inclui as áreas necessárias para aplicar a tecnologia para criação e evolução dos ativos principais e produtos: definição de arquitetura, avaliação da arquitetura, desenvolvimento de componentes, mineração de ativos existentes, engenharia de requisitos, integração de sistemas de software, teste, compreensão de domínios relevantes e uso de software disponível externamente (Northrop *et al.*, 2007).

O grupo de Gerenciamento Técnico inclui as práticas para gerenciar o desenvolvimento e evolução dos produtos e ativos principais: gerenciamento de configurações, análise *make/buy/mine/commission*, medição e rastreamento, disciplina do processo, escopo, planejamento técnico, gerenciamento técnico de riscos e suporte de ferramenta (Northrop *et al.*, 2007).

O gerenciamento organizacional contém as atividades para a orquestração de todo o esforço de LPS: construindo um caso de negócios, gerenciamento de interface do cliente, desenvolvendo uma estratégia de aquisição, financiamento, lançamento e institucionalização, análise de mercado, operações, planejamento organizacional, gerenciamento de risco organizacional, estruturando a organização e previsão de tecnologia e treinamento (Northrop *et al.*, 2007).

Já Coallier e Champagne (2005), inspirados no framework PLP e VRASP definiram 31 áreas de atuação divididas em 5 grupos:

- Gerenciamento de linha de produtos: compreende as áreas de atuação em nível organizacional: Definir o escopo de LPS, definir a visão de LPS, realizar análise de mercado da LPS, realizar análise estratégica de LPS, gerenciar a evolução da LPS, estabelecer e gerenciar um modelo de LPS, gerenciar atuação da LPS, realizar comunicação, suporte e apoio da LPS.

- Gerenciamento de arquitetura: preocupam-se com a elaboração e arquitetura de LPS: definir a visão arquitetural, definir e manter requisitos de arquitetura, definir a arquitetura, especificar e otimizar a arquitetura, executar gerenciamento de riscos arquiteturais, definir e implementar utilização de COTS/OSS e executar ativos principais/particionamento do produto;

- Gerenciamento de requisitos: incluem as práticas padrões de engenharia de Software em LPS: Elicitação de requisitos, executar análise e consolidação de requisitos, estabelecer e manter especificação de requisitos, executar verificação e validação de requisitos e gerenciamento de requisitos;

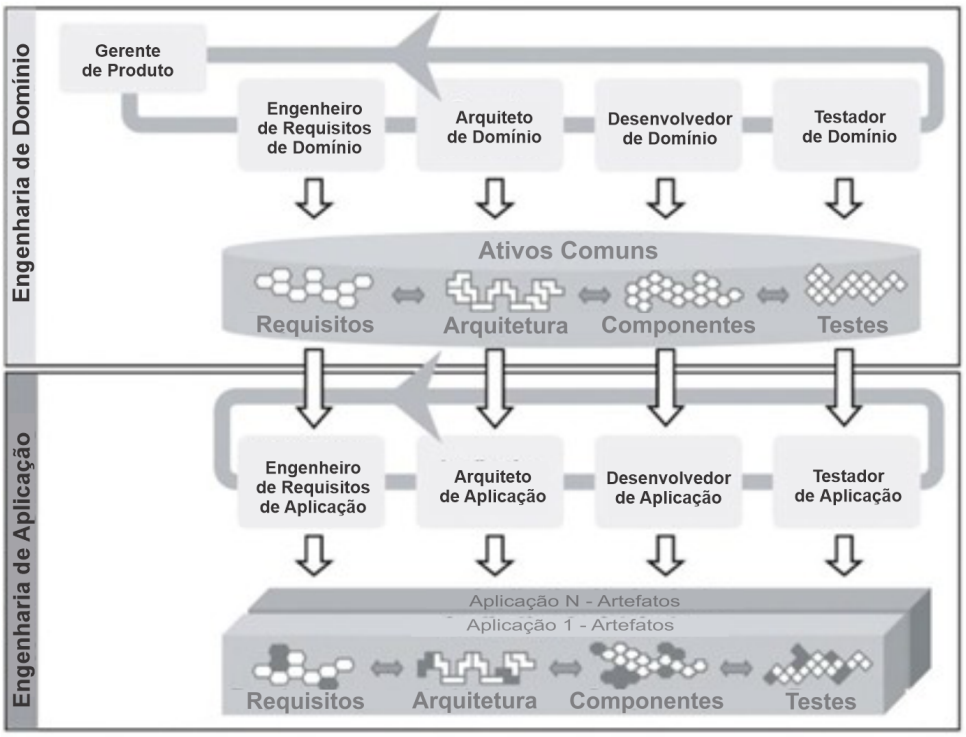
- Desenvolvimento e gerenciamento de ativos: estas áreas preocupam-se com a gestão de recursos ativos usados na LPS em seu ciclo de vida: estabelecer e manter repositório de ativos, garantir integridade dos ativos principais, realizar mineração dos ativos principais, estabelecer e manter um processo de produção, executar o gerenciamento de liberação e executar manutenção dos ativos principais; e

- Síntese e suporte de produtos: áreas de atuação para o desenvolvimento e suporte dos produtos individuais de uma LPS: executar o gerenciamento da interface do usuário, performance da síntese do sistema, executar engenharia do cliente, executar síntese do sistema, suporte ao produto e executar manutenção do sistema.

Van der Linden *et al.* (2007) estrutura os papéis e responsabilidades da Engenharia de Aplicação e de Domínio (Figura 2.3). Na primeira, o trabalho deve ser concentrado em

entregar uma plataforma com consistência e alta qualidade, enquanto na Engenharia de Aplicação, em render receita a empresa com a entrega rápida dos sistemas baseados na plataforma de ativos utilizando dos mecanismos de variabilidade para a configuração das aplicações.

Figura 2.3: Papéis em LPS



Fonte: Traduzido de Van der Linden *et al.* (2007)

O Gerente de Produto é um papel que pode ser encontrado em ambas as atividades de LPS. Já para a Engenharia de Domínio os papéis específicos são: Arquiteto de Domínio, Engenheiro de Requisitos de Domínio, Desenvolvedor de Domínio, Testador de Domínio e Gerente de Ativos do Domínio (Van der Linden *et al.*, 2007). Enquanto para a Engenharia de Aplicação: Engenheiro de Requisitos de Aplicação, Arquiteto de Aplicação, Desenvolvedor de Aplicação e Testador de Aplicação (Van der Linden *et al.*, 2007).

A seguir são explicadas as principais funções de cada um dos papéis de Van der Linden *et al.* (2007), baseadas em suas definições e na dos processos de Pohl *et al.* (2005). Para fins de entendimento da diferença entre funções na Engenharia de Aplicação e Engenharia de Domínio, elas foram agrupadas por sua função mais geral, como por exemplo: arquiteto.

Gerente de Produto

O Gerente de Produto é responsável pelo planejamento e evolução do portfólio de acordo com os objetivos da empresa. Este papel está presente tanto na Engenharia da Aplicação, quanto na de Domínio, porém seus esforços são maiores na Engenharia de Domínio, pois é quando o portfólio da LPS é definido com as características comuns e variáveis dos produtos futuros derivados da LPS, bem como seu valor de negócio e estratégia de mercado.

Engenheiro de Requisitos

O engenheiro de requisitos de domínio analisa, identifica e documenta os requisitos comuns e variáveis para a base de ativos de acordo com o plano definido pelo Gerente de Produto, provendo o *feedback* dos custos e da viabilidade dos requisitos para ele, assim como sugestões de novas características para a LPS, além de antecipar as potenciais mudanças dos requisitos para os produtos futuros, como os padrões.

Além das tarefas já conhecidas de um engenheiro de requisitos no processo tradicional de desenvolvimento, como elicitação, documentação, negociação, validação e verificação e gerenciamento dos requerimentos, o engenheiro de domínio deve analisar as comunalidades, analisar as variabilidades e modelá-las para que os requisitos satisfaçam todas as aplicações que podem ser derivadas da LPS e com a documentação explícita das variabilidades para que sirva de guia para as outras atividades do processo.

Já na Engenharia de Aplicação, não são definidos novos requisitos, eles são derivados a partir dos requisitos comuns e variáveis definidos para a LPS na Engenharia de Domínio, deve-se se preocupar aqui com a escolha dos requisitos de acordo com a capacidade da plataforma e dos componentes que ela contém.

Arquiteto de Domínio e de Aplicação

O arquiteto de domínio é responsável pelo desenvolvimento e manutenção da arquitetura de referência da gama de produtos. Ele valida se os projetos dos ativos reutilizáveis atendem a arquitetura de referência e determina os mecanismos de configuração que serão usados nas aplicações.

Este papel difere do processo tradicional de software pois ele deve incorporar a configuração de variabilidades na arquitetura de referência, além de se preocupar com sua flexibilidade, para que ela possa cumprir a exigência de todos as aplicações que podem ser configuradas e as regras comuns que devem ser seguidas a partir da arquitetura de referência. Com base nisso, é função do arquiteto de domínio definir quais serão os

componentes que serão desenvolvidos pelo desenvolvedor na Engenharia de Domínio como também na Engenharia de Aplicação.

Já na Engenharia de Aplicação, o arquiteto não desenvolve nada, ele apenas deriva da arquitetura de referência já definida quais serão as configurações utilizadas a partir das escolhas do cliente, por isso, ele deve cumprir as regras estabelecidas pelo arquiteto de domínio, para quem deve se reportar também caso haja algum problema com a arquitetura de referência. Assim, seu papel é determinar quais componentes e interfaces específicos deverão ser selecionados da base de ativos para a aplicação em desenvolvimento.

Desenvolvedor de Domínio e de Aplicação

Os componentes e interfaces reutilizáveis são implementados pelo desenvolvedor de domínio seguindo o projeto arquitetural do arquiteto de domínio. Desenvolve também os mecanismos de configuração para o suporte das variâncias entre os produtos da LPS.

Sua principal diferença em relação ao processo tradicional de software para sistemas únicos, é que na Engenharia de Domínio a realização é fracamente acoplada, pois não há a implementação de uma aplicação funcionando, apenas os componentes que integrarão a base de ativos. O desenvolvedor de domínio deve então, se preocupar com os diferentes contextos da LPS e desenvolver os componentes e interfaces que atendam todas as necessidades e todas as possíveis configurações da LPS.

Enquanto na Engenharia de Aplicação, há a entrega de uma aplicação funcionando, os componentes da base de ativos são selecionados e configurados pelo desenvolvedor para o desenvolvimento da aplicação única, ou seja, não são criados novos componentes na Engenharia de Aplicação, eles são apenas combinados para satisfazer as necessidades do cliente.

Testador de Domínio e de Aplicação

O testador de domínio provê feedback a todos os outros papéis, informando a testabilidade dos requisitos e das escolhas de projeto. Ele gerencia e desenvolve os ativos de testes reutilizáveis da base de ativos. Além disso, faz testes de integração e de sistemas em ativos de domínio, planejando a estratégia do que será testado em nível de domínio e de aplicação.

O testador de domínio não testa uma aplicação completa, mas sim, componentes separadamente e partes integradas da LPS que contém partes variáveis, diferentemente do testador de aplicação que faz testes adicionais para verificar defeitos nas configurações e da seleção de variantes de acordo com os requisitos da aplicação.

Gerente de Ativos de Domínio

A gerência das versões e das variantes dos ativos de domínio, assim como sua rastreabilidade é responsabilidade do gerente de ativos de domínio por meio de um controle de versão. Este papel deve manter a configuração e versão válida dos ativos de domínio e sua rastreabilidade para serem usadas em todas as etapas de desenvolvimento por outros papéis.

2.3 Inspeção de Software

Um software com qualidade deve estar em conformidade com os requisitos explícitos do usuário e com as características implícitas que são esperadas de todo software (Pressman e Maxim, 2016). Quando se trata de uma LPS, em que se deve ter confiança nos artefatos reutilizáveis, um defeito não encontrado poderá afetar não só um produto, mas toda uma família.

Para Zhu (2016) é impossível não injetar defeitos durante o desenvolvimento de software, pois muitos dos artefatos são gerados por humanos, e que a melhor coisa a se fazer, então, é removê-los antes que atinjam os usuários ou que passe para a próxima fase de desenvolvimento, o que pode custar até 10 vezes mais à organização. Sendo assim, para atingir certo nível de qualidade é importante haver uma gestão de qualidade de software.

O gerenciamento de qualidade estabelece a infraestrutura que dá suporte a qualquer tentativa de construir um produto de software de alta qualidade (Pressman e Maxim, 2016). Inclui os processos necessários para garantir que o projeto irá satisfazer as necessidades e requisitos do usuário, por meio de atividades que determinam as políticas de qualidade, objetivos e responsabilidades e a implementação do processo de qualidade (PMBOK).

As atividades que englobam o gerenciamento de qualidade são: planejamento de qualidade, em que são identificados os padrões de qualidade relevantes ao projeto, garantia de qualidade, que compreende a estruturação, sistematização e execução das atividades para assegurar a qualidade, e controle de qualidade, um processo para monitorar se o projeto está atendendo aos padrões de qualidade no processo de desenvolvimento (Bartié, 2002).

A garantia de qualidade de software é uma abordagem estruturada que envolve as atividades de contenção, prevenção, detecção e remoção de defeitos (Zhu, 2016) que tem como desafio tornar o risco de encontrar defeitos o mais próximo de zero possível (Bartié,

2002), definindo como a qualidade do produto pode ser atingida e como a organização sabe que ele atingiu o grau de qualidade necessário (Sommerville, 2011).

Atividades de verificação e validação (V&V) auxiliam na quantificação do processo de garantia de qualidade ao considerar que o processo não foi conduzido de maneira correta na organização e que há a possibilidade de haverem defeitos no produto que devem ser corrigidos (Koscianski e dos Santos Soares, 2007). Assim, o objetivo da V&V é estabelecer a confiança de que o software atende as necessidades do usuário e o propósito para o qual foi criado (Sommerville, 2011).

A verificação é uma técnica estática que checa se o software está sendo desenvolvido de acordo com as especificações do usuário, enquanto a validação, uma técnica dinâmica, tem como objetivo garantir que o software atende às expectativas do usuário, mesmo as implícitas que não foram especificadas nos requisitos, por meio de testes em protótipos ou em programas executáveis (Sommerville, 2011; Zhu, 2016).

A grande vantagem da verificação é que ela pode ser utilizada em todas as fases do desenvolvimento de software, inclusive nas etapas iniciais, para detectar defeitos em modelos de especificação do sistema e do projeto sem necessitar de uma versão executável. Além disso, o processo de verificação não precisa ser interrompido quando for detectado um defeito como no teste de software, que pode mascarar defeitos, e ao remover um deles, o teste deve ser refeito (Sommerville, 2011).

A revisão de software pode ser aplicada para a verificação da qualidade de um produto. Segundo Pressman e Maxim (2016), a revisão é um "filtro" para purificar os artefatos gerados no processo de software por meio da descoberta de erros que podem ser removidos. Sua necessidade está aliada a dois fatos: (i) que os artefatos são gerados por humanos, assim, podem haver erros; e (ii) erros são dificilmente detectados por quem cria o artefato. Portanto, é importante a leitura dos documentos por outra pessoa: o revisor.

Para diminuir os custos e esforços da revisão é importante detectar os defeitos antes de serem retrabalhados no processo (Fagan, 1979) e se disseminem pelas próximas atividades e artefatos. Para essa atividade, a norma IEEE 1028 (IEEE, 2008) descreve cinco tipos de revisão de software: revisão gerencial, revisões técnicas, inspeção, *walk-throughs* e auditoria.

Na revisão gerencial uma avaliação sistemática é realizada em nome ou pela gerência de organização. Enquanto na revisão técnica toda a equipe é responsável por fazer essa avaliação, examinando e adequando o produto de software conforme necessário por meio de perguntas e comentários sobre as possíveis anomalias (IEEE, 2008).

A revisão *walk-throughs* é uma técnica estática em que um grupo de desenvolvimento se reúne para encontrar defeitos e melhorar o produto, ficando a cargo da equipe concordar

com o caminho a ser seguido nas mudanças. Já na auditoria, a avaliação do produto de software é executada de maneira independente por terceiros para avaliar conforme especificações, acordos ou critérios definidos (IEEE, 2008).

Um exame visual na identificação de defeitos é realizado revisão técnica formal, mais conhecida como inspeção, que guia o revisor na leitura do software por meio de um conjunto de instruções (Felizardo, 2004; IEEE, 2008). É essencial "para entender uma dada representação do artefato de software e compará-la com um conjunto de expectativas em relação à estrutura, conteúdo e qualidade" (Biffl e Halling, 2003), sendo considerada um dos métodos mais eficientes e efetivos que detecta antecipadamente as falhas para que não se propaguem durante o processo de desenvolvimento.

A inspeção é um processo bem definido de revisão por pares para verificar as propriedades de qualidade de produtos de software por meio da busca por erros na leitura formal, eficiente e econômica de um artefato (Fagan, 1979). A inspeção pode ser aplicada ao final de todas as etapas de desenvolvimento de software assim que um artefato for criado, o que reduz os custos de retrabalho, já que defeitos são geralmente encontrados próximos ao ponto em que foram inseridos (Höhn, 2003) e se passarem para a próxima atividade podem custar até dez vezes mais (Zhu, 2016).

Para realizar a inspeção de software, Fagan (1979) descreveu o processo (Figura 2.4) sob seis atividades:

1. Planejamento: nessa etapa são definidos os artefatos que serão revisados, os integrantes da equipe de software que farão parte da revisão e como será realizada a sessão de inspeção;

2. Visão geral: é apresentada ao revisor uma visão geral do material que será inspecionado, das regras e atributos de qualidade esperados. Essa etapa pode ser realizada em conjunto com a próxima, ficando a cargo do planejamento da organização;

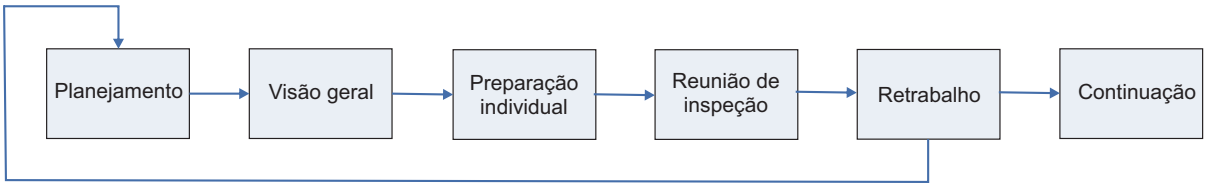
3. Preparação individual: os responsáveis pela revisão são treinados e se preparam para a atividade examinando o documento para compreendê-lo;

4. Reunião de inspeção: é realizada nesta etapa a sessão de inspeção por meio da análise e avaliação dos documentos para detectar e caracterizar os defeitos;

5. Retrabalho: o responsável pelo artefato o reavalia e então com base na inspeção realizada corrige os defeitos detectados. Após essa fase é feita uma nova inspeção para verificar o novo estado do documento; e

6. Continuação: nesta etapa são assegurados que todos os defeitos reportados foram corrigidos e durante o retrabalho não foram inseridos novos.

Figura 2.4: Processo de Inspeção de Fagan



Fonte: traduzido de Lahtinen (2012)

Um revisor de artefato "deve ter conhecimento sobre o domínio geral para garantir que o artefato descreve um sistema que seja significativo e possa ser construído" (Travassos *et al.*, 1999). Logo, a maneira com que um inspetor lê o artefato durante a reunião de inspeção contribui para a detecção de defeitos. Assim, o processo individual da leitura é considerado uma atividade chave para essa atividade.

Para dar suporte ao revisor durante a leitura de um documento na prática, existem as técnicas de leitura, que fornecem uma série de instruções para o leitor sobre como ler e o que deve ser procurado no artefato (Basili *et al.*, 1996a). Elas proporcionam o entendimento geral da atividade de inspeção por meio do processo característico da técnica escolhida e as classes de defeito cobertos por ela para a verificação de atributos de qualidade.

As técnicas de leitura são um meio fundamental para obter a alta qualidade de um produto (Basili *et al.*, 1996a) buscando aumentar a eficácia do inspetor na V&V do software (Travassos *et al.*, 1999), por isso, foram criadas diversas técnicas para atender necessidades específicas dos projetos de software (Zhu, 2016).

Entre as técnicas de leitura, a menos formal é a *Ad hoc*, que não fornece orientações explícitas ao revisor de como deve proceder ou o que deve ser procurado (Höhn, 2003). Ele fica dependente apenas da sua capacidade de leitura e da experiência adquirida ao longo do tempo (Biffl e Halling, 2003).

As técnicas de Leitura Sistemáticas orientam o leitor em como devem ser extraídas as informações e como avaliá-las de acordo com os requisitos de qualidade. Essas técnicas possuem características importantes de usabilidade, adaptabilidade, repetibilidade, cobertura e foco (Biffl e Halling, 2003), como as técnicas da família da Leitura Baseada em Cenários. Já entre as técnicas não-sistemáticas, que não fornece diretrizes de como ler o artefato, a Leitura Baseada em *Checklist* se destaca.

2.3.1 Leitura Baseada em *Checklist*

Na Leitura Baseada em *Checklist* (CBR) os leitores recebem uma lista (*checklist*) de questões com dicas e recomendações para que possam encontrar defeitos nos artefatos de software (Zhu, 2016). A técnica mostra "o que" os inspetores devem inspecionar e não "como" essa atividade deve ser realizada.

Os itens individuais da lista de verificação priorizam diferentes defeitos, enumeram características, ou ainda, fazem perguntas que ajudam os revisores a descobrir defeitos com sugestões da forma que ele deve procurar no documento. Assim, as características de qualidade são definidas a priori pela organização (Travassos e Biolchini, 2007).

A CBR é considerada não-sistemática, pois os leitores não possuem nenhuma estratégia para ler o documento e responder o *checklist*, geral e idêntica, já que os leitores inspecionam sob a mesma ótica, com as mesmas perguntas e conferem todos os aspectos do documento (BERTINI, 2006).

Geraldi e OliveiraJr (2017b) destacam os benefícios da CBR encontrados por diferentes autores: redução do nível de defeitos de um software de modo relevante, redução de custos relacionados ao ciclo de vida, tempo de teste e de entrega do produto e melhora da produtividade em equipe.

Na CBR as questões são feitas de acordo com as experiências passadas dos leitores, que permite que classes de defeitos que não tenham sido detectadas possam passar despercebidas e não estarem no *checklist* (Höhn, 2003). Além disso, uma mesma lista de verificação pode ser utilizada para diferentes projetos, assim, a má adaptação e a repetição podem ser um problema para o processo de revisão (Geraldi *et al.*, 2015).

Como a produtividade é individual e todos os inspetores ficam responsáveis por procurarem todas as classes de defeitos sem ter qualquer reavaliação de um determinado documento por outra pessoa (Höhn, 2003), o processo pode ser afetado, pois não se pode garantir que o inspetor leu o documento de forma correta mesmo com a lista de verificação tendo os atributos de qualidade (Travassos e Biolchini, 2007).

2.3.2 Leitura Baseada em Cenários

(Parnas e Weiss, 1987) criticaram as técnicas de leitura convencionais (CBR e *Ad hoc*) ao apontar problemas relacionados à grande quantidade de documentação existente em um projeto. Eles alegam que os revisores são solicitados a examinar questões além de sua competência, já que nem todos estão familiarizados com todas as partes do projeto por serem especialistas em um determinado aspecto do sistema e a pressa em terminar a leitura pode fazer com que a revisão não tenha o efeito esperado (Parnas e Weiss, 1987).

Para resolver essa questão Parnas e Weiss (1987) propuseram que cada inspetor deve fazer a revisão de acordo com sua própria experiência, focando a atenção em um aspecto da avaliação que domina, o que torna a inspeção mais eficiente que as técnicas convencionais.

Para conduzir a revisão do artefato de modo a levar em consideração as especialidades e experiências, a Leitura Baseada em Cenários (*Scenario-Based Reading - SBR*), uma família de técnicas, foi projeta sob dois fatores chaves (Lanubile *et al.*, 2004): (i) orientação ativa, já que o leitor trabalha ativamente com o documento enquanto é orientado na leitura; e (ii) separação de interesse, pois, restringe o foco do leitor para o que ele deve inspecionar de acordo com o aspecto específico do documento de seu interesse.

A SBR decompõe cenários do processo de desenvolvimento de software por meio de diferentes caminhos (Zhu, 2016). Um cenário para SBR pode ser descrito como uma "descrição do mundo real que envolve atores interagindo dentro de um determinado contexto" (Breitman e do Prado Leite, 1998), são as descrições das situações que podem acontecer durante o uso do sistema (Zhu, 2016). Ele fornece procedimento para o inspetor de como extrair as informações e de como examiná-las para detectar defeitos (Laitenberger e Kohler, 2001).

Durante o ciclo de vida de desenvolvimento de um sistema os seus cenários evoluem, podendo mudar em um conjunto de cenários (inter cenário) ou dentro do mesmo âmbito (intra cenário) (Breitman e do Prado Leite, 1998). Estudos realizados por Breitman e do Prado Leite (1998) apontaram que os cenários evoluem em grupo, com relações fortemente acopladas de equivalência, coincidência parcial e total, entre outras.

A SBR utiliza o conceito de relação entre os cenários partindo da premissa que se cada um dos inspetores utilizar diferentes técnicas sistemáticas e ler sobre uma determinada ótica específica do projeto, quando juntarem todos os cenários, terão uma maior cobertura do documento, combinação dos cenários (Zhu, 2016) e classes de defeitos, sejam eles sob a ótica de defeitos específicos do ponto de vista dos usuários dos produtos gerados.

Entre as técnicas da família SBR destacam-se: a Leitura Baseada em Defeitos, em que cada leitor procura uma classe de defeito específico e os itens da *checklist* são substituídos por procedimentos designados para implementá-los e a Leitura Baseada em Perspectiva, que o leitor procura defeitos sob objetivos específicos dos consumidores do documento (Zhu, 2016).

2.3.3 Leitura Baseada em Perspectiva

Um artefato pode dar suporte a diferentes usuários com necessidades específicas para a realização de suas tarefas durante o processo de desenvolvimento. Por exemplo, a partir de

um mesmo documento de requisitos, um usuário verifica se os requisitos descritos captam adequadamente as funcionalidades que necessita, enquanto um projetista o utiliza como base para o desenvolvimento, e o testador define um plano de testes para garantir que o software esteja de acordo com os requisitos funcionais e de desempenho descritos (Höhn, 2003).

A Leitura Baseada em Perspectiva (*Perspective-Based Reading* - PBR), proposta por Basili *et al.* (1996a), é uma família de técnicas de leitura que contempla os diferentes objetivos dos consumidores do artefato, como por exemplo, as perspectivas de um desenvolvedor, usuário e testador. Cada leitor fica responsável por uma perspectiva particular, e ao juntá-las, terá uma maior cobertura do documento sob diferentes visões dos envolvidos (Zhu, 2016), por isso pode ser aplicada durante todo o ciclo de vida do projeto.

A PBR colabora com os revisores para que eles respondam duas questões sobre os artefatos inspecionados: "quais informações nesses artefatos devem ser verificadas? Como identificar defeitos nessas informações?" (Shull *et al.*, 2000).

Para a criação de um modelo de PBR devem ser atingidas algumas metas: (i) adaptar a técnica segundo o documento e sua notação; (ii) adaptar a técnica ao projeto e as características do ambiente; (iii) a técnica deve ser detalhada o suficiente para colaborar com o inspetor, possibilitando que seja menos dependente da experiência do revisor de artefato; (iv) ser focada numa perspectiva particular; e (v) deve ser específica (Ciolkowski *et al.*, 1997).

Para Shull *et al.* (2000) os benefícios de utilizar diferentes perspectivas podem ser definidos por meios dos atributos que caracterizam a técnica:

- sistemática: a técnica fornece aos revisores dos artefatos um procedimento definitivo para guiá-lo em como o artefato deve ser lido e verificado e se os usos identificados dos requisitos podem ser alcançados;

- focada: as diferentes perspectivas permitem aos leitores focarem em tipos de defeitos específicos para aquele cenário, ao invés de procurar por todos os defeitos possíveis. Esta característica colabora para a eficácia da técnica, já que permite que os revisores encontrem mais defeitos e evita o esforço duplicado entre membros da equipe, já que os leitores detectam apenas defeitos relacionados ao seu ponto de vista particular;

- orientada a objetivo e personalizável: em cada organização seus revisores definem quais serão as perspectivas selecionadas para refletir os usos dos requisitos e os objetivos específicos da inspeção, adaptando o procedimento às suas necessidades; e

- transferível por meio de treinamento: como a PBR depende do procedimento definido pela organização e não só da experiência do revisor, um novo revisor pode receber o treinamento dos passos do procedimento.

Cenários para PBR

Um cenário é uma coleção de procedimentos que operacionaliza estratégias para a detecção de defeitos (de Mello *et al.*, 2014). Na PBR, um cenário é uma descrição algorítmica do documento das atividades e questões a partir do artefato inspecionado, que direciona como o inspetor deve proceder durante a leitura do documento de um produto de software sob uma perspectiva em particular (Ciolkowski *et al.*, 1997; Laitenberger e Kohler, 2001).

Um cenário em PBR é desenvolvido "com base no conhecimento sobre o ambiente no qual o processo de leitura é aplicado: papéis no processo de desenvolvimento de software e classes de defeitos" (Ciolkowski *et al.*, 1997), orientando o leitor em como ler e examinar as informações no documento trabalhando ativamente com o artefato para ganhar uma compreensão mais profunda ao longo da leitura para que possa se concentrar depois em defeitos mais sutis (Denger e Ciolkowski, 2003).

A orientação ativa auxilia o leitor a estruturar e reduzir as informações do documento de modo a facilitar a inspeção por meio da construção de um modelo mental, caracterizada como compreensão, dos objetos e relações semânticas do documento. Esse processo deve ser apoiado pela coerência do cenário e na capacidade de evitar a sobrecarga cognitiva (Ciolkowski *et al.*, 1997; Laitenberger e Kohler, 2001). Portanto, a eficiência da PBR é determinada pelos cenários, por isso, devem ser gerados bons cenários para atender os atributos de qualidade esperados pela organização em seus artefatos (Basili *et al.*, 1996a).

A Figura 2.5 apresenta um excerto do cenário proposto por Sabaliauskaite *et al.* (2003) para o papel de *designer* de sistema. Para os autores, esse papel é responsável pelos diagramas de classe e de sequência. No excerto, é possível visualizar o passo 3 do cenário em que os inspetores devem relacionar os objetos existentes entre esses dois diagramas para a identificar defeitos.

Figura 2.5: Excerto do cenário de Designer

CENÁRIO DE DESIGNER Assuma que você é um designer de sistema. A preocupação do projetista é definir a estrutura estática do sistema (Diagrama de Classe), a fim de garantir que o comportamento requerido seja obtido em termos de interações entre objetos (Diagrama de Sequência). Por favor, execute as tarefas seguindo os passos de 1 a 3...		
Passo 3	**Localize o Diagrama de Classe e Diagrama de Sequência**	
	Diagramas de Sequência mostram interações entre objetos no sistema. Essas interações ocorrem em uma determinada sequência no momento apropriado. Por favor, faça uma lista de Classe. Certifique-se de que todos os objetos dos Diagramas de Sequência estejam definidos no Diagrama de Classe. Examine as mensagens entre os OBjetos no Diagrama de Sequência para certificar-se de que eles são definidos como métodos ou atributos da Classe correspondente no Diagrama de Classe também. Examine os relacionamentos entre Objetos no Diagrama de Sequência. Certifique-se de que a reação entre dois Objetos que existe no Diagrama de Sequência exista entre as mesmas Classes de Objeto em Diagramas de Classes. Depois disso, responda às seguintes perguntas.	
	3.1.	Todos os Objetos do Diagrama de Sequência estão definidos no Diagrama de Classe?
	3.2.	Tods as mensagens entre objetos do Diagrama de Sequência são definidos como métodos de atributos da classe correspondente no Diagrama de Classe?
	3.3.	O relacionamento entre dois objetos que existem no Diagrama de Sequêcia, também existem entre as mesmas classes de objetos no Diagrama de Classe?
	3.4.	Não há elementos redundantes ou ausentes no Diagrama de Sequência.

Fonte: traduzido de Sabaliauskaite *et al.* (2003)

Papéis

A PBR fornece um conjunto de revisões individuais em que cada uma delas é um ponto de vista específico de um usuário de artefato de software e juntas cobrem todos os aspectos relevantes do documento de requisitos. Respeitando assim, a especialidade de um papel e os aspectos importantes no documento para a realização das tarefas particulares a ele de acordo com seu conhecimento natural sobre o artefato.

O papel determina as perspectivas utilizadas na PBR e na forma como o leitor deverá ler o documento e quais produtos deverão ser inspecionados, assim, as perspectivas devem ser construídas a partir dos papéis mais relevantes que as pessoas assumem no processo de desenvolvimento e manutenção de software, para atender as necessidades específicas da organização ou do projeto em questão.

Na literatura há diversos exemplos de papéis que foram definidos para atender as necessidades particulares de cada estudo de técnicas PBR. No trabalho de Sabaliauskaite et al. (2003) foram definidos os papéis de usuário, designer e desenvolvedor. Para Laitenberger e Atkinson (1999) foram identificados engenheiro de requisitos, designer,

testador e mantenedor. Enquanto no trabalho de Laitenberger e Debaud (1997) os de analista, testador de módulo e testador de integração.

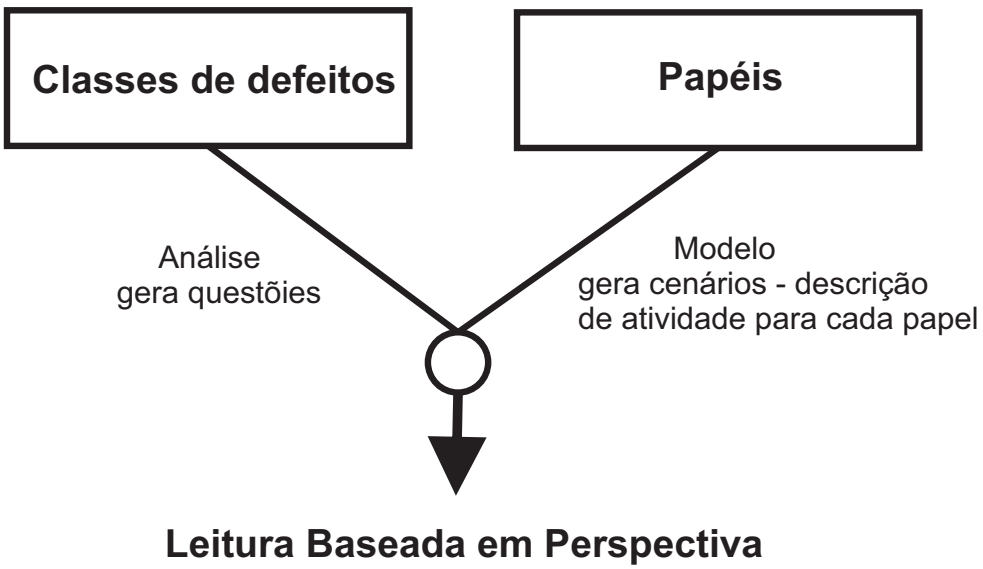
Depois de definidos os papéis relevantes para a criação dos cenários, é necessário determinar quais defeitos podem existir nos documentos definidos, para, então, definir as classes de defeitos que serão atendidas pela técnica.

Criação de cenários PBR

Basili *et al.* (1996a) definiram uma abordagem para gerar um cenário operacional em PBR (Figura 2.6) em que os diversos papéis e a descrição das atividades realizadas por eles em um processo de desenvolvimento de software são usados como modelo para gerar os cenários, enquanto que a caracterização das classes de defeitos com o refinamento dos papéis, são utilizadas para gerar as questões para os inspetores realizarem a análise (Basili *et al.*, 1996a; Ciolkowski *et al.*, 1997). As questões e cenários gerados formam o procedimento PBR respeitando o conjunto de objetivos da organização.

Laitenberger e Atkinson (1999) descreveram uma estrutura genérica de um cenário de leitura para PBR composta de três partes: introdução, instruções e questões. A introdução contém uma breve descrição da perspectiva, o papel que o instrutor assumirá no decorrer da atividade de inspeção e os atributos de qualidade mais relevantes.

Figura 2.6: Gerando cenários para PBR



Fonte: traduzido de Basili *et al.* (1996a)

As instruções descrevem as atividades que serão executadas pelo inspetor durante a inspeção e como devem ser feitas. Laitenberger e Kohler (2001) defendem a leitura das instruções, pois, ajudam a decompor grandes documentos em partes menores o que facilita detecção de grandes defeitos, além de exigir que os inspetores trabalhem ativamente com o documento a ser examinado e estejam preparados e concentrados nas informações que são relevantes ao papel que está executando, evitando sobrecarga com detalhes desnecessários (Laitenberger e Kohler, 2001).

Questões são feitas para que os inspetores julguem as informações lidas no documento, verificando se preenchem um conjunto de fatores de qualidade previamente determinados por meio de uma série de perguntas sobre aspectos específicos da informação (Laitenberger e Kohler, 2001).

Laitenberger e Kohler (2001) mencionam que o sucesso da PBR depende da capacidade do Engenheiro de Software seguir os cenários durante a inspeção e de criar novos cenários, caso seja preciso acrescentar novos papéis ou artefatos. Para colaborar com a atividade de criação de novos cenários definiram sete etapas para este processo, sendo elas:

1. Determine o documento a ser inspecionado: a primeira etapa do processo consiste em determinar quais são as informações que devem atender certo nível de qualidade e em que documentos elas são encontradas, sejam eles descrições textuais, design textual ou modelos gráficos, ou, contêm informações relevantes sobre o sistema;

2. Especifique as partes interessadas para o tipo de documento: depois de identificados os documentos, são necessários especificar quais papéis utilizam os artefatos definidos em suas funções;

3. Crie tabela com mapeamento de partes interessadas para documentos: com os artefatos e papéis definidos, o próximo passo é identificar para cada um dos papéis quais são os artefatos e que tipo de informações importantes para que realize as tarefas das especificadas para a sua função. Para representar esta relação é criada um mapeamento entre os artefatos e os papéis por meio de uma tabela;

4. Configure a parte de introdução do cenário: com as perspectivas definidas, o próximo passo é selecionar quais são as informações mais importantes para cada uma delas realizar sua tarefa dentro da organização e como as informações devem ser identificadas e extraídas. Estas informações são necessárias para a configuração das introduções dos cenários;

5. Configure a parte de instrução do cenário: a PBR é uma técnica sistemática, por isso, devem ser repassadas orientações para inspetor de como ele deverá olhar para

o artefato inspecionado, sendo assim, são criadas instruções detalhadas em como ele deverá extrair as informações e como devem ser documentadas após a análise;

6. Configure a parte da questão do cenário: as perguntas que o inspetor deverá responder durante a inspeção são configuradas nesta etapa com base em seu entendimento das informações extraídas durante a leitura. Devem levar em consideração as classes de defeitos apontadas com base nos problemas particulares da organização; e

7. Crie (se necessário) a documentação necessária para as instruções: caso seja necessário, são criados os documentos com informações necessárias para o leitor realizar a inspeção.

2.3.4 Classes de Defeitos

Basili *et al.* (1996a) ao propor a técnica PBR definiu que "um defeito em documento de requisitos é uma omissão, imprecisão, inconsistência, ambuiguidade ou qualquer coisa que leve à uma solução insatisfatória do problema a ser resolvido".

Para a norma IEEE (2010), um defeito é uma imperfeição ou deficiência em um produto de software, que não atende os requisitos e especificações, portanto deve ser reparado ou substituído.

Laitenberger e Atkinson (1999) acreditam que a definição de defeito limita a PBR por tratar a correção como único fator de qualidade, e, para eles, as diferentes perspectivas podem ter outras medidas de qualidade, citando o exemplo do mantenedor que espera que um artefato além de correto, seja bem construído e de fácil modificação para a manutenção.

Para resolver esta questão, Laitenberger e Atkinson (1999) trocaram o termo defeito para falha, que é definida por eles como "qualquer propriedade de um artefato ou descrição que a impeça de atender aos requisitos de qualidade" (Laitenberger e Atkinson, 1999).

A IEEE (2010) relaciona que uma falha é um subtipo de defeito e que apesar de toda falha ser um defeito, um defeito não é uma falha quando ele for detectado por inspeção ou análise estática e removido antes do software ser executado. Seguindo os conceitos fundamentais de PBR e devido à relação entre falha e defeito, neste trabalho será mantido o termo defeito com extensão da definição para englobar as diferentes medidas de qualidade envolvidas nas várias perspectivas que podem ser utilizadas por uma organização.

Uma taxonomia de defeitos bem definida colabora com a eficácia das técnicas e do processo de inspeção, pois, "é útil para apoiar o gerenciamento de projetos, aprendizagem

e consequente melhoria da qualidade" (Travassos, 2014) ao orientar os revisores durante a inspeção e ajudar a detectar falhas persistentes e tipos de defeitos já conhecidos para aqueles tipos de artefatos (Alshazly *et al.*, 2014; Geraldi e OliveiraJr, 2017a).

Não há uma classe de defeito universal para ser utilizada em conjunto com a PBR. A escolha fica a cargo da organização que deve considerar os atributos de qualidade definidos por ela para seus produtos e as informações que são relevantes para o sistema que está sendo desenvolvido. Para tal, devem ser consideradas as informações dos artefatos gerados para assegurar que o sistema descrito por ele corresponde ao sistema que deveria estar sendo desenvolvido (Travassos *et al.*, 1999).

Entre as taxonomias utilizadas nos trabalhos de técnicas de leitura baseadas em PBR, muitos deles utilizam direta ou indiretamente as taxonomias semelhantes de Basili *et al.* (1996b); Shull (1998) que foram definidas especialmente para especificação de requisitos. Entre os trabalhos Ciolkowski *et al.* (1997); Lahtinen (2012); Mafra e Travassos (2005); Sabaliauskaite *et al.* (2003), podemos citar o de Travassos *et al.* (1999) que adaptaram a taxonomia de Basili *et al.* (1996a) para modelos Orientados a Objetos.

Os defeitos que compõem as classificações de Basili *et al.* (1996b); Shull (1998) para especificação de requisitos são: omissão, ambiguidade, informação estranha, inconsistência, fato incorreto e outros:

- omissão: acontece quando algum requisito importante relacionado a funcionalidade de algum requisito que não foi incluso, ou nem todas as situações de entrada de dados possíveis de dados foram definidas, falta informações nas seções no documento de requisitos ou da definição de termos e unidades de medidas;
- ambiguidade: quando vários termos são utilizados para representar a mesma característica ou múltiplos significados de um termo para um contexto específico;
- inconsistência: este defeito acontece quando uma informação do artefato revisado está em discordância com outra informação do documento, ou seja, ocorre pois dois ou mais requisitos são conflitantes;
- fato incorreto: um fato afirmado em um requisito e que não pode ser verdadeiro sob condições especificadas para o sistema;
- informação estranha: são as informações consideradas desnecessárias ou que não são utilizadas; e
- outros: envolve qualquer outro erro não classificado pelo inspetor nas outras categorias.

Principais tipos de defeitos em diagramas UML

Dois estudos identificaram os tipos de defeitos mais comuns cometidos para alguns tipos de diagramas UML (Chren *et al.*, 2019; Ma, 2017). Estes trabalhos colaboram assim, para a definição de taxonomia de defeitos para inspeção diagramas UML, pois, norteiam os tipos de defeitos mais cometidos.

No trabalho de Ma (2017) foram identificados os tipos de defeitos mais cometidos em diagramas de casos de uso, de classe e de sequência de 68 estudantes de graduação do terceiro ano do curso de Ciência da Computação sob os aspectos da sintática, pragmática e semântica.

Chren *et al.* (2019) catalogaram mais de 2.700 diagramas de 123 alunos durante o semestre do curso de Engenharia de Software da Faculdade de Informática na Masaryk University. Dos diagramas ensinados durante o curso, foram incluídos no catálogo os: diagrama de classe, de casos de uso, sequência, comunicação, atividades, máquina de estados e entidade-relacionamento.

Foram agrupados nas Tabela 2.1, Tabela 2.2 e Tabela 2.3 os principais tipos de defeitos encontrados nos estudos de Ma (2017) e Chren *et al.* (2019) para diagramas de caso de uso, classe e sequência, respectivamente. Nenhum dos trabalhos catalogou os principais defeitos cometidos nos diagramas de componentes.

Tabela 2.1: Tipos de defeitos mais encontrados em diagramas de caso de uso

Ma (2017)	Chren *et al.* (2019)
Notações não padronizadas de casos de uso ou atores	Erros na especificação do texto de casos de uso
Nomenclatura inadequada para casos de uso	Relação "incluir"inadequada
Casos de uso sem descrição de interação	Nomenclatura do casos de uso
Descrição incompleta da etapa	Uso indevido de incluir / estender / herança para decomposição funcional
Descrição incorreta da etapa	
Nomenclatura ambígua para casos de uso ou atores	
Tamanhos irracionais para casos de uso (ou seja,casos de uso com muita ou pouca função)	
Descrição ambígua da etapa	

Tabela 2.2: Tipos de defeitos mais encontrados em diagrama de classe

Ma (2017)	Chren *et al.* (2019)
Notações não padronizadas de classes e relações	Falta de operações de classes
"Nomenclatura imprópria para classes, atributos, operações e relações"	Falta de dependências
Classes sem operações ou atributos	Falta de argumentos em operações
Atributos ou operações em locais errados	Tipo de método errado/faltante.
Intervalos de cardinalidade incorretos para associações	
Definição imprópria de relações	
Tamanhos irracionais para as classes	

Tabela 2.3: Tipos de defeitos mais encontrados em diagrama de sequência

Ma (2017)	Chren *et al.* (2019)
Notações não padronizadas de objetos ou mensagens	Nomenclatura incorreta da linha de vida
Nomenclatura inadequada para objetos e mensagens ou mensagens sem nomes	As mensagens não são consistentes com os diagramas de classes
Mensagens que não estão entre os objetos de classes relacionadas	Uso de mensagens assíncronas em vez de síncronas
Mensagens sem operações relacionadas em diagramas de classe	A mensagem usa entidade desconhecida
Nomenclatura ambígua para mensagens ou objetos	

2.4 Trabalhos Relacionados

Nesta seção serão discutidos os estudos relacionados a este trabalho. Não foram encontrados trabalhos que relacionavam os temas PBR e LPS ou PBR com *SMarty*. Portanto, três dos trabalhos envolvem técnicas de CBR para LPS, sendo um específico para diagramas *SMarty*. E o outro é uma avaliação experimental que compara CBR e PBR em diagramas UML.

2.4.1 SPLIT e FMCheck

As técnicas FMCheck e SPLIT são técnicas de inspeção para LPS. A FMCheck, por meio de três atividades, apoia a verificação semântica do diagrama de *features* com base na CBR (de Mello *et al.*, 2014). A técnica foi avaliada por meio de um experimento e uma replicação deste experimento comparando-a com a técnica *Ad hoc*. Em ambos os casos, ela se mostrou mais eficiente que a *Ad hoc* para diagramas de *features* (Souza *et al.*, 2016).

A técnica SPLIT (*Software Product Line Inspection Techniques*) desenvolvida por Cunha *et al.* (2012) é um conjunto de técnicas de inspeção CBR para avaliar modelos de LPS, comparando documentos de requisitos de software, mapa de produtos e diagramas

de *features*.Em dois estudos, a técnica encontrou um número maior de defeitos do que uma abordagem de inspeção baseada em tipo de defeitos.

Há duas diferenças entre estes trabalhos em relação a proposta da *SMartyPerspective*: primeiramente as técnicas apesar de serem desenvolvidas para LPS, são voltadas apenas para diagrama de *features*, enquanto o objetivo deste trabalho é criar uma técnica para inspeção de diagramas de gerenciamento de variabilidades em LPS que englobe tanto o diagrama de *features* como os diagramas *SMarty*.

A segunda diferença está relacionada à abordagem de inspeção. Enquanto as técnicas citadas utilizam um *checklist* para a inspeção, a *SMartyPerspective* propõe uma técnica PBR que define a inspeção a partir de cenários e perspectivas.

Apesar das diferenças apresentadas entre a *SMartyPerspective* e a técnica FMCheck, elas se assemelham não só na inspeção de diagramas de *features*, como também na taxonomia de defeito. Ambas adaptaram a taxonomia de (Travassos *et al.*, 1999).

2.4.2 PBR para Diagramas UML

Sabaliauskaite *et al.* (2002) realizaram uma avaliação experimental para comparar as técnicas de leitura CBR e PBR em relação ao tempo individual da inspeção, custo por defeito encontrado e eficácia individual e da equipe simulada de 3 integrantes para documento de requisitos de projetos orientados a objetos escritos em linguagem UML ((Sabaliauskaite *et al.*, 2002, 2003).

Para o estudo, os autores elaboraram o *checklist* da CBR e os cenários de PBR para os papéis de usuário, desenvolvedor e projetista para os diagramas UML de classe, atividade, sequência e componentes. As classes de defeitos que foram utilizadas por eles foram: defeitos sintáticos, que incluem omissão e informação extra, semânticos, que são os fatos incorretos e ambiguidade e de consistência, que correspondem aos defeitos de inconsistência.

Apesar deste estudo não ser relacionado diretamente com LPS e diagramas *SMarty*, que correspondem ao artefatos objetivo deste trabalho, o estudo propõe cenários específicos para diagramas UML que podem colaborar com a especificação da técnica *SMartyPerspective*.

Laitenberger *et al.* (2000) também definiram em seu estudo uma técnica de inspeção baseada em PBR e a avaliaram por meio de um experimento controlado com CBR com 18 participantes. Deste estudo experimental o resultado mostrou que a PBR foi mais eficaz que a técnica CBR.

Para o estudo de Laitenberger *et al.* (2000) foram definidos os cenários de desenvolvedor, projetista e testador por meio do processo definido por Laitenberger e Atkinson (1999) para inspecionar os diagramas de: caso de uso, sequência, classe e colaboração. Os cenários desenvolvidos por Laitenberger *et al.* (2000) colaboraram com este trabalho para o entendimento do processo que será utilizado e uma visão geral das questões e dos atributos de qualidade esperados nos diagramas UML que são inspecionados por estes cenários.

2.4.3 *SMartyCheck*

A técnica *SMartyCheck*, atualmente na versão 3.0 (Bettin *et al.*, 2018; Geraldi *et al.*, 2015; Vieira, 2017), foi projetada com base na técnica CBR com o objetivo de detectar defeitos em diagramas *SMarty* de casos de uso, classe e componente por meio de um *checklist*.

A técnica pode ser utilizada para dois fins: (i) inspecionar artefatos de uma LPS por meio de diagramas UML *SMarty* de casos de uso, classe e componente; ou (ii) inspecionar as várias versões de uma LPS que pode ser modelada e comparada com base em uma LPS original (oráculo) da qual foi derivada (Geraldi e OliveiraJr, 2017b; Vieira, 2017).

Na *SMartyCheck* cada diagrama suportado pela técnica tem uma lista de verificação específica para que a inspeção seja organizada e direcionada. A lista de verificação contém os tipos de defeitos pré-definidos com sua descrição, local com o questionamento com um *check* (sim/não) que identifica se o defeito foi encontrado e se for, um campo para escrever as observações sobre ele (Geraldi e OliveiraJr, 2017b). A Figura 2.7 apresenta parte do *checklist* da *SMartyCheck* para diagrama de caso de uso e classe.

Figura 2.7: Parte do *checklist* da *SMartyCheck* para diagrama de caso de uso e classe

Tipo de Defeito	Itens do Checklist	Sim	Não	Defeito Identificado
Regras de Negócio (BR)	**BR. 1** O caso de uso/classe não é claro com o propósito e as funcionalidades desejadas com base no domínio.			
Inconsistência (Incons)	**Incons.1** Existe algum componente especificado com o estereótipo <<variationPoint>> cujo o número de variantes especificadas é maior ou menor que o definido em maxSelection ou minSelection da variabilidade (<<variability>>) associada?			
	Incons.2 Existe algum componente especificado com o estereótipo <<optional>> cujo número de variantes especificadas é maior ou menor que o definido em maxSelection ou minSelection da variabilidade (<<variability>>) associada?			

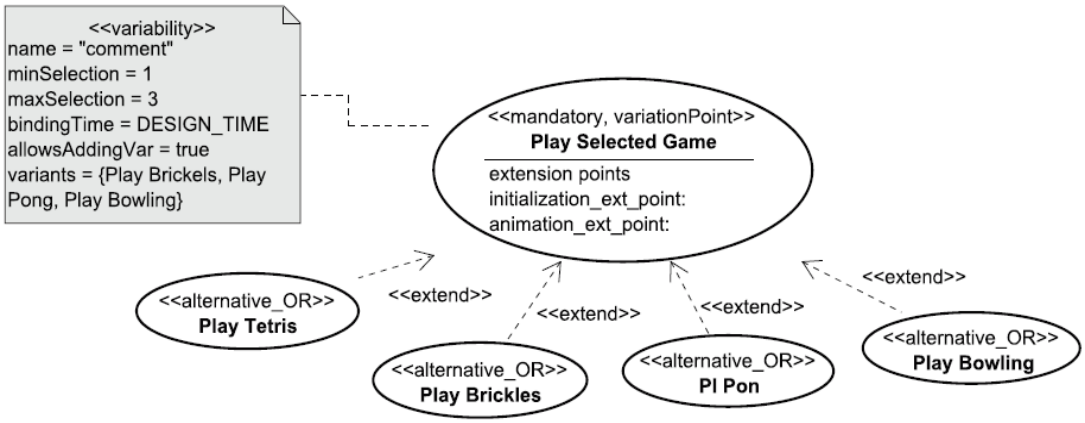
Adaptado de: (Geraldi e OliveiraJr, 2017b)

Na totalidade das três listas de verificação (diagrama de casos de uso, classe e componente) os defeitos pré-definidos da técnica são: regras de negócio, inconsistência, fato

incorreto, imodificável, omissão, informação estranha, desvio intencional, ambiguidade, anomalia, instável, inviável e omissão (Geraldi e OliveiraJr, 2017b).

A Figura 2.8 mostra um exemplo de um defeito que pode ser identificado utilizando a lista de verificação para casos de uso da *SMartyCheck* (Figura 2.7). Tendo a Figura 2.2 como oráculo (diagrama sem defeitos), temos na Figura 2.8 um defeito de Inconsistência, pois há mais casos de uso como alternativa para solução do ponto de variação `Play Select Game` (`Play Tetris`, `Play Brickles`, `Play Pong` e `Play Bowling`) do que foi especificado como máximo na variabilidade associada (`maxSelection` = 3).

Figura 2.8: Exemplo de tipo de defeito



Fonte: (Geraldi e OliveiraJr, 2017b)

A técnica *SMartyCheck* foi avaliada empiricamente duas vezes. No primeiro estudo obtiveram-se evidências inicias da efetividade, eficácia e eficiência da técnica perante *Ad hoc* (Geraldi e OliveiraJr, 2017b) para modelos *SMarty* de casos de uso e de classe. Já no segundo, as evidências iniciais apontaram para a eficiência e efetividade da técnica em relação a Ad hoc (Bettin *et al.*, 2018) para diagramas de *SMarty* de componentes.

Como o objetivo deste trabalho é especificar e avaliar a técnica *SMartyPerspective*, a *SMartyCheck* está fortemente relacionada a ela por também ser uma técnica de inspeção para detecção de defeitos em diagramas *SMarty*. A grande diferença entre as técnicas *SMartyCheck* e *SMartyPerspective* se deve à abordagem de inspeção utilizada como base para sua criação: CBR e PBR, respectivamente.

2.4.4 Comparação Entre os Trabalhos Relacionados

No total, cinco trabalhos são relacionados com o presente trabalho. Três deles apresentam técnicas de inspeção para diagramas de gerenciamento de variabilidades com base na CBR

(Cunha *et al.*, 2012; Geraldi e OliveiraJr, 2017b; de Mello *et al.*, 2014). Já os trabalhos de Sabaliauskaite *et al.* (2002) e Laitenberger *et al.* (2000) colaboraram para o estudo do comportamento das técnicas de inspeção PBR para diagramas UML.

A Tabela 2.4 apresenta os principais conceitos de comparação entre os estudos relacionados com a proposta da técnica *SMartyPerspective*:

- técnica base: refere-se a técnica que foi utilizada pelos autores como base para a criação da técnica. Três trabalhos utilizaram CBR como base e duas delas a PBR.

- gerência de variabilidades (GV): apresenta se técnica definida nos estudos foram definidas para o gerenciamento de variabilidades. Três trabalhos foram específicos para o gerenciamento de variabilidades: as técnicas FMCheck, SPLIT e *SMartyCheck*. As duas primeiras para diagramas de *features* e a *SMartyCheck* para diagramas *SMarty* de caso de uso, classe e componente. Apesar dos estudos de Sabaliauskaite *et al.* (2002) e Laitenberger *et al.* (2000) não serem específicos para a GV, eles se relacionam a este trabalho por serem técnicas PBR para diagramas UML;

- UML: refere-se aos trabalhos que definiram técnicas de inspeção para diagramas UML. Os trabalhos de Sabaliauskaite *et al.* (2002) e Laitenberger *et al.* (2000) são específicos para diagramas UML. Já o de Geraldi e OliveiraJr (2017b) são para diagramas UML com base na abordagem *SMarty* para GV em LPS;

- diagramas inspecionados: são identificados os diagramas inspecionados pela técnica. Duas delas são exclusivas para os diagramas de *features*. Os outros trabalhos são referentes à diagramas UML, entre eles estão os de caso de uso, classe, componente, sequência, atividade e colaboração.

- estudo: nesta coluna estão os estudos experimentais das técnicas. Todas elas foram avaliadas empiricamente por meio de um estudo quantitativo que as comparou com outras técnicas. A técnica CBR foi a mais utilizada nos estudos, pois, três técnicas que relacionam LPS e inspeção foram baseadas neste tipo de técnica e os dois estudos que usam PBR como base, utilizaram a técnica CBR para comparação.

- Colaboração: as técnicas SPLIT e FMCheck colaboraram com este trabalho com entendimento geral da atividade de inspeção para LPS e diagramas de *features*. Já a técnica *SMartyCheck* por inspecionar diagramas *SMarty* colaborou com o entendimento para inspeção destes artefatos específicos. Por fim, as técnicas PBR de Laitenberger *et al.* (2000) e de Sabaliauskaite *et al.* (2002) colaboraram com a especificação de cenários para diagramas UML em relação as instruções e questões.

Tabela 2.4: Comparação entre os trabalhos relacionados

Trabalho	**Nome da Técnica**	**Técnica Base**	**GV**	**UML**	**Diagramas inspecionados**	**Estudo**	**Colaboração**
de Mello *et al.* (2014)	FMCheck	CBR	Sim	Não	*features*	FMCheck X Ad hoc	Inspeção para LPS e diagrama de *features*
Cunha *et al.* (2012)	SPLIT	CBR	Sim	Não	*features*	SPLIT X DBR	
Geraldi e OliveiraJr (2017b)	SMartyCheck	CBR	Sim	Sim	caso de uso e classe	CBR X Ad hoc	Inspeção para diagramas *SMarty*
Bettin *et al.* (2018)					componente	CBR X Ad hoc	
Sabaliauskaite *et al.* (2002)	-	PBR	Não	Sim	classe, atividade, sequência, componente	PBR X CBR	Especificação de cenários para diagramas UML
Laitenberger *et al.* (2000)	-	PBR	Não	Sim	caso de uso, sequência, classe e colaboração	PBR X CBR	

Fonte: o autor

2.5 Considerações Finais

Neste capítulo foram abordados os principais conceitos para o entendimento da família de técnicas proposta por este trabalho, a *SMartyPerspective* (Capítulo 4), para inspeção de artefatos de LPS. Portanto, foram revisados conceitos específicos de LPS, como suas atividades principais, papéis e seu gerenciamento.

Foi apresentada ainda neste capítulo a abordagem *SMarty* para o gerenciamento de variabilidades, permitindo assim, compreender a atividade de LPS que a *SMartyPerspective* está inserida (Engenharia de Domínio) , os artefatos a serem inspecionados (diagramas *SMarty* e diagrama de features) e os papéis que geraram seus cenários.

A seção de inspeção abordou a importância da atividade de verificação de software, para a qualidade dos artefatos da Engenharia de Domínio que devem ser reutilizados por toda a LPS. Para suporte a inspeção, foram apresentadas diversas técnicas de inspeção que colaboram com a atividade, em especial a PBR a qual foi baseada a *SMartyPerspective*.

Finalmente, são apresentados os trabalhos relacionados a este, que permitiram ter um panorama da área de inspeção de software para LPS. O trabalho de Geraldi e OliveiraJr (2017b) é diretamente relacionado a este, pois, define uma técnica CBR para inspeção de diagramas *SMarty*, que assim, como as técnicas CBR, possui algumas questões relacionados a orientação e sobrecarregamento dos leitores, fatores estes que este trabalho visa solucionar com uma técnica orientada e focada.

O processo de criação da *SMartyPerspective* é explicada junto dos cenários que a compõem no próximo capítulo. Nos capítulos que o seguem, os cenários são avaliados para verificar sua viabilidade em um estudo qualitativo.

3

Caracterização da SMartyPerspective

3.1 Considerações Iniciais

Neste capítulo é caracterizada a técnica *SMartyPerspective*, uma família de técnicas baseadas em PBR que tem como objetivo principal inspecionar diagramas UML *SMarty* (caso de uso, classe, componente e sequência) e diagramas de *features* durante a atividade de Engenharia de Domínio de LPS.

Este capítulo também apresenta a criação dos cenários PBR das técnicas por meio do processo de sete etapas de Laitenberger e Kohler (2001), a exceção da última etapa, apresentada no Capítulo 4.

O processo de Laitenberger e Kohler (2001) é constituído das seguintes etapas:

1. Determine o documento a ser inspecionado (Seção 3.2);
2. Especifique as partes interessadas para o tipo de documento (Seção 3.3);
3. Crie um mapeamento de partes interessadas para cada documento (Seção 3.3);
4. Especifique a introdução de cada cenário (Seção 3.4.1);
5. Especifique a instrução de cada cenário (Seção 3.4.3);
6. Especifique a questão de cada cenário (Seção 3.4.3); e
7. Crie a documentação necessária para as Instruções (Capítulo 4).

A especificação da taxonomia de defeitos (Seção 3.4.2) foi um passo importante para a definição dos cenários da técnica *SMartyPerspective*, pois, orientou, por meio dos objetivos específicos da inspeção dos artefatos e dos atributos de qualidade esperados, a definição das questões. Para tal, foi atualizado o mapeamento de Geraldi e OliveiraJr (2017a) para identificar na literatura as classificações de defeitos de software utilizadas nos últimos anos (maio/2017 a maio/2020) como apresentado no Apêndice A.

Neste capítulo, e no restante do trabalho, será utilizado para se referir a *SMarty-Perspective* tanto "uma família de técnicas", como "técnica" somente. Pois, as técnicas que compreendem a família podem também ser descritas simplesmente como cenários da *SMartyPerspective*.

3.2 Artefatos Inspecionados

Dentre os diversos artefatos que estão contidos em um núcleo de artefatos de uma LPS, os diagramas UML se destacam para a modelagem do sistema permitindo que os *stakeholders* do processo consigam entender as funcionalidades e estrutura dos futuros produtos.

A *SMartyPerspective* além dos diagramas UML *SMarty*, inspeciona também diagramas de *features*, que é essencialmente usado na Engenharia de Domínio para compreender os elementos obrigatórios e variáveis. Na lista que segue são apresentados os principais conceitos de cada um destes diagramas:

- Diagrama de *features*: permite modelar hierarquicamente as características comuns e variáveis de todos os produtos que podem ser derivados de uma LPS, podendo, assim, ser entendido pelos *stakeholders* do processo de desenvolvimento, incluindo o cliente;

- Diagrama de caso de uso: é importante para a descrição das funcionalidades do sistema. Com ele é possível compreender o comportamento do sistema e como as funcionalidades se relacionam entre si e com os atores. Na Engenharia de Domínio, este diagrama deve mostrar as funcionalidades comuns e variáveis do sistema;

- Diagrama de classes: modela estruturalmente as classes dos sistema orientado a objetos e pode ser visto sob três perspectivas:

  - Conceitual: se aproxima mais das regras de negócios e do domínio do problema, ou seja, da visão que o cliente tem das classes e possui apenas os conceitos e atributos principais;

  - Especificação: é modelado de acordo com a perspectiva do projetista do sistema, inclui detalhes da estrutura como interfaces e métodos principais;
  - Implementação: é a versão para o desenvolvedor do sistemas. Neste diagrama são incluídos detalhes de todos os atributos, métodos, seus parâmetros e visibilidade.

- Diagrama de componente: um componente é uma parte independente e reutilizável do sistema, mostrando assim, uma visão da estrutura da arquitetura do sistema e como os componentes se relacionam por meio das interfaces. Para o gerenciamento de variabilidades são definidos os componentes da LPS que podem ser configurados.

- Diagrama de sequência: este diagrama mostra o comportamento do sistema, em como os objetos se relacionam por meio da troca de mensagens de um determinado cenário enfatizando a ordem das mensagens. Como nos demais diagramas, para a Engenharia de Domínio são descritos todos os objetos do cenário e as mensagens, sejam elas obrigatórias, opcionais ou alternativas.

3.3 Partes Interessadas

A segunda etapa do processo de Laitenberger e Kohler (2001) é especificar as partes interessadas para o tipo de documento a ser inspecionado. Portanto, para este trabalho é necessário identificar os papéis geralmente usados durante a Engenharia de LPS.

No Capítulo 2 foram apresentados os papéis estruturados por Van der Linden *et al.* (2007) para a Engenharia Domínio, que mantém o núcleo de artefatos, e para a Engenharia de Aplicação, que desenvolve os produtos específicos.

Para a *SMartyPerspective* serão consideradas apenas as perspectivas sob o ponto de vista da Engenharia de Domínio, já que os artefatos a serem inspecionados são os diagramas *SMarty* que permitem o gerenciamento de variabilidades em diagramas UML.

Os seis papéis da Engenharia de Domínio definidos por Van der Linden *et al.* (2007) são: gerente de produto, arquiteto de domínio, engenheiro de requisitos de domínio, desenvolvedor de domínio, testador de domínio e gerente de ativos de domínio.

Com os papéis e os artefatos definidos, o próximo passo para a construção dos cenários operacionais é atribuir às partes interessadas os artefatos a serem inspecionados.

A Tabela 3.1 apresenta uma distribuição entre diagramas UML *SMarty* contemplados pela *SMartyPerspective* e diagrama de *features* com os papéis de Van der Linden *et al.* (2007) em Engenharia de Domínio.

Tabela 3.1: Artefatos inspecionados X papéis em LPS para Engenharia de Domínio

Diagramas	Gerente de Produto	Engenheiro de Requisitos de Domínio	Arquiteto de Domínio	Desenvolvedor de Domínio	Testador de Domínio	Gerente de Ativos de Domínio
Features	X					X
Caso de uso	X	X				X
Classe		X	X	X	X	X
Componente			X	X	X	X
Sequência		X		X	X	X

É possível notar na Tabela 3.1 que os papéis de Desenvolvedor e Testador, apesar de executarem tarefas diferentes, utilizam os mesmos diagramas e analisam as mesmas características nestes artefatos. Portanto, foi decidido que os dois papéis na *SMartyPerspective* correspondem a uma única perspectiva.

Assim, a *SMartyPerspective* engloba cinco perspectivas da Engenharia de Domínio (Figura 3.1): Gerente de Produto (GRP), Engenheiro de Requisitos de Domínio (ERD), Arquiteto de Domínio (AQD), Desenvolvedor de Domínio (ERD) e Gerente de Ativos de Domínio (GAD).

Figura 3.1: Perspectivas da *SMartyPerspective*

3.4 Construção de Cenários

Nesta seção será apresentado o passo a passo da construção dos cenários que compreendem a *SMartyPerspective*. Para isso é necessário definir a taxonomia de defeitos para que as questões sejam elaboradas a partir das classes de defeitos encontradas para os diagramas inspecionados e da visão respectiva da perspectiva.

3.4.1 Introdução dos Cenários

A introdução é a primeira seção de um cenário e é responsável por contextualizar o inspetor sobre os artefatos que serão inspecionados e seus interesses neles a partir da perspectiva que ele deverá assumir para aquele cenário. Contém, ainda, informações dos requisitos de qualidade mais relevantes que devem ser considerados.

Para a configuração da introdução dos cenários foram utilizadas como base algumas das questões propostas por Laitenberger e Kohler (2001), como por exemplo: "Qual é a principal tarefa de uma pessoa com determinada função?" , "Qual é o principal interesse dessa função?" , "Quando ela fez um bom trabalho?" e "Qual é a medida de sucesso?"

A partir das perguntas pré-definidas, as introduções foram especificadas para cada uma das perspectivas da *SMartyPerspective*, pela adaptação das descrições dos papéis e suas funções feitas por Van der Linden *et al.* (2007), dos processos da Engenharia de Domínio e dos diagramas *SMarty* específicos para cada perspectiva conforme definido na Tabela 3.1.

A introdução dos cenários da *SMartyPerspective* é dividida em duas seções. A primeira (Tabela 3.2) contextualiza o inspetor com uma descrição da função da perspectiva, os diagramas que serão inspecionados e a medida do sucesso da inspeção, que descreve o "porquê" os diagramas devem ser inspecionados. Para a perspectiva do GAD, por exemplo, o objetivo é garantir que os diagramas sejam rastreáveis entre si e entre versões, por isso, tal perspectiva deve detectar as inconsistências a partir da comparação entre os diagramas.

A segunda seção da introdução é comum a todos os cenários da *SMartyPerspective*, pois, informa o procedimento que deve ser realizado ao detectar um defeito: "*Para que os diagramas expressem corretamente os requisitos do usuário sem inconsistências, você deve inspecionar os diagramas definidos para seu papel. Para atingir o seu objetivo, execute os passos descritos a seguir para inspecionar cada um dos diagramas. Ao encontrar um defeito em um dos passos, informe no formulário de identificação de defeitos o diagrama, item do passo (questão), o elemento e o defeito encontrado*".

De forma resumida, para qualquer uma das técnicas da *SMartyPerspective*, o inspetor ao encontrar um defeito, deve informar no Formulário de Identificação de Defeitos (FID) os dados sobre o elemento e qual o defeito encontrado (Figura 3.2):

- Diagrama: o leitor deve assinalar a célula correspondente ao diagrama onde encontrou o defeito: *features* (FT), caso de uso (UC), classe (CL), componente (CP) e sequência (SQ);

Tabela 3.2: Introduções de cenários por perspectiva

Pespectiva	Introdução
Gerente de Produto (GRP)	Essa perspectiva deve planejar as características e o valor de negócio dos produtos atuais e futuros da LPS. Para isso, deve garantir que os casos de uso estejam claros com as funcionalidades com base no domínio definido e que as características que serão comuns e variáveis para a LPS sejam definidas corretamente para serem repassadas para o cliente e para a equipe de desenvolvimento.
Engenheiro de Requisitos de Domínio (ERD)	Essa perspectiva deve analisar e especificar os requisitos variáveis e comuns do portfólio de produtos de modo a facilitar a visão da aplicação que será desenvolvida para diferentes *stakeholders*. Essa visão é dada pelos diagramas de caso de uso, classe e sequência que juntos definem as funcionalidades, troca de mensagens dos sistemas pertencentes à (LPS).
Arquiteto de Domínio (AQD)	Essa perspectiva deve desenvolver e manter a arquitetura de LPS para todos os produtos do portfólio da empresa. Deve garantir que os diagramas de componentes, por meio de componentes e interfaces incorporem as classes de domínio propostas e modeladas nos diagramas de classe, mostrando assim, a estrutura física do sistema como um todo, para que os componentes possam depois serem rearranjados de acordo com as regras de negócio da LPS e os requisitos do usuário.
Desenvolvedor de Domínio (DSD)	Essa perspectiva deve implementar e testar os componentes e interfaces que serão reutilizáveis por todo o portfólio dos produtos. Deve desenvolver de acordo com os requisitos de uma gama de produtos, para isso, deve garantir que os elementos representados nos diagramas *SMarty* classe, sequência e componentes não estejam em discordância com os requisitos, contemplem as funções esperadas e permita-lhe ter uma visão da implementação dos componentes reutilizáveis.
Gerente de Ativos de Domínio (GAD)	Essa perspectiva interage com todas as atividades da Engenharia de Domínio de forma a gerenciar as versões e as variantes dos artefatos produzidos nessa fase. Portanto, deve garantir que as versões dos diagramas *SMarty* de caso de uso, classe, componente, sequência e *features* sejam válidos e rastreáveis.

- Nº questão: as questões, que serão apresentadas na próxima subseção, são enumeradas de acordo com o passo que pertencem. Logo, deverá ser indicado no formulário o número da questão que orientou o leitor a detectar o defeito;

- Elemento: um diagrama é composto de diversos elementos que o caracterizam, portanto, deve ser informado o nome do elemento em que foi encontrado o defeito; e

- Defeito identificado: nesta coluna, o inspetor indicará uma descrição do defeito encontrado para aquele elemento e diagrama. Não é necessário o inspetor identificar a classe de defeito, pois, cada questão já está relacionada a uma classe da Taxonomia da *SMartyPerspective* (Seção 3.4.2). É possível também informar neste campo uma solução para corrigir o defeito para facilitar as próximas fases da inspeção.

Figura 3.2: Formulário de Identificação de Defeitos

nº	DIAGRAMA					Nº QUESTÃO	ELEMENTO	DEFEITO IDENTIFICADO
	FT	UC	CL	CP	SQ			
1								
2								
3								
4								

Fonte: o autor

3.4.2 Tipos de Defeitos

Com as introduções definidas, o próximo passo é configurar as instruções, as quais irão orientar em como deverão ser extraídas e documentadas as informações, e as questões que serão respondidas durante a inspeção para identificar se há defeitos nos modelos *SMarty* inspecionados.

A configuração das questões dos cenários foi dividida em duas partes: a primeira, identificar a taxonomia de defeitos da técnica; e a segunda é criar as perguntas de acordo com as necessidades específicas das perspectivas selecionadas em relação à classificação de defeitos definida.

Para definir a taxonomia de defeitos da *SMartyPerspective* foi estudada a classificação utilizada na *SMartyCheck* (Geraldi e OliveiraJr, 2017b), pois, as técnicas possuem o mesmo objetivo de inspecionar diagramas *SMarty* e utilizar uma mesma taxonomia poderia colaborar com um esquema unificado para inspeção deste tipo de artefato.

Geraldi e OliveiraJr (2017a) realizaram um estudo secundário e identificaram na literatura até maio/2017 os tipos de defeitos no contexto de inspeção de software. Deste

estudo, surgiu a taxonomia de defeitos da *SMartyCheck* adaptada dos estudos de Travassos *et al.* (1999) e IEEE (2012).

Para *SMartyCheck* a taxonomia para diagramas de classe e de caso de uso é: Ambiguidade, Incompleto, Inconsistência, Incorreto, Instável, Imodificável, Fato Incorreto, Informação Estranha, Desvio Intencional, Inviável, Omissão, Regras de Negócio e Anomalia (Geraldi e OliveiraJr, 2017b). E para diagramas de componente: Inconsistência, Imodificável, Fato Incorreto, Informação Estranha, Desvio Intencional, Omissão, e Anomalia (Bettin *et al.*, 2018).

O estudo experimental da *SMartyCheck* para diagramas de componentes ((Bettin *et al.*, 2018) permitiu observar como os participantes interagiram com os diagramas *SMarty* e com o *checklist* da técnica, surgindo durante a realização do experimento, dúvidas dos participantes acerca da classificação correta dos defeitos encontrados, por haverem descrição parecida dos itens do *checklist* e seus tipos de defeitos.

Desta observação e do estudo dos tipos de defeitos da *SMartyCheck* surgiu a necessidade de estudar outras classificações de defeitos existentes na literatura para serem usadas na inspeção dos diagramas *SMarty*, além das definidas no trabalho de Geraldi e OliveiraJr (2017a).

Para atingir este objetivo, a revisão sistemática de Geraldi e OliveiraJr (2017a) foi atualizada com os estudos publicados no período de maio/2017 a maio/2020 (Apêndice A), identificando assim, as taxonomias utilizadas pelos pesquisadores da área de inspeção de software no últimos três anos.

Como apresentado no Apêndice A, as classes de defeitos mais utilizadas para adaptações é a de Shull (1998), que analisou no padrão IEEE os atributos que um documento de requisitos deveria ter. Da falta desses atributos consideraram os defeitos para definir a seguinte taxonomia com base nos requisitos: Omissão, Ambiguidade, Fato Incorreto, Inconsistência e Informação Estranha.

As classes de defeitos encontradas no estudo apresentado no Apêndice A foram analisadas e verificadas quanto a viabilidade de serem utilizadas para a *SMartyPerspective* em relação aos atributos de qualidade esperados dos diagramas previamente definidos. A taxonomia de Shull (1998) foi escolhida para análise mais aprofundada, pois, pode ser utilizada e adaptada em diversos contextos e tipos de artefatos. Além disso, suas classes conseguem englobar diversos tipos de defeitos sem a necessidade de muitas categorias. Assim, o inspetor pode se concentrar nas questões e etapas do processo de inspeção ao invés da classificação que pode se tornar confusa entre mais de um tipo.

Da análise da taxonomia de (Shull, 1998) foi verificado que ela se assemelha a do trabalho de (Basili *et al.*, 1996b) que serviu como base para a taxonomia definida no

trabalho de (Travassos *et al.*, 1999). Esta classificação é utilizada em diversos estudos, como encontrados na atualização (Apêndice A) e nos trabalhos relacionados a este: a técnica *SMartyCheck* (Geraldi *et al.*, 2015), que adaptou alguns dos tipos de defeitos, e a técnica FmCheck (de Mello *et al.*, 2014) que utilizou a taxonomia completa.

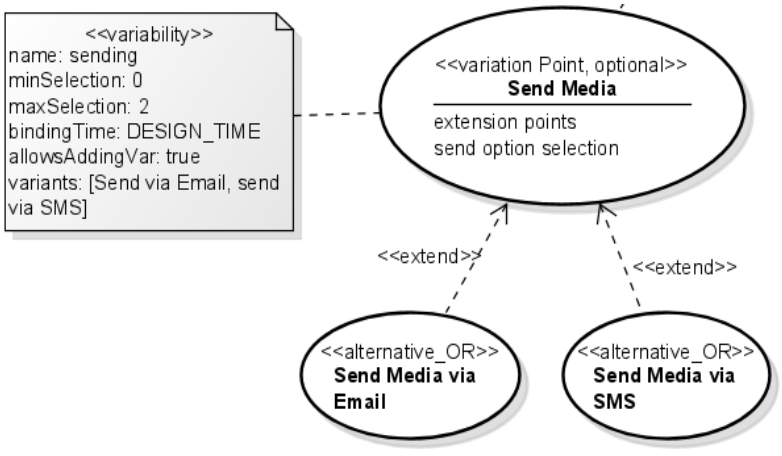
Para a taxonomia do trabalho (Travassos *et al.*, 1999) as especificações das classes de defeitos foram adaptadas de (Basili *et al.*, 1996b) e definidas para modelos orientados a objetos, se aproximando assim, dos objetivos deste trabalho. Da sua especificação e por atender os atributos de qualidade definidos para este trabalho, a taxonomia escolhida para a *SMartyPerspective* foi a de (Travassos *et al.*, 1999).

A taxonomia de defeitos da *SMartyPerspective* compreende os tipos: Ambiguidade, Fato Incorreto, Inconsistência, Informação Estranha e Omissão. Os defeitos serão identificados a partir da comparação com a especificação de requisitos que será considerada aqui, livre de defeitos.

A Figura 3.3 mostra uma parte do diagrama de caso de uso da LPS Mobile Media (MM) que é considerado correto (sem injeção de defeitos) para o domínio da LPS. Nela, está modelado o caso de uso opcional `Send Media`, que é também um ponto de variação e tem as variantes inclusivas `Send Media via Email` e `Send Media via SMS`.

Foram injetados alguns defeitos na parte do diagrama apresentado na Figura 3.3 para colaborar com o entendimento das classificações de defeitos da técnica *SMartyPerspective*, como é mostrado nos tópicos que seguem:

Figura 3.3: Parte do diagrama de caso de uso da LPS MM



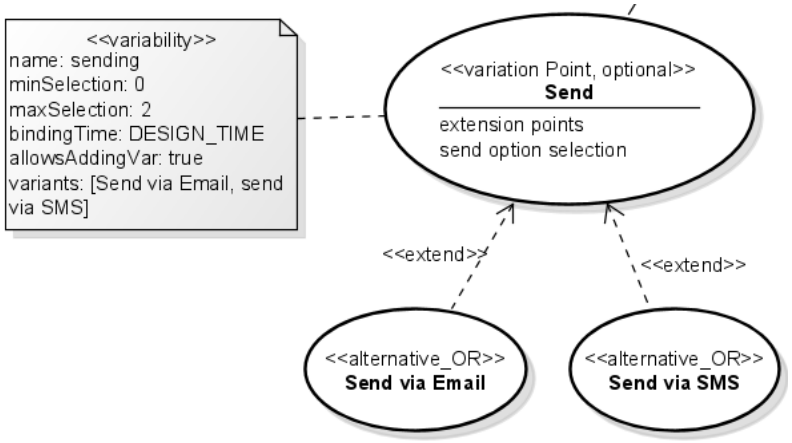
Fonte: adaptado de Geraldi *et al.* (2015)

- Ambiguidade: refere-se à informação que não foi especificada clara o suficiente para seu entendimento e pode gerar múltiplas interpretações de um mesmo elemento

para cada leitor diferente do diagrama, o que fará que cada um realize suas tarefas a partir da sua compreensão sobre aquela informação.

– A Figura 3.4 apresenta um exemplo de ambiguidade para os casos de uso apresentados. Ao ser especificado o caso de uso `Send`, `Send via SMS` e `Send via Email` não foi tomado o cuidado para que refletissem corretamente as funcionalidades que descrevem. O leitor ao analisar este diagrama sem o conhecimento claro do domínio da LPS, não saberia informar o que "*send*"representa. Enviar qualquer tipo de mídia por email e/ou SMS? Só foto? Só músicas? Só vídeos? E assim, poderia planejar, desenvolver, testar erroneamente os componentes desta LPS e mais tarde serem configurados produtos que não condizem com as especificações da LPS.

Figura 3.4: Exemplo Tipo de Defeito *SMartyPerspective* - Ambiguidade



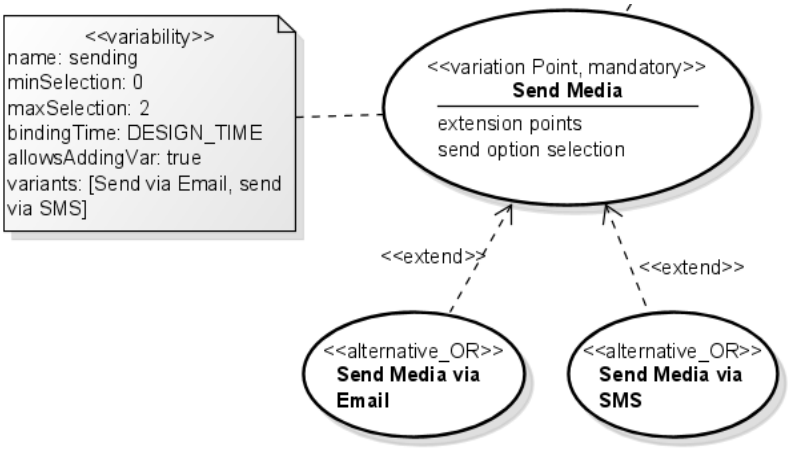
Fonte: adaptado de Geraldi *et al.* (2015)

- Fato Incorreto: a informação descrita em um diagrama *SMarty* está em desacordo com a especificação de requisitos e/ou conhecimento do domínio da LPS, apresentando assim, uma informação errada para o diagrama que não estará modelando corretamente a LPS, e assim, podendo trazer problemas futuros com escolhas erradas de configurações de produto ou dados de entrada/saída do sistema. Pode ser considerado neste tipo de defeito: definição incorreta de atributos, métodos, estereótipos da *SMarty* ou específico do diagrama.

  – `Send Media` foi definido no exemplo da Figura 3.5 como um elemento obrigatório (`«mandatory»`), assim, o envio de mídia, seja por email e/ou SMS, deverá

ser obrigatório para todos as configurações da LPS. Esta declaração é incorreta de acordo com a especificação de requisitos, pois ela define este elemento como sendo opcional, ou seja, não é necessário configurar o envio de mídia no produto gerado da LPS MM.

Figura 3.5: Exemplo Tipo de Defeito *SMartyPerspective* - Fato Incorreto

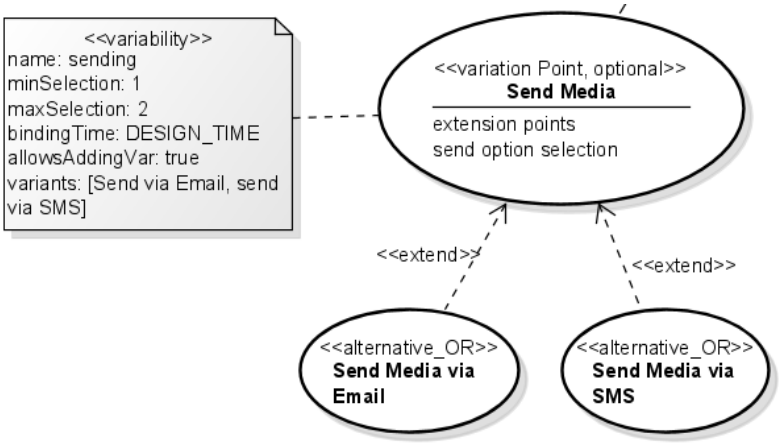


Fonte: adaptado de Geraldi *et al.* (2015)

- Inconsistência: quando as informações de um elemento do diagrama em inspeção não estão consistentes com outro elemento do mesmo diagrama ou outro artefato, ou seja, a informação sobre o mesmo elemento diverge.

  – Na Figura 3.6 é possível observar um exemplo de informações inconsistentes ao analisarmos na notação `«variability»` o meta-atributo `minSelection`=1. Ao ser especificado com 1, isso obriga o cliente a escolher uma das duas opções de variantes definidas para o elemento, mas, `Send Media` trata-se de um elemento opcional e pode não estar incluso em uma configuração desta LPS. Logo, nenhuma das variantes estará na configuração se o elemento também não estiver. `MinSelection` deve então ser especificado com 0 (`minSelection`=0).

Figura 3.6: Exemplo Tipo de Defeito *SMartyPerspective* - Inconsistência

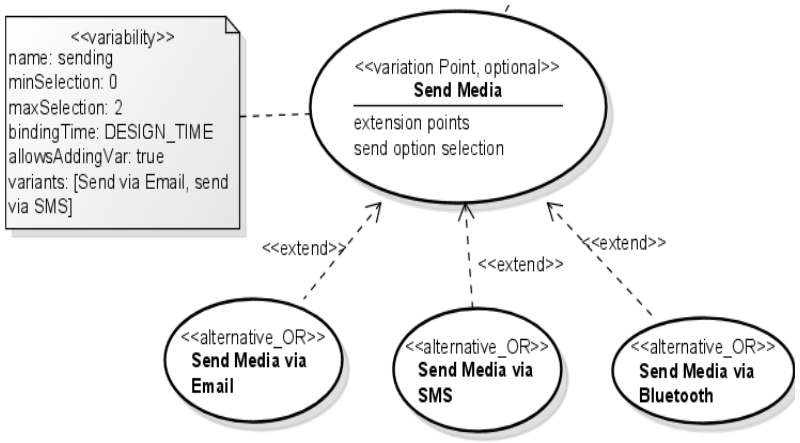


Fonte: adaptado de Geraldi *et al.* (2015)

- Informação Estranha: refere-se a uma informação que foi incluída no diagrama, mas que, não é necessária para o contexto modelado ou não pertence ao domínio da LPS, ou seja, a informação está excedente no artefato.

  – Na Figura 3.7, a informação estranha no diagrama é o caso de uso `Send Media via Bluetooth`, pois, é descrito no documento de requisitos que é possível enviar uma mídia por *email* ou SMS, mas não por *bluetooth*. Logo, este elemento não pertence ao domínio da LPS MM e está "sobrando" no diagrama.

Figura 3.7: Exemplo Tipo de Defeito *SMartyPerspective* - Informação Estranha

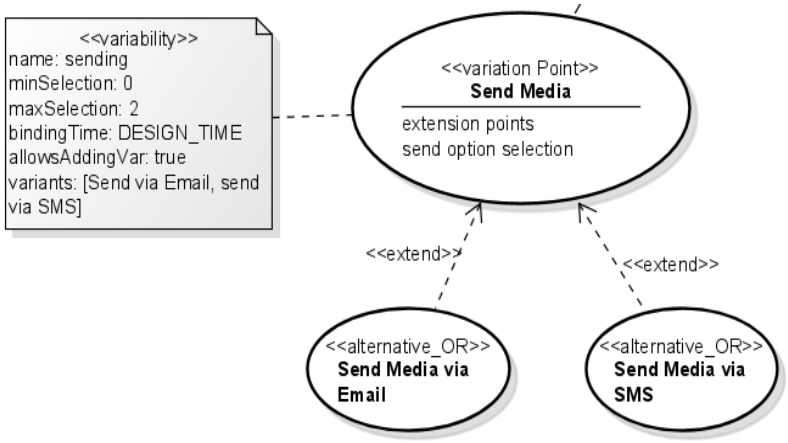


Fonte: adaptado de Geraldi *et al.* (2015)

- Omissão: refere-se a informações necessárias que foram omitidas do artefato em inspeção, seja ele um elemento ou um estereótipo que não foi definido no diagrama. Sendo assim, poderá faltar alguma funcionalidade no sistema ou não será possível compreender corretamente a característica do elemento na LPS.

  – O elemento `SendMedia` foi especificado na Figura 3.8 sem o estereótipo *SMarty* que define se ele é obrigatório ou opcional. Dessa forma, não será possível compreendê-lo corretamente para os próximos processos da Engenharia de Domínio e na Engenharia de Aplicação com a configuração do produto, pois os papéis não irão saber se o envio de mídia é obrigatório ou não se basearem apenas neste diagrama.

Figura 3.8: Exemplo Tipo de Defeito *SMartyPerspective* - Omissão



Fonte: adaptado de Geraldi *et al.* (2015)

3.4.3 Instruções e Questões dos Cenários

A segunda seção de um cenário operacional para PBR é a instrução. Ela é a responsável pela característica sistemática da técnica, pois orienta o leitor em como ele deverá encontrar, ler e extrair as informações dos diagramas *SMarty*.

As instruções da *SMartyPerspective* foram criadas a partir das perguntas úteis de Laitenberger e Kohler (2001), como: "Quais são as partes do documento em que o papel está interessado?", "Como é encontrada a informação?" e "Quais tarefas são realizadas e como?".

Para tornar a orientação detalhada o suficiente para que não hajam dúvidas ao leitor do elemento que deve ser inspecionado por aquela instrução, foram especificados no inicio

das instruções uma breve descrição da característica e do estereótipo *SMarty* do elemento a qual o bloco de questões se refere.

Além das orientações para buscar um elemento, as instruções da técnica *SMarty-Perspective* permitem que o leitor participe ativamente da inspeção, mantendo-o focado no documento como um todo. As instruções sugerem que o inspetor faça listas com as informações retiradas do documento de requisitos candidatas a ser um elemento do diagrama ou que ele faça marcações nos elementos visitados e/ou que não pertencem ao diagrama.

Com as instruções definidas, a última seção de um cenário da PBR são as questões. Nela são apresentadas as perguntas referentes ao elemento direcionado pela instrução para verificar se corresponde aos atributos de qualidade definidos para aquele artefato.

As questões da *SMartyPerspective* foram definidas considerando informações específicas da abordagem *SMarty*, como seus estereótipos, e dos tipos de diagramas UML:

- taxonomia de defeitos de (Shull, 1998);
- diretrizes da *SMarty* (Apêndice B);
- os atributos de qualidade de LPS definidos por Pohl *et al.* (2005): suporte a variabilidade, flexibilidade, evolutividade e capacidade de manutenção; e
- principais defeitos cometidos em diagramas UML de classe, caso de uso, sequência e componente dos trabalhos de Bettin *et al.* (2018); Chren *et al.* (2019); Ma (2017)

Baseados na lei de Miller (1956), Laitenberger e Kohler (2001) mencionam em seu trabalho que a seção de questões não deve conter mais de 7 +/- 2 questões, pois, segundo Miller, é o número de itens que uma pessoa pode gravar em sua memória de curto prazo.

Por causa do número de estereótipos da *SMarty*, diretrizes e estereótipos adicionais inerentes a atividade de gerenciamento de variabilidades adicionado aos defeitos específicos da modelagem dos diagramas UML, foi necessário retornar para a configuração das instruções, para quebrá-las em pequenas instruções, para que o grande número de questões não distraia o leitor durante a inspeção.

De maneira geral, em cada uma das técnicas da *SMartyPerspective* a seção de instruções e questões são fundidas em uma só por causa da divisão da instrução em várias outras, ou seja, são divididas em blocos. Há uma primeira instrução que apresenta o diagrama a ser inspecionado e qual o objetivo dele para o papel tomado pelo leitor com as primeiras orientações para ele iniciar o processo.

A primeira instrução é importante pois, é a partir dela que o leitor vai entender como o artefato deverá ser lido e onde/como ele deverá extrair as informações verdadeiras de acordo com os requisitos da LPS para que o diagrama seja "livre de defeitos" perante a Engenharia de Domínio.

Depois da instrução principal, as próximas instruções foram definidas de acordo com a necessidade para aquele papel e quantidade de elementos do diagrama. Essas instruções são mais simples e procuram de forma clara indagar o leitor sobre o tipo do elemento que ele está analisando no momento. Se o elemento for do tipo descrito na sub instrução, então, o inspetor deverá olhar as questões que seguem aquele bloco.

As sub instruções além de permitirem que o número de questões não seja maior que a Lei de Miller (Miller, 1956), permitem também agrupar as características e defeitos específicos para aquele determinado elemento de forma a evitar esforço desnecessário, ou seja, o leitor só lerá aquele bloco de questões, caso o elemento seja daquele tipo, caso contrário, ele poderá passar para o próximo e assim adiante.

Na *SMartyPerspective*, depois da instrução inicial, pode haver mais dois níveis de instruções. O primeiro nível é definido como "Passo", já os grupos dentro desses passos, são enumerados como as questões dentro de grupo: a numeração inicia com o número do passo em que se encontra e a numeração da ordem da questão de leitura no grupo, por exemplo, o grupo 1.8 e a questão 1.1 localizados no Passo 1.

Na Figura 3.9 é apresentado um exemplo de como houve a quebra das instruções em subgrupos do cenário de ERD para o diagrama de caso de uso. Nela é possível observar que as instruções foram divididas em três blocos: (i) a instrução principal, que orienta inicialmente o inspetor; (ii) sub instruções ou subgrupos nível 1, para busca de elementos específicos do diagrama (Passos 1 e 2); e (iii) subgrupos nível 2 para atributos e/ou relações específicas do elemento do subgrupo nível 1 (Grupos 1.8, 2.1 e 2.2).

Figura 3.9: Exemplo das instruções para o cenário de ERD para diagrama de caso de uso

LOCALIZE O DIAGRAMA DE CASOS DE USO E A ESPECIFICAÇÃO DE REQUISITOS		
O diagrama de casos de uso na Engenharia de Domínio deve descrever corretamente o conjunto de funcionalidades de todos os sistemas que podem ser configurados da LPS. Leia atentamente a especificação de requisitos. Durante a leitura, faça uma lista com todos os requisitos e atores especificados no documento colocando uma notação com o tipo de estereótipo do elemento (mandatory, optional, etc.) e seus possíveis relacionamentos. Compare a lista feita com o digrama de casos de uso para garantir que não há inconsistência entre eles. Para tal, responda as questões que seguem.		
Passo 1	Considere como elementos neste passo os atores e os casos de uso. Para cada elemento no diagrama de casos de uso, verifique se há correspondência com os elementos da lista feita. Faça uma marcação no elemento após sua análise, para evitar que seja analisado novamente.	
	1.8	Para cada elemento do diagrama de casos de uso verifique e analise seus relacionamentos para responder as questões que seguem. Lembre-se de fazer uma marcação nos relacionamentos já verificados para evitar reanálise.
Passo 2	Depois da análise de todos os elementos no passo anterior (todos demarcadas como visitados), verifique e analise as questões que seguem.	
	2.1	Se o elemento for do tipo opcional ou ponto de variação (<<optional>> ou <<variationPoint>>), verifique e analise seus relacionamentos e estereótipos para responder as questões que seguem.
	2.2	O estereótipo <<variability>> representa a variabilidade por meio de um comentário UML. Para cada um desses comentários definidos nos diagramas, vá até o comentário, analise-o e responda as questões abaixo.

A instrução principal define o diagrama de caso de uso e as primeiras orientações, que de forma ativa, sugerem ao leitor fazer uma lista com as funcionalidades do sistema descritas na especificação de requisitos com os candidatos a elementos (caso de uso/ator). Com a lista definida, o leitor passa para a sub instrução em que lhe é recomendado ir em cada elemento do diagrama, marcando os já visitadas, para que não haja repetições e confusões durante a inspeção.

Ao analisar o elemento atual, o leitor deverá, com as sub instruções, verificar as próximas sub instruções, que compreendem as relações entre os elementos, para o Passo 1. Já o passo 2, é analisado após o fim do Passo 1, e contém questões específicas de gerenciamento de variabilidade.

Como dito anteriormente, cada uma das instruções ou sub instruções contêm um conjunto de questões (apresentadas no Capítulo 4) específicas para aquele elemento. Elas ajudam o leitor a decidir se há um defeito daquela tipo no elemento que ele está analisando no momento (elemento atual).

As questões foram enumeradas de acordo com a localização em que se encontra dentro do cenário. Sua numeração inicia com o número do grupo e quando houver, um subgrupo,

em que está localizada seguido do número da ordem de leitura (exemplo, a questão 1.8.1 para o subgrupo 1.8 e questão 1.2 do Passo 1 Figura 3.9).

As questões indagam de maneira simples o leitor a procurar e analisar se o elemento atual está de acordo com os requisitos da LPS e os atributos de qualidade definidos para este trabalho, como por exemplo a questão: "Há uma notação UML que representa variabilidade (`«variabilitty»`) associada à classe de controle?", faz com que o leitor verifique se há uma notação de variabilidade definida para a classe de controle, caso ele não tenha encontrado, então, ele deverá assumir que há um defeito nesta classe.

Cada uma das questões da *SMartyPerspective* foi relacionada à uma das classes de defeitos da taxonomia de Shull (1998), para identificar que tipo de defeito a questão ajuda e encontrar. Além disso, as questões foram relacionadas as diretrizes da *SMarty*. As tabelas com o relacionamento entre questões, tipos de defeitos e diretrizes são apresentadas no Apêndice C para cada um dos diagramas.

Para não confundir o leitor e fazer com que ele se preocupe com detalhes desnecessários de classificação de defeito (e se transforme em uma técnica de Leitura Baseada em Defeitos), a classificação definida para as questões não é apresentada ao leitor durante a inspeção, e sim, na documentação da técnica, para que a organização utilize por exemplo, para fins estatísticos, as classes de defeitos mais propensas a erros.

O guia completo com a numeração das instruções e questões das técnicas da família *SMartyPerspective* pode ser visto no Capítulo 4 que contém a técnica para cada uma das perspectivas.

3.5 Considerações Finais

Neste capítulo a *SMartyPerspective* foi caracterizada a partir da estudo e escolha dos artefatos a serem inspecionados e as partes interessadas com o uso da técnica. Destas informações foram especificados os cenários para cada uma das técnicas da família, com a definição de seus elementos principais: introdução, instrução e questão

As introduções permitem o leitor compreender a perspectiva que irá assumir. Elas foram desenvolvidas a partir da descrição dos papéis de Van der Linden *et al.* (2007) e da sua relação com os diagramas de gerenciamento de variabilidades de caso de uso, classe, componente, sequência e *features*.

As instruções e questões fazem a diferença de uma técnica PBR, pois, elas orientam o leitor "onde" e "como" encontrar as informações nos diagramas inspecionados para detectar os defeitos. Elas foram especificadas por meio da classificação de defeitos de Shull (1998), as diretrizes da abordagem *SMarty* e principais defeitos encontrados nestes diagramas.

No Capítulo 4, são apresentados o resultado do processo executado neste capítulo: os cenários para cada uma das perspectivas com o guia completo de instruções e questões para cada uma delas.

4

Perspectivas da SMartyPerspective

4.1 Considerações Iniciais

No capítulo anterior foi apresentado o processo de especificação dos cenários da *SMarty-Perspective* para inspeção dos diagramas de gerenciamento de variabilidade no processo de Engenharia de Domínio de Linha de Produto de Software (LPS).

Neste capítulo será apresentado como resultado do processo do capítulo anterior, os cenários da *SMartyPerspective* (introdução, instruções e questões) para a inspeção dos diagramas importantes para a realização das tarefas das perspectivas de Gerente de Produto, Engenheiro de Requisitos, Arquiteto de Domínio, Desenvolvedor de Domínio e Gerente de Ativos de Domínio.

Além dos cenários, são apresentados os conceitos principais que caracterizam a *SMarty-Perspective* e serviram de base para sua criação, como também um exemplo de como um pacote do diagrama de classe é inspecionado pelas diferentes visões dos papéis da Engenharia de Domínio.

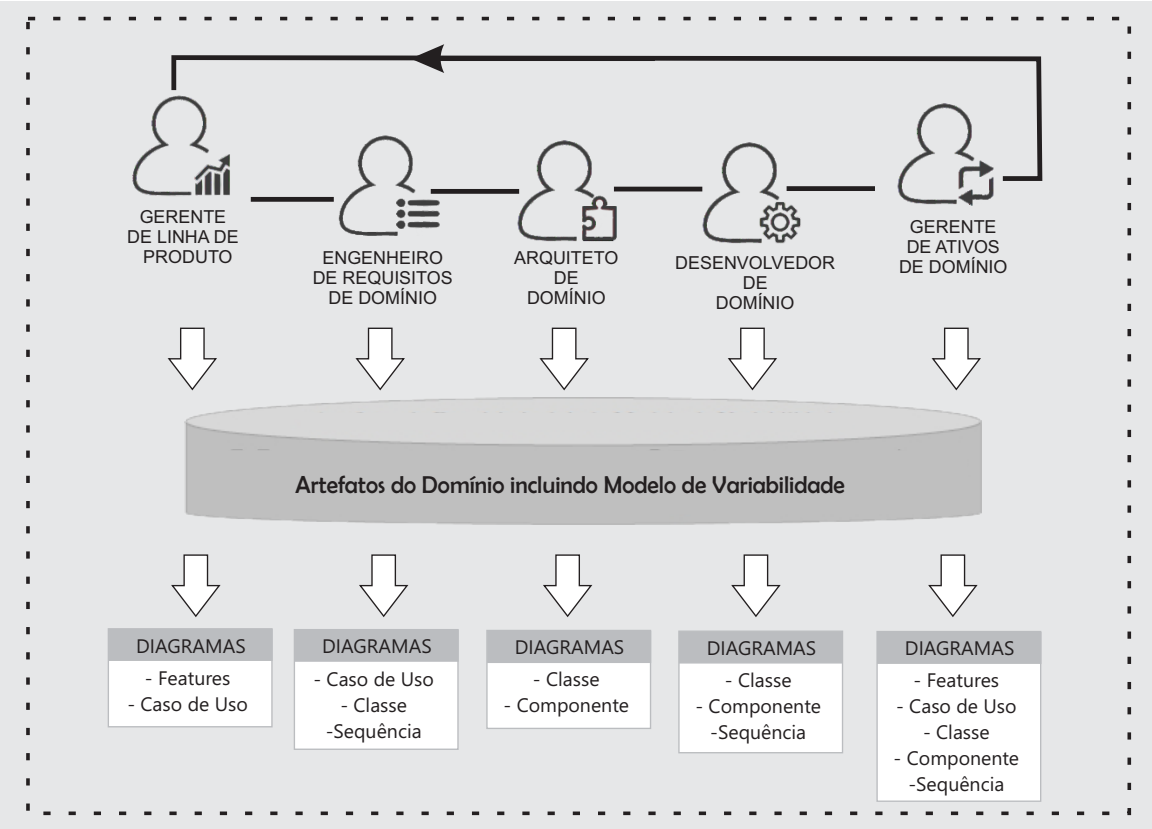
4.2 Caracterização das Perspectivas (Papéis)

A *SMartyPerspective*, é uma família de técnicas de inspeção com base na Leitura Baseada em Perspectiva (PBR) que apoia a inspeção para todos os subprocessos da Engenharia de Domínio e sub processos relacionados que necessitam do gerenciamento de variabilidades, facilitando assim, a comunicação entre toda a equipe de desenvolvimento e diminuindo as chances de que um artefato de entrada de um subprocesso contenha defeitos nos artefatos que já foram inspecionados anteriormente por outro papel.

A *SMartyPerspective* inspeciona os diagramas de *features* e UML *SMarty* de caso de uso, classe, componente e sequência, para garantir, o máximo possível, a qualidade desses artefatos. Estes diagramas foram divididos entre os subprocessos da Engenharia de Requisitos da de acordo com sua importância para suas atividades:

- Gerenciamento de Produto: pensando nos requisitos de negócio, os diagramas são os de *features* e de caso de uso;
- Engenharia de Requisitos de Domínio: os diagramas que modelam a identificação e análise dos requisitos são os de caso de uso, classe (visão conceitual), sequência;
- Projeto de Domínio: o projeto dos componente e interfaces é apoiado pelos diagramas de classe (visão de especificação) e de componente;
- Realização de Domínio: a implementação dos códigos dos componentes do núcleo de artefatos são apoiados pelos diagramas de classe (visão de implementação), componente e sequência; e

Figura 4.1: Engenharia de Domínio e perspectivas da *SMartyPerspective*



Fonte: baseado em Pohl *et al.* (2005)

- Teste de Domínio: os diagramas que apoiam os testes dos componentes são os diagramas de classe (visão de implementação), componente e sequência. Como a visão do testador destes diagramas para o gerenciamento de variabilidades é a mesma da Realização, o cenário usado por ele é o mesmo do Desenvolvedor de Domínio.

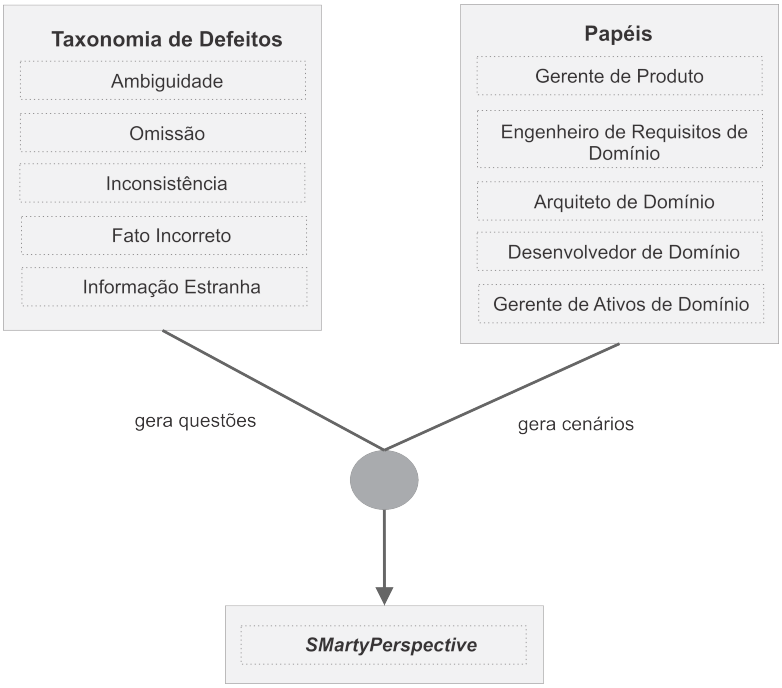
A Figura 4.1 apresenta o processo de Engenharia de Domínio com os papéis definidos para a *SMartyPerspective* e os artefatos deste processo contidos na base de ativos importantes para a realização das tarefas, que para este trabalho, compreendem os diagramas de gerenciamento de variabilidade inspecionadas pela técnica.

Os cenários operacionais da família de técnicas *SMartyPerspective* foram definidos sob duas bases (Figura 4.2):

- as taxonomias de defeitos de (Travassos *et al.*, 1999): Ambiguidade, Fato Incorreto, Inconsistência, Informação Estranha e Omissão; e

- os papéis na Engenharia de Domínio de Van der Linden *et al.* (2007) (Gerente de Produto, Engenheiro de Requisitos de Domínio, Arquiteto de Domínio, Desenvolvedor de Domínio, Gerente de Ativos de Domínio.

Figura 4.2: Geração de cenários da técnica *SMartyPerspective*



Adaptado de Basili *et al.* (1996a)

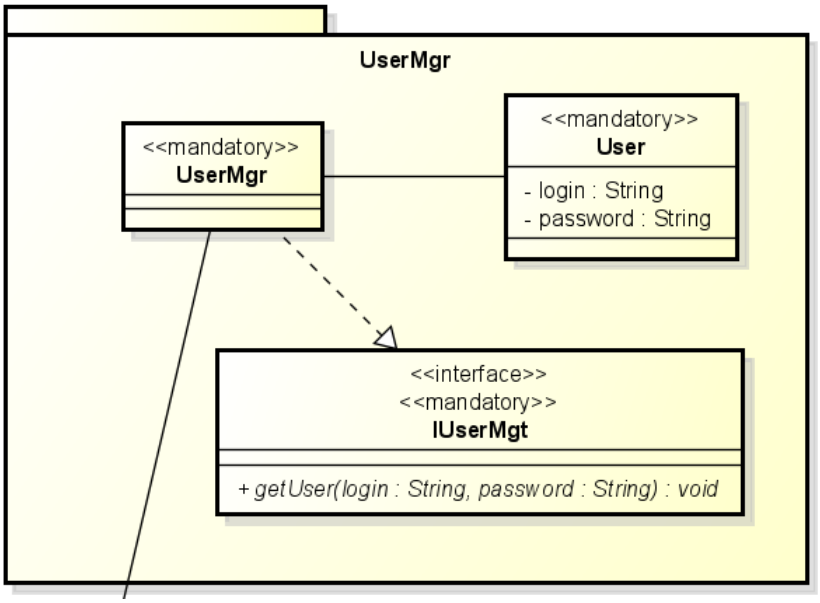
Os papéis serviram de modelo para a geração dos cenários, pois, do estudo de cada um deles foram definidas as perspectivas, introduções e níveis de qualidade esperados dos artefatos para a realização de suas tarefas. As questões para cada um dos cenários operacionais foram definidas e derivadas das classes de defeitos da taxonomia que englobam os principais tipos de defeitos encontrados nos diagramas, contexto e particularidades de LPS.

A grande diferença da PBR em relação às outras técnicas de inspeção é que um mesmo documento/artefato pode ser lido de diferentes maneiras e juntas, as técnicas da família devem englobar todo o documento e as classes de defeitos pertencentes à taxonomia definida evitando a sobreposição de defeitos entre as perspectivas.

Nesta seção será apresentado um exemplo de como alguns tipos de defeitos em um mesmo elemento de um diagrama de classe (pacote `UserMgr`) são percebidos e detectados pelas diferentes perspectivas da *SMartyPerspective*.

O pacote obrigatório `UserMgr` da LPS Mobile Media agrupa as classes referentes ao login do usuário no sistema. Ele é composto das classes `UserMgr` (classe de controle para gerência das atividades), `User` (classe de entidade que contém os atributos) e `IUserMgt` (interface), conforme mostra a Figura 4.3

Figura 4.3: Pacote `UserMgr` da Mobile Media



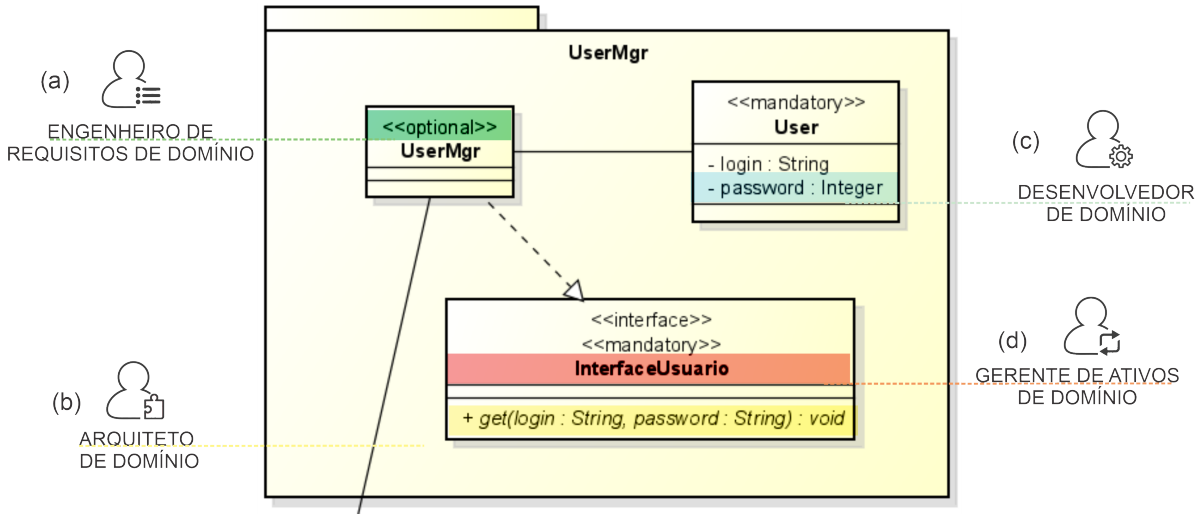
Fonte: o autor

Na técnica *SMartyPerspective* o diagrama de classe pode ser inspecionado por quatro perspectivas diferentes: (i) o Engenheiro de Requisitos de Domínio (ERD), que lê do ponto

de vista conceitual, destacando o problema em si; (ii) O Arquiteto de Domínio (AQD) que analisa sob a perspectiva do projeto da estrutura técnica, identificando os métodos principais que devem ser desenvolvidos; (iii) o Desenvolvedor de Domínio (DSD) que lê o diagrama de classe como um todo, destacando a solução do problema, com os métodos, atributos e seus tipos; e (iv) o Gerente de Ativos de Domínio (GAD) que lê buscando por defeitos de inconsistência entre as versões dos diagramas e da rastreabilidade entre os tipos de diagramas.

Na Figura 4.4 é apresentado o pacote obrigatório `UserMgr` com quatro defeitos incorporados: estereótipo e atributo *SMarty* incorreto de acordo com a especificação de requisitos e divergência entre a nomenclatura da classe entre diferentes versões.

Figura 4.4: Defeitos incorporados no pacote `UserMgr` da Mobile Media



Fonte: o autor

No retângulo verde da Figura 4.4 é possível observar que o estereótipo da classe `UserMgr` foi escrito como sendo opcional, neste caso, seriam possíveis aplicações sem o login do usuário, passando assim, direito para a interface do menu dos jogos sem haver uma autenticação de quem está jogando.

Porém, de acordo com a especificação de requisitos (E), o login do usuário deve ser obrigatório para qualquer aplicação gerada da LPS Mobile Media. Apesar de ser um defeito que pode ser identificado pelos papéis de DSD e AQD, ele já pode ser detectado no ponto de vista conceitual das classes do ERD logo no início do processo de desenvolvimento. Este é um defeito de "Fato Incorreto", já que o estereótipo não está de acordo com a especificação de requisitos.

No retângulo amarelo da Figura 4.4 é possível observar que o método foi definido apenas como "*get*", não sendo possível identificar qual sua função e objetivo. Este tipo de defeito de Ambiguidade pode já ser detectado pelo AQD, pois, ele deve garantir que os métodos principais da estrutura técnica do projeto do portfólio estejam presentes no diagrama de classe. Já detalhes dos parâmetros de entrada e saída e seus tipos, são características da visão de implementação do DSD, ficando a seu cargo analisar esse nível de detalhamento.

No retângulo azul da Figura 4.4, o atributo `password` foi tipificado como **Integer**, mas, na especificação de requisitos foi definido que as senhas podem conter qualquer caracter, Logo, será implementado pelo desenvolvedor de maneira incorreta, permitindo que os usuários insiram apenas números em sua senha, ao invés de qualquer caractere, fazendo assim, com que a senha fique mais fraca pela quantidade limitada que os números oferecem.

Atributos incorretos podem ser identificados já pela perspectiva de ERD, porém, detalhes de seus tipos, são detectados pelo DSD que está mais próximo da implementação e se preocupa com detalhes mais técnicos da estrutura do que o ERD que deve fazer o levantamento de requisitos do usuário.

A interface *IUserMgt* Figura 4.4 foi nomeada como *InterfaceUsuario*, o que estaria correto nas visões das perspectivas de ERD, AQD e DSD, pois, a nomenclatura reflete a classe de login do usuário, porém, se considerarmos que a Figura 4.3 é a primeira versão da LPS Mobile Media e a Figura 4.4 a segunda, os diagramas não seguem o mesmo padrão definido no resto do documento e não são consistentes entre si pela "diferença" entre os nomes do mesmo elemento.

Este tipo de defeito é detectado na *SMartyPerspective* pela perspectiva de GAD, pois, ela é responsável pela rastreabilidade entre os diagramas *SMarty*, garantindo que não haja inconsistência entre eles e seja possível rastrear um elemento de um para o outro: entre versões do mesmo diagrama ou entre tipos diferentes pelo meta-atributo `realizes+` e `realizes-` da notação de variabilidade.

As próximas subseções evidenciam as características de cada uma das cinco técnicas (cenários) da *SMartyPerspective* com uma descrição da importância de cada um dos diagramas para as perspectivas.

4.3 Gerente de Produto (GRP)

O Gerente de Produto tem uma visão de negócio como um todo, pois sua função está relacionada com o planejamento dos produtos presentes e futuros da LPS, do seu valor

de negócio e da estratégia de marketing, atentando-se para as características comuns e variáveis (Van der Linden *et al.*, 2007; Pohl *et al.*, 2005).

Os diagramas abordados pela *SMartyPerspective* que dão uma visão econômica e do escopo da LPS abordada são os diagramas de *features*, que apresenta as características dos produtos e o diagrama UML *SMarty* de caso de uso com as funcionalidades de todos os produtos que podem ser gerados.

O cenário de GRP é composto de quatro passos, como definido na Figura 4.5.

Figura 4.5: Cenário para a perspectiva de GRP

GERENTE DE PRODUTO
Essa perspectiva deve planejar as características e o valor de negócio dos produtos atuais e futuros da LPS. Para isso, deve garantir que os casos de uso estejam claros com as funcionalidades com base no domínio definido e que as características que serão comuns e variáveis para a LPS sejam definidas corretamente para serem repassadas para o cliente e para a equipe de desenvolvimento.
Para que os diagramas expressem corretamente os requisitos dos usuários sem inconsistências, você deve revisar os diagramas que colaboram com a sua função. Para atingir seu objetivo, execute os passos descritos abaixo para inspecionar cada um desses diagramas. Ao encontrar um defeito em um dos passos, preencha no Formulário de Identificação de Defeitos o diagrama, item do passo (questão), o elemento e o defeito que foi encontrado.

LOCALIZE O DIAGRAMA DE *FEATURES* A ESPECIFICAÇÃO DE REQUISITOS	
Passo 1	Inspeção diagrama de features
Passo 2	
LOCALIZE O DIAGRAMA DE CASO DE USO E A ESPECIFICAÇÃO DE REQUISITOS	
Passo 3	Inspeção diagrama de caso de uso
Passo 4	

Fonte: o autor

4.3.1 GRP - Diagrama de *Features*

O diagrama de *features* é um diagrama importante para o papel de Gerente de Produto, pois é a partir dele que ele poderá gerenciar todas as características do portfólio da LPS e montar a estratégia de *marketing*, os custos e o que será desenvolvido e reutilizado pelo restante da equipe.

Na *SMartyPerspective* o diagrama de *features* é inspecionado em dois passos principais (Figura 4.6). O primeiro (Passo 1) inicia com a análise do nó raiz da árvore que define as características, pois, para o diagrama estar correto, o nó raiz deve apresentar o domínio da LPS.

Após a pergunta inicial, o leitor deverá ler cada uma das características presentes no diagrama, marcando os elementos que já tiverem sido analisados procurando, primeiramente, por ambiguidade entre as características e se o nome a representa corretamente.

Figura 4.6: Inspeção de diagramas de *features* para GRP

LOCALIZE O DIAGRAMA DE *FEATURES* A ESPECIFICAÇÃO DE REQUISITOS			
O diagrama de *features* descreve de forma hierárquica as características de todos os produtos que podem ser gerados de uma LPS, identificando as características comuns e variáveis dos produtos atuais e futuros. Leia atentamente a especificação de requisitos. Durante a leitura, faça uma lista com todas as características especificadas no documento colocando uma notação com o tipo de estereótipo do elemento (mandatory, optional, etc.) e seus possíveis relacionamentos. Compare a lista feita com o diagrama de *features* para garantir que não haja inconsistência entre eles. Para tal, responda as questões que seguem.			
Passo 1	1.1	O nó raiz representa corretamente o Domínio?	
	Para cada característica descrita no diagrama de features, analise-a e verifique suas relações para responder as questões que seguem. Faça uma marcação no elemento após sua análise, para evitar que seja analisado novamente.		
	1.2	O nome da feature expressa corretamente a característica que ela representa?	
	1.3	A *feature* representa uma característica que não foi definida no documento de requisitos? Se sim, desconsidere-a com suas relações para os próximos passos e vá para o próximo elemento.	
	1.4	A característica descrita por essa *feature* já foi especificada por outro elemento? Se sim, desconsidere-a com suas relações para os próximos passos e vá para o próximo elemento.	
	1.5	Verifique a aresta que chega nessa *feature* e responda as questões que seguem:	
		1.5.1	Há realmente uma relação entre essa característica e o nó anterior?
		1.5.2	A característica obrigatória está definida com um círculo preenchido?
		1.5.3	A característica opcional está definida com um círculo vazado?
		1.5.4	Se houver descrita a cardinalidade, ela está correta de acordo com a especificação de requisitos?
	1.6	Se a aresta pertencer a um arco vazado ou preenchido, analise e responda as questões que seguem:	
		1.6.1	Esta *feature* realmente é uma característica alternativa da feature do nó anterior?
		1.6.2	Características alternativas estão dentro de arcos vazados ou com relação XOR?
		1.6.3	Características inclusivas estão dentro de arcos preenchidos ou com relação OR?
	1.7	Se houver uma linha ligando esta característica a outra e ela não tiver sido analisada ainda. Verifique e responda as questõe que seguem:	
		1.7.1	Há realmente um relacionamento de inclusão/exclusão entre essas features?
		1.7.2	Se a *feature* requer outra, a linha está definida como uma seta direcionada?
		1.7.3	Se a *feature* exclui outra, a linha está definida como uma seta bidirecional?
	1.8	Faltou para esta feature alguma relação de inclusão/exclusão com outra?	
Passo 2	Quando todas as *features* já estiverem analisadas (todas demarcadas como visitadas), verifique se não faltou nenhuma *feature* importante para o domínio da LPS em inspeção. Analise as questões que seguem:		
	2.1	Há alguma característica opcional ou obrigatória para a LPS que não foi descrita no diagrama?	
	2.2	Verifique os arcos vazados ou preenchidos. Faltou uma característica alternativa para a *feature*?	

Fonte: o autor

Em seguida, as questões são subdivididas em passos:

- são analisadas as arestas que chegam nesta característica, procurando assim por defeitos relacionados às marcações do tipo que a representa, seja ela obrigatória ou opcional;

- se o arco for marcado com um círculo preenchido ou vazado ele é então definido como alternativo. Neste sub passo é analisado se as características ali definidas são mesmo variantes do nó anterior; e
- o último sub passo verifica os relacionamentos de exclusão, requerimento e inclusão da característica com as demais.

O Passo 2 inicia quando todas as características forem descritas. Portanto, é analisada se alguma ficou de fora do diagrama ou se alguma variante não foi definida nas características alternativas.

4.3.2 GRP - Diagrama de Caso de Uso

O diagrama UML *SMarty* de caso de uso apoia o GRP com a modelagem das funcionalidades, atuando como uma extensão do diagrama de *features*. Pois, no diagrama de *features* são apresentadas as características da LPS, enquanto que no diagrama de caso de uso, elas são apresentadas como funcionalidades do sistema, possibilitando que o GRP tenha uma visão melhor do negócio e possa, assim, criar as estratégias da LPS.

Como a visão do GRP deve ser mais em relação ao negócio, não são inspecionados por este papel os meta-atributos do estereótipo `«variability»`, tendo assim, apenas a visão do estereótipo do elemento e não dados como mínimo e máximo de escolhas.

O diagrama de caso de uso para este papel é inspecionado em 2 passos (Figura 4.7): no primeiro há uma análise individual de cada um dos elementos do diagrama: caso de uso, atores e como eles se relacionam. Se foram definidas corretamente, se apresentam corretamente a funcionalidade do elemento e se as relações realmente existem conforme especificado.

O segundo se inicia com a finalização da inspeção individual de todos os elementos do diagrama. Os estereótipos das variantes são analisadas buscando algum elemento que não foi modelado no diagrama.

Figura 4.7: Inspeção de diagramas de caso de uso para GRP

LOCALIZE O DIAGRAMA DE CASO DE USO E A ESPECIFICAÇÃO DE REQUISITOS			
O diagrama de caso de uso na Engenharia de Domínio deve descrever corretamente o conjunto de funcionalidades de todos os produtos que podem ser configurados da LPS. Leia atentamente a especificação de requisitos. Durante a leitura, faça uma lista com todos os requisitos e atores especificados no documento colocando uma notação com o tipo de estereótipo do elemento (mandatory, optional, etc.) e seus possíveis relacionamentos. Compare a lista feita com o diagrama de caso de uso para garantir que não haja inconsistência entre eles. Para tal, responda as questões que seguem.			
Passo 3	Considere como elementos neste passo os atores e os casos de uso. Para cada elemento no diagrama de caso de uso, verifique se há correspondência com os elementos da lista feita. Faça uma marcação no elemento após sua análise, para evitar que seja analisado novamente.		
	3.1	O nome do elemento expressa corretamente a funcionalidade?	
	3.2	Este elemento corresponde a uma funcionalidade/ator que não foi definida no documento de requisitos? Se sim, desconsidere ele e suas relações para os próximos passos e vá para o próximo elemento.	
	3.3	A funcionalidade descrita por esse elemento já foi especificada por outro elemento? Se sim, desconsidere ele e suas relações para os próximos passos e vá para o próximo elemento.	
	3.4	O elemento no diagrama de caso de uso está estereotipado?	
	3.5	No diagrama de caso de uso, se o elemento for opcional ou obrigatório, ele foi especificado com o estereótipo correto (<<optional>> ou <<mandatory>>)?	
	3.6	caso de uso que são pontos de variação foram demarcados com o estereótipo <<variationPoint>>?	
	3.7	Para cada elemento do diagrama verifique e analise seus relacionamentos para responder as questões que seguem. Lembre-se de fazer uma marcação nos relacionamentos já verificados para evitar reanálise.	
		3.7.1	O relacionamento está de acordo com a especificação de requisitos? Verifique se há realmente relação entre os elementos para o contexto.
		3.7.2	O relacionamento foi identificado como extensão (<<extend>>) ou inclusão (<<include>>) erroneamente de acordo com a especificação de requisitos?
		3.7.3	Os relacionamentos de inclusão entre os elementos foram especificados com o estereótipo <<include>>? Obs.: A *SMarty* sugere que os relacionamentos de inclusão estão associados a variantes obrigatórias (<<mandatory>>) ou opcionais (<<optional>>).
		3.7.4	Se o elemento requer outro, a relação entre eles foi estereotipada com <<requires>> ?
		3.7.5	Se a relação for exclusão mútua, a relação está estereotipada no diagrama com <<mutex>> ?
	3.8	Faltou algum relacionamento para este elemento que não foi especificado no diagrama de caso de uso?	
Passo 4	Quando já tiver visitado todos os elementos do diagrama de caso de uso (todos estiverem demarcados como visitados), verifique e analise as questões que seguem:		
	4.1	Para cada elemento do tipo opcional ou ponto de variação (<<optional>> ou <<variationPoint>>), verifique e analise seus relacionamentos e estereótipos para responder as questões que seguem.	
		4.1.1	As variantes especificadas para esta variabilidade estão com a notação de variante correta (<<OR>>, <<XOR>> ou <<optional>>) ?
		4.1.2	As variantes do tipo <<OR>> e <<XOR>> relacionadas ao ponto de variação tem a relação <<extend>> para o ponto de variação associado?
	4.2	Ainda há itens em sua lista que não foram especificados por nenhum elemento? Ou seja, está faltando no diagrama de caso de uso. (Se forem variantes que já foram identificadas no passo anterior, desconsidere)	

Fonte: o autor

4.4 Engenheiro de Requisitos de Domínio (ERD)

O engenheiro de requisitos deve ter um entendimento do domínio do negócio. Quando ele analisa os diagramas na documentação de requisitos, ele deve garantir que as necessidades de todos os futuros usuários estejam especificadas corretamente por meio dos requisitos comuns e variáveis para entendimento das próximas fases de desenvolvimento.

Os diagramas que dão suporte para o ERD documentar os requisitos comuns e variáveis são: caso de uso, classe (visão conceitual) e sequência, que juntos, permitem obter uma visão do domínio do negócio com a definição das funcionalidades, classes principais e o comportamento do sistema por meio da troca de mensagens.

O cenário de um ERD na *SMartyPerspective* é composto de oito passos (Figura 4.8) os quais são inspecionados na ordem (caso todos estejam modelados) os diagramas de caso de uso, classe e sequência, pois assim, um diagrama mais abstrato pode ajudar a detectar defeitos em um diagrama menos abstrato.

Figura 4.8: Cenário para a perspectiva de ERD

ENGENHEIRO DE REQUISITOS DE DOMÍNIO
Essa perspectiva deve analisar e especificar os requisitos variáveis e comuns do portfólio de produtos de modo a facilitar a visão da aplicação que será desenvolvida para diferentes stakeholders. Essa visão é dada pelos diagramas de caso de uso, classe e sequência que juntos definem as funcionalidades, troca de mensagens e atividades dos sistemas.
Para que os diagramas expressem corretamente os requisitos dos usuários sem inconsistências, você deve revisar os diagramas que colaboram com a sua função. Para atingir seu objetivo, execute os passos descritos abaixo para inspecionar cada um desses diagramas. Ao encontrar um defeito em um dos passos, preencha no Formulário de Identificação de Defeitos o diagrama, item do passo (questão), o elemento e o defeito que foi encontrado.

LOCALIZE O DIAGRAMA DE CASO DE USO E A ESPECIFICAÇÃO DE REQUISITOS	
Passo 1	Inspeção diagrama de caso de uso
Passo 2	
LOCALIZE O DIAGRAMA DE CLASSE E A ESPECIFICAÇÃO DOS REQUISITOS	
Passo 3	Inspeção diagrama de classe
Passo 4	
LOCALIZE O DIAGRAMA DE SEQUÊNCIA E A ESPECIFICAÇÃO DE REQUISITOS	
Passo 5	Inspeção diagrama de sequência
Passo 6	
Passo 7	
Passo 8	

Fonte: o autor

Os grupos de questões para cada um dos diagramas sob a análise do ERD são apresentados fragmentados nas próximas seções para facilitar a compreensão de como foram especificados.

4.4.1 ERD - Diagrama de Caso de uso

O diagrama de caso de uso é um diagrama importante para o ERD, pois, com ele consegue descrever de forma clara todos as funções que os produtos da LPS poderão ter. De maneira geral, a inspeção deste diagrama sob a visão do ERD verifica os caso de uso, atores e como eles estão relacionados.

Pela característica da *SMartyPerspective* de leitura ativa, a técnica orienta que o ERD analise a especificação de requisitos da LPS para identificar as funcionalidades requeridas pelo usuário e fazer uma lista com todas elas, incluindo seu estereótipo. Esta lista será usada para colaborar com o inspetor a identificar mais rápido se as funções estão presentes no diagrama de caso de uso, evitando ler repetidamente o documento.

A inspeção do diagrama de caso de uso é realizada em dois passos (Figura 4.9 e Figura 4.10): no primeiro deles, o inspetor deve visitar cada um dos atores e casos de uso do diagrama, identificando na lista feita anteriormente, se estão de acordo com a especificação de requisitos.

Na primeira parte deste passo, é verificado se o nome do elemento está refletindo de fato à função que ele representa, se seus estereótipos *SMarty* estão corretos e se não há ambiguidade entre os elementos.

Neste passo, há um subgrupo de questões que analisam as relações do elemento atual com os outros do diagrama de caso de uso, para tal, são verificados se há realmente uma relação entre eles, se há alguma relação não definida e se os estereótipos (`include, requires, extend e mutex`) foram especificados corretamente.

Após a analise de todos os elementos do diagrama, a primeira parte do segundo passo é exclusivo para elementos que representam um ponto de variação ou são opcionais. Por causa das especificações da *SMarty*, estes elementos devem conter uma notação de variabilidade associada.

O Passo 2 é composto de três grupos de questões:

- o primeiro sub passo verifica se as variantes associadas ao elemento estão com seu estereótipo correto e se o comentário de UML para notação de variabilidade foi definido;

Figura 4.9: Inspeção de diagramas de caso de uso para ERD - Parte 1

LOCALIZE O DIAGRAMA DE CASO DE USO E A ESPECIFICAÇÃO DE REQUISITOS			
O diagrama de caso de uso na Engenharia de Domínio deve descrever corretamente o conjunto de funcionalidades de todos os sistemas que podem ser configurados da LPS. Leia atentamente a especificação de requisitos. Durante a leitura, faça uma lista com todos os requisitos e atores especificados no documento colocando uma notação com o tipo de estereótipo do elemento (mandatory, optional, etc.) e seus possíveis relacionamentos. Compare a lista feita com o diagrama de caso de uso para garantir que não há inconsistência entre eles. Para tal, responda as questões que seguem.			
Passo 1	Considere como elementos neste passo os atores e os casos de uso. Para cada elemento no diagrama de caso de uso, verifique se há correspondência com os elementos da lista feita. Faça uma marcação no elemento após sua análise, para evitar que seja analisado novamente.		
	1.1	O nome do elemento expressa corretamente a funcionalidade?	
	1.2	Este elemento corresponde a uma funcionalidade/ator que não foi definida no documento de requisitos? Se sim, desconsidere ele e suas relações para os próximos passos e vá para o próximo elemento.	
	1.3	A funcionalidade descrita por esse elemento já foi especificada por outro elemento? Se sim, desconsidere o elemento que está incorreto e suas relações para os próximos passos e vá para o próximo elemento.	
	1.4	O elemento no diagrama de caso de uso está estereotipado?	
	1.5	No diagrama de caso de uso, se o elemento for opcional ou obrigatório, ele foi especificado com o estereótipo correto (<<optional>> ou <<mandatory>>)?	
	1.6	O elemento que representa um ponto de variação foi demarcado com o estereótipo <<variationPoint>>?	
	1.7	Para cada elemento do diagrama de caso de uso verifique e analise seus relacionamentos para responder às questões que seguem. Lembre-se de fazer uma marcação nos relacionamentos já verificados para evitar reanálise.	
		1.7.1	O relacionamento está de acordo com a especificação de requisitos? Verifique se há realmente relação entre os elementos para o contexto.
		1.7.2	O relacionamento foi identificado como extensão (<<extend>>) ou inclusão (<<include>>) erroneamente de acordo com a especificação de requisitos?
		1.7.3	Os relacionamentos de inclusão entre os elementos foram especificados com o estereótipo <<include>>? Obs.: A *SMarty* sugere que os relacionamentos de inclusão estão associados a variantes obrigatórias (<<mandatory>>) ou opcionais (<<optional>>).
		1.7.4	Se o elemento requer outro, a relação entre eles foi estereotipada com <<requires>> ?
		1.7.5	Se a relação for exclusão mútua, o relacionamento está estereotipado no diagrama de caso de uso com <<mutex>> ?
	1.8	Faltou algum relacionamento para este elemento que não foi especificado no diagrama de caso de uso?	

Fonte: o autor

- o segundo sub passo analisa os meta-atributos definidos na notação de variabilidade para identificar defeitos de mínimo e máximo de seleção de variantes no conjunto; e
- o ultimo sub passo, contém uma única questão que faz o leitor procurar em sua lista de itens se algum deles não foi modelado por nenhum elemento do diagrama.

Figura 4.10: Inspeção de diagramas de caso de uso para ERD - Parte 2

Passo 2	Depois da análise de todos os elementos no passo anterior (todos demarcados como visitados), verifique e analise as questões que seguem.		
	2.1	Se o elemento for do tipo opcional ou ponto de variação (<<optional>> ou <<variationPoint>>), verifique e analise seus relacionamentos e estereótipos para responder as questões que seguem.	
		2.1.1	Há uma notação de variabilidade (<<variabilitty>>) associada ao elemento?
		2.1.2	As variantes especificadas para esta variabilidade estão com a notação de variante correta (<<OR>>, <<XOR>> ou <<optional>>) ?
		2.1.3	As variantes do tipo <<OR>> e <<XOR>> relacionadas ao ponto de variação tem a relação <<extend>> para o ponto de variação associado?
	2.2	O estereótipo <<variability>> representa a variabilidade por meio de um comentário UML. Para cada um desses comentários definidos nos diagramas, vá até o comentário, analise-o e responda as questões abaixo.	
		2.2.1	As variantes definidas no conjunto *variants* realmente são variantes para este elemento? Se alguma não for, desconsidere-a para as próximas questões.
		2.2.2	Há variantes definidas na coleção de variantes que não está descrita no diagrama de caso de uso?
		2.2.3	Todas as variantes relacionadas a este elemento definidas com (<<OR>>), (<<XOR>>) ou (<<optional>>) estão na coleção de variantes do meta-atributo variants com seu nome correto?
		2.2.4	Verifique o tipo das variantes associadas: • Se forem do tipo <<optional>>, minSelection= 0 e maxSelection= 1? • Se forem do tipo <<OR>>, minSelection=1 e maxSelection=total de variantes?. • Se forem do tipo <<XOR>>, minSelection=maxSelection=1?
	2.3	Ainda há itens em sua lista que não foram especificados por nenhum elemento? Ou seja, está faltando no diagrama de caso de uso (Se forem variantes que já foram identificadas no passo anterior, desconsidere).	

Fonte: o autor

4.4.2 ERD - Diagrama de Classe

O ERD revisa o diagrama de classe sob o ponto de vista do negócio, por isso, são considerados apenas os conceitos das classes, sem considerar os tipos dos atributos e os métodos para implementação.

Assim que o inspetor localiza o diagrama de classe, ele/ela é convidado(a) a ler a especificação de requisitos procurando por candidatos a objetos das classes, para então utilizar essa lista para dar início a leitura do diagrama de classe.

A inspeção é realizada em dois passos: Passo 3 (Figura 4.11) e Passo 4 (Figura 4.12). O primeiro orienta o leitor a analisar uma classe por vez, verificando primeiramente se está de acordo com a especificação de requisitos, seus atributos e estereótipos. Em seguida, são analisadas algumas informações específicas que estão divididas em dois blocos:

- o primeiro bloco verifica, caso haja, se o pacote que a classe está inserida no diagrama está correto; e

- no segundo bloco, são verificados os tipos e quais são os relacionamentos entre as classes. O leitor analisa se existe relação entre as classes de acordo com a especificação de requisitos e, se os estereótipos *SMarty* e a cardinalidade está correta de acordo com a relação.

O segundo passo (Passo 4) é dividido em três blocos semelhante ao Passo 3. Ele inicia apenas quando todas as classes já tiverem sido visitadas no passo anterior, para então investigar se as classes de controle opcionais e/ou que representam um ponto de variação possuem a notação de variabilidade associada, as variantes definidas e alguns dos meta-atributos pertinentes a esse papel definido na notação de variabilidade. Além de conferir no final, se faltou alguma classe no diagrama.

Figura 4.11: Inspeção de diagrama de classe para ERD - Parte 1

LOCALIZE O DIAGRAMA DE DE CLASSE E A ESPECIFICAÇÃO DOS REQUISITOS			
O diagrama de classes para seu papel deve de maneira conceitual, apresentar a estrutura das classes e suas relações para todos os sistemas da LPS no domínio do negócio. Leia atentamente a especificação de requisitos e durante a leitura, faça uma lista com os objetos mencionados e os dados que caracterizam este objeto. Compare a lista feita com o diagrama de classes para garantir que não há inconsistência entre as classes e os requisitos do usuário. Para tal, responda as questões que seguem.			
Passo 3	Para cada classe do diagrama de classes, verifique as classes, os atributos, os relacionamentos entre as classes com a lista feita e responda as questões que seguem. Faça uma marcação na classe após sua análise, para evitar que seja analisada novamente.		
	3.1	O nome da classe expressa corretamente os objetos desta classe?	
	3.2	Esta classe corresponde a um objeto que não foi definido no documento de requisitos? Se sim, desconsidere ele e suas relações para os próximos passos e vá para o próximo elemento.	
	3.3	Esta classe está em redundância com outra já especificada no diagrama de classes? Se sim, desconsidere ele e suas relações para os próximos passos e vá para o próximo elemento.	
	3.4	A classe está estereotipada no diagrama de classes?	
	3.5	Se o elemento for opcional ou obrigatório, ele foi especificado com o estereótipo correto (<<optional>> ou <<mandatory>>)?	
	3.5	Se as classes estiverem agrupadas em pacotes, verifique e responda as questões que seguem.	
		3.5.1	O nome do pacote foi definido e expressa corretamente o agrupamento? Se já tiver colocado no formulário o defeito para este pacote não precisa colocar novamente.
		3.5.2	A classe está dentro do pacote correto?
	3.6	Para cada relacionamento para esta classe, verifique as classes que compõem este relacionamento, para garantir que o relacionamento entre elas esteja de acordo com a especificação dos requisitos. Para tal, responda as questões que seguem e faça uma marcação nos relacionamentos já analisados.	
		3.6.1	Este relacionamento está de acordo com a especificação de requisitos? Verifique se há realmente relação entre os elementos para o contexto.
		3.6.2	A cardinalidade deste relacionamento está correta de acordo com a especificação de requisitos?
		3.6.3	Verifique o tipo do relacionamento para garantir que as classes estejam com seus estereótipos definidos e corretos de acordo com a abordagem *SMarty*: • Generalização: Os classificadores mais gerais, são pontos de variação (<<variationPoint>>) e os mais específicos, variantes? • Realização de interface: As especificações são pontos de variação (<<variationPoint>>) e as implementações são variantes? • Agregação ou Composição: As instâncias tipadas com losangos (preenchidos ou não) são pontos de variação (<<variationPoint>>) e instâncias associadas são variantes?
		3.6.4	As classes que exigem a presença de outra estão relacionadas com o estereótipo <<requires>> ?
		3.6.5	As classes mutuamente exclusivas estão relacionadas com o estereótipo <<mutex>>?
	3.7	Faltou algum relacionamento para esta classe que não foi especificado no diagrama de classes?	

Fonte: o autor

Figura 4.12: Inspeção de diagrama de classe para ERD - Parte 2

Passo 4	Depois da análise de todas as classes no passo anterior (todas demarcadas como visitadas), verifique e analise as questões que seguem.		
	4.1	As classes de controle gerenciam as atividades da classe. Para cada classe deste tipo que esteja estereotipada com <<optional>> e/ou <<variationPoint>>. Verifique-a para garantir que as notações de variabilidade/variantes estejam corretas de acordo com a especificação de requisitos. Para tal, vá até cada uma dessas classes e responda as questões que seguem.	
		4.1.1	Há uma nota da UML que representa variabilidade (<<variabilitty>>) associada à classe de controle?
		4.1.2	Todas as variantes foram definidas e estão com a notação de variante correta (<<OR>>, <<XOR>> ou <<optional>>) ?
	4.2	O estereótipo <<variability>> representa a variabilidade por meio de um comentário UML. Para cada um desses comentários definidos nos diagramas, vá até o comentário, analise-o e responda as questões abaixo.	
		4.2.1	As variantes definidas no conjunto *variants* realmente são variantes para este elemento? Se alguma não for, desconsidere-a para as próximas questões.
		4.2.2	Há variantes definidas na coleção de variantes que não está descrita no diagrama de classes?
		4.2.3	Todas as variantes relacionadas a este elemento definidas com (<<OR>>), (<<XOR>>) ou (<<optional>>) estão na coleção de variantes do meta-atributo variants com seu nome correto?
		4.2.4	Verifique o tipo das variantes associadas: • Se forem do tipo <<optional>>, minSelection= 0 e maxSelection= 1? • Se forem do tipo <<OR>>, minSelection=1 e maxSelection=total de variantes?. • Se forem do tipo <<XOR>>, minSelection=maxSelection=1?
	4.3	Ainda há itens em sua lista que não foram especificados por nenhum elemento? Ou seja, está faltando no diagrama de classe (Se forem variantes que já foram identificadas no passo anterior, desconsidere).	

Fonte: o autor

4.4.3 ERD - Diagrama de Sequência

O diagrama de sequência para a perspectiva de ERD é importante para dar uma visão da lógica do sistema, das trocas e a ordem das mensagens que são esperadas entre as funcionalidades descritas pelos diversos usuários de maneira gráfica, ao invés de descrever passo por passo textualmente.

Para o ERD, os elementos importantes para seu uso neste diagrama são: linha de vida, que deve estar com seu nome correto, e as mensagens definidas e sua ordem. Para este papel, não são considerados se os métodos definidos para as mensagens estão de acordo com o diagrama de classe, pois, ele tem apenas uma visão conceitual do mesmo, sendo assim, se o nome refletir a mensagem corretamente, não será um defeito.

A leitura do diagrama de sequência é dividida em quatro passos (Passos 5-8), como é mostrado na Figura 4.13 e Figura 4.14:

- Passo 5: verificação das linhas de vida, "cabeças de objeto" e mensagens. Neste passo são inspecionados se os elementos definidos para o diagrama estão de acordo com a especificação de requisitos, incluindo a ordem das mensagens e sua nomenclatura;
- Passo 6: os elementos alternativos do diagrama são analisados quanto a seus estereótipos e das mensagens alternativas relacionadas a ele;
- Passo 7: verifica os elementos opcionais em relação aos estereótipos das mensagens e da cabeça de objeto da linha de vida; e
- Passo 8: os elementos alternativos e opcionais do diagrama de sequência devem possuir uma notação de variabilidade associada, portanto, neste passo são analisados os meta-atributos específicos desta notação para verificar se as informações ali contidas estão de acordo com as apresentadas pelo diagrama. Se possuem por exemplo, linhas de vidas associadas para cada uma das variantes definidas corretamente no conjunto de variantes.

Figura 4.13: Inspeção de diagramas de sequência para ERD - Parte 1

LOCALIZE O DIAGRAMA DE SEQUÊNCIA E A ESPECIFICAÇÃO DE REQUISITOS			
O diagrama de sequência na Engenharia de Domínio deve expressar a interação do sistema: a troca das mensagens entre os objetos dos sistemas que podem ser configurados de uma LPS. Com o diagrama de sequências em mãos, verifique cada um dos objetos/atores descritos e busque pela classe correspondente no diagrama de classe (se estiver definido) ou na especificação de requisitos. Para cada um deles, verifique os relacionamentos e as mensagens que são trocadas entre os elementos para garantir que estejam consistentes com o diagrama de classe/especificação de requisitos. Em seguida, responda as questões que seguem.			
Passo 5	Considere como elemento as "cabeças de objeto" e os atores da linha de vida do diagrama de sequência. Para cada um deles verifique em sua linha de vida as mensagens e estereótipos dados a elas para responder as questões que seguem.		
	5.1	O nome do elemento expressa corretamente o objeto/ator?	
	5.2	O elemento está representado em alguma classe do sistema?	
	5.3	O elemento faz parte desta interação de acordo com a Especificação de Requisitos? Se não, desconsidere toda a linha de vida e suas mensagens para os próximos passos e vá para o próximo elemento.	
	5.4	O elemento é redundante com outro elemento definido? Se sim, desconsidere toda a linha de vida e suas mensagens para os próximos passos e vá para o próximo elemento.	
	5.5	Para cada uma das mensagens definidas na linha de vida deste elemento, analise-a e responda as questões que seguem	
		5.5.1	A mensagem está nomeada?
		5.5.2	A interação representada por esta mensagem está descrita na especificação de requisitos? Se não, desconsidere-a e passe para a próxima mensagem.
		5.5.3	O nome da mensagem expressa corretamente a informação que está sendo transmitida?
		5.5.4	A ordem da mensagem está correta de acordo com a especificação de requisitos?
		5.5.5	Verifique as outras mensagens nesta linha de vida. Esta mensagem está redundante com outra? Se sim desconsidere-a e passe para a próxima mensagem.
		5.5.6	Mensagens que não estejam relacionadas diretamente à uma variabilidade e seus elementos, não precisam de estereótipo e são consideradas obrigatórias. A mensagem deste tipo está estereotipada?
	5.6	Todas as mensagens importantes para este elemento foram definidas no diagrama de sequências de acordo com a especificação de requisitos?	
	5.7	Elementos que exigem a presença de outro estão relacionados com o estereótipo <<requires>> ?	
	5.8	Elementos mutuamente exclusivos estão relacionados com o estereótipo <<mutex>>?	
Passo 6	Para cada elemento alternativo no diagrama de sequência como o CombinedFragment com interactionOperator “alt”(alternative) e elemento interactionUse “ref”, verifique seus estereótipos e as mensagens relacionadas a este elemento para responder as questões que seguem.		
	6.1	O elemento está definido com o estereótipo <<variationPoint>> ?	
	6.2	Há uma notação de UML que representa variabilidade (<<variabilitty>>) associada a este elemento/CombinedFragment?	
	6.3	As variantes correspondentes às mensagens estão estereotipadas corretamente? Verifique os estereótipos das mensagens variantes para esta variabilidade. • Para as variantes do elemento interactionUse “ref”: <<OR>> • Para as variantes com interactionOperator "alt": <<XOR>>	

Fonte: o autor

Figura 4.14: Inspeção de diagramas de sequência para ERD - Parte 2

Passo 7	Para cada elemento opcional no diagrama de sequência, como: combinedFragment com interactionOperator "opt"(optional) e troca de mensagens entre dois objetos não obrigatórios ou entre um objeto obrigatório e outro não, verifique seus estereótipos e as mensagens relacionadas a ele para responder as questões que seguem.	
	7.1	Há uma notação de UML que representa variabilidade (<<variabilitty>>) associada a este elemento/CombinedFragment?
	7.2	As variantes correspondentes as mensagens/CombinedFragment estão estereotipadas corretamente com <<optional>>?
	7.3	Para elementos demarcados com <<optional>> as linha de vidas que fazem parte do CombinedFragment também estão estereotipadas com <<optional>>?
Passo 8	O estereótipo <<variability>> representa a variabilidade por meio de um comentário UML. Para cada um desses comentários definidos nos diagramas, vá até o comentário, analise-o e responda as questões abaixo.	
	8.1	As variantes definidas no conjunto variants realmente são variantes para este elemento? Se alguma não for, desconsidere-a para as próximas questões.
	8.2	Todas as variantes do conjunto de variantes tem uma linha de vida associada no diagrama de sequência?
	8.3	Todas as variantes relacionadas a este elemento definidas com (<<OR>>), (<<XOR>>) ou (<<optional>>) estão na coleção de variantes do meta-atributo variants com seu nome correto?
	8.4	Verifique o tipo das variantes associadas: • Se forem do tipo <<optional>>, minSelection= 0 e maxSelection= 1? • Se forem do tipo <<OR>>, minSelection=1 e maxSelection=total de variantes?. • Se forem do tipo <<XOR>>, minSelection=maxSelection=1?

Fonte: o autor

4.5 Arquiteto de Domínio (AQD)

O AQD é responsável por projetar a arquitetura de LPS (PLA) que seja válida para todos os produtos configuráveis, estruturando assim, os componentes e interfaces que servirão de guia técnico para a fase de implementação.

O AQD deve garantir que os projetos da base de ativos atendam a PLA para qualquer aplicação, assim, deve se preocupar com a estrutura das classes em nível de projeto, verificando se os principais elementos de nível técnico estão presentes no diagrama e verificando as configurações e especificações dos componentes e interfaces no diagrama de componentes.

Sendo assim, o cenário de AQD (Figura 4.15) para a *SMartyPerspective* é constituído de quatro passos nos quais são inspecionados os diagramas UML *SMarty* de classe e de componente. O detalhamento do cenário é descrito nas próximas subseções, divididos pelos tipos de diagramas inspecionados.

Figura 4.15: Cenário para a perspectiva de AQD

ARQUITETO DE DOMÍNIO
Essa perspectiva deve desenvolver e manter a arquitetura de LPS para todos os produtos do portfólio da empresa. Deve garantir que os diagramas de componentes, por meio de componentes e interfaces incorporem as classes de domínio propostas e modeladas nos diagramas de classe, mostrando assim, a estrutura física do sistema como um todo, para que os componentes possam depois serem rearranjados de acordo com as regras de negócio da LPS e os requisitos do usuário.
Para que os diagramas expressem corretamente os requisitos dos usuários sem inconsistências, você deve revisar os diagramas que colaboram com a sua função. Para atingir seu objetivo, execute os passos descritos abaixo para inspecionar cada um desses diagramas. Ao encontrar um defeito em um dos passos, preencha no Formulário de Identificação de Defeitos o diagrama, item do passo (questão), o elemento e o defeito que foi encontrado.

LOCALIZE O DIAGRAMA DE DE CLASSE E A ESPECIFICAÇÃO DOS REQUISITOS	
Passo 1	Inspeção diagrama de classe
Passo 2	
LOCALIZE O DIAGRAMA DE COMPONENTE E DE CLASSE	
Passo 3	Inspeção diagrama de componente
Passo 4	

Fonte: o autor

4.5.1 AQD - Diagrama de Classe

Considerando a evolução do processo de desenvolvimento de software, o diagrama de classe já deve ter sido inspecionado pelo ERD durante suas tarefas, porém, sua visão foi apenas conceitual da relação entre as classes e objetos, agora, sob a perspectiva do AQD, a inspeção deve considerar características do projeto preparando para a implementação.

O AQD inspeciona o diagrama de classe sob o ponto de vista do projeto, então, devem ser revisados além da representação da classes e seus atributos, os tipos dos atributos e os principais métodos das classes.

A inspeção deste diagrama para a perspectiva de AQD é realizada em dois passos: o primeiro orienta o leitor a verificar uma classe por vez do diagrama, analisando os compartimentos de nome, atributos e de métodos (que ainda não tinham sido inspecionados pelo ERD). As questões deste passo são divididas em três grupos (Figura 4.16):

- são verificados no primeiro grupo o nome do pacote e se a classe em inspeção está dentro do pacote correto;
- são verificadas neste sub passo as relações entre as classes e sua cardinalidade. Dependendo do tipo de relacionamento, as diretrizes da *SMarty* sugerem alguns estereótipos que são analisados por questões deste grupo para orientar a encontrar defeitos que estejam em desacordo destas diretrizes; e

- o último grupo analisa se os principais métodos das interfaces foram especificados.

Por fim, o Passo 2 (Figura 4.17) verifica em dois grupos as classes de gerenciamento (controle) e as notações UML que representam variabilidade e uma pergunta final para o inspetor analisar se faltou algum elemento:

- para as classes de controle, se forem opcionais ou pontos de variação serão verificadas se as variantes estão com seus estereótipos corretos e se há uma notação de variabilidade associada ao elemento; e
- a variabilidade associada é analisada quanto às variantes, se estão de acordo com a especificação e em número mínimo e máximo corretos.

Figura 4.16: Inspeção de diagrama de classe para AQD - Parte 1

LOCALIZE O DIAGRAMA DE DE CLASSE E A ESPECIFICAÇÃO DOS REQUISITOS			
O diagrama de classe para seu papel deve apresentar a estrutura das classes e suas relações para todos os sistemas da LPS no domínio do projeto, atentando-se para as interfaces de aplicação e seus métodos principais. Para garantir que este diagrama descreve as classes de projeto da maneira correta, leia atentamente a especificação de requisitos e durante a leitura, faça uma lista com os objetos mencionados, os dados que caracterizam este objeto e os possíveis métodos. Compare a lista feita com o diagrama de classes para garantir que não haja inconsistência entre as classes e os requisitos do usuário. Para tal, responda as questões que seguem.			
Passo 1	Para cada classe do diagrama de classes, verifique as classes, os atributos, os relacionamentos entre as classes com a lista feita e responda as questões que seguem. Faça uma marcação na classe após sua análise, para evitar que seja analisada novamente.		
	1.1	O nome da classe expressa corretamente os objetos desta classe?	
	1.2	Esta classe corresponde a um objeto que não foi definido no documento de requisitos? Se sim, desconsidere ela e seus relacionamentos para os próximos passos e vá para a próxima classe.	
	1.3	Esta classe está em redundância com outra já especificada no diagrama de classes? Se sim, desconsidere ela e seus relacionamentos para os próximos passos e vá para a próxima classe.	
	1.4	A classe está estereotipada no diagrama de classes?	
	1.5	Se as classes estiverem agrupadas em pacotes, verifique e responda as questões que seguem.	
		1.5.1	O nome do pacote foi definido e expressa corretamente o agrupamento? Se já tiver colocado no formulário o defeito para este pacote não precisa colocar novamente.
		1.5.2	A classe está dentro do pacote correto?
	1.6	Para cada relacionamento para esta classe, verifique as classes que compõem este relacionamento, para garantir que o relacionamento entre elas esteja de acordo com a especificação dos requisitos. Para tal, responda as questões que seguem e faça uma marcação nos relacionamentos já analisados.	
		1.6.1	Este relacionamento está de acordo com a especificação de requisitos? Verifique se há realmente relação entre os elementos para o contexto.
		1.6.2	A cardinalidade deste relacionamento está correta de acordo com a especificação de requisitos?
		1.6.3	Verifique o tipo do relacionamento para garantir que as classes estejam com seus estereótipos definidos e corretos de acordo com a abordagem SMarty: • Generalização: Os classificadores mais gerais, são pontos de variação (<<variationPoint>>) e os mais específicos, variantes? • Realização de interface: As especificações são pontos de variação (<<variationPoint>>) e as implementações são variantes? • Agregação ou Composição: As instâncias tipadas com losangos (preenchidos ou não) são pontos de variação (<<variationPoint>>) e instâncias associadas são variantes? • Associação: Verifique o atributo AgregationKind • se for none: as variantes sugerem ser obrigatórias (<<mandatory>>) ou opcionais (<<optional>>). Verifique de acordo com a especificação de requisitos. O estereótipo está correto? • se o valor * ou 0..n são do tipo opcionais (<<optional>>)?
		1.6.4	As classes que exigem a presença de outra estão relacionadas com o estereótipo <<requires>> ?
		1.6.5	As classes mutuamente exclusivas estão relacionadas com o estereótipo <<mutex>>?
	1.7	Faltou algum relacionamento para esta classe que não foi especificado no diagrama de classes?	
	1.8	Interfaces especificam os métodos externamente visíveis para outras. Se a interface for deste tipo, analise cada um dos métodos definidos e responda as questões que seguem.	
		1.8.1	A interface está estereotipada com <<interface>>?
		1.8.2	O nome do método expressa corretamente sua função?
		1.8.3	O método está redundante com outro já definido anteriormente para esta classe?
	1.9	Os principais métodos foram definidos?	

Fonte: o autor

Figura 4.17: Inspeção de diagrama de classe para AQD - Parte 2

Passo 2	Depois da análise de todas as classes no passo anterior (todas demarcadas como visitadas), verifique e analise as questões que seguem.		
	2.1	As classes de controle gerenciam as atividades da classe. Para cada classe deste tipo que esteja estereotipada com <<optional>> e/ou <<variationPoint>>. Verifique-a para garantir que as notações de variabilidade/variantes estejam corretas de acordo com a especificação de requisitos. Para tal, vá até cada uma desses classes e responda as questões que seguem.	
		2.1.1	Há uma notação de UML que representa variabilidade (<<variabilitty>>) associada à classe de controle?
		2.1.2	Todas as variantes foram definidas e estão com a notação de variante correta (<<alternative OR>>, <<XOR>> ou <<optional>>) ?
	2.2	O estereótipo <<variability>> representa a variabilidade por meio de um comentário UML. Para cada um desses comentários definidos nos diagramas, vá até o comentário, analise-o e responda as questões abaixo.	
		2.2.1	As variantes definidas no conjunto variants realmente são variantes para este elemento? Se alguma não for, desconsidere-a para as próximas questões.
		2.2.2	Há variantes definidas na coleção de variantes que não estão descritas no diagrama de classes?
		2.2.3	Todas as variantes relacionadas a este elemento definidas com (<<OR>>), (<<XOR>>) ou (<<optional>>) estão na coleção de variantes do meta-atributo *variants* com seu nome correto?
		2.2.4	Verifique o tipo das variantes associadas: • Se forem do tipo <<optional>>, minSelection= 0 e maxSelection= 1? • Se forem do tipo <<OR>>, minSelection=1 e maxSelection=total de variantes?. • Se forem do tipo <<XOR>>, minSelection=maxSelection=1?
	2.3	Ainda há itens em sua lista que não estão em nenhuma classe? Ou seja, está faltando no diagrama de classe (Se forem variantes que já foram identificadas no passo anterior, desconsidere).	

Fonte: o autor

4.5.2 AQD - Diagrama de Componente

O diagrama de componente é um artefato importante para o AQD pois, ele expressa como as classes do sistema estão agrupadas em componentes e como eles se relacionam por meio de interfaces, representando assim, a arquitetura de LPS.

A inspeção do diagrama de componente acontece em dois passos: o primeiro passo compreende a inspeção de cada elemento do diagrama de componentes, seja ele componente ou interface. Por causa da grande quantidade de informações que devem ser analisadas, elas foram agrupadas em quatro sub passos (Figura 4.18):

- primeiramente são verificadas as relações entre os elementos, em especial a ordem dos contratos entre componentes e interfaces;
- os elementos que são do tipo opcional são verificados quanto a seu estereótipo e a notação de variabilidade associada;

- para elementos que representam um ponto de variação, eles são verificados para validar se estão de acordo com a especificação de requisitos; e
- no quarto sub passo, o inspetor deve revisar os componentes e interfaces que estão descritas no formato de classificador, analisando se as variantes associadas a ele então visíveis no compartimento com seu estereótipo correto.

Após repetir as questões do Passo 3 para todos os elementos do diagrama, o último passo (Passo 4 - Figura 4.19) verifica os meta-atributos de todas as notações de variabilidade presentes no modelo, analisando se as variantes estão definidas corretamente. Por fim, indaga o leitor a verificar se alguma das classes do diagrama de classe não foi modelada em um dos componentes, causando um defeito de omissão.

Figura 4.18: Inspeção de diagrama de componente para AQD - Parte 1

LOCALIZE O DIAGRAMA DE COMPONENTES E DE CLASSES			
O diagrama de componentes na Engenharia de Domínio deve mostrar a estrutura física da implementação por meio de componentes, interfaces e suas relações de todos os componentes do sistema, visto que, alguns serão obrigatórios por todos os produtos e outros serão arranjados de acordo com as necessidades específicas do usuário. Para inspecioná-lo, faça uma lista a partir do diagrama de classe com todas as classes do sistema (lembre-se de ter inspecionado o diagrama de classes antes). Compare a lista com o diagrama de componentes para garantir que não haja nenhuma inconsistência entre eles e que todas as classes estejam agrupadas em componentes. Para tal, responda as questões que seguem.			
Passo 3	Considere para os próximos passos um elemento como sendo um componente ou interface. Para cada um deles especificado no diagrama de componentes verifique sua correspondência com a lista feita, analise-o para responder cada uma das questões que seguem. Lembre-se de marcar os elementos de sua lista que já foram definidos em algum dos componentes.		
	3.1	O nome expressa corretamente o elemento?	
	3.2	O elemento está em redundância com outro já especificado anteriormente? Se sim, desconsidere-o para os próximos passos e vá para o próximo elemento	
	3.3	O elemento está em sua lista? Se não, desconsidere-o para os próximos passos e vá para o próximo elemento.	
	3.4	O elemento está estereotipado no diagrama de componentes?	
	3.5	Se o componente for obrigatório, ele está demarcado com o estereótipo <<mandatory>> ?	
	3.6	Para cada relação deste elemento, verifique-a e responda as questões que seguem.	
		3.6.1	O relacionamento do componente/interface com outro elemento realmente existe?
		3.6.2	Se a relação for do tipo de contrato, a ordem fornecida/requerida ou requerida/fornecida está correta?
		3.6.3	Variantes que exigem a presença de outra estão relacionadas com o estereótipo <<requires>> ou <<use>>?
		3.6.4	As variantes mutuamente exclusivas estão relacionadas com o estereótipo <<mutex>>?
	3.7	Se o elemento for do tipo opcional, verifique-o e responda as questões que seguem.	
		3.7.1	O estereótipo <<optional>> foi definido para o elemento?
		3.7.2	Há uma notação de UML que representa variabilidade (<<variabilitty>>) associada ao elemento?
	3.8	Se o elemento representar um ponto de variação, verifique-o e responda as questões que seguem.	
		3.8.1	O estereótipo <<variationPoint>> foi definido para o elemento?
		3.8.2	Há uma notação de UML que representa variabilidade (<<variabilitty>>) associada ao elemento?
		3.8.3	Todas as variantes foram definidas e estão com a notação de variante correta (<<OR>> ou <<XOR>>) ?
		3.8.4	Há alguma variante descrita no diagrama de componente que não pertence a este elemento?
		3.8.5	Há alguma variante em redundância com outra já especificada?
		3.8.6	Há alguma variante relacionada ao elemento que foi definida com (<<OR>>) ou (<<XOR>>) que não foi especificada nos requisitos? Se sim, retire-o do diagrama e desconsidere esta variante e suas relações para as próximas questões.

Fonte: o autor

Figura 4.19: Inspeção de diagrama de componente para AQD - Parte 2

Passo 4	Depois da análise de todos os elementos no passo anterior (todos demarcados como visitados), verifique e analise as questões que seguem.		
	4.1	O estereótipo <<variability>> representa a variabilidade por meio de um comentário UML. Para cada um desses comentários definidos nos diagramas, vá até o comentário, analise-o e responda as questões abaixo.	
		4.1.1	As variantes definidas no conjunto *variants* realmente são variantes para este elemento? Se alguma não for, desconsidere-a para as próximas questões.
		4.1.2	Há variantes definidas na coleção de variantes que não estão descritas no diagrama de componente?
		4.1.3	Todas as variantes relacionadas a este elemento definidas com (<<OR>>), (<<XOR>>) ou (<<optional>>) estão na coleção de variantes do meta-atributo *variants* com seu nome correto?
		4.1.4	Verifique o tipo das variantes associadas: • Se forem do tipo <<optional>>, minSelection= 0 e maxSelection= 1? • Se forem do tipo <<OR>>, minSelection=1 e maxSelection=total de variantes?. • Se forem do tipo <<XOR>>, minSelection=maxSelection=1?
	4.2	Para componentes e interfaces descritas no formato de classificador. Vá até o compartimento do classificador e verifique as operações descritas.	
		4.2.1	O componente/interface está com o estereótipo de <<variationPoint>> definido?
		4.2.2	Todos os componente/porta/interface variantes para resolver este ponto de variação estão visíveis no compartimento de operações?
		4.2.3	Há alguma porta/componente/interface especificada no compartimento de operações que não pertence a este elemento?
		4.2.4	Há alguma redundância das portas/componentes/interfaces especificadas no compartimento de operações?
		4.2.5	Os estereótipos no compartimento de operações estão corretos de acordo com cada um dos elementos ali descritos?
	4.3	Ainda há itens em sua lista que não pertencem a nenhum componente? Ou seja, está faltando no diagrama de componente (Se forem variantes que já foram identificadas no passo anterior, desconsidere).	

Fonte: o autor

4.6 Desenvolvedor de Domínio (DSD)

O DSD é responsável por implementar todos os componentes e interfaces que serão reutilizados pela gama de produtos da LPS e que serão configurados e executados como uma aplicação na atividade de Engenharia de Aplicação dependendo das escolhas do cliente.

A perspectiva do DSD deve revisar os diagramas que permitem ter uma visão geral da LPS, com informações sobre as funcionalidades que ela oferece e as atividades que devem ser executadas por um usuário para que ele possa implementar as interfaces e componentes da plataforma. Portanto, os diagramas que oferecem essa visão para o papel são os diagramas de classes, sequência e componente (Figura 4.20).

Figura 4.20: Cenário para a perspectiva de DSD

DESENVOLVEDOR DE DOMÍNIO
Essa perspectiva deve implementar e testar os componentes e interfaces que serão reutilizáveis por todo o portfólio dos produtos. Deve desenvolver de acordo com os requisitos de uma gama de produtos, para isso, deve garantir que os elementos representados nos diagramas *SMarty* de classe, sequência e componente não estejam em discordância com os requisitos, contemplem as funções esperadas e permita-lhe ter uma visão da implementação dos componentes reutilizáveis.
Para que os diagramas expressem corretamente os requisitos dos usuários sem inconsistências, você deve revisar os diagramas que colaboram com a sua função. Para atingir seu objetivo, execute os passos descritos abaixo para inspecionar cada um desses diagramas. Ao encontrar um defeito em um dos passos, preencha no Formulário de Identificação de Defeitos o diagrama, item do passo (questão), o elemento e o defeito que foi encontrado.

LOCALIZE O DIAGRAMA DE DE CLASSE E A ESPECIFICAÇÃO DOS REQUISITOS	
Passo 1	Inspeção diagrama de classe
Passo 2	
LOCALIZE O DIAGRAMA DE CLASSE E DE COMPONENTE	
Passo 3	Inspeção diagrama de componente
Passo 4	
LOCALIZE O DIAGRAMA DE SEQUÊNCIA, DE CLASSE E A ESPECIFICAÇÃO DE REQUISITOS	
Passo 5	Inspeção diagrama de sequência
Passo 6	
Passo 7	
Passo 8	

Fonte: o autor

Apesar de não ser o intuito da *SMartyPerspective*, o papel de DSD tem uma sobreposição com o papel do arquiteto em relação à inspeção de diagramas de componente, pois, o DSD deve realizar a arquitetura de LPS elaborada pelo AQD. Sendo assim, do ponto de vista destes artefatos, ambos devem verificar os mesmos elementos neste diagrama relacionados à visão técnica do sistema.

Em relação ao diagrama de sequência (também inspecionado pela perspectiva de ERD) as questões sofreram algumas alterações, pois, agora as mensagens definidas no diagrama de sequência devem estar em conformidade com o diagrama de classe para que não hajam dúvidas durante a inspeção das funções que o método devem conter.

4.6.1 DSD - Diagrama de Classe

O diagrama de classe oferece ao leitor diversas perspectivas para a extração das informações ali contidas: visão conceitual, de projeto e de implementação. Apesar do diagrama já ser inspecionado pelos cenários de AQD e ERD, para o DSD há ainda mais questões que devem ser analisadas para que a implementação esteja correta de acordo com o proposto para esta perspectiva.

Para o DSD informações relativas à implementação devem ser inspecionadas no diagrama de classe como, por exemplo, se os tipos dos dados estão corretos e os parâmetros de entrada dos métodos.

O diagrama de classe para a perspectiva de DSD é composto de questões "avulsas" e de dois passos principais (Figura 4.21 e Figura 4.22): no primeiro (Passo 1) são analisados cada um dos elementos e o segundo (Passo 2), são as mesmas questões encontradas no cenário de AQD em que são inspecionadas informações referentes às variantes nas classes de controle e na notação de variabilidade.

No Passo 1 as questões foram agrupadas em quatro seções que correspondem a: questões sobre pacote que a classe está inserida, relacionamentos, classes de entidade e classes de interface. A grande diferença desta perspectiva para o AQD são os grupos (1.8 e 1.10) que contém informações específicas para implementação.

As classes de entidade contém as informações que armazenam os dados que identificam os conceitos do mundo real, portanto, no grupo 1.8, o compartimento de atributos é inspecionado para verificar os atributos em relação a seu nome, tipo e visibilidade.

O compartimento de métodos, geralmente relacionados a classes de interfaces, é inspecionado no grupo 1.10. Diferentemente da perspectiva de AQD, o desenvolvedor deve garantir que todos os métodos estão descritos para a implementação. Além disso, são verificados sua visibilidade e se os parâmetros de entrada e saída bem como seus tipos estão corretos.

Figura 4.21: Inspeção de diagramas de classe para DSD - Parte 1

LOCALIZE O DIAGRAMA DE DE CLASSES E A ESPECIFICAÇÃO DOS REQUISITOS			
O diagrama de classes para seu papel deve apresentar a estrutura das classes e suas relações para todos os sistemas da LPS no domínio do projeto, incluindo detalhes específicos para implementação, como tipos de atributos e parâmetros dos métodos. Para garantir que este diagrama descreve as classes de projeto da maneira correta, leia atentamente a especificação de requisitos e durante a leitura, faça uma lista com os objetos mencionados, os dados que caracterizam este objeto e os possíveis métodos. Compare a lista feita com o diagrama de classes para garantir que não há inconsistência entre as classes e os requisitos do usuário. Para tal, responda as questões que seguem.			
Passo 1	Para cada classe do diagrama de classe, verifique os atributos, os relacionamentos entre as classes com a lista feita e responda as questões que seguem. Faça uma marcação na classe após sua análise, para evitar que seja analisada novamente.		
	1.1	O nome da classe expressa corretamente os objetos desta classe?	
	1.2	Esta classe corresponde a um objeto que não foi definido no documento de requisitos? Se sim, desconsidere ela e seus relacionamentos para os próximos passos e vá para a próxima classe.	
	1.3	Esta classe está em redundância com outra já especificada no diagrama de classe? Se sim, desconsidere ela e seus relacionamentos para os próximos passos e vá para a próxima classe.	
	1.4	A classe está estereotipada no diagrama de classe?	
	1.5	Se as classes estiverem agrupadas em pacotes, verifique e responda as questões que seguem.	
		1.5.1	O nome do pacote foi definido e expressa corretamente o agrupamento? Se já tiver colocado no formulário o defeito para este pacote não precisa colocar novamente.
		1.5.2	A classe está dentro do pacote correto?
	1.6	Para cada relacionamento para esta classe, verifique as classes que compõem este relacionamento, para garantir que o relacionamento entre elas esteja de acordo com a especificação dos requisitos. Para tal, responda as questões que seguem e faça uma marcação nos relacionamentos já analisados.	
		1.6.1	Este relacionamento está de acordo com a especificação de requisitos? Verifique se há realmente relação entre os elementos para o contexto.
		1.6.2	A cardinalidade deste relacionamento está correta de acordo com a especificação de requisitos?
		1.6.3	Verifique o tipo do relacionamento para garantir que as classes estejam com seus estereótipos definidos e corretos de acordo com a abordagem *SMarty*: • Generalização: Os classificadores mais gerais, são pontos de variação (<<variationPoint>>) e os mais específicos, variantes? • Realização de interface: As especificações são pontos de variação (<<variationPoint>>) e as implementações são variantes? • Agregação ou Composição: As instâncias tipadas com losangos (preenchidos ou não) são pontos de variação (<<variationPoint>>) e instâncias associadas são variantes? • Associação: Verifique o atributo AgregationKind • se for none: as variantes sugerem ser obrigatórias (<<mandatory>>) ou opcionais (<<optional>>). Verifique de acordo com a especificação de requisitos. • se o valor * ou 0..n são do tipo opcionais (<<optional>>)?
		1.6.4	As classes que exigem a presença de outra estão relacionadas com o estereótipo <<requires>> ?
		1.6.5	As classes mutuamente exclusivas estão relacionadas com o estereótipo <<mutex>>?
	1.7	Faltou algum relacionamento para esta classe que não foi especificado no diagrama de classes?	
	1.8	As classes de entidade armazenam os dados que identificam os conceitos do mundo real. Se no compartimento de atributos tiverem sido definidos os dados e/ou seus tipos, analise essas informações e para cada atributo desta classe responda as questões que seguem.	
		1.8.1	O nome do atributo expressa corretamente o dado do mundo real?
		1.8.2	O atributo realmente pertence a esta classe?
		1.8.3	O atributo está redundante com outro já definido para esta classe?
		1.8.4	O tipo do atributo está correto?
		1.8.5	A visibilidade do atributo na classe está correta?
	1.9	Os principais atributos foram definidos para esta classe?	

Fonte: o autor

Figura 4.22: Inspeção de diagramas de classe para DSD - Parte 2

Passo 1	1.10	As interfaces especificam os métodos externamente visíveis para outras. Se a classe for deste tipo, analise cada um dos métodos definidos e responda as questões que seguem.	
		1.10.1	A classe está estereotipada com <<interface>>?
		1.10.2	O nome do método expressa corretamente sua função?
		1.10.3	O método está redundante com outro já definido anteriormente para esta classe?
		1.10.4	Verifique os parâmetros de entrada para o método e responda as questões que seguem: • Faltou algum parâmetro de entrada para o método? • Algum parâmetro foi especificado além do necessário? • Os tipos dos parâmetros foram especificados corretamente?
		1.10.5	Verifique os parâmetros de saída para o método e responda as questões que seguem: • O valor do retorno está correto? • O tipo do retorno está correto?
		1.10.6	A visibilidade do método está correta?
	1.11	Faltou algum método importante para a implementação a ser definido?	
Passo 2	Depois da análise de todas as classes no passo anterior (todas demarcadas como visitadas), verifique e analise as questões que seguem.		
	2.1	As classes de controle gerenciam as atividades da classe. Para cada classe deste tipo que esteja estereotipada com <<optional>> e/ou <<variationPoint>>. Verifique-a para garantir que as notações de variabilidade/variantes estejam corretas de acordo com a especificação de requisitos. Para tal, vá até cada uma dessas classes e responda as questões que seguem.	
		2.1.1	Há uma notação de UML que representa variabilidade (<<variabilitty>>) associada à classe de controle?
		2.1.2	Todas as variantes foram definidas e estão com a notação de variante correta (<<OR>>, <<XOR>> ou <<optional>>) ?
	2.2	O estereótipo <<variability>> representa a variabilidade por meio de um comentário UML. Para cada um desses comentários definidos nos diagramas, vá até o comentário, analise-o e responda as questões abaixo.	
		2.2.1	As variantes definidas no conjunto *variants* realmente são variantes para este elemento? Se alguma não for, desconsidere-a para as próximas questões.
		2.2.2	Há variantes definidas na coleção de variantes que não está descrita no diagrama de classe?
		2.2.3	Todas as variantes relacionadas a este elemento definidas com (<<OR>>), (<<XOR>>) ou (<<optional>>) estão na coleção de variantes do meta-atributo *variants* com seu nome correto?
		2.2.4	Verifique o tipo das variantes associadas: • Se forem do tipo <<optional>>, minSelection= 0 e maxSelection= 1? • Se forem do tipo <<OR>>, minSelection=1 e maxSelection=total de variantes?. • Se forem do tipo <<XOR>>, minSelection=maxSelection=1?
	2.3	Ainda há itens em sua lista que não estão em nenhuma classe? Ou seja, está faltando no diagrama de classe. (Se forem variantes que já foram identificadas no passo anterior, desconsidere)	

Fonte: o autor

4.6.2 DSD - Diagrama de Componente

Como dito anteriormente, o DSD deve implementar a estrutura arquitetural definida pelo AQD, desenvolvendo os componentes reutilizáveis de acordo com o diagrama de componentes. Logo, as duas perspectivas têm a mesma visão deste artefato.

A parte do cenário do DSD que inspeciona os diagramas de componentes é a mesma do AQD, além do número que corresponde aos passos e itens. Sendo assim, as explicações deste cenário podem ser encontradas na Subseção 4.5.2 e a parte do cenário na Figura 4.18 e Figura 4.19.

4.6.3 DSD - Diagrama de Sequência

O diagrama de sequência para o DSD deve refletir a sequência entre os métodos em um determinado componente, já, que não há na Engenharia de Domínio uma implementação e teste de uma aplicação como um todo, mas sim, de partes do software que serão reutilizados na etapa de Engenharia de Aplicação.

Para o diagrama de sequência, a inspeção das mensagens deve ir além da feita pelo ERD, pois aqui, devem ser analisados detalhes mais próximos da implementação. As mensagens devem ser especificadas com a mesma assinatura utilizada pelo diagrama de classe para seus métodos que implementam essa função, assim como seus atributos caso existam.

A inspeção do diagrama de sequência é realizada em quatro passos (Figura 4.23 e Figura 4.24):

- no primeiro passo, as primeiras questões são basicamente iguais a da perspectiva de ERD, salvo, a relação do nome da linha de vida que deve estar de acordo com os objetos e classes do diagrama de classe. Este passo contém um sub passo (5.5) em que as mensagens são analisadas. Nele, ao contrário da perspectiva de ERD que verifica apenas se a mensagem foi definida, é analisado se as mensagens foram definidas de acordo com os métodos e seus parâmetros de entrada e de saída do diagrama de classe;

- o segundo passo, assim como no ERD, verifica os elementos alternativos do diagrama;

- os elementos opcionais são verificados no terceiro passo quanto ao estereótipo de opcional; e

- o Passo 8 (Figura 4.24) verifica os meta-atributos das notações UML que representam a variabilidade, identificado se as variantes foram definidas corretamente, bem como seu mínimo e máximo de seleção.

Figura 4.23: Inspeção de diagramas de sequência para DSD - Parte 1

Passo 7	Para cada elemento opcional no diagrama de sequência, como: combinedFragment com interactionOperator “opt”(optional) e troca de mensagens entre dois objetos não obrigatórios ou entre um objeto obrigatório e outro não, verifique seus estereótipos e as mensagens relacionadas a ele para responder as questões que seguem.	
	7.1	Há uma notação de UML que representa variabilidade (<<variabilitty>>) associada a este elemento/CombinedFragment?
	7.2	As variantes correspondentes às mensagens/CombinedFragment estão estereotipadas corretamente com <<optional>>?
	7.3	Para elementos demarcados com <<optional>>, as linha de vidas que fazem parte do CombinedFragment também estão estereotipadas com <<optional>>?
Passo 8	O estereótipo <<variability>> representa a variabilidade por meio de um comentário UML. Para cada um desses comentários definidos nos diagramas, analise as questões abaixo.	
	8.1	As variantes definidas no conjunto *variants* realmente são variantes para este elemento? Se alguma não for, desconsidere-a para as próximas questões.
	8.2	Todas as variantes do conjunto de variantes tem uma linha de vida associada no diagrama de sequência?
	8.3	Todas as variantes relacionadas a este elemento definidas com (<<OR>>), (<<XOR>>) ou (<<optional>>) estão na coleção de variantes do meta-atributo *variants* com seu nome correto?
	8.4	Verifique o tipo das variantes associadas: • Se forem do tipo <<OR>>, minSelection=1 e maxSelection=total de variantes?. • Se forem do tipo <<OR>>, minSelection=1 e maxSelection=total de variantes?. • Se forem do tipo <<XOR>>, minSelection=maxSelection=1?

Fonte: o autor

Figura 4.24: Inspeção de diagramas de sequências para DSD - Parte 2

LOCALIZE O DIAGRAMA DE SEQUÊNCIA, DE CLASSES E A ESPECIFICAÇÃO DE REQUISITOS			
O diagrama de sequência na Engenharia de Domínio deve expressar a interação do sistema: a troca das mensagens entre os objetos dos sistemas que podem ser configurados de uma LPS do ponto de vista técnico, próximo da implementação que você deverá realizar. Com o diagrama de sequência em mãos, verifique cada um dos objetos/atores descritos e busque pela classe correspondente no diagrama de classes (se estiver definido) ou na especificação de requisitos. Para cada um deles, verifique os relacionamentos e as mensagens que são trocadas entre os elementos para garantir que estejam consistentes com o diagrama de classes/especificação de requisitos. Em seguida, responda as questões que seguem.			
Passo 5	Considere como elemento as "cabeças de objeto" e os atores da linha de vida do diagrama de sequência. Para cada um deles verifique em sua linha de vida as mensagens e estereótipos dados a elas para responder as questões que seguem.		
	5.1	O nome do elemento expressa corretamente o objeto/ator?	
	5.2	O elemento está representado em alguma classe no diagrama de classe?	
	5.3	O elemento faz parte desta interação de acordo com a Especificação de Requisitos? Se não, desconsidere toda a linha de vida e suas mensagens para os próximos passos e vá para o próximo elemento.	
	5.4	O elemento é redundante com outro elemento definido? Se sim, desconsidere toda a linha de vida e suas mensagens para os próximos passos e vá para o próximo elemento.	
	5.5	Para cada uma das mensagens definidas na linha de vida deste elemento, analise-a e responda as questões que seguem.	
		5.5.1	A mensagem está nomeada?
		5.5.2	A interação representada por esta mensagem está descrita na especificação de requisitos? Se não, desconsidere-a e passe para a próxima mensagem.
		5.5.3	A ordem da mensagem está correta de acordo com a especificação de requisitos?
		5.5.4	O nome da mensagem está de acordo com o nome do método no diagrama de classe?
		5.5.5	Se estiver especificado, os parâmetros de entrada estão corretos de acordo com o diagrama de classe?
		5.5.6	Se estiver especificado, os parâmetros de saída estão corretos de acordo com o diagrama de classe?
		5.5.7	Mensagens que não estejam relacionadas diretamente à uma variabilidade e seus elementos, não precisam de estereótipo e são consideradas obrigatórias. A mensagem deste tipo está estereotipada?
	5.6	Todas as mensagens para este elemento foram definidas no diagrama de sequência de acordo com a especificação de requisitos e o diagrama de classe?	
	5.7	Há mensagens redundantes nesta linha de vida? Se sim, desconsidere-a e passe para a próxima mensagem.	
	5.8	Elementos que exigem a presença de outro estão relacionados com o estereótipo <<requires>> ?	
	5.9	Elementos mutuamente exclusivos estão relacionados com o estereótipo <<mutex>>?	
Passo 6	Para cada elemento alternativo no diagrama de sequência como o CombinedFragment com interactionOperator “alt”(alternative) e elemento interactionUse “ref”, verifique seus estereótipos e as mensagens relacionadas a este elemento para responder às questões que seguem.		
	6.1	O elemento está definido com o estereótipo <<variationPoint>> ?	
	6.2	Há uma notação de UML que representa variabilidade (<<variabilitty>>) associada a este elemento/CombinedFragment?	
	6.3	As variantes correspondente às mensagens estão estereotipadas corretamente? Verifique os estereótipos das mensagens variantes para esta variabilidade. • Para as variantes do elemento interactionUse “ref”: <<OR>> • Para as variiantes com interactionOperator "alt": <<XOR>>	

Fonte: o autor

4.7 Gerente de Ativos de Domínio (GAD)

O cenário do GAD compreende a inspeção dos quatro diagramas UML *SMarty*: diagramas de caso de uso, classe, componente e sequência, e o diagrama de *features*, pois, como já dito, ele deve manter a versão mais recente e correta na base de ativos de todos os artefatos ali contidos.

As questões presentes no cenário do GAD são em sua grande maioria as mesmas para todos os diagramas, pois, a maioria questões não estão relacionadas especificamente aos elementos dos tipos de diagramas, mas sim, da comparação entre diferentes versões de um mesmo modelo. Exceto informações de atributos e métodos no diagrama de classe, compartimento de classificadores no diagrama de componente e ordem das mensagens no diagrama de sequência.

O cenário do GAD é composto de quatro passos (Figura 4.25). Os três primeiros são comuns para todos os diagramas inspecionados pela *SMartyPerspective*, enquanto o segundo, é especifico para diagramas *SMarty*, pois, trata da rastreabilidade entre elementos suportada pela abordagem.

Figura 4.25: Cenário para a perspectiva de GAD

GERENTE DE ATIVOS DE DOMÍNIO
Essa perspectiva interage com todas as atividades da Engenharia de Domínio de forma a gerenciar as versões e as variantes dos artefatos produzidos nessa fase. Portanto, deve garantir que as versões dos diagramas de caso de uso, classe, componente, sequência e *features* sejam válidas e rastreáveis.
Para que os diagramas expressem corretamente os requisitos dos usuários sem inconsistências, você deve revisar os diagramas que colaboram com a sua função. Para atingir seu objetivo, execute os passos descritos abaixo para inspecionar cada um desses diagramas. Ao encontrar um defeito em um dos passos, preencha no Formulário de Identificação de Defeitos o diagrama, item do passo (questão), o elemento e o defeito que foi encontrado.

LOCALIZE O DIAGRAMA A SER INSPECIONADO	
Passo 1	Inspeção diagrama de *features*, caso de uso, classe, componente e sequência
Passo 2	
Passo 3	
LOCALIZE TODOS OS DIAGRAMAS *SMARTY* A SEREM INSPECIONADOS	
Passo 4	Inspeção dos diagramas *SMarty* de caso de uso, classe, componente e sequência

Fonte: o autor

Figura 4.26: Inspeção de diagramas para GAD - Parte

LOCALIZE O DIAGRAMA A SER INSPECIONADO			
Por favor, vá até a primeira versão do diagrama da LPS (oráculo), verifique o registro de mudanças para a nova versão da LPS. Corrija o diagrama excluindo ou adicionando os novos elementos conforme descrito no registro (lembre-se das diretrizes para a definição dos diagramas). Depois, vá até a versão que será inspecionada e compare-a com o oráculo para evitar a inconsistência entre eles, marcando os elementos conforme estiverem sendo inspecionados. Para tal, responda as questões que seguem.			
Passo 1	Para cada elemento do diagrama que está sendo inspecionado, verifique-o e faça uma marcação de visitado quando terminar as questões sobre ele do Passo 1:		
	1.1	É possível rastrear o elemento para o oráculo? Se não, desconsidere-o com suas relações e vá para o próximo elemento (Lembre-se que elementos adicionados no registro de mudança não irão aparecer no oráculo)	
	1.2	O nome do elemento na segunda versão difere/está abreviado da primeira versão ?	
	1.3	Na segunda versão, o estereótipo do elemento foi definido?	
	1.4	O estereótipo do elemento na segunda versão está correto?	
	1.5	Para cada relacionamento/mensagem deste elemento, verifique e responda as questões que seguem. Lembre-se de fazer uma marcação nas relações já analisadas.	
		1.5.1	Esta relação/mensagem existe no oráculo? Se não existir, desconsidere as próximas questões e vá para a próxima.
		1.5.2	Se houver, o estereótipo da relação/mensagem foi definido corretamente?
		1.5.3	Para o diagrama de sequência, verifique as mensagens: • A ordem da mensagem está correta de acordo com o oráculo? • Se houver definido, os parâmetros de entrada e saída estão corretos de acordo com o oráculo?
	1.6	Se estiver inspecionando o diagrama de classe, analise os atributos e métodos das classes para responder as questões que seguem:	
		1.6.1	Se houver, os atributos desta classe estão corretos de acordo com o oráculo? Verifique o nome e o tipo para responder a questão.
		1.6.2	Se houver, os métodos desta classe estão corretos de acordo com o oráculo? Verifique os parâmetros de entrada e saída e seus tipos para responder a questão.
	1.7	Se estiver inspecionando o diagrama de componente, analise as interfaces e componentes definidos no formato de classificador para responder as questões que seguem:	
		1.7.1	Todos os componente/porta/interface variantes estão visíveis no compartimento de operações assim como no oráculo? Verifique também os estereótipos
Passo 2	Se estiver inspecionando algum diagrama *SMarty*. Para cada elemento com uma variabilidade relacionada (<<variability>>), vá até a notação UML e procure-a na segunda versão:		
	2.1	A notação de variabilidade está presente na versão inspecionada?	
	2.2	O nome (meta-atributo name) da variabilidade está redundante com outra já definida?	
	2.3	O nome (meta-atributo name) da variabilidade a reflete corretamente ?	
	2.4	minSelection e maxSelection estão definidos corretamente? Lembre-se de analisar se foi inserida alguma nova variante de acordo com o registro de mudanças.	
	2.5	Verifique as variantes associadas. Estão presentes com seu nome correto como no oráculo?	
	2.6	Verifique o meta-atributo alllowsAddingVar. Está correto de acordo com o oráculo?	
	2.7	Verifique o meta-atributo bindingTime. Está correto de acordo com o oráculo?	
	2.8	Verifique os meta-atributos realizes+ e realizes-. Eles estão especificados na segunda versão, assim como na primeira?	
Passo 3	Depois da análise de todos os elementos nos passos anteriores (todos os elementos do diagrama inspecionado demarcados como visitados), verifique e analise as questões que seguem no diagrama oráculo.		
	3.1	Algum elemento do oráculo não está presente na versão inspecionada?	
	3.2	Algum relacionamento do oráculo não está presente na versão inspecionada?	

Fonte: o autor

O cenário de GAD pode ser dividido em duas etapas. Em ambas, o inspetor deve ter em mãos as duas versões de um mesmo diagrama para poder comparar se estão consistentes.

Na primeira etapa são verificadas em três passos as inconsistências entre as versões do elemento relacionadas ao nome, relacionamentos, estereótipos *SMarty* e defeitos nos meta-atributos da notação UML de variabilidade. Especialmente o nome das variabilidades para que sejam rastreadas na etapa seguinte.

Neste passo, por exemplo, são encontrados defeitos em que o nome dado ao elemento nas duas versões refletem sua característica, mas, não são exatamente os mesmos (como `MediaCtrl` e `MediaControl`) entre as versões do diagrama. Este tipo de defeito passa despercebido pelo questionário das outras perspectivas, mas, no GAD ele é verificado para que não haja problemas quanto à rastreabilidade dos elementos e por não seguir o padrão definido.

Já na segunda etapa é verificada a rastreabilidade entre os elementos definidos no conjunto do meta-atributo `realizes+` e `realizes-` do estereótipo `«variability»` da abordagem *SMarty*, para garantir a rastreabilidade entre os diagramas e que os elementos descritos no conjunto dos meta-atributos estão presentes no diagrama correspondente.

Figura 4.27: Inspeção de diagramas *SMarty* para GAD

LOCALIZE TODOS OS DIAGRAMAS SMARTY A SEREM INSPECIONADOS		
Os meta-atributos realizes+ e realizes- permitem a rastreabilidade em diagramas SMarty. Para cada diagrama inspecionado verifique para quais diagramas ele é rastreável. Para garantir que é realmente possível rastrear esses elementos de um diagrama para o outro, analise o diagrama e responda as questões que seguem.		
Passo 4	Separe o diagrama a ser inspecionado. Para cada notação de comentário UML com o estereótipo <<variability>> verifique se há elementos nos conjuntos dos meta-atributos realizes+ e realizes-. Caso haja, procure a variabilidade com o nome definido no conjunto, no seguinte diagrama: • Para realizes+: Vá até o diagrama mais abstrato correspondente. • Para realizes-: Vá até o diagrama menos abstrato correspondente.	
	4.1	Você encontrou a variabilidade definida no diagrama inspecionado no diagrama correspondente?
	4.2	A variabilidade no diagrama correspondente realiza a variabilidade do diagrama inspecionado?
	4.3	Analise os diagramas correspondentes. Faltou alguma variabilidade em realizes+ e realizes- do diagrama inspecionado que é rastreável de um diagrama para o outro?

Fonte: o autor

4.8 Considerações Finais

Neste capítulo foi apresentada a família de técnicas para inspeção de diagramas de gerenciamento de variabilidades na Engenharia de Domínio de Linha de Produto de

Software denominada *SMartyPerspective*, por meio dos seus cinco cenários (ou técnicas) da família.

Os cenários da *SMartyPerspective* foram definidos para cinco papéis da Engenharia de Domínio: Gerente de Produto, Engenheiro de Requisitos de Domínio, Arquiteto de Domínio, Desenvolvedor de Domínio e Gerente de Ativos de Domínio. Para cada um deles foram definidos os diagramas de gerenciamento de variabilidades que são importantes para a realização da sua tarefa no processo de desenvolvimento de software.

Além dos diagramas, cada papel ficou responsável por uma determinada perspectiva desses artefatos. Assim, cada um deles engloba uma quantidade diferente de questões e tipos de defeitos, apresentado no Apêndice C, de acordo com as necessidades específicas de sua função.

No Capítulo 5 será apresentado um exemplo para cada uma das perspectivas para a inspeção de alguns tipos de diagramas que podem ser inspecionados pela *SMartyPerspective* para compreensão do funcionamento dos cenários operacionais da técnica.

5

Exemplos de Aplicação da SMartyPerspective

5.1 Considerações Iniciais

Para fins didáticos do funcionamento das técnicas da *SMartyPerspective*, serão apresentados nesta seção seis exemplos de inspeção para diagramas da LPS Mobile Media (MM): (i) diagrama de *features* sob a ótica do Gerente de Produto; (ii) diagrama de caso de uso para o Engenheiro de Requisitos de Domínio; (iii) diagrama de componente para o Arquiteto de Domínio; (iv) diagrama de classe para o papel de Desenvolvedor de Domínio; e (vi) diagrama de caso de uso para o Gerente de Ativos de Domínio (GAD). Para cada um dos exemplos será apresentado o preenchimento do Formulário de Identificação de Defeitos (FID) e a leitura ativa com as marcações.

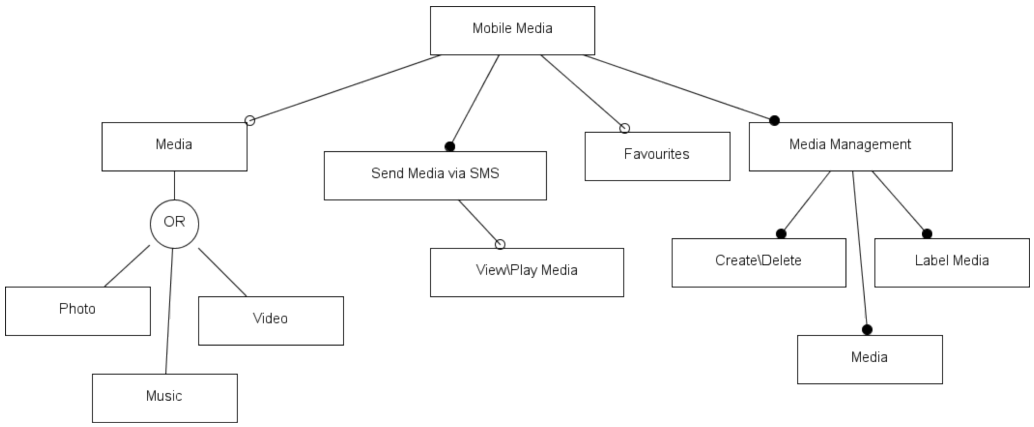
5.2 Exemplo #1 - Gerente de Produto (GRP)

A perspectiva de Gerente de Produto (GRP) na *SMartyPerspective* pode inspecionar os diagramas de *features* e de caso de uso. Para apresentar um exemplo de inspeção de GRP foram injetados defeitos no diagrama de *features* da LPS Mobile Media (Figura 5.1).

A instrução principal para o cenário de GRP para o diagrama de *features* orienta o GRP a analisar o documento de requisitos (Apêndice E) para verificar as principais características da LPS, sejam elas comuns ou variáveis, e criar uma lista com as características

com seus possíveis estereótipos, para depois comparar com o diagrama de *features* para iniciar a inspeção.

Figura 5.1: Diagrama de *features* da LPS MM com defeitos



Adaptado de Choma Neto (2017)

A Figura 5.2 apresenta a lista com as principais características e seus estereótipos da LPS MM. Esta lista servirá de base durante a inspeção para verificar os estereótipos das *features* e as que foram omissas ou inseridas erroneamente no diagrama.

Figura 5.2: Lista com as características da MM

Send Media	-	*Opcional/Ponto de variação*
- Email	-	*Alternativo*
- SMS	-	*Alternativo*
Manage Favourite Media	-	*Opcional*
Manage Media	-	*Obrigatório/Ponto de variação*
- Video	-	*Altenativo*
- Músic	-	*Altenativo*
- Photo	-	*Altenativo*
Media Management	-	*Obrigatório*
Crete/Delete	-	*Obrigatório*
Link Media with Address Book Entry	-	*Opcional*
View/Play Media	-	*Obrigatório*
Label Media	-	*Opcional*

Fonte: o autor

O cenário de GRP é composto de dois passos: (i) no primeiro são verificadas cada uma das características quanto a seus estereótipos, relações e se não está ambígua ou não faz parte do domínio; e (ii) no segundo é analisado se alguma característica ou relação não foi representada no diagrama.

Para cada passo, o inspetor deverá seguir o procedimento do cenário, verificando para cada elemento cada uma das questões na ordem que aparecem no cenário e riscando na lista da Figura 5.2 as características já encontradas no diagrama. A primeira questão deste cenário refere-se a análise do nó raiz do diagrama de *features*.

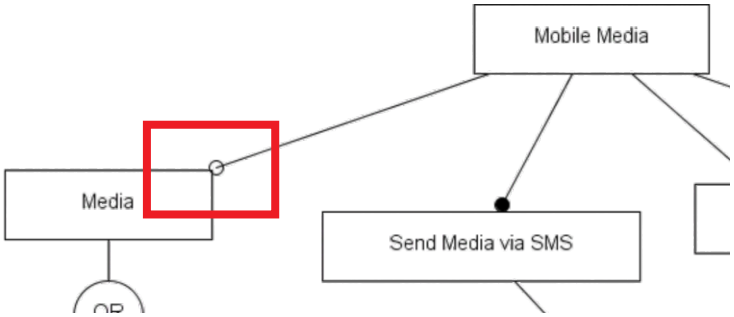
O nó raiz do diagrama de *features* deve informar aos seus leitores o domínio da LPS que ele representa. Portanto, antes de verificar os outros elementos do diagrama, o leitor deve verificar se o nó raiz apresenta de forma clara o domínio da LPS. Caso contrário, deverá ser anotado um defeito no FID.

A primeira característica selecionada para inspeção neste exemplo é a `Media`, ela se refere aos tipos de mídia suportados pela LPS MM. A partir da questão 1.2 até a 1.4 são analisadas informações referentes ao nome da *feature*, se está ambígua e se já foi especificada por outra característica. Para essas questões, não foi identificado nenhum defeito.

O grupo 1.5 identifica as arestas que chegam à *feature* e os estereótipos relacionados a ela. Para `Media`, a relação com o nó principal está correta de acordo com as especificações da LPS MM pela questão 1.5.1.

A questão 1.5.2 questiona se o elemento obrigatório foi definido com um círculo vazado. Ao analisarmos a especificação de requisitos, será verificado que `Media` é uma característica obrigatória e então deve ter o círculo preenchido, porém, não foi assim que a relação foi definida (Figura 5.3). A *feature* `Media` foi definida com um círculo vazado definindo-a como opcional. Logo, um defeito foi identificado para esta perspectiva.

Figura 5.3: Defeito na *feature* `Media`



Adaptado de Choma Neto (2017)

O defeito deverá ser informado no FID. Nele será preenchido o diagrama em inspeção, o elemento com defeito, a questão que orientou a encontra-lo e a descrição do defeito (Tabela 5.1):

- Diagrama: *feature* (FT);
- Nº questão: 1.5.2;
- Elemento: `Media`;
- Defeito identificado: A *feature* `Media` foi definida como sendo opcional, mas, na verdade é uma característica obrigatória para esta LPS.

Para as próximas questões do Passo 1, não terá mais nenhuma informação a ser analisada para esta *feature*, pois, o sub grupo é para relações alternativas(XOR) e OR e o sub grupo 1.7 para relações de inclusão e exclusão. Assim, para este exemplo, o leitor já poderá passar a próxima característica.

A inspeção irá continuar para os demais elementos. Neste exemplo, foram inspecionadas na ordem as *features*: `Photo, Music, Video, Send Media via SMS`. Para esses, não foram identificados nenhum tipo de defeito.

A próxima característica a ser analisada é a `View/Play Media`. Da questão 1.2 até a 1.4 não será identificado nenhum defeito para ela. Porém, ao verificar se a relação está correta por meio da questão 1.5.1, um defeito será encontrado: `View/Play Media` não tem relação com a *feature* **Send Media via SMS**:

- Diagrama: *feature* (FT);

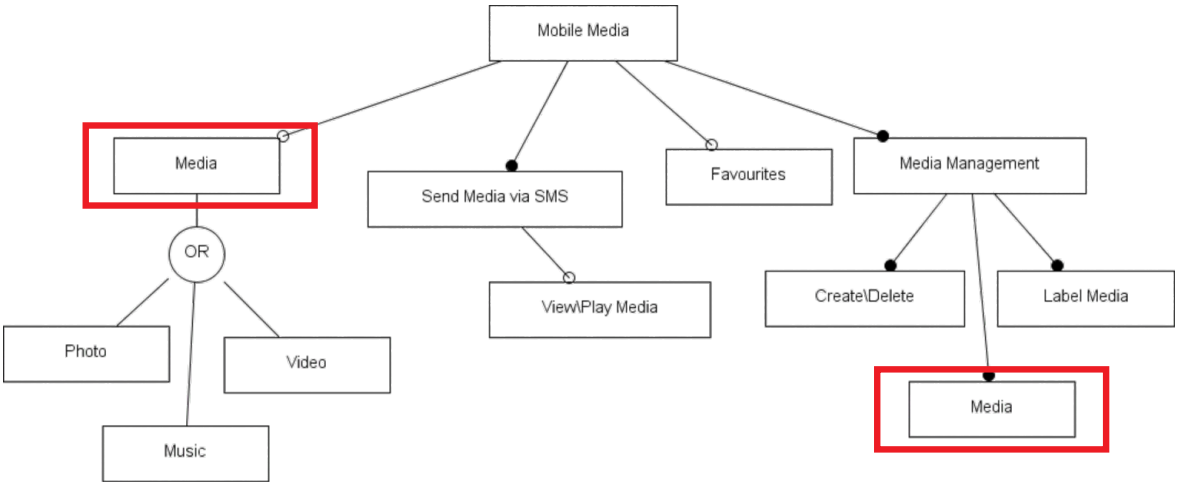
- Nº questão: 1.5.1;
- Elemento: `View/Play Media`;
- Defeito identificado: A *feature* `View/Play Media` não tem relação com `Send Media via SMS` de acordo com a especificação de requisitos.

Para `Media` foi identificado também o defeito de estereótipo incorreto pela questão 1.5.2, que verifica se a característica obrigatória está definida com um círculo preenchido, que não aconteceu para esta característica.

Foram ainda inspecionadas as *features* `Favourite, Media Management, LabelMedia e Create/Delete`. Para todas estas, nenhum defeito foi identificado por meio do procedimento do Passo 1.

Durante a inspeção, por meio da questão 1.4 foi identificado que a *feature* `Media` relacionada a `Media Management` está duplicada com outra *feature* (Figura 5.4) já inspecionada relacionada diretamente ao nó principal. Portanto, deverá ser identificado no FID o defeito encontrado (Tabela 5.1):

Figura 5.4: Defeito na *feature* `Media` - duplicada



Adaptado de Choma Neto (2017)

- Diagrama: *feature* (FT);
- Nº questão: 1.4;

- Elemento: `Media`;
- Defeito identificado: a *feature* Media relacionada a Media Management está duplicada. Já *feature* Media relacionada ao nó principal está correta.

Depois da análise de todos os itens do diagrama de *features* para o Passo 1, o GRP pode então passar para as questões do Passo 2. A questão 2.1 orienta o leitor a verificar se na lista feita com as características há alguma *feature* que não foi definida no diagrama.

Para este exemplo foi identificado na questão 2.1 por meio da lista feita na instrução principal (Figura 5.2) que as características relacionadas ao envio de mídia, ao envio de mídia por *email* e de vincular a mídia ao contato não foram definidas para este diagrama de *features* (Figura 5.5), configurando assim, um defeito que deve ser reportado um a um no FID (Tabela 5.1). Para o envio de mídia via *email* o FID ficará desta forma:

Figura 5.5: Lista com as características da MM após inspeção do Passo 1

Fonte: o autor

- Diagrama: *feature* (FT);
- Nº questão: 1.4;
- Elemento: `Send Media via SMS`;
- Defeito identificado: A *feature* `Send Media via Email não foi` especificada no diagrama de *features* da LPS Mobile Media.

Não foi identificado nenhum defeito referente à questão 2.2 para este exemplo, finalizando assim, a inspeção do diagrama de *features* da Figura 5.1. O FID final para este exemplo foi detalhado na Tabela 5.1.

Tabela 5.1: FID após inspeção do diagrama de *features* Figura 5.1 pelo GRD

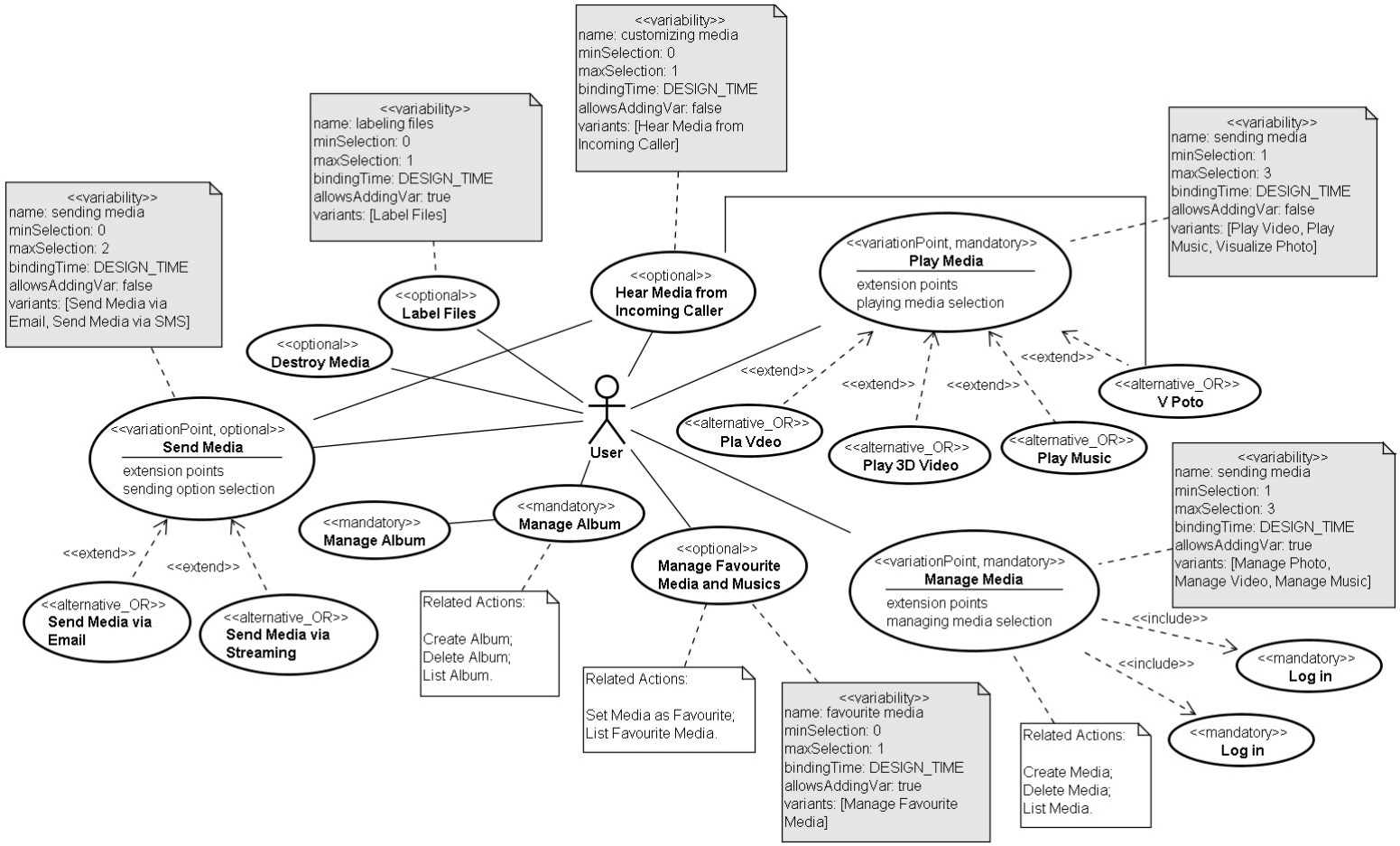
nº	DIAGRAMA					Nº QUESTÃO	ELEMENTO	DEFEITO IDENTIFICADO
	FT	UC	CL	CP	SQ			
1	X					1.5.2	Media	A *feature* Media foi definida como sendo opcional, mas, na verdade é uma característica obrigatória para esta LPS.
2	X					1.5.1	View/Play Media	A *feature* View/Play Media não tem relação com Send Media via SMS de acordo com a especificação de requisitos
3	X					1.5.1	View/Play Media	A *feature* View/Play Media foi definida como sendo opcional, mas, na verdade é uma característica obrigatória para esta LPS.
4	X					1.4	Media	A *feature* Media relacionada a Media Management está duplicada. Já *feature* Media relacionada ao nó principal está correta.
5	X					2.1	Send <edia	A *feature* Send Media não foi especificada no diagrama de *features* da LPS Mobile Media.
6	X					2.1	Send Media via Email	A *feature* Send Media via Email não foi especificada no diagrama de *features* da LPS Mobile Media.
7	X					2.1	Link Media	A *feature* Link Media não foi especificada no diagrama de *features* da LPS Mobile Media.

Fonte: o autor

5.3 Exemplo #2 - Engenheiro de Requisitos de Domínio (ERD)

Para não direcionar a técnica para os tipos de defeitos e as questões definidas para a *SMartyPerspective*, foi utilizado o diagrama de caso de uso da LPS MM com os defeitos incluídos por Geraldi e OliveiraJr (2017b) na instrumentação do estudo quantitativo da técnica *SMartyCheck*, como mostra a Figura 5.6.

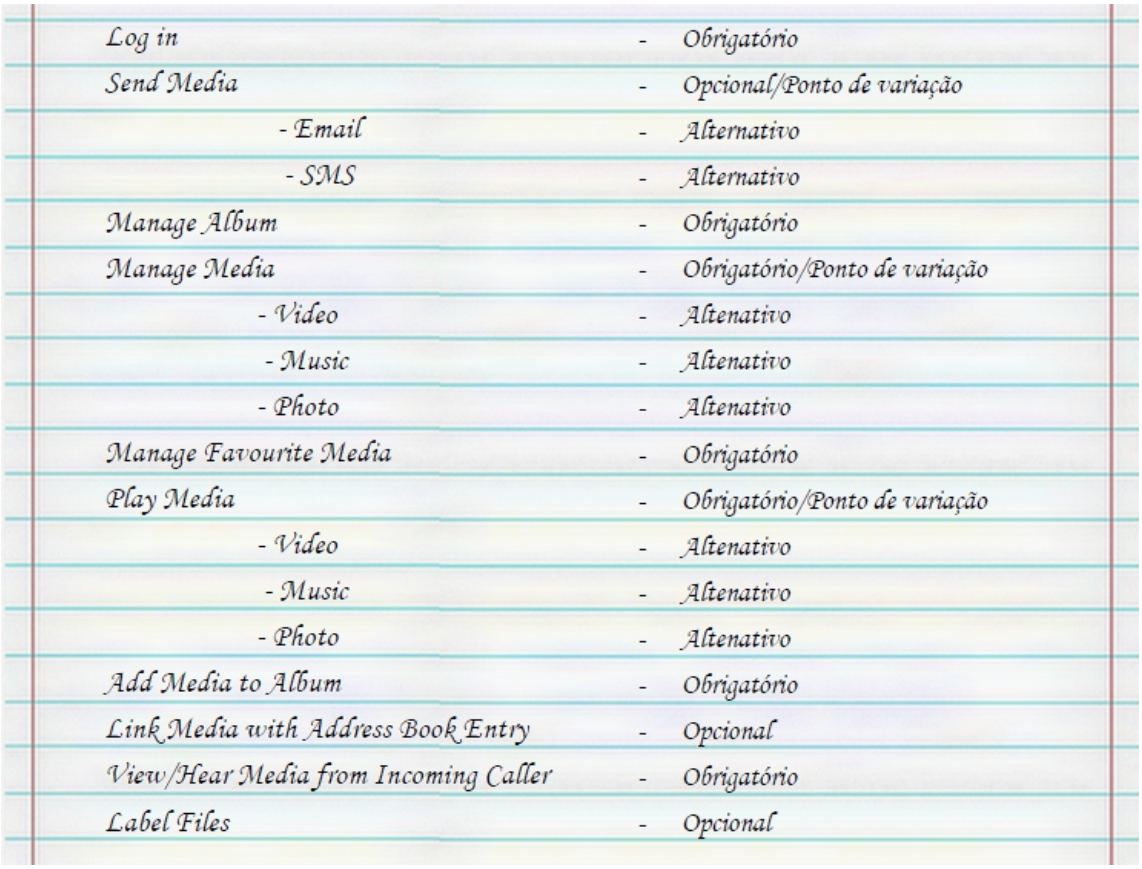
Figura 5.6: Diagrama de caso de uso da LPS MM baseado na *SMarty* - Com Defeitos



Fonte: Geraldi e OliveiraJr (2017b)

O leitor é livre para fazer a lista da sua maneira, com as marcações de estereótipos da forma que for mais segura para compreender durante a leitura do diagrama. A Figura 5.7 apresenta um exemplo de uma lista com itens da especificação de requisitos da MM conforme apresentada no Apêndice E.

Figura 5.7: Lista com as funcionalidades da MM



Fonte: o autor

Com a lista pronta, o inspetor deve seguir os passos definidos para este diagrama e perspectiva. Como já dito anteriormente, a leitura do diagrama de caso de uso para o ERD deve ser feita em 2 passos: o primeiro percorre todas os elementos e o segundo verifica especificidades do gerenciamento de variabilidade.

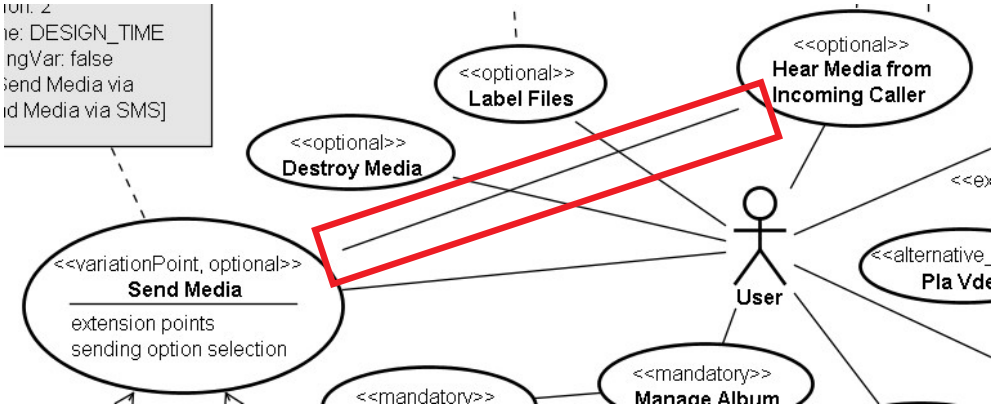
O Passo 1 deve ser concluído para cada elemento do diagrama de caso de uso, seja ele do tipo ator ou caso de uso. Para este exemplo, a inspeção foi iniciada a partir do caso de uso `LabelFiles`, porém, não foram encontradas nenhuma inconsistência entre o diagrama e a especificação de requisitos, pois, para todas as questões a resposta foi "não" para este elemento.

`Destroy Media` foi o próximo elemento inspecionado para o exemplo. O leitor ao se deparar com a questão 1.2 do Passo 1 foi indagado a analisar se este caso de uso representa uma funcionalidade do sistema, porém, ao analisarmos a Figura 5.7, nenhum dos itens da lista tem correspondência para esta funcionalidade, logo, um defeito foi identificado.

- Diagrama: caso de uso (UC);
- Nº questão: 1.2;
- Elemento: `Destroy Media`;
- Defeito identificado: Este caso de uso não foi definido na especificação de requisitos.

Ao chegar a vez da inspeção do caso de uso `Send Media`, o leitor irá ler as primeiras questões, e, ao chegar ao subgrupo de questões 1.7 referente às relações do elemento, será identificado que a relação com o `Hear Media from Incoming Caller` não existe (Figura 5.8):

Figura 5.8: Defeito no caso de uso `Send Media` - relação incorreta



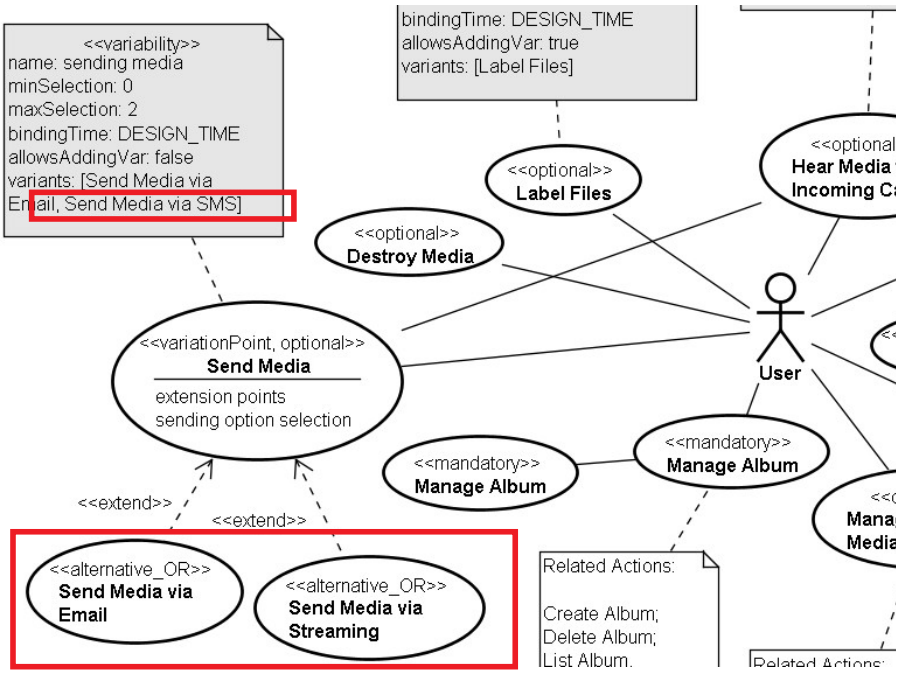
Fonte: adaptado de Geraldi *et al.* (2015)

- Diagrama: caso de uso (UC);
- Nº questão: 1.7.1;
- Elemento: Send Media;
- Defeito identificado: não há relação entre os casos de uso `Send Media e Hear Media from Incoming Caller`.

Após visitar todos os elementos no diagrama de caso de uso, o Passo 1 da técnica foi finalizado. No passo 2, o inspetor deve analisar primeiramente os elementos opcionais e que representam um ponto de variação e depois os comentários UML associados a estes estereótipos, buscando por defeitos específicos do gerenciamento variabilidade.

Ao iniciar o Passo 2 é detectado o defeito no caso de uso `Send Media`. No diagrama em inspeção está definido na notação de variabilidade no meta-atributo `variants` que o caso de uso `Send Media via SMS` é uma variante para o ponto de variação `Send Media`, porém, este elemento foi omitido do diagrama (Figura 5.9). No formulário FID (Tabela 5.2 - linha 11) foi reportado este defeito da seguinte forma:

Figura 5.9: Defeito no caso de uso `Send Media` - variante não encontrada



Fonte: adaptado de Geraldi *et al.* (2015)

- Diagrama: caso de uso;
- Nº questão: 2.2.2;
- Elemento: `Send Media`;
- Defeito identificado: O caso de uso `Send Media via SMS` foi definido em `variants` e não está presente no diagrama de caso de uso.

A última questão é analisada após inspeção de todos os subgrupos do Passo 2 anteriores que orientaram a identificar as variantes dos elementos opcionais e as notações que indicam uma variabilidade. O leitor deverá nesta questão olhar para a lista com as funcionalidades descritas e verificar se algum dos itens não foi representado por nenhum elemento do diagrama de caso de uso, caracterizado assim, como um defeito de omissão (Figura 5.10).

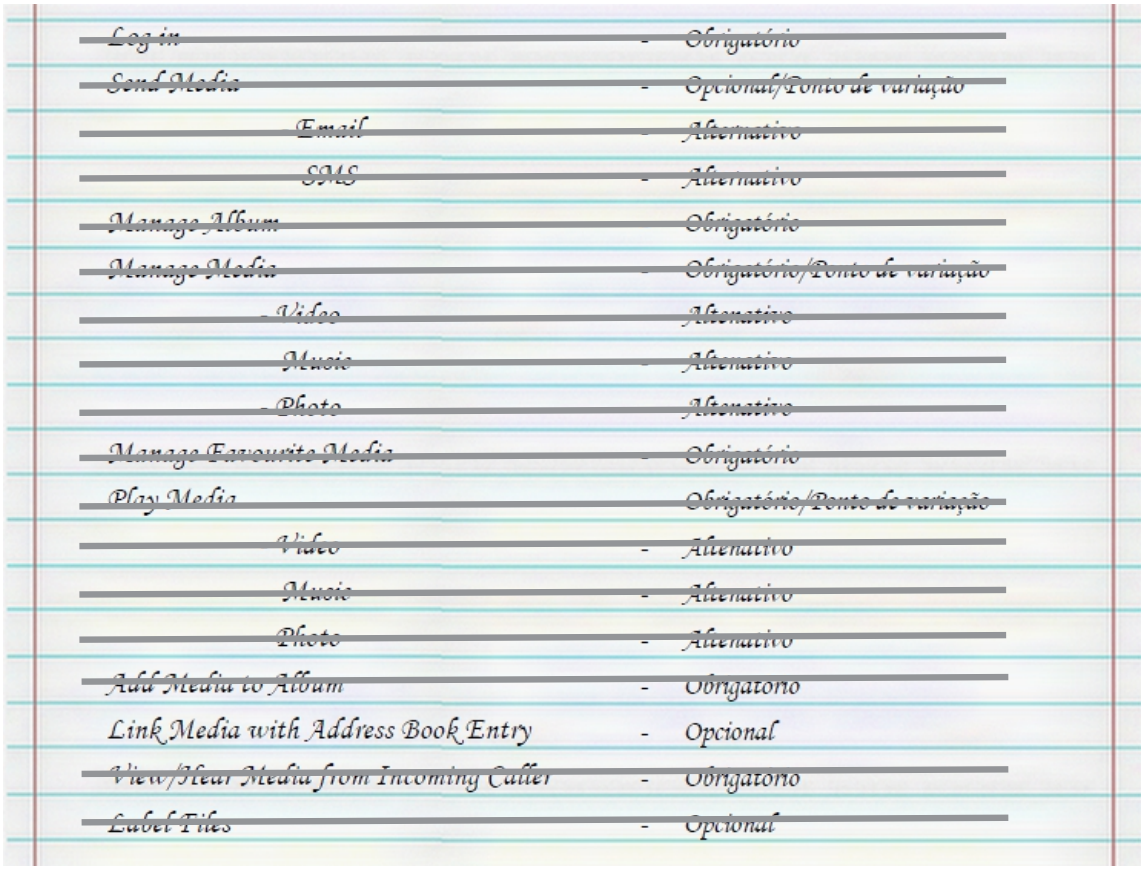
Tabela 5.2: FID após inspeção do diagrama de caso de uso da Figura 5.6 pelo ERD

nº	DIAGRAMA					Nº QUESTÃO	ELEMENTO	DEFEITO IDENTIFICADO
	FT	UC	CL	CP	SQ			
1		X				1.2	Destroy Media	Este caso de uso não foi definido na especificação de requisitos.
2		X				1.2	Send Media via Streaming	Este caso de uso não foi definido na especificação de requisitos.
3		X				1.7.1	Send Media	Não há relação entre os caso de uso Send Media e Hear Media from Incoming Caller.
4		X				1.7.1	Hear Media from Incoming Call	Não há relação entre os caso de uso HearMedia from Incoming Caller e V Photo.
5		X				1.3	Manage Album	O caso de uso está duplicado.
6		X				1.1	Manage Favourite Media and Musics	Este caso de uso não expressa corretamente sua funcionalidade, pois, é gerenciamento de todas as mídias,não havendo necessidade de colocar "Musics".
7		X				1.1	Pla Vdeo	O nome do caso de uso não expressa corretamente sua função.
8		X				1.2	Play 3D Video	Este caso de uso expressa uma função que não foi definida na especificação de requisitos. Não tem Video "3D".
9		X				1.1	V Photo	Este caso de uso não expressa corretamente sua funcionalidade
10		X				1.3	Log In	O caso de uso está duplicado.
11		X				2.2.2	Send Media	O caso de uso Send Media via SMS foi definido em variants e não está presente no diagrama de caso de uso.
12		X				2.2.2	Manage Media	Os casos de uso Manage Photo, Manage Video e Manage Music foram definidos em variants e não estão presentes no diagrama de caso de uso.
13		X				2.3	Link Media with Address Book Entry	O caso de uso que relaciona foto com registro na agenda não foi identificado no diagrama de caso de uso.

Fonte: o autor

O caso de uso `Link Media with Address Book Entry` não foi encontrado no diagrama em inspeção. No formulário FID (Tabela 5.2 - linha 13) foi reportado este defeito da seguinte forma:

Figura 5.10: Lista com as funcionalidades da MM após inspeção



Fonte: o autor

- Diagrama: caso de uso (UC);
- Nº questão: 2.3;
- Elemento: `Link Media with Address Book Entry`
- Defeito identificado: O caso de uso que relaciona foto com registro na agenda não foi identificado no diagrama de caso de uso.

Com a inspeção terminada e o FID preenchido (Tabela 5.2), o inspetor poderá passar para o próximo diagrama da técnica para esta perspectiva.

5.4 Exemplo #3 - Arquiteto de Domínio (AQD)

A instrução principal para o diagrama de componente para a perspectiva de AQQ orienta o leitor a fazer uma lista a partir do diagrama de classe dos componentes e interfaces da

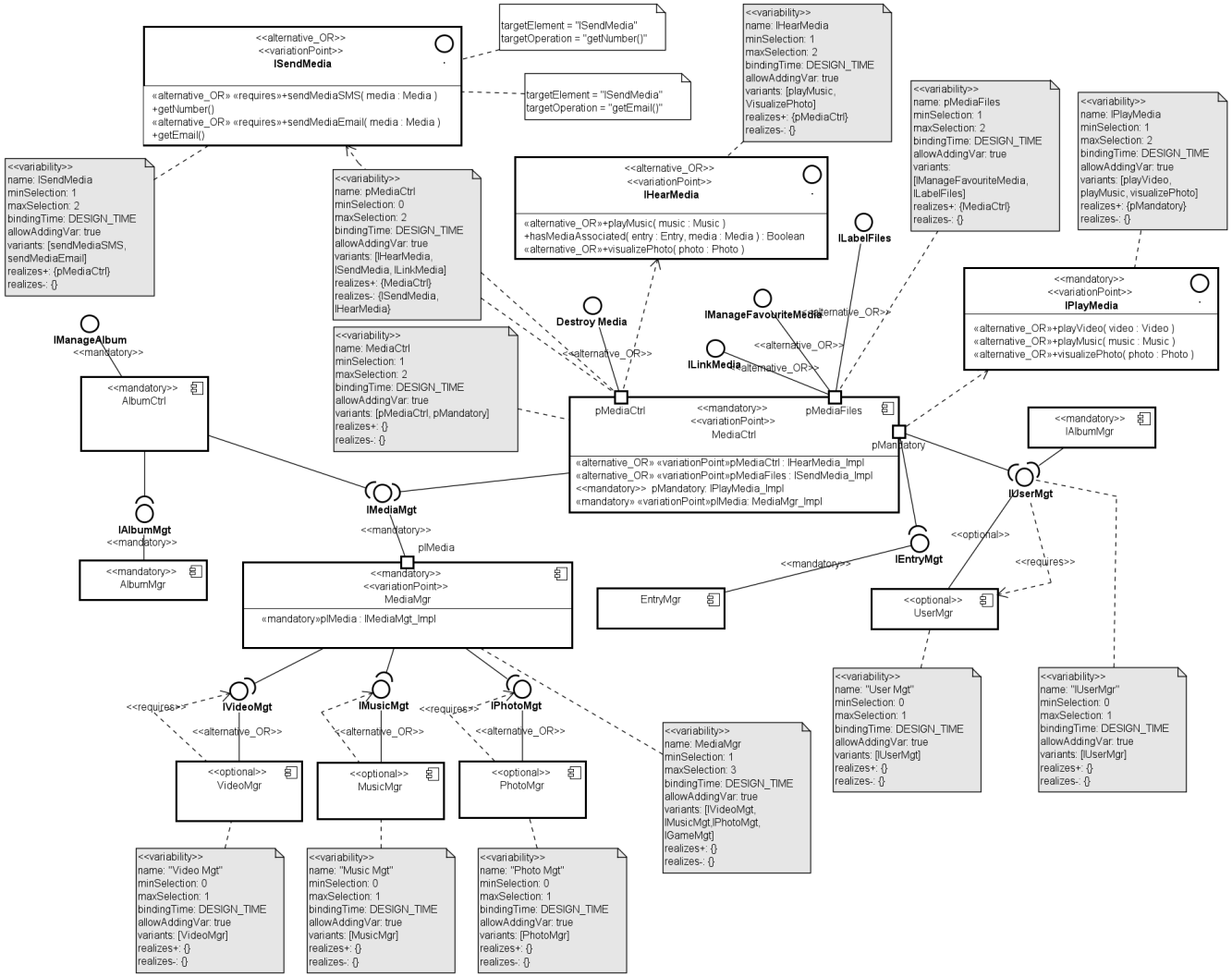
LPS. Depois, o inspetor deve seguir os passos definidos para este diagrama e perspectiva. Como já dito anteriormente, a leitura do diagrama de componente para o AQD deve ser feita em 2 passos (Passo 3 e 4): o primeiro percorre todos os elementos (componentes ou interfaces) e o segundo verifica especificidades do gerenciamento de variabilidades e de elementos descritos no formato de classificador.

Para cada elemento do diagrama, o inspetor deverá ler todas as questões do passo em que ele está, para só depois, prosseguir para o próximo elemento, salvo, quando a questão orientar que o leitor pode parar a inspeção e seguir para o próximo elemento, ou, que sejam questões que não se encaixam ao elemento. Como por exemplo, um subgrupo de questões para elementos opcionais e o elemento em inspeção for obrigatório

A inspeção do diagrama de componente para AQD é iniciada no Passo 3. Ele deverá ser concluído para todos os elementos do diagrama (Figura 5.11), seja ele do tipo componente ou interface, para só então, prosseguir no cenário (Passo 4), pois, o próximo passo pode precisar de informações que já tenham sido analisadas anteriormente.

A questão 4.3 orienta o leitor a detectar omissão de elementos ao verificar os que não foram ainda riscados da lista de componente e interfaces (Figura 5.12) e por consequência, não foram descritos no diagrama.

Figura 5.11: Diagrama de componente da LPS MM baseado na *SMarty* - Com Defeitos



Fonte: adaptado de Bera *et al.* (2015)

Figura 5.12: Lista com as classes e interfaces da MM

AlbumCtrl	-	mandatory
IManageAlbum	-	mandatory
IaddMediaAlbum	-	mandatory
AlbuMgr	-	mandatory
IAlbumMgt	-	mandatory
IMediaMgt	-	mandatory
MediaMgr	-	mandatory, variationPoint
MusicMgr	-	optional
IMusicMgr	-	alternative_or
PhotoMgr	-	optional
IPhotoMgt	-	alternative_or
VideoMgr	-	optional
IVideoMgt	-	alternative_or
EntryMgr	-	mandatory
IEntryMgt	-	mandatory
MediaCtrl	-	mandatory
ILinkMedia	-	mandatory
IPlayMedia	-	mandatory
IHearMedia	-	mandatory
IManageMedia	-	mandatory
ILabelFiles	-	mandatory
SendMgr	-	optional
ISendMedia	-	optional
FavouriteMgr	-	optional
IManageFavouriteMedia	-	optional
UserMgr	-	mandatory
IUserMgt	-	mandatory

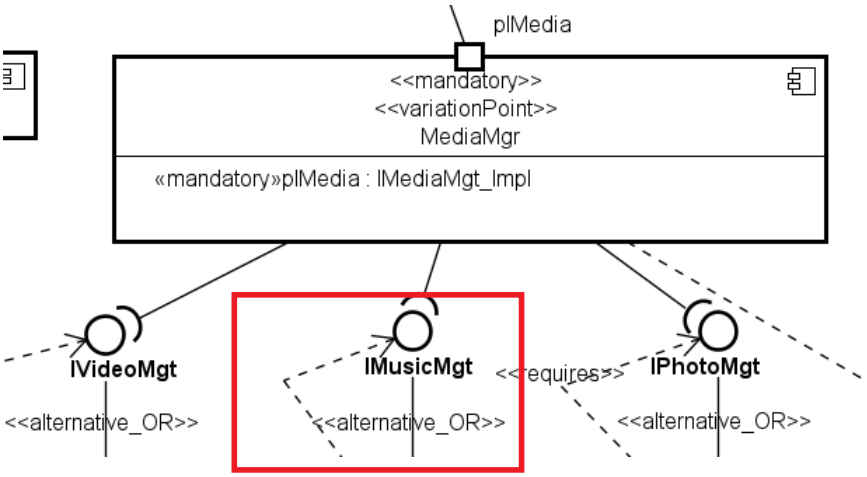
Fonte: o autor

O leitor é livre para iniciar a inspeção pelo elemento que preferir. Para este exemplo, a inspeção foi iniciada a partir da interface `IManageAlbum`, porém, após seguir todo o procedimento do Passo 3, ou seja, ler todas as questões deste passo, não foram encontradas

nenhuma inconsistência entre o diagrama de componente e de classe, pois, para todas as questões a resposta foi negativa, ou seja, não foram encontrados defeitos no elemento.

`MusicMgr` foi o próximo elemento inspecionado para o exemplo. O leitor ao se deparar com a questão 3.6.3 do Passo 3, após ter passado pelas questões anteriores, foi indagado se o elemento que requer outro foi estereotipado com `«requires»`. Ao analisar o diagrama, o componente `MusicMgr` só estará no diagrama, se a interface `IMusicMgt` também estiver. Sendo assim, há a necessidade de estereotipar a relação com `«requires»`, o que não foi feito no diagrama (Figura 5.13).

Figura 5.13: Defeito no componente `MusicMgr` - omissão de estereótipo



Fonte: adaptado de Bera *et al.* (2015)

Ao encontrar um defeito, o inspetor deverá preencher as informações no FID corretamente para que mais tarde o elemento seja encontrado e o diagrama seja corrigido. O defeito encontrado em `Music Mgr` foi descrito no FID na primeira linha (Tabela 5.3) com as seguintes informações:

- Diagrama: componente (CP);
- Nº questão: 3.6.3
- Elemento: `Music Mgr`;
- Defeito identificado: A relação entre `MusicMgr` requer a interface `IMusigMgt`, e este relacionamento não foi estereotipado com `«requires»`

Continuando a inspeção, o leitor irá identificar em `Destroy Media` o próximo defeito para o exemplo, pela leitura da questão 3.3, que analisa se o elemento em inspeção está presente na lista feita pelo inspetor, porém, ao analisarmos a Figura 5.11, nenhum dos itens da lista tem correspondência para esta funcionalidade.

O defeito encontrado deverá então ser preenchido no formulário FID da seguinte forma (Tabela 5.3 - linha 2):

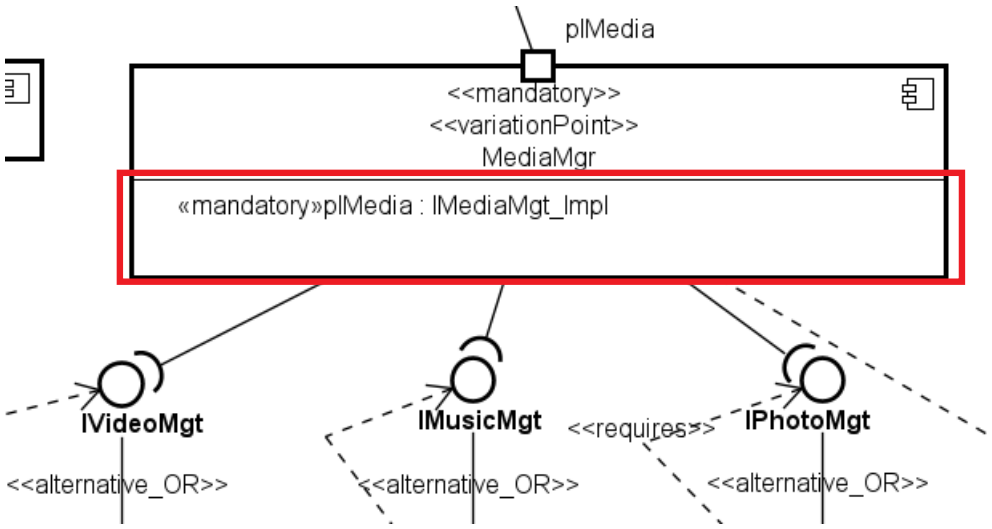
- Diagrama: componente (CP);
- Nº questão: 3.3;
- Elemento: `Destroy Media`;
- Defeito identificado: `Destroy Media` não é uma funcionalidade do sistema, logo não há uma interface para ela.

Após a visita de todos os elementos no diagrama de componente, o Passo 3 da técnica foi finalizado. O Passo 4 é composto de dois subgrupos e uma questão sozinha. O inspetor deve analisar primeiramente as representações de variabilidade na nota UML relacionada aos elementos e no subgrupo 4.2 componentes e interfaces descritas no formato de classificador.

Um tipo de defeito específico para o diagrama de componente é a falta da descrição das variantes de um elemento que representa um ponto de variação e que foi descrito no formato de classificador, como é orientado pela diretriz CP6 da abordagem *SMarty* (Apêndice B). Este tipo de defeito é encontrado no exemplo da Figura 5.11 no elemento `MediaMgr`.

No componente `MediaMgr` não foi especificado nas operações os elementos alternativos para o ponto de variação, que neste caso, inclui: `IPhotoMgt, IVideoMgt e IMusicMgt`. Por ser o mesmo tipo de defeito (mesma questão) para o mesmo elemento, pode ser descrita a omissão das 3 interfaces em uma única linha do formulário (Figura 5.14).

Figura 5.14: Defeito no componente `MediaMgr` - omissão de estereótipo



Fonte: adaptado de Bera *et al.* (2015)

- Diagrama: componente (CP);
- Nº questão: 4.2.2;
- Elemento:`MediaMgr`;
- Defeito identificado: Não foi definido no compartimento do elemento as interfaces alternativas: `IPhotoMgt, IVideoMgt, IMusicMgt`.

A última questão, 4.3, é analisada após inspeção dos subgrupos 4.1 e 4.2, que orientou o leitor a identificar os estereótipos das variantes dos elementos opcionais e dos pontos de variação, como também sua descrição no compartimento do classificador. O leitor deverá nesta questão olhar para a lista com as funcionalidades descritas e verificar se algum dos itens não foi representado por nenhum elemento do diagrama de componente, caracterizado assim, como um defeito de omissão.

Para este exemplo, 2 elementos não foram definidos no diagrama de componente. Logo, deve ser anotado no formulário FID a omissão destes itens de acordo com a questão 4.3. Para o primeiro item da lista (IAddMediaAlbum), foi reportado este defeito no formulário FID (Tabela 5.3 - linha 11) da seguinte forma:

- Diagrama: componente (CP);

- Nº questão: 4.3;
- Elemento: `IAddMediaAlbum`;
- Defeito identificado: Não foi definida uma interface para adicionar mídia ao álbum.

Tabela 5.3: FID após inspeção do diagrama de componente Figura 5.11 pelo GAD

nº	DIAGRAMA					Nº QUESTÃO	ELEMENTO	DEFEITO IDENTIFICADO
	FT	UC	CL	CP	SQ			
1				X		3.6.3	MusicMgr	A relação entre MusicMgr requer a interface IMusigMgt, e este relacionamento não foi estereotipado com <<requires>>
2				X		3.3	Destroy Media	Destroy Media não é uma funcionalidade do sistema, logo não há uma interface para ela.
3				X		3.6.1	ILinkMedia	A interface ILinkMedia não está relacionada com a porta de pMediaFiles, mas, sim, com pMediaCtrl.
4				X		3.2	IAlbumMgr	O componente está duplicado com outro já definido no diagrama.
5				X		3.4	EntryMgr	O estereótipo do componente não foi definido no diagrama.
6				X		4.1.1	MediaMgr	IGameMgt definido em variants na notação de variabilidade não é uma variante para o componente.
7				X		4.1.4	IPlayMedia	O componente tem 3 variantes, mas, maxSelection foi definido como sendo 2.
8				X		4.2.2	MediaMgr	Não foi definido no compartimento do elemento as interfaces alternativas: IPhotoMgt, IVideoMgt, IMusicMgt
9				X		4.2.3	MediaCtrl	Foi definido no compartimento do elemento que a porta plMedia é uma porta para MediaCtrl e não há esta relação entre os elementos.
10				X		4.3	IAddMediaAlbum	Não foi definida uma interface para adicionar mídia ao álbum.
11				X		4.3	IManageMedia	Não foi definida uma interface para gerenciamento de mídia.

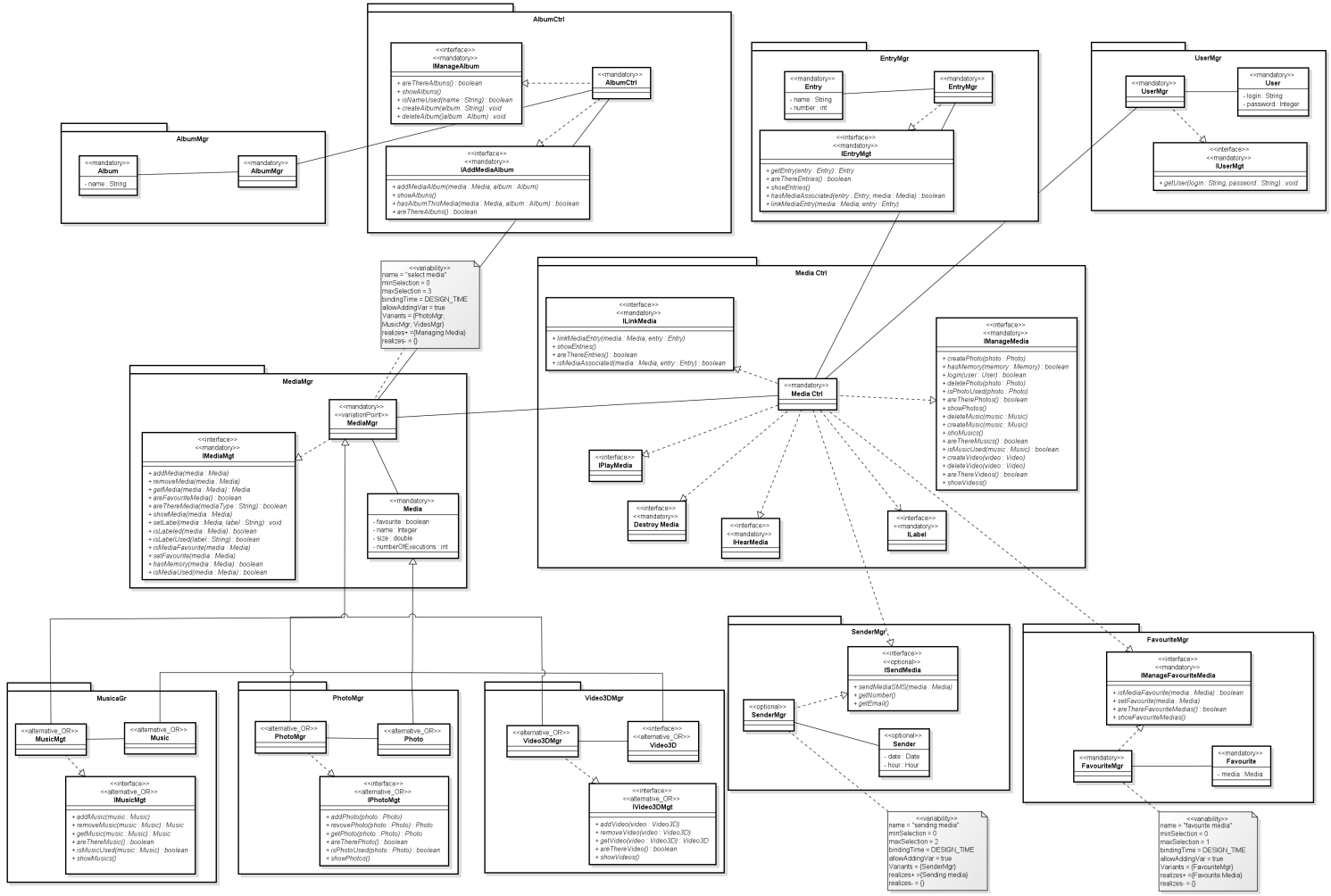
Fonte: o autor

Após a questão 4.3 e preenchimento do FID (Tabela 5.3), a inspeção para o AQD estará terminada para o diagrama de componente com o total de 11 defeitos identificados no diagrama da Tabela 5.1. O cenário visto neste exemplo pode ser executado também pela perspectiva de Desenvolvedor de Domínio, pois, para o diagrama de componente esta parte dos cenários de ambas as perspectivas é igual.

5.5 Exemplo #4 - Desenvolvedor de Domínio (DSD)

O cenário para o DSD específico para o diagrama e classe é composto de 2 passos: no primeiro são inspecionadas cada uma das classes e suas relações, verificando inclusive os estereótipos. Já o segundo passo verifica os meta-atributos de `«variability»` na nota UML associada e as classes faltantes no diagrama.

Figura 5.15: Diagrama de classe da LPS MM baseado na *SMarty* - Com defeitos



Fonte: adaptado de Nepomuceno *et al.* (2020)

Com a lista com as classes e interfaces candidatadas ao diagrama feita sob orientação da instrução principal para o diagrama de classe da *SMartyPerspective*, a próxima instrução para o diagrama de classe orienta o leitor a analisar cada um dos elementos do diagrama de classe para verificar alguns atributos de qualidade.

Neste exemplo, será inspecionado o diagrama de classe da LPS MM da Figura 5.15. A inspeção foi iniciada pela classe de controle `MediaMgr` (Figura 5.15).

Entre a questão 1.1 à 1.4 do cenário de DSD para o diagrama de classe, foram analisadas para `MediaMgr` se seu nome expressa corretamente os objetos que a compõe, se está redundante ou não pertence ao domínio da LPS e se está estereotipada. Para todas essas informações a resposta foi "não". Logo, até o momento não foram encontrados defeitos para esta classe.

As questões do grupo 1.5 verificam informações do pacote em que a classe está. Para este exemplo, a classe `MediaMgr` está dentro de um pacote que não foi nomeado, logo, pela questão 1.5.1 foi identificado um defeito no diagrama, pelo pacote não expressar corretamente o agrupamento. Este defeito será preenchido no formulário FID da seguinte forma (Tabela 5.4):

- Diagrama: classe (CL);
- Nº questão: 1.5.1;
- Elemento: `MediaMgr`;
- Defeito identificado: O pacote que o elemento está inserido não foi nomeado e por isso, não expressa corretamente o agrupamento.

Seguindo as questões até a 1.11 não será encontrado mais nenhum defeito para este elemento, sendo assim, a inspeção para ele estará finalizada pelo Passo 1. A inspeção seguirá para os elementos dos pacotes `MusicaGr e PhotoMgr` (Figura 5.15) e nenhum defeito será identificado para as classes e interfaces deste pacotes.

Quando o leitor verificar a questão 1.2 para a classe `Video3D` ele irá identificar comparando a classe com a lista das possíveis classes do sistema feita anteriormente, que não há uma a funcionalidade de vídeo 3D para esta LPS. O mesmo acontecerá para `Video3DMgr e IVideo3DMgt`. Será então informado este tipo de erro no formulário FID da seguinte forma:

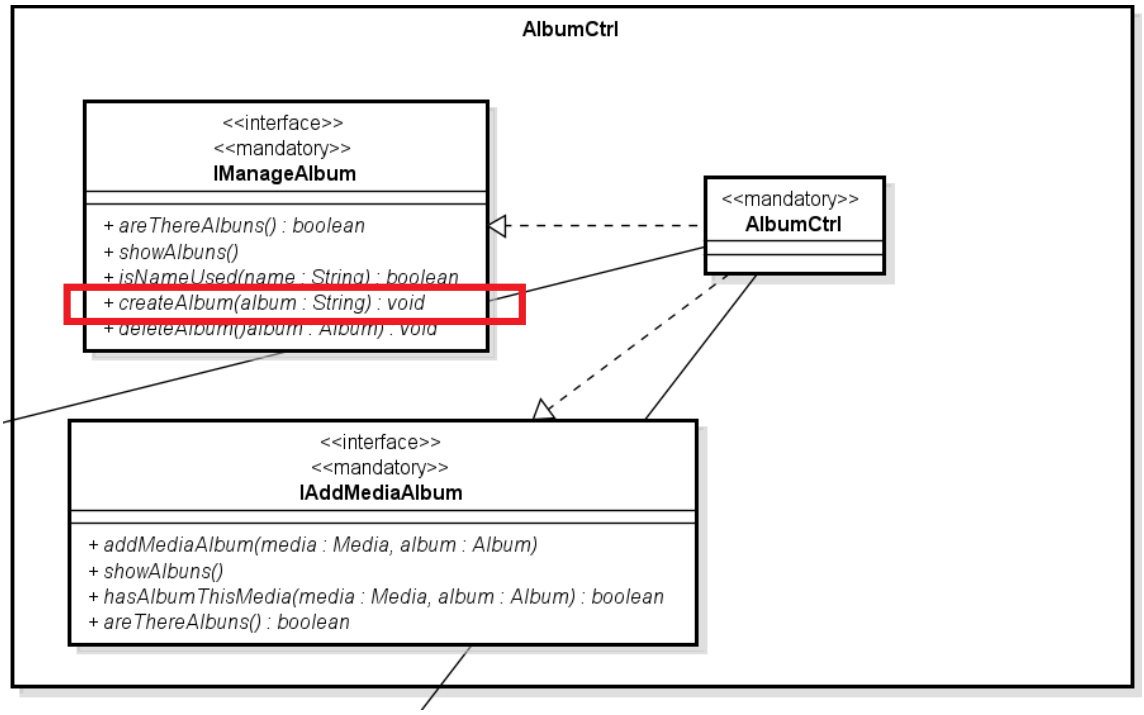
- Diagrama: classe (CL);
- Nº questão: 1.2;

- Elemento: `Video3D`;
- Defeito identificado: Não foi definido para o diagrama de classe, uma classe para controle de vídeos 3D, já que vídeo 3D não faz parte das funcionalidades do sistema.

Como os elementos em que foram encontrados os defeitos por meio da questão 1.2 não fazem parte do diagrama, então, não há a necessidade de continuar a inspeção para eles conforme é orientado pela própria questão. Mais alguns elementos do diagrama de classes serão analisados e detectados defeitos. Para cada um deles, pode ser consultado a tabela FID para verificação.

Durante a inspeção da interface `IManageAlbum`, não foram identificados defeitos para ela até a questão 1.9 do cenário. Porém, ao entrar no grupo 1.10 que é específico para interfaces, foi identificado na questão 1.10.4 que no método `createAlbum`, o parâmetro de entrada `album` foi definido como sendo `String`, quando na verdade é do tipo `Album` (Figura 5.16). Este defeito, poderia fazer com que o desenvolvedor implementasse este método de forma errada. Por isso, deve ser corrigido e é específico para a perspectiva de implementação.

Figura 5.16: Defeito na classe `IManageAlbum` - Tipo do parâmetro de entrada errado



Fonte: adaptado de Nepomuceno *et al.* (2020)

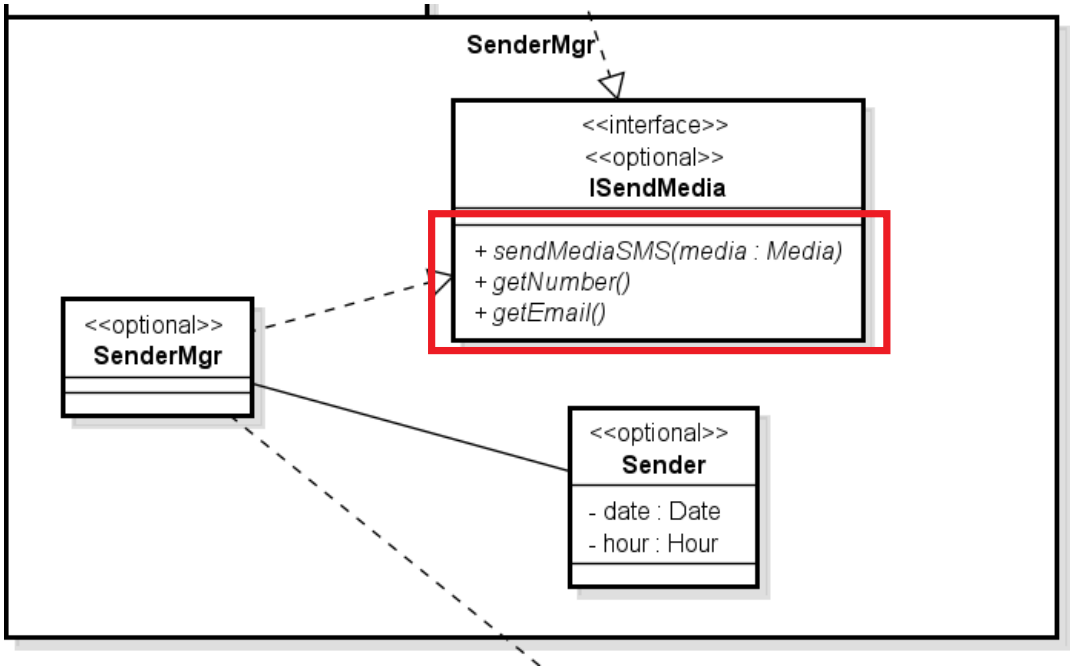
No FID o defeito será informado da seguinte forma (Tabela 5.4):

- Diagrama: classe (CL);
- Nº questão: 1.10.4;
- Elemento: `IManageAlbum`;
- Defeito identificado: o tipo do parâmetro de entrada para o método createAlbum (Album) foi definido incorretamente. Deveria ser `Album`, mas, foi definido como `String`.

A inspeção continuará e mais alguns defeitos serão identificados para este exemplo, como na classe `User`, em que o atributo `password` foi definido como `Integer`, quando deveria ser do tipo `String`, entre outros defeitos.

A LPS permite o envio de mídia via SMS e via *email*. Porém, ao analisarmos as questões referentes a interface `ISendMedia`, será verificado que não foi definido um método para o envio de mídia via email para a LPS. Esta omissão, será identificada por meio da questão 1.11 que questiona o leitor se todos os métodos importantes foram definidos para a interface.

Figura 5.17: Defeito na interface `ISendMedia` - Omissão



Fonte: adaptado de Nepomuceno *et al.* (2020)

Como este método não foi definido, será então informado no FID o defeito encontrado (Tabela 5.4):

- Diagrama: classe (CL);
- Nº questão: 1.10.4;
- Elemento: `ISendMedia`;
- Defeito identificado: não foi definido um método para envio de mídia via email no diagrama inspecionado.

Durante a inspeção das demais classes e interfaces do diagrama da Figura 5.15 serão identificados mais alguns defeitos até que todos os elementos do diagrama tenham sido inspecionados. Assim, o leitor poderá passar para o Passo 2.

No Passo 2 serão analisadas as notas UML que representam uma variabilidade e seus meta-atributos, como também, se houve omissão de algum elemento no diagrama que não tenha sido encontrado ainda.

Há 3 notações de variabilidade no diagrama utilizado como exemplo nesta seção: `select media, sending media e favourite media`. Iniciando a inspeção por `select media`, será identificado pela questão 2.2.4 que o meta-atributo `minSelection` foi definido com valor 0, quando na verdade deveria ser 1. Pois, este ponto de variação, há três variantes «alternative_OR», e de acordo com a abordagem *SMarty*, deve ser selecionado pelo menos uma variante:

- Diagrama: classe (CL);
- Nº questão: 1.10.4;
- Elemento:`MediaMgr`;
- Defeito identificado: o meta-atributo da variabilidade relacionada ao elemento `MediaMgr` foi definido como sendo 0, quando na verdade deveria ser `minSelection=1`;

A inspeção para o diagrama de classes continuará até que todos as variabilidades tenham sido analisadas e os elementos omissos tenham sido identificados. O FID final para este exemplo pode ser visualizado na Tabela 5.4.

Tabela 5.4: FID após inspeção do diagrama de classe Figura 5.15 pelo DSD

nº	DIAGRAMA					Nº QUESTÃO	ELEMENTO	DEFEITO IDENTIFICADO
	FT	UC	CL	CP	SQ			
1			X			1.5.1	MediaMgr	O pacote que o elemento está inserido não foi nomeado e por isso, não expressa corretamente o agrupamento.
2			X			1.2	Video3D	Não foi definido para o diagrama de classe, uma classe para controle de Videos 3D, já que vídeo 3D não faz parte das funcionalidades do sistema.
3			X			1.2	Video3DMgr	Não foi definido para o diagrama de classe, uma classe para controle de Videos 3D, já que vídeo 3D não faz parte das funcionalidades do sistema.
4			X			1.2	IVideo3DMgt	Não foi definido para o diagrama de classe, uma interface para controle de Videos 3D, já que vídeo 3D não faz parte das funcionalidades do sistema.
5			X			1.10.4	IManageAlbum	O tipo do parâmetro de entrada para o método createAlbum(Album) foi definido incorretamente. Deveria ser Album, mas, foi definido como String.
6			X			1.8.4	User	A senha do usuário foi definida como sendo Integer, quando na verdade deveria ser do tipo String.
7			X			1.2	Destroy Media	Não há uma interface para destruir mídia, pois, não é uma funcionalidade da LPS Mobile Media.
8			X			1.1	ILabel	O nome da interface não a reflete corretamente.
9			X			1.6.3	IManageFavouriteMedia	O estereótipo deste elemento está incorreto. Está como mandatory, mas, deveria ser optional.
10			X			1.6.3	FavouriteMgr	O estereótipo deste elemento está incorreto. Está como mandatory, mas, deveria ser optional.
11			X			1.6.3	Favourite	O estereótipo deste elemento está incorreto. Está como mandatory, mas, deveria ser optional.
12			X			1.11	ISendMedia	Não foi definida uma função para o o envio de mídia por email.
13			X			2.2.4	MediaMgr	o meta-atributo da variabilidade relacionada ao elemento MediaMgr foi definido como sendo 0, quando na verdade deveria ser minSelection=1
14			X			2.2.4	ISendMedia	o meta-atributo da variabilidade relacionada ao elemento ISendMedia foi definido como sendo 2, quando na verdade deveria ser maxSelection=1
15			X			2.3	AlbumMgr	Não foi definido uma interface para o gerenciamento do álbum.
16			X			2.3	Video	Não foi definido as classes e interfaces para gerenciamento de vídeos.

Fonte: o autor

5.5.1 Considerações sobre o uso de diagramas incorretos

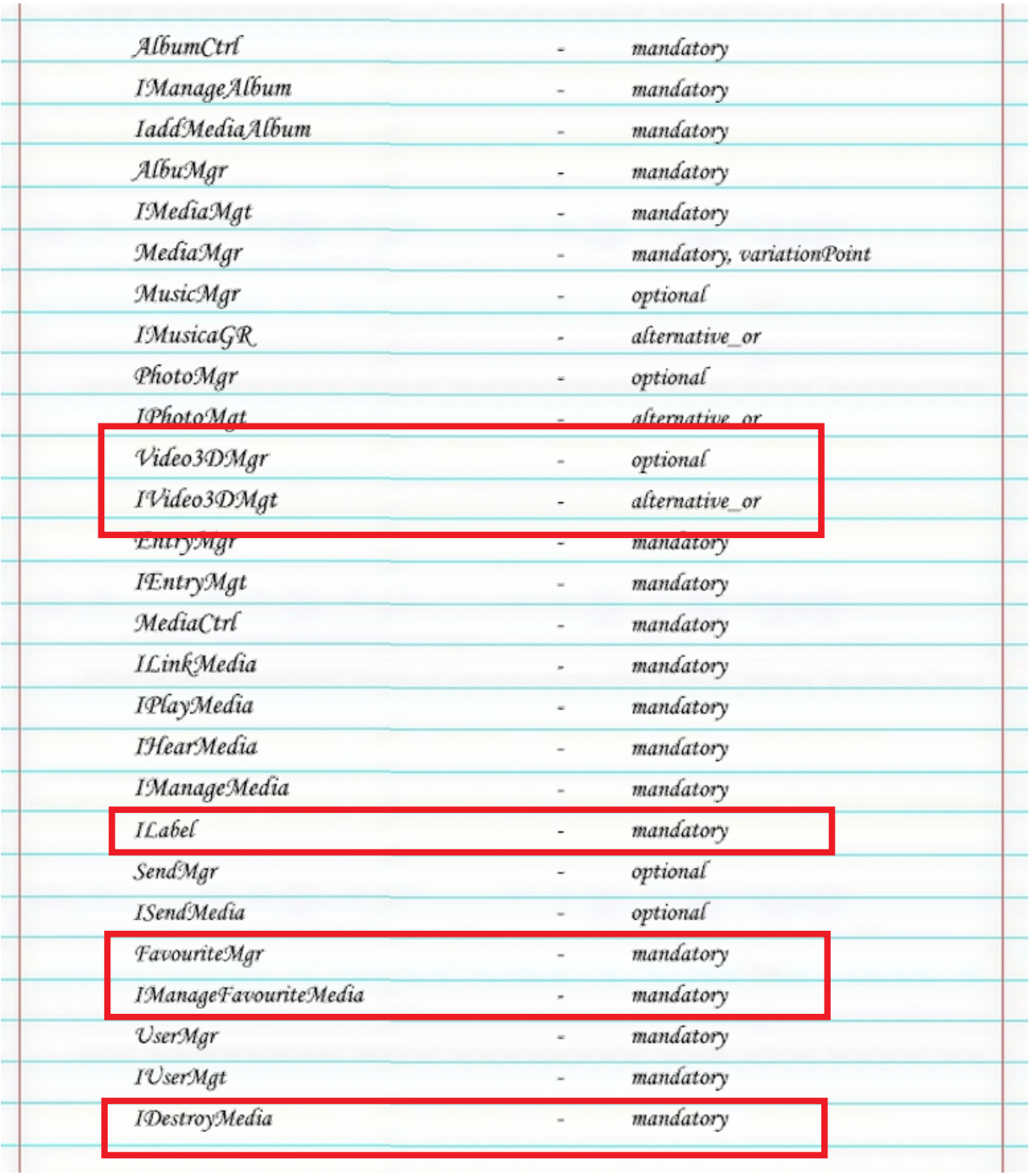
A instrução principal do diagrama de componente tanto para a perspectiva de AQD quanto de DSD orienta o leitor a fazer uma lista a partir do diagrama de classes, para verificar quais os componentes e interfaces que devem ser definidos no diagrama de componente.

Se o leitor separar o diagrama de classe com defeito que ainda não tenha sido corrigido, ele poderá definir uma lista incorreta para ajuda-lo na inspeção. Dessa forma, ao seguir a lista ele sentirá falta ou irá adicionar elementos que não fazem parte do domínio da LPS ou com informações incorretas.

Tendo como exemplo o diagrama de classe com defeito da Figura 5.15 para inspecionar o diagrama de componente da Figura Figura 5.11, o leitor ao ler a instrução principal irá

criar sua lista, como mostra a Figura Figura 5.18. Nela, é possível observar que foram definidas classes, interfaces, estereótipos valores para meta-atributos incorretos.

Figura 5.18: Lista com as classes e interfaces da LPS MM a partir de um diagrama com defeitos



Fonte: o autor

Entre as informações incorretas repassadas pelo diagrama de classe estão: foi definida classe e interface para `Video3D`, sendo que o domínio da LPS não compreende vídeos 3D, estereótipos incorretos para as classes do pacote `FavouriteMgr`, que neste exemplo foram definidos como obrigatórios, quando na verdade são opcionais. Além de faltarem classes e interfaces para alguns elementos como a variante vídeo para a classe Media.

Estes defeitos poderão fazer com que a arquitetura da LPS, apresentada por meio do diagrama de componente, seja definida incorretamente para todos os produtos da LPS, além de serem implementados com erros ou gerarem configurações incorretas na Engenharia da Aplicação. Como o caso da troca do estereótipo opcional para obrigatório. Isso faria com que todas as aplicações configuradas desta LPS obrigatoriamente tenham que ter a funcionalidade de definir a mídia favorita.

Além de problemas da arquitetura de LPS e de implementação, o artefato usado como base para inspeção com defeito irá também trazer erros para a gerência de versões que na hora de comparar os elementos entre as versões dos diagramas irá considerar como defeito elementos que estão corretos de acordo com o domínio, mas, que foram considerados errados no diagrama de classe usado como base, como o da Figura 5.15.

5.6 Exemplo #5 - Gerente de Ativos de Domínio (GAD)

O GAD deve garantir que a última versão do diagrama está correto na base de ativos para que seja utilizado por outros papéis ou para que sejam derivadas outras LPS. Portanto, para este papel, há a necessidade de ter duas versões de um mesmo diagrama: a primeira versão (oráculo) irá servir de comparação para o identificar os defeitos na segunda versão do diagrama.

Neste exemplo foi considerado o diagrama de caso de uso da Figura 5.6 como a segunda versão e a segunda versão (o oráculo diagrama correto para a LPS - (sem defeitos)) retirado do trabalho de Geraldi *et al.* (2015), como mostra a Figura 5.19.

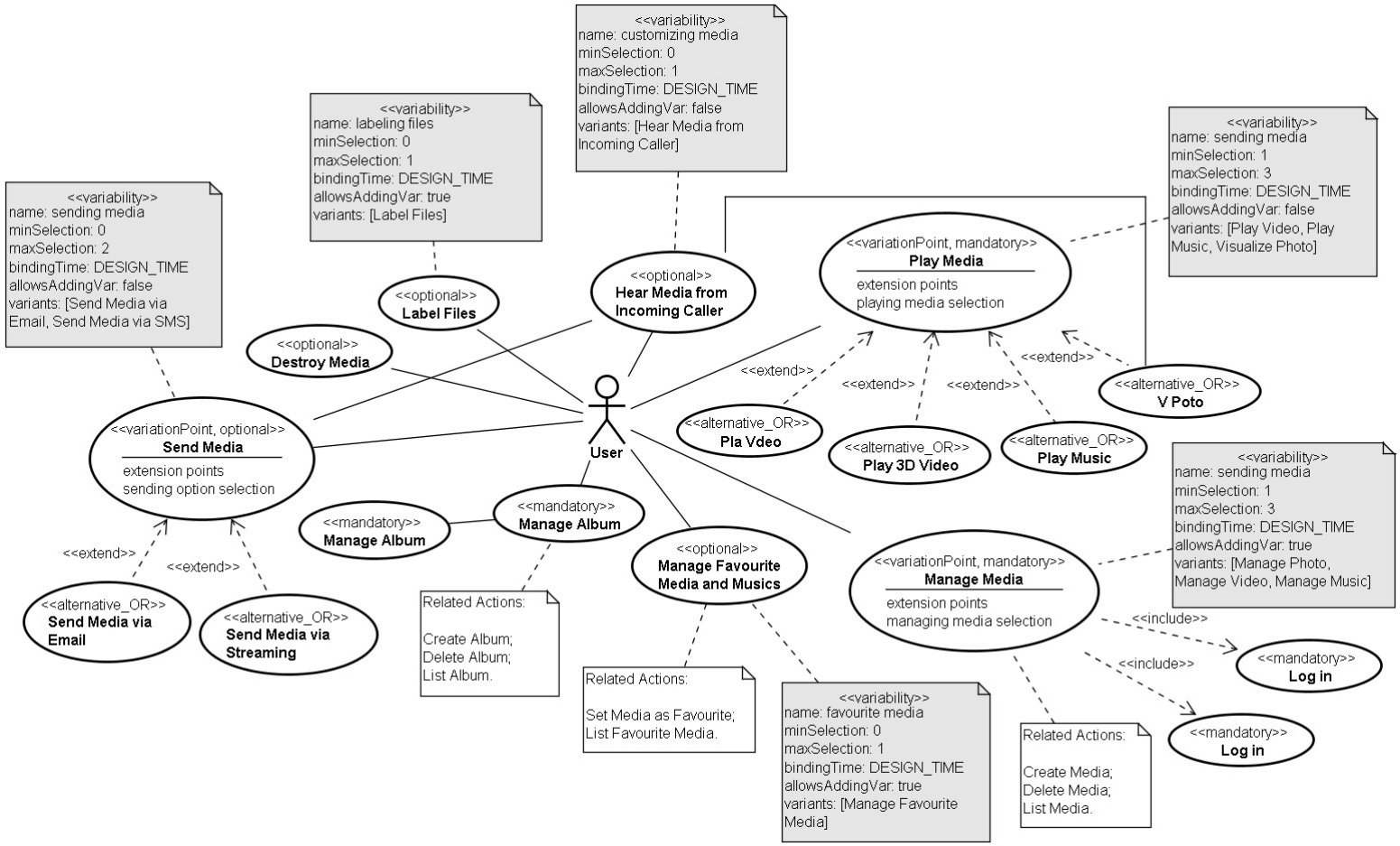
A instrução inicial da técnica para o cenário de GAD orienta o leitor a localizar os dois diagramas que serão usados para a inspeção (oráculo e com defeitos). Com o diagrama com defeito em mãos, ele servirá de base para identificação dos defeitos, pois, a cada elemento ali definido será buscado no diagrama oráculo para verificar se estão consistentes entre si.

No passo 1, o gerente de ativos deverá analisar um elemento por vez, incluindo suas relações/fluxos/mensagens dependendo do diagrama que está sendo inspecionado.

Neste segundo exemplo, assim como no primeiro, a inspeção foi iniciada pelo caso de uso `LabelFiles` e não foi encontrado nenhum defeito para este elemento.

Como no primeiro exemplo, foi identificado pelo GAD que `Destroy Media` não é um caso de uso para o diagrama desta LPS. A perspectiva de ERD teve que retirar essa informação da lista que ele fez a partir da especificação dos casos de uso, verificando que nela não havia nenhuma função compatível com este caso de uso. Já a perspectiva de GAD, faz uma comparação do diagrama com defeitos (versão 2) com o diagrama oráculo (versão 1) para verificar que não há este caso de uso.

Figura 5.19: Diagrama de caso de uso da LPS MM baseado na *SMarty*



Fonte: (Geraldi e OliveiraJr, 2017b)

No formulário FID no campo "Defeito Identificado" será descrito de maneira diferente do que no ERD, pois, aqui, será informado que não foi possível rastrear o elemento, como é mostrado na Tabela 5.5 (linha 1):

- Diagrama: caso de uso (UC);
- Nº questão: 1.1;
- Elemento: `Destroy Media`;
- Defeito identificado: Não é possível rastrear esse elemento para a versão original.

O processo seguirá para todos os elementos do diagrama de caso de uso com defeitos, para então passar para o Passo 2 em que são identificados defeitos específicos relacionados a notação UML que representa a variabilidade.

Para esta inspeção, o exemplo de um tipo de defeito característico do Passo 2 foi encontrado para o caso de uso `Send Media`. No diagrama com defeitos, o meta-atributo `allowAddingVar` (permissão para incluir novas variantes) na especificação da variabilidade foi definido como `true` e no oráculo `como false`.

No FID, apresentado na Tabela 5.5 (linha 11), este defeito foi preenchido da seguinte forma:

- Diagrama: caso de uso;
- Nº questão: 2.6;
- Elemento: `Send Media`
- Defeito identificado: O valor para o meta-atributo `allowAddingVar` difere da versão original. Está definido como `false`, quando deveria ser `true`.

Com todas as variabilidades analisadas, o Passo 3 consiste em verificar no oráculo quais os elementos que não foram identificados no diagrama com defeitos (e não foram removidos da LPS por meio de alguma atualização descrita no gerenciamento de versões).

Para este exemplo, o caso de uso do gerenciamento de vídeos, `Manage Video`, não foi encontrado no diagrama com defeitos, sendo então preenchido no formulário FID (linha 15 - Tabela 5.5) da seguinte forma:

- Diagrama: caso de uso (UC);
- Nº questão: 3.2;

- Elemento: `Manage Video`
- Defeito identificado: O elemento não está presente na versão inspecionada.

Com a inspeção terminada e o formulário preenchido (Tabela 5.5), o próximo passo do cenário do GAD é investigar os meta-atributos `realizes+` e `realizes-` para identificar se é possível encontrar os elementos definidos nestes conjuntos no diagrama especificado. No diagrama tido como exemplo para esta seção os meta-atributos `realizes+` e `realizes-` não foram definidos para nenhum elemento, logo, a inspeção é finalizada no Passo 3.

Tabela 5.5: FID após inspeção do diagrama de caso de uso Figura 5.6 pelo GAD

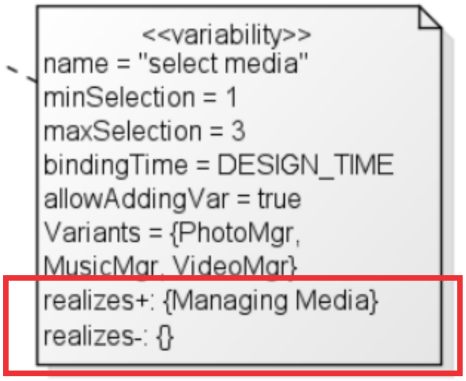
nº	DIAGRAMA					N DA QUESTÃO	ELEMENTO	DEFEITO IDENTIFICADO
	FT	UC	CL	CP	SQ			
1		X				1.1	Destroy Media	Não é possível rastrear esse elemento para a versão original.
2		X				1.5.1	Send Media	Não há relação entre os casos de uso Send Media e Hear Media from Incoming Caller.
3		X				1.1	Send Media via Streaming	Não é possível rastrear esse elemento para a versão original.
4		X				1.5.1	Hear Media from Incoming Call	Não há relação entre os casos de uso HearMedia from Incoming Caller e V Photo.
5		X				1.1	Manage Album	Não é possível rastrear esse elemento para a versão original (duplicado).
6		X				1.2	Manage Favourite Media and Musics	O nome na segunda versão não está de acordo com a primeira.
7		X				1.1	Pla Vdeo	Não é possível rastrear esse elemento para a versão original.
8		X				1.2	Play 3D Video	O nome na segunda versão não está de acordo com a primeira.
9		X				1.2	V Photo	O nome na segunda versão não está de acordo com a primeira.
10		X				1.1	Log In	Não é possível rastrear esse elemento para a versão original (duplicado).
11		X				2.6	Send Media	O valor do meta-atributo allowsAddingVar difere da versão original. Está definido como false, quando deveria ser true.
12		X				2.2	PlayMedia	O nome da variabilidade já foi definido por outra variabilidade.
13		X				2.2	ManageMedia	O nome da variabilidade já foi definido por outra variabilidade.
14		X				3.1	Link Media with Address Book Entry	O elemento não está presente na versão inspecionada.
15		X				3.1	ManageVideo	O elemento não está presente na versão inspecionada.
16		X				3.1	ManageMusic	O elemento não está presente na versão inspecionada.
17		X				3.1	ManagePhoto	O elemento não está presente na versão inspecionada.

Fonte: o autor

5.7 Exemplo #6 - Rastreabilidade entre os diagramas de classe e caso de uso

No Passo 4 do cenário da perspectiva de GAD é analisada a rastreabilidade entre elementos dos diagramas. Para tal, são verificados os conjuntos dos meta-atributos `realizes+` e `realizes-` que definem a coleção de variabilidades de modelos de nível inferior e superior, respectivamente, que realizam a variabilidade (Figura 5.20).

Figura 5.20: meta-atributo realizes+ e realizes-



Fonte: adaptado de Nepomuceno *et al.* (2020)

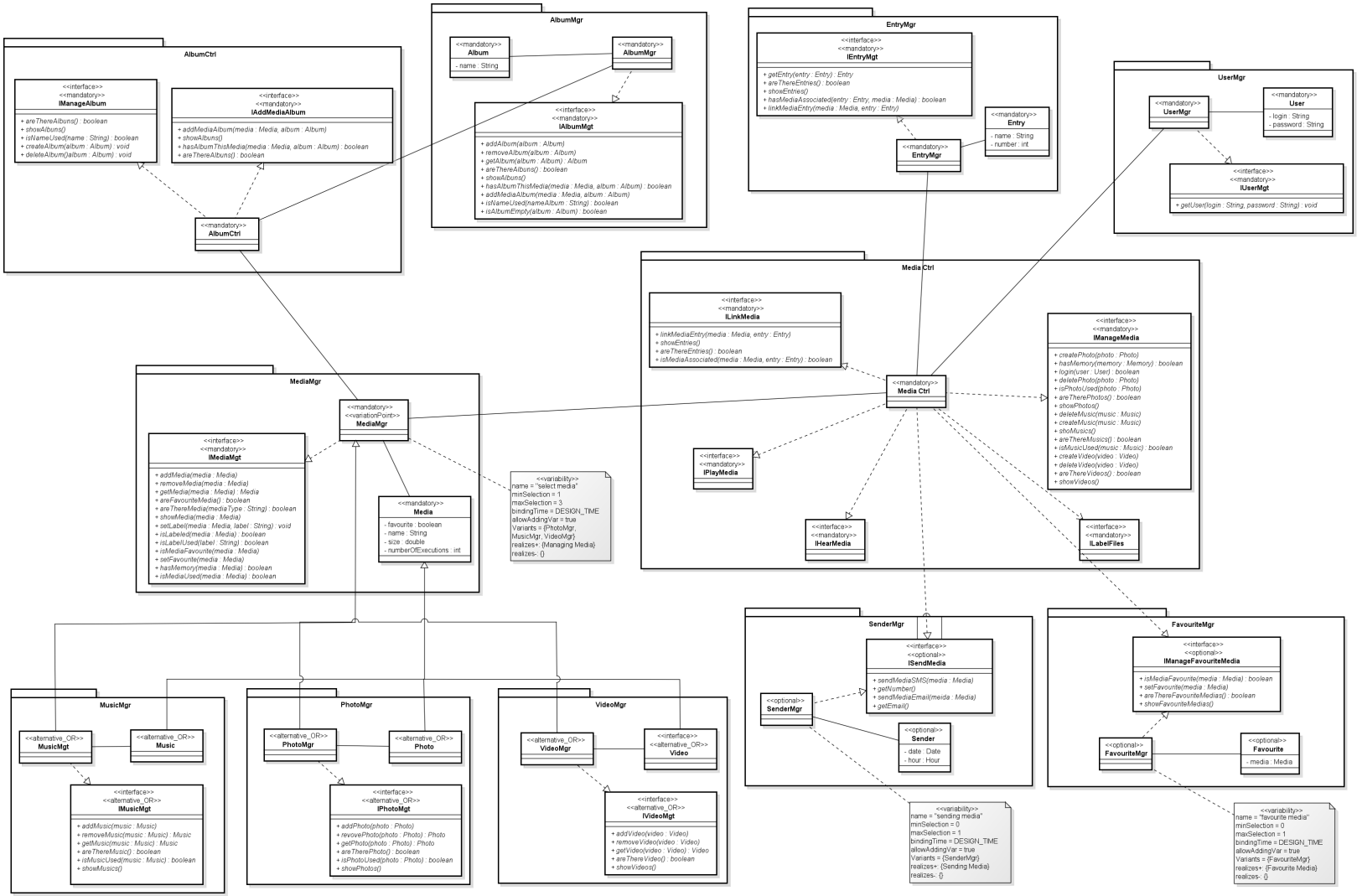
Antes de começar a análise, deve ser verificado quais diagramas são rastreáveis. Para este exemplo, foi definido para o diagrama de classe que o conjunto `realizes+` contém elementos rastreáveis para o diagrama de caso de uso e realizes- para o diagrama de componente.

Para este exemplo foi considerado o diagrama de classe da Figura 5.21 da LPS Mobile Media e o diagrama de caso de uso com defeito da Figura 5.6. Portanto, o GAD irá inspecionar diagrama de classe quanto a rastreabilidade em relação a versão mais recente do diagrama de caso de uso.

No Passo 4 o GAD irá verificar todos as notações UML que representam a variabilidade analisando os meta-atributos `realizes+` e `realizes-` conforme instrução. Para o exemplo, o diagrama de classe possui 3 notações de variabilidade: `select media, sending media e favourite media`.

Iniciando a inspeção para a variabilidade `select media` da classe `MediaMgr` o conjunto `realizes+` contém a variabilidade `managing media`. Quando o GAD for verificar o diagrama de caso de uso da Figura 5.6 , ele não irá encontrar a variabilidade, não sendo possível então realizar o rastreamento entre os diagramas.

Figura 5.21: Diagrama de classe da LPS MM baseado na *SMarty*



Será informado no FID este defeito da seguinte forma (Tabela 5.6):

- Diagrama: classe (CL);
- Nº questão: 4.1;
- Elemento: `MediaMgr`;
- Defeito identificado: Não foi encontrado no diagrama de caso de uso a variabilidade definida em `realizes+` no diagrama de classe.

A variabilidade `favourite media` para gerenciamento de mídias foi encontrada no diagrama de caso de uso e ela realiza a variabilidade `favourite media` da classe `FavouriteMgr` conforme especificado no diagrama de classe.

Se o inspetor já tivesse realizado a inspeção para o diagrama de caso de uso conforme o exemplo anterior na Tabela 1, ele teria verificado que há redundância entre os nomes das variabilidades, alterando-os conforme necessário. Assim, ele teria evitado defeitos de rastreabilidade entre elementos.

Considerando que o foco do GAD é na rastreabilidade entre os elementos e iniciou a inspeção pelo Passo 4, ele irá verificar que para a variabilidade associada da classe opcional `Sender Mgr`, o conjunto `realizes+` contém a variabilidade `sending media`. O inspetor irá encontrar no diagrama de caso de uso da Figura 5.6 três variabilidades com o este nome. Porém, duas delas são associadas aos casos de uso `Manage Media e Play Media`. Portanto, seguindo o contexto da LPS, estes elementos não são rastreáveis e há um defeito no diagrama que deverá ser identificado no formulário FID (Tabela 5.6) para cada um deles:

- Diagrama: classe (CL);
- Nº questão: 4.2;
- Elemento: `ManageMedia`;
- Defeito identificado: A variabilidade associada ao caso de uso `Manage Media` não realiza a variabilidade do diagrama de classe `sending media`.

Na questão 4.3 o GAD é orientado a analisar se faltou algum elemento nos conjuntos `realizes+` e `realizes-`. Como três das variabilidades foram nomeadas igualmente, a variabilidade rastreável da classe `MediaMgr, managing media,` não foi encontrada no diagrama de caso de uso. Logo, ele irá analisar os dois diagramas e verificar que a

variabilidade associada ao caso de uso `Manage Media` é rastreável para a variabilidade da classe `MediaMgr`.

O FID (Tabela 5.6) será preenchido da seguinte forma:

- Diagrama: classe (CL);
- Nº questão: 4.3;
- Elemento: `MediaMgr`;
- Defeito identificado: Faltou no conjunto `realizes+` a variabilidade do diagrama de caso de uso relacionada a caso de uso `Manage Media`, nomeada por enquanto como ‘‘`sending media`’’.

Tabela 5.6: FID após inspeção do diagrama de classe Figura 5.6 pelo GAD

nº	DIAGRAMA					Nº QUESTÃO	ELEMENTO	DEFEITO IDENTIFICADO
	FT	UC	CL	CP	SQ			
1			X			4.1	MediaMgr	Não foi encontrado no diagrama de caso de uso a variabilidade definida em realizes+ no diagrama de classe.
2			X			4.3	MediaMgr	Faltou no conjunto realizes+ a variabilidade do diagrama de caso de uso relacionada a caso de uso ManageMedia, nomeada por enquanto como “sending media”.
3			X			4.2	ManageMedia	A variabilidade associada ao caso de uso ManageMedia não realiza a variabilidade do diagrama de classe sending media.
4			X			4.2	PlayMedia	A variabilidade associada ao caso de uso PlayMedia não realiza a variabilidade do diagrama de classe sending media.

Fonte: o autor

5.8 Considerações Finais

Neste capítulo foram apresentados seis exemplos de defeitos para as perspectivas da *SMartyPerspective*. Para cada um dos diagramas definidos para o exemplo foram incorporados alguns defeitos característicos dos diagramas de caso de uso, *features*, classe e componente para mostrar o funcionamento do procedimento da técnica e a forma de preencher o Formulário de Identificação de Defeito e como fazer as listas (para alguns cenários) para colaborar ativamente com a inspeção.

No Capítulo 6 será apresentado o estudo qualitativo da técnica *SMartyPerspective* para analisar a viabilidade do uso da técnica apresentada e caracterizada nos capítulos anteriores.

6

Estudo de Viabilidade da *SMartyPerspective*

6.1 Considerações Iniciais

Neste capítulo é apresentado o planejamento, execução, análise, interpretação dos resultados e ameaças a validade obtidos por meio do estudo qualitativo para verificar a viabilidade da técnica *SMartyPerspective* definida no Capítulo 4 para posterior refinamento da técnica.

O estudo foi conduzido *on-line* em fevereiro de 2021 por meio de formulários eletrônicos. Ele contou com a participação de 21 participantes, entre mestrandos, mestres, doutorandos e doutores das áreas acadêmica, industrial e estudantes.

Por meio da resposta dos participantes no Questionário de Caracterização, eles foram divididos entre as cinco perspectivas da *SMartyPerpective*: Gerente de Produto, Engenheiro de Requisitos de Domínio, Arquiteto de Domínio, Desenvolvedor de Domínio e Gerente de Ativos de Domínio. A confiabilidade das informações preenchidas pelos participantes neste questionário foi analisada por meio do coeficiente α de Cronbach.

Os participantes deveriam preencher o Formulário de Identificação de Defeitos (FID) durante a execução da tarefa de inspeção proposta no estudo. Destes dados, foi possível calcular evidências da eficiência, eficácia e efetividade da técnica *SMartyPerspective*. Esta análise aproveitou apenas os dados obtidos no estudo, portanto, não foram definidas hipóteses e comparação com outras técnicas.

Após a execução das tarefas pelos 21 participantes, as respostas obtidas no Questionário de *Feedback* foram analisadas quantitativamente considerando o número de

concordâncias para cada um dos níveis da escala *Likert* para cada uma das afirmações e dimensões do modelo TAM definidas para este estudo.

Além desta análise quantitativa, as questões abertas foram analisadas qualitativamente por meio do procedimento de *Grounded Theory*, como codificação, com definição dos códigos e categorias extraídas das citações obtidas da análise textual das respostas dados pelos participantes.

Das análises foram definidos tópicos que devem ser melhorados tanto para estudos futuros quanto para maior aceitação e eficiência da técnica *SMartyPerspective* para inspeção de diagramas de gerenciamento de variabilidades.

Este estudo qualitativo seguiu os princípios de *Open Science* para transparência, disseminação e disponibilidade dos dados por meio do compartilhamento de toda a instrumentação e resultados com *open access*[1].

6.2 Planejamento

O principal propósito deste estudo qualitativo é analisar a viabilidade da técnica *SMarty-Perspective* para os cenários de Gerente de Produto, Engenheiro de Requisitos de Domínio, Arquiteto de Domínio, Desenvolvedor de Domínio e Gerente de Ativos de Domínio para inspeção de diagramas de gerenciamento de variabilidade na Engenharia de Domínio.

Assim, espera-se com este estudo responder à questão: "Os cenários da técnica *SMartyPerspective* são viáveis para a detecção de defeitos em diagramas UML *SMarty* e diagrama de *features*?"

Deste modo, o objetivo do estudo pode ser descrito por meio da abordagem de Basili *et al.* (1994):

Analisar a técnica *SMartyPerspective*

Com o propósito de caracterizar a sua viabilidade

Em relação à detecção de defeitos em diagramas UML *SMarty* de caso de uso, classe, componente e sequência e diagrama de *features*.

Do ponto de vista de inspetores de software

No contexto de profissionais do setor de software, professores de ensino superior e estudantes de mestrado e doutorado com experiência em LPS.

[1] https://doi.org/10.5281/zenodo.4574618

6.2.1 Seleção de Contexto

Segundo Travassos *et al.* (2002), o contexto da pesquisa pode ser caracterizado sob quatro dimensões: processo (*on-line/off-line*), participantes (estudantes/ profissionais), realidade (o problema real/modelado) e generalidade (específico/geral). Para este estudo, temos:

- O processo: O processo será *off-line*, pois, não será em um ambiente controlado pelo pesquisador, visto que os questionários serão enviados *on-line* para os participantes responderem no momento e local mais propício a ele;
- Os participantes: Os participantes convidados são participantes na área de Engenharia de Software;
- A realidade: A LPS Arcade Game Maker (AGM) que será inspecionada pelos participantes é um problema modelado para fins acadêmicos, ou seja, não é de um caso real da indústria de software;
- Generalidade: O propósito do experimento é específico para inspeção de diagramas UML; e *SMarty* de casos de uso, classes, componentes, sequência e diagrama de *features* para gerenciamento de variabilidades.

6.2.2 Seleção de Participantes

A seleção dos participantes não se deu de forma aleatória, mas sim, por conveniência.

Foram convidados para este estudo participantes em Engenharia de Software graduados em Ciência da Computação ou cursos relacionados e que são estudantes da área (mestrandos ou doutorandos), professores ou atuantes na área industrial.

Para cada um dos profissionais previamente selecionados foi enviado um *email* convidando-os para a avaliação qualitativa da *SMartyPerspective*.

6.2.3 Instrumentação

Por meio de questões inerentes à saúde pública impostas pela COVID 19, o treinamento e o estudo não puderam ser realizados presencialmente. Portanto, a instrumentação para o estudo da viabilidade da técnica *SMartyPerspective* foi disponibilizada de forma *on-line* para os participantes voluntários.

A coleta dos dados foi realizada por meio de dois formulários eletrônicos no *Google Forms*. Foi utilizada esta plataforma para os documentos que deveriam ser preenchidos

pelos usuários, bem como um link para download da parte teórica e para acesso ao vídeo de treinamento e de *upload* do FID preenchido.

A instrumentação do estudo pode ser dividida em dois conjuntos: i) documentos que devem ser preenchidos pelo participante assim que aceitarem participar do estudo; e ii) documentos referentes ao treinamento e avaliação da técnica.

Convite

Para cada um dos participantes previamente selecionados foi enviado um *email* com a apresentação do estudo e um convite para que participassem com o *link* para acesso ao formulário eletrônico do *Google Forms* com os dois primeiros documentos que devem ser preenchidos, caso aceitassem colaborar. Os documentos foram descritos no formulário eletrônico para serem preenchidas neste meio.

Os documentos deste primeiro formulário são:

- **Documento 1 - Termo de Consentimento Livre e Esclarecido (TCLE)**: este termo contém as principais informações sobre o estudo a ser aplicado como, por exemplo: confidencialidade, procedimentos e benefícios. Este documento permitiu que o participante tomasse sua decisão sobre a sua participação na pesquisa de forma justa e sem constrangimentos. Além de proteger o pesquisador de que o sujeito concordou com a sua participação;

- **Documento 2 - Questionário de Caracterização**: o questionário foi aplicado aos participantes para analisar o nível de conhecimento e experiência sobre UML, inspeção de software, LPS e papéis em Engenharia de Domínio. Esse documento foi enviado antes do treinamento, pois, havia uma questão referente às funções que o participante acreditava poder atuar em uma organização que desenvolve LPS. As respostas destas questões foram necessárias para a divisão dos blocos do estudo. Os participantes foram divididos por meio dessas respostas em cinco grupos referentes a cada uma das perspectivas da *SMartyPerspective*.

Treinamento e Estudo

Após o aceite dos participantes em realizar o estudo por meio do preenchimento do TCLE, eles receberam um link do *Google Forms* com o formulário eletrônico correspondente à perspectiva que lhe foi atribuída. Pelo formulário o participante teve acesso ao link com os documentos de 3 a 6. O documento 7 foi descrito como questões para serem preenchidas no próprio formulário.

- **Documento 3 - Documento com Síntese Teórica**: serviu como base para a fase de treinamento dos participantes, pois contém um resumo com o conteúdo importante para que o participante consiga compreender o contexto e o uso da técnica. A fim de organização, este documento foi dividido em duas seções:
  - conceitos sintetizados de LPS para então descrever a abordagem *SMarty* e os estereótipos do perfil *SMartyProfile*, importantes para a detecção de defeitos específicos de gerenciamento de variabilidades;
  - descrição geral da técnica *SMartyPerspective* com um resumo sobre inspeção de software.
- **Documento 4 - Exemplo da Perspectiva**: este documento contém um exemplo específico para o papel definido para o participante. É um documento diferente para cada um dos papéis, por causa das especificidades de cada um dos cenários da *SMartyPerspective*. Portanto, contém cinco exemplos diferentes de acordo com as seguintes perspectivas: Gerente de Produto, Engenheiro de Requisitos de Domínio, Arquiteto de Domínio, Desenvolvedor de Domínio e Gerente de Ativos de Domínio;
- **Documento 5 - Cenário da Perspectiva**: contém o cenário completo correspondente à perspectiva atribuída ao participante. Logo, assim como no Documento 4, há cinco versões diferentes para cada perspectiva. Este documento além de ser utilizado para o treinamento da técnica, colabora para orientar o participante a detectar os defeitos nos diagramas que lhe foram atribuídos;
- **Documento 6 - Formulário de Identificação de Defeitos (FID)**: o formulário de identificação de defeitos é parte integrante da técnica *SMartyPerspective*. Nele são preenchidos os defeitos encontrados pelo participante, o item da pergunta que o orientou e o elemento em que foi identificado. O seu modelo servirá tanto para o treinamento, quanto para a realização das tarefas. O participante realiza o *download* deste documento para preencher durante a inspeção dos diagramas do estudo. Existe no formulário um link para que o documento após preenchido seja entregue aos pesquisadores por meio do seu *upload*;
- **Documento 7 - Documento de Requisitos**: este documento se refere à descrição geral da LPS Arcade Game inspecionada pelos participantes. O documento de requisitos é composto da especificação de requisitos e um diagrama UML *SMarty* de caso de uso, classe, componente, sequência e diagrama de *features* com os defeitos incorporados nestes diagramas pelos pesquisadores. Para a perspectiva de Gerente

de Domínio, o documento de requisitos incluiu duas versões dos diagramas para serem comparados durante a inspeção: a primeira (oráculo) sem defeitos e que foi usada como base para a segunda versão, com defeitos incorporados;

- **Documento 8 - Questionário de *feedback***: foi disponibilizado em uma seção do formulário que contém as perguntas para serem respondidas pelos participantes referente às suas impressões durante o uso da técnica *SMartyPerspective* e dos elementos dos cenários para verificar sua viabilidade perante à inspeção dos diagramas que a técnica suporta. O questionário é formado por 15 questões fechadas baseadas no modelo TAM e 9 questões abertas.

Como o estudo teve que ser *on-line* foi enviado para os participantes, além dos documentos para entendimento da técnica, três vídeos de treinamento: (i) um com os conceitos principais referentes ao Documento 3; (ii) um com a apresentação da técnica *SMartyPerspective* também referente ao Documento 3; e (iii) um com a execução do exemplo referente à perspectiva atribuída ao participante.

A Tabela 6.1 apresenta o *link* para os vídeos do treinamento e da pasta com os documentos da instrumentação. O vídeo sobre os conceitos principais (LPS e inspeção) e apresentação da técnica *SMartyPerspective* é o mesmo para todos as perspectivas. Já o vídeo com a execução do exemplo do Documento 4 e o link com os documentos da instrumentação são específicos para cada uma das perspectiva.

Tabela 6.1: Link para os documentos e vídeos dos treinamento e tarefas

GERAL PARA TODAS AS PERSPECTIVAS			
	LINK VÍDEOS		TEMPO
LPS e inspeção	`https://youtu.be/00dtaJugqqE`		12:08 min
SMartyPerspective	`https://youtu.be/w14GhrrhXw8`		11:35 min
ESPECÍFICOS PARA AS PERSPECTIVAS			
	LINK DOCUMENTOS	LINK VÍDEOS	TEMPO
Gerente de Produto	`https://encurtador.com.br/ryGPS`	`https://youtu.be/nmeJQ0wYzq4`	12:55 min
Engenheiro de Requisitos de Domínio	`https://encurtador.com.br/fipG4`	`https://youtu.be/SCPPtxfVfrM`	12:54 min
Arquiteto de Domínio	`https://encurtador.com.br/rCGO3`	`https://youtu.be/_6XZvrhzrHA`	11:23 min
Desenvolvedor de Domínio	`https://encurtador.com.br/chrGQ`	`https://youtu.be/OtbedZgN5uY`	11:36 min
Gerente de Ativos de Domínio	`https://encurtador.com.br/cikqs`	`https://youtu.be/WkqnfVVxqyg`	13:28 min

Fonte: o autor

6.2.4 *Design* do Estudo

O estudo foi dividido em duas etapas:

1. os participantes foram convidados a participarem do estudo e preencher o termo de aceite e o Questionário de Caracterização do estudo. Foi estimado em torno de 5 minutos para tal; e

2. os participantes foram divididos em cinco blocos com base nas perspectivas da técnica, onde foram ministrados o treinamento e a execução das tarefas pelos participantes. O tempo de treinamento para cada perspectiva diverge de acordo com a Tabela 6.1. Os primeiros vídeos que juntos somam 23m:43s é comum para cada participante, porém o exemplo pode variar de 11m:23s a 12m:55s dependendo da perspectiva. Além do exemplo, o tempo de de execução da tarefa também varia de acordo com a perspectiva, de acordo com o número de diagramas relacionados à ela.

Considerando os diferentes cenários operacionais da técnica *SMartyPerspective*, os participantes não tiveram o mesmo treinamento. Para cada cenário foi ministrado um treinamento diferente na seção de exemplo da técnica, visto que cada um possui diagramas específicos para a sua função.

Aleatoriedade

O universo restrito de participantes em LPS e inspeção, fez com que a seleção dos participantes não fosse aleatória, mas, por conveniência.

Com a distribuição dos Documentos de Requisitos com diagramas com defeitos diferentes, a escolha do cenário para cada um dos participantes não foi aleatória, pois, tais participantes foram balanceados considerado as suas respostas no Questionário de Caracterização.

Separação em Blocos

Os participantes foram divididos de acordo com o perfil em cinco blocos, um para cada cenário da *SMartyPerspective*: Gerente de Produto, Engenheiro de Requisitos de Domínio, Arquiteto de Domínio, Desenvolvedor de Domínio e Gerente de Ativos de Domínio.

Balanceamento

As tarefas foram distribuídas de forma balanceada dentro de um mesmo cenário. Os participantes com a mesma perspectiva atribuída tiveram a mesma quantidade de tarefas. Porém, entre as perspectivas tal distribuição não foi igual, já que cada perspectiva teve uma quantidade diferente de diagramas a serem inspecionados.

6.3 Execução

Nesta seção é detalhada a ordem cronológica dos procedimentos realizados para participação dos participantes neste estudo, conforme segue:

1. o pesquisador envia um *email* convidando os participantes previamente selecionados com o *link* do *Google Forms* com duas seções: uma contendo os termos do estudo e a outra o questionário de caracterização;

2. o participante faz a leitura do TCLE (Documento 1) e, caso esteja de acordo, prossegue para o próximo passo;

3. o participante preenche o Questionário de Caracterização (Documento 2). Nele o participante informa seu conhecimento prévio sobre LPS, inspeção de software, diagramas suportados pela abordagem *SMartyPerspective* e tarefas em uma organização em LPS;

4. o participante envia as respostas para o pesquisador analisar;

5. o pesquisador seleciona entre as respostas do participante uma das perspectivas para o ele assumir durante o estudo;

6. o *link* do formulário *Google Forms* com a instrumentação relacionada à perspectiva atribuída ao participante é enviado. O formulário é dividido em três seções principais: informações do treinamento, tarefas a serem realizadas durante o estudo e avaliação da técnica;

7. o participante ao acessar o *link* realiza o treinamento da técnica. Para tal, acessa o *link* do *Google Drive* com os documentos necessários: Documento 3, Documento 4, Documento 5 e Documento 6;

8. o participante pode assistir aos vídeos ministrados pelo pesquisador como parte integrante do treinamento. Para cada perspectiva, o pesquisador deixa disponível ao participante o *link* de três vídeos referente aos conceitos principais de LPS e inspeção, a técnica *SMartyPerspective* e um exemplo de inspeção para a perspectiva;

9. ao participante é permitido a qualquer momento enviar um *email* para o pesquisador para solucionar dúvidas que não foram resolvidas durante o treinamento;

10. após o treinamento, o participante prossegue para a próxima seção do formulário para iniciar a execução das tarefas do estudo: inspecionar os diagramas da LPS AGM (Documento 7) referente à sua perspectiva utilizando o cenário (Documento 5) correspondente à perspectiva que lhe foi atribuída. Para tanto, deve baixar o FID (Documento 6) e preencher com os defeitos encontrados durante a inspeção;

11. ao finalizar a inspeção, o participante deve anotar no lugar indicado no formulário, o horário inicial e final da inspeção. Além disso, deve anexar o FID preenchido no *link* correspondente no formulário;

12. com o fim da inspeção, o participante prossegue para a próxima seção do formulário eletrônico para responder o grupo de questões abertas e fechadas relacionadas à sua impressão durante o uso da técnica *SMartyPerspective*; e

13. o participante envia o formulário para que o pesquisador analise as respostas e seu desempenho durante a inspeção.

6.3.1 Questionário de Caracterização

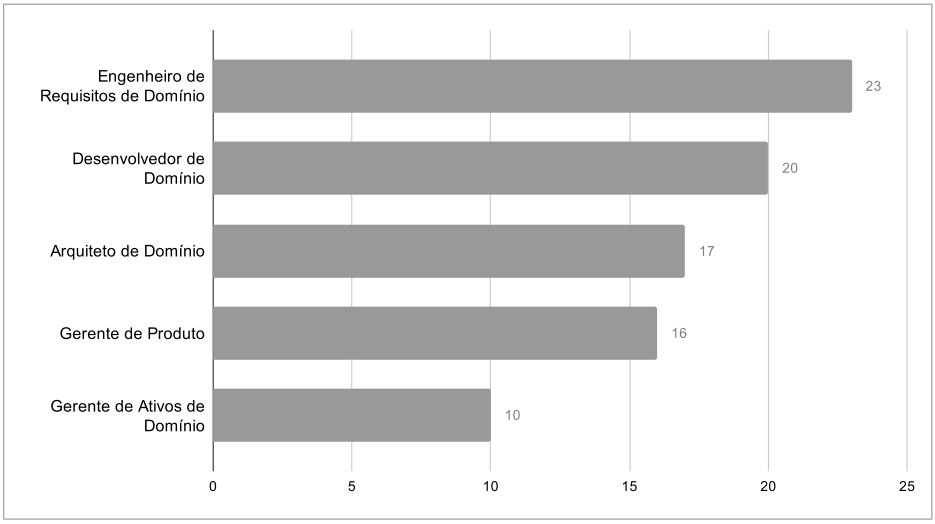
A primeira etapa do estudo consiste no aceite do participante em participar por meio do TCLE e preenchimento do Questionário de Caracterização para definir o nível de conhecimento do participante em alguns temas importantes para compreensão da técnica *SMartyPerspective*.

Dos participantes previamente selecionados e convidados por *email*, 32 deles aceitaram o convite para participar, respondendo assim, o Questionário de Caracterização. Este questionário foi importante para poder definir a divisão dos blocos durante o treinamento e execução das tarefas.

Em uma das questões, o participante foi indagado a selecionar entre as cinco perspectivas da *SMartyPerspective* quais ele se considera apto a exercer dentro de uma organização que desenvolve LPS: "*A SMartyPerspective é composta de 5 papéis da atividade de Engenharia de Domínio em Linha de Produto de Software. Com base na descrição dos papéis, marque TODAS as opções que você considera que seu conhecimento sobre LPS permitiria você exercer dentro de uma organização:*".

A Figura 6.1 apresenta o quantitativo das respostas dos participantes para a questão acima. Dos 32 participantes, 23 se sentiram aptos a exercer a função de Engenheiro de Requisitos de Domínio, enquanto para Gerente de Ativos de Domínio, apenas 10 deles.

Figura 6.1: Papéis que os participantes podem exercer na indústria



Fonte: o autor

Dos 32 participantes que aceitaram participar do estudo e responderam o Questionário de Caracterização apenas 21 deles realizaram as tarefas do estudo nos 15 dias que sua instrumentação ficou disponível para os participantes.

Por meio das respostas dadas pelos 21 participantes para a questão dos papéis que poderiam exercer e da sua experiência em relação aos diagramas de *features*, caso de uso, classe, componente e sequência, eles foram divididos em 5 blocos referentes as perspectivas da *SMartyPerspective*. Para os participantes que assinalaram apenas um dos papéis, não foi analisado sua experiência com os diagramas necessários para o papel, ficando assim, com a perspectiva escolhida por ele.

O ID dos participantes utilizado durante este trabalho é constituído do abreviação da perspectiva que lhe foi atribuída (Gerente de Produto (GRP), Engenheiro de Requisitos de Domínio (ERD), Arquiteto de Domínio (AQD), Desenvolvedor de Domínio (DSD) ou Gerente de Ativos de Domínio (GAD)) e um número que o diferencia dentro do conjunto de sua perspectiva (Tabela 6.2 e Tabela 6.3).

A Tabela 6.2 apresenta os dados informados pelos 21 participantes no Questionário de Caracterização dividas em três grupos principais: (i) Caracterização, que se refere às informações de nível de formação e setor de atuação; (ii) tempo de experiência, que é o tempo que o participante tem de experiência em Engenharia de Software (ES) e LPS; e (iii) Experiência, que se refere ao nível de experiência em LPS e gerenciamento de variabilidade (LPS-GV), a abordagem *SMarty* (SM) e inspeção (IP).

Tabela 6.2: Dados de caracterização e conhecimento prévio dos participantes

Nº	CARACTERIZAÇÃO			TEMPO		EXPERIÊNCIA		
	ID	**Formação**	**Setor**	**ES**	**LPS**	**LPS-GV**	**SM**	**IP**
1	GRP1	Mestrando	Acadêmico	1	1	2	2	2
2	GRP2	Doutor	Acadêmico	4	1	3	1	5
3	GRP3	Doutor	Acadêmico	5	5	5	3	4
4	GRP4	Doutor	Acadêmico	5	5	5	5	5
5	ERD1	Mestre	Industrial	3	2	5	4	2
6	ERD2	Mestre	Acadêmico	5	4	5	5	4
7	ERD3	Mestrando	Acadêmico	5	1	3	1	2
8	ERD4	Mestre	Industrial	2	2	4	4	3
9	AQD1	Mestrando	Estudante	2	2	4	4	2
10	AQD2	Mestrando	Estudante	2	2	4	3	2
11	AQD3	Mestre	Industrial	3	2	5	5	4
12	AQD4	Doutor	Estudante	3	2	5	4	4
13	DSD1	Mestre	Acadêmico	2	2	5	5	2
14	DSD2	Doutorando	Acadêmico	5	3	5	5	5
15	DSD3	Mestre	Industrial	2	1	3	1	2
16	DSD4	Mestre	Industrial	2	2	3	3	2
17	DSD5	Doutorando	Acadêmico e Industrial	5	2	5	4	3
18	GAD1	Doutorando	Acadêmico	2	1	4	1	2
19	GAD2	Doutorando	Acadêmico	3	1	3	3	4
20	GAD3	Mestrando	Acadêmico	3	1	4	4	3
21	GAD4	Mestre	Industrial	5	3	4	4	2
Moda				**2/5**	**2**	**4**	**5**	**2**

LEGENDA:
ES = Engenharia de Software / LPS = Linha de Produto de Software / SM = Abordagem SMarty
LPS-GV = LPS e Gerenciamento de Variabilidade / IP = Inspeção de Software

Tempo:
1 = menos de 2 anos / 2 = entre 2 e 4 anos / 3 = entre 5 e 7 anos / 4 = entre 8 e 10 anos
5 = acima de 10 anos

Experiência:
1 = nunca ouviu falar / 2 = conhece superficialmente / 3 = experiência básica
4 = experiência moderada / 5 = experiência avançada

Fonte:o autor

A Tabela 6.3 apresenta as respostas dos 21 participantes referente ao seu nível de experiência em relação aos diagramas UML de caso de uso, classe, componente e sequência e o diagrama de *features*.

O cálculo da moda para o tempo de experiência em ES, resultou em dois valores: 2 e 5 (Tabela 6.2). O que mostra os extremos em relação à experiência dos participantes: 7 deles tem mais que 10 anos de experiência e 7 tem apenas de 2 a 4 anos. Isso se deve a amostra ser bem diversificada entre mestrandos, mestres, doutorandos e doutores.

Pela Tabela 6.2 é possível observar que apesar da maioria dos participantes ter apenas de 2 a 4 anos de experiência com LPS, a maioria deles possuem um conhecimento moderado sobre os conceitos principais relacionados à LPS e gerenciamento de variabilidades,

Tabela 6.3: Dados referentes à experiência dos participantes com os diagramas

Nº	ID	DIAGRAMAS				
		features	**caso de uso**	**classe**	**componente**	**sequência**
1	GRP1	2	3	3	2	2
2	GRP2	2	3	3	3	3
3	GRP3	5	5	5	5	5
4	GRP4	5	5	5	5	5
5	ERD1	4	5	4	4	4
6	ERD2	4	5	5	5	4
7	ERD3	2	4	5	3	3
8	ERD4	4	4	4	3	4
9	AQD1	3	3	4	3	3
10	AQD2	1	3	4	3	4
11	AQD3	4	4	5	4	3
12	AQD4	5	5	5	5	5
13	DSD1	4	5	5	3	3
14	DSD2	5	5	5	5	5
15	DSD3	1	4	4	2	3
16	DSD4	1	4	4	4	4
17	DSD5	4	3	3	4	4
18	GAD1	3	4	4	4	4
19	GAD2	4	5	5	5	5
20	GAD3	3	4	4	4	4
21	GAD4	3	4	4	4	4
Moda		**4**	**4/5**	**4/5**	**4**	**4**

LEGENDA:
1 = conhece superficialmente / 2 = nunca modelou software com o diagrama
3 = experiência básica / 4 = experiência moderada / 5 = experiência avançada

Fonte: o autor

pontos de variação, variantes e os seus relacionamentos, resolução de variabilidades e tempos de resolução.

Quanto à experiência com inspeção de software (IP), a maioria dos participantes tem apenas um conhecimento superficial sobre o assunto. Esses dados são importantes para analisar se a *SMartyPerspective* pode ser utilizada independentemente da experiência do leitor em inspeção de software.

6.3.2 Questionário de *Feedback*

Após a inspeção dos diagramas relacionados à perspectiva que foi atribuída ao participante, ele deveria responder o Questionário de *Feedback* com questões fechadas (afirmações), referentes a suas impressões ao utilizar a *SMartyPerspective*, e questões abertas para verificar aspectos positivos e negativos dos elementos que a compõe (introdução, instrução e questão).

Foram definidas quinze questões afirmativas divididas em três dimensões do modelo TAM (Modelo de Aceitação de Tecnologia) (Davis, 1989) (Tabela 6.4) para verificar o grau de aceitação dos participantes perante a técnica *SMartyPerspective* após terem realizado a inspeção dos diagramas referentes à perspectiva que lhe foi atribuída para a análise.

As dimensões definidas do modelo TAM para este trabalho foram:

- Facilidade de Uso: refere-se o quanto o participante acredita que usar um determinado sistema o deixará livre de esforços (Davis, 1989);
- Utilidade: define o quanto o participante acredita que usar um sistema pode melhorar seu desempenho (Davis, 1989); e
- Intenção de Uso: o participante mede o grau de possibilidade em usar o sistema futuramente.

Como respostas para estas questões, o participante poderia escolher entre seis opções da escala *Likert*: discordo totalmente, discordo amplamente, discordo parcialmente, concordo parcialmente, concordo amplamente e concordo totalmente.

Tabela 6.4: Questões relacionadas ao modelo TAM do Questionário de *Feedback*

Dimensões TAM	**Questões**
Facilidade de Uso	1. Foi fácil aprender a utilizar a técnica *SMartyPerspective*.
	2. Eu considero que as instruções foram fáceis para serem seguidas.
	3. Eu entendia o que acontecia na minha interação com o cenário da técnica *SMartyPerspective*.
	4. Foi fácil ganhar habilidade no uso da técnica *SMartyPerspective*.
	5. Considero que é fácil utilizar a técnica *SMartyPerspective*.
	6. Os estereótipos da SMarty me ajudaram a propiciar uma melhor promoção por meio da técnica *SMartyPerspective*.
Utilidade	7. A introdução inicial ficou clara o suficiente para entender qual minha perspectiva e objetivos com a inspeção.
	8. As instruções me orientaram a encontrar os elementos nos diagramas inspecionados.
	9. As instruções são detalhadas o suficiente para me orientar a encontrar os elementos a serem inspecionados.
	10. As questões me ajudaram a detectar os defeitos nos diagramas inspecionados.
	11. Usar a *SMartyPerspective* melhorou o meu desempenho durante a inspeção.
	12. Considero a *SMartyPerspective* útil para detecção de defeitos.
Intenção de Uso	13. Utilizar a *SMartyPerspective* para detecção de defeitos é uma ótima ideia.
	14. Eu pretendo usar a técnica *SMartyPerspective* para inspeção de diagramas, sempre que possível.
	15. Eu adotaria a *SMartyPerspective* para inspeção de diagramas, no futuro.

Além das questões afirmativas, foram definidas 9 questões abertas para o participante exprimir sua opinião em relação aos elementos do cenário, sugestões, e os pontos positivos e negativos da *SMartyPerspective*.

As questões abertas são as seguintes:

16. Considere os diagramas de gerenciamento de variabilidade (caso de uso, classe, sequência, componente, atividade e *features*) e a perspectiva que lhe foi atribuída para responder as questões que seguem:
    (a) Faltou algum diagrama a ser considerado para que o trabalho seja feito corretamente para sua perspectiva? Quais diagramas devem ser inseridos? Justifique.
    (b) Algum diagrama definido para seu papel não é importante para que suas funções sejam exercidas? Qual diagrama deve ser removido deste cenário? Justifique.
17. Considere as questões definidas para o cenário da perspectiva que lhe foi atribuída, para responder as questões que seguem:
    (a) Alguma questão não ficou clara o suficiente? Informe o item e o que a torna ambígua.
    (b) Sob seu ponto de vista, alguma questão deve ser inserida no cenário? Qual item e porquê?
    (c) Sob seu ponto de vista, alguma questão deve ser removida do cenário? Qual item e porquê?
18. Você considera o processo de inspeção da técnica *SMartyPerspective* adequado com relação à identificação de defeitos em diagramas de gerenciamento de variabilidades? Justifique os pontos positivos.
19. Você considera que o formato dos cenários contribuiu de maneira clara, objetiva e precisa para a inspeção dos diagramas de gerenciamento de variabilidades? Justifique.
20. Você encontrou alguma dificuldade em identificar defeitos com o uso da técnica *SMartyPerspective*? Justifique os pontos negativos.
21. Você tem alguma sugestão para melhorar o cenário da *SMartyPerspective*?

As respostas dos participantes para cada uma dessas questões foram transcritas integralmente no Apêndice D e são analisadas na Seção 6.4.4.

6.4 Análise e Interpretação

Para este trabalho foram executadas quatro análises dos dados obtidos dos questionários preenchidos pelos 21 participantes. A primeira foi o coeficiente α de Cronbach para verificar a confiabilidade dos dados obtidos por meio do Questionário de Caracterização. Já para as questões fechadas baseadas no modelo TAM foi analisada quantativamente por meio do cálculo das porcentagens da concordância em relação às respostas dos participantes para cada um dos níveis e uma análise qualitativa pelo método de codificação das questões abertas do mesmo formulário.

Por fim, foi realizado uma análise da eficiência, eficácia e efetividade da técnica *SMarty-Perspective* por meio da quantidade de defeitos preenchidos no FID pelos participantes durante a inspeção dos diagramas da instrumentação.

6.4.1 Confiabilidade das Respostas de Caracterização

O coeficiente α de Cronbach mede a precisão dos dados de um instrumento em uma escala de 0 a 1. Ele calcula a estimativa da confiabilidade dos valores observados de um questionário aplicado para itens que utilizam a mesma escala de medição (Cronbach, 1951; Freitas e Rodrigues, 2005).

O α de Cronbach foi aplicado para as questões fechadas do estudo da viabilidade da técnica *SMartyPerspective* para determinar a consistência interna das respostas obtidas pelos participantes e da confiabilidade das respostas dadas entre os participantes.

Freitas e Rodrigues (2005) reuniu alguns trabalhos da literatura que definem os fatores que podem influenciar na confiabilidade dos questionários: (i) o número de itens, que pode gerar desinteresse, fadiga ou pressa do participante; (ii) tempo de aplicação, pois questionários grandes podem gerar respostas intensivas; e (iii) amostra de avaliadores, caso seja uma amostra de indivíduos semelhantes, ocasionado principalmente por avaliadores com mesma formação profissional e natureza.

Considera-se pela maioria dos estudos como valor aceitável para o α de Cronbach superior a 0,7. Porém, há diversas interpretações na literatura para estes valores, como a escala de Freitas e Rodrigues (2005). Nesta análise será utilizada a escala sugerida por George e Mallery (2003) para os valores de α, pois, seus níveis de confiabilidade são adequados ao propósito desta pesquisa:

- $\geq$ 0,9 - Excelente;
- $\geq$ 0,8 - Bom;

- $\geq$ 0,7 - Aceitável;
- $\geq$ 0,6 - Questionável;
- $\geq$ 0,5 - Ruim; e
- $<$ 0,5 - Inaceitável.

Foi aplicada para as respostas do Questionário de Caracterização, a correlação de α de Cronbach para cada uma das perspectivas em relação ao tempo de experiência do participante nas áreas de LPS e ES e experiência nos conceitos básicos para entendimento da *SMartyPerpective* (LPS, GV, *SMarty* e inspeção de software). Além desses grupos, foi calculado para todas as perspectivas de modo geral a coesão entre as respostas para o nível de conhecimento nos diagramas inspecionados pela *SMartyPerpective* (Tabela 6.5).

Tabela 6.5: Coeficiente de Cronbach para o Questionário de Caracterização

Grupos de respostas	**Perspectivas**	**Coeficiente de Cronbach**
TEMPO	GRP	0,8735294
	ERD	0,3571429
	AQD	0,0000000
	DSD	0,6382979
	GAD	0,9491525
EXPERIÊNCIA	GRP	0,7833333
	ERD	0,8409091
	AQD	0,8823529
	DSD	0,8320312
	GAD	-0,08333333
DIAGRAMAS		0,88798220

Fonte: o autor

Em relação ao **tempo de experiência** dos participantes para cada uma das perspectivas, o α de Cronbach encontrado para a perspectiva de ERD e AQD foi de 0,3571429 e 0,0, respectivamente. O que mostra um baixo nível de confiabilidade entre as respostas do Questionário de Caracterização. Este fato pode estar relacionado aos valores de tempo de experiência dos usuários serem próximos um dos outros para a perspectiva de AQD , variando apenas entre anos dois itens da escala (2 e 3). E para ERD, apesar de ser mais variado, as respostas são semelhantes.

Para os participantes que foram atribuídos os papéis de GRP e GAD o nível de confiabilidade para as respostas de acordo com a escala de George e Mallery (2003) é Bom e Excelente. Como dito anteriormente, esses valores são considerados bons para o trabalho. Já a para a perspectiva de DSD, o coeficiente teve valor igual α 0,6382979,

e, segundo a escala de George e Mallery (2003), este é um valor moderado para α e a confiabilidade para este grupo pode ser questionada.

A **experiência** (Tabela 6.5) agrupa os dados dos participantes referentes aos conceitos básicos relacionados a técnica *SMartyPerspective*. Três perspectivas (ERD, AQD, DSD) alcançaram níveis alto de heterogeneidade entre as respostas, ou seja, com $\alpha > 0{,}8$. Já a de GRP ficou bem próximo desse valor (α=0,7833333), sendo considerado um nível aceitável de confiabilidade para este trabalho.

Quanto à perspectiva de GAD, o valor encontrado para α= -0,08333333 para as questões relacionadas à experiência dos participantes. Este valor é inaceitável para este trabalho, pois, sugere uma baixa coesão para estes valores encontrados e pode estar relacionado a níveis de experiência bastante diferentes.

Por fim, foi calculado o α de Cronbach para os **diagramas** inspecionados pela técnica *SMartyPerpective* de modo geral, sem estar agrupado por perspectivas. Para tal foi calculado o coeficiente dessa amostra e o valor obtido foi de α=0,88798220. Este valor apresenta uma alta heterogeneidade entre os valores de experiência dos participantes nesses artefatos e, por consequência, uma maior confiabilidade do conjunto de dados dos 21 participantes.

Foi aplicado também a correção de α de Cronbach para os 15 itens do Questionário de *Feedback* divididos nas três dimensões do modelo TAM (Tabela 6.6).

Tabela 6.6: Coeficiente de Cronbach para o Questionário de *Feedback*

Dimensões	**Coeficiente de Cronbach**
Facilidade de uso	0,8771002
Utilidade	0,8716789
Intenção de uso	0,9104205

Fonte: o autor

Para as respostas relacionadas à **facilidade de uso** e **utilidade**, o valor resultante de α de Cronbach é de 0,8771002 e 0,8716789, respectivamente. Estes valores correspondem à uma alta coesão das respostas dos participantes. Já para **intenção de uso** foi obtido o valor 0,9104205, que é considerado excelente pela escala de George e Mallery (2003) e mostra uma alta heterogeneidade entre as respostas para esta dimensão.

6.4.2 Análise Quantitativa das Respostas

Foi analisado o nível de concordância para as questões definidas e classificadas conforme o modelo TAM. Nesta seção são apresentadas as porcentagens para cada um dos níveis

da escala *Likert* em relação ao número da questão e uma geral por dimensão. Ambas análises foram realizadas também para as perspectivas separadas.

Análise por Questão Afirmativa

A Tabela 6.7 apresenta o nível de concordância dos participantes em relação às quinze questões sobre Facilidade de Uso, Utilidade e Intenção de Uso da técnica *SMartyPerspective*.

O nível de concordância utilizado no estudo foi definido como (Tabela 6.7): discordo totalmente (DT), discordo amplamente (DA), discordo parcialmente (DP), concordo parcialmente (CP), concordo amplamente (CA) e concordo totalmente (CT).

Tabela 6.7: Porcentagens das respostas por afirmações

Dimensão	Nº questão	DT	DA	DP	CP	CA	CT
Facilidade de Uso	1	0,0%	0,0%	9,5%	42,9%	28,6%	19,0%
	2	0,0%	4,8%	0,0%	42,9%	23,8%	28,6%
	3	0,0%	0,0%	9,5%	28,6%	47,6%	14,3%
	4	0,0%	0,0%	14,3%	28,6%	47,6%	9,5%
	5	0,0%	0,0%	9,5%	33,3%	33,3%	23,8%
	6	0,0%	0,0%	4,8%	23,8%	23,8%	47,6%
Utilidade	7	0,0%	0,0%	0,0%	19,0%	42,9%	38,1%
	8	0,0%	0,0%	0,0%	33,3%	42,9%	23,8%
	9	0,0%	0,0%	0,0%	33,3%	38,1%	28,6%
	10	0,0%	0,0%	0,0%	19,0%	57,1%	23,8%
	11	0,0%	0,0%	4,8%	38,1%	19,0%	38,1%
	12	0,0%	0,0%	4,8%	19,0%	33,3%	42,9%
Intenção de Uso	13	0,0%	0,0%	4,8%	28,6%	19,0%	47,6%
	14	0,0%	0,0%	9,5%	38,1%	19,0%	33,3%
	15	0,0%	0,0%	9,5%	33,3%	19,0%	38,1%

Fonte: o autor

A Tabela 6.8 apresenta a porcentagem das respostas para os níveis de concordância para cada questão afirmativa separadas por perspectiva. Enquanto que a Tabela 6.9 apresenta a porcentagem para os níveis de discordância da escala por perspectiva.

As duas primeiras afirmações referenciam-se a facilidade em aprender utilizar a técnica *SMartyPerspective* e facilidade em seguir suas instruções. Apesar da maioria das respostas alternar entre os níveis de concordância, 42,9% concordaram parcialmente para cada uma delas e 9,5% discordaram parcialmente para a primeira afirmação e 4,8% discordaram amplamente para a segunda (Tabela 6.7).

Esses valores mostram que os inspetores possuem ainda ressalvas sobre a facilidade da técnica, o que pode ser entendida pela grande carga de questões dos cenários e que as instruções podem não ter ficado claras o suficiente para serem seguidas, principalmente,

Tabela 6.8: Porcentagens das respostas de concordância separadas por perspectiva

Dimen.	Nº ques.	Concordo Parcialmente					Concordo Amplamente					Concordo Totalmente				
		GRP	ERD	AQD	DSD	GAD	GRP	ERD	AQD	DSD	GAD	GRP	ERD	AQD	DSD	GAD
Facilidade de Uso	1	50%	0%	25%	60%	75%	25%	25%	75%	20%	0%	25%	50%	0%	0%	25%
	2	50%	25%	0%	60%	75%	0%	0%	100%	20%	0%	50%	75%	0%	0%	25%
	3	50%	25%	0%	40%	25%	50%	50%	50%	20%	75%	0%	25%	25%	20%	0%
	4	50%	25%	0%	40%	25%	25%	50%	75%	20%	75%	25%	25%	0%	0%	0%
	5	25%	0%	0%	80%	50%	50%	50%	50%	0%	25%	25%	50%	25%	0%	25%
	6	50%	0%	0%	20%	50%	0%	25%	25%	40%	25%	50%	75%	50%	40%	25%
Utilidade	7	25%	0%	0%	20%	50%	50%	25%	50%	60%	25%	25%	75%	50%	20%	25%
	8	25%	25%	0%	60%	50%	50%	25%	100%	20%	25%	25%	50%	0%	20%	25%
	9	25%	25%	0%	40%	75%	25%	0%	75%	60%	25%	50%	75%	25%	0%	0%
	10	25%	0%	25%	20%	25%	75%	50%	75%	60%	25%	0%	50%	0%	20%	50%
	11	25%	25%	0%	80%	50%	50%	0%	25%	20%	0%	25%	75%	50%	0%	50%
	12	25%	0%	0%	40%	25%	50%	50%	0%	40%	25%	25%	50%	75%	20%	50%
Intenção de Uso	13	50%	0%	0%	60%	25%	0%	50%	0%	0%	50%	50%	50%	75%	40%	25%
	14	50%	0%	0%	60%	75%	0%	25%	25%	40%	0%	25%	75%	50%	0%	25%
	15	50%	0%	0%	80%	25%	0%	25%	25%	0%	50%	25%	75%	50%	20%	25%

Fonte: o autor

Tabela 6.9: Porcentagens das respostas de discordância separadas por perspectiva

Dimen.	Nº ques.	Discordo Totalmente					Discordo Amplamente					Discordo Parcialmente				
		GRP	ERD	AQD	DSD	GAD	GRP	ERD	AQD	DSD	GAD	GRP	ERD	AQD	DSD	GAD
Facilidade de Uso	1	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	25%	0%	20%	0%
	2	0%	0%	0%	0%	0%	0%	0%	0%	20%	0%	0%	0%	0%	0%	0%
	3	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	25%	20%	0%
	4	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	25%	40%	0%
	5	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	25%	20%	0%
	6	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	25%	0%	0%
Utilidade	7	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	8	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	9	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	10	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	11	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	25%	0%	0%
	12	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	25%	0%	0%
Facilidade de Uso	13	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	25%	0%	0%
	14	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	25%	0%	25%	0%	0%
	15	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	25%	0%	25%	0%	0%

Fonte: o autor

para os papéis de GAD e DSD, 75% e 60%, respectivamente, para concordo parcialmente (Tabela 6.8).

Em relação a perspectiva de DSD, três participantes concordaram parcialmente (60%) que foi fácil aprender a utilizar a *SMartyPerspective* e um discordou parcialmente (20%) como é apresentado na Tabela 6.8 e Tabela 6.9. Para o GAD, a concordância parcial tanto para a facilidade da técnica quanto de seguir as instruções foi de 75%.

Esses valores podem estar relacionados diretamente aos cenários dessas perspectivas que exigem mais dos inspetores do que outros. O DSD com o maior número de questões e GAD com o maior número de artefatos inspecionados.

Para as afirmações 3 e 4, que compreendem o entendimento em relação â interação e à facilidade de ganhar habilidade utilizando a técnica, respectivamente, os papéis que mais

concordaram parcialmente e/ou discordaram parcialmente foram o GRP e ERD . Porém, 47,6% (Tabela 6.7) dos participantes concordaram amplamente para as duas afirmações, a perspectiva com o menor valor de concordância para essas duas questões é a de DSD (Tabela 6.8). O que dá alguns indícios que este cenário é mais difícil em ganhar habilidade e de interagir com ele.

Quanto à facilidade em utilizar a técnica *SMartyPerspective*, 61,6% dos participantes concordaram com essa afirmação, enquanto que 33,3% concordaram parcialmente e discordo parcialmente, 9,5% Tabela 6.7. Essas porcentagens revelam que há ainda pontos a serem melhorados para que a técnica seja considerada fácil de ser utilizada.

Por fim, a última afirmação da dimensão de **Facilidade de Uso** verifica como os inspetores consideram que os estereótipos da abordagem *SMarty* colaboram para uma melhor inspeção da técnica e 47,6% concordaram totalmente com ela. Logo, utilizar os estereótipos, ainda que com algumas ressalvas, podem colaborar para uma melhor inspeção, pelo foco e limitação dos elementos a serem inspecionados (Tabela 6.7).

A dimensão de **Utilidade** agrupou as afirmações referentes à utilidade dos elementos que fazem parte do cenário e do desempenho da técnica *SMartyPerspective* pela percepção dos participantes após a inspeção na busca de defeitos.

Para as afirmativas referentes à clareza das introduções para entendimento da perspectiva e das orientação das instruções (afirmativas 7 e 8, respectivamente), 42,9% para ambas concordaram amplamente com as afirmações, 38,1% concordaram totalmente para a primeira e 23,8% para a segunda. Concluindo assim, um bom resultado para a primeira versão da técnica (Tabela 6.7).

A Tabela 6.7 mosta que para os 33,8% que concordaram parcialmente que as instruções colaboraram na orientação, a maioria deles foi para o papel de DSD (60 %), com três pessoas, e GAD, com duas (50%). Novamente, os inspetores dessas perspectivas deram indícios de pontos que devem ser melhorados para estes cenários. Lembrando aqui, que as instruções fazem parte das características principais das técnicas de Leitura Baseada em Perspectiva e por isso, devem tornar o trabalho eficiente por meio do guia do que deve ser encontrado.

Dos participantes, 38,1% e 33,3% concordaram amplamente e parcialmente que as instruções são detalhadas o suficiente para orientá-los durante a inspeção (afirmação 9). Quanto à utilidade às afirmações na detecção de defeitos, a maioria dos participantes concordou amplamente com esta afirmação (57,1% - afirmação 10).

Essas porcentagens sugerem que apesar de algumas alterações serem necessárias por meio das porcentagens anteriores, os participantes compreendem a importância das afirmações no processo de inspeção.

O uso da *SMartyPerspective* para melhoria do desempenho no processo de inspeção em relação a uma leitura *Ad hoc* sem o procedimento do cenário operacional foi constatado na afirmação 11. Dos participantes que responderam este questionário, 38,1% concordaram totalmente e a mesma quantia concordou parcialmente. Desses, quatro DSD concordaram parcialmente e três ERD concordaram totalmente. Apontando mais uma vez problemas na definição dos cenários de DSD.

A ultima afirmação em relação à utilidade fez os participantes analisarem se a técnica *SMartyPerspective* é útil para seu propósito de detecção de defeitos. Desses, 42,9% concordaram totalmente, 33,3% amplamente, 19,3% parcialmente e 4,8% discordaram parcialmente desta afirmação (Tabela 6.7).

A última dimensão escolhida para agrupar as afirmações deste estudo é a Intenção de Uso, que define a intenção do participante em utilizar a técnica *SMartyPerspective* para inspeção dos diagramas de gerenciamento de variabilidades e de *features*.

Dos 21 participantes, 42,9% concordaram que utilizar a técnica é uma ótima ideia pela afirmação 13 e 33,3% concordaram amplamente. Para a afirmação 14, 38,1% concordaram parcialmente que utilizaram a técnica sempre que possível e 38,1% e 33,3% dos participante concordaram totalmente com essa afirmação (Tabela 6.7).

Finalizando o questionário baseado no modelo TAM, a afirmação 14 afirma que o participante utilizaria a técnica para inspeção de diagramas no futuro. Do total, 38,1% concordaram totalmente e 33,3% parcialmente.

Com o fim da análise das porcentagens de respostas para cada uma das opções da escala *Likert* é possível observar que apesar da grande porcentagem entre os níveis de concordância, há a necessidade de algumas melhorias nos cenários da *SMartyPerspective* para que haja uma maior orientação dos participantes pelos elementos da técnica, especialmente para a perspectiva de DSD e de GAD que tiveram a maior porcentagem de "concordo parcialmente" e alguns "discordo parcialmente".

Portanto, a análise desses dados mostra evidências de que a técnica é viável para a detecção de defeitos para os diagramas de gerenciamento de variabilidades e *features*, mas, que devem ser realizadas algumas melhorias para aumentar a porcentagem de participantes que concordam totalmente com a facilidade e utilidade da técnica.

Uma análise mais detalhada das dificuldades e pontos positivos apontados pelos participantes é apresentada na Seção 6.4.4 com análise qualitativa das respostas textuais dos participantes para as questões abertas da instrumentação. Além da análise inicial da eficiência, eficácia e efetividade da técnica na Selão 6.4.3.

Análise por Dimensão

Além das porcentagens de concordância para cada uma das quinze afirmações do Questionário de *Feedback*, as respostas foram analisadas por dimensões conforme mostra a Tabela 6.10 e por dimensões separadas por perspectivas Tabela 6.11 para cada um dos itens da escala *Likert*: discordo totalmente (DT), discordo amplamente (DA), discordo parcialmente (DP), concordo parcialmente (CP), concordo amplamente (CA) e concordo totalmente (CT).

Tabela 6.10: Porcentagens das respostas por dimensões do modelo TAM

Dimensão	DT	DA	DP	CP	CA	CT
Facilidade de Uso	0,00%	0,79%	7,94%	33,33%	34,13%	23,81%
Utilidade	0,00%	0,00%	1,59%	26,98%	38,89%	32,54%
Intenção de Uso	0,00%	0,00%	7,94%	33,33%	19,05%	39,68%

Fonte: o autor

Tabela 6.11: Porcentagens das respostas por dimensões do modelo TAM e perspectivas

Dimensão	Perspectiva	DT	DA	DP	CP	CA	CT
Facilidade de Uso	GRP	0,00%	0,00%	0,00%	45,83%	25,00%	29,17%
	ERD	0,00%	0,00%	4,17%	12,50%	33,33%	50,00%
	AQD	0,00%	0,00%	16,67%	4,17%	62,50%	16,67%
	DSD	0,00%	3,33%	16,67%	50,00%	20,00%	10,00%
	GAD	0,00%	0,00%	0,00%	50,00%	33,33%	16,67%
Utilidade	GRP	0,00%	0,00%	0,00%	25,00%	50,00%	25,00%
	ERD	0,00%	0,00%	0,00%	12,50%	25,00%	62,50%
	AQD	0,00%	0,00%	8,33%	4,17%	54,17%	33,33%
	DSD	0,00%	0,00%	0,00%	43,33%	43,33%	13,33%
	GAD	0,00%	0,00%	0,00%	45,83%	20,83%	33,33%
Intenção de Uso	GRP	0,00%	0,00%	16,67%	50,00%	0,00%	33,33%
	ERD	0,00%	0,00%	0,00%	0,00%	33,33%	66,67%
	AQD	0,00%	0,00%	25,00%	0,00%	16,67%	58,33%
	DSD	0,00%	0,00%	0,00%	66,67%	13,33%	20,00%
	GAD	0,00%	0,00%	0,00%	41,67%	33,33%	25,00%

Fonte: o autor

Ao analisarmos as respostas dos participantes pelas dimensões do método TAM, ao invés das questões individuais, temos que a maioria das respostas variaram entre os níveis de concordância: parcialmente, amplamente e totalmente.

Considerando as respostas concordo totalmente e concordo amplamente, foi obtido 57,94% de coincidência dos participantes em relação a **Facilidade de Uso** da técnica *SMartyPerspective*. Apesar de ser um bom valor, é possível analisar que há indícios de

pontos que devem ser melhorados na técnica. Tais pontos serão analisados nas próximas seções na análise textual e discussão dos resultados.

Ao fazermos um *link* com as as respostas divididas por questões da seção anterior, esses valores estão relacionados principalmente com a clareza e facilidade das instruções.

As perspectivas de ERD (83,33%) e AQD (79.17%) tiveram os maiores níveis de concordância para a **Facilidade de Uso** (Tabela 6.11), já a perspectiva de DSD (30%) foi a perspectiva com o menor valor de concordância para a facilidade de usar e aprender a utilizar a técnica.

O valor encontrado para DSD de concordância, assim como concordo parcialmente (50%) pode estar relacionado ao tamanho do cenário, já que em número de questões, era a perspectiva com a maior carga de elementos para serem revisados. Para GAD, o nível de concordo e concordo parcialmente foi de 50% para cada uma das escalas.

A **Utilidade** da técnica em um modo geral, também apresentou altos níveis de variação de concordância, 71,43% das respostas concordaram totalmente e parcialmente com as questões referentes à utilidade da *SMartyPerspective*. Este valor é um bom indicativo da viabilidade da técnica para ser utilizada como inspeção de diagramas *SMarty* e *features*.

Para a **Utilidade** da técnica em relação as perspectivas, a Tabela 6.11 apresenta novamente as perspectivas de ERD e AQD, ambas com 87,5%, com os maiores valores para concordância para **Utilidade** da técnica *SMartyPerspective*.

O menor valor para a concordância da **Utilidade** da técnica foi encontrado para a perspectiva de GAD (54,16%) e DSD (56,66%). O cenário de DSD tem o maior número de questões para inspeção de três diagramas (classe, componente e sequência) e a perspectiva de GAD tem o maior número de diagramas inspecionados, já que inspeciona os cinco diagramas suportados pela abordagem *SMartyPerspective*. Esta carga de trabalho, apesar de diferentes, podem estar relacionadas a percepção dos usuários de que a técnica não é útil para a eficiência da inspeção.

Porém, como na dimensão anterior, há ainda ressalvas dos participantes principalmente em relação ao desempenho do uso da técnica e da ajuda das questões e clareza das instruções para serem seguidas. O que reforça mais a necessidade de uma nova versão da técnica.

Em relação à **Intenção de Uso**, 58,73% concordaram totalmente e amplamente que possuem intenção de utilizar a *SMartyPerspective* sempre que possível no futuro e que seu uso é uma ótima ideia para detecção de defeitos em diagramas de gerenciamento de variabilidade.

Para esta dimensão, houve um valor relevante de respostas para concordo parcialmente (39% - Tabela 6.11). Essa porcentagem pode ser explicada com o tempo gasto pelos

participantes para realização das tarefas do experimento, como será mostrado melhor na Seção 6.4.3.

A perspectiva de ERD apresentou nível de concordância para **Intenção de Uso** de 100% como mostra a Tabela 6.11. já para AQD, o valor encontrado foi de 75% para concordo e 25% para discordo parcialmente. Enquanto que para DSD e GRP, o valor encontrado para **Intenção de Uso** foi de apenas 33,33%.

Estes valores para a **Intenção de Uso** ressaltam a necessidade de uma revisão nos cenários das perspectivas de GRP, DSD e AQD principalmente em relação a simplificação e menor sensação de esforço para ser mais aceita na acadêmia e indústria.

Foi identificado por Ma e Yuen (2011) que os fatores que mais influenciam a intenção de uso são as expectativas de desempenho e esforço. Logo, a utilidade da técnica e esforço gasto podem ter influenciado diretamente nas respostas do participante. Pois, apesar da alta utilidade percebida por eles (71,4%), o tempo gasto para realizar as tarefas e os cenários longos podem ter impactado nas respostas pelo esforço despedindo por eles.

6.4.3 Análise da Eficiência, Eficácia e Efetividade

Durante a execução do estudo, os participantes foram orientados a inspecionar os diagramas da LPS Arcade Game Maker (AGM) com defeitos injetados referentes à perspectiva que lhe foi atribuída por meio da técnica *SMartyPerspective*.

Portanto, como parte integrante da técnica, eles deveriam preencher o FID a cada defeito encontrado nos diagramas em inspeção. Os dados coletados do FID e do tempo que os participantes gastaram para realizar a inspeção foram analisados sem definir hipóteses, sem planejamento formal e sem comparação experimental com outras técnicas de inspeção.

Os dados obtidos foram apenas reaproveitando os dados disponíveis do estudo qualitativo para ter uma base inicial da eficiência, eficácia e efetividade dos participantes ao utilizar a técnica proposta para colaborar com o estudo das melhorias futuras a serem feitas na *SMartyPerspective*.

A Eficiência na detecção de defeitos é calculada por meio do número total de defeitos detectados durante a inspeção, dividido pelo tempo total despendido pelo participante em minutos.

$$\text{Eficiência(t)} = \frac{\text{Total de defeitos detectados}}{\text{Tempo total de inspeção}}$$

A Eficácia é dada pelo número total de defeitos detectados na inspeção, entre corretos e incorretos, dividido pelo número total de defeitos existentes no diagrama. Será calculado

neste trabalho a Eficácia do número total de defeitos para a perspectiva e não, para cada um dos diagramas inspecionados por ela separadamente.

$$\text{Eficácia(t)} = \frac{\text{Total de defeitos detectados}}{\text{Total de defeitos existentes}}$$

A Efetividade é dada pelo número total de defeitos detectados corretamente pelo inspetor dividido pelo número total de defeitos no diagrama. Assim como na Eficácia, será calculado neste trabalho pelo número total de defeitos para a perspectiva considerando todos os diagramas.

$$\text{Efetividade(t)} = \frac{\text{Total de defeitos detectados corretamente}}{\text{Total de defeitos existentes}}$$

A Tabela 6.12 mostra o total de defeitos (TD) que foram injetados em cada um dos diagramas para cada perspectiva, o número total de defeitos encontrados pelo participante (TE) e o total de defeitos encontrados corretamente para cada um dos diagramas inspecionados por eles (TC).

A partir dos dados da Tabela 6.12 e do tempo gasto em minutos durante a inspeção informado pelos participantes no formulário eletrônico e do total de defeitos incorretos encontrados pelo participante (TI) (Tabela 6.13), foi possível calcular evidências inicias da eficiência, eficácia e efetividade da técnica *SMartyPerspective* a partir do resultado de dezenove participantes, pois, o AQD4 entregou o FID vazio e o DSD4 fez a inspeção de apenas um dos diagramas.

Tabela 6.12: Número de defeitos observados durante tarefa de inspeção dos diagramas

ID	***feature***			**caso de uso**			**classe**			**componente**			**sequência**		
	TD	TE	TC	TD	TE	TC	TD	TE	TC	TD	TE	TC	TD	TE	TC
GRP1	5	1	1	6	2	2									
GRP2	5	3	2	6	3	2									
GRP3	5	6	1	6	6	5									
GRP4	5	5	5	6	6	5									
ERD1				7	8	6	6	5	4				7	7	7
ERD2				7	7	6	6	4	1				7	7	10
ERD3				7	5	5	6	2	1				7	2	2
ERD4				7	7	7	6	5	3				7	4	4
AQD1							8	4	3	8	5	5			
AQD2							8	4	3	8	4	4			
AQD3							8	2	2	8	4	4			
AQD4							8	0	0	8	0	0			
DSD1							10	10	9	8	5	3	7	4	4
DSD2							10	8	7	8	7	5	7	4	7
DSD3							10	4	4	8	4	1	7	4	6
DSD4							10		2	8	3		7		
DSD5							10	6	6	8	5	0	7	5	5
GAD1	5	3	3	9	9	6	11	7	5	9	8	7	7	4	4
GAD2	5	0	0	9	8	5	11	4	2	9	4	4	7	0	0
GAD3	5	4	4	9	10	7	11	5	3	9	7	5	7	4	4
GAD4	5	5	5	9	5	5	11	9	8	9	6	4	7	5	6

LEGENDA:
TD = Total de defeitos incorporados / TE = Total de defeitos encontrados
TC = Total de defeitos detectados corretamente

Fonte: o autor

Tabela 6.13: Valores referente à Eficiência, Eficácia e Efetividade da amostra de defeitos encontrados

ID	**Tempo**	**TD**	**TE**	**TI**	**TC**	**Eficiência**	**Eficácia**	**Efetividade**
GRP1	47	11	3	0	3	0,06	0,27	0,27
GRP2	60	11	6	2	4	0,10	0,55	0,36
GRP3	82	11	12	6	6	0,15	1,09	0,55
GRP4	180	11	11	1	10	0,06	1,00	0,91
ERD1	155	20	20	3	17	0,13	1,00	0,85
ERD2	127	20	21	7	14	0,17	1,05	0,70
ERD3	285	20	9	1	8	0,03	0,45	0,40
ERD4	157	20	16	2	14	0,10	0,80	0,70
AQD1	80	16	9	1	8	0,11	0,56	0,50
AQD2	50	16	8	1	7	0,16	0,50	0,44
AQD3	135	16	6	0	6	0,04	0,38	0,38
DSD1	120	25	19	3	16	0,16	0,76	0,64
DSD2	190	25	22	6	16	0,12	0,88	0,64
DSD3	105	25	14	5	9	0,13	0,56	0,36
DSD5	120	25	16	5	11	0,13	0,64	0,44
GAD1	90	41	31	6	25	0,34	0,76	0,61
GAD2	220	41	16	5	11	0,07	0,39	0,27
GAD3	105	41	30	7	23	0,29	0,73	0,56
GAD4	48	41	31	4	27	0,65	0,76	0,66
Média	**124,00**	**22,95**	**15,79**	**3,42**	**12,37**	**0,13**	**0,73**	**0,55**
Desvio Padrão	**61,43**	**10,44**	**8,29**	**2,35**	**6,75**	**0,14**	**0,24**	**0,18**
Mediana	**120,00**	**20,00**	**16,00**	**3,00**	**11,00**	**0,13**	**0,73**	**0,55**

LEGENDA:
TD = Total de defeitos incorporados / TE = Total de defeitos encontrados
TC = Total de defeitos detectados corretamente / TI = Total de defeitos detectados incorretamente

Fonte: o autor

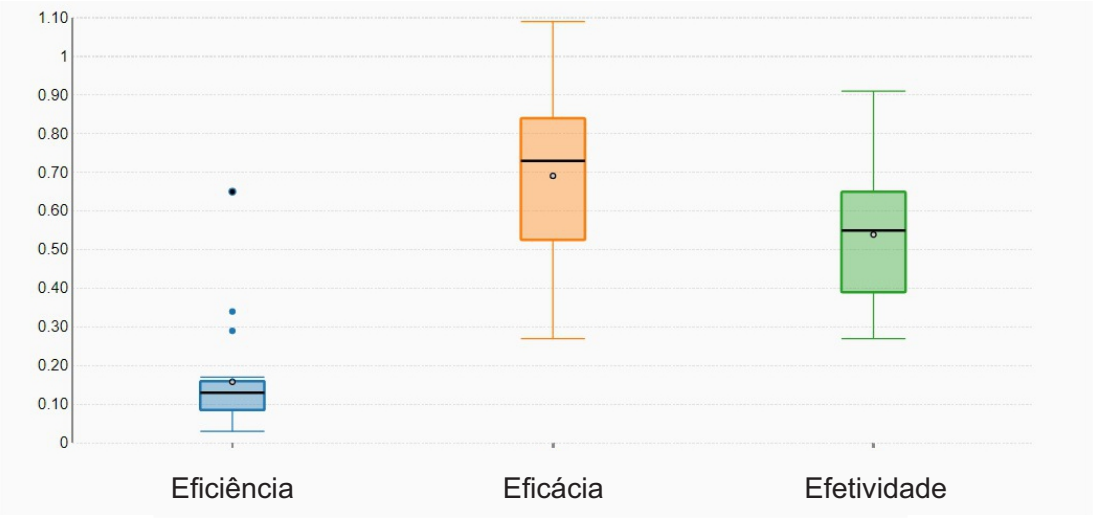
A Tabela 6.14 apresenta a média, mediana e desvio padrão para valores de Eficiência, Eficácia e Efetividade para cada uma das perspectivas da técnica *SMartyPerspective*. Dessa forma, é possível observar como foi a variação na detecção dos defeitos injetados para cada uma das perspectivas.

O box plot da **Eficiência** é mostrado na Figura 6.2. É possível observar na distribuição dos valores que apesar de terem vários valores discrepantes, 50% dos valores estão próximos e pouco distribuídos e não há uma grande diferença entre os valores máximo e mínimo definidos pelas caudas. Portanto, os participantes tiveram valores próximos de eficiência, ou seja, o número de defeitos encontrados por minuto por eles foi próximo um do outro.

O valor calculado para a mediana da **Eficiência** foi de 0,13 defeitos por minuto e ela se aproxima mais do limite máximo, assim como 50% dos valores obtidos. Logo, a cada 10 minutos é identificado 1,3 defeitos. Embora, seja um valor calculado para uma LPS acadêmica Um valor considerado excelente quando comparado ao custo de

[2]https://goodcalculators.com/box-plot-maker/

Figura 6.2: Box plot da Eficiência, Eficácia e Efetividade da técnica *SMartyPerspective*



Fonte: o autor [2]

Tabela 6.14: Média, Mediana e Desvião padrão relacionadas à Eficiência, Eficácia e Efetividade da amostra de defeitos encontrados por perspectiva

Perspectiva	Medida	Tempo	TD	TT	TI	TC	Eficiência	Eficácia	Efetividade
GRP	Média	92,25	11,00	8,00	2,25	5,75	0,09	0,73	0,52
	Desvio Padrão	52,18	0,00	3,67	2,28	2,68	0,03	0,33	0,24
	Mediana	71,00	11,00	8,50	1,50	5,00	0,08	0,77	0,45
ERD	Média	181,00	20,00	16,50	3,25	13,25	0,11	0,83	0,66
	Desvio Padrão	61,20	0,00	4,72	2,28	3,27	0,05	0,24	0,16
	Mediana	156,00	20,00	18,00	2,50	14,00	0,12	0,90	0,70
AQD	Média	88,33	16,00	7,67	0,67	7,00	0,11	0,48	0,44
	Desvio Padrão	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	Mediana	47,00	11,00	3,00	0,00	3,00	0,06	0,27	0,27
DSD	Média	133,75	25,00	17,75	4,75	13,00	0,14	0,71	0,52
	Desvio Padrão	33,05	0,00	3,03	1,09	3,08	0,02	0,12	0,12
	Mediana	120,00	25,00	17,50	5,00	13,50	0,13	0,70	0,54
GAD	Média	115,75	41,00	27,00	5,50	21,50	0,34	0,66	0,52
	Desvio Padrão	63,71	0,00	6,36	1,12	6,22	0,20	0,16	0,15
	Mediana	97,50	41,00	30,50	5,50	24,00	0,32	0,74	0,59

LEGENDA:
TD = Total de defeitos incorporados / TE = Total de defeitos encontrados
TC = Total de defeitos detectados corretamente / TI = Total de defeitos detectados incorretamente

Fonte: o autor

correção de defeitos em nível de código, caso esses defeitos passem para a próxima fase de desenvolvimento de software. As perspectivas que tiveram os maiores valores de Eficiência durante a execução do estudo foram DSD e em seguida GAD com mediana de 0,13 e 0,32, respectivamente (Tabela 6.14).

Diferente da **Eficiência**, o box plot da **Eficácia** (Figura 6.2) mostra uma maior distribuição dos dados e valores máximos e mínimos distantes um do outro. Assim como no gráfico anterior, a distribuição não é simétrica e a mediana (0,73) se aproxima mais do limite superior. Portanto, 50% dos participantes tiveram 73% de eficácia na detecção de de defeitos.

As perspectivas que obtiveram uma maior taxa de **Eficácia**, ou seja, encontraram uma maior porcentagem de defeitos verdadeiros foram às perspectivas de GRP e ERD, com mediana de 77% e 90%, respectivamente (Tabela 6.14). Esses valores podem estar relacionados a um menor número de questões e artefatos que essas perspectivas analisam nos diagramas quando comparada as outras, como também o nível de conhecimento do domínio da LPS AGM para estes participantes já conhecido pelos pesquisadores.

A amostra do box plot da **Efetividade** na Figura 6.2 é mais distribuído que a **Eficiência**, mas, menos do que a **Eficácia**. A mediana (0,55), assim como nas medidas anteriores não é simétrica e 50% dos valores da amostra obtiveram efetividade $\geq$ a 55%. A perspectiva de ERD (mediana de 70% - Tabela 6.14) foi a que teve uma maior efetividade na inspeção dos diagramas atribuídos nas tarefas. Já, a de AQD a foi que teve os menores valores para essa medida (27%).

A perspectiva de AQD é a que teve os menores valores tanto para eficácia, quanto para a efetividade, e ficando entre os menores para a eficiência. Logo, são necessárias melhorias nos cenários destas perspectivas para que os inspetores tenham um maior aproveitamento na atividade de detecção de defeitos.

Os participantes que assumiram o papel de ERD obteve os melhores valores de **Eficácia** e **Eficiência**, mas, também ficou entre os valores mais baixos para a **Eficiência** (Tabela 6.14). Este resultado mostra que apesar da inspeção ser demorada, estes participantes encontrou uma grande quantidade de defeitos nos diagramas.

Embora os participantes de DSD tiverem sido os que mais mencionaram melhorias e dificuldades no entendimento do cenário como mostraram a análise qualitativa (Seção 6.4.4). A perspectiva foi a que teve o melhor resultado para Eficiência e figurou entre os maiores para a Eficácia e Efetividade. Logo, pode talvez, estar relacionada a nível de conhecimento dos participantes ou o tamanho do cenário, que era o maior entre todos os outros.

De modo geral, esta análise mostra a necessidade de refinamentos nos cenários da técnica *SMartyPerspective* para que ela possa se tornar mais eficiente, efetiva e eficaz, para que possa ser assumida como uma boa alternativa na detecção de defeitos de diagramas *SMarty*.

6.4.4 Análise Qualitativa

Para a análise qualitativa foram utilizados neste trabalho os procedimentos do método de pesquisa qualitativo *Grounded Theory* de Strauss (1987), que consiste em fazer uma análise textual das respostas dos participantes e rotular as categorias encontradas e as relações que possuem entre si (Conte *et al.*, 2009).

O conceito de código definido para teoria "*dá um nome a um fenômeno de interesse para o pesquisador*" (Conte *et al.*, 2009), abstraindo informações que podem ser categorizadas e importantes para a análise do fenômeno pelo pesquisador (Corbin e Strauss, 2014). O processo de codificação é realizado em três etapas:

- codificação aberta (*open coding*): a partir da leitura textual, o pesquisador identifica os conceitos principais nos dados contidos no texto e cria as categorias;
- codificação axial (*axial coding*): com as categorias definidas, elas são relacionadas entre si por meio de conectores. "*Explicitam-se causas e efeitos, condições intervenientes e estratégias de ação, em proposições*" (Conte *et al.*, 2009); e
- codificação seletiva (*selective coding*): o pesquisador define a teoria escolhendo uma categoria central da qual todas as demais estão relacionadas a ela por meio de um refinamento.

Para este trabalho foram realizadas as duas primeiras etapas do processo, a codificação aberta e a axial. Logo, não foi definida uma categoria central que relaciona as duas categorias definidas na análise dos dados obtidos no questionário.

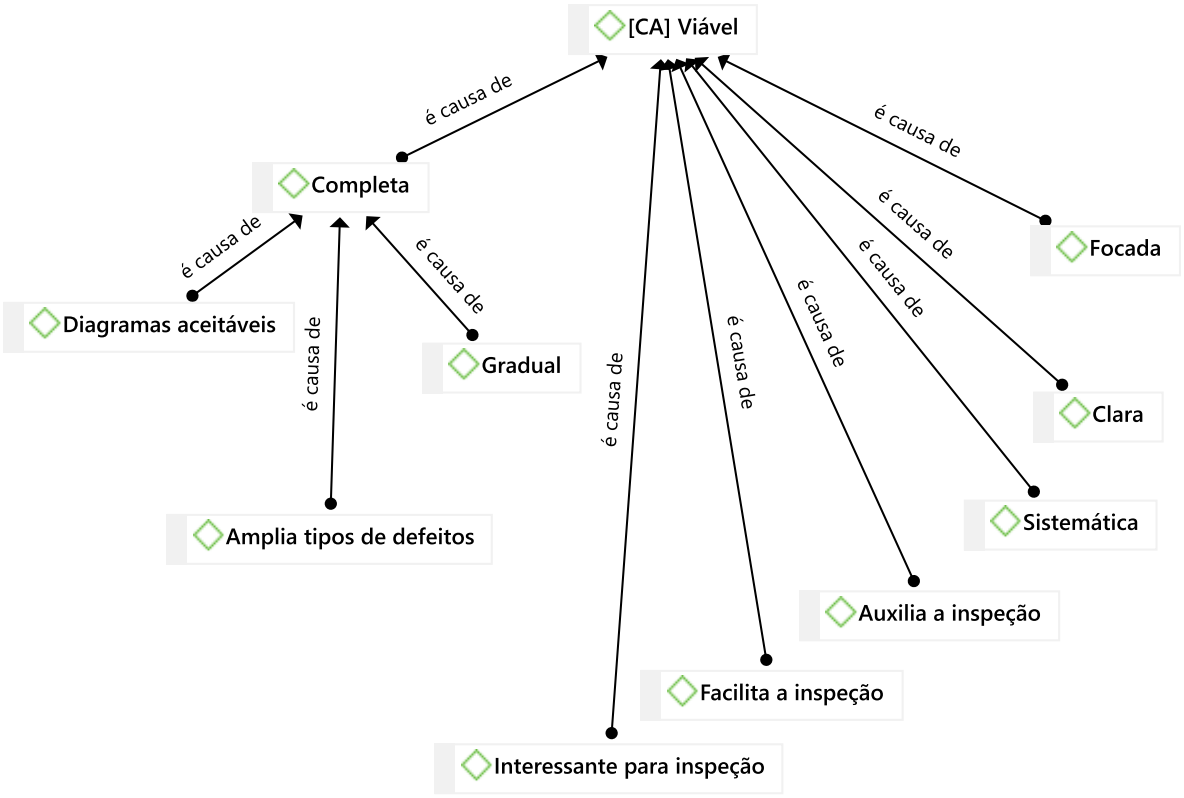
Primeiramente foi executada uma análise das respostas dos participantes para as questões abertas do Questionário de *Feedback* respondidos por eles após a inspeção dos diagramas definidos na instrumentação. Durante a leitura das respostas foram definidos os códigos de primeira ordem *in vivo*, ou seja, foram definidos conforme seguia a análise das respostas dos participantes e foram nomeados por meio das citações das respostas (Conte *et al.*, 2009).

As respostas dos participantes para as questões podem ser encontradas no Apêndice D e no pacote experimental do estudo. As respostas dos participantes foram analisadas em único corpus textual, ou seja, não foram agrupadas por questões, mas sim, colocadas todas juntas para extração dos códigos.

A Figura 6.3 e Figura 6.4 apresentam a representação gráfica das categorias Viável para inspeção e Revisão, respectivamente, analisadas e criadas no software ATLAS.TI. As categorias receberam o *label* [CA] identificando-a entre os códigos.

A categoria **Viável**, foi definida com base no agrupamento dos códigos que ressaltam os pontos positivos mencionados pelos participantes da técnica *SMartyPerspective*. Os códigos relacionados a esta categoria, são causas de porque ela pode ser considerada, para estes participantes viável para inspecionar os diagramas de gerenciamento de variabilidades e de *features* (Figura 6.3).

Figura 6.3: Categoria Viável



Fonte: o autor

A técnica *SMartyPerspective* foi considerada por alguns participantes como sendo Interessante, pois, ela fornece um guia e "*subsídios para que defeitos sejam detectados com maus facilidade*" (ERD1), por meio do seu formato de cenários com as instruções e questões subdivididas que faz com que ela seja segundo um participante interessante, eficaz para a área de inspeção.

Além de ser interessante, os participantes mencionaram que a técnica proposta apoia a atividade de inspeção, auxiliando na detecção de defeitos, como mencionado pelo AQD1 em: "*a técnica me auxiliou de maneira que permitisse encontrar problemas bem*

específicos", podendo ser, como mencionado pelo GRP2 pela sua estrutura de cenário operacional.

Para Shull *et al.* (2000), os benefícios de utilizar as técnicas PBR estão relacionadas as suas características: sistemática, focada, orientada a objetivo e personalizável e transferível por meio de treinamento. Algumas dessas características foram observadas e mencionadas diretamente pelos participantes.

Um dos fatores que podem estar relacionadas a observação da clareza da *SMartyPerspective* pelos participantes é sua característica Sistemática relacionada ao processo (ou cenário) ser fácil, bem detalhado e subdivido em grupos de instruções e questões para tipos de elementos específicos. Logo, ter um processo sistemático colabora para detectar defeitos como mencionado por DSD5: "*o processo é adequado e extremamente sistemático no que tange as características de diagramas UML modelados com o apoio da SMarty*".

Outra vantagem das técnicas PBR observadas pelos participantes na *SMartyPerspective* é sua característica de ser Focada em uma perspectiva particular e assim, limitar o que o inspetor deve procurar e os artefatos que deve ler, já que "*perspectivas ajudam a capturar interpretações diferentes com papéis e responsabilidades*" (DSD2).

Por ser focada em uma perspectiva em particular e colaborar com os inspetores a detectar tipos de defeitos específicos daquele tipo de diagrama e perspectiva, foi mencionado por alguns participantes que a técnica Amplia tipos de Defeitos. Esta característica está relacionada ao processo e questões que " *orienta "o que"procurar*" (GRP2) e permite que o inspetor amplie os tipos de defeitos que podem ser detectados e " *avalie todos os tópicos apresentados*" (ERD3) .

Pela sua característica sistemática do processo de inspeção por meio do cenário, os participantes mencionaram que a técnica *SMartyPerspective* é Gradual , pois, o passo a passo permite que as informações sejam verificadas um grupo de elementos cada vez e assim "*a profundidade da análise se desenvolve de forma gradual*" (AQD3) e "*as etapas para a avaliação contribuem para a construção do conhecimento*" (GRP1).

Apesar das respostas terem sido analisadas em conjunto, a questão 17 (Seção 6.3.2) indaga o leitor sobre a clareza das questões. Os participantes responderem variações de "*as questões ficaram claras*" e "*nenhuma*" questão não estava clara o suficiente. Já para outras questões, o código Clara foi mencionado em relação a sua característica (" *esta técnica é clara e objetiva*") e "*a distribuição dos casos de uso e das classes ficaram claras*".

Outra questão que indagou o participante especificamente sobre uma parte do cenário é a 16, que pergunta se alguns dos diagramas devem ser inseridos ou removidos para que cada cenário atenda as necessidades da perspectiva que foi atribuída ao participante.

A maioria, 18 deles, consideraram que os "*diagramas inspecionados estão apropriados"* (GRP2). Desta extração foi definido o código Diagramas apropriados.

Além dos diagramas serem aceitáveis para os papéis desenvolverem seu trabalho, o detalhamento das informações e instruções para guiar o inspetor o que deve ser feito durante a revisão e por colaborar para que ele encontre tipos defeitos que não sejam relembrados e detectados tão facilmente pelos inspetores, foi definido e relacionado aos códigos Gradual, Diagramas Aceitáveis e Amplia tipos de defeitos o código Completa, que sintetiza esses códigos, ressaltando que a técnica *SMartyPerspective* "*se apresenta de forma completa e objetiva"*, e as questões e diagramas já estão completos para cada perspectiva.

Logo, as menções a completeza da técnica e suas características relacionadas nos permite ter evidências iniciais de que a técnica atende seu propósito e é completa para inspeção de diagramas de gerenciamento de variabilidades.

As diversas características já citadas nos códigos anteriores, especialmente relacionadas aos elementos da técnica *SMartyPerspective*, fez com que ela fosse considerada por alguns participantes como fácil de ser utilizada, pois, "*a forma como foi organizado facilita muito na inspeção*" (ERD4) e "*é fácil utilizar e seguir o processo*" (AQD1). Dessas respostas, foi definido o código Facilita a inspeção.

A quantidade de participantes que mencionaram cada um dos códigos definidos na categoria Viável pode ser analisada na Tabela 6.15 que relaciona o ID dos participantes que mencionaram os códigos a cada um deles.

Tabela 6.15: Menções aos códigos da categoria viável

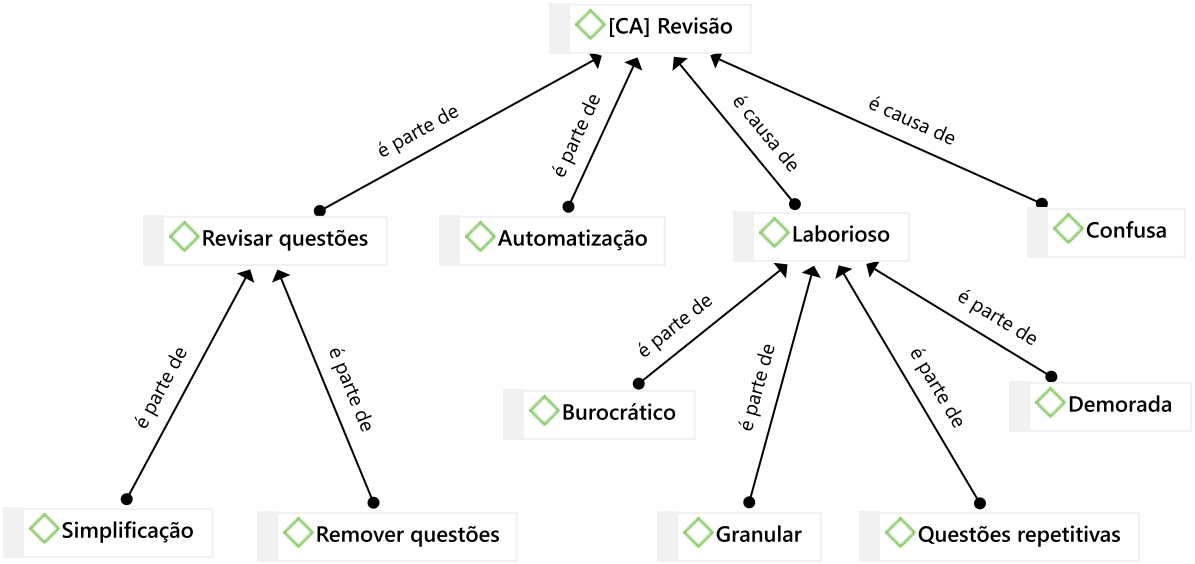
CÓDIGOS	**ID PARTICIPANTES**
Completa	GRP2, GRP4 e GAD3
Diagramas aceitáveis	ERD1, ERD4, DSD3, DSD5, GRP4, GRP1, GRP3, AQD3, AQD4, GAD1, GAD2 e GAD3
Gradual	GRP1, AQD1, AQD3 e DSD2
Amplia tipos de defeitos	GRP2 e ERD3
Focada	GRP2, DSD2, DSD3 e DSD4
Clara	AQD1 e ERD3
Auxilia a inspeção	GRP2, AQD1, DSD3 e GAD2
Sistemática	GRP4, AQD1, DSD5
Facilita a inspeção	DSD4,ERD1, ERD4, AQD1, AQED2, DSD1, DSD3, DSD4 e GAD4.
Interessante para inspeção	GRP1, GRP3 e ERD1

Fonte: o autor

Apesar dos pontos positivos citados pelos participantes nas questões abertas, eles sugeriram também revisões que devem ser realizadas na *SMartyPerspective* para tornar a atividade de inspeção e o processo da técnica mais eficiente. As melhorias sugeridas por

eles e as possíveis causas dos problemas apontados por eles foram reunidas na categoria Revisão (Figura 6.4). As sugestões serão analisados e servirão de base para as melhorias e estudos futuros da técnica.

Figura 6.4: Categoria Revisão



Fonte: o autor

Pela quantidade de diretrizes e estereótipos da abordagem SMarty e dos elementos dos tipos de diagramas UML a *SMartyPerspective* reuniu uma carga grande de questões que foram sentidas por alguns participantes. Os formulários longos fez com que os participantes achassem o processo de inspeção coma a técnica Demorada e cansativa: "*tornaram o experimento longo e um pouco cansativo, tornando a inspeção demorada*" (DSD2).

Além dos cenários longos, foi mencionado pelo participante DSD5 que algumas questões eram repetitivas, desta análise foi gerado o código Questões repetitivas. Outro problema relacionado à *SMartyPerspective* relatada pelos participantes é a burocracia da técnica "*considerando a tendência de processos de desenvolvimento cada vez mais ágeis*" e pelo processo ser restrito para abordagem SMarty, e assim, organizações que utilizam outras abordagens de gerenciamento de variabilidades não conseguem utilizar a *SMartyPerspective*.

Apesar da menção por alguns participantes sobre o cenário ser longo e trabalhoso, outros, relataram que as questões deveriam ser revisadas em relação a sua granularidade, para dividir as questões já existentes em questões menores, além de quebrar em etapas

menores, por ser uma técnica manual: "*já que vai ser manual, que seja por etapas menores*" (AQD4). Da extração dessas respostas foi definido o código Granular.

A junção dos quatro códigos anteriores (Demorada, Burocrática, Questões Repetitivas e Granular), trouxe aos participantes a sensação que o trabalho de inspeção com a técnica *SMartyPerspective* é trabalhoso e cansativo, pois, "*identificar todos os erros por um questionário torna o trabalho adhoc e cansativo*" (AQD4). Além disso, o participante DSD2 ressaltou que a quantidade de questões pode "*gerar cansaço e erros ao examinar projetos grandes*".

O participante DSD2 mencionou alguns que alguns elementos específicos do cenário da *SMartyPerspective* para a perspectiva de Desenvolvedor estão confusa e devem ser revisadas, como: "*as instruções para os Passos 6 e 7*", "*questões referente a inspeção do diagrama de sequência*" e "*a questão 5.5.7 está muito confusa*". Para tal, foi definido o código Confusa.

As questões devem ser revisadas para melhorar a interação do inspetor com a técnica de acordo com diversas citações pelo DSD2: "*de forma geral, sugiro revisar os itens das questões e tentar diminuir a quantidade de itens para tornar a inspeção mais produtiva*", "*recomendo a revisão dos itens das questões*", "*sugiro revisar a questão*" , "*recomendo revisar todas as questões referente a inspeção do diagrama de sequência*" e "*descrições dos cenários precisam ser revisadas*"

Todos os códigos extraídos das respostas dos participantes foram definidas como sendo causas da necessidade de se fazer uma revisão na técnica *SMartyPerspective*. Já os códigos que seguem foram definidos como sendo parte dos elementos que devem ser revisados e das melhorias futuras na técnica.

A necessidade de simplificar o cenário foi relatada por dois participantes. Para a Simplificação foi mencionado especificamente uma parte do cenário em relação ao diagrama de classes e seus relacionamentos, como também, no geral em relação a burocracia da técnica em ser algo específico para a abordagem SMarty e que "*imaginando uma aplicação prática, (...) o formulário tenha de ser simplificado*".

Ainda em relação à revisão e melhorias no cenário, foi sugerido pelo GRP3 que os elementos triviais relacionados ao diagrama de *features* devem ser removidos, pois, segundo ele "*ninguém desenha diagrama de features manualmente, mas sim conta com as ferramentas; assim sendo, se diversos elementos que são triviais fossem removidos desse checklist, a atividade seria mais fácil de entregar*". Além deste cenário, foi mencionado pelo DSD2 uma questão específica para ser removida para não influenciar na modelagem incorreta. Dessas respostas foi então extraído o código Remover Questões.

Outro ponto levantado para melhoria da técnica *SMartyPerspective* é tornar seu processo automatizado, pois, o processo foi considerado "*massante de fazer manualmente*" (GAD1). O desenvolvimento de uma ferramenta que "*automatize esse processo de inspeção e preenchimento da planilha* (GAD2)" foi citado por alguns participantes como sendo interessante. Dessas citações foi atribuído o código Automatização.

A Figura 6.4 apresenta a relação entre os códigos e os participantes que mencionaram cada um deles.

Tabela 6.16: Menções aos códigos da categoria Revisão

CÓDIGOS	**ID PARTICIPANTES**
Revisar Questões	DSD2
Simplificação	AQD4 e DSD5
Remover Questões	DSD2 e GRP3
Automatização	GAD1, GAD2 e AQD3
Laborioso	GRP3, AQD4 E DSD2
Burocrático	DSD4 e AQD4
Demorada	DSD2 e DSD5
Questões Repetitivas	DSD4 e DSD5
Granular	AQD4 e GAD4
Confusa	DSD2 e DSD5

Fonte: o autor

6.5 Discussão dos Resultados

Após a análise quantitativa, qualitativa e da eficiência, eficácia e efetividade da técnica *SMartyPerspective* pelas respostas dadas pelos participantes ao longo das tarefas realizadas dei evidências iniciais da resposta para a questão objetivo da avaliação qualitativa projetada e executada neste capítulo: "Os cenários da técnica *SMartyPerspective* são viáveis para a detecção de defeitos em diagramas UML *SMarty* e diagrama de *features*?".

A técnica *SMartyPerspective* teve 57,94% de concordância para a facilidade de uso, 58,73% para intenção de uso e 71,43% para a utilidade. Este último valor, apresenta evidências iniciais que a técnica é útil para a detecção de defeitos em diagramas de gerenciamento de variabilidades.

A *SMartyPerspective* apresentou também neste estudo inicial valores médios para a eficiência (0,13 defeito por minuto), eficácia (73%) e efetividade (55%) na detecção de defeitos pelos participantes no estudo.

Outro ponto que colabora com as evidências inciais da viabilidade da técnica se deu pela codificação das respostas abertas do Questionário de *Feedback*. Foram extraídos

diversos códigos que evidenciam os pontos positivos da técnica que permitem a técnica ser viável.

Apesar dos valores iniciais positivos foram apontados também alguns pontos negativos em relação ao esforço despendido pelos participantes durante o processo de inspeção do estudo, como: tempo gasto para inspeção, confusão nas questões e cenários muito longos. Portanto, serão necessárias algumas melhorias.

Nesta seção são discutidas algumas citações dos participantes no Questionário de *Feedback* após a inspeção. Essas respostas foram agrupadas para guiar as melhorias futuras em relação a técnica *SMartyPerpective* e em seu processo de avaliação para estudos futuros.

6.5.1 Melhorias no Processo de Avaliação

Nas questões sobre os diagramas e dificuldade, dois participantes mencionaram algumas melhorias não do uso, mas, sim, no processo do estudo. Baseadas nestas respostas, foram definidas melhorias para os próximos estudos de avaliação da técnica.

O diagrama oráculo (sem defeitos incorporados) foi entregue somente para o cenário de GAD, pois, esta perspectiva deve fazer a comparação entre duas versões do diagrama para encontrar os defeitos incorporados na segunda versão.

Para dois participantes que foram atribuídos outros papéis, GRD e DSD, mencionaram a falta de um oráculo para comparação entre os diagramas: "*Falta um oráculo*" (GRP3), "*Eu senti falta dos diagramas originais. (...) um diagrama correto ou já inspecionado deveria ser entregue ao participante para iniciar a inspeção do respectivo diagrama com defeitos injetados artificialmente*" (DSD2).

O propósito da *SMartyPerspective* não é fazer comparação entre os diagramas para os papéis, exceto o GAD, mas sim, auxiliar os leitores na tarefa de inspeção para que detecte os defeitos apenas seguindo o cenário e tendo conhecimento do domínio do problema, mesmo que superficial.

Outros participantes sugeriram a necessidade de um documento de requisitos mais detalhado para que possam buscar as informações para detectar os defeitos, como por exemplo "*sugestão é a descrição das classes e elementos dos demais diagramas*" (ERD1), "*a falta deste documento faz com que diversas questões não possam ser avaliadas e dificulta a comparação entre os diagramas*" (AQD1), "*fornecer uma documentação mais rica*" (AQD2) e "*o documento de requisitos não tinha muitas informações*", entre outras citações.

De fato, a especificação de requisitos entregue era resumida demais para encontrar defeitos de operações e atributos das classes. Portanto, será entregue para estudos futuros um documento de requisitos mais compatível com a atividade de inspeção.

A LPS utilizada neste estudo qualitativo é considerada uma LPS acadêmica. Logo, sua complexidade pode não refletir a indústria de software e grandes projetos. Neste caso, o uso de um documento de requisitos com uma LPS real pode aumentar exponencialmente os valores calculados a partir dos dados obtidos neste estudo. Porém, em estudos futuros, é importante conseguir fazer essa análise para verificar a interação dos participantes com os cenários para grandes projetos.

Outro problema em relação ao estudo foi apontado pelo participante GRP2 em que ele menciona "*o tempo "extra" gasto para "entender" o experimento*". Logo, deve ser tomado o cuidado para explicar mais detalhadamente as tarefas que devem ser realizadas durante o estudo, principalmente se o estudo for realizado *on-line* e não tiver o auxílio do pesquisador no momento que o participante tiver executando as tarefas.

Devido a alta diversificação das respostas entre concordo totalmente, concordo parcialmente e concordo amplamente dos participantes. Foi considerado pela pesquisadora que esta escala deve ser melhorada e simplificada para garantir resultados mais condensados e conclusivos. Portanto será adotada para os próximos experimentos a escala *Likert* contendo as seguintes opções: concordo totalmente, concordo, neutro ou indiferente, discordo e discordo totalmente.

Acredita-se que diminuindo os níveis de concordância e incluindo na escala "*neutro ou indiferente*", trará benefícios para os resultados, por possibilitar compreender até que certo ponto o participante está concordando (muito ou pouco, apenas) ou é indiferente a afirmativa. Tornando assim, a análise mais completa para entendimento dos pontos positivos e negativos da *SMartyPerspective*.

Após a conclusão das melhorias no processo de avaliação da técnica, ela será avaliada em mais alguns estudos:

- análise da viabilidade da nova versão da técnica;
- avaliar se as perspectivas estão corretas do ponto de vista dos especialistas em LPS;
- análise de uma reunião de inspeção para verificar se as perspectivas realmente estão detectando defeitos diferentes; e
- avaliação experimental em relação a eficiência, eficácia e efetividade da técnica comparando-a com as técnicas SMartyCheck e *Ad hoc*.

6.5.2 Melhorias Futuras na Técnica *SMartyPerspective*

Muitas das melhorias que devem ser realizadas na técnica *SMartyPerspective* foram mencionadas na seção anterior por meio da categoria "*Revisão*" e dos códigos relacionados a ela que foram extraídos de citações dos participantes nas respostas do questionário.

O plano de melhoria da técnica inclui duas etapas principais: na primeira, serão analisadas as dificuldades dos participantes neste primeiro estudo para sintetizar os cenários nas informações que são importantes aos revisores e evitar a fadiga dos mesmos pela quantidade de questões e informações contidas nele. Já a segunda etapa, será iniciada com a constatação de que a técnica está mais leve e pode ser automatizada.

O GRP4 apontou que sentiu falta "*de um passo para inspecionar se as variabilidades foram especificadas apropriadamente*". De fato, foi definido durante a construção do cenário que detalhes relacionados a notação de variabilidade que são mais específicos não seriam inspecionados pelo GRP que deve ter apenas uma visão do negócio como um todo e quais são as variabilidades definidas. Porém, pode ser incluído conforme seja necessário depois de uma segunda análise para a próxima versão da técnica.

Foi comentado por alguns participantes o quanto o processo de inspeção utilizando a *SMartyPerspective* é demorado. Estes comentários são reafirmados ao analisar a média de tempo de inspeção gasto e preenchido por eles no formulário. A média sem distinção entre os papéis foi de 02:04h, um valor bastante alto, considerando que haviam papéis com menos diagramas para serem inspecionados ou com menos detalhes para verificar que outros.

O grupo com maior tempo gasto pelos participantes foram os que ficaram com a perspectiva de ERD que gastou em média 03:01h para inspecionar os diagramas de caso de uso, classe e sequência, em segundo, o grupo de DSD com 02:29h para inspecionar os diagramas de caso de uso, classe e sequência. Já o grupo com menor tempo foi de AQD com apenas 01:19h para inspecionar os diagramas de classe e componente.

Apesar desses valores de tempo serem considerados altos pelos participantes, é importante compreender que eles eram voluntários na pesquisa. Logo, o tempo gasto (média de 2 horas) pode ser considerado longo, porém, se formos considerar a indústria de software e o preço a ser pago por um defeito que passe para as próximas fases de desenvolvimento e até para o usuário, o tempo gasto para encontrar defeitos (cerca de 0,13 por minuto neste estudo) pode custar muito mais barato para indústria e compensar utilizar a técnica *SMartyPerspective* para inspeção dos diagramas *SMarty*.

Além de ressaltar a demora em realizar as inspeções, alguns autores citaram a dificuldade em entender as questões referentes ao diagrama de sequência e de classe,

entre eles o DSD2 mencionou que as questões do "*diagrama de sequência, são muito confusas*", GAD4, "*acredito que a dificuldade maior foi saber aplicar a técnica no diagrama de sequência*" e ERD1, "*o único ponto que levanto com sugestão é a descrição das classes e elementos dos demais diagramas*".

Vale ressaltar que as questões que foram consideradas difíceis para os participantes, para o diagrama de classe e de sequência, são inspecionados pelas perspectivas que tiveram neste estudo a maior média de tempo de inspeção: ERD e DSD. Logo, estes cenários devem ser revisados e melhorados para uma maior eficiência da técnica. Sendo a perspectiva de DSD, inclusive que mais apontaram melhorias nas respostas textuais para os cenários da *SMartyPerspective*.

Outro cenário que deverá ser revisado com cuidado é o de AQD que tiveram valores baixos de Eficiência, Eficácia e Efetividade na detecção de defeitos. Sendo necessário então, uma revisão para tronar o cenário mais simples e completo, para que o leitor não tenha tanta dificuldade em relação ao tempo e em detectar defeitos que não são.

Além dos cenários mencionados, será revisado também as questões e instruções de todas as perspectivas da *SMartyPerspective*, para que diminua a burocracia da técnica, questões repetidas e com elementos que são considerados triviais e podem ser resolvidos facilmente nos softwares para modelagem dos diagramas.

A burocrácia da técnica será analisada também em relação a permitir apenas a inspeção de diagramas *SMarty*. Será analisado e considerado após a revisão da técnica englobar outras abordagens de gerenciamento de variabilidades buscando um cenário mais leve e que atenda as mais diversas perspectivas e organizações sem gerar confusões ou dúvidas por partes dos inspetores.

Como visto em trabalhos da literatura, há a motivação de criar cenários que sejam ainda mais ativos para os inspetores e colabore não só na inspeção, mas, na criação de outros artefatos. Por isso, será analisado a possibilidade de realizar estudos que contemplem a criação de artefatos por meio dos cenários da *SMartyPerspective*, deixando-a assim, mais completa.

Depois da revisão da técnica *SMartyPerspective*, serão realizados alguns estudos experimentais para verificar se a efetividade e o número de defeitos detectados pelos participantes aumentou após a simplificação da técnica. Com essas evidências iniciais, será então passada para a próxima fase de melhoria da técnica: a automatização.

Considerando que os diagramas inspecionados na *SMartyPerspective* são os diagramas *SMarty* e os diagramas de caso de uso, e, sua modelagem é suportada pelos ferramenta *SMartyModelling*. Será estudada uma maneira de incluir a *SMartyPerspective* dentro da ferramenta, para que fique mais fácil para o inspetor detectar os defeitos já na fase de

criação dos diagramas e não exija um esforço a mais dele de buscar as informações em outros locais ou tenha que exportar os diagramas para serem analisados.

6.6 Ameaças à Validade

Quanto às validades do estudo e suas ameaças, foram consideradas relevantes e identificadas com relação aos impactos para tal estudo são apresentadas nesta seção.

6.6.1 Validade de Conclusão

Apesar da preocupação em selecionar participantes com um nível significativo de conhecimento no contexto do estudo. Os dados obtidos da análise deste estudo são apenas indicadores e não conclusivos.

Mesmo com uma amostra relevante, 21 participantes, essa quantidade ainda é pequena para ter resultados conclusivos, especialmente porque o estudo necessitava dividir os participantes em cinco blocos, um para cada uma das perspectivas da *SMartyPerspective*. Assim, o número total de participantes para cada bloco foi ainda mais reduzido.

Alguns dos participantes,já conhecidos pelo grupo de pesquisa por trabalhar com LPS e ter conhecimento prévio da LPS AGM, embora não haja uma análise oficial dessas informações, já que não foi questionado essa informação no estudo, supõe-se que este conhecimento pode ter influenciado na quantidade de defeitos identificados. Pois, por uma análise parcial, os participante que já se sabe que conhecem a LPS AGM detectaram mais defeitos quando comparado a outros.

Outra ameaça a validade de conclusão está relacionada a divisão feita entre as perspectivas. Como a divisão seguiu as respostas dos participantes em relação aos papéis que consideravam capaz de exercer, eles podem ter sido divididos de forma errônea pelos pesquisados, pois, participantes que escolheram apenas um dos papéis ficaram com o papel escolhido mesmo que a experiência com os diagramas atribuídos a ele não fosse alto.

Os participantes que escolheram vários papéis foram divididos de acordo com a experiência citada com os diagramas, o que pode não refletir verdadeiramente que possam ser especialistas naquele papel. Portanto, essa divisão pode ser uma ameaça aos valores obtidos e a conclusão da análise.

6.6.2 Validade de *Constructo*

Apesar do nível de experiência do grupo de GAD convergir para um mesmo perfil de participante (α de Cronbach = -0,08333333), os participantes receberam um treinamento sobre os conceitos de LPS, técnicas de inspeção de software, abordagem *SMarty* e a técnica *SMartyPerspective*. Logo, eles tiveram a compreensão necessária para responder o Questionário de *Feedback* e detectar os defeitos do diagrama.

Devido às questões de falta de tempo, não foi executado um estudo piloto antes do estudo em si para verificar sua viabilidade. Portanto, isso pode ser considerada uma ameaça ao estudo, por não ter sido verificado as melhores condições para sua execução, como ressaltou o participante GRP2 que mencionou a demora para compreender as tarefas que deveriam ser executadas por ele.

A construção do experimento seguiu as características da LPS AGM que foi utilizada para inspeção. Porém, a descrição da LPS não era completa o suficiente para detectar alguns tipos de defeitos, especialmente, em relação aos atributos e métodos das classes.

Além disso, a LPS AGM por ser acadêmica, não tem a complexidade que LPS reais da indústria podem ter. Desta forma, os valores encontrados neste estudo inicial para um estudo controlado, pode não refletir corretamente a realidade da eficiência, eficácia e efetividade da técnica, assim como as percepções dos participantes.

6.6.3 Validade Externa

Quanto à validade externa, podem ser detectadas ameaças quanto à instrumentação, pois, os diagramas da LPS AGM não são comerciais, ou seja, não são de casos reais. Logo, em estudos adicionais devem ser consideradas LPSs reais para obter resultados diferentes na indústria.

6.6.4 Validade Interna

Algumas considerações podem ser feitas para a avaliação a ser realizada quanto à validade interna: i) como todos os participantes do estudo serão participantes em Engenharia de Software, não houve diferença significativa entre as habilidades do grupo; (ii) o treinamento quanto aos assuntos abordados no experimento, nivelou o conhecimento quanto a inspeção de software, podendo assim, ser considerado que as respostas fornecidas são válidas e significantes; e (iii) não houve influência entre os participantes durante o estudo, já que será *on-line* e os participantes não interagiram entre si e saber um dos outros.

Uma importante ameaça a validade interna desta avaliação é o cansaço e exaustão dos participantes durante a execução do experimento. Alguns relataram como mostra a análise qualitativa, que a inspeção foi longa e demorada. Logo, o cansaço pode ter sido uma ameaça a capacidade do participante em detectar os defeitos no diagrama ao passar despercebido alguma informação na hora de ler as questões do cenário.

6.7 Considerações Finais

Neste capítulo foi apresentado o planejamento e execução do estudo qualitativo para verificar a viabilidade da técnica *SMartyPerspective* para detecção de defeitos em diagramas de gerenciamento de variabilidades, realizada em fevereiro de 2021 com 21 participantes da área de Engenharia de Software.

As análises das respostas dos participantes por meio do coeficiente α de Cronbach para verificar a confiabilidade dos dados, a porcentagem da concordância das questões de *feedback* e da aplicação do método *Grounded Theory* para verificar as impressões dos participantes perante a técnica *SMartyPerspective*, resultaram em uma lista de melhorias para a técnica que serão realizadas como trabalhos futuros.

Além disso, as análises dos dados do Questionário de *Feedback* mostraram evidências de que a técnica *SMartyPerspective* é viável para inspeção, necessitando apenas, de melhorias em relação a simplificação e diminuição da carga de questões para que torna a inspeção mais eficiente com a realidade das organizações.

7

Conclusão

Atividades de verificação e validação devem ser iniciadas o quanto antes no ciclo de desenvolvimento de software. Para promover a qualidade de LPS, estas atividades são necessárias desde a concepção do núcleo de artefatos, pois, entre os ativos ali contidos há tanto os que são variáveis e específicos de cada produto, como também os que serão reutilizados por toda a família. Logo, um defeito não encontrado em um desses artefatos se propagará por todos os membros de uma LPS e dos produtos gerados.

A inspeção é um dos métodos de revisão de software capaz de encontrar defeitos nas etapas iniciais do processo de desenvolvimento de software de forma efetiva e eficiente para que falhas não se propaguem durante todo o ciclo de vida. Para colaborar com suas atividades, técnicas de leitura dão diretrizes aos inspetores norteando-os para o que deve ser inspecionado no artefato.

Estudos iniciais realizados com a técnica de Leitura Baseada em Perspectiva (PBR) mostram em sua maioria que ela é mais efetiva e eficiente para a inspeção de software quando comparada a técnicas tradicionais mais usadas na literatura. Isso porque ela considera, por meio de cenários, as diferentes perspectivas dos usuários dos produtos de software facilitando o processo de inspeção e do uso de informações relevantes respeitando as especialidades específicas dos revisores.

Neste trabalho foi especificada a técnica *SMartyPerspective*, uma técnica baseada em PBR para gerenciamento de variabilidades em diagramas *SMarty* de caso de uso, classe, componente e sequência e diagramas de *features* considerando as perspectivas de Gerente de Produto, Engenheiro de Requisitos de Domínio, Arquiteto de Domínio, Desenvolvedor de Domínio e Gerente de Ativos de Domínio.

Para verificar a viabilidade da técnica *SMartyPerspective* para detecção de defeitos em diagramas de gerenciamento de variabilidades, foi realizado um estudo qualitativo com vinte e um participantes da área de Engenharia de Software entre mestrandos, mestre, doutorandos e doutores. Os resultados, apesar de incipientes, fornecem evidência de que a técnica é viável para o contexto aplicado. Porém, ela deve ser melhorada para que haja maior aceitação e eficiência.

7.1 Contribuições

A principal contribuição deste trabalho é a especificação da técnica *SMartyPerspective* para gerenciamento de variabilidades em diagrama de *features* e diagramas *SMarty* de caso de uso, classe, componente e sequência, considerando as diversas perspectivas da Engenharia de Domínio de LPS. Isto se deve ao fato de que então não haviam trabalhos que relacionam LPS com técnicas de leitura baseada em perspectiva.

As técnicas encontradas que relacionam LPS com técnicas de leitura baseada em *checklist* inspecionam apenas alguns tipos de diagramas, duas delas, inspecionam diagramas de *features*, e uma delas, os diagramas *SMarty* de caso de uso, classe e componente.

Dessa forma, a técnica *SMartyPerspective* especificada contribui não só para o avanço da área de inspeção com uma nova perspectiva para inspeção em LPS, mas também, fornece subsídios para mais tipos de diagramas entre seus cenários. Logo, ela pode colaborar com mais leitores de artefatos na indústria que utilizem estes diagramas para o processo de desenvolvimento, bem como, colaborar com a literatura que carece de técnicas baseadas em PBR, especialmente as relacionadas com LPS.

Apesar de diversos trabalhos apresentarem papéis para inspeção de software que evidenciam a eficiência destas técnicas, não foram encontrados até o momento trabalhos que relacionam perspectivas de LPS com inspeção de software. Logo, as perspectivas definidas na *SMartyPerpective* para Engenharia de Domínio de LPS contribui de maneira significativa para a área de inspeção, com uma inspeção focada, e de LPS, visto que, há uma escassez de trabalhos relacionados aos papéis em LPS de modo geral.

O estudo de viabilidade da técnica *SMartyPerspective* para inspeção de diagramas de gerenciamento de vatiabilidaes também pode ser considerado uma contribuição, pois permite exercitar as várias perspectivas definidas para a técnica no contexto de LPS. Assim, cenários e defeitos puderam ser analisados quanto à viabilidade da técnica com relação a vários critérios como eficiência, eficácia e efetividade, além das dimensões de facilidade de uso, utilidade e intenção de uso do modelo TAM.

Outro ponto a ser mencionado como contribuição foi a atualização do mapeamento (Apêndice A) de Geraldi e OliveiraJr (2017a) para identificar defeitos utilizados na literatura. Essa atualização pode contribuir para os pesquisadores da área de inspeção com as taxonomias mais recentes utilizadas para definição das técnicas de inspeção de software e então ser utilizada para novas propostas de técnicas.

7.2 Limitações

Alguns pontos podem ser mencionados como fatores limitantes dos resultados obtidos durante a realização deste estudo.

A principal limitação pode ser relacionada às perspectivas definidas para este trabalho, pois, elas foram identificadas da literatura do trabalho de Van der Linden *et al.* (2007). Este trabalho apesar de reunir diversos casos da indústria da época, pode não refletir mudanças e as funções exercidas atualmente nas organizações que desenvolvem LPS. Assim, foi tentado definir a *SMartyPerspective* considerando os papéis mais gerais em qualquer organização.

As perspectivas definidas para a *SMartyPerspective* foram retiradas no estudo de Van der Linden *et al.* (2007) e apesar dos autores terem reunido diversos casos da indústria de LPS podem não refletir os papéis atualmente utilizados na indústria. Logo, serão necessários estudos futuros para avaliar com especialistas se as perspectivas estão adequadas.

A técnica *SMartyPerspective*é específica para diagramas *SMarty* de gerenciamento de variabilidades e de *features*. Essa característica pode ser considerada uma limitação, pois, há outras abordagens na literatura para o gerenciamento de variabilidades. Essas abordagens podem ser utilizadas por outros participantes que já estejam acostumados e, então, criarem uma barreira para o uso da *SMartyPerspective*.

A amostra limitada de participantes que aceitaram realizar o estudo da viabilidade da técnica *SMartyPerspective* permite que os resultados da avaliação sejam apenas evidências iniciais dos resultados. E que apesar de ter sido de grande valia o estudo para apontar os pontos que devem ser melhorados, os resultados não podem ser generalizados para qualquer ambiente ou características dos participantes.

Outra limitação pode ser relacionada à LPS utilizada na instrumentação do estudo de viabilidade, pois, ela é uma LPS acadêmica, o que pode não representar completamente um ambiente real para inspeção, e assim, detalhes importantes terem ficado de fora da técnica ou até mesmo da avaliação. Além disso, uma LPS real da indústria pode

aumentar bastante a complexidade dos elementos como também dificultar o uso da técnica *SMartyPerspective* pela quantidade de questões nos cenários.

Não foram encontradas na literatura, até então, outra técnica de inspeção baseada em PBR para LPS ou especificamente para gerenciamento de variabilidades. Isto limita a comparação da técnica com outras baseadas em PBR tanto em relação aos estudos empíricos qualitativos, como também, para comparar as perspectivas e cenários que têm sido utilizados em estudos mais atuais para LPS.

Por fim, este trabalho se limita também em relação à estudos experimentais para comparar a técnica *SMartyPerspective* com outras técnicas de inspeção de artefatos de LPS que já existem na literatura, especialmente a *SMartyCheck* que se baseia em *checklists* para inspecionar diagramas *SMarty* de caso de uso, classe e componente.

7.3 Trabalhos Futuros

Durante o desenvolvimento deste trabalho foram identificados alguns pontos que devem ser melhorados na técnica *SMartyPerspective* para permitir a sua futura adoção para detecção de defeitos. Para tal, serão descritos nesta seção os trabalhos futuros para tornar a técnica mais completa e documentada.

Os códigos atribuídos pela codificação a partir dos textos das respostas dos participantes para o Questionário de *Feedback*, permitiram identificar algumas questões que devem ser melhoradas para que a técnica *SMartyPerspective* seja menos confusa e trabalhosa como foi mencionado por alguns dos participantes.

A confusão causada por alguns elementos dos cenários pode ser resolvida removendo alguns itens que são triviais e readaptando alguns que não ficaram claros o suficiente. Portanto, será necessário fazer uma revisão em todos os elementos do cenário, especialmente os das perspectivas do Desenvolvedor de Domínio e do Gerente de Ativos, pelos níveis de concordância com as dimensões de Facilidade de Uso, Utilidade e Intenção Comportamental, e, do Engenheiro de Requisitos de Domínio pelo tempo gasto pelos participantes para condução da inspeção.

Após refinamento da técnica *SMartyPerspective* é sugerida uma nova avaliação qualitativa para verificar com outros participantes se a nova versão tem um maior grau de aceitação e menos itens para ser revisados.

Outros estudos que serão realizados futuramente incluem verificar com especialistas de LPS se as perspectivas definidas para este trabalho refletem a indústria de software atualmente, como também, verificar por meio de experimentos se os cenários definidos

para *SMartyPerspective* estão de fato detectando diferentes tipos de defeitos, ou, se há a necessidade de reformular a quantidade de cenários para a técnica.

Com a técnica *SMartyPerspective* bem definida, será importante fazer uma avaliação quantitativa buscando comparar a técnica em relação à eficiência, eficácia e efetividade com a técnica *SMartyCheck*, que inspeciona diagramas *SMarty* de caso de uso, classe e componente com base nas técnicas de leitura baseada em *checklist* e a técnica *Ad hoc*, em que o inspetor não recebe nenhuma orientação durante a inspeção.

As técnicas PBR são orientadas e ativas, elas devem fazer com que o inspetor trabalhe ativamente com os artefatos durante a detecção de defeitos. Como trabalho futuro, a técnica *SMartyPerpective* pode ser analisada quanto à sua capacidade de criar novos diagramas partindo de um diagrama existente e tendo como base o cenário que é utilizado para detecção de defeitos.

Outro trabalho futuro mencionado pelos participantes é a automatização da técnica *SMartyPerspective*, colaborando para que a inspeção fique "menos maçante" e "mais ágil". É sugerido pela pesquisadora, que a automação da técnica seja relacionada à ferramenta *SMartyModelling* (Silva e OliveiraJr, 2020, 2021; Silva. *et al.*, 2020), que permite a modelagem e o gerenciamento dos diagramas *SMarty* e de *features*. Dessa forma, os estereótipos próprios da abordagem serão gerenciados mais facilmente.

Outros estudos podem e devem ser planejados e executados para aprimorar a técnica e generalizar os resultados, que até aqui, são apenas evidências iniciais. Além dos estudos, um possível trabalho futuro é expandir a técnica para garantir a inspeção em outras abordagens de gerenciamento de variabilidade, tornando-a mais geral para ser usada em diferentes tipos de organização.

REFERÊNCIAS

de Almeida, E. S. Software reuse and product line engineering. In: *Handbook of Software Engineering*, Springer, p. 321–348, 2019.

Alshazly, A. A.; Elfatatry, A. M.; Abougabal, M. S. Detecting defects in software requirements specification. *Alexandria Engineering Journal*, v. 53, n. 3, p. 513 – 527, 2014.

Bartié, A. *Garantia da qualidade de software*. Gulf Professional Publishing, 2002.

Basili, V.; Caldiera, G.; Rombach, H. Goal Question Metric Approach. *Encyclopedia of Software Engineering*, 1994.

Basili, V. R.; Green, S.; Laitenberger, O.; Lanubile, F.; Shull, F.; S o rumg aa rd, S.; Zelkowitz, M. V. Lab package for the empirical investigation of perspective-based reading. v. 1, n. 2, p. 133–164, 1996a.

Basili, V. R.; Green, S.; Laitenberger, O.; Lanubile, F.; Shull, F.; Sørumgård, S.; Zelkowitz, M. V. The empirical investigation of perspective-based reading. *Empirical Software Engineering*, v. 1, n. 2, p. 133–164, 1996b.

Bera, M. H. G.; OliveiraJr, E.; Colanzi, T. E. Evidence-based SMarty Support for Variability Identification and Representation in Component Models. In: *Int. Conf. on Enterprise Information Systems*, Barcelona, Spain: ScitePress, 2015, p. 295–302.

Bettin, G. C.; Geraldi, R. T.; OliveiraJr, E. Experimental evaluation of the smartycheck techinique for inspecting defects in uml component diagrams. In: *Anais do 17º Simpósio Brasileiro de Qualidade de Software*, 2018, p. 101–110.

Biffl, S.; Halling, M. Investigating the defect detection effectiveness and cost benefit of nominal inspection teams. *IEEE Transactions on Software Engineering*, v. 29, n. 5, p. 385–397, 2003.

Breitman, K. K.; do Prado Leite, J. C. S. Suporte automatizado à gerência da evolução de cenários. In: *WER*, 1998, p. 49–56.

Carneiro, G.; Laigner, R.; Kalinowski, M.; Winkler, D.; Biffl, S. Investigating the influence of inspector learning styles on design inspections: Findings of a quasi-experiment. In: *CIbSE 2017 - XX Ibero-American Conference on Software Engineering*, 2017, p. 222–235.

Chen, L.; Ali Babar, M.; Ali, N. Variability management in software product lines: A systematic review. In: *Proceedings of the 13th International Software Product Line Conference*, USA: Carnegie Mellon University, 2009, p. 81–90.

Choma Neto, J. *Uma abordagem memetica para otimizar projeto de linha de produto de software.* Dissertação de Mestrado, Universidade Estadual de Maringá, 2017.

Chren, S.; Buhnova, B.; Macak, M.; Daubner, L.; Rossi, B. Mistakes in uml diagrams: analysis of student projects in a software engineering course. In: *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, IEEE, 2019, p. 100–109.

Ciolkowski, M.; Differding, C.; Laitenberger, O.; Münch, J. Empirical investigation of perspective-based reading: A replicated experiment. *Fraunhofer Institute for Experimental Software Engineering, Germany*, 1997.

Clements, P.; Northrop, L. *Software product lines.* Addison-Wesley Boston, 2002.

Coallier, F.; Champagne, R. A product line engineering practices model. *Science of Computer Programming*, v. 57, n. 1, p. 73–87, 2005.

Conte, T.; Cabral, R.; Travassos, G. H. Aplicando grounded theory na análise qualitativa de um estudo de observação em engenharia de software–um relato de experiência. In: *V Workshop"Um Olhar Sociotécnico sobre a Engenharia de Software"(WOSES 2009)*, sn, 2009, p. 26–37.

Corbin, J.; Strauss, A. *Basics of qualitative research: Techniques and procedures for developing grounded theory.* Sage publications, 2014.

Cronbach, L. J. Coefficient alpha and the internal structure of tests. *psychometrika*, v. 16, n. 3, p. 297–334, 1951.

Cunha, R.; Conte, T. U.; de Almeida, E. S.; Maldonado, J. C. A Set of Inspection Technique on Software Product Line Models. In: *Int. Conf. on Software Engineering and Knowledge Engineering*, KSI Research Inc. and Knowledge Systems Institute Graduate School, 2012, p. 657–662.

Damian, A.; Campos, U.; Conte, T.; de Souza, C. Comd2: Family of techniques for inspecting defects in models that affect team communication. In: *The 30th International Conference on Software Engineering and Knowledge Engineering*, 2018, p. 298–341.

Davis, F. D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, p. 319–340, 1989.

Denger, C.; Ciolkowski, M. High quality statecharts through tailored, perspective-based inspections. In: *Proceedings of 29th Euromicro Conference*, IEEE, 2003, p. 316.

Denger, C.; Kolb, R. Testing and inspecting reusable product line components. In: *Proceedings of ACM/IEEE international symposium on Emppirical software engineering*, 2006, p. 184–193.

Fagan, M. E. Design and Code Inspections to Reduce Errors in Program Development. *IBM Systems Journal*, v. 38, n. 2/3, p. 258, 1979.

Fagan, M. E. Advances in software inspections. *IEEE Transactions on Software Engineering*, v. SE-12, n. 7, p. 744–751, 1986.

Felizardo, K. R. Apoio computacional para inspeção de software. *INFOCOMP Journal of Computer Science*, v. 3, n. 2, p. 14–18, 2004.

Fernández, D.; Monperrus, M.; Feldt, R.; Zimmermann, T. The open science initiative of the empirical software engineering journal. *ESE*, v. 24, n. 3, p. 1057–1060, 2019.

Fiori, D.; Gimenes, I.; Maldonado, J.; OliveiraJr, E. Variability management in software product line activity diagrams. In: School, K. S. G., ed. *Proceedings of the International Conference on Distributed Multimedia Systems*, 2012, p. 89–94.

Freitas, A.; Rodrigues, S. A avaliaçao da confiabilidade de questionário: uma análise utilizando o coeficiente alfa de cronbach. in: Simpósio de engenharia de produção. *Anais... Bauru-SP: UNESP*, 2005.

George, D.; Mallery, P. *SPSS para Windows passo a passo: um guia simples e referência. Atualização 11.0* . 4 ed. Allyn & Bacon, 2003.

Geraldi, R. T.; Conte, T. U.; Steinmacher, I. F.; OliveiraJr, E. Checklist-based Inspection of SMarty Variability Models - Proposal and Empirical Feasibility Study. In: *Int. Conf. on Enterprise Information Systems*, ScitePress, 2015, p. 268–275.

Geraldi, R. T.; OliveiraJr, E. Defect types and software inspection techniques: a systematic mapping study. *J. Comput. Sci.*, v. 13, p. 470–495, 2017a.

Geraldi, R. T.; OliveiraJr, E. Towards Initial Evidence of SMartyCheck for Defect Detection on Product-Line Use Case and Class Diagrams. *Journal of Software*, v. 12, p. 379–392, 2017b.

Gomaa, H. Designing software product lines with uml 2.0: From use cases to pattern-based software architectures. In: Morisio, M., ed. *Reuse of Off-the-Shelf Components*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, p. 440–440.

Granda, M.; Parra, O.; Condori-Fernández, N. A metrics-driven inspection framework for model transformations. In: *22nd Ibero-American Conference on Software Engineering, CIbSE, 2019*, 2019, p. 321–334.

Höhn, E. N. *Técnicas de leitura de especificação de requisitos de software: estudos empíricos e gerência de conhecimento em ambientes acadêmico e industrial.* Tese de Doutoramento, Universidade de São Paulo, 2003.

IEEE Ieee standard for software reviews and audits. *IEEE Std 1028-2008*, p. 1–53, 2008.

IEEE Ieee standard classification for software anomalies. *IEEE Std 1044-2009 (Revision of IEEE Std 1044-1993)*, p. 1–23, 2010.

IEEE Ieee standard for system and software verification and validation. *IEEE Std 1012-2012 (Revision of IEEE Std 1012-2004)*, p. 1–223, 2012.

Institute, S. S. E. Software product lines. [*On-line*].
Disponível em `http://www.sei.cmu.edu/productlines/`

Koscianski, A.; dos Santos Soares, M. *Qualidade de software-2ª edição: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software.* Novatec Editora, 2007.

KRUEGER, C. W. Software Reuse. *ACM Comput. Surv.*, v. 24, n. 2, p. 131–183, 1992.

LAHTINEN, J. Application of the perspective-based reading technique in the nuclear i&c context. *CORSICA work report*, 2012.

LAITENBERGER, O.; ATKINSON, C. Generalizing perspective-based inspection to handle object-oriented development artifacts. In: *Proceedings of the 21st International Conference on Software Engineering*, 1999, p. 494–503.

LAITENBERGER, O.; ATKINSON, C.; SCHLICH, M.; EL EMAM, K. An experimental comparison of reading techniques for defect detection in uml design documents. *Journal of Systems and Software*, v. 53, n. 2, p. 183–204, 2000.

LAITENBERGER, O.; KOHLER, K. The systematic adaptation of perspective-based inspections to software development projects. In: *Proceedings of the 1st Workshop on Inspection in Software Engineering, SQRL, Mc Master University*, 2001, p. 105–114.

LANUBILE, F.; MALLARDO, T.; CALEFATO, F.; DENGER, C.; CIOLKOWSKI, M. Assessing the impact of active guidance for defect detection: a replicated experiment. In: *10th International Symposium on Software Metrics, 2004. Proceedings.*, IEEE, 2004, p. 269–278.

LEE, M.-C. Software quality factors and software quality metrics to enhance software quality assurance. *Current Journal of Applied Science and Technology*, p. 3069–3095, 2014.

VAN DER LINDEN, F. J.; SCHMID, K.; ROMMES, E. *Software product lines in action: the best industrial practice in product line engineering.* Springer Science & Business Media, 2007.

MA, W.; YUEN, A. E-learning system acceptance and usage pattern. In: *Technology acceptance in education*, Brill Sense, p. 201–216, 2011.

MA, Z. An approach to improve the quality of object-oriented models from novice modelers through project practice. *Frontiers of Computer Science*, v. 11, n. 3, p. 485–498, 2017.

MAFRA, S. N.; TRAVASSOS, G. H. Técnicas de leitura de software: Uma revisão sistemática. *XIX Simpósio Brasileiro de Engenharia de Software (SBES 2005)*, 2005.

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, 2014.

Marcolino, A.; Oliveira, E.; Gimenes, I. Variability identification and representation in software product line uml sequence diagrams: Proposal and empirical study. In: *2014 Brazilian Symposium on Software Engineering*, 2014a, p. 141–150.

Marcolino, A.; Oliveira, E.; Gimenes, I.; Barbosa, E. F. Empirically based evolution of a variability management approach at uml class level. In: *2014 IEEE 38th Annual Computer Software and Applications Conference*, 2014b, p. 354–363.

Marcolino, A.; OliveiraJr, E. Comparing smarty and plus for variability identification and representation at product-line uml class level: A controlled quasi-experiment. *Journal of Computer Science*, v. 13, p. 617–632, 2017.

Marcolino, A.; OliveiraJr, E.; M.S. Gimenes, I.; Barbosa, E. Variability resolution and product configuration with smarty: An experimental study on uml class diagrams. *Journal of Computer Science*, v. 13, p. 307–319, 2017.

de Mello, R. M.; Teixeira, E. N.; Schots, M.; Werner, C. M. L.; Travassos, G. H. Verification of Software Product Line Artifacts: a Checklist to Support Feature Model Inspections. *Journal of Universal Computer Science*, v. 20, n. 5, p. 720–745, 2014.

Miller, G. A. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, v. 63, n. 2, p. 81, 1956.

Nepomuceno, T.; OliveiraJr, E.; Geraldi, R.; Malucelli, A.; Reinehr, S.; Silva, M. A. G. Software product line configuration and traceability: An empirical study on smarty class and component diagrams. In: *IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, 2020, p. 979–984.

Nepomuceno, T. S.; OliveiraJr, E. Configurando produtos específicos da linha de produtos de software com smarty e PLUS: um estudo experimental sobre diagramas de caso de uso. In: ACM, ed. *Anais do 17º Simpósio Brasileiro de Qualidade de Software, SBQS 2018, Curitiba, Brasil*, 2018, p. 81–90.

Nepomuceno, T. S.; OliveiraJr, E.; Penteado, R. R.; 'e lio Graciotto Silva, M. A.; Zorzo, A. F. Estudo empírico sobre configuração de produto e rastreabilidade com base em uml linhas de produto. In: Associates, C., ed. *Proceedings of the XXIII*

Iberoamerican Conference on Software Engineering, CIbSE 2020, Curitiba, Paraná 'a, Brasil, 9 a 13 de novembro de 2020, 2020, p. 166–179.

Neto Gracioli, C.; Neto Anderlin, A.; Kalinowski, M.; de Oliveira, D. C. M.; Sabou, M.; Winkler, D.; Biffl, S. Using model scoping with expected model elements to support software model inspections: Results of a controlled experiment. In: *ICEIS (2)*, 2019, p. 107–118.

Northrop, L.; Clements, P.; Bachmann, F.; Bergey, J.; Chastek, G.; Cohen, S.; Donohoe, P.; Jones, L.; Krut, R.; Little, R.; outros A framework for software product line practice, version 5.0. *SEI*, 2007.
Disponível em `http://www.sei.cmu.edu/productlines/index.html`

Northrop, L. M. Sei's software product line tenets. *IEEE Software*, v. 19, n. 4, p. 32–40, 2002.

OliveiraJr, E.; Gimenes, I.; Maldonado, J. *Systematic management of variability in uml-based software product lines.* Tese de Doutoramento, 2010a.

OliveiraJr, E.; Gimenes, I. M. S.; Maldonado, J. C. Systematic Management of Variability in UML-based Software Product Lines. *Journal of Universal Computer Science*, v. 16, n. 17, p. 2374–2393, 2010b.

OMG Unified modeling language (omg uml), version 2.4. [*On-line*].
Disponível em `https://www.omg.org/spec/UML/2.4.1/Superstructure/PDF`

OMG Object Constraint Language - Version 2.4. [*On-line*].
Disponível em `http://www.omg.org/spec/OCL/2.4`

OMG Unified modeling language (omg uml), version 2.5. [*On-line*].
Disponível em `https://www.omg.org/spec/UML/2.5.1/PDF`

Parnas, D. L.; Weiss, D. M. Active design reviews: Principles and practices. *Journal of Systems and Software*, v. 7, n. 4, p. 259–265, 1987.

Pohl, K.; Böckle, G.; van Der Linden, F. J. *Software Product Line Engineering: Foundations, Principles and Techniques.* Springer Science & Business Media, 2005.

Pressman, R.; Maxim, B. *Software engineering: a practitioner's approach.* 8 ed. Palgrave macmillan, 2016.

Sabaliauskaite, G.; Matsukawa, F.; Kusumoto, S.; Inoue, K. An experimental comparison of checklist-based reading and perspective-based reading for uml design document inspection. In: *Proceedings International Symposium on Emppirical Software Engineering*, 2002, p. 148–157.

Sabaliauskaite, G.; Matsukawa, F.; Kusumoto, S.; Inoue, K. Further investigations of reading techniques for object-oriented design inspection. *Information And Software Technology*, v. 45, n. 9, p. 571–585, 2003.

Santos, L. B. R. D.; Júnior, V. A. D. S.; Povoa, L. V.; Freitas, A. V.; Mario, C. D. C. Software inspections: comparing a formal method based with a classical reading methodology. *International Journal of Computer Applications in Technology*, v. 59, n. 4, p. 296–317, 2019.

Shull, F.; Rus, I.; Basili, V. How perspective-based reading can improve requirements inspections. *Computer*, v. 33, n. 7, p. 73–79, 2000.

Shull, F. J. *Developing techniques for using software documents: A series of empirical studies*. Tese de Doutoramento, University of Maryland at College Park, USA, 1998.

Silva, F.; Santos, A.; Soares, S.; França, C.; Monteiro, C. Critical appraisal of systematic reviews in software engineering : a tertiary study. v. 4, 2010.

Silva, L. F.; OliveiraJr, E. Evaluating usefulness, ease of use and usability of an uml-based software product line tool. In: *Proceedings of the 34th Brazilian Symposium on Software Engineering*, SBES '20, New York, NY, USA: Association for Computing Machinery, 2020, p. 798–807 (*SBES '20*, v.).

Silva, L. F.; OliveiraJr, E. SMartyModeling: An Environment for Engineering UML-Based Software Product Lines. In: *15th International Working Conference on Variability Modelling of Software-Intensive Systems*, VaMoS'21, New York, NY, USA: Association for Computing Machinery, 2021 (*VaMoS'21*, v.).

Silva., L. F.; OliveiraJr, E.; Zorzo., A. F. Feasibility Analysis of SMartyModeling for Modeling UML-based Software Product Lines. In: *Proceedings of the 22nd International Conference on Enterprise Information Systems - Volume 2: ICEIS,*, INSTICC, SciTePress, 2020, p. 442–449.

da Silva Lima, A. *Uml 2.5 - do requisito a solução*. Saraiva Educação S.A., 2018.

Soares, L. Reuso e seus obstáculos na engenharia de software. *Revista de Engenharia e Pesquisa Aplicada*, v. 2, n. 1, 2016.

Sommerville, I. *Engenharia de Software*. Pearson Brasil, 2011.

Sousa, A.; Uchôa, A.; Fernandes, E.; Bezerra, C. I.; Monteiro, J. M.; Andrade, R. M. Rem4dspl: A requirements engineering method for dynamic software product lines. In: *Proceedings of the XVIII Brazilian Symposium on Software Quality*, 2019, p. 129–138.

de Souza, B.; Motta, R.; de Costa, D.; Travassos, G. H. An iot-based scenario description inspection technique. In: *Proceedings of the XVIII Brazilian Symposium on Software Quality*, New York, NY, USA: ACM, 2019a, p. 20–29.

de Souza, B.; Motta, R.; Travassos, G. H. The first version of scenariotcheck: A checklist for iot based scenarios. In: *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, New York, NY, USA: ACM, 2019b, p. 219–223.

Souza, I.; de Mello, R.; Almeida, E.; Werner, C.; Travassos, G. Experimental evaluation of fmcheck: A replication study. 2016.

Strauss, A. *Qualitative analysis for social scientists*. Cambridge university press, 1987.

Szyperski, C.and Gruntz, D.; Murer, S. Component Software: Beyond Object-Oriented Programming. 2002.

Tianual, P.; Pohthong, A. Defects detection technique of use case views during requirements engineering. In: *Proceedings of the 2019 8th International Conference on Software and Computer Applications*, 2019, p. 277–281.

Tokdemir, G.; Metin, D. Understanding bpmn through defect detection process. In: *Proceedings of the Seventh International Symposium on Business Modeling and Software Design*, 2017, p. 180–185.

Travassos, G.; Biolchini, J. Revisões sistemáticas aplicadas a engenharia de software. In: *XXI SBES-Brazilian Symposium on Software Engineering*, 2007.

Travassos, G.; Shull, F.; Fredericks, M.; Basili, V. R. Detecting defects in object-oriented designs: using reading techniques to increase software quality. *ACM Sigplan Notices*, v. 34, n. 10, p. 47–56, 1999.

Travassos, G. H. Software defects: Stay away from them. do inspections! *2014 9th International Conference on the Quality of Information and Communications Technology*, p. 1–7, 2014.

Travassos, G. H.; Gurov, D.; Amaral, E. *Introdução à engenharia de software experimental*. Relatório Técnico, COPPE/UFRJ, 2002.

Vaish, N.; Sharma, A. Semi-automated system based defect detection in software requirements specification document. In: *5th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering*, 2018, p. 1–5.

Van Gurp, J.; Bosch, J.; Svahnberg, M. On the notion of variability in software product lines. In: *Proceedings Working IEEE / IFIP Conference on Software Architecture*, 2001, p. 45–54.

Vieira, M. F. P. *SMartyCheck 3.0: uma Extensão da Técnica para Detecção de Defeitos em Diagramas UML do SMartyComponents*. Relatório Técnico, Universidade Estadual de Maringá, Maringá, 2017.

Villamizar, H.; Anderlin Neto, A.; Kalinowski, M.; Garcia, A.; Méndez, D. An approach for reviewing security-related aspects in agile requirements specifications of web applications. In: *2019 IEEE 27th International Requirements Engineering Conference (RE)*, 2019, p. 86–97.

Walia, G.; Carver, J. A systematic literature review to identify and classify software requirement errors. *Information and Software Technology*, v. 51, n. 7, p. 1087–1109, 2009.

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, 2006.

Young, T. J. *Using Aspect to Build a Software Product Line for Mobile Devices*. Tese de Doutoramento, University of British Columbia, 2005.

Zhang, H.; Wang, S.; Yue, T.; Ali, S.; Liu, C. Search and similarity based selection of use case scenarios: An empirical study. *Empirical Softw. Engg.*, v. 23, n. 1, p. 87–164, 2018.

Zhu, Y. *Software Reading Techniques: Twenty Techniques for More Effective Software Review and Inspection*. Apress, 2016.

Apêndice A: Atualização do Mapeamento Sistemático

A atualização de uma revisão sistemática pode ser realizada por três motivos: (i) atualização temporal, para ampliar o período das publicações; (ii) extensão de pesquisa, que acontece no mesmo período da original para ampliar as estratégias de busca e (iii) combinação das duas anteriores (Silva *et al.*, 2010).

Este artigo apresenta um estudo secundário com o objetivo de fazer uma atualização temporal da revisão realizada por (Geraldi e OliveiraJr, 2017a). Os autores identificaram na literatura, até maio/2017, 32 estudos primários focados em adaptações/aplicações de tipos de defeitos de software relacionados às técnicas/abordagens de inspeção para "fornecer taxonomias e classificações para orientar pesquisadores e profissionais que conduzem estudos neste tópico" (Geraldi e OliveiraJr, 2017a).

As técnicas de inspeção colaboram com orientações na detecção de defeitos. Para tal, o processo deve ser padronizado e não ambíguo para o artefato em revisão (IEEE, 2008). Logo, uma classificação de defeitos bem definida colabora para guiar os revisores em concentrar quais tipos de perguntas serão definidas pela técnica e garantir que os principais defeitos conhecidos na literatura sejam detectados (Alshazly *et al.*, 2014).

O estudo sistemático realizado ampliou os resultados encontrados em (Geraldi e OliveiraJr, 2017a) para o período de maio/2017 a maio/2020, possibilitando assim, análise e identificação de quais tipos de defeitos foram mais frequentemente utilizados e associados a técnicas de inspeção e artefatos de software nos últimos 3 anos. Para tal, foi mantido o protocolo executado pelos autores sem mudanças significativas.

A.1 Processo da Revisão Sistemática

Os principal objetivo deste estudo secundário é atualizar a revisão sistemática realizada por Geraldi e OliveiraJr (2017a), expandindo-a temporalmente. A revisão original

compreendeu os estudos publicados até maio de 2017. Este estudo, ampliou os resultados para maio de 2020, inserindo assim, artigos publicados nos últimos três anos.

Em relação à investigação e coleta dos dados, o objetivo deste estudo, e as questões de pesquisa, seguem sendo as mesmas de Geraldi e OliveiraJr (2017a):

> "(i) identificar na literatura os tipos de defeitos utilizados em ambientes / domínios e técnicas de inspeção de software; (ii) apresentar uma visão geral dos tipos de defeitos evidenciados empiricamente; e (iii) discutir estudos primários sobre técnicas de inspeção com tipos de defeitos".

As questões de pesquisa são:

- (RQ1). Quais tipos de defeitos foram levados em consideração pelas técnicas de inspeção de software?
- (RQ2). Que tipo de evidência as técnicas/abordagens de inspeção que adotam tipos de defeitos classificados fornecem?

A.1.1 Critérios de Inclusão e Exclusão

Os critérios de inclusão e exclusão foram definidos por Geraldi e OliveiraJr (2017a) para colaborar com a seleção dos trabalhos relevantes para serem lidos por completo. Tais critérios são:

Critérios de Inclusão

Para cada questão de pesquisa, foram definidos os critérios de inclusão, a seguir:

- RQ1. Estudos que apresentam tipos de defeitos relacionados a técnicas de inspeção de software:

  RQ1a. Estudos que apresentam ambientes ou domínios com tipos de defeitos aplicados; e

  RQ1b. Estudos que apresentam tipos de defeitos evidenciados empiricamente

- RQ2. Estudos que propõem técnicas ou abordagens de inspeção associadas a tipos de defeitos:

  RQ2a. Estudos que propõem técnicas de inspeção com tipos de defeitos classificados

  RQ2b. Estudos que documentam os resultados da avaliação

Critérios de Exclusão

Para cada questão de pesquisa, foram definidos critérios de exclusão, como segue:

- RQ1. Estudos que não apresentam tipos de defeitos relacionados a técnicas de inspeção de software:

  RQ1a. Estudos que não apresentam ambientes ou domínios com tipos de defeitos aplicados e

  RQ1b. Estudos que não apresentam tipos de defeitos evidenciados empiricamente

- RQ2. Estudos que não propõem técnicas de inspeção nem abordagens associadas a tipos de defeitos:

  RQ2a. Estudos que não propõem técnicas de inspeção nem utilizaram tipos de defeitos classificados; e

  RQ2b. Estudos que não documentam resultados de avaliação

Além dos critérios já definidos, Geraldi e OliveiraJr (2017a) estabeleceram outros critérios de para exclusão de artigos: (I) idioma dos estudos diferente do inglês; (ii) estudos em formatos diferentes de formatos de arquivo: PDF, DOC ou ODT; (iii) estudos que correspondem a artigos de opinião/filosóficos; (iv) estudos duplicados encontrados em mais de uma fonte de dados; (v) estudos indisponíveis, como por exemplo, estudo sem acesso gratuito; e (vi) estudos com menos de quatro páginas.

Além dos critérios de exclusão definidos por (Geraldi e OliveiraJr, 2017a), foram acrescentados nesta revisão mais dois:

- Estudos retornados do ano de 2017 e que são anteriores ao mês de maio.
- Estudos com técnicas de inspeção específicas para código fonte, pois o objetivo é identificar tipos de defeitos de software em artefatos de projeto.

A.1.2 Dados de extração

No processo de seleção final, após a leitura completa dos estudos, serão coletados alguns dados relevantes para análise desta revisão. Os metadados dados extraídos, são:

- Título: título do trabalho;
- Autores: nome dos autor(es) do estudo;

- Ano de publicação: ano em que o estudo foi publicado;
- Taxonomia de defeitos: tipos de defeitos de software utilizados no estudo;
  - Adaptado de: para taxonomia adaptadas, inclui o trabalho original da classificação.
- Artefato: os artefatos inspecionados.
- Técnica de inspeção: se houver, a técnica de inspeção utilizada/proposta no estudo;
  - Técnica base: para novas técnicas, se houver, técnica de inspeção que serviu como base;
  - Técnica comparada: se houver experimento, técnicas de inspeção utilizada nos estudos para comparar métricas.
- Base de dados: nome da fonte de dados que retornou o resultado;
- Local de publicação: nome do veículo em que foi publicado o estudo;
- Pacote experimental: o link, se houver um pacote experimental das técnicas de inspeção ou dos dados das avaliações;
- Tipo de publicação: o tipo de publicação do artigo, como por exemplo conferências, congressos, revistas e livros.
- Tipo de pesquisa: a pesquisa é enquadrada em um dos tipos de pesquisa pela classificação Wieringa *et al.* (2006), que será detalhada na próxima subseção.
- Visão geral: um resumo com as informações importantes do estudo;
- Palavras-chave: as palavras chaves do autor; e
- Resumo do autor: resumo retirado do estudo.

A.1.3 Esquema de classificação

Uma das informações extraídas após a leitura dos artigos selecionados para o formulário de extração é o tipo de publicação que a pesquisa se enquadra. Para esta revisão sistemática, a classificação dos tipos de pesquisa é a mesma de Geraldi e OliveiraJr (2017a): a classificação de Wieringa *et al.* (2006), que compreende no total 6 categorias:

- Pesquisa de avaliação: são trabalhos que investigam um problema ou a implementação de uma técnica na prática, ou seja, na indústria;
- Proposta de solução: trabalhos que propõe uma nova técnica ou contém uma revisão significativa de tecnologia;
- Pesquisa de validação: engloba propostas de soluções que ainda não foram empregadas na prática;
- Artigo filosófico: neste tipo de artigo, o autor dá um novo olhar sobre o tema e que podem ser explorado para pesquisas futuras;
- Documento de opinião: o autor dá sua opinião sobre o tema. ;e
- Documento de experiência pessoal: o artigo contém o relato do autor sobre sua experiência com a prática do tema ou ferramenta de pesquisa.

A.1.4 String de Busca

Como é uma atualização temporal da revisão sistemática de Geraldi e OliveiraJr (2017a), a String de Busca utilizada é a mesma dos autores, é proveniente das combinação das palavras-chave ("inspeção de software" e "tipo de defeito") e suas variações utilizando operadores lógicos.

Porém, ao aplicarmos a String de Busca na base Google Scholar, foi retornado mais de cem mil resultados, portanto, devido à limitação natural de tempo e recursos humanos, resolveu-se fazer uma alteração para não perder muitos artigos publicados e ser um número possível para leitura no tempo proposto.

Nesta revisão sistemática, temos então, duas Strings de Busca: uma geral para cinco bases de dados que serão nomeadas na sessão (1.7) e uma específica para o Google Scholar (Tabela 1.1).

Tabela 1.1: Strings de Busca

Base de Busca	Strings de Busca
String de Busca Geral	("software") AND (("inspection"AND ("technique"OR "activity"OR "strategy")) OR ("defect type"OR "type of defects"OR "defect detection"OR "requirements defect"OR "fault detection"))
String de Busca Google Scholar	(("software") AND (("inspection technique"OR "inspection activity"OR "inspection strategy")) ("defect type"OR "type of defects"OR "defect detection"OR "requirements defect"OR "fault detection")))

Fonte: o autor

A.1.5 Busca e seleção dos dados

O processo de busca desta revisão deu-se nos seguintes passos:

1. Seleção das fontes de dados.
2. Aplicação das strings de busca nas fontes de dados selecionadas na etapa anterior. cadeias de pesquisa definidas a bancos de dados digitais e tais mecanismos.
3. Seleção dos estudos em duas etapas: uma preliminar e uma seleção final após a leitura completas do artigos e aplicação dos critérios de inclusão e exclusão para filtrar esses estudos.
4. Para seguir o trabalho de Geraldi e OliveiraJr (2017a), os artigos selecionados foram categorizados seguindo a classificação utilizada por eles.
5. Com o apoio do formulário de extração, foram coletados dados relevantes dos artigos selecionados.
6. Os dados extraídos foram agregados, analisados e discutidos e apresentado nas próximas seções

A.1.6 Seleção dos dados

As bases de dados eletrônicas selecionadas para realizar a busca dos estudos primários foram: Biblioteca Digital da ACM, ELSEVIER, ScienceDirect, Google Scholar, IEEE Xplore e Scopus.

A String de busca geral foi adaptada de acordo com cada base de dados digital e ferramenta de busca do mecanismo. As modificações originais feitas por Geraldi

e OliveiraJr (2017a) sofreram algumas alterações para este trabalho (Tabela 1.2) se adaptando ao formato exigido de cada base de dados e respeitando o período que compreende esta revisão.

Para cada consulta nas bases eletrônicas foi aplicado diretamente na String de Busca ou na interface do mecanismo de busca o período de 2017 até maio de 2020, que corresponde ao mês em que foi realizada as consultas. Completando assim, uma atualização de 3 anos do trabalho de Geraldi e OliveiraJr (2017a).

Tabela 1.2: Strings de Busca por base de dados

Base de Busca	**Strings de Busca**
ACM	Title:(("software") AND (("inspection"AND ("technique"OR "activity"OR "strategy")) OR ("defect type"OR "type of defects"OR "defect detection"OR "requirements defect"OR "fault detection"))) or Abstract:(("software") AND (("inspection"AND ("technique"OR "activity"OR "strategy")) OR ("defect type"OR "type of defects"OR "defect detection"OR "requirements defect"OR "fault detection"))) or Keyword:(("software") AND (("inspection"AND ("technique"OR "activity"OR "strategy")) OR ("defect type"OR "type of defects"OR "defect detection"OR "requirements defect"OR "fault detection")))
IEEE Xplore	("software") AND (("inspection"AND ("technique"OR "activity"OR "strategy")) OR ("defect type"OR "type of defects"OR "defect detection"OR "requirements defect"OR "fault detection"))
Compendex	("software") AND (("inspection"AND ("technique"OR "activity"OR "strategy")) OR ("defect type"OR "type of defects"OR "defect detection"OR "requirements defect"OR "fault detection"))
ELSEVIER ScienceDirect	("software") AND (("inspection technique"OR "inspection activity"OR "inspection strategy") OR ("defect type"OR "type of defects"OR "defect detection"OR "requirements defect"OR "fault detection")) AND LIMIT-TO(topics, "fault,system,defect,fault detection,model,control,technology,process")
Google Scholar	(("software") AND (("inspection technique"OR "inspection activity"OR "inspection strategy")) ("defect type"OR "type of defects"OR "defect detection"OR "requirements defect"OR "fault detection")))

Scopus	("software")AND (("inspection"AND ("technique"OR "activity"OR "strategy")) OR ("defect type"OR "type of defects"OR "defect detection"OR "requirements defect"OR "fault detection")) AND (LIMIT-TO(PUBYEAR,2017) OR LIMIT-TO(PUBYEAR,2018) OR LIMIT-TO(PUBYEAR,2019) OR LIMIT-TO (PUBYEAR,2020)) AND (LIMIT-TO(SUBJAREA,"COMP") OR LIMIT-TO (SUBJAREA,"ENGI")) AND (LIMIT-TO(LANGUAGE,"English")) AND (LIMIT-TO(SRCTYPE,"j") OR LIMIT-TO(SRCTYPE,"p") OR LIMIT-TO(SRCTYPE,"k") OR LIMIT-TO(SRCTYPE,"b")) AND (EXCLUDE(SUBJAREA,"MATH") OR EXCLUDE(SUBJAREA,"PHYS") OR EXCLUDE(SUBJAREA,"MATE") OR EXCLUDE(SUBJAREA,"ENER") OR EXCLUDE (SUBJAREA,"DECI") OR EXCLUDE(SUBJAREA,"SOCI") OR EXCLUDE(SUBJAREA,"BUSI") OR EXCLUDE(SUBJAREA,"CHEM") OR EXCLUDE(SUBJAREA,"MEDI") OR EXCLUDE(SUBJAREA,"EART") OR EXCLUDE(SUBJAREA,"BIOC") OR EXCLUDE(SUBJAREA,"CENG") OR EXCLUDE(SUBJAREA,"ENVI") OR EXCLUDE(SUBJAREA,"AGRI") OR EXCLUDE(SUBJAREA,"HEAL") OR EXCLUDE(SUBJAREA,"ARTS") OR EXCLUDE(SUBJAREA,"NEUR") OR EXCLUDE(SUBJAREA,"PSYC") OR EXCLUDE(SUBJAREA,"ECON") OR EXCLUDE(SUBJAREA,"MULT") OR EXCLUDE(SUBJAREA,"PHAR") OR EXCLUDE(SUBJAREA,"DENT") OR EXCLUDE(SUBJAREA,"IMMU")) AND (LIMIT-TO (EXACTKEYWORD, "Fault Detection") OR LIMIT-TO (EXACTKEYWORD, "Software Testing") OR LIMIT-TO (EXACTKEYWORD, "Software Engineering") OR LIMIT-TO (EXACTKEYWORD,"Computer Software") OR LIMIT-TO (EXACTKEYWORD, "Fault Tolerance") OR LIMIT-TO(EXACTKEYWORD, "Testing") OR LIMIT-TO (EXACTKEYWORD, "Defects")) AND (LIMIT-TO(EXACTKEYWORD,"Failure Analysis") OR LIMIT-TO (EXACTKEYWORD, "Software Design") OR LIMIT-TO (EXACTKEYWORD, "Fault Diagnosis") OR LIMIT-TO (EXACTKEYWORD, "Inspection") OR LIMIT-TO (EXACTKEYWORD, "Errors") OR LIMIT-TO(EXACTKEYWORD,"Program Debugging") OR LIMIT-TO (EXACTKEYWORD,"Software Quality") OR LIMIT-TO (EXACTKEYWORD, "Regression Testing") OR LIMIT-TO (EXACTKEYWORD, "Fault Tolerant Computer Systems") OR LIMIT-TO (EXACTKEYWORD, "Quality Control") OR LIMIT-TO (EXACTKEYWORD, "Verification") OR LIMIT-TO (EXACTKEYWORD, "Model Checking") OR LIMIT-TO (EXACTKEYWORD, "Error Detection") OR LIMIT-TO(EXACTKEYWORD, "Fault Localization") OR LIMIT-TO(EXACTKEYWORD, "Maintenance") OR LIMIT-TO(EXACTKEYWORD, "Software Systems") OR LIMIT-TO (EXACTKEYWORD,"Test Case") OR LIMIT-TO(EXACTKEYWORD, "Defect Detection") OR LIMIT-TO (EXACTKEYWORD, "Defect Prediction") OR LIMIT-TO (EXACTKEYWORD, "Fault-tolerant") OR LIMIT-TO (EXACTKEYWORD, "Error Correction") OR LIMIT-TO(EXACTKEYWORD,"Information Systems") OR LIMIT-TO(EXACTKEYWORD, "Systems Analysis"))

Após realizado a busca dos artigos primários. Foi utilizada a ferramenta StArt (State of the Art through Systematic Review) para dar suporte ao gerenciamento das referências bibliográficas retornadas das consultas e ao processo de filtragem destes estudos. O processo de seleção dos artigos deu-se em três etapas (Tabela 1.3):

- Filtro #1: Da aplicação das Strings de Busca foram recuperados 2897 estudos primários, dos quais, 713 eram duplicados, totalizando assim 2184 artigos.
- Filtro #2: A partir dos estudos primários recuperados do Filtro #1, foram selecionados os potenciais estudos por meio da leitura do título, resumo e introdução, e avaliação da potencialidade do estudo considerando os critérios de inclusão e exclusão nas partes lidas. Deste Filtro, foram selecionados 38 artigos.
- Filtro #3: Os 38 potenciais estudos primários foram filtrados por meio da leitura completa e avaliação com a aplicação dos critérios de inclusão e exclusão, e definição das questões de pesquisa. Deste filtro, foram selecionados 15 artigos (Tabela Tabela 1.4) que tiveram os dados agregados, analisados e discutidos.

Tabela 1.3: Processo de seleção dos artigos

Base de dados	**FILTRO #1**	**FILTRO #2**	**FILTRO #3**
ACM Digital Library	126	9	3
IEEE Xplore	705	2	2
ELSEVIER ScienceDirect	130	0	0
Compendex	742	8	6
Scopus	754	9	3
Google Scholar	440	10	1
TOTAL (COM DUPLICADOS)	2897	38	15
DUPLICADOS	713	0	0
TOTAL	2184	38	15

Fonte: o autor

Os 15 estudos primários finais selecionados após o Filtro 3# são apresentados na Tabela 1.4 com os dados retirados do formulário de extração: autor, título, ano, base de dados, local de publicação, tipo de pesquisa e de publicação.

Tabela 1.4: Conjunto final de estudos selecionados (Filtro #3)

RQ	**Autores**	**Título**	**Ano**	Id	Eficiênc	Eficáia	Efetivide
RQ1	Carneiro *et al.* (2017)	Investigating the influence of inspector learning styles on design inspections: Findings of a quasi-experiment	2017	CX	CIbSE	Pesquisa de validação	Evento
RQ1	Geraldi e OliveiraJr (2017b)	Towards Initial Evidence of SMartyCheck for Defect Detection on Product-Line Use Case and Class Diagrams.	2017	GS	J. Softw	Pesquisa de validação	Periódico
RQ1	Neto Gracioli *et al.* (2019)	Using model scoping with expected model elements to support software model inspections: Results of a controlled experiment	2019	CX	ICEIS	Pesquisa de validação	Evento
RQ1	Granda *et al.* (2019)	A Metrics-driven Inspection Framework for Model Transformations	2019	SC	CIbSE	Proposta de solução	Evento
RQ1	Ma (2017)	An approach to improve the quality of object-oriented models from novice modelers through project practice	2017	CX	Front. Comput. Sci.	Proposta de solução	Periódico
RQ1	Santos *et al.* (2019)	Software inspections: comparing a formal method based with a classical reading methodology	2019	CX	IJCAT	Pesquisa de validação	Periódico
RQ1	Sousa *et al.* (2019)	REM4DSPL: A requirements engineering method for dynamic software product lines	2019	SC	SBQS	Pesquisa de avaliação	Evento
RQ1	Tianual e Pohthong (2019)	Defects Detection Technique of Use Case Views during Requirements Engineering	2019	AC	ICSCA	Pesquisa de validação	Evento
RQ1	Tokdemir e Metin (2017)	Understanding BPMN through defect detection process	2017	CX	BMSD	Pesquisa de avaliação	Evento
RQ2	Damian *et al.* (2018)	ComD2: Family of techniques for inspecting defects in models that affect team communication	2018	CX	SEKE	Pesquisa de avaliação	Evento
RQ2	de Souza *et al.* (2019b)	The first version of $SCENARI_{OT}CHECK$	2019	AC	SBES	Pesquisa de validação	Evento
RQ2	de Souza *et al.* (2019a)	An IoT-Based Scenario Description Inspection Technique	2019	AC	SBQS	Pesquisa de validação	Evento
RQ2	Vaish e Sharma (2018)	Semi-Automated System Based Defect Detection in Software Requirements Specification document	2018	IE	UPCON	Pesquisa de validação	Evento
RQ2	Villamizar *et al.* (2019)	An Approach for Reviewing Security-Related Aspects in Agile Requirements Specifications of Web Applications	2019	IE	RE	Pesquisa de validação	Evento
RQ2	Zhang *et al.* (2018)	Search and similarity based selection of use case scenarios: An empirical study	2018	SC	Empir Soft Eng	Pesquisa de validação	Periódico

Fonte: o autor

A.1.7 Visão Geral do Estudo Secundário

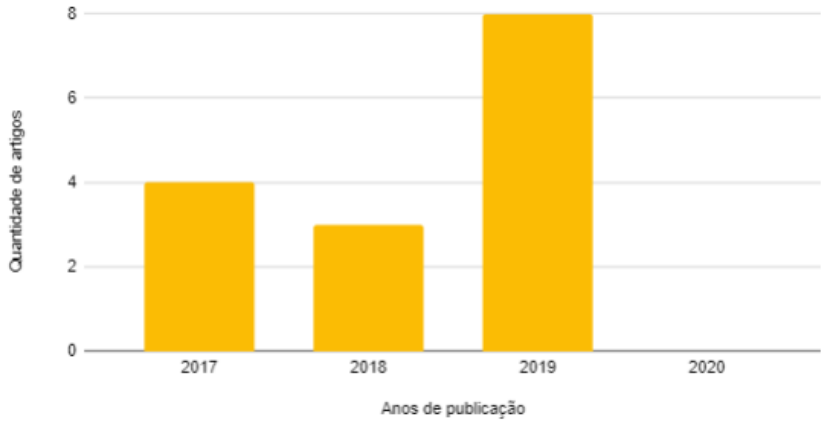
Esta seção apresenta uma visão geral do estudo secundário com um resumo das principais informações coletadas após a fase de extração dos dados. Os dados serão relacionados com dados encontrados na revisão sistemática de Geraldi e OliveiraJr (2017a). Alguns assuntos mais importantes detectados após a leitura completa dos estudos foram agrupados nas subseções subsequentes, como as taxonomias de defeitos e técnicas de inspeção.

Compendex foi a base de dados com mais artigos selecionados ao final desta atualização, totalizando 6 dos 15 estudos primários. Em seguida, Scopus e ACM Digital Library com 3 estudos cada. Já o ELSEVIER ScienceDirect não retornou nenhum artigo que atendesse às questões de pesquisa deste estudo. Até maio/2017 a base de dados com mais trabalhos encontrados foi IEEE Xplore e ACM Digital Library e Compendex Geraldi e OliveiraJr (2017a).

Em relação ao local de publicação, no trabalho de Geraldi e OliveiraJr (2017a) o local de publicação com mais artigos publicados selecionados foi o International Conference on Software Engineering and Knowledge Engineering (SEKE), com 3 estudos. Já no período de maio/2017 a maio/2020, o SBQS (Simpósio Brasileiro de Qualidade de Software) e o CIbSE (Ibero-American Conference on Software Engineering), tiveram duas publicações cada, enquanto as outras conferências e periódicos tiveram apenas uma cada.

A Figura 1.1 contém a distribuição temporal dos 15 trabalhos analisados nesta revisão sistemática. Até o mês de maio, não foram encontrados nenhum estudo em 2020, isto pode mudar no percorrer do ano conforme forem acontecendo as conferências e publicações em revistas.

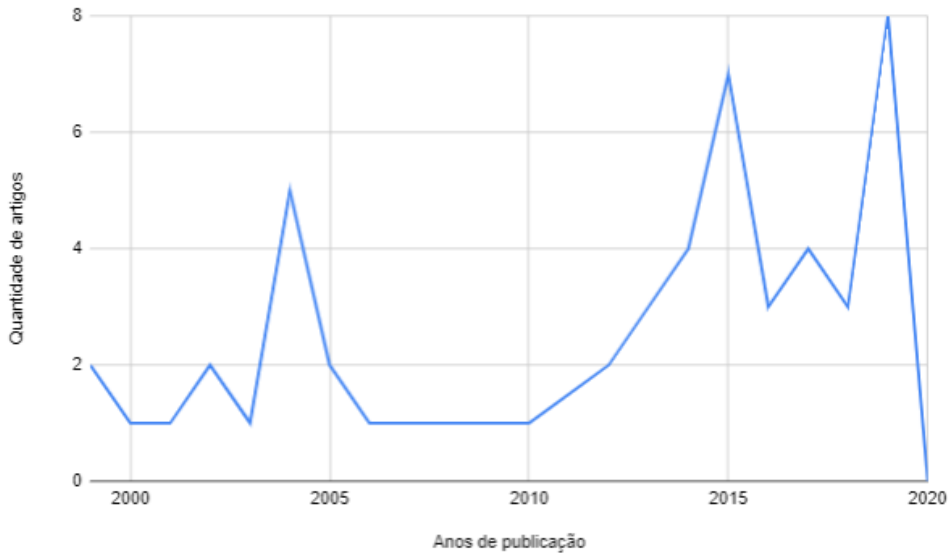
Figura 1.1: Quantidade de estudos primários por ano de publicação



Fonte: o autor

Ao adicionarmos os anos de publicação desta atualização com o trabalho original de Geraldi e OliveiraJr (2017a), é possível notar na distribuição total da Figura 1.2 que houve um aumento no interesse dos pesquisadores em utilizar uma classificação de defeitos relacionada a inspeção de software nos últimos anos, com dois picos em 2015 e 2019 com 7 e 8 trabalhos, respectivamente.

Figura 1.2: Distribuição temporal dos estudos primários: Geraldi e OliveiraJr (2017a) e atualização



Fonte: o autor

Dos 15 trabalhos do conjunto final selecionados, apenas 2 dos trabalhos não foram evidenciados empiricamente, os outros 13 foram documentados por meio de experimentos ou estudos empíricos, gerando assim, evidências mais precisas em relação a classificação de defeitos para serem utilizados e adaptados em novos estudos.

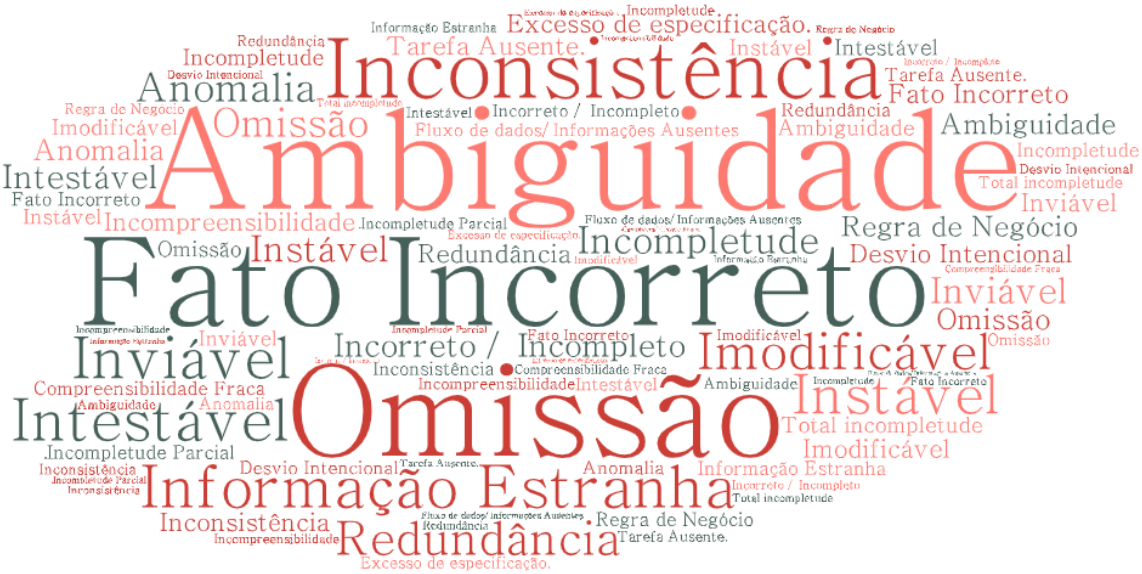
Apesar do grande número de estudos evidenciados empiricamente, poucos se preocuparam com o compartilhamento das informações, tanto das técnicas de inspeção utilizadas como dos dados obtidos nos experimentos e das análises feitas.

A maioria dos estudos primários selecionados é de Pesquisa de Validação, RQ2 foi respondida levando em consideração 9 ocorrências do tipo de pesquisa, nas quais a maioria dos estudos é de Pesquisa de Validação e Proposta de Solução, com vários experimentos que apoiam técnicas/abordagens de inspeção.

Respondendo o principal objetivo da pesquisa em relação às taxonomias de defeitos encontradas nos estudos primários, há uma predominância nos tipos de defeitos de

Ambiguidade, Omissão, Fato Incorreto e Inconsistência como é possível observar na nuvem de palavras Figura 1.3. Estes dados sobre os tipos de defeitos relacionados a temas como técnicas de inspeção e artefatos serão detalhados e discutidos na próxima seção.

Figura 1.3: Nuvem de palavras dos tipos de defeitos encontrados



Fonte: o autor

A.2 Discussões dos estudos selecionados

As próximas subseções respondem às questões de pesquisa desta revisão divididos em tópicos com informações consideradas importantes de serem analisadas a partir dos metadados extraídos dos estudos primários selecionados em relação aos tipos de defeitos identificados, técnicas de inspeção, artefatos inspecionados e compartilhamento dos materiais usados nos estudos.

A.2.1 Visão Geral dos Estudos Selecionados

Nesta subseção é apresentada uma visão geral obtida após a leitura dos estudos selecionados, importante para compreender o que está sendo trabalhado na área de inspeção de software e as taxonomias utilizadas.

ComD2 (*Communication between Designers and Developers*) Damian *et al.* (2018) é uma família de técnicas de inspeção para detectar defeitos que impactam a comunicação entre equipes de desenvolvimento de software. É formada por lista de verificações

específicas para cada um dos diagramas UML suportados pela técnica: classes, atividades e máquina de estado. Os itens de verificação foram criados com base das 4 máximas do Princípio Cooperativo de Grice: Qualidade, quantidade, relevância e modo, pois, segundo os autores, quando essas máximas não são respeitadas no modelo eles podem causar defeito no software.

A técnica $Scenari_{ot}Check$, é uma técnica baseada em *checklist* para detectar defeitos em descrições de cenários baseados em IoT gerados pela técnica de especificação SCENARIOT (de Souza *et al.*, 2019a,b).

Villamizar *et al.* (2019) propôs uma técnica baseada em *checklists* para inspecionar aspectos relacionados a segurança em especificações de requisitos ágeis de aplicativos da *web* em formato de história do usuário. Além dos tipos de defeito, ela cobre aspectos de segurança de integridade, confidencialidade e identificação e autenticação.

A Model Scoping proposta por Neto Gracioli *et al.* (2019) define um escopo de modelo com base em uma parte eleita do documento de referência que funciona como um filtro para mostrar apenas elementos relevantes de grandes modelos. Estes elementos esperados do modelo são usados para definir o escopo e orientar a inspeção.

O artigo de Santos *et al.* (2019) compara as técnicas SOLIMVA 3.0 (método de verificação formal) e a leitura vertical da OORT para diagramas UML diferentes (sequência, máquina de estado e atividade) em um sistema de transição único para verificação do modelo. Os autores concluem que as técnicas podem ser usadas como completo para inspeção de software, já que diferenciam-se na detecção de certas classes de defeitos.

Tokdemir e Metin (2017), por meio da técnica de leitura vertical, analisaram quais são os tipos de defeitos mais detectados por usuários iniciantes em diagramas BPMN (*Business Process Modeling Notation*) comparando com um cenário dado. Como resultado, obtiveram evidências que defeitos de Tarefa Ausente e Fluxo de dados/Informações Ausentes são mais difíceis de serem detectados que a classe de Incorreto / Incompleto.

Carneiro *et al.* (2017) investigam a influência dos estilos de aprendizagem do inspetor e do time na eficiência e eficácia na detecção de defeitos. Para isso, fizeram um quasi-experimento para inspecionar um documento de projeto utilizando a técnica de Leitura Baseada em Defeitos. Os autores observaram no final do estudo que as dimensões de estilo de aprendizagem não influenciam a eficiência e eficácia do inspetor e que a atividade de inspeção parece ser afetada mais por outros fatores como: técnicas de detecção de defeitos.

Geraldi e OliveiraJr (2017b) avaliaram a técnica *SMartyCheck*, baseada em lista de verificação para modelos de casos de uso e de classe *SMarty* para gerenciamento de variabilidades em Linhas de Produto de Software. Foram realizados dois estudos, uma

avaliação qualitativa e uma quantitativa que mostrou que a técnica é viável em termos de eficiência, eficácia e eficácia para tais modelos.

Vaish e Sharma (2018) propõem uma abordagem semi-automatizada para inspeção de documentos de requisitos baseada em regras, para que elas correspondam ao conteúdo de texto dos documentos. A abordagem proposta foi comparada em um estudo experimental com as técnicas PBR, DBR e CBR e mostrou melhores resultados para todos os tipos de defeitos.

A REM4DSPL é um método de engenharia de requisitos para Linhas de Produtos de Software Dinâmicas (DPSL) para engenheiros de domínio. O método consiste em três fases principais: requisitos, modelagem de requisitos e gerenciamento de variabilidades. Na fase de gerenciamento de variabilidades é garantida a consistência do modelo de características DPSL, por meio da inspeção utilizando uma adaptação da técnica FMCheck de acordo com a classificação de feature (Sousa *et al.*, 2019).

Zhang *et al.* (2018) avalia empiricamente a abordagem S3RUCM que seleciona os casos de uso descritos no modelo RUCM (modelagem de casos de uso restrito baseada em linguagem natural) em diversos cenários. O terceiro processo da abordagem é a inspeção manual, que é apoiada por um subconjunto de cenários produzidos pela técnica. Os autores não definem uma técnica específica, mas, mostram diversas técnicas de inspeção de requisitos que podem ser usados com a abordagem.

Tianual e Pohthong (2019) propõe uma técnica e a ferramenta de suporte para detectar defeitos em visualizações de casos de uso durante sua criação por meio de tabelas de decisão e relação. Resultados preliminares mostraram que a ferramenta proposta é mais eficiente que a detecção manual de defeitos.

Granda *et al.* (2019) apresentam uma estrutura de inspeção semi-automatizada orientada por métricas. Foram definidos um conjunto de métricas básicas e derivadas, que podem ser calculadas e usadas para inspecionar o modelo de destino em comparação com o modelo esperado (oráculo), elas foram úteis para identificar e localizar anomalias nas regras de transformação, o que melhorou a eficácia do transformação do modelo sob inspeção.

A.2.2 Taxonomias de defeitos de software

Nesta seção são apresentados os tipos de defeitos relacionados à inspeção de software apresentados pelos 13 estudos aceitos neste trabalho e ao final, uma análise com as classes de defeitos mais abordadas nesses estudos.

A taxonomia proposta por (Shull, 1998) contém 5 classes de defeitos: Omissão, Ambiguidade, Fato Incorreto, Inconsistência e Informação Estranha. Três trabalhos selecionados utilizaram adaptações desta taxonomia: Carneiro *et al.* (2017), Neto Gracioli *et al.* (2019), Villamizar *et al.* (2019).

Na abordagem de (Neto Gracioli *et al.*, 2019), a Model Scoping, a classificação de Shull (1998) foi adaptada utilizando apenas as classes de Ambiguidade, Fato incorreto, Omissão e Informação estranha. A ausência da classe de Inconsistência foi justificada pelos autores por não permitir a comparação entre modelos para identificar os defeitos.

Já no trabalho de Carneiro *et al.* (2017) mencionam que os tipos de defeitos foram "*definidos pela equipe de pesquisa em colaboração com especialistas em inspeção com base em defeitos típicos nesse tipo de aplicação*", subentende-se, assim, que a justificativa da classe de Inconsistência não ter sido aplicada no trabalho é por não ser comum para o tipo de aplicação utilizada no experimento. Ao contrário das classes de Omissão, Ambiguidade, Fato Incorreto e Informação Estranha utilizadas.

No estudo de Villamizar *et al.* (2019) foram mantidos da taxonomia original de Shull (1998) as classes de Omissão, Fato Incorreto, Inconsistência e Ambiguidade. Os autores excluíram o tipo de defeito de Informação Estranha, por, segundo eles, a classificação ser relacionada à especificação de requisitos que não eram necessários ao trabalho. Além de tipos de defeitos, a abordagem proposta pelos autores cobre os aspectos de segurança: integridade, confidencialidade e identificação e autenticação.

A técnica $Scenari_{ot}Check$ em suas duas versões (de Souza *et al.*, 2019a,b) utilizou a taxonomia de Travassos *et al.* (1999): Omissão, Fato Incorreto, Inconsistência, Ambiguidade e Informação Estranha. Como mencionado no trabalho de Shull (1998), é possível perceber que a classificação é similar ao trabalho de Basili *et al.* (1996b) que foi utilizada no trabalho de Travassos *et al.* (1999).

No trabalho de Geraldi e OliveiraJr (2017b) a classificação de defeitos definidas para a técnica *SMartyCheck* foram: Inconsistência, Fato Incorreto, Ambiguidade, Informação Estranha, Regra de Negócio, Imodificável, Omissão, Anomalia, Instável, Inviável e Desvio Intencional. Tais defeitos, foram adaptados de Travassos *et al.* (1999) e IEEE (2008).

Sousa *et al.* (2019) recomendam que os erros no modelo de recurso sejam classificados em: Omissão, Fato Incorreto, Ambiguidade, Inconsistência e Informação Estranha.

Zhang *et al.* (2018) simula a abordagem proposta com todas as classes de defeitos encontradas em Zhang *et al.* (2018): Incorreto, Incompletude, Inconsistência, Ambiguidade, Incompreensibilidade, Intestável, Imodificável, Inviável e Excesso de Especificação. Em Tokdemir e Metin (2017) a classificação adotada para defeitos para diagramas de BPMN foram: Tarefa Ausente, Fluxo de dados/Informações Ausentes e Incorreto / Incompleto.

Em sua comparação entre os métodos SOLIMVA e OORT (Santos *et al.*, 2019) os tipos de defeitos analisados foram: Total incompletude, Inconsistência (Incorreto), Ambiguidade, Incompletude Parcial, Inconsistência (Informação Extra) e Compreensibilidade Fraca. Na abordagem proposta por Vaish e Sharma (2018) para inspeção de documento de requisitos a taxonomia utilizada foi a de Walia e Carver (2009): Omissão, Incorreto e não conformidade com padrão.

A família de técnicas ComD2 Damian *et al.* (2018) utilizou a taxonomia de Granda *et al.* (2019) que contém os tipos de defeito: Omissão, Fato Incorreto, Inconsistência, Ambiguidade, Informação Estranha e Redundância. Além disso, cada item de verificação das técnicas são relacionadas as dimensões de representação que foi afetada pelos defeitos nos modelos, sendo estas: Sintáticas, Semânticas e Pragmáticas.

Tianual e Pohthong (2019) cita uma taxonomia de defeitos para modelos de casos de uso, mas, na investigação preliminar do estudo usam apenas fluxo incorreto entre casos de uso e meta não alcançada. Já Granda *et al.* (2019) menciona os defeitos de regras de transformação ausentes, incorretas ou desnecessárias para a transformação de modelos.

Ma (2017) lista os principais tipos de defeitos encontrados em modelos de casos de uso, de classe e de sequência, para isso, ela classifica os tipos defeitos sob aspectos de qualidade Semântica, Sintática e Pragmática. Como por exemplo, em diagramas de classe sob o aspecto de Sintática, os típicos defeitos são: Descrição da etapa incompleta e Descrição incorreta da etapa.

Ao analisar conjunto de estudos selecionados evidenciados empiricamente neste trabalho (13 no total) é possível observar que apesar de mudarem algumas das classificações em quantidade de vezes em que aparecem (Tabela 1.5), assim como na revisão realizada por Geraldi e OliveiraJr (2017a), os erros mais recorrentes são: Ambiguidade, Omissão , Fato Incorreto e Inconsistência.

A predominância destes tipos de defeitos podem estar relacionadas às adaptações das classificações similares dos estudos de Shull (1998) e (Basili *et al.*, 1996b) por serem de fácil adaptação para diversas técnicas de inspeções e artefatos. Apesar da frequência não há um esquema unificado para o mesmo artefato, o que pode estar relacionado aos domínios que foram aplicados, considerando assim, os defeitos mais recorrentes em um tipo de sistema.

Tabela 1.5: Tipos de defeitos mais encontrados

Tipos de defeitos	**(Geraldi e OliveiraJr, 2017a)**	**Revisão atual**
Ambiguidade	23	9
Inconsistência	19	7

Fato Incorreto	14	8
Omissão	14	8

Fonte: o autor

A.2.3 Técnicas de Inspeção

É importante ressaltar que o foco deste trabalho é identificar os tipos de defeitos relacionados a inspeção de software, por isso, ele não engloba todas as técnicas de inspeção publicadas no período de maio/2017 a maio/2020 retornadas pela String de Busca, mas sim, as técnicas que foram relacionadas à alguma taxonomia de defeitos. Dos 15 trabalhos selecionados, 13 possuem técnicas de inspeção relacionadas.

Assim como na revisão sistemática de Geraldi e OliveiraJr (2017a) a técnica mais frequentemente utilizada e adaptada nos últimos três anos foi a CBR que por meio de itens de verificação (*checklist)* colabora com o inspetor na detecção de defeitos. A CBR foi encontrada em 7 dos 13 trabalhos selecionados.

Depois da CBR, as técnicas de inspeção que mais apareceram nos estudos primários foram a *Ad hoc*, utilizada em 5 trabalhos e como comparação nos experimentos realizados, e a DBR com 2 trabalhos. Outras técnicas apareceram uma única vez.

A.2.4 Artefatos

Como o objetivo do trabalho é detectar as técnicas de leitura e a taxonomia adotada para inspeção de software, os trabalhos considerados envolvem apenas artefatos de projeto e não código.

Diagramas UML foram os artefatos mais detectados nesta revisão, estão presentes em 6 de 15 trabalhos. Em especial o diagrama de classes, que aparece nos trabalhos de Damian *et al.* (2018), Carneiro *et al.* (2017),Neto Gracioli *et al.* (2019) e Geraldi e OliveiraJr (2017b). Além dos diagramas de classes, os estudos mencionam os diagramas de casos de uso, estados, sequência e atividades.

Dois trabalhos usaram abordagens para inspeção de especificação de requisitos: Villamizar *et al.* (2019) que detecta defeitos relacionados à segurança nas especificações de requisitos ágeis de aplicativos da web, e Vaish e Sharma (2018) que utiliza uma abordagem semi-automatizada.

Artefatos específicos como diagramas BPMN, modelo de recurso DSPL e cenários baseados em IoT foram também utilizados nos estudos primários. Como também, um

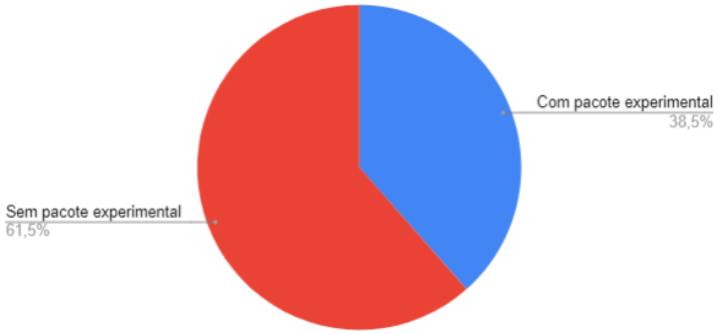
exemplo de aplicabilidade na transformação de modelo de análise comunicacional para modelo de teste do CoSTest (Granda *et al.*, 2019).

A.2.5 Open Science

Uma preocupação atual é o compartilhamento dos artefatos utilizados na pesquisa. O *Open Science* é movimento para disponibilizar publicamente todos os artefatos de pesquisa, aumentando assim a transparência e a reprodutibilidade do processo científico (Fernández *et al.*, 2019).

Dos 13 trabalhos evidenciados empiricamente, apenas 5 tiveram a preocupação de compartilhar os artefatos de pesquisa (aproximadamente 61% - Figura 1.4), o que representa um número muito baixo ainda quando comparado aos benefícios que se pode ter com o compartilhamento dos dados.

Figura 1.4: Artigos que seguiram os princípios de Open Science



Fonte: o autor

Os benefícios incluem uma maior confiabilidade e transparência dos dados das avaliações empíricas que foram analisados, como também no reúso dos artefatos para novas pesquisas e adaptações para novas aplicações e ambientes, em especial, para a reprodução dos estudos com os materiais experimentais originais.

Dos 5 trabalhos que colocaram o *link* do pacote experimental, apenas dois estavam completos: continham todos os materiais usados no experimento, como por exemplo termos de consentimento e as tarefas, além dos dados e análise dos resultados. Um dos trabalhos não tinha os dados do experimento e dois compartilharam apenas as informações da técnica de inspeção utilizada.

Destes artigos selecionados, é possível observar que apesar de haverem pesquisadores adeptos ao *Open Science*, há ainda um longo caminho de mudança de comportamento e cultura dos pesquisadores na comunidade de Engenharia de Software para que comecem

a realizar o compartilhamento dos dados da pesquisa como já fazem em outras áreas do conhecimento.

A.2.6 Considerações finais dos estudos selecionados

Ao acrescentarmos os dados de ano de publicação obtidos do trabalho de Geraldi e OliveiraJr (2017a) é possível notar um aumento dos estudos nos últimos anos, principalmente nos picos de 2015 e 2019, da preocupação dos pesquisadores em definir a taxonomia que abrange o estudo, o que facilita na hora da inspeção para guiar o inspetor "o que" ele deve procurar e dar segurança em focar nos defeitos mais presentes no domínio e artefatos específicos.

Outro dado motivador encontrado nesta revisão é a quantidade de estudos que foram avaliados empiricamente, sejam por experimentos ou estudos de viabilidade. Esta cultura, colabora com a Engenharia de Software para a verificação e validação das teorias, melhorando assim todo o processo, bem como o resultado final e a evolução/adaptação futura da pesquisa e especificamente aqui, das técnicas de inspeção e taxonomias adotadas.

Em contramão ao aumento dos estudos evidenciados empiricamente, está o compartilhamento dos materiais utilizados nas avaliações. O que acaba por dificultar futuras reproduções e adaptações das técnicas/abordagens de inspeção retornadas nesta revisão.

Apesar da predominância de alguns tipos de defeitos, não há um esquema unificado para o mesmo tipo de artefato, ou seja, há uma certa variação de tipos de defeitos para o mesmo artefato, o que pode estar relacionado aos domínios que foram aplicados, considerando assim, os defeitos mais encontrados em um tipo de sistema, melhorando assim, a qualidade da revisão, com um guia mais específico e direto.

O estudo de Geraldi e OliveiraJr (2017a) mencionou algumas lacunas encontradas pelos autores, entre elas, artefatos de engenharia de software que não eram inspecionados por nenhuma técnica, como por exemplo: diagramas UML de SPL de componente e diagramas BPM. Ao final desta atualização, pode-se notar que apesar do avanço em números de trabalhos, esta lacuna persiste. Faltando ainda, artefatos a serem considerados pelas técnicas de inspeção que contenham uma taxonomia de defeitos bem definidas.

Outras lacunas encontradas pelos autores e que não houveram grandes avanços nos últimos 3 anos é carência na proposta de novas taxonomias ou adaptações de tipos de defeitos importantes a serem considerados e da falta de artefatos de domínios específicos.

Ainda que o foco desta revisão sistemática não seja puramente encontrar técnicas de inspeção de software, foi possível observar um número alto de trabalhos que utilizaram a

CBR, seja talvez, por ser uma das técnicas principais mais conhecida e utilizada, como também, da facilidade em relacionar o tipo de defeito ao item verificação.

Assim, aumentar o número de técnicas de inspeção que se preocupem com estes fatores é de extrema importância para uma base mais sólida da área da pesquisa em inspeção de software e de uma maior garantia da eficiência da técnica para ser levada a indústria, visto que a grande maioria dos trabalhos deste estudo completo são ainda em ambiente acadêmico.

A.2.7 Ameaças a validade

Apesar de seguirmos rigorosamente o protocolo para extração e seleção dos dados, há ameaças a validade, como o risco de interpretacões pessoais e o total de base eletrônicas usadas, pois, mesmo sido escolhidas pela sua importância para a Engenharia de Software, quanto mais fontes, mais estudos poderão ser selecionados. Além disso, não se pode garantir que todos os estudos relacionados a string foram recuperados devido a precisão dos mecanismos de buscas. Logo, podem ter ficado estudos importantes fora desta revisão.

Foi citado por (Geraldi e OliveiraJr, 2017a) que palavras-chaves genéricas retornam estudos primários mais importantes. Porém, após a condução desta revisão, foram selecionados apenas 0,07% dos trabalhos, resultando em esforço desnecessário para um objetivo muito específico com o retorno de trabalhos de diversas áreas de pesquisa, como por exemplo: inspeção de defeitos em placas fotovoltaicas. Portanto, aprimorar as palavras-chaves podem gerar outros estudos secundários para diversas áreas.

A.3 Conclusão

Este estudo secundário foi realizado para atualizar temporalmente o estudo de (Geraldi e OliveiraJr, 2017a), acrescentando trabalhos publicados no período de maio/2017 a maio/2020, com o objetivo de identificar na literatura estudos primários com tipos de defeitos no contexto de inspeção de software.

Os resultados obtidos são motivadores ao observar a crescente dos trabalhos nos últimos anos que se encaixam na questão de pesquisa adotada neste estudo. Mostra a preocupação dos pesquisadores em tornar a técnica mais completa e eficiente para os inspetores de artefatos de software ao adotarem taxonomias de defeitos para dar suporte às abordagens ou ainda, como no trabalho de (Ma, 2017) listar os tipos de defeitos mais encontrados em um artefato, para que possam ser considerados pelos demais pesquisadores.

Apesar do cenário motivador descritos em números crescentes, há ainda lacunas já relatadas por (Geraldi e OliveiraJr, 2017a) em relação a quantidade de tipos de artefatos que são inspecionados e de domínios abordados, além da baixa diversidade das técnicas de inspeção, falta de proposta de novas taxonomias ou adaptações e do compartilhamento dos dados de pesquisa.

Espera-se com este estudo, atualizar os pesquisadores com trabalhos mais recentes no contexto de defeitos de software para inspeção, colaborando com trabalhos futuros e adaptações das técnicas e tipos de defeitos apresentados, expandindo as áreas de pesquisa até as lacunas que foram encontradas nesta revisão e (**?**) para uma base sólida de pesquisa e de maior garantia da eficiência da técnicas para serem levadas a indústria.

Para atualizações futuras, sugere-se que haja uma revisão nas palavras-chaves para diminuir o esforço de seleção dos artigos, visto que a porcentagem de artigos aproveitados é muito baixa devido a generalidade das palavras-chaves que engloba muitas áreas de pesquisa que não se encaixam nas motivações desta revisão.

Apêndice B: Diretrizes para Identificação e Representação de Variabilidades

B.1 Considerações Iniciais

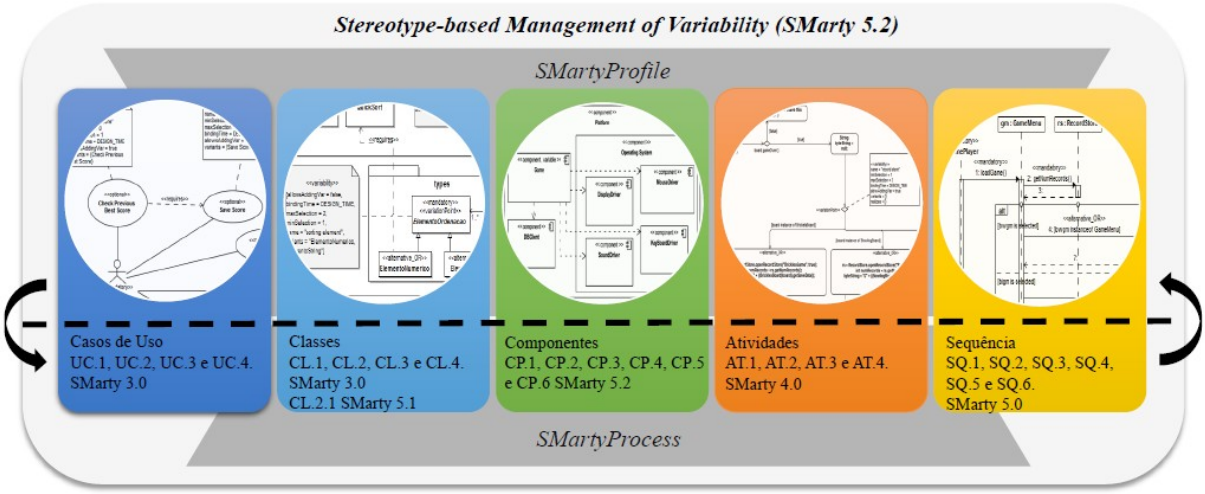
Os cenários da *SMartyPerspective* foram definidas sob duas bases principais: a taxonomia de defeitos e os papéis da Engenharia de Domínio da LPS. Porem, além das classes de defeitos, as diretrizes da abordagem *SMarty* e as especificações dos elementos dos diagramas suportados bela abordagem, colaboraram para definir as informações que deviam ser inspecionadas para atender os atributos de qualidade esperados para estes diagramas.

A abordagem *SMarty* atualmente está na versão 5.2 com suporte aos diagramas de caso de uso, classe, atividade, sequência e componente. Ela possui um perfil, o *SMartyProfile* para representar as variabilidades graficamente e um processo, *SMartyProcess*, que contém diretrizes para guiar o usuário a identificar as variabilidades.

Será apresentado neste apêndice os principais elementos para cada um dos diagramas *SMarty* inspecionados pela técnica *SMartyPerspective*: diagrama de caso de uso, classe, componente e sequência e as diretrizes do *SMartyProcess* da abordagem *SMarty* para gerenciamento de variabilidades nestes diagramas.

A Figura 2.1 apresenta uma visão da geral da *SMarty* com as versões que a abordagem foi estendida e as diretrizes do *SMartyProcess* para cada um dos diagramas suportados pela abordagem: caso de uso (versão 3.0 por OliveiraJr *et al.* (2010a)), classe (versão 3.0 por OliveiraJr *et al.* (2010a) e versão 5.1 por Marcolino (2014)), atividade (versão 4.0 por (Fiori *et al.*, 2012)) sequência (versão 5.0 por Marcolino (2014)) e componente (versão 5.2 por Bera *et al.* (2015)).

Figura 2.1: Visão Geral da SMarty 5.2



Fonte: (Bera *et al.*, 2015)

Para identificação e representação das variabilidades nos diagramas de caso de uso, classe, componente, atividade e sequência foram definidas seus diretrizes (OliveiraJr *et al.*, 2010a):

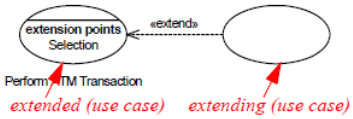
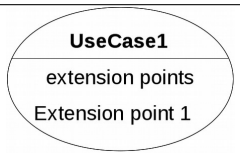
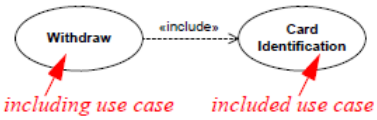
- RV.1 Variabilidades com variantes opcionais (`«optional»`) possuem multiplicidade `minSelection` = 0 e `maxSelection` = 1;
- RV.2 Variabilidades com variantes exclusivas (`«alternative_XOR»`) possuem multiplicidade `minSelection` = `maxSelection` = 1;
- RV.3 Variabilidades com variantes inclusivas (`«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.

B.2 Diagrama de Caso de Uso

Um caso de uso é uma uma ocorrência do comportamento do sistema. O diagrama de caso de uso captura os requisitos e a especificação dos requisitos das funcionalidades que se aplicam a um sistema e os atores que interagem com ele (OMG, 2017).

A Tabela 2.1 apresenta os principais elementos gráficos, suas notações, descrição e os caminhos gráficos (relações entre os casos de uso) que eles podem seguir para especificar os comportados do sistema por meio do diagrama de caso de uso.

Tabela 2.1: Elementos gráficos e caminhos para o diagrama de caso de uso

Elemento	Notação	Descrição
UseCase		Ele especifica um conjunto de comportamentos realizadas pelo assunto, que produz um resultado observável para os atores ou partes interessadas.
Actor		Um ator especifica um papel desempenhado por um usuário ou outro sistema que interage com o sistema especificado. É representado por um *"stick man"* com o nome do ator próximo.
Extend		É uma relacionamento de estende o caso de uso (*the extension*) para o caso de uso (*extendCase*), que especifica como e quando o comportamento definido em estende pode ser inserido no *extendeCase*
ExtensionPoint		Identifica um ponto no comportamento do caso de uso, que pode ser estendido (*extend*) por meio do comportamento de outro caso de uso (*extending*) especificado por meio do relacionamento *extend*.
Include		é um relacionamento que entre dois casos de uso que indica que o comportamento de caso de uso (*including*) é inserido no comportamento do caso de uso incluído (*included*)

Adaptado de (OMG, 2011, 2017)

Diretrizes SMarty para diagrama de caso de uso

As diretrizes da abordagem *SMarty* para o diagrama de caso de uso são apresentadas integralmente do trabalho de OliveiraJr *et al.* (2010a), como segue:

- 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»`);

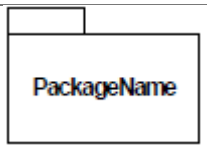
- UC.2 Modelos de casos de uso com o relacionamento de inclusão (`«include»`) ou associados a atores sugerem variantes obrigatórias («mandatory») ou opcionais (`«optional»`);
- UC.3 Variantes que, ao serem selecionadas para fazer parte de um produto, exigem a presença de outra(s) determinada(s) variante(s) 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»`.

B.3 Diagrama de Classes

Uma diagrama de classe mostra a estrutura estática do sistema, representando os elementos como classes com sua estrutura interna e relacionamentos, com o propósito de obter uma estrutura e do que é armazenado no sistema (da Silva Lima, 2018).

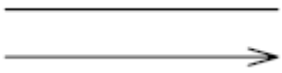
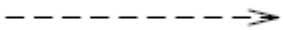
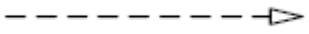
Classe e pacote são os principais elementos do diagrama de classe. A Tabela 2.2 apresenta a notação gráfica destes elementos e uma breve descrição. Já a Tabela 2.3 apresenta as mesmas informações para os caminhos gráficos (relações entre as classes) para o diagrama (OMG, 2011, 2017).

Tabela 2.2: Elementos gráficos para o diagrama de classe

Elemento	**Notação**	**Descrição**
classe		É um tipo de classificador que descreve um conjunto de objetos que compartilham a mesma especificação das características (atributos e operações), constantes e semântica.
pacote		É usado para agrupar os elementos e prover um *namespace* (conjunto de elementos nomeados que podem ser identificados pelo nome) para o grupo.

Adaptado de (OMG, 2011)

Tabela 2.3: Caminhos gráficos para o diagrama de classe

Tipo de Caminho	Notação	Descrição
Association		É um tipo de enumeração que especifica as literais definidos para o tipo de agregação de propriedade
Aggregation		descreve um conjunto de tuplas cujos valores referem-se a instâncias tipi cadas. Uma instância de uma associação é chamada de link, que é uma tupla com um valor para cada extremidade da associação.
Composition		É um tipo de agregação com um valor literal *composite*. Indica que a propriedade é agregada por composição, um objeto composto é responsável pela existência e armazenamento dos objetos compostos (partes).
Dependency		um único elemento ou um conjunto de elementos do diagrama requere outros elementos do diagrama para especificação ou implementação.
Generalization		uma relação taxonômica entre um classificador (classe ou elemento) mais geral e um classificador mais específico. Cada instância do classificador específico e também uma instância indireta do classificador mais geral. Assim, o classificador específico herda as características do classificador mais geral.
InterfaceRealization		É uma relação entre um classificador e uma interface.
Realization		é relacionamento de abstração especializado entre dois conjuntos de elementos do diagrama, um representando uma especificação (fornecedor) e o outro representa uma implementação do último (cliente).

Adaptado de (OMG, 2011)

B.3.1 Diretrizes SMarty para diagrama de classe

As diretrizes da *SMarty* ara o diagrama de classe foram definidas em duas versões. Na versão 3.0 de OliveiraJr *et al.* (2010a) e na versão 5.1 de Marcolino (2014). Na versão 5.1 foi adicionado o termo "sugerem"na diretriz CL.1 e na CL.2 foi acrescentado a diretriz suplementar CL.2.1 para representação de variabilidade em classes variantes opcionais (Marcolino, 2014).

Para o diagrama de classes, as diretrizes são (Marcolino, 2014; OliveiraJr *et al.*, 2010a):

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

    1. generalização, os classificadores mais gerais são os pontos de variação, enquanto os mais específicos são as variantes;
    2. realização de interface, os "suppliers"(especificações) são os pontos de variação e as implementações (clientes) são as variantes;
    3. agregação, as instâncias tipadas com losangos não preenchidos são os pontos de variação e as instâncias associadas são as variantes; e
    4. 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.2.1. Elementos de modelos de classes, relacionadas às associações nas quais os seus atributos `aggregationKind` possuem valor * (zero ou mais) ou 0..n onde n é um número inteiro qualquer, diferente de zero, sugerem que tal classe é opcional.

- 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»`;

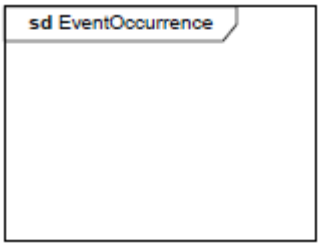
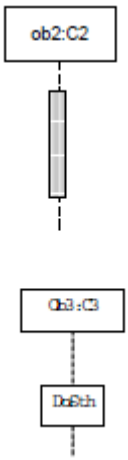
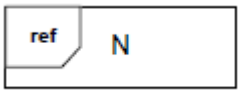
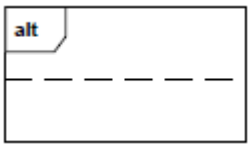
- CL.4 Variantes em modelos de classes, mutuamente exclusivas para um determinado produto devem ter seus relacionamentos de dependência marcados com o estereótipo `«mutex»`.

B.4 Diagrama de Sequência

Diagramas de interação são usados para compreender o comportamento interativo dos sistemas. Entre os diversos tipos, o diagrama de sequência é o mais comum. Ele modela o intercâmbio de mensagens entre uma série de linhas de vida (`lifelines`) com foco na sequência em que as mensagens são trocadas (OMG, 2017)

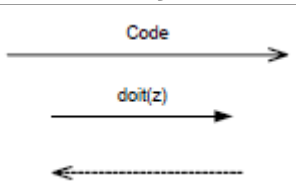
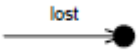
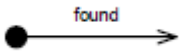
A Tabela 2.4 apresenta os principais elementos do diagrama de sequência com sua representação gráfica e sua descrição. Já na Tabela 2.5 são apresentados os tipos de mensagens que podem ser trocadas entre os elementos nas interações.

Tabela 2.4: Principais elementos gráficos do diagrama de sequência

Elemento	Notação	Descrição
Frame		A notação apresenta uma moldura retangular em volta do diagrama com o nome do compartimento no canto superior esquerdo.
Lifeline		É apresentado como um retângulo formando uma "cabeça"seguida de uma linha vertical que representa a linha de vida do objeto.
ExecutionSpecication		É a especificação da execução da ação que está sendo executada. É representada por um retângulo na linha vida identificando a ação.
InteractionUse		É a decomposição de uma linha de vida dentro de uma interação por uma interação. É apresentado como um retângulo com "ref"no canto superior esquerdo.
CombinedFragment		são fragmentos de interação. Sua semântica depende da especificação do operador, como: alt, opt, entre outros. É apresentado por um retângulo com o operador definido no canto superior esquerdo.
Destruction Occurrence Specification		representa a destruição de uma instância descrita pela linha de vida com este estereótipo. Podendo resultar na destruição de outros objetos que ele possui por composição.

Adaptado de OMG (2015)

Tabela 2.5: Caminhos gráficos para o diagrama de sequência

Tipo de Caminho	Notação	Descrição
Message		Sua representação depende do tipo da mensagem que ela está transmitindo.
LostMessage		Mensagens perdidas são mensagens para as quais o destino da mensagem é fora do âmbito da descrição.
FoundMessage		Mensagens encontradas são mensagens com receptor conhecido, mas o envio da mensagem não é descrita dentro da especificação
GeneralOrdering		linha pontilhada conectando as duas OccurrenceSpecifications

Adaptado de OMG (2015)

B.4.1 Diretrizes SMarty para diagramas de sequência

Foram incluídas na versão 5.0 da *SMarty* seis diretrizes no *SMartyProcess* para auxiliar no gerenciamento de variabilidades em diagramas UML de sequência por meio da identificação dos elementos e novos meta-modelos para estes diagramas de acordo com o padrão da UML 2.0 (Marcolino, 2014).

As diretrizes da *SMarty* para o diagrama de componente foram transcritas na integra do trabalho de Marcolino (2014):

- 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 onde 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 opcio-

nais, sendo estereotipados como «optional», e são relacionados a um comentário da UML especificando a variabilidade («variability»). Os lifelines contidos nesse 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 da UML, que identifica os elementos da variabilidade («variability»). Os diagramas de sequência referenciados pelo interactionUse "ref"correspondem 'as 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) que são independentes dos fluxos contidos no Cobined Fragment "alt", "opt", interactionUse "ref", ou não estejam relacionadas diretamente a uma variabilidade e seus elementos, são mantidas sem estereótipos e consideradas assim, obrigatórias;

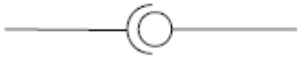
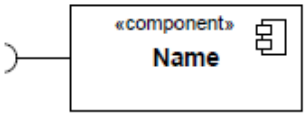
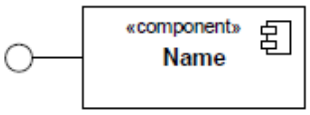
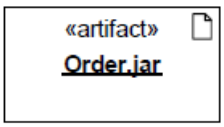
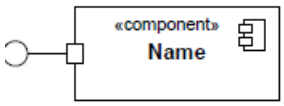
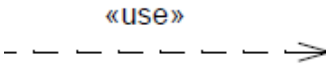
- SQ.5 Variantes em diagramas de sequência que, ao serem selecionadas para fazer parte de um produto específico, exigirem a presença de outra(s) determinada(s) variante(s) devem ter seus relacionamentos de dependência marcados com o estereótipo «requires»;

- 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».

B.5 Diagrama de Componentes

Um componente é módulo do sistema que encapsula seu conteúdo que pode ser substituído por outro componente que tenha interfaces fornecidas exigidas compatível. Ele encapsula o estado e o comportamento de vários classificadores e permite que o sistema seja estendido adicionando novas funcionalidades por meio de um componente (OMG, 2017).

Tabela 2.6: Caminhos gráficos para o diagrama de componente

Elemento	Notação	Descrição
Component		É uma unidade independente que encapsula o estado e o comportamento de vários classificadores. Ele especifica um contrato formal dos serviços que fornece aos seus clientes e aqueles que requer de outros componentes ou serviços no sistema em termos de suas interfaces fornecidas e necessárias.
Interface		É um tipo de classificador que representa uma declaração de um conjunto de características e obrigações públicas que juntas constituem um serviço coerente. Ela especifica um contrato.
Provide interface		Especificam o conjunto de operações e recepções que o classificador oferecem cliente
Required interface		Especificam o conjunto de operações e recepções que o classificador espera do cliente
Artifact		Artefatos representam elementos concretos no mundo físico que são o resultado do processo de desenvolvimento.
Port		É um ponto de interação que conduz interações entre as partes internas do classificador e seu ambiente ou entre o classificador e as partes internas.
ComponentRealization		É especializada para (opcionalmente) definir os classificadores que realizam o contrato oferecido por um Componente em termos de suas interfaces fornecidas e necessárias.
Interface Realization		É um relacionamento de realização especializada entre um classificador e uma interface, que significa que a realização do classificador está em conformidade com o contrato especializado pela Interface.
Usage Dependencies		É um relacionamento em que um elemento requer outro elemento (ou conjunto de elementos) para sua implementação completa ou operação.

Adaptado de (OMG, 2011)

O diagrama de componente permite representar os subsistemas de implementação com suas relações e arquivos de código-fonte. Ele descreve os componentes e os classificadores

que os especificam e os artefatos que devem implementar os componentes (da Silva Lima, 2018).

A Tabela 2.6 apresenta os principais elementos e relações entre eles de um diagrama de componente, sua notação e uma breve descrição entendimento de sua função.

B.5.1 Diretrizes SMarty para diagrama de componente

A versão 5.1 da abordagem *SMarty* já oferecia suporte para o gerenciamento de variabilidades em diagramas de componentes. Porém, só havia um estereótipo para colaborar com esta atividade, e ele não conseguia expressar de forma significativa a representação de variabilidades nestes diagramas (Bera *et al.*, 2015).

Para resolver o problema e acomodar as mudanças na versão 2.5 da UML de alguns elementos do diagrama de componente relacionados aos componentes e seus compartimentos para portas, interfaces e operações, a abordagem *SMarty* foi estendida para a versão 5.2. A nova versão permitiu a representação de variabilidades nestes diagramas e definiu seis diretrizes para o *SMartyProcess* Bera *et al.* (2015):

- CP.1 Componentes formados por classes com variabilidades são marcados com o estereótipo `variable`.
- CP.2 Componentes que identifiquem conjuntos de interfaces com mais de um tipo de variabilidades do tipo alternativa inclusiva e/ou alternativa exclusiva, sugere-se o uso de portas identificadas com o estereótipo `«variationPoint»`.
- CP.3 Sugere-se o uso de portas quando as interfaces relacionadas não sejam identificadas como opcionais, evitando a possibilidade de existir em uma possível seleção de variantes de modo que exista um componente com portas sem nenhuma interface relacionada. Sugere-se o uso de interfaces opcionais relacionadas diretamente com o próprio componente em si, ou em portas que possuem algum tipo de interfaces como sendo variante do tipo obrigatória e/ou alternativa inclusiva e/ou alternativa exclusiva.
- CP.4 Portas e operações que possuem alguma tipo de representação de variabilidade, o estereótipo referente a tal representação deve se tornar visível no compartimento do classificador.
- CP.5 Interfaces que possuem operações contendo algum tipo de identificação de variabilidades, deve tornar visível as operações em seu compartimento, apresentando-se no formato de classificador.

- CP.6 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 ou use.
- CP.7 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`.

B.6 Considerações Finais

Neste apêndice foram apresentados os principais conceitos, elementos e diretrizes dos diagramas *SMarty* que são inspecionados pela técnica *SMartyPerspective*. Essas informações foram relevantes para definir as questões da *SMartyPerspective* com base nas diretrizes e elementos necessários em cada diagrama.

Apêndice C: Relação entre questões e a taxonomia segundo SMartyPerspective

C.1 Considerações Iniciais

A *SMartyPerspective* foi definida sob duas bases principais: os papéis da Engenharia de Domínio e a taxonomia de defeitos da técnica. Uma taxonomia bem definida colabora com o inspetor para guiá-lo nos tipos de defeitos já conhecidos (Alshazly *et al.*, 2014) e para derivar as questões de uma técnica de Leitura Baseada em Perspectiva, como a *SMartyPerspective*, sob os atributos de qualidade esperados.

Nesta seção serão apresentadas as questões para cada um dos cenários da *SMartyPerspective* (sem instruções e introdução) relacionadas as diretrizes da abordagem *SMarty* que colaboraram para sua definição e a classe de defeito da taxonomia da *SMartyPerspective* da qual a questão foi derivada para cada um dos diagramas da técnica: para cada um dos diagramas: *features*, casos de uso, classe, componente, sequência para as perspectivas, sem considerar os números das questões e o passo que ela está inserida.

Para cada uma das tabelas são apresentadas as seguintes informações: as questões da *SMartyPerspective* para aquela perspectiva e diagrama, as diretrizes que serviram de base, caso houver, para a questão e a classe de defeito referente a questão: Ambiguidade (AM), Fato Incorreto (FI), Informação Estranha (IE), Inconsistência (IC) ou Omissão (OM).

C.2 Classes de Defeitos para Gerente de Produto

O cenário da *SMartyPerspective* para o Gerente de Produto (GPR) inspeciona os diagramas de *features* e de caso de uso sob a perspectiva do negócio da LPS para identificar as funcionalidades e características do sistema.

A Tabela 3.1 apresenta as classes de defeitos e as diretrizes da abordagem *SMarty* (para diagrama de caso de uso) relacionadas a cada uma das questões do cenário da *SMartyPerspective* para o diagrama de *features* e a Tabela 3.2 para o diagrama de caso de uso.

Tabela 3.1: Classes de defeitos para as questões do diagrama de *features* para GRP

Questões	**Diretriz**	**Classe**
O nó raiz representa corretamente o Domínio?		FI
O nome da *feature* expressa corretamente a característica que ela representa?		AM
A *feature* representa uma característica que não foi definida no documento de requisitos? Se sim, desconsidere-a com suas relações para os próximos passos e vá para o próximo elemento.		FI
A característica descrita por essa *feature* já foi especificada por outro elemento? Se sim, desconsidere-a com suas relações para os próximos passos e vá para o próximo elemento.		IE
Há realmente uma relação entre essa característica e o nó anterior?		IC
A característica obrigatória está definida com um círculo preenchido?		FI
A característica opcional está definida com um círculo vazado?		FI
Se houver descrita a cardinalidade, ela está correta de acordo com a especificação de requisitos?		FI
Esta *feature* realmente é uma característica alternativa da *feature* do nó anterior?		IC
Características alternativas estão dentro de arcos vazados ou com relação XOR?		FI
Características inclusivas estão dentro de arcos preenchidos ou com relação OR?		FI
Há realmente um relacionamento de inclusão/exclusão entre essas *features*?		IE
Se a *feature* requer outra, a linha está definida como uma seta direcionada?		OM
Se a *feature* exclui outra, a linha está definida como uma seta bidirecional?		OM
Faltou para esta *feature* alguma relação de inclusão/exclusão com outra?		OM
Há alguma característica opcional ou obrigatória para a LPS que não foi descrita no diagrama?		OM
Verifique os arcos vazados ou preenchidos. Faltou uma característica alternativa para a *feature*?		OM

Tabela 3.2: Classes de defeitos para as questões do diagrama de caso de uso para GRP

Questões	**Diretriz**	**Classe**
O nome do elemento expressa corretamente a funcionalidade?		FI
Este elemento corresponde a uma funcionalidade/ator que não foi definida no documento de requisitos? Se sim, desconsidere ele e suas relações para os próximos passos e vá para o próximo elemento.		IE
A funcionalidade descrita por esse elemento já foi especificada por outro elemento? Se sim, desconsidere ele e suas relações para os próximos passos e vá para o próximo elemento.		AM
O elemento no diagrama de caso de uso está estereotipado?		OM
No diagrama de caso de uso, se o elemento for opcional ou obrigatório, ele foi especificado com o estereótipo correto (`«optional»` ou `«mandatory»`)?		IC
caso de uso que são pontos de variação foram demarcados com o estereótipo «variationPoint»?		OM
O relacionamento está de acordo com a especificação de requisitos? Verifique se há realmente relação entre os elementos para o contexto.		IC
O relacionamento foi identificado como extensão (`«extend»`) ou inclusão («include») erroneamente de acordo com a especificação de requisitos?		FI
Os relacionamentos de inclusão entre os elementos foram especificados com o estereótipo «include»? Obs.: A *SMarty* sugere que os relacionamentos de inclusão estão associados a variantes obrigatórias (`«mandatory»`) ou opcionais (`«optional»`).		OM
Se o elemento requer outro, a relação entre eles foi estereotipada com `«requires»` ?	UC.3	OM
Se a relação for exclusão mútua, a relação está estereotipada no diagrama com `«mutex»` ?	UC.4	OM
Faltou algum relacionamento para este elemento que não foi especificado no diagrama de caso de uso?		OM
As variantes especificadas para esta variabilidade estão com a notação de variante correta (`«OR»`, `«XOR»` ou `«optional»`) ?		OM
As variantes do tipo «OR» e «XOR» relacionadas ao ponto de variação tem a relação `«extend»` para o ponto de variação associado?	UC.1	OM
Ainda há itens em sua lista que não foram especificados por nenhum elemento? Ou seja, está faltando no diagrama de caso de uso. (Se forem variantes que já foram identificadas no passo anterior, desconsidere)		OM

C.3 Classes de Defeitos para Engenheiro de Requisitos

O cenário da *SMartyPerspective* para o Engenheiro de Requisitos (ERD) inspeciona os diagramas de caso de uso, classe e sequência sob a perspectiva das funcionalidades, os objetos e as mensagens que são trocadas entre eles de todas das aplicações que podem ser configuradas a partir da LPS.

As classes de defeitos para cada uma das questões do cenário de ERD são apresentados nas tabelas que seguem para cada um dos diagramas característicos para executar as tarefas do papel: A Tabela 3.3 para o diagrama de caso de uso, a Tabela 3.4 para o de classe e a Tabela 3.5 para o de sequência.

Tabela 3.3: Classes de defeitos para as questões do diagrama de caso de uso para ERD

Questões	**Diretriz**	**Classe**
O nome do elemento expressa corretamente a funcionalidade?		FI
Este elemento corresponde a uma funcionalidade/ator que não foi definida no documento de requisitos? Se sim, desconsidere ele e suas relações para os próximos passos e vá para o próximo elemento.		IE
A funcionalidade descrita por esse elemento já foi especificada por outro elemento? Se sim, desconsidere o elemento que está incorreto e suas relações para os próximos passos e vá para o próximo elemento.		AM
O elemento no diagrama de caso de uso está estereotipado?		OM
No diagrama de caso de uso, se o elemento for opcional ou obrigatório, ele foi especificado com o estereótipo correto (`«optional»` ou `«mandatory»`)?		IC
O elemento que representa um ponto de variação foi demarcado com o estereótipo «variationPoint»?		OM
O relacionamento está de acordo com a especificação de requisitos? Verifique se há realmente relação entre os elementos para o contexto.		IC
O relacionamento foi identificado como extensão (`«extend»`) ou inclusão («include») erroneamente de acordo com a especificação de requisitos?		FI
Os relacionamentos de inclusão entre os elementos foram especificados com o estereótipo «include»?	UC.2	OM
Obs.: A *SMarty* sugere que os relacionamentos de inclusão estão associados a variantes obrigatórias (`«mandatory»`) ou opcionais (`«optional»`).		
Se o elemento requer outro, a relação entre eles foi estereotipada com `«requires»` ?	UC.3	OM
Se a relação for exclusão mútua, o relacionamento está estereotipado no diagrama de caso de uso com `«mutex»` ?	UC.4	OM
Faltou algum relacionamento para este elemento que não foi especificado no diagrama de caso de uso?		OM

Continua na próxima página

Tabela 3.3 – continuação da página anterior

Questões	Diretriz	Classe
Há uma notação de variabilidade (`«variabilitty»`) associada ao elemento?		OM
As variantes especificadas para esta variabilidade estão com a notação de variante correta (`«OR»`, `«XOR»` ou `«optional»`) ?		OM
As variantes do tipo `«OR»` e `«XOR»` relacionadas ao ponto de variação tem a relação `«extend»` para o ponto de variação associado?	UC.1	OM
As variantes definidas no conjunto *variants* realmente são variantes para este elemento? Se alguma não for, desconsidere-a para as próximas questões.		IC
Há variantes definidas na coleção de variantes que não está descrita no diagrama de caso de uso?		FI
Todas as variantes relacionadas a este elemento definidas com (`«OR»`), (`«XOR»`) ou (`«optional»`) estão na coleção de variantes do `meta-atributo` *variants* com seu nome correto?	RV.6	IC
Verifique o tipo das variantes associadas: • Se forem do tipo `«optional»` , `minSelection=0` e `maxSelection=1`? • Se forem do tipo `«OR»`, `minSelection=1` e `maxSelection=total` de variantes? • Se forem do tipo `«XOR»`, `minSelection=maxSelection=1`?	RV.1, RV.2 e RV.3	FI
Ainda há itens em sua lista que não foram especificados por nenhum elemento? Ou seja, está faltando no diagrama de caso de uso (Se forem variantes que já foram identificadas no passo anterior, desconsidere).		OM

Tabela 3.4: Classes de defeitos para as questões do diagrama de classe para ERD

Questões	Diretriz	Classe
O nome da classe expressa corretamente os objetos desta classe?		FI
Esta classe corresponde a um objeto que não foi definido no documento de requisitos? Se sim, desconsidere ele e suas relações para os próximos passos e vá para o próximo elemento.		IE
Esta classe está em redundância com outra já especificada no diagrama de classes? Se sim, desconsidere ele e suas relações para os próximos passos e vá para o próximo elemento.		AM
A classe está estereotipada no diagrama de classes?		OM
Se o elemento for opcional ou obrigatório, ele foi especificado com o estereótipo correto (`«optional»` ou `«mandatory»`)?		FI
O nome do pacote foi definido e expressa corretamente o agrupamento? Se já tiver colocado no formulário o defeito para este pacote não precisa colocar novamente.		FI
A classe está dentro do pacote correto?		FI
Este relacionamento está de acordo com a especificação de requisitos? Verifique se há realmente relação entre os elementos para o contexto.		IC

Continua na próxima página

Tabela 3.4 – continuação da página anterior

Questões	**Diretriz**	**Classe**
A cardinalidade deste relacionamento está correta de acordo com a especificação de requisitos?		IC
Verifique o tipo do relacionamento para garantir que as classes estejam com seus estereótipos definidos e corretos de acordo com a abordagem *SMarty*:	CL.1	OM
• Generalização: Os classificadores mais gerais, são pontos de variação (`«variationPoint»`) e os mais específicos, variantes? • Realização de interface: As especificações são pontos de variação (`«variationPoint»`) e as implementações são variantes? • Agregação ou Composição: As instâncias tipadas com losangos (preenchidos ou não) são pontos de variação (`«variationPoint»`) e instâncias associadas são variantes? As classes que exigem a presença de outra estão relacionadas com o estereótipo `«requires»` ?	CL.3	OM
As classes mutuamente exclusivas estão relacionadas com o estereótipo `«mutex»`?	CL.4	OM
Faltou algum relacionamento para esta classe que não foi especificado no diagrama de classes?		OM
Há uma nota da UML que representa variabilidade (`«variabilitty»`) associada à classe de controle?		OM
Todas as variantes foram definidas e estão com a notação de variante correta (`«OR»`, `«XOR»` ou `«optional»`) ?		OM
As variantes definidas no conjunto *variants* realmente são variantes para este elemento? Se alguma não for, desconsidere-a para as próximas questões.		IE
Há variantes definidas na coleção de variantes que não está descrita no diagrama de classes?		IC
Todas as variantes relacionadas a este elemento definidas com (`«OR»`), (`«XOR»`) ou (`«optional»`) estão na coleção de variantes do `meta-atributo` *variants* com seu nome correto?	RV.6	IC
Verifique o tipo das variantes associadas: • Se forem do tipo `«optional»` , `minSelection=0` e `maxSelection=1`? • Se forem do tipo `«OR»`, `minSelection=1` e `maxSelection=total` de variantes?. • Se forem do tipo `«XOR»`, `minSelection=maxSelection=1`?	RV.1, RV.2 e RV.3	IC
Ainda há itens em sua lista que não foram especificados por nenhum elemento? Ou seja, está faltando no diagrama de classe (Se forem variantes que já foram identificadas no passo anterior, desconsidere).		OM

Tabela 3.5: Classes de defeitos para as questões do diagrama de sequência ara ERD

Questões	**Diretriz**	**Classe**
O nome do elemento expressa corretamente o objeto/ator?		FI
O elemento está representado em alguma classe do sistema?		IC
O elemento faz parte desta interação de acordo com a Especificação de Requisitos? Se não, desconsidere toda a linha de vida e suas mensagens para os próximos passos e vá para o próximo elemento.		FI
O elemento é redundante com outro elemento definido? Se sim, desconsidere toda a linha de vida e suas mensagens para os próximos passos e vá para o próximo elemento.		AM
A mensagem está nomeada?		OM
A interação representada por esta mensagem está descrita na especificação de requisitos? Se não, desconsidere-a e passe para a próxima mensagem.		FI
O nome da mensagem expressa corretamente a informação que está sendo transmitida?		FI
A ordem da mensagem está correta de acordo com a especificação de requisitos?		FI
Verifique as outras mensagens nesta linha de vida. Esta mensagem está redundante com outra? Se sim desconsidere-a e passe para a próxima mensagem.		AM
Mensagens que não estejam relacionadas diretamente à uma variabilidade e seus elementos, não precisam de estereótipo e são consideradas obrigatórias. A mensagem deste tipo está estereotipada?	SQ. 4	IE
Todas as mensagens importantes para este elemento foram definidas no diagrama de sequências de acordo com a especificação de requisitos?		OM
Elementos que exigem a presença de outro estão relacionados com o estereótipo «requires» ?	SQ.5	OM
Elementos mutuamente exclusivos estão relacionados com o estereótipo «mutex»?	SQ.6	OM
O elemento está definido com o estereótipo «variationPoint»?	SQ.1 e SQ.3	OM
Há uma notação de UML que representa variabilidade («variabilitty») associada a este elemento/CombinedFragment?		OM
As variantes correspondentes às mensagens estão estereotipadas corretamente? Verifique os estereótipos das mensagens variantes para esta variabilidade. • Para as variantes do elemento interactionUse "ref": «OR» • Para as variantes com interactionOperator "alt": «XOR»		FI
Há uma notação de UML que representa variabilidade («variabilitty») associada a este elemento/CombinedFragment?	SQ.2	OM
As variantes correspondentes as mensagens/CombinedFragment estão estereotipadas corretamente com «optional» ?		OM
Para elementos demarcados com «optional» as linha de vidas que fazem parte do CombinedFragment também estão estereotipadas com «optional» ?		OM

Continua na próxima página

Tabela 3.5 – continuação da página anterior

Questões	Diretriz	Classe
As variantes definidas no conjunto *variants* realmente são variantes para este elemento? Se alguma não for, desconsidere-a para as próximas questões.		IC
Todas as variantes do conjunto de variantes tem uma linha de vida associada no diagrama de sequência?		FI
Todas as variantes relacionadas a este elemento definidas com (`«OR»`), (`«XOR»`) ou (`«optional»`) estão na coleção de variantes do `meta-atributo` *variants* com seu nome correto?	RV.6	IC
Verifique o tipo das variantes associadas: • Se forem do tipo `«optional»` , `minSelection=0` e `maxSelection=1`? • Se forem do tipo `«OR»`, `minSelection=1` e `maxSelection=total` de variantes?. • Se forem do tipo `«XOR»`, `minSelection=maxSelection=1`?	RV.1, RV.2 e RV.3	IC

C.4 Classes de Defeitos para Arquiteto de Domínio

O Arquiteto de Domínio (AQD) inspeciona os diagramas de classe, sob a perspectiva do projeto, e componente para configurar e gerenciar a evolução da arquitetura da LPS para todos os produtos configuráveis da família.

As classes de defeitos e as diretrizes foram relacionadas nas tabelas Tabela 3.6 Tabela 3.7 com as questões características do cenário de inspeção para AQD para os diagramas de classe e de componente, respectivamente.

Tabela 3.6: Classes de defeitos para as questões do diagrama de classe para AQD

Questões	Diretriz	Classe
O nome da classe expressa corretamente os objetos desta classe?		FI
Esta classe corresponde a um objeto que não foi definido no documento de requisitos? Se sim, desconsidere ela e seus relacionamentos para os próximos passos e vá para a próxima classe.		IE
Esta classe está em redundância com outra já especificada no diagrama de classes? Se sim, desconsidere ela e seus relacionamentos para os próximos passos e vá para a próxima classe.		AM
A classe está estereotipada no diagrama de classes?		OM
O nome do pacote foi definido e expressa corretamente o agrupamento? Se já tiver colocado no formulário o defeito para este pacote não precisa colocar novamente.		FI
A classe está dentro do pacote correto?		FI
Este relacionamento está de acordo com a especificação de requisitos? Verifique se há realmente relação entre os elementos para o contexto.		IC

Continua na próxima página

Tabela 3.6 – continuação da página anterior

Questões	Diretriz	Classe
A cardinalidade deste relacionamento está correta de acordo com a especificação de requisitos?		IC
Verifique o tipo do relacionamento para garantir que as classes estejam com seus estereótipos definidos e corretos de acordo com a abordagem *SMarty*: • Generalização: Os classificadores mais gerais, são pontos de variação (`«variationPoint»`) e os mais específicos, variantes? • Realização de interface: As especificações são pontos de variação (`«variationPoint»`) e as implementações são variantes? • Agregação ou Composição: As instâncias tipadas com losangos (preenchidos ou não) são pontos de variação (`«variationPoint»`) e instâncias associadas são variantes? • Associação: Verifique o atributo `AgregationKind` • se for none: as variantes sugerem ser obrigatórias (`«mandatory»`) ou opcionais (`«optional»`). Verifique de acordo com a especificação de requisitos. O estereótipo está correto? • se o valor * ou 0..n são do tipo opcionais (`«optional»`)?	CL.1	OM
As classes que exigem a presença de outra estão relacionadas com o estereótipo `«requires»` ?	CL.3	OM
As classes mutuamente exclusivas estão relacionadas com o estereótipo `«mutex»`?	CL.4	OM
A interface está estereotipada com «interface»?		OM
O nome do método expressa corretamente sua função?		FI
O método está redundante com outro já definido anteriormente para esta classe?		AM
Os principais métodos foram definidos?		OM
Há uma notação de UML que representa variabilidade (`«variabilitty»`) associada à classe de controle?		OM
Todas as variantes foram definidas e estão com a notação de variante correta («alternative OR», `«XOR»` ou `«optional»`) ?		OM
As variantes definidas no conjunto *variants* realmente são variantes para este elemento? Se alguma não for, desconsidere-a para as próximas questões.		IE
Há variantes definidas na coleção de variantes que não estão descritas no diagrama de classes?		IC
Todas as variantes relacionadas a este elemento definidas com (`«OR»`), (`«XOR»`) ou (`«optional»`) estão na coleção de variantes do `meta-atributo` *variants* com seu nome correto?	RV.6	IC
Verifique o tipo das variantes associadas: • Se forem do tipo `«optional»` , `minSelection=0` e `maxSelection=1`? • Se forem do tipo `«OR»`, `minSelection=1` e `maxSelection=total` de variantes? • Se forem do tipo `«XOR»`, `minSelection=maxSelection=1`?	RV.1, RV.2 e RV.3	IC
Ainda há itens em sua lista que não estão em nenhuma classe? Ou seja, está faltando no diagrama de classe (Se forem variantes que já foram identificadas no passo anterior, desconsidere).		OM

Tabela 3.7: Classes de defeitos para as questões do diagrama de componente para AQD

Questões	**Diretriz**	**Classe**
O nome expressa corretamente o elemento?		FI
O elemento está em redundância com outro já especificado anteriormente? Se sim, desconsidere-o para os próximos passos e vá para o próximo elemento		AM
O elemento está em sua lista? Se não, desconsidere-o para os próximos passos e vá para o próximo elemento.		IE
O elemento está estereotipado no diagrama de componentes?		OM
Se o componente for obrigatório, ele está demarcado com o estereótipo «mandatory» ?		FI
O relacionamento do componente/interface com outro elemento realmente existe?		FI
Se a relação for do tipo de contrato, a ordem fornecida/requerida ou requerida/-fornecida está correta?		FI
Variantes que exigem a presença de outra estão relacionadas com o estereótipo «requires» ou «use»?		OM
As variantes mutuamente exclusivas estão relacionadas com o estereótipo «mutex»?		OM
O estereótipo «optional» foi definido para o elemento?		OM
Há uma notação de UML que representa variabilidade («variabilitty») associada ao elemento?		OM
O estereótipo «variationPoint» foi definido para o elemento?		OM
Há uma notação de UML que representa variabilidade («variabilitty») associada ao elemento?		OM
Todas as variantes foram definidas e estão com a notação de variante correta («OR» ou «XOR») ?		OM
Há alguma variante descrita no diagrama de componente que não pertence a este elemento?		OM
Há alguma variante em redundância com outra já especificada?		AM
Há alguma variante relacionada ao elemento que foi definida com («OR») ou («XOR») que não foi especificada nos requisitos? Se sim, retire-o do diagrama e desconsidere esta variante e suas relações para as próximas questões.		FI
As variantes definidas no conjunto *variants* realmente são variantes para este elemento? Se alguma não for, desconsidere-a para as próximas questões.		IE
Há variantes definidas na coleção de variantes que não estão descritas no diagrama de componente?		IC
Todas as variantes relacionadas a este elemento definidas com («OR»), («XOR») ou («optional») estão na coleção de variantes do `meta-atributo` *variants* com seu nome correto?	RV.6	IC

Continua na próxima página

Tabela 3.7 – continuação da página anterior

Questões	Diretriz	Classe
Verifique o tipo das variantes associadas: • Se forem do tipo «optional» , minSelection=0 e maxSelection=1? • Se forem do tipo «OR», minSelection=1 e maxSelection=total de variantes? • Se forem do tipo «XOR», minSelection=maxSelection=1?	RV.1, RV.2 e RV.3	IC
O componente/interface está com o estereótipo de «variationPoint» definido?	CP.5	OM
Todos os componente/porta/interface variantes para resolver este ponto de variação estão visíveis no compartimento de operações?		IC
Há alguma porta/componente/interface especificada no compartimento de operações que não pertence a este elemento?		OM
Há alguma redundância das portas/componentes/interfaces especificadas no compartimento de operações?		FI
Os estereótipos no compartimento de operações estão corretos de acordo com cada um dos elementos ali descritos?	CP.6	AM
Ainda há itens em sua lista que não pertencem a nenhum componente? Ou seja, está faltando no diagrama de componente (Se forem variantes que já foram identificadas no passo anterior, desconsidere).		OM

C.5 Classes de Defeitos para Desenvolvedor de Domínio

O papel de Desenvolvedor de Domínio (DSD) é responsável por desenvolver os componentes que serão configurados e relacionados para um produto específico. Por isso, o ele deve inspecionar os diagramas de classe, componente e sequência para ter uma visão da arquitetura, dos métodos e da troca de mensagens entre os objetos.

A Tabela 3.8 apresenta a relação entre as questões e as classes de defeitos da Taxonomia da *SMartyPerspective* e as diretrizes da abordagem *SMarty* e a Tabela 3.9 para o diagrama de sequência. Para o diagrama de componente a tabela é a mesma da Tabela 3.7, pois essa parte do cenário é igual para as duas perspectivas.

Tabela 3.8: Classes de defeitos para as questões do diagrama de classe para DSD

Questões	**Diretriz**	**Classe**
O nome da classe expressa corretamente os objetos desta classe?		FI
Esta classe corresponde a um objeto que não foi definido no documento de requisitos? Se sim, desconsidere ela e seus relacionamentos para os próximos passos e vá para a próxima classe.		AM
Esta classe está em redundância com outra já especificada no diagrama de classe? Se sim, desconsidere ela e seus relacionamentos para os próximos passos e vá para a próxima classe.		IE
A classe está estereotipada no diagrama de classe?		OM
O nome do pacote foi definido e expressa corretamente o agrupamento? Se já tiver colocado no formulário o defeito para este pacote não precisa colocar novamente.		FI
A classe está dentro do pacote correto?		FI
Este relacionamento está de acordo com a especificação de requisitos? Verifique se há realmente relação entre os elementos para o contexto.		IC
A cardinalidade deste relacionamento está correta de acordo com a especificação de requisitos?		IC
Verifique o tipo do relacionamento para garantir que as classes estejam com seus estereótipos definidos e corretos de acordo com a abordagem *SMarty*: • Generalização: Os classificadores mais gerais, são pontos de variação (`«variationPoint»`) e os mais específicos, variantes? • Realização de interface: As especificações são pontos de variação (`«variationPoint»`) e as implementações são variantes? • Agregação ou Composição: As instâncias tipadas com losangos (preenchidos ou não) são pontos de variação (`«variationPoint»`) e instâncias associadas são variantes? • Associação: Verifique o atributo `AgregationKind` • se for none: as variantes sugerem ser obrigatórias (`«mandatory»`) ou opcionais (`«optional»`). Verifique de acordo com a especificação de requisitos. • se o valor * ou 0..n são do tipo opcionais (`«optional»`)?	CL.1	OM
As classes que exigem a presença de outra estão relacionadas com o estereótipo `«requires»` ?	CL.3	OM
As classes mutuamente exclusivas estão relacionadas com o estereótipo `«mutex»`?	CL.4	OM
Faltou algum relacionamento para esta classe que não foi especificado no diagrama de classes?		OM
O nome do atributo expressa corretamente o dado do mundo real?		FI
O atributo realmente pertence a esta classe?		FI
O atributo está redundante com outro já definido para esta classe?		AM
O tipo do atributo está correto?		FI
A visibilidade do atributo na classe está correta?		FI
Os principais atributos foram definidos para esta classe?		OM

Continua na próxima página

Tabela 3.8 – continuação da página anterior

Questões	**Diretriz**	**Classe**
A interface está estereotipada com «interface»?		OM
O nome do método expressa corretamente sua função?		FI
O método está redundante com outro já definido anteriormente para esta classe?		AM
Verifique os parâmetros de entrada para o método e responda as questões que seguem: • Faltou algum parâmetro de entrada para o método?		OM
• Algum parâmetro foi especificado além do necessário?		FI
• Os tipos dos parâmetros foram especificados corretamente?		OM
Verifique os parâmetros de saída para o método e responda as questões que seguem: • O valor do retorno está correto?		OM
• O tipo do retorno está correto?		FI
A visibilidade do método está correta?		FI
Faltou algum método importante para a implementação a ser definido?		OM
Há uma notação de UML que representa variabilidade (`«variabilitty»`) associada à classe de controle?		OM
Todas as variantes foram definidas e estão com a notação de variante correta (`«OR»`, `«XOR»` ou `«optional»`) ?		OM
As variantes definidas no conjunto *variants* realmente são variantes para este elemento? Se alguma não for, desconsidere-a para as próximas questões.		IE
Há variantes definidas na coleção de variantes que não está descrita no diagrama de classe?		IC
Todas as variantes relacionadas a este elemento definidas com (`«OR»`), (`«XOR»`) ou (`«optional»`) estão na coleção de variantes do `meta-atributo` *variants* com seu nome correto?	RV.6	IC
Verifique o tipo das variantes associadas: • Se forem do tipo `«optional»` , `minSelection=0` e `maxSelection=1`? • Se forem do tipo `«OR»`, `minSelection=1` e `maxSelection=total` de variantes?. • Se forem do tipo `«XOR»`, `minSelection=maxSelection=1`?	RV.1, RV.2 e RV.3	IC
Ainda há itens em sua lista que não estão em nenhuma classe? Ou seja, está faltando no diagrama de classe. (Se forem variantes que já foram identificadas no passo anterior, desconsidere)		OM

Tabela 3.9: Classes de defeitos para as questões do diagrama de sequência para DSD

Questões	**Diretriz**	**Classe**
O nome do elemento expressa corretamente o objeto/ator?		FI
O elemento está representado em alguma classe no diagrama de classe?		IC
O elemento faz parte desta interação de acordo com a Especificação de Requisitos? Se não, desconsidere toda a linha de vida e suas mensagens para os próximos passos e vá para o próximo elemento.		FI
O elemento é redundante com outro elemento definido? Se sim, desconsidere toda a linha de vida e suas mensagens para os próximos passos e vá para o próximo elemento.		AM
A mensagem está nomeada?		OM
A interação representada por esta mensagem está descrita na especificação de requisitos? Se não, desconsidere-a e passe para a próxima mensagem.		FI
A ordem da mensagem está correta de acordo com a especificação de requisitos?		FI
O nome da mensagem está de acordo com o nome do método no diagrama de classe?		FI
Se estiver especificado, os parâmetros de entrada estão corretos de acordo com o diagrama de classe?		IC
Se estiver especificado, os parâmetros de saída estão corretos de acordo com o diagrama de classe?		IC
Mensagens que não estejam relacionadas diretamente à uma variabilidade e seus elementos, não precisam de estereótipo e são consideradas obrigatórias. A mensagem deste tipo está estereotipada?	SQ. 4	IE
Todas as mensagens para este elemento foram definidas no diagrama de sequência de acordo com a especificação de requisitos e o diagrama de classe?		OM
Há mensagens redundantes nesta linha de vida? Se sim, desconsidere-a e passe para a próxima mensagem.		AM
Elementos que exigem a presença de outro estão relacionados com o estereótipo `«requires»` ?	SQ.5	OM
Elementos mutuamente exclusivos estão relacionados com o estereótipo `«mutex»`?	SQ.6	OM
O elemento está definido com o estereótipo `«variationPoint»`?	SQ.1 e SQ.3	OM
Há uma notação de UML que representa variabilidade (`«variabilitty»`) associada a este elemento/CombinedFragment?		OM
As variantes correspondente às mensagens estão estereotipadas corretamente? Verifique os estereótipos das mensagens variantes para esta variabilidade. • Para as variantes do elemento interactionUse "ref": `«OR»` • Para as variantes com interactionOperator "alt": «alternative xor»		FI
Há uma notação de UML que representa variabilidade (`«variabilitty»`) associada a este elemento/CombinedFragment?	SQ.2	OM

Continua na próxima página

Tabela 3.9 – continuação da página anterior

Questões	Diretriz	Classe
As variantes correspondentes às mensagens/CombinedFragment estão estereotipadas corretamente com «optional» ?		OM
Para elementos demarcados com «optional» , as linha de vidas que fazem parte do CombinedFragment também estão estereotipadas com «optional» ?		OM
As variantes definidas no conjunto *variants* realmente são variantes para este elemento? Se alguma não for, desconsidere-a para as próximas questões.		IC
Todas as variantes do conjunto de variantes tem uma linha de vida associada no diagrama de sequência?		FI
Todas as variantes relacionadas a este elemento definidas com («OR»), («XOR») ou («optional») estão na coleção de variantes do `meta-atributo` *variants* com seu nome correto?	RV.6	IC
Verifique o tipo das variantes associadas: • Se forem do tipo «OR», `minSelection=1` e `maxSelection=total` de variantes?. • Se forem do tipo «OR», `minSelection=1` e `maxSelection=total` de variantes?. • Se forem do tipo «XOR», `minSelection=maxSelection=1`?	RV.1, RV.2 e RV.3	IC

C.6 Classes de Defeitos para Gerente de Ativos de Domínio

A perspectiva de Gerente de Domínio (GAD) deve manter as versões de todos os artefatos e a rastreabilidade entre eles e entre as variabilidades correta na base de ativos, para que sejam reutilizados por outros papéis. Portanto, este papel inspeciona os cinco diagramas de gerenciamento de variabilidades suportados pela técnica *SMartyPerspective*: *features*, caso de uso, classe, componente e sequência.

Como o GAD deve verificar a consistência entre as diversas versões de um mesmo artefato, ele deve em seu cenário ele deve sempre fazer a comparação entre dois diagramas do mesmo tipo, ou entre diagramas diferentes buscando a rastreabilidade entre as variabilidades. Por isso, a maioria das questões são relacionadas a classe de defeito de Inconsistência (Tabela 3.10).

Tabela 3.10: Classes de defeitos para as questões do cenário de GAD

Questões	Diretriz	Classe
É possível rastrear o elemento para o oráculo? Se não, desconsidere-o com suas relações e vá para o próximo elemento (Lembre-se que elementos adicionados no registro de mudança não irão aparecer no oráculo)		IC

Continua na próxima página

Tabela 3.10 – continuação da página anterior

Questões	Diretriz	Classe
O nome do elemento na segunda versão difere/está abreviado da primeira versão?		IC
Na segunda versão, o estereótipo do elemento foi definido?		OM
O estereótipo do elemento na segunda versão está correto?		FC
Esta relação/mensagem existe no oráculo? Se não existir, desconsidere as próximas questões e vá para a próxima.		IC
Se houver, o estereótipo da relação/mensagem foi definido corretamente?		FC
Para o diagrama de sequência, verifique as mensagens:		IC
• A ordem da mensagem está correta de acordo com o oráculo?		
• Se houver definido, os parâmetros de entrada e saída estão corretos de acordo com o oráculo?		
Se houver, os atributos desta classe estão corretos de acordo com o oráculo? Verifique o nome e o tipo para responder a questão.		IC
Se houver, os métodos desta classe estão corretos de acordo com o oráculo? Verifique os parâmetros de entrada e saída e seus tipos para responder a questão.		IC
Todos os componente/porta/interface variantes estão visíveis no compartimento de operações assim como no oráculo? Verifique também os estereótipos		IC
A notação de variabilidade está presente na versão inspecionada?		OM
O nome (`meta-atributo name`) da variabilidade está redundante com outra já definida?		AM
O nome (`meta-atributo name`) da variabilidade a reflete corretamente ?		FC
`minSelection e maxSelection` estão definidos corretamente? Lembre-se de analisar se foi inserida alguma nova variante de acordo com o registro de mudanças.		IC
Verifique as variantes associadas. Estão presentes com seu nome correto como no oráculo?		IC
Verifique o `meta-atributo alllowsAddingVar` . Está correto de acordo com o oráculo?		IC
Verifique o `meta-atributo bindingTime`. Está correto de acordo com o oráculo?		IC
Verifique os meta-atributos `realizes+` e `realizes-`. Eles estão especificados na segunda versão, assim como na primeira?		OM
Algum elemento do oráculo não está presente na versão inspecionada?		OM
Algum relacionamento do oráculo não está presente na versão inspecionada?		OM
Você encontrou a variabilidade definida no diagrama inspecionado no diagrama correspondente?		IC
A variabilidade no diagrama correspondente realiza a variabilidade do diagrama inspecionado?		FC
Analise os diagramas correspondentes. Faltou alguma variabilidade em `realizes+` e `realizes-` do diagrama inspecionado que é rastreável de um diagrama para o outro?		OM

Apêndice D: Transcrição das Respostas Abertas da Análise Qualitativa

As respostas dos participantes para as questões abertas do Questionário de *Feedback* da avaliação da viabilidade da técnica *SMartyPerspective* foram transcritas em tabelas (Tabela 4.1) integralmente conforme preenchido pelo participante no questionário.

Tabela 4.1: Mapeamento das transcrições das respostas por questão aberta

Questão	**Tabela**
16. Considere os diagramas de gerenciamento de variabilidade (caso de uso, classe, sequência, componente, atividade e *features*) e a perspectiva que lhe foi atribuída para responder as questões que seguem:	-
16 a. Faltou algum diagrama a ser considerado para que o trabalho seja feito corretamente para sua perspectiva? Quais diagramas devem ser inseridos? Justifique.	Tabela 4.2
16 b. Algum diagrama definido para seu papel não é importante para que suas funções sejam exercidas? Qual diagrama deve ser removido deste cenário? Justifique.	Tabela 4.3
17. Considere as questões definidas para o cenário da perspectiva que lhe foi atribuída, para responder as questões que seguem:	-
17 a. Alguma questão não ficou clara o suficiente? Informe o item e o que a torna ambígua.	Tabela 4.4
17 b. Sob seu ponto de vista, alguma questão deve ser inserida no cenário? Qual item e porquê?	Tabela 4.5
17 c. Sob seu ponto de vista, alguma questão deve ser removida do cenário? Qual item e porquê?	Tabela 4.6
18. Você considera o processo de inspeção da técnica *SMartyPerspective* adequada com relação à identificação de defeitos em diagramas de gerenciamento de variabilidades? Justifique os pontos positivos.	Tabela 4.7
19. Você considera que o formato dos cenários contribuiu de maneira clara, objetiva e precisa para a inspeção dos diagramas de gerenciamento de variabilidades? Justifique.	Tabela 4.8
20. Você encontrou alguma dificuldade em identificar defeitos com o uso da técnica *SMartyPerspective* ? Justifique os pontos negativos.	Tabela 4.9
21. Você tem alguma sugestão para melhorar o cenário da *SMartyPerspective* ?	Tabela 4.10

Fonte: o autor

Tabela 4.2: Respostas dos especialistas para a questão 16 a.

Persp.	ID	Questão
GRP	GRP4	Não senti falta de nenhum diagrama.
	GRP2	Os diagramas inseridos são suficientes.
	GRP1	Sem apontamentos.
	GRP3	Não.
ERD	ERD4	Não faltou diagrama para que conseguisse fazer dentro da minha perspectiva.
	ERD3	Estavam completos. Não senti falta de nenhum
	ERD2	Entendo que os diagramas considerados são suficientes.
	ERD1	Não
AQD	AQD2	Não. Os diagramas presentes suprem as informações necessárias para a inspeção.
	AQD1	Não. A quantidade de diagramas é boa. Aumentar excessivamente a quantidade iria cansar o avaliador e aumentaria demais o tempo de avaliação.
	AQD3	Nenhum
	AQD4	um diagrama com menos classes
DSD	DSD2	Sim. Eu senti falta dos diagramas originais da LPS Arcade Game Maker (sem defeitos injetados artificialmente). Recomendo que não seja responsabilidade do participante criar uma lista de elementos porque isso gera inconsistência na inspeção e a especificação da LPS Arcade Game Maker já existe no Software Engineering Institute. Um diagrama correto ou já inspecionado deveria ser entregue ao participante para iniciar a inspeção do respectivo diagrama com defeitos injetados artificialmente. Acredito que a ideia do experimento está em avaliar o potencial da *SMartyPerspective* e não avaliar a modelagem dos participantes. Eu poderia ter colaborado ainda mais olhando para a técnica ao invés de me preocupar também com uma lista de elementos e uma modelagem inicial. Um diagrama com requisitos mais detalhados seria muito bom também.
	DSD3	Não.
	DSD1	Considerando que tenho experiência em inspeção, acredito que os diagramas contemplam um escopo completo da Linha de Produto de Software.
	DSD5	Não, todos os diagramas necessários para a análise na minha perspectiva estavam disponíveis. Entretanto, diagramas mais abstratos como de casos de uso e features, na prática, podem ser úteis para contextualização do desenvolvedor.
	DSD4	Não acredito que tenha faltado algum
GAD	GAD1	Nao.
	GAD4	Não que eu me lembre.
	GAD3	Não minha opinião não, os diagramas apresentados estão aceitáveis
	GAD2	Não

Fonte: o autor

Tabela 4.3: Respostas dos especialistas para a questão 16b

Persp.	ID	Questão
GRP	GRP4	Acho que os diagramas inspecionados estão apropriados.
	GRP2	Nenhum. Os dois avaliados estão de acordo com o papel seguido.
	GRP1	Sem apontamentos.
	GRP3	Não.
ERD	ERD4	Acredito que todos os diagramas são importantes e um complementa o outro.
	ERD3	Todos os diagramas foram bem empregados para visualizar o cenário apresentado
	ERD2	Entendo que todos os diagramas definidos são importantes.
	ERD1	Todos os diagramas utilizados por mim são importantes para minha função
AQD	AQD2	Não. Todos são de grande valia.
	AQD1	Todos os diagramas são úteis e nenhum deve ser removida. estes diagramas são importantes a serem avaliados
	AQD3	Nenhum
	AQD4	estao ok
DSD	DSD2	Não é necessário o Diagrama de Features e o Diagrama de Caso de Uso. Não consultei tais diagramas durante a inspeção. Na minha opinião, não são relevantes para o meu papel. Além disso, acho mais importante a disponibilização dos respectivos diagramas originais da LPS Arcade Game Maker.
	DSD3	Não.
	DSD1	Talvez do ponto de vista de analista, os diagramas de classes e sequência fossem essenciais, já o diagrama de componentes nem tanto.
	DSD5	Acredito que todos sejam importantes, cada um tem uma visão e nível de abstração distintos, o que os torna totalmente complementares.
	DSD4	Nenhum deve ser removido
GAD	GAD1	Nao.
	GAD4	Acredito que todos são importantes.
	GAD3	na minha visão o de Sequencia, por não apresenta um detalhamento tão preciso quanto aos demais para a rastreabilidade.
	GAD2	Todos tem um papel fundamental, porém se tivesse que retirar um, seria o diagrama de sequência UML. Acho importante a documentação de software, mas o diagrama de sequência é tão específico que o tempo exigido para modelá-lo não vale o ganho futuro.

Fonte: o autor

Tabela 4.4: Respostas dos especialistas para a questão 17a

Persp.	ID	Resposta
GRP	GRP4	Não é o caso.
	GRP2	Ficou claro o suficiente para mim.
	GRP1	Sem apontamentos.
	GRP3	Não.
ERD	ERD4	Todas ficaram claras, algumas que poderiam causar dúvida havia uma observação já logo abaixo, facilitando o entendimento.
	ERD3	Foi bem definido e estava claro o que deveria ser analisado
	ERD2	No meu entendimento, as questões foram elaboradas de maneira clara e precisa.
	ERD1	Todas as questão são claras
AQD	AQD2	Sim. o item 3.8.1 está duplicado. Os ids dos items 1.8 estão com id errados.
	AQD1	Tem alguns problemas, como id das questões duplicadas (3.8.1) ou com valores diferentes do id geral (1.8), está como 1.10.x.
	AQD3	Nenhuma
	AQD4	as questoes estao claras e bem especificas para o uso e aplicação no smarty
	DSD2	A questão 1.6.2 achei ruim para responder... o que você quer dizer com cardinalidade? Não ficou claro para mim. 1.6.2 A cardinalidade deste relacionamento está correta de acordo com a especificação de requisitos? Na questão 1.7 como eu consigo garantir isso? Apenas a especificação de requisitos é suficiente? Acredito que o diagrama original da LPS pode ajudar a avaliar a *SMartyPerspective* 1.7 Faltou algum relacionamento para esta classe que não foi especificado no diagrama de classes? Na questão 1.9 como garantir que realmente os principais atributos foram modelados? Não é fácil garantir isso. Certo? 1.9 Os principais atributos foram definidos para esta classe? O estereótipo <<use>>não foi explicado no treinamento. Sugiro revisar a questão 3.6.3. Variantes que exigem a presença de outra estão relacionadas com o estereótipo <<requires>>ou <<use>>? Favor revisar a questão 4.2.2 e criar questões diferentes para os componentes, portas e interfaces. 4.2.2 Todos os componente/porta/interface variantes para resolver este ponto de variação estão visíveis no compartimento de operações? A questão 5.5.7 está muito confusa. Favor revisar talvez para: Mensagens obrigatórias estão estereotipadas? 5.5.7 Mensagens que não estejam relacionadas diretamente à uma variabilidade e seus elementos, não precisam de estereótipo e são consideradas obrigatórias. A mensagem deste tipo está estereotipada? As instruções para os Passos 6 e 7 são muito confusas. Favor tentar deixá-las mais objetivas e sucintas.

Continua na próxima página

Tabela 4.4 – continuação da página anterior

Persp.	ID	Resposta
DSD	DSD3	Todas foram claras.
	DSD1	Algumas questões me deixaram confuso no decorrer da atividade. Eu fiquei perdido por algumas questões mencionarem • Para as variantes do elemento interactionUse "ref": <<alternative_OR>>• Para as variiantes com interactionOperator "alt": <<alternative xor>>.
	DSD5	As questões estavam relativamente claras, mas tive dificuldades talvez relacionadas ao desconhecimento das capacidades da SMarty alguns dos diagramas. Com isso, em algumas situações, usei minha experiência na área de desenvolvimento de software (cerca de 12 anos) para algumas sugestões.
	DSD4	Os passos 1, 2 e 5, em um primeiro momento não ficam claros se devem ser executados. No vídeo eram indicados apenas os passos 3 e 4 a serem seguidos...
GAD	GAD1	Nao.
	GAD4	1.7.1 - Talvez seja melhor quebrar em diferentes questões para deixar mais granular.
	GAD3	tudo de acordo
	GAD2	Não tive problemas.

Fonte: o autor

Tabela 4.5: Respostas dos especialistas para a questão 17 b.

Persp.	ID	Questão
GRP	GRP4	Na inspeção do diagrama de caso de uso, senti falta de um passo para inspecionar se as variabilidades foram especificadas apropriadamente. No caso da variabilidade relacionada a Play Selected Game, falta incluir a variante Play Bowling.
	GRP2	Está bem completo.
	GRP1	Sem apontamentos.
	GRP3	Restrições de dependência, que são elementos fundamentais para modelos de features, não estão tão claramente representados neste checklist.
ERD	ERD4	No momento não vejo questões para acrescentar. As que tem hoje já englobam tudo que é necessário.
	ERD3	Não precisa incluir nenhum item
	ERD2	Considerando minha primeira avaliação, não percebi a necessidade de inserir outras questões.
	ERD1	Considero as questões suficientes e completas
AQD	AQD2	Não.
	AQD1	Não.
	AQD3	Nenhuma
	AQD4	no
DSD	DSD2	Por enquanto não. Entretanto, recomendo que todas as questões sejam revisadas com cuidado. Há vários itens que tive dificuldade em entender em uma primeira leitura. Demorei um tempo para compreender as questões em si. Recomendo revisar todas as questões referente a inspeção do diagrama de sequência, são muito confusas. Durante a inspeção tive que repetir algumas classes para explicar os defeitos em questões diferentes. Todos esses detalhes que mencionei tornaram o experimento longo e um pouco cansativo, tornando a inspeção demorada.
	DSD3	Acredito que o questionário está bem completo.
	DSD1	Eu não possuo experiência no assunto, portanto não tenho uma noção mais ampla pra recomendar outras questões.
	DSD5	Não, acredito que a quantidade de questões é até maior do que o ideal.
	DSD4	Não vejo uma questão a ser inserida no cenário...
GAD	GAD1	Nao.
	GAD4	Acredito que não.
	GAD3	Não tudo de acordo
	GAD2	Não.

Fonte: o autor

Tabela 4.6: Respostas dos especialistas para a questão 17c

Persp.	ID	Questão
GRP	GRP4	Não se aplica.
	GRP2	Eu li e apliquei todas.
	GRP1	Sem apontamentos.
	GRP3	1.7.1 e 1.8; são muito parecidas. Idem para 2.2 e 1.5.2 e 2.2. 1.5.3; bastaria incluir a 2.2.
ERD	ERD4	Não, acredito que todas as questões são importantes.
	ERD3	Não precisa excluir nenhum item
	ERD2	Considerando minha primeira avaliação, não percebi a necessidade de remover alguma questão.
	ERD1	Não
AQD	AQD2	Não.
	AQD1	Não.
	AQD3	Nenhuma
	AQD4	no
DSD	DSD2	Sugiro a remoção das seguintes questões: Como garantir a rastreabilidade na questão 1.6.1? Achei que está em alto nível: 1.6.1 Este relacionamento está de acordo com a especificação de requisitos? Na questão 1.8.1 fica difícil identificar o significado de dado do mundo real, mesmo sendo no contexto de classes de entidade. A rastreabilidade pode ser difícil aqui. 1.8.1 O nome do atributo expressa corretamente o dado do mundo real? Sugiro remover a item 2.3 para não influenciar uma modelagem de atributos ou métodos incorretos para o cenário e os requisitos fornecidos. 2.3. Ainda há itens em sua lista que não estão em nenhuma classe? Ou seja, está faltando no diagrama de classe. (Se forem variantes que já foram identificadas no passo anterior, desconsidere) Como identificar tal omissão na questão 3.8.3? Acredito que a engenharia de requisitos não é suficiente. 3.8.3. Há alguma variante descrita no diagrama de componente que não pertence a este elemento?
	DSD3	Não removeria nenhuma questão.
	DSD1	As questões citadas na resposta 17.a me levaram a ficar confuso algumas vezes, no entanto, consegui compreender as questões para realizar a atividade.
	DSD5	De modo geral, achei algumas questões repetitivas e os formulários de inspeção extremamente longos e trabalhosos. Por isso, imaginando uma aplicação prática, acredito que o formulário tenha de ser simplificado para contextos mais ágeis de desenvolvimento. Nesse contexto, mesmo em uma estratégia sistemática de reuso como às LPS, o processo de desenvolvimento pode ser iterativo (em sprints, por exemplo). Sendo assim, pensar a esse respeito pode ser algo relevante em termos de trabalhos futuros.
	DSD4	Não vejo uma questão a ser removida do cenário...
GAD	GAD1	Nao.
	GAD4	Nenhuma.
	GAD3	todas as informações são pertinentes
	GAD2	Não.

Fonte: o autor

Tabela 4.7: Respostas dos especialistas para a questão 18

Persp.	ID	Resposta
GRP	GRP4	A técnica organiza as ideias para a inspeção e sistematiza a busca por defeitos.
	GRP2	As principais vantagens incluem: - o processo guia a inspeção; - o processo (questões) orienta "o que"procurar; - amplia os tipos de defeitos achados; - o papel ajuda a limitar o que olhar (e como).
	GRP1	Sim, o processo é uma abordagem analítica com usabilidade interessante há área. Além disso, as etapas para a avaliação contribuem para a construção do conhecimento.
	GRP3	Interessante como um guia.
ERD	ERD4	Sim, pois possuem questões específicas que ajudam a identificar os defeitos nos diagramas de gerenciamento de variabilidades.
	ERD3	Considero importante, pois faz com que o engenheiro avalie todos os tópicos apresentados
	ERD2	Considero o processo de inspeção adequado, pois tal atividade poderia ser realizada em etapas, de maneira precisa. Por exemplo, no diagrama de classes, considera-se primeiro as classes. Depois os relacionamentos e na sequência, os elementos relacionados com as variabilidades.
	ERD1	Sim, a técnica *SMartyPerspective* é iinteressante e fornece subsídios para que defeitos sejam detectados com maus facilidade
AQD	AQD2	Sim. O passo a passo das instruções de inspeção são bem claros e diretos, sendo fator facilitador para atacar os problemas possíveis. Além disso, como estou familiarizado com os termos envolvidos em LPS, a tarefa se torna mais fácil.
	AQD1	O processo é bem detalhado e sistemático. É fácil utilizar e seguir o processo para encontrar os problemas existentes no projeto.
	AQD3	Sim. Eu tenho mais controle do que estou fazendo e a profundidade da análise se desenvolve de forma gradual.
	AQD4	Considero parcialmente adequada, pois é um processo restrito à abordagem smarty. Vejo q tera que adaptar para outras abordagens de FM.
	DSD2	Sim. As perspectivas ajudam a capturar interpretações diferentes com papéis e responsabilidades distintos(as). O processo funciona bem por causa do conjunto de artefatos disponibilizados para cada tipo de papel. Isso faz com que a inspeção seja detalhada e assertiva. O papel fica informado sobre "o que"deve encontrar e ajuda a melhorar no raciocínio durante a modelagem de variabilidades e, consequentemente, o gerenciamento de variabilidades. As questões são excelentes para guiar o processo de inspeção, favorecendo melhor qualidade dos diagramas.
	DSD3	É uma ótima técnica de apoio para a atividade de inspeção, pois se apresenta de forma completa e objetiva.
	DSD1	Sim. No meu caso, apesar de não possuir experiência em técnicas de inspeção, as questões me guiaram com clareza no decorrer da atividade.

Continua na próxima página

Tabela 4.7 – continuação da página anterior

Persp.	ID	Resposta
DSD	DSD5	Sim, o processo é adequado e extremamente sistemático no que tange as características de diagramas UML modelados com o apoio da SMarty.
	DSD4	Com essa inspeção através de um checklist, a verificação torna-se mais fácil, no sentido de se ater a detalhes que são importantes na detecção de defeitos...
GAD	GAD1	Sim. Aponta para onde prestar atencao quando analisando os diagramas.
	GAD4	Sim, a técnica cria uma template na cabeça que ajuda a realizar um passa-a-passo bem definido de como os defeitos podem ser identificados.
	GAD3	Considero adequado, entretendo fazer a inspeção em diversos diagramas diferentes sequencialmente pode atrapalhar um pouco o entendimento
	GAD2	Sim. Quando se trata de inspeções de defeitos em diagramas, muitos detalhes são verificados e, por essa questão, é difícil conseguir inspecionar corretamente todas as minúcias sem cometer erros. Consequentemente, todas as técnicas que auxiliem este papel, é bem-vinda.

Fonte: o autor

Tabela 4.8: Respostas dos especialistas para a questão 19

Persp.	ID	Questão
GRP	GRP4	Sim.
	GRP2	Com certeza os cenários auxiliam a atividade de inspeção.
	GRP1	Sim, o formato com as propriedades subdivididas foram interessantes para avaliar e interpretar cada ponto da abordagem.
	GRP3	Não considero. O processo é muito laboroso. Forçar um engenheiro a seguir todos esses passos é algo que na prática pode ser tornar uma atividade de difícil acompanhamento.
ERD	ERD4	Sim, a forma como foi organizado facilita muito na inspeção dos diagramas.
	ERD3	Sim. A distribuição dos casos de uso e das classes ficaram claras para o gerenciamento.
	ERD2	Sim, porque cada cenário apresenta um propósito específico. No caso do cenário estabelecido para a minha avaliação, entendo que os diagramas de gerenciamento de variabilidades são adequados.
	ERD1	Sim, os cenários permitem melhor entendimento para realizar a inspeção
AQD	AQD2	Sim. O formato está legível, bem organizado e sucinto. Somente alguns erros de numeração em alguns items especificados.
	AQD1	Sim. o cenário auxilia na inspeção. Apesar de ter erros de numeração, esta técnica é clara e objetiva. Teve muitas questões, o que pode ajudar a encontrar diferentes problemas, mas a quantidade alta também pode gerar cansaço e erros ao examinar projetos grandes.
	AQD3	Sim, pois há coerência no formato.
	AQD4	Acho que se o foco eh avaliar a abordagem, poderia ser um cenario mais simples com menos classe focando mais na proposta de inspeção.
DSD	DSD2	Os cenários ajudam a posicionar o trabalho do inspetor ou do papel. Acredito que é sempre bom ser guiado de maneira objetiva e precisa. Isso é proporcionado pela *SMartyPerspective*. No entanto, as descrições dos cenários precisam ser revisadas para deixá-las mais claras para extrair o melhor da *SMartyPerspective*.
	DSD3	Sim, pois permitiu que a inspeção fosse realmente mais focada em uma perspectiva, o que facilita o trabalho, considerando a experiência e vivência do profissional de inspeção.
	DSD1	Sim. A partir do direcionamento de cenários de acordo com o papel, tornou mais "justa"a avaliação e permitiu uma inspeção mais segura de todas as questões.
	DSD5	Sim, considero. Entretanto, acredito que o processo de inspeção tenha que possuir uma variação mais objetiva para contextos mais dinâmicos de desenvolvimento.
	DSD4	Acredito que sim. Ajuda a focar no que deve ser inspecionado..
GAD	GAD1	Sim. É possível entender as variabilidades e semelhancas.
	GAD4	Sim, foi fácil analisar os diagramas utilizados no estudo. Acredito que a dificuldade maior foi saber aplicar a técnica no diagrama de sequência.
	GAD3	sim
	GAD2	Sim. Os documentos fornecidos, bem como os vídeos de treinamentos auxiliaram bastante o entendimento, além do exemplo abordado ser de fácil compreensão.

Fonte: o autor

Tabela 4.9: Respostas dos especialistas para a questão 20

Persp.	ID	Resposta
GRP	GRP4	Não propriamente com a técnica, mas achei que o documento de requisitos da AGM não ajuda a identificar bem o que é opcional e o que é obrigatório. Respondi com o meu conhecimento da LPS, por exemplo, sei que save costuma ser considerada uma feature opcional, mas no documento de requisitos isto não está claro.
	GRP2	Não detectei dificuldades. Como ponto negativo poderia mencionar o tempo "extra"gasto para "entender"o experimento. O tempo para aplicar o experimento e entendê-lo foram equivalentes.
	GRP1	Sim, pois não tinha conhecimento prévio sobre o modelo base utilizado. Contudo, elas foram supridas com as explicações. As dificuldades se relacionam com a falta de conhecimento do requisitos do modelo original em contrapartida com o itens a serem avaliados.
	GRP3	Eu não sei se de fato encontrei os defeitos prescritos. Falta um oráculo de comparação de resultados. Em linhas gerais, eu sempre acho interessante seguir um guia. Entretanto, eu não utilizaria este guia em particular, por acha-lo contraproducente. Explico: há diversos elementos no checklist que são muito triviais (ex: verificar se um arco está vazado ou preenchido etc., visto que qualquer boa ferramenta de gerenciamento de variabilidade já resolve isso. Eu entendo que na prática ninguém desenha diagrama de features manualmente, mas sim conta com as ferramentas; assim sendo, se diversos elementos que são triviais fossem removidos desse checklist, a atividade seria mais facil de entregar.
ERD	ERD4	O documento de requisitos não tinha muitas informações que ajudavam a identificar os defeitos no diagrama de classe, principalmente, em relação aos relacionamentos, enquanto que no de caso de uso e no de sequência não tive essa dificuldade.
	ERD3	Não tenho muita familiaridade com o diagrama de sequência e dificultou encontrar os erros.
	ERD2	Fiquei em dúvida sobre avaliar interfaces. Tal dúvida surgiu pelas orientações apresentadas na inspeção do diagrama de classes.
	ERD1	Não, as questões colaboram de maneira muito significativa na identificação de defeitos
	AQD2	Em geral não. A técnica me auxiliou de maneira que permitisse encontrar problemas bem específicos, e muitas vezes que passam despercebidos, permitindo identificar estes erros com uma visão mais rica em acurácia. Porém seria interessante compartilhar o documento de requisitos para inspecionar alguns itens especificados.

Continua na próxima página

Tabela 4.9 – continuação da página anterior

Persp.	ID	Resposta
AQD	AQD1	Sim. Como não foi fornecido o documento de requisitos, não foi possível avaliar diferentes questões relativas a verificação de nomes e relacionamentos (podendo apenas avaliar procurando padrões). O diagrama de componentes foi mais difícil de avaliar que o diagrama de classes, por apresentar diversos nomes com erros, estrutura aparentemente diferente e diversos métodos não vistos anteriormente.
	AQD3	Não encontrei dificuldade
	AQD4	Uma inspeção mais simples e mais geral poderia ajudar mais do que algo muito especifico e orientado a smarty.
DSD	DSD2	De forma geral não tive muitas dificuldades. Entretanto, recomendo a revisão dos itens das questões. Como mencionado, tive que ler várias vezes as questões para compreender o que era solicitado na inspeção. Acredito que *SMartyPerspective* evoluirá com revisões das questões ao longo do tempo. Por fim, tenho dúvidas quanto a escalabilidade da *SMartyPerspective* durante a inspeção de diagramas com 100 ou 1000 classes. Isso pode inviabilizar a proposta ao longo do tempo e os inspetores escolherem um conjunto de questões para utilizar durante a inspeção.
	DSD3	Não tive problemas com a técnica em si, mas com o domínio do projeto que foi usado para a inspeção. Algumas informações estavam um pouco confusas. Talvez um projeto do domínio comercial facilitaria a participação no estudo.
	DSD1	Não. Acredito que o *SMartyPerspective* apresenta um resumo das principais questões referentes à inspeção, e no meu caso, avalio que facilitou consideravelmente no decorrer da inspeção.
	DSD5	Sim, tive muitas dificuldades em entender algumas notações e relacionamentos nos diagramas. Além disso, o nível de conhecimento intermediário na SMarty pode ter comprometido algumas das minhas reflexões e análises.
	DSD4	Sim. Contudo, creio que seja mais em relação mais a não ter domínio de diagramação UML do que da técnica em si.
GAD	GAD1	Sim. Achei uma tarefa massante de fazer manualmente.
	GAD4	Sim, foi um pouco difícil analisar a rastreabilidade entre variabilidades. Poderia ser apresentado mais exemplos durante a fase aprendizado, principalmente envolvendo diagramas de sequência.
	GAD3	Não na técnica em si, mas na quantidade de diagramas sequenciais analisados.
	GAD2	Algumas detalhes ficaram confusos, como por exemplo, como inspeciono em outros diagramas além do diagrama de classe e caso de uso? Talvez mais alguns vídeos abordando os outros diagramas facilitariam o entendimento.

Fonte: o autor

Tabela 4.10: Respostas dos especialistas para a questão 21

Persp.	ID	Questão
GRP	GRP4	Senti falta de um passo para inspecionar se as variabilidades foram especificadas apropriadamente, me refiro ao preenchimento dos itens da nota que especifica as variabilidades SMarty.
	GRP2	Gostaria de parabenizá-los pelo trabalho :)
	GRP1	A abordagem é interessante e eficaz. Contudo acredito que há uma carga muito grande de informações nos vídeos para que posteriormente a analise possa ser feita. Acredito que possa ir direto ao ponto, como os exemplos, sem muita contextualização.
	GRP3	Remover todos os elementos triviais que já estão disponíveis em ferramentas de gerenciamento de variabilidade.
ERD	ERD4	No momento não, acredito que já está bem completa para atender os diagramas de gerenciamento de variabilidades.
	ERD3	Não
	ERD2	Na verdade, eu tenho sugestões mais gerais para a técnica. Não observei orientações a nível de cenário nesta primeira avaliação. Se desejar, entre em contato, que eu explico as sugestões. Acredito que tais sugestões não caberão neste campo de texto.
	ERD1	O único ponto que levanto com sugestão é a descrição das classes e elementos dos demais diagramas, assim como foi detalhado para o diagrama de casos de uso, pois dessa maneira, em conjunto com as questões fornecidas, a identificação dos defeitos poderia ser ainda mais efetiva.
AQD	AQD2	Sim. Fornecer uma documentação mais rica (documento de requisitos)
	AQD1	Fornecer o documento de requisitos. A falta deste documento faz com que diversas questões não possam ser avaliadas e dificulta a comparação entre os diagramas.
	AQD3	Uma ferramenta para automatização seria interessante.
	AQD4	Fazer algo mais simples que identifique erros de composição ou erros de relacionamentos das classes ajudaria mais o expert da área. Tentar identificar todos os erros por um questionário torna o trabalho adhoc e cansativo. Ja que vai ser manual, que seja por etapas menores.
DSD	DSD2	Eu mencionei algumas sugestões e recomendações nas outras questões. De forma geral, sugiro revisar os itens das questões e tentar diminuir a quantidade de itens para tornar a inspeção mais produtiva.
	DSD3	No momento, não. Parabéns pelo trabalho!
	DSD1	Pela minha falta de experiência, não possuo uma visão ampla para sugestão de melhorias. No entanto, do ponto de vista do experimento, considero que um link mais claro entre os diagramas de classes, componentes e sequência deixaria mais fácil a execução, mas ainda assim não comprometeu a realização, haja vista que você deixou disponível no Google Drive os diagramas para acompanhar em paralelo.
	DSD5	Com base nos feedbacks obtidos, buscar formas de simplificar o processo de inspeção, que (na minha opinião) é um pouco burocrático, principalmente considerando a tendência de processos de desenvolvimento cada vez mais ágeis. Com isso, penso que uma variação essencial dessa técnica, poderia estender seu campo de atuação.
	DSD4	No momento não.
GAD	GAD1	Automatizar esse processo de análise.
	GAD4	Incluir no treinamento um exemplo de todos os tipos de diagramas, como o de componente e sequência.
	GAD3	Não
	GAD2	A técnica é muito bem-vinda, pois auxilia a inspeção, porém ainda é realizada "manualmente". O gerente de domínio precisa ficar comparando imagens e abrindo a planilha com os itens a serem avaliados. Um maneira de melhorar a técnica e desenvolver uma ferramenta que, de certa maneira, automatize esse processo de inspeção e preenchimento da planilha.

Fonte: o autor

Apêndice E: LPS Mobile Media

A Mobile Media (MM) 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 (Geraldi e OliveiraJr, 2017b; Young, 2005).

A descrição dos casos de uso Mobile Media foram retirados do trabalho de Choma Neto (2017) e são apresentados a seguir:

. . . . . .

UC1: Log in
Categoria: mandatory
Resumo: Usuário insere login e senha e o sistema valida os dados.
Pré-Condições: Usuário deve estar registrado no sistema.
Cenário Principal:
1. Sistema apresenta a tela com opção de login e senha.
2. Usuário insere login e senha.
3. Sistema verifica se os dados digitados são válidos e permite que o usuário execute ações restritas.
Alternativos:
- Usuário pode cancelar operação a qualquer momento.
4. Sistema rejeita os dados, uma vez que login ou senha estavam incorretos. Retorna ao passo 1.
Pós-Condições: Usuário foi autenticado ou houve falha na autenticação, ou ainda o usuário cancelou a operação.

. . . . . .

UC2: Send Media

Categoria: optional
Resumo: Usuário seleciona e envia uma determinada mídia para outro dispositivo.
Pré-Condições: Deve haver mídias armazenadas no sistema.
Cenário Principal:
1. Usuário seleciona uma mídia. A mídia selecionada pode ser música, vídeo ou foto.
2. Usuário aciona a opção de enviar a mídia.
3. Sistema transfere o arquivo via SMS.
Alternativos:
- Usuário pode cancelar a operação em qualquer uma de suas ações.
Pós-Condições: Usuário enviou uma mídia para outro dispositivo ou a operação foi cancelada.

. . . . . .

UC3: Manage Album
Categoria: mandatory
Resumo: Usuário realiza o controle geral dos cadastros dos álbuns. Engloba as operações de criação, exclusão e listagem dos
álbuns.
Pré-Condições: Para a operação de exclusão, o sistema deve possuir pelo menos um álbum cadastrado, e o álbum a ser
excluído deve estar vazio. Para a listagem, devem existir mídias armazenadas no sistema.
Cenário Principal:
1. Usuário seleciona a opção de gerenciamento de álbuns.
2. Sistema apresenta as operações disponíveis para o gerenciamento do tipo de mídia selecionado.
3. Usuário seleciona uma operação.
- Create Album: ver item 3.1.
- Delete Album: ver item 3.2.
- List Album: ver item 3.3.
4. Sistema conclui a operação.
Alternativos:
- Usuário pode cancelar a operação em qualquer uma de suas ações.
Itens:
3.1. Create Album
3.1.1. Usuário acessa a opção de criar um álbum.

3.1.2. Sistema apresenta a tela para criação de um álbum.
3.1.3. Usuário entra com os dados do álbum.
3.1.4. Sistema verifica se já não existe algum álbum com as características informadas pelo usuário.
3.1.5. Sistema cria o álbum.
Alternativos desse item:
- Usuário pode cancelar a operação em qualquer uma de suas ações.
3.1.4. Já existe um álbum com as configurações estabelecidas. Sistema apresenta uma mensagem de erro para o
usuário e cancela a operação.
3.2. Delete Album
3.2.1. Usuário aciona a opção de excluir um álbum.
3.2.2. Sistema exige confirmação do usuário sobre a exclusão permanente do álbum.
3.2.3. Usuário confirma a exclusão.
3.2.4. Sistema verifica se o álbum selecionado está vazio.
3.2.5. Sistema exclui o álbum selecionado pelo usuário.
Alternativos desse item:
- Usuário pode cancelar a operação em qualquer uma de suas ações.
3.2.3. Usuário não confirma a exclusão, e a operação é cancelada.
3.2.4. Sistema verifica que o álbum selecionado possui mídias. Dessa forma, alerta o usuário que o álbum não está
vazio e cancela a operação.
3.3. List Album
3.3.1. Usuário seleciona a opção de visualizar todos os álbuns presentes no dispositivo.
3.3.2. Sistema apresenta uma lista com todos os álbuns disponíveis.
Alternativos desse item:
3.3.2. Não existem álbuns criados. O sistema exibe uma mensagem de erro ao usuário e cancela a operação.
Pós-Condições: Sistema realizou a operação desejada ou a ação foi cancelada.

. . . . . .

UC4: Manage Media
Categoria: mandatory
Resumo: Usuário realiza o controle geral dos cadastros de mídia. Engloba as operações de criação, exclusão e listagem dos registros.

Pré-Condições: Para a criação, o usuário deve estar logado no sistema e deve haver espaço em memória suficiente para o armazenamento da mídia criada. Para a exclusão, usuário deve estar logado e deve haver mídias armazenadas no sistema. Para a listagem, devem existir mídias armazenadas no sistema.

Cenário Principal:

1. Usuário seleciona a opção de gerenciamento de mídias.
2. Sistema apresenta os tipos de mídia disponíveis. Os tipos são foto, vídeo ou música.
3. Usuário seleciona um tipo de mídia.
4. Sistema apresenta as operações disponíveis para o gerenciamento do tipo de mídia selecionado.
5. Usuário seleciona uma operação.

- Create Media: ver item 3.1.
- Delete Media: ver item 3.2.
- List Media: ver item 3.3.

6. Sistema conclui a operação.

Alternativos:

- Usuário pode cancelar a operação em qualquer uma de suas ações. 3. Caso a mídia escolhida seja uma foto, «extend» Manage Photo. Caso seja um vídeo, «extend» Manage Video. Caso seja uma música, «extend» Manage Music.

Itens:

3.1. Create Media

3.1.1. Usuário acessa a opção de criar uma mídia.

3.1.2. «include» Log in.

3.1.3. Sistema apresenta a tela e as opções para a criação da mídia desejada.

3.1.4. Usuário utiliza as opções apresentadas para criar uma mídia.

3.1.5. Usuário escolhe salvar a mídia criada.

3.1.6. Sistema verifica se é possível salvar a mídia.

3.1.7. Sistema armazena a mídia criada pelo usuário.

Alternativos desse item:

- Usuário pode cancelar a operação em qualquer uma de suas ações.

3.1.6. Sistema não consegue encontrar espaço suficiente para armazenamento da mídia criada. Sistema apresenta
uma mensagem de erro ao usuário e a operação é cancelada.

3.2. Delete Media

3.2.1. Usuário indica a ação de excluir uma mídia.

3.2.2. «include» Log In.

3.2.3. Sistema exige confirmação sobre a exclusão permanente.
3.2.4. Usuário confirma sua opção de excluir.
3.2.5. Sistema verifica se a mídia excluída não está associada a um contato.
3.2.6. Sistema exclui a mídia da memória do dispositivo.
Alternativos desse item:
- Usuário pode cancelar a operação em qualquer uma de suas ações.
3.2.4. Usuário não confirma a exclusão, e a operação é cancelada.
3.2.5. A mídia está sendo utilizada, podendo estar, por exemplo, vinculada a um determinado contato. Neste caso,
sistema alerta o usuário sobre a utilização e cancela a exclusão.
3.3. List Media
3.3.1. Usuário acessa a opção de listar mídias disponíveis.
3.3.2. Sistema exibe uma lista de todas as mídias ordenadas pelas mais vistas.
Alternativos desse item:
3.3.2. Não existem mídias armazenadas. O sistema exibe uma mensagem de erro ao usuário e cancela a operação.
Pós-Condições: Sistema realizou a operação desejada ou a ação foi cancelada.

. . . . . .

UC5: Manage Favourite Media
Categoria: optional
Resumo: Usuário realiza o controle acerca de mídias favoritas. Engloba as operações de definição de mídia favorita e de
listagem das mídias favoritas.
Pré-Condições: Para a operação de marcar mídia como favorita, deve haver mídias armazenadas no sistema. Para a listagem,
devem existir mídias armazenadas no sistema e marcadas como favorita. **Cenário Principal:**
1. Usuário seleciona a opção de gerenciamento de mídias favoritas.
2. Sistema apresenta as operações disponíveis para o gerenciamento do tipo de mídia selecionado.
3. Usuário seleciona uma operação.
- Set Media as Favourite: ver item 3.1.
- List Favourite Media: ver item 3.2.
4. Sistema conclui a operação.

Alternativos:
- Usuário pode cancelar a operação em qualquer uma de suas ações.

Itens:
3.1. Set Media as Favourite
3.1.1. Usuário seleciona a opção de marcar a mídia como favorita.
3.1.2. Sistema verifica se a mídia já não está marcada como favorita.
3.1.3. Sistema seta a mídia como favorita.

Alternativos desse item:
- Usuário pode cancelar a operação em qualquer uma de suas ações.

3.1.2. A mídia já estava marcada como favorita. Sistema exibe uma mensagem de erro para o usuário e cancela a
operação.

3.2. List Favourite Media
3.2.1. Usuário acessa a opção de visualizar suas mídias favoritas.
3.2.2. Sistema exibe uma lista com todas as mídias marcadas como favoritas.

Alternativos desse item:

3.2.2. O usuário ainda não marcou nenhuma mídia como favorita. Sistema exibe uma mensagem de erro para o
usuário. A operação é cancelada.

Pós-Condições: Sistema realizou a operação desejada ou a ação foi cancelada.

UC6: Play Media
Categoria: mandatory
Resumo: Usuário executa uma mídia de acordo com as possibilidades do produto.
Pré-Condições: Devem existir mídias no sistema de arquivos do dispositivo.
Cenário Principal:
1. Usuário seleciona uma mídia de acordo com as possibilidades do produto (a mídia pode ser música, foto ou vídeo).
2. Usuário seleciona a opção de executar a mídia.
3. Sistema executa a mídia selecionada pelo usuário.

Alternativos:
- Usuário pode cancelar a operação em qualquer uma de suas ações.

3. Caso a mídia escolhida seja uma foto, «extend» Visualize Photo. Caso seja um vídeo, «extend» Play Video. Caso seja

uma música, «extend» Play Music.
Pós-Condições: Usuário executou uma mídia ou a operação foi cancelada.

. . . . . .

UC7: Add Media to Album
Categoria: mandatory
Resumo: Usuário seleciona uma mídia e a vincula a um determinado álbum armazenado no sistema.
Pré-Condições: Devem existir mídias e álbuns armazenados no sistema.
Cenário Principal:
1. Usuário seleciona uma mídia.
2. Usuário aciona a opção de adicionar essa mídia a um álbum.
3. Sistema verifica se a mídia já está vinculada a um álbum.
4. Sistema exibe as opções de álbuns disponíveis.
5. Usuário seleciona um dos álbuns disponíveis.
6. Sistema adiciona a mídia ao álbum selecionado.
Alternativos:
- O usuário pode cancelar a operação em qualquer uma de suas ações.
3. Usuário seleciona uma mídia já vinculada a um determinado álbum. Sistema verifica se o usuário deseja criar uma cópia da
mídia para o álbum a ser selecionado. Caso o usuário confirme, uma cópia é criada. Caso contrário o sistema move a mídia
para o álbum a ser escolhido, eliminando-a do álbum antigo.
4. Não existe qualquer álbum. O sistema exibe uma mensagem de erro para o usuário e cancela a operação.
6. A mídia já se encontra no álbum selecionado. O sistema exibe uma mensagem de erro para o usuário e cancela a operação.
Pós-Condições: Usuário adicionou uma mídia a um álbum ou a operação foi cancelada.

. . . . . .

UC8: Link Media with Address Book Entry
Categoria: optional
Resumo: Usuário vincula uma mídia a um determinado contato para que ela seja reproduzida quando há uma chamada

desse contato.

Pré-Condições: Devem existir mídias e contatos armazenados no dispositivo.

Cenário Principal:

1. Usuário seleciona uma música armazenada no sistema.
2. Usuário aciona a opção de relacionar essa mídia a um contato.
3. Sistema exibe uma lista com os contatos contidos no dispositivo.
4. Usuário seleciona um determinado contato na lista.
5. Sistema verifica se o contato selecionado já não possui uma mídia vinculada do mesmo tipo da mídia selecionada.
6. Sistema vincula a mídia ao contato selecionado.

Alternativos:

- Usuário pode cancelar a operação em qualquer uma de suas ações.

1. A mídia selecionada pode ser uma foto.

3. Não existem contatos registrados no dispositivo. Sistema apresenta uma mensagem de erro ao usuário e cancela a operação.

5. O contato selecionado já apresenta uma mídia do tipo da selecionada vinculada. O sistema informa o usuário e exibe a opção de vincular a nova mídia ou manter a que já estava. Retorna ao passo 6.

Pós-Condições: Usuário vinculou uma mídia a um contato ou a operação foi cancelada.

. . . . . .

UC9: View/Hear Media from Incoming Caller

Categoria: optional

Resumo: Ao receber uma chamada, o sistema reproduz a mídia vinculada ao contato.

Pré-Condições: O contato deve ter alguma mídia vinculada.

Cenário Principal:

1. Usuário recebe uma chamada de um determinado contato.
2. Sistema verifica se esse contato está vinculado a uma determinada mídia (música ou foto).
3. Sistema intercepta a chamada e reproduz a mídia personalizada.

Alternativos: 2. O contato não apresenta nenhuma mídia vinculada. O sistema reproduz o toque e a imagem padrão de chamadas.

Pós-Condições: Sistema reproduziu uma mídia personalizada para um contato ou reproduziu a mídia padrão para todos os contatos.

. . . . . .

UC10: Label Files
Categoria: optional
Resumo: Usuário insere um rótulo para uma determinada mídia, podendo ser vídeo, foto ou música.
Pré-Condições: Devem existir mídias no sistema de arquivos do dispositivo.
Cenário Principal:
1. Usuário seleciona uma mídia. A mídia selecionada pode ser música, vídeo ou foto.
2. Usuário acessa a opção de inserir um rótulo para aquela mídia.
3. Sistema verifica se a mídia já possuía um rótulo associado.
4. Sistema apresenta a tela para a inserção do rótulo.
5. Usuário insere um rótulo para aquele arquivo.
6. Sistema verifica se algum arquivo já possui aquele rótulo.
7. Sistema vincula o rótulo ao arquivo.
Alternativos:
- Usuário pode cancelar a operação em qualquer uma de suas ações.
3. O arquivo já possuía um rótulo. Sistema informa ao usuário se ele deseja substituir o rótulo existente.
5. Usuário deixou o rótulo em branco. Sistema informa o erro ao usuário. Retorna ao passo 4.
6. Sistema verifica que já existe um arquivo com aquele rótulo e apresenta uma mensagem de erro para o usuário. Retorna
ao passo 4.
Pós-Condições: Usuário inseriu um rótulo para uma mídia ou a operação foi cancelada.