Universidade Federal da Bahia
Instituto de Matemática e Estatística

Programa de Pós-Graduação em Ciência da Computação

PROCESS SMELLS: UM CATÁLOGO DE BAD SMELLS PARA PROCESSO DE SOFTWARE

Edison de Jesus Santos

DISSERTAÇÃO DE MESTRADO

Salvador
28 de Novembro de 2019

EDISON DE JESUS SANTOS

PROCESS SMELLS: UM CATÁLOGO DE BAD SMELLS PARA PROCESSO DE SOFTWARE

Esta Dissertação de Mestrado foi apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientadora: Profa. Dra. Rita Suzana Pitangueira Maciel
Co-orientador: Prof. Dr. Cláudio Sant'Anna

Salvador
28 de Novembro de 2019

Sistema de Bibliotecas - UFBA

Santos, Edison de Jesus (usado em CITACOES).

Process Smells: Um Catálogo de *Bad Smells* para Processo de *Software* / Edison de Jesus Santos – Salvador, 2019.

180p.: il.

Orientadora: Prof. Dr. Profa. Dra. Rita Suzana Pitangueira Maciel.

Co-orientador: Prof. Dr. Prof. Dr. Cláudio Sant'Anna.

Dissertação (Mestrado) – Universidade Federal da Bahia, Instituto de Matemática e Estatística, 2019.

1. Detecção de *Process Smell*. 2. Qualidade de Processo de Software. 3. Engenharia de Software . I. Maciel, Rita Suzana Pitangueira. II. Sant'Anna, Cláudio. III. Universidade Federal da Bahia. Instituto de Matemática e Estatística. IV. Título.

CDD – XXX.XX

CDU – XXX.XX.XXX

TERMO DE APROVAÇÃO

EDISON DE JESUS SANTOS

PROCESS SMELLS: UM CATÁLOGO DE BAD SMELLS PARA PROCESSO DE SOFTWARE

Esta Dissertação de Mestrado foi julgada adequada à obtenção do título de Mestre em Ciência da Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação da Universidade Federal da Bahia.

Salvador, 28 de Novembro de 2019

Profa. Dra. Ana Patricia Fontes Magalhães Mascarenhas
Universidade Estadual da Bahia - UNEB

Prof. Dr. Ivan do Carmo Machado
Universidade Federal da Bahia - UFBA

Profa. Dra. Rita Suzana Pitangueira Maciel
Universidade Federal da Bahia - UFBA

"*Process Smell : Um Catálogo de Bad Smells para Processos de Software*"

Edison de Jesus santos

Dissertação apresentada ao Colegiado do Programa de Pós-Graduação em Ciência da Computação na Universidade Federal da Bahia, como requisito parcial para obtenção do Título de Mestre em Ciência da Computação.

Banca Examinadora

Profª. Drª. Rita Suzana Pitangueira Maciel (orientadora-UFBA)

Prof. Dr. Ivan do Carmo Machado (UFBA)

Profª. Drª. Ana Patrícia Fontes Magalhães (UNEB)

Dedico a chama para iniciar esse trabalho aos professores da minha graduação, Indymar Oliveira pelos estímulos, a Nidyana Miranda e Pedro Kislansky pelos impulsionamentos e a Jocelma Rios que me inspira.
Eu dedico essa trabalho a um resultado coletivo e social para todos os meus familiares, em especial Ingridi Nascimento e Ricardo Souza, aos amigos, principalmente Érique Batista, Maiana Lima, Danilo Pires, Igor da Hora, Luís Reis, e colegas, entre estes Larrise Cerqueira e Lucinéia Souza, que se reconhecem, me incentivaram e se sentiram representados nesta conquista.

AGRADECIMENTOS

Agradeço a Deus por todas as coisas, assim como pelas forças sempre renovadas. Sou muito grato aos meus pais Aneilda Mendes de Jesus e Eciomar Evangelista dos Santos que me deram suporte e referências para que eu pudesse superar os desafios. Agradeço também aos meus guias, Rita Suzana e Cláudio Sant'Anna pelo direcionamento, discussões e paciência para que este resultado fosse alcançado. Em especial à professora Rita, com quem mais compartilhei as angústias durante o processo da produção acadêmica. Por últimos, mas muito especial, agradeço a minha esposa Vanessa Souza pelo respeito e compreensão ao esforço e tempo que eu dediquei a essa pesquisa. Não esquecendo também de todos os amigos, colegas de trabalho, colegas da pós-graduação e participantes das entrevistas que opinaram e colaboraram com o produto da pesquisa dessa dissertação.

Não se pode criar experiência. É preciso passar por ela.

—ALBERT CAMUS

RESUMO

O uso sistemático de especificação de processo de software favorece a qualidade do produto gerado e orienta os passos para a construção do software aderente a qualidade esperada dos projetos de desenvolvimento de software. Processos de software evoluem junto às necessidades da instituição e dos profissionais que o utilizam, e necessitam ser monitorados e avaliados constantemente para manter as suas qualidades. Para avaliar processos, as formas mais conhecidas entre as práticas da indústria e a literatura da Engenharia de Software se utilizam de (i) dados obtidos após a execução do processo ou (ii) de simulação. Em ambos os casos não é possível antever problemas na execução do processo de software em determinado projeto de desenvolvimento. Processos de software são comumente especificados e representados por linguagens de modelagem de processo de software conhecidas como *Software Process Modeling Language* (SPML). Dentre essas linguagens, a *Software & Systems Process Engineering Meta-Model* (SPEM) se destaca por ser um perfil *Unified Modeling Language* (UML) para modelagem de processo de software e sistemas. Apesar das SPML, como o SPEM, serem usadas para melhorar o entendimento de um processo, a especificação de um processo pode ser feita de forma inadequada, ferindo fatores de qualidade desejadas para tal processo. Este fenômeno pode ser comparado ao conceito de *bad smells*, que são anomalias de *design* em código de software. Assim, o conceito de *process smell*, introduzido neste trabalho, observa problemas no *design* de processos de software. Sendo assim, a ocorrência de *process smell* na especificação de um processo pode gerar impactos negativos à qualidade do processo e por consequência afetar a qualidade do produto de software. Neste sentido, esta pesquisa teve como objetivo especificar um catálogo de *process smells* para apoiar a identificação de violações estruturais que correspondem a anomalias em processos de software especificados com SPEM. Para tanto foi estabelecida uma metodologia para especificação e avaliação da proposta composta de duas etapas, ambas contando com validações feitas por estudo de entrevista com profissionais da Engenharia de Software. Na primeira etapa foi estabelecido o catálogo de *process smells*. Na segunda etapa foram estabelecidas estratégias e detectados os *process smells*. Como resultados, esta pesquisa mostrou que o catálogo proposto de *process smells* é notadamente aceito pelos engenheiros de software. Contudo, avaliando o contexto da execução de um processo na prática, podem existir *process smells* mais significativos do que outros. Espera-se que o catálogo produzido possa apoiar a identificação de *process smells* em modelos de processo de *software* com o objetivo de indicar pontos nos quais o processo pode ser melhorado, antes mesmo da sua primeira execução, evitando problemas que afetam negativamente os atributos de qualidade do processo. Adicionalmente, foram obtidos aspectos que podem orientar melhorias deste catálogo em trabalhos futuros.

Palavras-chave: Modelo de Processo de Software, Bad Smells, Atributos de Qualidade, Propriedades de Qualidade, Process Smells

ABSTRACT

The systematic use of software process specification favors the quality of the generated product and guides the steps for the construction of the software adhering to the expected quality of software development projects. Software processes evolve with the needs of the institution and the professionals who use it, and need to be monitored and evaluated constantly to maintain their qualities. To evaluate processes, the most well-known forms among industry practices and the Software Engineering literature use (i) data obtained after the execution of the process or (ii) simulation. In both cases it is not possible to foresee problems in the execution of the software process in a given development project. Software processes are commonly specified and represented by software process modeling languages known as *Software Process Modeling Language* (SPML). Among these languages, *Software & Systems Process Engineering Meta-Model* (SPEM) stands out for being a *Unified Modeling Language* (UML) profile for modeling software and systems processes. Although SPML, like SPEM, are used to improve the understanding of a process, the specification of a process can be done inappropriately, hurting the desired quality factors for that process. This phenomenon can be compared to the concept of bad smells, which are design anomalies in software code. Thus, the concept of process smell, introduced in this work, observes problems in the design of software processes. Thus, the occurrence of process smell in the specification of a process can generate negative impacts on the quality of the process and consequently affect the quality of the software product. In this sense, this research aimed to specify a catalog of process smells to support the identification of structural violations that correspond to anomalies in software processes specified with SPEM. For this purpose, a methodology for specification and evaluation of the proposal was established, composed of two stages, both with validations made through an interview study with professionals of Software Engineering. In the first stage, the process smells catalog was established. In the second stage, strategies were established and process smells were detected. As a result, this research showed that the proposed catalog of process smells is notably accepted by software engineers. However, assessing the context of running a process in practice, there may be more significant process smells than others. It is hoped that the catalog produced can support the identification of process smells in software process models in order to indicate points at which the process can be improved, even before its first execution, avoiding problems that negatively affect quality attributes. of the process. Additionally, aspects were obtained that can guide improvements to this catalog in future works.

Keywords: Software Process Model, Bad Smells, Quality Attributes, Property Attributes, Process Smells

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

PML *Process Modeling Language* 10
SPML *Software Process Modeling Language* 10
OMG *Object Management Group* 23
BNF *Backus-Naur form* 10
CMMI *Capability Maturity Model Integration* 60
MPS.BR Melhoria do Processo de Software Brasileiro 60
UML *Unified Modeling Language* 21
BPMN *Business Process Modeling Notation* 118
SPEM *Software & Systems Process Engineering Meta-Model* 118
MOF *MetaObject Facility Specification* 23
XML *eXtensible Markup Language* 120
HTML *Hypertext Markup Language* 120
EPF *Eclipse Process Framework* 120
CCSA *Certification in Control Self-Assessment* 60

Capítulo 1

INTRODUÇÃO

Um processo de software é um conjunto de atividades utilizadas por empresas de desenvolvimento de software para a construção de um produto de software. Estes processos incluem diferentes tipos de atividades desde as diretamente voltadas para a confecção do produto como especificação, codificação, validação ou atividades gerenciais e de acompanhamento da produção, entre outras (SOMMERVILLE, 2011). Um processo de software fornece suporte para a construção, mudança e melhoria do produto de software, assim como é modificado para comportar evolutivamente esta produção (GENVIGIR; FILHO; SANT'ANNA, 2003). Modelos genéricos como o cascata e o espiral são alicerces para a construção dos processos que serão especificados e executados no dia a dia das instituições (SOMMERVILLE, 2011).

Processos de desenvolvimento de software assumem um papel importante para a indústria de software, pois pode ser considerado um guia que orienta a construção do software. Comumente, o uso de um processo de forma sistemática busca proporcionar benefícios para a construção de um produto de software. Alguns dos benefícios podem ser alcançar o sucesso do projeto, assim como a qualidade do produto de software a ser entregue. Estes benefícios são mais facilmente alcançados se as atividades do processo forem seguidas adequadamente pelas pessoas envolvidas na construção do software. As pessoas envolvidas na construção do produto de software precisam compreender e interagir com processos, assim, normalmente são utilizados modelos de processo (diagramas visuais) para representar os processos de desenvolvimento de software. Sendo assim, um modelo de processo é uma representação simplificada do processo e contém uma visão sob determinadas perspectivas e informações em relação ao processo como um todo (GENVIGIR; FILHO; SANT'ANNA, 2003).

Para alcançar especificações de modelos de processo de software de forma mais precisa, com elementos comuns e sintaxe definida, foram criadas as linguagens de modelagem de processo, *Process Modeling Language* (PML) e, para processos de software, as *Software Process Modeling Language* (SPML). SPMLs fornecem recursos para modelar elementos básicos e avançados de um modelo de processo de software, e seus relacionamentos. Com

este propósito foi desenvolvida a SPML *Software & Systems Process Engineering Meta-Model* (SPEM), proposto pela *Object Management Group* (OMG) (OMG, 2008). SPEM é um metamodelo para definição de processos de software, seus componentes, descrições e restrições de comportamentos (GENVIGIR; FILHO; SANT'ANNA, 2003). A SPEM fornece características que apoiam a modelagem de processo de software, possui uma semântica rica para definição de elementos do processo de software, fornece capacidade de relacionamento entre diferentes processos e modelos com diferentes ciclos de vida, reutilização de padrões, verificação da variabilidade de processos e extensões por *plug-ins*.

Contudo, ter uma linguagem para modelar o processo de software não garante por si só que o processo de software terá qualidade. Os processos variam de acordo com cada empresa de desenvolvimento de software, assim a especificação e modelagem se utilizam de autonomia para estabelecer as atividade e tarefas do processo conforme venha a ser necessário. Porém esta liberdade pode dar margem a modelagens difíceis de serem compreendidas ou modificadas. Por exemplo, sendo especificados muitos fluxos de decisão, conforme na figura 1.1, o processo pode se tornar complexo devido a quantidade de decisões (losangos) necessárias para ser completada uma atividade com suas as tarefas (pentágonos). Ao possuir muitas atividades o processo se torna longo e assim mais difícil de ser entendido. Assim em ambos casos a compreensibilidade é impactada negativamente. Em casos onde a especificação de um processo possui muitas dependências entre as atividades, este processo se torna difícil de modificar impactando assim na sua modificabilidade. Estas situações podem ser percebidas como anomalias e para evitá-las, processos precisam ser avaliados continuamente.

Diversas são as propostas para a avaliação de qualidade do processo encontradas na literatura (KROEGER; DAVIDSON; COOK, 2014) (MOHAGHEGHI; DEHLEN; NEPLE, 2009) (OCA et al., 2015). A engenharia de processo de software possui quatro métodos de avaliação de qualidade mais difundidos que são avaliação: (i) do produto frente ao processo, (ii) por modelos de referência, (iii) através especificações formais e a (vi) baseada na visão sistêmica da engenharia de software centrada no ser humano (esta última será tratada desta parte do texto em diante por "avaliação centrada no ser humano") (KROEGER, 2011).

A (i) avaliação do produto frente ao processos avalia o processo por inferências estatísticas com base nos produtos resultantes. A (ii) avaliação por modelos de referência estabelece um modelo de referência que servirá de base para a avaliação do processo de um projeto ou da organização. A (iii) avaliação do processo a partir de uma linguagem formal se utiliza de verificações formais ou de simulação. A (iv) avaliação centrada no ser humano utiliza a abordagem de avaliação por múltiplas perspectivas e se utiliza de dados obtidos após execução do processo, podendo também utilizar simulação (KROEGER; DAVIDSON; COOK, 2014).

A estratégia do uso de simulação favorece a avaliação antes mesmo da execução do processo. A simulação requer a preparação de dados simbólicos, procedimentos e configuração de ferramentas para atingir a eficiência adequada. Apesar da simulação ser uma alternativa para analisar a execução do processo antes da sua primeira execução em um projeto de desenvolvimento de software, ela requer uma especialização profissional para

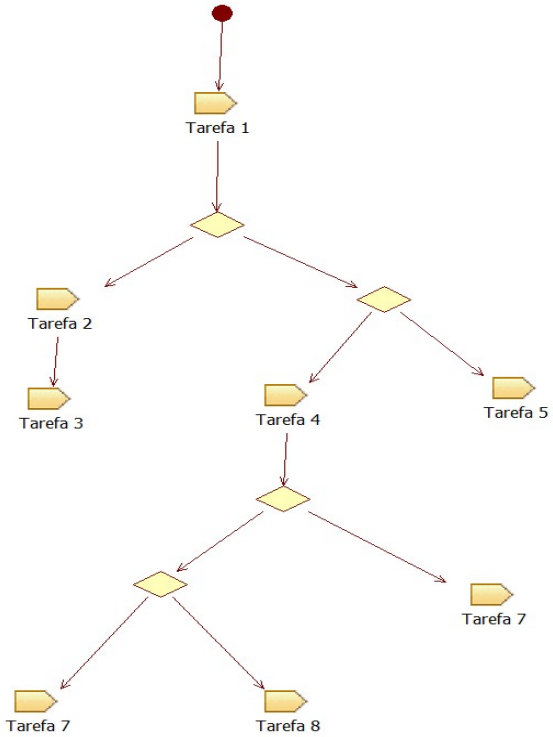


Figura 1.1 Exemplo de *process smell*

a sua utilização, o que não é trivial e pode nem sempre representar a realidade de forma generalizada da indústria (SILVA et al., 1999).

Alguns estudos de avaliação de processo através da medição de propriedades estruturais, como por exemplo (GARCÍA et al., 2003); (GARCÍA et al., 2005); (RUIZ-RUBE; DODERO; COLOMO-PALACIOS, 2015), avaliam o processo na sua totalidade, o que não favorece uma localização precisa da origem das anomalias no processo que precisam ser tratadas. Estes estudos apresentam uma abordagem de avaliação de baixo para cima (*bottom - up*), uma vez que os resultados das medições precisam ser interpretados de modo a compreender as origens de problemas que afligem o processo medido, assim são descobertas os impactos mas não as causas dos problemas (MARINESCU, 2004).

Explorando um pouco outros cenários de avaliação de qualidade em engenharia de software, algumas estratégias realizam avaliação de elementos específicos da estrutura de um software, como é o caso de avaliação de código fonte (CHIDAMBER; KEMERER, 1994a). O código fonte, o principal recurso para entrega do produto de software, possui vasta pesquisa quanto a sua qualidade, tão como a ferramentas de apoio a estas avaliações. As estratégias de avaliação consistem em boa parte na detecção de estruturas do código como métodos, classes, bibliotecas e outras que afetam alguma qualidade esperada pelos envolvidos com a produção do código. De modo a caracterizar alguns problemas de *design* de código costumeiramente percebidos pelos desenvolvedores, Fowler et. al. (1999) estabelecem algumas taxonomias nomeadas como *bad smells*.

Fowler et. al. (1999) definiram o termo *bad smells*, também conhecido por *code smells*

uma vez que se trata de código fonte, para apontar decisões tomadas sobre o *design* que infringem boas práticas ou padrões de código fonte de software que devem ser evitadas. Estes *bad smells* afetam características fundamentais da estrutura do código e reduzem a sua qualidade. A *Large Class*, por exemplo, é um *bad smell* onde uma classe entrega diferentes objetivos do software, se tornando extensa e complexa. Contudo, Fowler não propõe de forma sistemática como detectar os *bad smells*.

Para realizar a detecção dos *bad smells* de forma sistemática, Marinescu (2004) propõe um método chamado de estratégia de detecção. Este método apoia a identificação de falhas no *design* de código, considerando o uso de métricas baseadas em regras que apontam violações aos princípios de *design*. As regras passam pela filtragem (definição dos valores limites esperados) e composição (relação das métricas entre si baseada no uso de operadores lógicos) para estabelecer a detecção de um determinado *bad smell*. O método de detecção dos *bad smells* é uma abordagem de cima para baixo (*top - down*), pois estabelece de forma antecipada os problemas de *design* que se espera identificar, assim como define regras que podem ser quantificadas por métricas para tal detecção. Com a abordagem de cima para baixo já existe uma forma prévia de interpretar os problemas de *design* e assim o tratamento do problema é facilitado.

Neste trabalho investigamos se o conceito de *bad smells* pode ser aplicado para processo de software, uma vez que o processo também pode ser considerado como software (OSTERWEIL, 1987). Uma similaridade entre processo e código fonte orientado a objeto está na existência de elementos que desempenham funções similares. As atividades do processo e classes do código orientado a objeto são os elementos que devem cumprir um propósito, por exemplo, a atividade de planejar o cronograma de um projeto ou a classe que deve gerar um arquivo de impressão. Já as tarefas representam cada ação que deve ser feita para completar uma atividade e podem ser comparadas aos métodos para as classes (SILINGAS; MILEVICIENE, 2007).

Algumas características apoiam a possibilidade de identificar *bad smells* em um processo de software. Uma característica é a similaridade entre código fonte orientado a objeto e os elementos do processo. Outra é o fato de que a especificação e os ajustes no processo são realizados em seus elementos estruturais, o que pode ocasionar arranjos desse elementos de forma a afetar qualidades do processo que são esperada por seus utilizadores. Em adaptação ao contexto de processo foi estabelecido o termo *Process Smell* que corresponde a identificação de anomalias no *design* do processo que impactem negativamente seus atributos de qualidade.

Um *process smell* deve ser composto de uma definição, impactos negativos, representação visual e estratégia de detecção. Além disso, por se tratar de uma avaliação com abordagem de cima para baixo se fazia necessário definir o conjunto de *process smells* a serem verificados, o que originou o catálogo de *process smells* desta pesquisa.

1.1 OBJETIVOS E CONTRIBUIÇÕES

Este trabalho tem como objetivo geral apresentar o conceito e especificar um catálogo de *process smells* para apoiar a identificação de anomalias em *designs* de processo de software especificados com SPEM. Como objetivos específicos, temos:

1. elencar e especificar os *process smells*;

2. estabelecer as estratégias de detecção dos *process smells*;

3. validar a aplicação do catálogo em modelos de processo de software.

A versão inicial do catálogo de *process smells* para especificação foi publicada em SPEM (SANTOS; SUZANA; SANT'ANNA, 2018).

Espera-se que através deste catálogo, seja possível identificar *process smells* ainda em tempo de *design* do processo. Sendo assim, evitar verificações tardias ou em tempo de execução, evitar prejuízos à qualidade do produto de software ou mesmo prejuízos aos projetos por ter sua execução sobre um processo com baixa qualidade.

1.2 METODOLOGIA

A metodologia para atender a proposta de *process smells* seguiu a abordagem de cima para baixo (*top – down*). Em uma abordagem de cima para baixo, inicialmente são determinados os objetos e como estes serão verificados. Desta forma inicialmente foram especificados os *process smells* para posteriormente realizar as suas detecções em processo de software. Para validar tanto os *process smells* como as suas detecções foram realizados dois estudos de entrevista, um para cada propósito. A exploração desse fenômeno requer a captura da percepção dos analistas de processo e demais os utilizadores.

Um estudo de natureza exploratória é estabelecido quando conceitos iniciais precisam ser identificados, assim como quando se quer descobrir novas possibilidades em um determinado campo de pesquisa (MARCONI; LAKATOS et al., 2002). Neste sentido o estudo de entrevista é um método de pesquisa que dispõe de flexibilidade, visto que possui maior foco em coletar as informações necessárias para a pesquisa do que em seguir estritamente um questionário predefinido, assim pode alcançar profundidade na coleta de informações (JÚNIOR; JÚNIOR, 2012). Entre diferentes tipos , a entrevista focalizada apresenta um estímulo (objeto do estudo) e a partir deste é obtida a percepção do entrevistado seguindo um guia com os tópicos a serem abordados (FLICK, 2008).

A metodologia desse trabalho foi dividido em duas etapas conforme a figura 1.2. A primeira etapa teve como objetivo preparar e validar o catálogo de *process smells*. A primeira etapa consistiu em três passos principais explicados a seguir:

1. definição das similaridades entre o paradigma de orientação a objetos e o SPEM – Este passo estabeleceu uma relação entre os elementos do paradigma de orientação a objetos e o SPEM;

2. seleção e adaptação de *Bad Smells* para o SPEM – Neste passo foi realizado um levantamento das características dos *bad smells* que fossem mais adequadas para adaptação para o SPEM. Posteriormente foi feita a seleção dos *bad smells* e adaptação para o SPEM.

3. seleção de fatores de qualidade – A seleção dos fatores de qualidade buscou identificar atributos de qualidade que pudessem ser avaliados antes da execução do processo e pudessem coincidir com atributos impactos pelos *process smells*.

Com os passos 1.2 e 1.3 concluídos foi possível preparar a versão inicial do catálogo de *process smells* e por fim aplicá-lo para validação através do estudo de entrevista com analistas de processos de software. O estudo de entrevista foi aplicado para validar as caracterizações dos *process smells* composta pelas definições, seus possíveis impactos que influenciam na redução da qualidade em processo de software.

A segunda etapa da metodologia teve como objetivo detectar e validar os *process smells* em processos reais de desenvolvimento de software. Esta etapa consistiu em dois passos principais explicados a seguir:

1. definição das estratégias de detecção de *process smells* - Neste passo foram estabelecidas as estratégias de detecção de cada *process smell* do catálogo.

2. definição das estratégias dos valores limiares - Neste passo foi estabelecido o conjunto de métricas e valores limiares incorporados as estratégias de detecção de cada *process smell* do catálogo.

A detecção dos *process smells* contou com um *script* semi automatizado aplicado aos processos de desenvolvimento de software para medir o processo detectando os *process smells*. Assim os *process smells* detectados foram apresentados para validação pelo estudo de entrevista com os profissionais envolvidos com os processos.

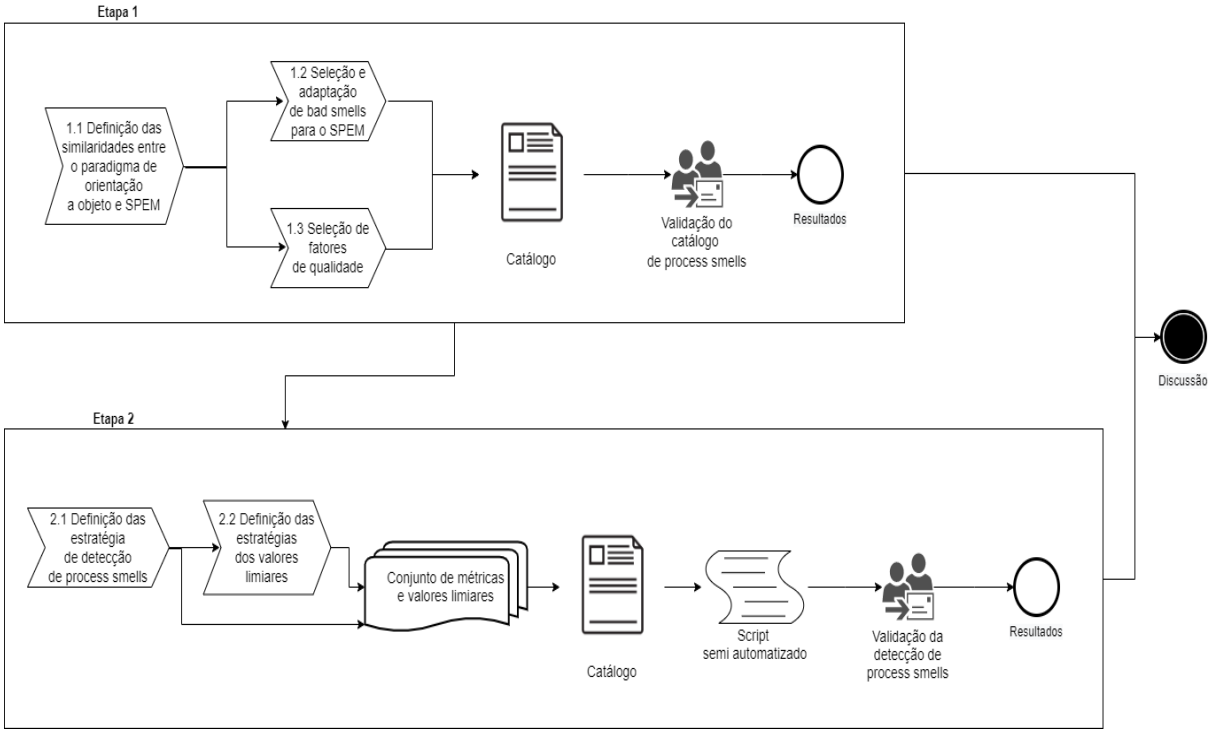


Figura 1.2 Metodologia

A dissertação está organizada em oito capítulos. O próximo capítulo contém a revisão bibliográfica, composta pelas subseções: processo de software e modelos de processo de

software, linguagem de modelagem de processo de software, modelos de avaliação de processo de software, e por fim *bad smells*. O terceiro capítulo contém os trabalhos relacionados. No quarto capítulo é apresentado o processo de elaboração e o catálogo de *process smells* resultante. O quinto, sexto e sétimo capítulo constam os resultados da pesquisa respectivamente com o primeiro e segundo estudo de entrevista e a discussão. Por fim, no oitavo capítulo são apresentadas as conclusões da pesquisa e as sugestões de trabalhos futuros.

Capítulo 2

REVISÃO BIBLIOGRÁFICA

Esta seção apresenta os conceitos relevantes para o entendimento da proposto da pesquisa desta dissertação.

2.1 PROCESSOS DE SOFTWARE E MODELOS DE PROCESSOS DE SOFTWARE

A elaboração de um processo busca generalizar a solução de um problema e cada execução individual do processo é considerada uma instância (OSTERWEIL, 1987). Por sua vez, um processo de software descreve os passos para o desenvolvimento de um produto de software. Assim, as instâncias do processo de software correspondem ao ciclo de vida deste software, desde o momento em que o usuário final fornece as informações iniciais que orientam a construção, até a entrega do produto final de software (OSTERWEIL, 1987). A declaração de Osterweil (1987) que estabelece o processo de software como software também foi um marco que apoiou a evolução de linguagens e modelos de processo de software promovendo maior capacidade de compreensão e institucionalização deste tipo de processos (GARCÍA-BORGOÑON et al., 2014).

A representação das atividades do mundo real de um processo de produção de software constitui um modelo de processo de software (GENVIGIR; FILHO; SANT'ANNA, 2003). Em um modelo de processo devem estar definidos seus elementos (atividades, produtos de trabalho, etc..) e como eles estão relacionados. Para cada tarefa devem ser estabelecidos os pré-requisitos, suas consequências e sincronização com as demais tarefas. A modelagem do processo deve tanto possibilitar a comunicação e entendimento do processo, bem como, apoiar o seu reuso, evolução e gerenciamento. Para isto, a modelagem do processo deve ser padronizada com propósito de fornecer capacidade de gerenciamento, revisão, suporte a ferramentas, contribuição para melhoria organizacional, base para medição e redução de esforços desnecessários (GENVIGIR; FILHO; SANT'ANNA, 2003).

De acordo com Sommerville (2011), os modelos cascata, desenvolvimento incremental e engenharia de software orientada a reuso, devem ser considerados modelos genéricos de processo de software. Estes não se apresentam como representações definitivas dos

processos de software, mas sim como um *framework*, definindo o alicerce sobre o qual o processo será estruturado e customizado para atender as necessidades de geração de um determinado produto de software. Nesta dissertação, o conceito de modelo de processo não corresponderá aos modelos genéricos citados acima e sim a representações definitivas das atividades que constituem um processo de desenvolvimento de software.

2.2 LINGUAGEM DE MODELAGEM DE PROCESSO DE SOFTWARE

Uma *Process Modeling Language* (PML) é uma linguagem para descrever modelos de processo, independe de ferramenta e geralmente expressada pela gramática *Backus-Naur form* (BNF). Assim uma PML possui declarações com palavras-chaves e valores que podem descrever diagramas, atividades, entradas, saídas e ícones para representar elementos de um processo (GRUBEL, 1995). As PMLs surgem da necessidade das indústrias em definir seus processos de negócios, processos produtivos, industriais de manufatura e outros. Uma PML oferece suporte às colaborações empresariais, suporte a estruturas hierárquicas, interpretação flexível, robustez na definição do processo e suporte à integração.

De forma específica, para atender a modelagem de processo de software, foi proposta uma PML denominada de *Software Process Modeling Language* (SPML) (GARCÍA-BORGOÑON et al., 2014). Entre várias SPML, destaca-se a *Software & Systems Process Engineering Meta-Model* (SPEM) por ser um padrão aberto. A SPML SPEM surgiu em 2002 como iniciativa da *Object Management Group* (OMG) e faz parte do grupo de SPML baseadas em modelo. A SPEM é um meta-modelo baseado em *MetaObject Facility Specification* (MOF) que é uma linguagem para especificação de meta-modelo da própria OMG. Adicionalmente, SPEM compõe também um *framework* conceitual que apoia a modelagem, documentação, apresentação, gerenciamento e interações, tendo como base métodos e processos de desenvolvimento. Sendo assim, é composto por elementos, que colaboram entre si, como atividades, tarefas, regras, produtos de trabalho e outros (OMG, 2008). A tabela 2.1 apresenta os elementos atividade, tarefa, passos, papel do usuário e produto de trabalho. Os elementos do SPEM possuem um ícone que os representam visualmente na modelagem do processo, assim como uma descrição que os caracterizam.

A figura 2.1 apresenta um exemplo de especificação com SPEM. Esta especificação exemplifica a etapa de execução de um ciclo de desenvolvimento de software. Na parte superior da figura 2.1 existe a atividade *Executar demanda* que possui três tarefas. A tarefa *Desenvolver demanda* é realizada pelo papel de usuário *Desenvolvedor*. A tarefa *Testar demanda* é realizada pelo papel de usuário *Testador*. Estas duas tarefas entregam o *Produto da demanda*, este corresponde a um produto de trabalho. Na figura 2.1 se apresenta a tarefa *Validar demanda* que utiliza o produto de trabalho *Produto da demanda*. Esta última tarefa cujo responsável é o papel do usuário *Cliente*, tem como produto de trabalho de saída a *Demanda aceita*. E por fim, ainda na figura 2.1 consta uma raia, divisão que agrupa elementos com finalidades em comum, representando o fluxo interno da tarefa *Testar demanda*. Esta tarefa é composta pelos passos *Verificar demanda* e *Registrar bug da demanda*.

Tabela 2.1 Elementos do SPEM.

Ícone	Elemento	Descrição
	atividade	define a unidade básica de trabalho de um processo
	tarefa	define ações executáveis para os papéis de usuários
	passos	descrevem subunidades de uma tarefa
	papel de usuário	representa um realizador ou participante de uma atividade
	produto de trabalho	são os resultados das tarefas dos processo e são classificados em: *outcomes* (resultados não tangíveis, ex.: entendimento de uma tarefa), artefatos (produto de trabalho tangível que pode ser composto de outros artefatos) e entregáveis (empacotam outros produtos de trabalhos).

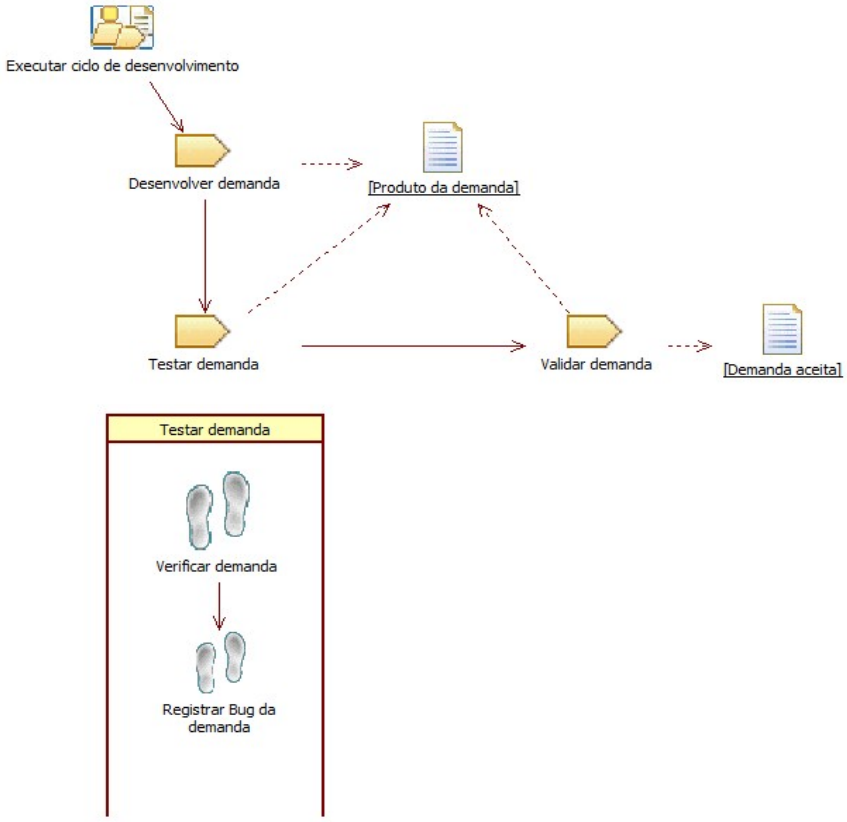


Figura 2.1 Exemplo de Modelo de Processo especificado com SPEM

2.3 MODELOS DE AVALIAÇÃO DE PROCESSOS DE SOFTWARE

Processos de software precisam ser avaliados para que continuem mantendo a qualidade esperada pelos seus utilizadores e apoiando a entrega dos produtos de software. Para que um processo seja avaliado é necessário definir quais qualidades ele deve possuir e como elas serão avaliadas. Qualidade pode ser representada como um atributo, grau de perfeição ou conformidade a certo padrão. Uma vez que uma determinada qualidade seja incorporada ao processo é necessário mantê-la durante as evoluções que o processo sofre ao longo do tempo acompanhando as mudanças organizacionais. Diferentes qualidades que precisem ser mantidas no processo podem ter diferentes formas de avaliações. De modo a avaliar determinadas qualidades de processo de software alguns modelos de avaliação vem sendo estabelecidos na Engenharia de Software

A qualidade de um processo pode ser entendida como o conjunto total de características e propriedades que impactam na capacidade de satisfazer necessidades implícitas e explícitas de um processo (ISO, 2000). Qualidade lida com a conformidade quanto aos requisitos e usa medição para garantir que os requisitos continuem sendo atingidos. A qualidade do processo de software deve ser considerada de forma multidimensional, pois, pode ser observada por diferentes perspectivas como por exemplo, da conformidade com os requisitos, da eficiência do processo, da qualidade estática, entre outras (TYRRELL, 2000). Além disso, modelos conceituais, assim como processos de software podem ser compreendidos como produtos intermediários do ciclo de desenvolvimento de software. Desta forma se um modelo conceitual tem baixa qualidade impactará negativamente a qualidade externa do software (ISO, 2000) (MOODY, 2005). Da mesma forma o processo de software pode impactar o produto final software.

Assim como a qualidade de um processo de software pode ser definida por diferentes atributos, as formas de avaliação do processo de software podem ser diversas. As avaliações de processo de software são organizadas por modelos de avaliação. Estes modelos de avaliação estabelecem uma perspectiva sobre como será avaliada a qualidade do processo de software e também o método como deve ser feita a avaliação. KROEGER (2011) e KROEGER, et. al. (2014).

Os modelos de avaliação da qualidade de processo de software podem ser separados em grupos a partir das perspectivas sobre como será avaliada a qualidade do processo de software. Entre os modelos mais difundidos podemos citar a avaliação:

1. do produto frente ao processo – utiliza critérios com base no produto resultante para avaliar a qualidade do processo;

2. por modelos de avaliação – estabelecem modelos de avaliação com os devidos conjuntos de critérios a serem seguidos. A avaliação acontece por base de consenso entre um conjunto de avaliadores;

3. através de especificações formais – utilizam de especificações formais para avaliar a qualidade do processo.

4. baseada na visão sistêmica da engenharia de software centrada no ser humano (este grupo será citado como avaliação centrada no ser humano) – onde os utilizadores

do processo determinam qualidades do processo e como avaliá-las.

Na (i) avaliação do produto frente ao processo, a perspectiva é que o produto é a base para avaliar a qualidade do processo. Assim o método de avaliação consiste em utilizar critérios referentes ao produto resultante para medir a qualidade do processo. Já na (ii) avaliações por modelos de avaliação, a perspectiva estabelece um conjunto de melhores práticas normalmente estruturadas por áreas, níveis ou estágios as quais os processos especificados devem ser aderentes. Neste grupo de avaliação de processo, o método de avaliação utiliza o consenso entre os avaliadores capacitados por treinamentos ou certificações. Os avaliadores verificam se um determinado escopo de projeto e os produtos do processo estão em conformidade com o conjunto de melhores práticas estabelecido e, em consenso, determinam a conformidade que a empresa alcançou. Como resultado da avaliação, comumente é determinada a maturidade da empresa frente ao modelo avaliado como por exemplo *Capability Maturity Model Integration* (CMMI) e Melhoria do Processo de Software Brasileiro (MPS.BR) (WEBER et al., 2006).

A (iii) avaliação através de especificações formais tem a perspectiva de validar se a especificação do processo está modelado atendendo aos critérios que são o próprio formalismo da especificação correspondente. Quanto ao métodos de avaliação duas formas são possíveis para validar a especificação: a própria validação formal quanto o método de avaliação por simulação do processo. E por fim (iv) a avaliação centrada no ser humano utiliza a perspectiva de avaliação do processo em termos de fatores de qualidades esperados conforme a perspectiva dos participantes do processo (ex.: efetividade, compreensibilidade, previsibilidade). Para cada fator deve ser estabelecida entre os participantes do processo e um profissional mais experiente as forma de avaliação desses fatores. Na avaliação é possível aplicar dois métodos diferentes tanto a coleta de dados após a execução do processo como a simulação da execução do processo.

A (iV) avaliação centrada no ser humano atende a fatores de qualidades esperados conforme a perspectiva dos participantes do processo. Já os demais grupos de avaliação se valem de critérios de avaliação previamente definidos como critérios referentes ao produto, conjunto de melhores práticas ou o formalismo da linguagem especificação de processo utilizada. Desta forma a avaliação centrada no ser humano torna o processo de software mais adaptado as necessidades cotidianas do participantes do processo.Apesar de ser menos difundida essa forma de avaliação será abordada com mais detalhes por ser mais aderente a proposta da pesquisa dessa dissertação.

A avaliação centrada no ser humano fundamenta a sua avaliação da qualidade do processo de software no conceito de múltiplas perspectivas. Essas múltiplas perspectivas são inseridas de acordo com os papéis envolvidos no processo de software e podem ser priorizadas ou selecionadas conforme o atributo de qualidade que se quer avaliar no processo. Esta avaliação tem como base dois conjuntos de elementos principais que são observados: os atributos e as propriedades. Os atributos correspondem às características que são avaliadas de forma subjetiva conforme a percepção do participante. Por exemplo a compreensibilidade para ser julgada dependerá de fatores pessoais como: experiência profissional, conhecimento prévio sobre o contexto do processo, entre outros. Tais fatores podem ocasionar diferentes percepção quanto a compreensibilidade do processo.

Enquanto as propriedades de qualidade são características que podem ser medidas de forma objetiva e recebem concordância dos participantes do processo quanto aos valores e forma de serem mensuradas. Um exemplo seria a medida de tamanho de tarefas de uma atividade, onde o valor resultante será comum a qualquer pessoa que busque essa medida.

Dada a ausência de uma taxonomia para unificar os atributos e propriedades de qualidade, Kroeger (2011) realizou um estudo com esta finalidade. Tal estudo aplicou um *survey* e algumas avaliações de processo baseado na avaliação centrada no ser humano. No *survey* foram coletados de profissionais de diferentes papéis do processo de software os fatores (atributos e propriedades) de qualidade que acreditavam serem necessários em um processo de software. Já as avaliações de processo puderam validar e inferir características e relações entre os fatores que qualidade levantados pelo *survey*. Como resultado de sua pesquisa foram obtidos os seguintes agrupamentos de atributos de qualidade, conforme figura 2.2. Da mesma forma foram identificados os seguintes agrupamentos de propriedades de qualidade, conforme figura 2.3 (KROEGER, 2011).

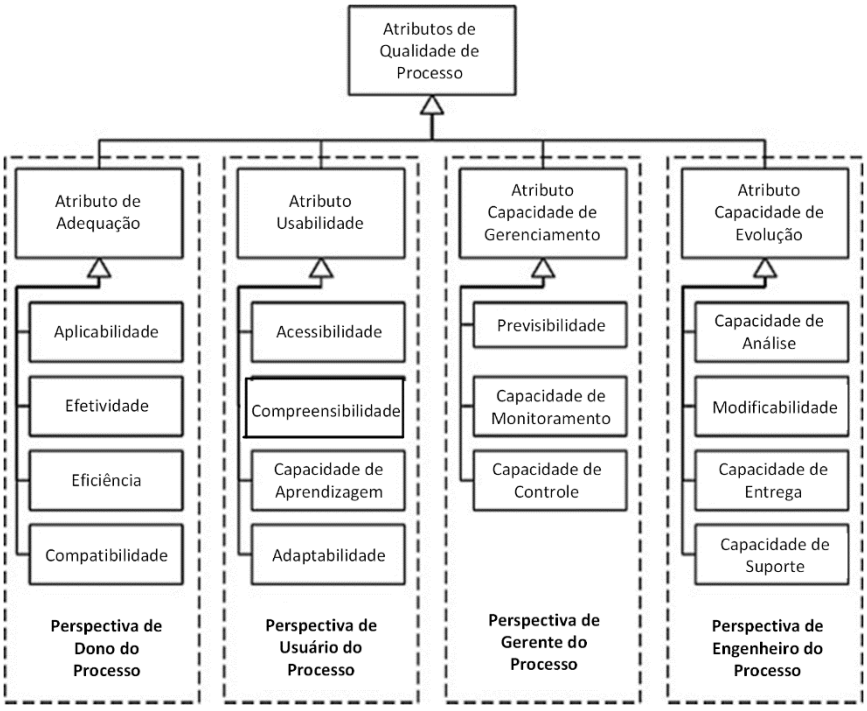


Figura 2.2 Lista de atributos de qualidade. Fonte: (KROEGER; DAVIDSON; COOK, 2014)

Os atributos de qualidade (figura 2.2) estabelecidos pelo estudo de Kroeger (2011) foram adequação, usabilidade, capacidade de gerenciamento e capacidade de evolução. Estes atributos foram separados conforme as respectivas perspectivas dos donos do processo (*process owners*), participantes do processo, gerentes do processo e engenheiros do processo. Os atributos de qualidade ainda foram subdivididos em subatributos. Os atributos podem ser compreendidos pelas seguintes características:

1. qualidade: Adequação é um atributo que contempla a capacidade de processo de

atender aos resultados esperados se mantendo dentro das restrições como custo e cronograma. Este atributo está dividido nos seguintes subatributos: aplicabilidade, eficácia, eficiência e compatibilidade.

2. usabilidade reflete a facilidade com que o processo é entendido e executado pelos seus participantes. Este atributo está dividido nos seguintes sub atributos: acessibilidade, compreensibilidade, capacidade de aprender e adaptabilidade.

3. capacidade de gerenciamento reflete a facilidade com que a execução (*enactment*) do processo pode ser prevista, monitorada e controlada atingindo os resultados esperados. Este atributo está dividido nos seguintes sub atributos: previsibilidade, monitorabilidade e controlabilidade.

4. o e capacidade de evolução reflete a capacidade do processo em ser modificado ou melhorado. Este atributo está dividido nos seguintes subatributos: capacidade de análise, modificabilidade, implantação e capacidade de suporte.

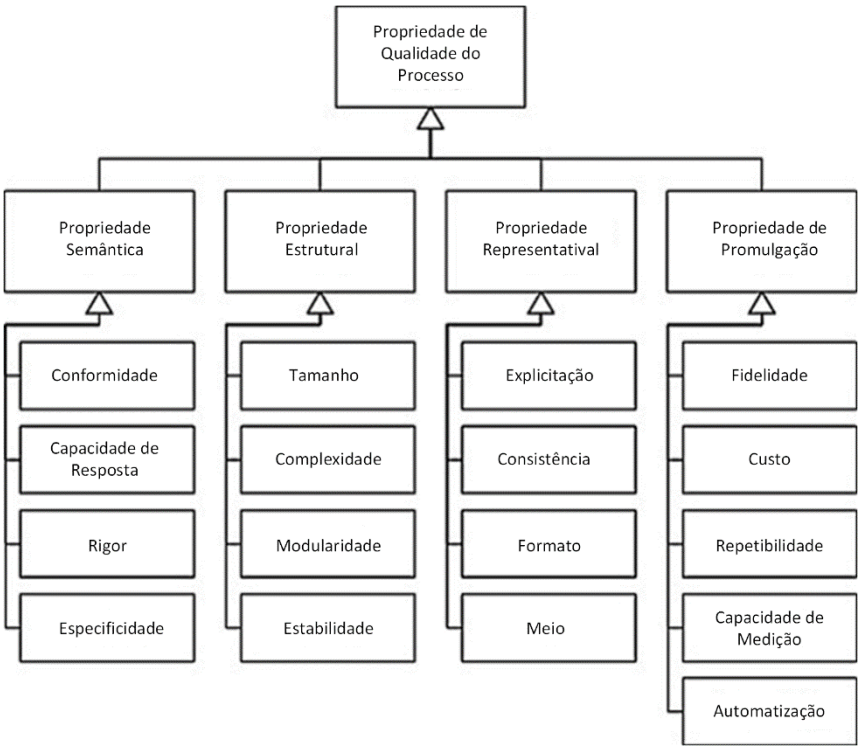


Figura 2.3 Lista de propriedades de qualidade. Fonte: (KROEGER; DAVIDSON; COOK, 2014).

Já as propriedades de qualidade (figura 2.3) estabelecidas pelo estudo de Kroeger (2011) foram as propriedades semânticas, estruturais, representativas e promulgação (*enactment*). As propriedades não precisaram ser agrupadas por perspectiva dos participantes do processo uma vez que suas medidas são objetivas. As propriedades foram subdivididas em sub propriedades.

Tabela 2.2 Matriz de relação entre os atributos e as propriedades. Fonte: (KROEGER; DAVIDSON; COOK, 2014).

Propriedade de Qualidade do Processo	Atributos de Qualidade de Processo	Adequação				Usabilidade				Gerenciamento			Evolução			
		Aplicabilidade	Efetividade	Eficiência	Compatibilidade	Acessibilidade	Compreensibilidade	Capacidade de Aprendizagem	Adaptabilidade	Previsibilidade	Capacidade de Monitoramento	Capacidade de Controle	Capacidade de Análise	Modificabilidade	Capacidade de Entrega	Capacidade de Suporte
Semântica	Conformidade		↑													
	Capacidade de Resposta										↑					
	Rigor		↑									↑	↑			
	Especificidade						↑		↓							
Estrutural	Tamanho			↓		↓	↓								↓	
	Complexidade			↓			↓	↓	↓			↓			↓	
	Modularidade				↑				↑							
	Estabilidade											↑				
Representacional	Explicitação						↕	↑	↓			↑	↑			
	Consistência						↑									
	Formato						↑									
	Meio		↑	↑		↑									↑	
Promulgação	Automatização			↑					↓						↑	
	Custo			↓												
	Fidelidade														↑	
	Capacidade de Medição									↑	↑	↑	↑			
	Repetibilidade		↑						↓	↑		↑	↑			

As propriedades de qualidade inerentes ao processo influenciam a percepção dos participantes sobre a qualidade dos atributos do processo. Estas influências podem ser consideradas positivas, negativas e compostas. Cada influência é definida pelas seguintes características:

1. positiva - quando o valor medido da propriedade de qualidade aumenta e o valor percebido para o atributo de qualidade também aumenta;

2. negativa - quando o valor medido da propriedade de qualidade aumenta e o valor percebido para o atributo de qualidade também diminui;

3. composta - quando para o valor medido da propriedade de qualidade até determinado limiar pode influenciar positivamente e o valor percebido para o atributo de qualidade também aumenta. E, para valores fora desse limiar, podem influenciar negativamente a percepção do atributo de qualidade.

A matriz de relação apresentada na tabela 2.2 demostra as influências das propriedades aos atributos de qualidade (KROEGER, 2011). As setas indicam as influências detectadas, onde a seta para cima representa influência positiva, como no caso da subpropriedade conformidade em relação ao subatributo efetividade. As setas para baixo indicam influência negativa como para a relação entre especificidade e adaptabilidade . Por fim a relação composta com a seta apontando tanto para cima como para baixo, como é o caso da explicitação e compreensibilidade.

Os modelos de avaliação da qualidade de processo de software possuem métodos de avaliação que envolvem diferentes formas de medição, como no exemplo de obter a medida de tamanho de tarefas de uma atividade. As medições costumam ser realizadas por métricas que de, forma repetível e monotônica, quantificam numericamente a grandeza de uma determinada propriedade. Métricas são repetíveis, pois os mesmos valores serão obtidos por pessoas diferentes. Métricas são monotônicas, pois qualquer alteração da propriedade medida influenciará aumentando ou diminuindo a métrica obtida (VANSUETENDAEL; ELWELL, 1991).

2.4 BAD SMELLS

Bad smells são estruturas de *design* de código que podem reduzir a qualidade destes e indicar alguns problemas, e consequentemente candidatas a refatoração (FOWLER; KENT, 1999). Inicialmente, foram propostos vinte e dois *bad smells* de código com suas características básicas, os contextos onde eles podem ser encontrados, as implicações negativas que os *bad smells* causam no *design* do software e a indicação das refatorações adequadas para reduzir o problema. As indicações de como identificar os *bad smells* são descritas a partir de algumas características destes que ferem princípios de programação. Essas indicações de identificação dos *bad smells* tiveram como validação a utilização da experiência profissional e verificações de acerto, com o propósito de indicar para experiência futura uma melhor capacidade de decisão sobre *design* do código.

Alguns trabalhos (MARTICORENA; LÓPEZ; CRESPO, 2006), (MANTYLA; VANNHANEN; LASSENIUS, 2003) apresentaram uma taxonomia (tabela 2.3) para fornecer

compreensão sobre os impactos dos *bad smells* nos *design* de software e constituem as seguintes categorias: *The Bloaters*, *The Object-Orientation Abusers*, *The Change Preventers*, *The Dispensables*, *The Couplers, Encapsulators* .

A seguir são apresentadas as definições das taxonomias e exemplos dos *bad smells* associados (MARTICORENA; LÓPEZ; CRESPO, 2006), (MANTYLA; VANHANEN; LASSENIUS, 2003), (FOWLER; KENT, 1999):

Tabela 2.3 Taxonomia dos *Bad Smells*.

Taxonomia	*Bad Smells*
The Bloaters	*Data Clumps*, *Large Class*, *Long Method*, *Long Parameter List*, *Primitive Obsession*
The Object-Orientation Abusers	*Alternative Classes with Different Interfaces*, *Refused Bequest*, *Switch Statements*, *Temporary Field*
The Change Preventers	*Divergent Change*, *Parallel Inheritance Hierarchies*, *Shotgun Surgery*
The Dispensables	*Data Class*, *Dead Code*, *Duplicate Code*, *Lazy Class*, *Speculative Generality*
The Couplers (Encapsulators)	*Feature Envy*, *Inappropriate Intimacy*, *Message Chains*, *Middle Man*
Not Defined	*Incomplete Library Class*

1. **The Bloaters** definem estruturas do código que tendem ao crescimento gradual se tornando mais complexos e difíceis de entender. Conforme a tabela, um dos *bad smells* desta categoria é o *Long Parameter List*, um método que deve ter uma média de parâmetros definida para manter a compreensão sobre o objetivo do mesmo e não aumentar a sua complexidade. Quanto maior a quantidade de parâmetros, o método se torna mais complexo de entender;

2. **The Object-Orientation Abusers** compreende situações em que as possibilidades oferecidas pelo paradigma de orientação a objeto não são utilizadas de forma adequada. O *Temporary Field* é um exemplo desta categoria. Este *bad smell* acontece quando uma variável mantida no escopo de uma classe deveria estar no escopo de um método, pois somente este método faz uso da variável;

3. **The Change Preventers** viola o princípio que uma classe e suas mudanças deve ter um relacionamento um-para-um. Um *bad smell* desta categoria é o *Divergent Change* indicando que uma classe sofre mudanças originadas por diferentes propósitos;

4. **The Dispensable** representa algo que pode ser removido do código sem causar impactos. Um exemplo desta categoria é o *bad smell Dead Code*, que representa um código que nunca é utilizado no sistema em que foi implementado;

5. **The Couplers** afetam o princípio de *design* que indica que o baixo acoplamento deve ser buscado no *design* do código. Um *bad smells* desta categoria é o *Featury Envy* que identifica um método que requisita parte ou todos dos dados necessários de outra classe e não da classe na qual ele foi implementado;

6. **Encapsulators** consiste na violação do princípio de encapsulamento do paradigma orientação a objeto, que busca tornar os elementos isolados e independentes. Os *bad smells* desta taxonomia também podem ser atribuídos às taxonomias *The Couplers* ou *The Object-Orientation Abusers*. Um exemplo de *bad smell* desta taxonomia é o *Middle Man*, que recebe esse título por fazer, de forma demasiada, delegações para outras classes e assim não processando as funcionalidades do sistema, tornando-se muito dependente de outras classes.

2.4.1 Estratégia de detecção de bad smells

Estratégias de detecção de *bad smells* estabelecem mecanismos para localização de problemas de *design* do código orientado a objeto. Uma estratégia é definida a partir de heurísticas (regras informais de *design*) que podem ser quantificadas de forma a identificar desvios específicos no *design* do código e destas heurísticas são extraídos sintomas. Os sintomas são características expressas de modo a serem verificadas por métricas que avaliam características específicas do código. Por exemplo, a complexidade de uma classe, o acoplamento entre métodos e outros. Desta forma usando uma estratégia de detecção de *bad smells* é possível identificar especificamente características de um elemento do código, como uma classe ou método compreendendo assim se estes afetam negativamente a qualidade do código orientado a objeto (MARINESCU, 2004).

O mecanismo de detecção de *bad smells* faz uma análise genérica no código de um software buscando os elementos de *design* em conformidade com as regras preestabelecidas que apontam os problemas. Esta localização e feita de "cima para baixo" (*top – down*) uma vez que o problema a ser identificado já foi estabelecido pelas suas regras e métricas. Estas regras passam por mecanismos de filtragem (definição de limites que serão verificados pelas métricas) e composição (relação que as regras têm entre si para detectar um *bad smell*) definindo a detecção dos *bad smells* (MARINESCU, 2004). Essa estratégia segue um processo para transformação das regras informais de *design* em estratégias de detecção, conforme os passos a seguir:

1. selecionar heurísticas que são comportamentos conhecidos na literatura que possam identificar *bad smells*;

2. identificar, a partir das heurísticas, os sintomas quantificáveis por métricas;

3. selecionar as métricas adequadas que medirão os sintomas;

4. definir a filtragem, apontar como mensurar os valores limiares para as métricas identificarem problemas no *design* do código;

5. fazer a composição, definir as relações lógicas entre as diferentes sintomas coexistentes que apoiam a detecção de um *bad smell*. As relações lógicas são definidas pelos operadores lógicos "E" e "OU".

Lanza e Marinescu (2007) apresentam alguns *bad smells* e suas estratégias de detecção (LANZA; MARINESCU, 2007). Entre os exemplos, o *Brain Method* é um *bad smell* que define métodos longos, de difícil compreensão e reuso. Este *bad smell* se caracteriza por algumas heurísticas como o fato de que o *Brain Method* tende a ser um método longo por executar mais de uma funcionalidade reduzindo assim a compreensibilidade e testabilidade do código, utilizar muitas ramificações (se e senão) representando um design divergente ao orientado a objeto e fazer uso de muitas variáveis locais e globais.

Os sintomas obtidos a partir das heurísticas do *Brain Method* se constituem ao fato de ser um método longo se o número de linhas de código exceder ao valor limiar estabelecido, assim a métrica que avalia a esse sintoma é a *LOC (line of code)* - número de linhas de código. Em relação a utilizarem muitas ramificações foram definidos dois sintomas. O primeiro é a compreensão de que muitas ramificações torna o código complexo e para verificar esse sintoma é aplicada a métrica *CYCLO (McCabe's Cyclomatic Complexity)* - complexidade ciclomática de McCabe. Já o segundo sintoma é a verificação se nível máximo de aninhamentos que o método possui, como *loops* ou fluxos de controle, exceder ao valor limiar estabelecido e é verificado pela métrica *MAXNESTING (Maximum Nesting Level)* - nível máximo de aninhamento. O último sintoma é definido pela heurística do uso de muitas variáveis. Por fim, usar muitas variáveis dificulta a memorização humana e facilita a introdução de erros no código, este sintoma é verificado pela métrica *NOAV (Number Of Accessed Variables)* - número de variáveis acessadas.

Quanto a filtragem, as métricas e limiares são definidas para que se verifiquem:

1. se o número de linhas de código excede ao valor limiar pela relação LOC > Alto,
2. se o método:
   2.1. possui muitas ramificações pela relação $CYCLO \geq$ Alto,
   2.2. e se o nível máximo de aninhamentos excede o limiar por $MAXNESTING \geq$ Vários
3. e se o número de linhas excede o limiar por NOAV > Muitos.

Os limiares Alto, Vários e Muitos são substituídos por valores específicos quando a detecção é aplicada. A composição dos sintomas para este *bad smell* indica que somente se todas os valores das métricas de um método avaliado excederem os valores limiares estabelecidos simultaneamente esse método será definido como um *Brain Method*. Assim a composição do *Brain Method* utiliza o operador lógico "E"para a relação de todos os sintomas.

Capítulo

3

TRABALHOS RELACIONADOS

A pesquisa desta dissertação apresenta um catálogo para apoiar a identificação de anomalias em processo de software, os *process smells*, utilizando como referência os estudo de Fowler (1999) e Marinescu (2004). Propostas de adaptações e detecção de *bad smells* para diferentes domínios tem sido realizadas conforme (MACÍA; SANT'ANNA; STAA, 2008) (DEURSEN et al., 2001) (ROMPAEY et al., 2007) (GREILER; DEURSEN; STOREY, 2013) (CORRADINI et al., 2018) (GARCÍA et al., 2003) (KHLIF et al., 2009), porém nenhuma delas tem como objetivo processos de software. Em comum estas propostas tem estratégias de adaptação e detecção similares quanto a forma que estão realizadas, sendo assim uma forma similar em relação as estratégia dos demais trabalhos foi adotada na pesquisa dessa dissertação.

As estratégias de adaptação e detecção das propostas seguem alguns passos em comum. É estabelecida a forma com que ocorrerá a adaptação dos *bad smells* para *smells* conforme o contexto aplicado. No geral essa adaptação se utiliza da especificação dos *bad smells* em comparação a especificação do domínio ao qual o *smells* será adaptado. Em seguida se define as estratégias de detecção, métricas e valores limiares. Adicionalmente, faz-se necessário definir a validação desta adaptação dos *smells* para um novo contexto.

Macía et. al. (2008) propôs um estudo para a verificação dos *bad smells God Class* e *Data Class* em diagramas de classes definidos com *Unified Modeling Language* (UML). O trabalho de Deursen et. al. (2001) propõe *bad smells* para o contexto de testes unitários definidos como *test code smells*. Rompaey et. al. (2007) estabeleceu estratégias de detecção para os *test code smells*.

Entre os trabalhos relacionados dois abordam *bad smells* em *Business Process Modeling Notation* (BPMN) (SILINGAS; MILEVICIENE, 2007) (CORRADINI et al., 2018). O primeiro adapta e o segundo detecta *bad smells* em BPMN. Em Silingas (2007) foi apresentado um conjunto de *bad smells* para notações de modelos de processo de negócio BPMN.No trabalho de Corradini (2018) é proposto um *framework* para compreensibilidade de modelos BPMN.

Em alguns trabalhos relacionados são propostas adaptações de métricas de orientação a objeto para BPMN (VANDERFEESTEN et al., 2007) (KHLIF et al., 2009). García

et al., (2003) e García e Piattini (2004) apresentam experimentos para validação de um conjunto de métricas sobre elementos estruturais proposta para *Software & Systems Process Engineering Meta-Model* (SPEM).

Os trabalhos Macía et. al, Deursen et. al. (2001) e Rompaey et. al. (2007) evidenciam que *bad smells* vem sendo atribuídos a diferentes contextos. O primeiro no contexto de diagramas de classes e o segundo em testes unitários. Diferentes destes dois trabalhos citados, os trabalho a seguir possuem maior aproximação ao tema da dissertação uma vez que estes elaboraram estratégias de adaptação de *bad smells* para BPMN por ser uma notação de processo assim como o SPEM.

Em Silingas (2007) apesar da referência aos *bad smells* de Fowler os *bad smells* apresentados para o BPMN são bem distintos. Uma exceção é o *Large Process Diagrams* que apresenta a preocupação em relação ao tamanho do diagrama do processo e o impacto negativo a compreensibilidade do processo. Quanto ao demais *bad smells* para BPMN eles retrataram a nomeação inadequada dos elementos do modelo de processo de negócio, usos inadequados de elementos do BPMN como fluxos de decisão, eventos e *loops* além de outras falhas de *layout* (*layouts* pobres) do modelo de processo de negócio.

No *framework* apresentado por Corradini (2018) são incorporadas as referências dos *bad smells* de Silingas (2007), assim como referências de outros autores e ainda é apresentada uma ferramenta que realiza a detecção automática das diretrizes do *framework*. As estratégias de detecção se estabelecem pela verificação feita, por algoritmos, para cada diretriz do *framework*. Estes algoritmos fazem a leitura do arquivo convertido em *eXtensible Markup Language* (XML) a partir de arquivos com extensão BPMN e se utilizam dos dados do XML para realizar medições que alcancem os elementos que divergem das suas diretrizes. Contudo o trabalho não apresenta a composição das estratégias de detecção de forma detalhada apenas referencia cada diretriz.

Os trabalhos relacionados anteriormente retratam o uso de métricas para detecção dos *bad smells*. Vanderfeesten et al. (2007) e Khlif et. al. (2009) utilizam como base para realizar a adaptação das métricas uma tabela comparativa entre os elementos de código de software e do BPMN. A adaptação busca poder medir a qualidade do *design* relacionada a cinco princípios. Ambos verificam acoplamento, coesão, complexidade, enquanto apenas o primeiro se estende verificando também modularidade e tamanho. Vanderfeesten et al. (2007) estabelece a comparação entre elementos de código e elementos do BPMN, o que orientou a comparação entre elementos do código e do SPEM. Além disso, assim como o segundo artigo também estabeleceu formas de adaptações de métricas de código de software para BPMN. As formas de adaptação apresentadas serviram como base para adaptação dos *bad smells* e métricas do paradigma orientado a objeto para *process smells* e para a medição de elementos estruturais do SPEM.

Os estudos García et al., (2003) e García e Piattini (2004) tinham objetivo de verificar a relação das métricas com características de manutenção e compreensão do modelo. O conjunto de métricas proposto será utilizado para composição das heurísticas de detecção de *bad smells* em SPEM seguindo as características indicadas por Marinescu.

O BPMN é uma linguagem de modelagem de processo de negócio reconhecida, que possui similaridade com o código de software. Estes trabalhos reforçam a proposta de estrutura metodológica da pesquisa, uma vez que apresentam os passos utilizados para

a adaptação e detecção dos *bad smells*. Além disso apresentam a proposta com BPMN. Está notação tem base em comum com o SPEM. Ambas fornecidas pelo *Object Management Group* (OMG) com a *MetaObject Facility Specification* (MOF) como núcleo em comum.

Por fim, outro trabalho relacionado (GARCÍA et al., 2003) apresenta um experimento para validação de um conjunto de métricas para elementos estruturais do SPEM. O objetivo foi verificar a relação do conjunto de métricas para avaliar a capacidade de compreensão e manutenção de modelos de processo de software. Quanto ao conjunto de métricas proposto foram selecionadas ou adaptadas as mais adequadas para composição das heurísticas de detecção de *process smells* em SPEM seguindo a estratégia proposta por Marinescu.

Diante aos trabalhos relacionados apresentados dois trabalhos abordam *bad smells* em BPMN porém a pesquisa desta dissertação se diferencia pelos seguintes aspectos. O primeiro, Silingas (2007) adapta *bad smells* a partir de características que contrariam as boas práticas de modelagem com BPMN. Os *process smells* tem como base os *bad smells* de orientação a objeto e verificam os impactos negativos a atributos de qualidade definidos evidenciados pelos usuários de processo conforme kroeger (2011). O segundo, Corradini(2018) detecta problemas no design de modelos BPMN seguindo referências de um conjunto de guias de qualidade de modelagem e métricas de verificação dos problemas. Este trabalho além de utilizar BPMN e não o SPEM, assume algumas verificação de problemas como o uso não estruturado do BPMN o que difere do catálogo de *process smells* que só avalia a qualidade estrutural da modelagem do processo de software.

Em relação aos elementos da estratégia de detecção a pesquisa desta dissertação também apresenta algumas diferenças.O trabalho de Garcia (2003) verifica o resultado de métricas aplicadas sobre todo o processo. Este trabalho utiliza a abordagem de baixo para cima *bottom - up* desta forma não estabelecendo previamente quais impactos negativos da qualidade do processo serão verificados. Já os *process smells* utilizam uma abordagem de cima para baixo *top - down* onde são previamente definidos atributos de qualidade a serem verificados e os possíveis impactos negativos aos quais eles podem ser submetidos. Vanderfeesten (2007) e Khlif (2009) abordam como adaptar métricas do paradigma de código orientado a objeto a parte de qualidade internas como coesão, acoplamento o que converge com a esta pesquisa, se diferenciando apenas pelo fato de utilizarem as adaptações das métricas para o BPMN enquanto a pesquisa o faz para SPEM.

A proposta apresentada por esta dissertação leva em consideração as estruturas metodológicas adotada para adaptar *bad smells*, propostas pelos trabalhos relacionados. Assim, a adaptação de *bad smells* de código orientado a objeto para *process smells* especificados em notação SPEM tem uma metodologia similar a dos trabalhos apresentados quanto a adaptação e detecção de *smells*. Na metodologia foram estabelecidas as adaptações de *bad smells* para *process smells*, as estratégias de detecção, métricas e valores limiares e as validações.

Nestes trabalhos relacionados buscam minimizar problemas em comum, como a caracterização e detecção de anomalias. A caracterização de anomalias é uma busca em comum diante das especificações de software apresentadas. Os trabalhos apresentam procuraram anomalias, seja para diagramas de classes, classes de teste, especificações BPMN

ou SPEM. Anomalias nestas diferentes especificações podem resultar em impactos negativos na qualidade e podem também impactar negativamente o produtos finais das especificações.

Detecção de anomalias também é uma necessidade percebidas conforme os trabalhos relacionados. A detecção consistem em como as anomalias caracterizadas previamente ou não podem ser identificadas na especificação. Elementos que fazem parte da detecção são as estratégias, as métricas e valores limiares. A padronização da forma de detecção favorece que diferentes modelagem de uma mesma especificação possam ter anomalias identificadas usando o mesmo padrão de detecção.

A pesquisa desta dissertação também expande o campo de detecção de *bad smells* para processos de desenvolvimento de software. Conforme os trabalhos relacionados outras especificações já possuem estudos para caracterização e detecção de *bad smells*. Em específico para a notação SPEM os estudos se concentraram em estabelecer métricas e realizar avaliações com a abordagem de baixo para cima (botton - up) e nesta pesquisa definir previamente os *bad smells* pode ser uma contribuição ao campo de pesquisa.

Capítulo

4

CATÁLOGO DE PROCESS SMELLS

Process smells são resultados das decisões tomadas durante as fases como concepção, modelagem e a produção do modelo do processo. Estes *smells* incidem no arranjo dos elementos do processo de modo a reduzir a qualidade de fatores esperados para um processo.

Este capítulo apresenta a definição e todas as atividades relativas a elaboração do catálogo de *process smells* e o catálogo resultante. O catálogo de *process smells* foi especificado conforme as fases que podem ser vistas na figura 4.1. Inicialmente foi feita a especificação dos *process smells*, esta primeira fase deu origem à versão inicial do catálogo. Posteriormente, na segunda fase foi feita a especificação das estratégias de detecção dos *process smells*, sendo que esta fase resultou na versão final do catálogo.

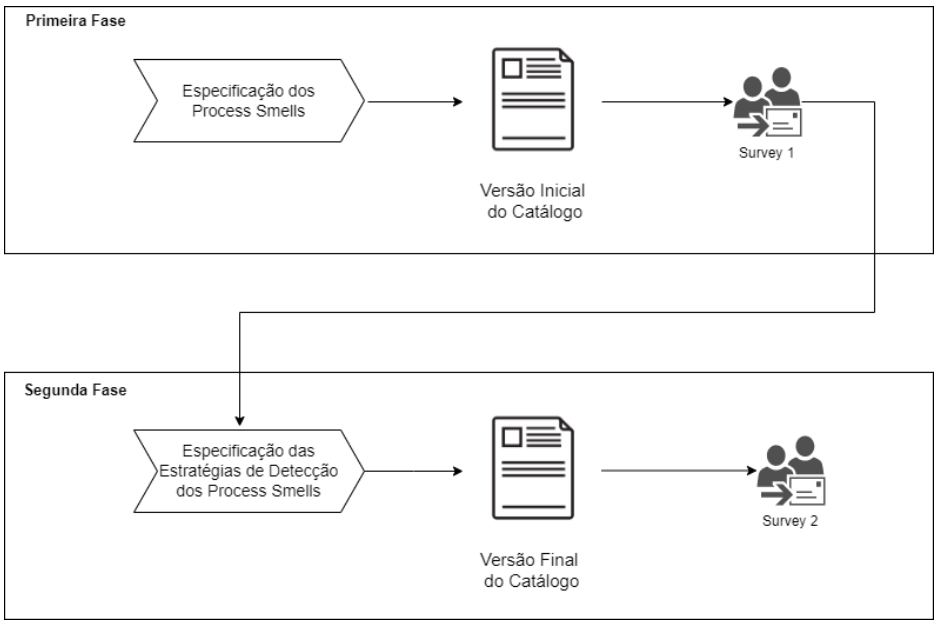


Figura 4.1 Fases da Elaboração do Catálogo de *Process Smells*.

4.1 PRIMEIRA FASE - ESPECIFICAÇÃO DOS PROCESS SMELLS

A seguir serão apresentadas as etapas que compõem a primeira fase desta pesquisa que corresponde ao processo de elaboração do do *process smells*. Conforme a figura 4.2 a primeira fase da pesquisa foi composta por atividades e seus respectivos artefatos de saída que anteciparam a versão inicial do catálogo de *Process Smells*, entre elas a Definição de Similaridades entre Paradigma de Orientação à Objeto e *Software & Systems Process Engineering Meta-Model* (SPEM), Seleção dos *Bad Smells* candidatos e Adaptação para SPEM e Consolidação inicial do Catálogo. A construção do catálogo de *process smell* teve como referências os *bad smells* de código orientado a objeto e foi organizada em quatro grandes etapas, elencadas a seguir:

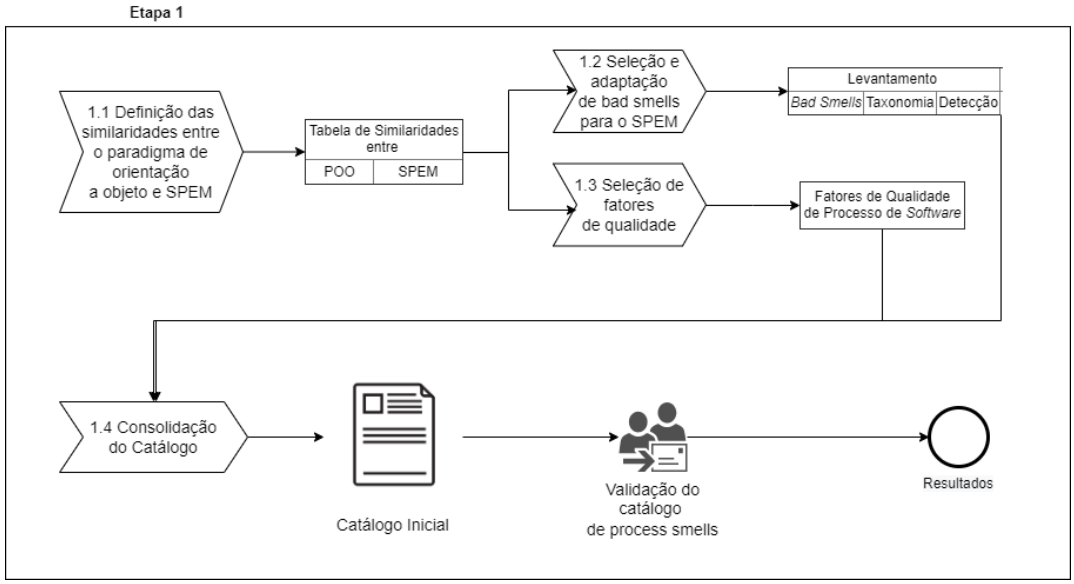


Figura 4.2 Etapas da Primeira Fase de Elaboração do Catálogo de *Process Smells*.

1. Definição de Similaridades entre Paradigma de Orientação à Objeto e SPEM. Esta etapa teve como objetivo estabelecer uma correlação entre os elementos básicos do paradigma de orientação a objeto que são usados para a definição de *bad smells* e os elementos do SPEM. Como resultado desta etapa foi estabelecida a tabela 2.1 apresentando os elementos similares entre o paradigma de orientação a objeto e o SPEM.

2. Seleção dos *Bad Smells* candidatos e Adaptação para SPEM. Esta etapa teve como objetivo selecionar e adaptar os *bad smells* para SPEM, os caracterizando como *process smells*. Para tanto foi realizado um levantamento sobre como os *bad smells* vinham sendo detectados e quais eram as taxonomia foram estabelecidas para eles. Com base no levantamento e nas taxonomias os *bad smells* foram aplicados critérios de exclusão e selecionados dez *bad smells* para a adaptação em *process smells* especificados com SPEM .

3. Seleção dos Fatores de Qualidade. Esta etapa teve como objetivo identificar os fatores de qualidade de processo de *software* aos quais os dez *process smells* poderiam estar relacionados. Com esta etapa foi possível definir, a partir dos fatores de qualidade de processo relacionados com *process smells*, os possíveis impactos negativos que os *process smells* podem causar a processo de *software*.

4. Consolidação Inicial do Catálogo de *Process Smells*. Esta etapa teve como objetivo elaborar de um catálogo contendo os *process smells* especificados com SPEM sendo apresentados com definição, elemento estrutural do SPEM impactado, representação visual dos *process smells* e possíveis impactos que poderiam afetar negativamente os atributos de qualidade do processo de *software*.

4.1.1 Definição de Similaridades entre Paradigma de Orientação à Objeto e SPEM

A seguir será apresentada a primeira etapa da primeira fase da elaboração do catálogos de *process smells* (item 1.1 da figura 4.2). A definição de elementos similares entre paradigma de orientação a objeto e os elementos do SPEM foi obtida a partir de trabalhos relacionados, usando o *Business Process Modeling Notation* (BPMN) como base intermediária para apoiar o raciocínio para a comparação. O BPMN é uma linguagem de modelagem de processo de negócio reconhecida, que possui similaridade com o código de software, conforme tabela 4.1.

Tabela 4.1 Similaridades entre programa de software e notação de modelo de process de negócio BPMN. Fonte: (VANDERFEESTEN et al., 2007).

Programa de software	Processo de negócio
Módulo/Classe	Atividade
Método/Função	Operação

A tabela 4.1 apresenta elementos que possuem semelhanças sendo que nas colunas da esquerda elementos de software, já na coluna a direita elementos do BPMN. É comum que um programa de software seja dividido em módulos ou classes que recebem dados de entrada, realizem um processamento e entreguem dados de saída. De forma semelhante um processo de negócio é subdividido em atividades que desempenham o mesmo papel análogo ao que representam os módulos ou classes de um software. Os módulos ou classes possuem métodos ou funções que ordenam a execução e processamento dos dados de forma a gerar os dados de saída. Para tanto as atividade de um processo de negócio são compostas das operações. Estas operações assim como os métodos ou funções são as etapas menores que compõem uma atividade para gerar os dados de saída.

Assim, a relação proposta na tabela 4.1 leva em consideração que os elementos comparados possuem a mesma finalidade para as respectivas linguagens. Desta forma comparando programas de software orientados a objeto ao SPEM também são obtidas algumas semelhanças. Uma classe é uma unidade principal na estrutura do código, que atende por um objetivo e se comunica com outras classes para trocar informações o tornar o

sistema completo. Similarmente, atividades possuem seus propósitos e juntas constituem os processos.

Os métodos entregam resultados que juntos compõem as classes. Isto funciona da mesma forma para as tarefas. As estruturas condicionais e instruções formam as funções e métodos, assim como os fluxos de decisão e passos formam as tarefas. Os métodos acessores ou propriedades são responsáveis por acessar as variáveis ou atributos dos códigos e para representar esses elementos temos os produtos de trabalho.

Com base no trabalho de Vanderfeesten, Irene et al. (2007) foi realizada a comparação dos elementos de código orientado a objeto e elementos do SPEM, levando em consideração elementos possuem a mesma finalidade para as suas respectivas linguagens. A comparação entre os elementos de código orientado a objeto e do SPEM não avançou a outros níveis, pois, neste ponto, já se apresentou satisfatória para compor os *process smells* (tabela 4.2). A comparação foi satisfeita, pois os *bad smells* selecionados para adaptação que serão apresentados na seção 4.1.2 conseguiram ser representados apenas com estes elementos.

Tabela 4.2 Adaptação da tabela de similaridades entre programa de software e BPMN de Vanderfeesten et al. (2007) para código orientado a objeto e SPEM.

Código orientado a objeto	**SPEM**
Classe	Atividade
Método	Tarefa
Estrutura condicionais	Fluxo de decisão
Instruções	Passos
Acessores/Propriedades	Produtos de Trabalho

4.1.2 Seleção de Bad Smells candidatos e Adaptação para o SPEM

A seguir será apresentada a segunda etapa da primeira fase da elaboração do catálogos de *process smells* (item 1.2 da figura 4.2). O objetivo desta fase foi selecionar entre os vinte e sete *bad smell* aqueles que pudessem ser adaptados para o contexto de processo de software. Estes vinte e sete *bad smell* foram identificados a partir de um levantados feito nesta etapa da pesquisa, onde foram verificados trabalhos que discutiam estratégias de detecção para *bad smell* conforme dispostos no apêndice 1. A seleção foi realizada através a aplicação de três critérios de exclusão:

i. não fossem característicos de elementos específicos de código orientado a objeto como herança e especializações, dificultando assim, a adaptação para SPEM;

ii. pudessem ser detectados apenas pela utilização de métricas, não dependendo de avaliações semânticas, facilitando o uso das estratégias proposta por Marinescu(2004);

iii. e não fossem genéricos ao ponto de estarem contidos em outros *bad smell* mais complexos.

Quanto aos critérios, o primeiro consiste em descartar os *bad smells* relacionados à herança e as especializações, características específicas do paradigma orientado a objeto. Apesar de ser possível representar algumas propriedades do paradigma orientado a objeto com elementos de variabilidade do SPEM, esta estratégia não é tão comum frente às outras formas de especificar processos de software, por esse motivo não será abordada neste trabalho.

O segundo critério consiste em descartar os *bad smells* cuja técnica de detecção necessitasse de avaliações com teor semântico. Avaliações com teor semântico requerem a compreensão e a interpretação do significado ou objetivo do código verificado. Por exemplo, métodos em que são feitas validações de dados podem receber nomenclatura diferentes, mas ainda assim desempenharem a mesma validação ou podem se repetir sendo invertidas as ordem das operações de validação e apesar da inversão promoverem as mesmas validação de dados. Assim com a validação semântica é possível verificar se dois códigos apesar de serem implementados de formas diferentes buscam resolver um mesmo problema. Este tipo de avaliação foge ao escopo da pesquisa, pois tem como objetivo fazer a detecção de *process smells* a partir da medição de elementos estruturais do processo especificado com SPEM.

O terceiro critério consiste em descartar *bad smells* que são genéricos demais e assim assim já possuíam suas características principais incorporadas em outros *bad smells*. No contexto de processos de software dessa pesquisa essa tipos de *bad smells* já terão suas características como parte de *process smells* mais complexos.

A partir do levantamento foram identificados vinte e sete *bad smells* candidatos e separados conforme suas taxonomias (tabela 4.3). Entretanto seis destes *bad smells* identificados propostos por Lanza e Marinescu (2007) não possuíam taxonomias estabelecidas, mas ainda assim foram adicionados na pesquisa. Estes foram: *Incomplete Library Class* (FOWLER; KENT, 1999), *Brain Class, Brain Method, Dispersed Coupling, Intensive Coupling, Significant Duplication* (LANZA; MARINESCU, 2007). Estes *bad smells* serão agrupados como "*Not Defined*".

Finalizado a atividade de seleção os candidatos, o primeiro critério de exclusão descartava *bad smells* característicos de propriedades do paradigma orientado a objeto. Assim, verificando os vinte e sete *bad smells* foram descartados todos os *bad smells* da taxonomia The Object-Oriented Abusers. Esta taxonomia refletia propriedades mais pertinentes ao paradigma como heranças e especializações e envolve quatro *bad smells*: *Switch Statements, Temporary Field, Refused Bequest, Alternative Classes with Different Interfaces.* Desta forma sobraram vinte três *bad smells* candidatos. Em seguida os *bad smells* das demais taxonomias *The Bloaters, The Change Preventers, The Couples* (incluindo *Encapsulators*), The Dispensables e Not Defined foram verificados. Ao verificar entre estes *bad smells* os aderentes ao primeiro critério de exclusão os seguintes *bad smells* foram descartados pelos motivos listados:

1. *Primitive Obsession* (*The Bloaters*) - Este *bad smell* avalia o uso de tipos primitivos (inteiro, flutuante, etc..) como atributos da classe. Esta característica não foi abordada na proposta, pois essa granularidade de informação não se mostrou necessária na definição de elementos similares entre o paradigma orientado a objeto

e o SPEM (seção 4.1.1).

2. *Parallel Inheritance Hierarchies* (*The Change Preventers*) - Este *bad smell* se apresenta quando acontece uma hierarquia de herança paralela. Esta situação é específica ao paradigma de orientação a objeto.

3. *Inappropriate Intimacy* (*The Couplers*) - Este *bad smell* fere o princípio de encapsulamento o que não foi tratado como similaridade entre o paradigma orientado a objeto e o SPEM.

Tabela 4.3 Taxonomias e *Bad Smells*

Taxonomias	***Bad Smells***
The Bloaters	*DataClumps*
	Large Class
	Long Method
	Long Parameter List
	Primitive Obsession
The Change Preventers	*Divergent Change*
	Parallel Inheritance Hierarchies
	Shotgun Surgery
The Couplers (*Encapsulators*)	*Feature Envy*
	Inappropriate Intimacy
	Message Chains
	Middle Man
The Dispensables	*Data class*
	Dead Code
	Duplicate Code
	Lazy class
	Speculative Generality
Not Defined	*Brain Class*
	Brain Method
	Dispersed Coupling
	Intensive Coupling
	Significant Duplication
	Incomplete Library Class
The Objetect-Oriented Abusers	*Switch Statements*
	Temporary Field
	Refused Bequest
	Alternative Classes with Different Interfaces

O segundo critério descarta quaisquer *bad smells* que possuíssem em suas regras de detecção a necessidade de avaliações semânticas. A proposta de *process smells* tem como

caracterização a avaliação de propriedades estruturais do processo, assim detecções com avaliações com teor semânticas que requerem requerem compreender e interpretação do processo não fazem parte do escopo. Por este motivo *bad smells* com esta caracterização seriam descartados. De forma avaliar os *bad smells* quanto ao segundo critério foi necessário completar o levantamento obtendo maior compreensão das características dos *bad smells* e suas regras de detecção. Neste momento da seleção sobraram vinte *bad smells* que por não terem sido descartados pelo primeiro critério passaram pela avaliação do segundo critério.

A aplicação do segundo critério de exclusão foi realizada com vinte *bad smells* que não foram descartados pelo primeiro critério. Os *bad smells* aderentes a este critério possuíam duas características básicas para a detecção dos *bad smells* que os distinguiam. A primeira característica (c1) comum a todos os *bad smells* foi a identificação da necessidade de detecção por avaliações semânticas do código e a segunda característica (c2) adicional para alguns *bad smells* foi o uso de regras de detecção por verificação dinâmica do código.

Avaliando os vinte *bad smells* não descartados que possuíam somente a primeira característica (c1) identificada para o segundo critério de exclusão, os *bad smells Duplicate Code* (*The Dispensables*) e *Significant Duplication* (Not Defined) possuíam regras de detecção que utilizavam avaliação semântica do código de forma a capturar similaridades reconhecendo códigos duplicados em parte ou no todo. Enquanto o *Middle Man* e *Lazy class* (*The Dispensables*) foram verificadas regras semânticas para avaliar se a classe com estes *smells* poderiam ser dispensadas dado que seu código para o *Middle Man* não executava muito mais que chamadas a outras classes e para o *Lazy Class* não continha muito mais que dados. Então todos estes *bad smells* avaliados foram descartados uma vez que não possuíam como principal método de detecção o uso de métricas do código, o que fugiria do contexto deste proposta. Desta forma restaram apenas dezesseis *bad smells* para serem avaliados pela segunda característica (c2) do segundo critério de exclusão.

A segunda característica (c2) do segundo critério de exclusão buscou identificar formas de detecção que possuíam regras em que se faziam necessária a execução do código. Três *bad smells* foram aderentes a característica (c2). O *Dead Code* e o *Speculative Generality*, ambos da taxonomia *The Dispensables* são detectados por regras que ao executar o código percebem trechos que não são acessados. Já o *Incomplete Library Class* (*Not Defined*) identifica a falha de execução do código diante da ausência de parte da biblioteca de código não disponível. Estes três *bad smells* por precisarem de execução do código para suas detecções foram descartados como candidatos a adaptação para *process smells*, pois divergem de um dos princípios da proposta que é detectar *smells* antes da execução do processo.

O terceiro critério de exclusão verificou *bad smells* genéricos ao ponto de ser percebido como já incorporado aos demais *bad smells*. Esta verificação foi aplicada a treze *bad smells*. Nesta verificação foram descartados o *Dispersed Coupling* e o *Intensive Coupling*, ambos do grupo *Not Defined*. Estes *bad smells* foram percebidos como impactos negativos dos *bad smells Shotgun Surgery* e *Data Class* respectivamente. E por fim o *Long Method* (*The Bloaters*) possui parte da característica do *Brain Method* que, além do tamanho do método, verifica também a complexidade, sendo descartado. Assim, estes três *bad smells* também foram descartados.

Sumarizando, dos vinte e sete *bad smells* inicialmente selecionados dezessete foram descartado e restaram dez. Portanto apenas dez *bad smells* seguem para serem adaptados.

Para realizar a adaptação dos *process smells* se baseia na estrutura do *bad smells* que é composta pela descrição, por heurísticas e por sintomas. A descrição caracteriza o *bad smells*. A heurística expressa as regras informais de design que podem reduzir a qualidade do código e estão contidas no *bad smells*. Já os sintomas são extraídos das heurísticas permitindo a quantificação da regra informal e facilitando a atribuição de uma uma métricas para tal verificação. Assim, a adaptação dos *bad smells* consistiu em dois passos.

O primeiro passo foi a adaptação das heurísticas dos *bad smells* para os *process smells* utilizando os elementos na notação SPEM ou adaptando alguns termos para o contexto de processo de *software*, assim como torna mais explicitas a descrição dos *process smells*. A visualização de todas as heurísticas consta no catálogo de *process smells* na seção 4.3.

O segundo passo foi adaptar os sintomas obtidos no levantamento (Apêndice 1 Levantamento dos *Bad Smells*). As adaptações das heurísticas dos *process smells* foram feitas a partir da notação SPEM, conforme exemplo da Tabela 4.4. A adaptação segue inicialmente de forma simples pela tradução dos respectivos termos identificado em código orientado a objeto para elementos do SPEM, como a troca do termo "comportamentos"para "tarefas". Em seguida, tornando mais explicita a descrição do *process smells*, por exemplo para uma atividade com o *process smell Large Activity*, os termos "grande" e "complexa" receberam traduções, onde "grande" passou a corresponder a "possui muitas tarefas" e "complexa" a corresponder a "possui muitos fluxos de decisão". Deste modo, o *Large Activity* um *process smells* que representa uma atividade grande e complexa passa a ser entendido como um *process smells* que representa uma atividade que possui muitas tarefas e possui muitos fluxos de decisão'.

Tabela 4.4 Adaptação das heurísticas para detecção de *smells* em código orientado a objeto para SPEM. Exemplo *Bad Smells Large Class* para *Process Smell Large Activity*.

Heurísticas em código orientado a objeto	Heurísticas adaptados para processo de software especificado com SPEM
1. Fazem fortemente acesso a dados de outras classes mais simples, diretamente ou usando métodos de acesso.	1. Requerem fortemente produtos de trabalho de saída de outras atividades.
2. São grandes e complexas	2. São grandes e complexas
3. Tem um monte de comportamentos que não estão integrados, ou seja, há uma baixa coesão entre os métodos que pertencem a essa classe.	3. Tem um monte de tarefas que não estão integrados, ou seja, há uma baixa coesão entre as tarefas que pertencem a essa atividade.

Em relação a adaptação dos sintomas, com base no exemplo para o *Large Activity* a heurística "grande" foi verificada pelo sintoma de atividade ser grande (possui muitas

tarefas), o que pode ser medido pelo tamanho da atividade. Já a heurística "complexa" (possui muitos fluxos de decisão) tem como sintoma a complexidade excessiva da atividade, o que pode ser medida pela quantidade de fluxos de decisão da atividade. Apesar de verificada nesta etapa da adaptação dos *process smells* a aplicação dos sintomas só foi realizada na segunda fase da elaboração do catálogo na seção 4.2.1. Nesta seção são definidas as estratégias de detecção dos *process smells*, o que requer os sintomas para relação as heurísticas e as métricas de detecção.

4.1.3 Seleção dos Fatores de Qualidade

A seguir será apresentada a terceira etapa da primeira fase da elaboração do catálogo de process smells (item 1.3 da figura 4.2). O objetivo desta etapa foi identificar fatores que determinam a qualidade de processo de software e dentre estes selecionar quais poderiam ser impactos pelos *process smells* e possuíam métodos de avaliação de forma antecipada a execução do processo. A avaliação antecipada foi a premissa adota para que a identificação de possíveis impactos aos fatores de qualidade causados pelas presença dos *process smells* não esperasse para ser verificado somente após a execução do processo de software e desta forma acarretando em problemas no processo, como possivelmente no produto final. A seleção dos fatores de qualidade considerou os atributos e propriedades listados nas figuras 2.2 e 2.3 respectivamente e aconteceu em dois passos, o primeiro para selecionar os atributos e o segunda para selecionar as propriedades.

Em relação ao primeiro passo de seleção dos fatores de qualidade a estratégia foi aplicada verificando quais sub atributos dos atributos de qualidade possuíam métodos de avaliação que antecipasse a execução do processo de software e excluindo os que não evidenciasse uma avaliação antecipada. A seleção considerou a definição dos atributos de qualidade propostos por Kroeger (2011) que por ser ampla foi dividida em sub atributos que tornam mais claro a percepção sobre os atributos de qualidade. Os atributos apresentam se apresentam de forma generalizada enquanto os sub atributos especificam as características do atributo. Por exemplo o atributo de qualidade capacidade de gerenciamento estabelece como um processo precisa ser gerenciado de acordo com os itens que se configuram como sub atributos: previsibilidade, capacidade de monitoramento e capacidade de controle. Os sub atributos representam características de qualidade do processo mais concretas e mais facilmente compreendidas pelos integrantes do processo. Dado esse contexto foram escolhidos subatributos de qualidade que a partir da interpretação das características pudessem ser mensurados ainda na fase de *design* do processo de acordo as propriedades estruturais do processo.

Os subatributos de qualidade compreensibilidade e modificabilidade podem ser mensuradas antes mesmo da execução do processo de software (GARCÍA; RUIZ; PIATTINI, 2004). Estes subatributos pode ser medidos a partir de avaliações aplicadas a voluntários. Estas avaliações consistem na leitura do processo para em seguida, ou responder algumas questões ou realizar algumas tarefas que possam comprovar a compreensibilidade e modificabilidade do processo.

Os demais subatributos como previsibilidade, eficiência, adaptabilidade, capacidade de entrega não foram claramente percebidos como passíveis a participarem de avaliações

que pudessem ser realizada antes da fase de execução do processo. Kroeger (2011) apresenta dois cenários onde, para um ele coleta dados reais e para o outro utilizou simulação com a ferramenta COCOMO II (BOEHM; STEECE, 2000) para alcançar os valores das propriedades e atributos de qualidade. Nos exemplos foram obtidos valores reais ou dados simbólicos para que a avaliação pudesse ocorrer o que diverge do propósito da pesquisa desta dissertação.

Sendo assim, os subatributos selecionados foram compreensibilidade e modificabilidade dos respectivos atributos usabilidade e capacidade de evolução (tabela 4.5). Apenas três das subpropriedades estruturais de qualidade foram selecionada pela influência nos atributos de qualidade. Estas subpropriedades são tamanho, complexidade e modularidade. limitar-se a estas subpropriedades estruturais reflete a adaptação de *process smells* que é realizada nesta pesquisa de dissertação. Por fim serão apresentadas algumas formas as quais as subpropriedades estruturais podem influenciar os subatributos de qualidade do processo. Tais subatributos e subpropriedades estruturais de qualidade foram selecionados também em outro trabalho que avaliavam qualidade no contexto de processo de desenvolvimento de software o que reforça esta escolha (GARCÍA et al., 2005).

Tabela 4.5 Subatributos de qualidade de processo de software selecionados.

Atributo	**Sub Atributo**
Usabilidade	Compreensibilidade
Capacidade de Evolução	Modificabilidade

Os subatributos de qualidade compreensibilidade e modificabilidade possuem definições distintas. A compreensibilidade verifica a facilidade com que um leitor consegue compreender o processo (KROEGER, 2011). Esse subatributo é um dos primeiros que deve ser alcançado para que se possa realizar um processo de software. A partir da compreensão do processo que as atividades e tarefas serão realizadas de forma adequada conforme especificado. Assim, um processo que não favorece uma boa compreensão para os seus participantes corre o risco de enganos que prejudicarão a qualidade do processo e possivelmente do produto final. A modificabilidade reflete a facilidade com que o processo pode ser modificado. Este subatributo de qualidade envolve a avaliação de esforço e impactos para realizar uma mudança no processo (KROEGER, 2011). A capacidade de modificabilidade do processo deve proporcionar facilidade de mudança do processo. As mudanças em um processo de software são necessárias para acompanhar as necessidades do negócio da instituição que possui o processo. A flexibilidade de mudança ma especificação favorece a manutenção evolutiva do processo de forma que seja refletida as atividade e tarefas do dia a dia como eles realmente acontecem.

Os subatributos de qualidade compreensibilidade e modificabilidade podem possuir certa relação pois, para modificar, de forma coerente, é necessário compreender. A compreensão do processo apoia na rastreabilidade necessária para realizar mudanças sem que passos do processo fiquem desconexos.

Em relação ao segundo passo de seleção dos fatores de qualidade a estrategia foi verificar os sintomas dos dez *process smells* adaptados e relacioná-los as sub propriedades

da propriedade estrutural de qualidade proposta por Kroeger (2011). Apesar de existirem mais três outras propriedades de qualidade, semânticas, representativas e a promulgação, estas não foram aplicadas na pesquisa desta dissertação, pois a estes não se caracterizavam de acordo a estratégia de detecção de *process smells* adotada que leve com conta a medição de elementos estruturais do processo. Assim somente a propriedade estrutural e seus sub atributos foram aplicados no contexto da pesquisa.

As propriedades de qualidade são interpretadas de forma objetiva quando avaliadas em relação ao processo de software, uma vez que representam estruturas do processo que podem ser facilmente mensuradas com métricas comuns a qualquer pessoa que avalie o processo. Da mesma forma que os atributos as propriedades de qualidade se apresentam de forma generalizada enquanto as sub propriedades especificam as características da propriedade. A propriedade estrutural que será aplicada nesta pesquisa caracteriza a arquitetura e organização refletindo o design do processo e é subdividida nas seguintes subpropriedades: tamanho, complexidade e modularidade e estabilidade. A subpropriedade estabilidade não foi aplicada no contexto da pesquisa desta dissertação, pois é definida como o período de tempo em que o processo é executado em um determinado ambiente sem sofrer modificações semânticas significativas. Kroeger (2011) ressalta que esta propriedade só pode ser medida ao longo do tempo diferentemente das outras subpropriedades e esta afirmação suporta a não aplicação desta subpropriedade dado que a detecção de *process smells* tem como fundamento a detecção ainda na fase de design do processo de software, de forma a não ser necessário executar ou mesmo obter produtos finais do processo como dados para avaliação.

As subpropriedades estruturais tamanho, complexidade e modularidade tem características especificas conforme suas capacidades de medir o processo. O tamanho é uma subpropriedade estrutural que estabelece a magnitude do processo em termos de quantificação de elementos como atividade, tarefas e outros. O número de atividade é uma das métricas possíveis para medir o tamanho em um processo. A complexidade é uma subpropriedade estrutural do processo em termos do número de fluxos de decisão, partes constituintes do processo e as interfaces disponíveis. As partes constituintes podem por exemplo ser representada pelo número de *stakeholders* envolvidos em uma atividade do processo. A modularidade é uma subpropriedade estrutural que define o quão autocontido pode ser o processo. Um métrica para verificar a modularidade é a quantidade de referências a atividades de um processo feitas por outros processos (KROEGER, 2011).

As subpropriedades estruturais de qualidade podem ser medidas logo que a especificação do processo é estabelecida. Desta forma não se faz necessário verificar a disponibilidade antes da execução do processo para se obter as medidas para estas subpropriedades de qualidade.

Consequentemente, as subpropriedades de qualidade podem influenciar a percepção de qualidade dos subatributos do processo. Cada subpropriedade tem características diferentes uma das outras e da mesma forma influencia de formas diferentes a percepção de qualidade dos subatributos do processo.

A compreensibilidade pode ser impactada negativamente a partir das subpropriedades tamanho e complexidade. A compreensibilidade é reduzida em relação a quantidade de elementos do processo (Kroeger, 2011). A quantidade de elementos de um processo é

verificada pela subpropriedade tamanho. Desta forma ao subpropriedade tamanho é capaz influenciar negativamente a compreensibilidade do processo.

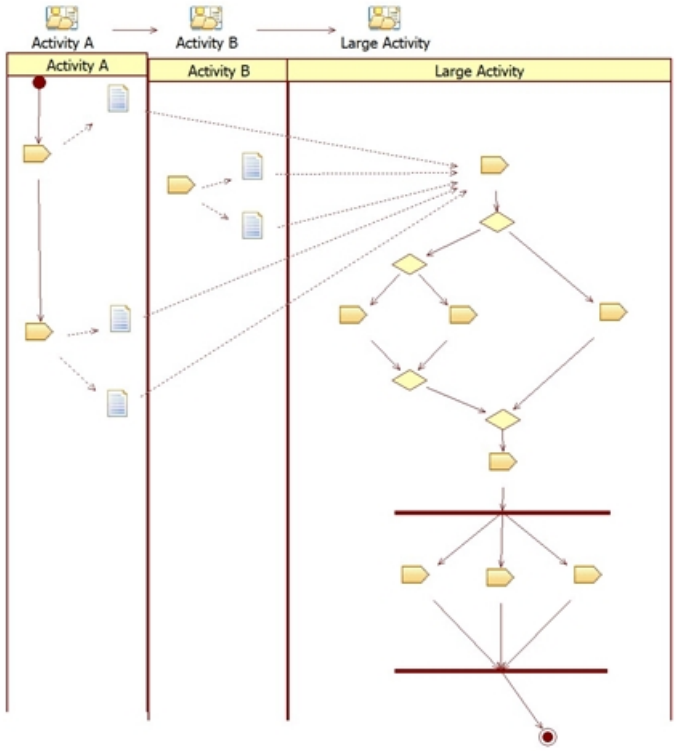
A compreensibilidade também pode ser impactada negativamente pela subpropriedade complexidade em termos de fluxos de decisão, número de partes constituintes e interfaces. Os fluxos de decisão proporciona complexidade, pois são necessários mais elementos no processo para atender os caminhos de decisão. A verificação da quantidade de fluxos de decisão é uma das medidas de complexidade. Quanto às partes constituintes e as interfaces disponíveis, alguns exemplos indicam que quanto mais interações, maior será a complexidade. Para verificar tais situações foi adaptada uma medida correspondente a coesão. A coesão para um processo pode ser compreendida pelo conjunto de operações dos elementos do processo entre si (VANDERFEESTEN et al., 2007) (KHLIF et al., 2009). No escopo da pesquisa desta dissertação a coesão foi observada do ponto de vista das interações entre as tarefas de uma atividade. E assim como em programas de software são considerados que os casos de baixa coesão podem ocasionar mais erros, o mesmo será considerado para processo de software.

A capacidade de modificabilidade pode ser impactada negativamente pela subpropriedade modularidade em específico sobre medidas referentes à acoplamento. O acoplamento faz referência a relação de dependência entre os elementos do processo. Apesar de Kroeger (2011) ter definido a propriedade de modularidade representando somente dependência entre interfaces entre processos, na pesquisa desta dissertação foi feita uma adaptação para esta subpropriedade. Uma vez que a análise de similaridade entre elementos do paradigma orientado a objetos e elementos do SPEM torna compreensível que uma atividade no SPEM é similar a uma classe de código ficará estabelecida que a propriedade de modularidade também corresponderá a relação de dependência entre os demais elementos do processo (atividades e tarefas) que se configuram pela troca de produtos de trabalho. Assim, quanto maior for essa medida, mais difícil será para realizar modificações no processo, uma vez que existem mais dependências no processo.

4.1.4 Consolidação Inicial do Catálogo de Process Smells

Finalizadas estas atividades da primeira etapa de elaboração do catálogo, os dez *process smells* estão caracterizados nesta pesquisa, contendo suas definições através do nome, elemento estrutural do SPEM impactado, descrição geral, possíveis impactos e representação visual. A tabela 4.6 mostra um exemplo da especificação do *process smells Large Activity*. Na descrição deste exemplo se apresenta a caracterização do *process smell* conforme a descrição como uma atividade grande que possui muitas tarefas e muitos fluxos de decisão, nos possíveis impactos constam as subatributos de qualidade compreensibilidade e modificabilidade do processo que podem ser afetados e a representação visual busca fornecer a ideia de como este *process smell* pode se assemelhar, caso encontrado em processo de software especificado com SPEM.

Tabela 4.6 Exemplo do *Process Smell Large Activity*.

Process Smell	*Large Activity*
Elemento impactado	Atividade
Descrição	
Uma *Large Activity* se caracteriza quando uma atividade é grande por conter muitas tarefas e possuir muitas responsabilidades. Ela centraliza as principais ações do processo, deixando, para outras atividades, apenas responsabilidades mais triviais. Além disto, a *Large Activity* utiliza produtos de trabalho destas outras atividades para concluir seus objetivos.	
Possíveis Impactos	
Apesar da sensação de simplificar o modelo usando poucas atividades, uma *Large Activity* possuir muitas tarefas e muitos fluxos de decisão, assim aumenta a complexidade do processo. A inserção de muitas tarefas em uma mesma atividade gera o risco de confundir a responsabilidade da atividade, adicionando tarefas que cuidam de diferentes necessidades do processo. Este fato pode também impactar negativamente a coesão entre as tarefas. Essas duas características podem reduzir a compreensibilidade da atividade. A dependência que uma *Large Activity* tem de produtos de trabalho de entrada insere a possibilidade de impactos negativos caso as atividades quais a *Large Activity* é dependente mudem. Este fato reduz a capacidade de modificabilidade do processo.	
Representação	
	

4.2 SEGUNDA FASE - ESTRATÉGIAS DE DETECÇÃO DOS PROCESS SMELLS

Para que um *process smell* possa ser detectado em um diagrama SPEM é necessário que seja feita a medição do processo buscando por medidas que estejam fora de limites aceitáveis para a manutenção da qualidade do processo de desenvolvimento de software. Quanto a esta medição são utilizadas as subpropriedades do processo, como a subpropriedade tamanho, que podem ser quantificáveis numericamente a partir de métricas. Na medição, a métrica estabelece de forma repetível e monotônica como deverá ser realizada a quantificação de uma determinada propriedade de qualidade de determinados elementos do processo. Assim, um exemplo, a métrica para quantificar a propriedade tamanho para uma tarefa de um processo de software especificado com SPEM realiza conta do total de passos pertencentes a uma tarefa qualquer identificando assim a sua medida de tamanho. Um *process smell* é encontrado quando as medidas das propriedades de qualidade de um determinado elemento do processo de software excedem o valor limiar estabelecido para está medida. Nesta seção apresentaremos as estratégias de detecção dos *process smells* de forma geral e como exemplo o resultado para um *process smells*.

Para esta fase do estudo foram realizados os seguintes passos (figura 4.3):

1. Definição das Estratégias de Detecção dos *Process Smells*. Esta etapa apresenta a estratégia e as métricas aplicada para estabelecer a forma de detecção de um *process smells*. A definição da estratégias de detecção partem da adaptação da proposta de Marinescu (2004) atribuindo as métricas que melhor caracterizaram cada dos *process smells* a técnica de filtragem (atribuição de valor limiar) e decomposição (interação existente entre as métricas a partir de operadores lógicos). As métricas estabelecidas foram atribuídas ao catálogo concluindo assim às estratégias de detecção de cada *process smells*; Estas métricas foram selecionadas de trabalhos relacionados e algumas outras foram adaptadas a partir de métricas existentes no contexto de código fonte.

2. Definição da Estratégia de Definição dos Valores Limiares. Esta etapa apresenta a estratégia aplicada para estabelecer valores limiares para as medidas obtidas pelas métricas aplicadas para detecção dos *process smells*. A definição da estratégia de definição dos valores limiares levou em consideração que a distribuição dos valores obtidos pelas métricas se apresentaram como distribuições não-normais assim a técnica aplicada se baseia no trabalho de Alves et al. (2010), onde foi usada uma função de distribuição de frequência. Com a função de distribuição de frequência os valores extremos de uma distribuição não-normal são classificados pela frequência acumulada se tornando mais expressivos para definir valores limiares para essa tipo de distribuição. A estratégia para identificação de valores limiares foi aplicada ao *script* semi automatizado para a realização de identificação dos process smells diante de processos especificados com SPEM contemplados neste trabalho de pesquisa. Para tanto foram usados estratégias conjuntas entre o armazenamento e tratamentos de dados com *Excel* e *scripts* em linguagem *Python*.

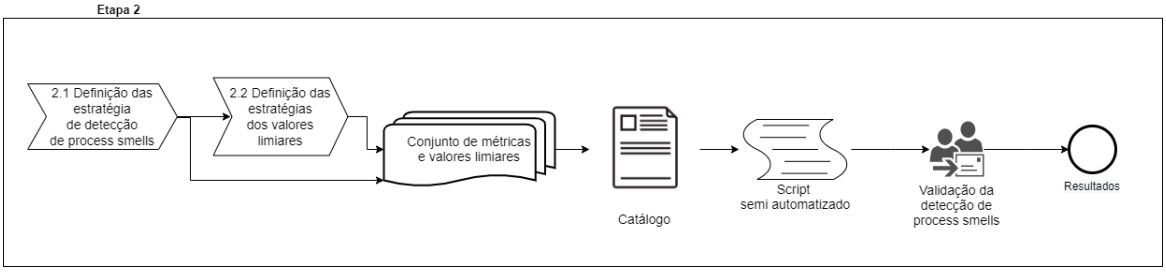


Figura 4.3 Etapas da Segunda Fase de Elaboração do Catálogo de *Process Smells*.

4.2.1 Definição das Estratégia de Detecção de Process Smells

As estratégias de detecção foram estabelecidas a partir da proposta de Marinescu (2004). Assim, para cada *process smell* do catálogo foram verificadas as heurísticas que os caracterizavam e os respectivos sintomas quantificáveis e com base nos sintomas foram selecionadas ou adaptadas as métricas que pudessem mensurá-los. As filtragens e composições foram definidas em seguida e como resultado desta etapa foram concluídas as estratégias de detecção de cada *process smell*. Os elementos heurística, filtragens e composições e métricas podem ser vistas na integra no catálogo na seção 4.3. Os sintomas só foram aplicados pontualmente para apoiar a definidas das métricas. Como exemplo os resultados obtidos para o *Divergent Change* cujo a descrição é:

1. Uma Atividade tem como propósito atender alguma “necessidade” do processo. Um exemplo de “necessidade” em um processo de software pode ser “construir o Termo de Abertura do projeto” ou “validar um documento de especificação”. Assim, o *Divergent Change* ocorre quando uma atividade é configurada para atender mais de uma “necessidade” do processo e, consequentemente, possui diferentes conjuntos de tarefas para cada necessidade a deve atender.

O *Divergent Change* caracteriza uma atividade centralizadora por acumular muitas ”necessidades”do processo. Para tanto este atividade precisa de inputs de diversas atividades das quais ela irá atender as necessidades. Assim, conforme a tabela 4.7, o *Divergent Change* tem como uma de suas heurísticas ”Possui alto acoplamento com atividades predecessoras”dado a dependência que este *process smell* tem em relação as atividades anteriores das quais ele precisará dos produtos de trabalho para atender todas as necessidades que se dispõe a entregar. Essa heurística se traduz no sintoma ”Alto acoplamento”.

O segundo resultado do *Divergent Change* por acumular muitas ”necessidades”do processo é que as suas tarefas nem sempre prestam ações em conjunto dada a diversidade de atendimento que esta atividade realiza. A construção de um termo de abertura do projeto tem ações diferentes da validação de um documento de especificação, se abas necessidades estiverem para uma mesma atividade do processo, possivelmente nela terão tarefas que tratarão o termo de abertura enquanto outras tratarão a validação da especificação. Assim as tarefas que atendem necessidades diferentes não realização trocas de produtos de trabalho, o que caracteriza a segunda heurística ”Atividade pouco coesa”. Esta heurística é devido ao fato da pouco interação entre as tarefas da atividade, a coesão representa o

quanto os elementos se relacionam entre si. O sintoma associado a esta heurística é a "Baixa coesão".

A estratégia de detecção se utiliza também de valores limiares que ajudam a caracterizar elementos que atendem a esses limiares como *process smells*, este valores são aplicáveis na filtragem da estratégia. A princípio esse valores podem ser estabelecidos de forma conceitual e logo que aplicados em um contexto prático devem ser convertidos aos respectivos valores conforme a técnica de valores limiar utilizada. Na pesquisa desta dissertação em virtude dos *process smells* dos dez estabelecidos (seção 4.1.2), só foram atribuídos os conceitos "maior ou igual a alto" e "menor ou igual a baixo" aos valores limiares para detectar os *process smells*. Estes dois conceitos foram suficientes para realizar a detecção dos *process smells* desta pesquisa.

Ao *Divergent Change* foram atribuídos os limiares conceituais na filtragem (tabela 4.7). Ao sintoma "Alto acoplamento"foi aplicado o limiar conceitual "maior ou igual a alto", pois se espera que uma atividade que venha a ser um *Divergent Change* tenha valores elevados de acoplamento, por depender dos produtos de trabalho de muitas atividades predecessoras para atender as várias necessidades que ela se propõe. Já para o sintoma "Baixa coesão"foi aplicado o limiar conceitual "menor ou igual a baixo", pois se espera que este *process smells* tenha baixa coesão, uma vez que as sua tarefas atendem a necessidades especificas do processo e assim tendo pouco interação entre elas.

Tabela 4.7 Caracteíticas do *Process Smells Divergent Change*.

Heurística	Sintoma	Métrica	Filtragem	Composição
1. Possui alto acoplamento com atividades predecessoras	1. Alto acoplamento	NPD(A)	Maior ou igual a alto	Possuir ambas regras válidas. Representado pelo operador lógico "E"
2. Atividade pouco coesa.	2. Baixa coesa	RCT(A)	Menor ou igual a baixo	

As métricas na estratégia de detecção foram selecionadas ou adaptadas de outros trabalhos e associadas a cada sintoma de cada *process smells*. Como primeiro passo foram associados os sintomas as respectivas métricas. Para tanto foram comparadas as características do sintoma e qual elemento estrutural do processo SPEM ele fazia referência com a descrição da métrica e a referência sobre a grandeza de qual elemento do processo SPEM ela conseguia mensurar. Assim os casos de correspondências entre os sintomas e métricas do trabalho de García et al. (2003) e (2004) foram associados. Aos sintomas que não obtiveram métricas preexistentes com a mesma correspondência foram feitas adaptações ou a elaboração de métricas baseadas em métricas do paradigma orientado a objetos.

A métrica tem como objetivo definir o valor de determinada grandeza observada. No contexto do *Divergent Change* a métrica NDP(A) (*Number of Activities which are predecessors (activities dependences of input) of Activity A*) foi definida para o sintoma

"Alto acoplamento". Está métrica estabelece o número de atividades que são predecessoras (atividades dependências de entrada) da Atividade A, dado que o sintoma aponta para o acoplamento esta métrica pode mensurar a quantidade de atividades das quais a atividade A depende, reportando assim o acoplamento, ou seja o valor numérico que representa as dependências da atividade A. Esta métrica foi obtida a partido do dos trabalhos de García et al. (2003) e (2004). Quanto a métrica RCT(A) (*Ratio of Cohesion between Task of Activity*) foi definida para o sintoma "Baixa coesão". Está métrica estabelece taxa de coesão entre tarefa de uma atividade, dado que o sintoma aponta para a coesão esta métrica pode mensurar a taxa de coesa da atividade A, reportando assim a coesão, ou seja dado o valor numérico que representa o total de possibilidade de interação por troca de produtos de trabalho entre as tarefas da atividade A frente a quantidade real de interações que ocorre entre as tarefas. A seleção e adaptações das métricas será apresentado a seguir.

Na estratégia de detecção se estabelece a composição que indica a relação que os sintomas e suas respectivas métricas terão entre si. Para que seja detectado um *Divergent Change* as medidas na filtragem tem que atender aos limiares, assim a composição requer que ambas as regras da filtragem sejam atendidas. Esta fato indica que o operador lógico entre os itens da filtragem precisa ser o "E"indicando que todas as condições precisam ser atendidas para acontecer a indicação de detecção deste *process smells*.

A tabela 4.7 apoia a explicação, com base no exemplo do *Divergent Change*, de como foram estabelecidas as estratégias de detecção para cada *process smell*. As heurísticas 1 (primeira linha da primeira coluna abaixo do cabeçalho Heurísticas) e 2 (segunda linha da primeira coluna abaixo do cabeçalho Heurísticas) foram representadas pelos seus respectivos sintomas mensuráveis 1 (primeira linha da segunda coluna abaixo do cabeçalho Sintomas) e 2 (segunda linha da segunda coluna abaixo do cabeçalho Sintomas). Na coluna métricas foram definidas respectivamente as métricas que poderiam mensurar os sintomas. Na coluna da filtragem se definiu os conceitos sobre os valores limiares esperados correspondentes os respectivos sintomas 1 e 2 e por fim, na coluna composição se compreende em que circunstâncias essas regras entre si deveriam configurar uma ocorrência válida para detecção de um *process smell Divergent Change*. Nesta pesquisa as composições só foram caracterizadas pelo operador lógico "E".

Muitas métricas estão disponíveis em trabalhos relativos as detecção de *bad smells* ou medição de processo contudo as métricas do trabalhos García et al. (2003) e (2004) se mostraram aderente a pesquisa dessa dissertação por se tratarem de métricas elaboradas diretamente para a especificação SPEM, assim, foram as selecionadas.

Das doze métricas disponíveis no catálogo nove foram baseadas nos trabalhos García et al. (2003) e (2004) e três foram elaboradas. A tabela 4.8 apresenta as métricas, sendo que a métrica NPD(T) foi adaptada e as três últimas RCT(A), EP(T) e EP1/3(T) foram elaboradas.

A métrica NPD(T) foi adaptada para atender a verificação de tarefas, pois no trabalho de García et al. (2003) está métrica só se aplicava a atividades. A seguir será explicada a adaptação.

NPD(T) *Number of Task which are predecessors* (*task dependences of input of Task* T). Esta métrica foi adaptada para verificar o número total das dependências predecesso-

ras da tarefa T. A dependência predecessora neste caso corresponde ao número de tarefas predecessoras que fornecem produtos de trabalho para a tarefa T independente destas tarefas serem da mesma atividade que a tarefa T pertence ou não.

Tabela 4.8 Métricas do Catálogo de *Process Smells*.

Nome	Definição	Medição
NSTP(A)	*Number of Tasks of an Activity*	Número de tarefas de uma atividade
NG(A)	*Number of Gateway of the Activity*	Número de fluxos de decisão da atividade
NPD(A)	*Number of Activities which are predecessors* (*activities dependences of input*) *of Activity* A	Número de Atividades que são predecessoras (atividades dependências de entrada) da Atividade A
NWPIn(A)	*Number of Input Work Products of the Activity*	Número de produtos de trabalho de entrada da atividade
NWPOut(A)	*Number of Output Work Products of the Activity*	Número de produtos de trabalho de saída da atividade
NSTP(T)	*Number of Steps of a Task*	Número de passos de uma tarefa
NG(T)	*Number of Gateway of the Task*	Número do fluxo de decisão da tarefa
NWPIn(T)	*Number of Input Work Products of the Task*	Número de produtos de trabalho de entrada da tarefa
NPD(T)	*Number of Task which are predecessors* (*task dependences of input*) *of Task* T	Número de tarefas predecessoras (dependências de tarefas de entrada) da tarefa T
RCT(A)	*Ratio of Cohesion between Task of Activity*	Taxa de coesão entre tarefa de uma atividade
EP(T)	*External Predecessors of the Task*	Número de dependências predecessoras externo da tarefa
EP1/3(T)	1/3 *External Predecessors of the Task*	Quantifica a existência de 1/3 de dependências predecessoras externo da tarefa

As métricas RCT(A), EP(T) e EP1/3(T) foram elaboradas pois eram necessárias para a detecção dos *process smells* adaptados dos *bad smells*. Métricas similares não foram identificadas em outros trabalhos verificados. A métrica RCT(A) é mais complexa entre as métricas elaboradas, assim sua forma de cálculos também será explicada.

RCT(A) *Ratio of Cohesion between Task of Activity*. O objetivo desta métrica é medir a coesão de uma atividade. A coesão corresponde a quantidade de interações, trocas de produtos de trabalho, que são realizadas pelas tarefas de uma atividade dada o total de interações possíveis para as tarefas desta atividade. Assim, é possível verificar o quanto as tarefas de um atividade interagem entre si verificando o quando as tarefas da atividade buscam alcançar um objetivo em comum. Esta métrica foi pensada do com base na métrica *Lack of Cohesion in Methods* (LCOM) (CHIDAMBER; KEMERER, 1994b).

A métrica RCT(A) contabiliza o número de tarefas de uma atividade que trocam produtos de trabalho entre si, dividido pelo número total de possibilidades de pares distintos de troca de mensagens entre todas as tarefas de uma atividade. Para as trocas de produtos de trabalho entre as tarefas foi considerado que estas são indistintas quanto a direção, podendo ser trocas aferentes ou eferentes. Assim foram identificadas apenas uma interação para cada par distinto de tarefas.

Esta métrica, em seu cálculo, utiliza combinações simples para determinar o total de possibilidades de pares distintos de troca de mensagens entre todas as tarefas de uma atividade. Após definidas as combinações possíveis é aplicado um algoritmo para identificação dos pares existentes na atividade. O algoritmo tem a seguinte estrutura:

Algoritmo 1: *Ratio of Cohesion between Task of Activity* RCT(A)

Entrada: Uma Atividade do processo
Saída: Taxa de coesão entre as tarefas da atividade
início
  Listar as tarefas da atividade a ser verificada;
  Estabelecer usando a fórmula de combinação simples o total de possíveis interação entre as tarefas da atividade;
  repita
    Verificar os produtos de trabalho de saída e de entrada;
    se *Verificar se o par de tarefas cuja tarefa em verificação troca produtos de trabalho já foi registrado* **então**
      se sim, então se faz o descarte deste par;
    senão
      senão é feito o seu registro;
  até *enquanto todas as tarefas da atividade não sejam alcançadas. Para cada tarefa*;
  Por fim é feita a divisão da quantidade de pares identificados pela quantidade de pares que seriam possíveis;

EP(T) *External Predecessors of the Task*. O objetivo desta métrica é medir a quantidade de dependências externas que uma tarefa possui. Neste caso, dependências externas

são consideradas como dependência de tarefas externa a atividade a que a tarefa medida pertence. Esta métrica foi pensada com base na métrica *Foreign Data Providers* (FDP) (LANZA; MARINESCU, 2007). A métrica EP(T) consiste em obter o número de dependências predecessoras as quais a tarefa T possua com outras tarefas não pertença a mesma atividade que tarefa T pertence. As dependências predecessoras correspondem ao número de tarefas predecessoras a tarefa T que não estão na mesma atividade que a tarefa T pertence e fornecem produtos de trabalho de entrada necessários para a execução da tarefa T.

EP1/3(T) 1/3 *External Predecessors of the Task*. O objetivo desta métrica é verificar se para uma dada tarefa existem mais dependências externas do que internas em relação a atividade a que ela pertence. Esta métrica foi pensada com base na métrica *Locality of Attribute Accesses* (LAA) (LANZA; MARINESCU, 2007). A métrica EP1/3(T) verifica se a divisão de EP(T) por NPD(T) resulta em um valor maior do que 1/3. Esta métrica verifica se uma tarefa T depende mais de tarefas externas a atividade a que a tarefa T pertence.

A partir das métricas estabelecidas as heurísticas do *Divergent Change* puderam ser atribuídas para serem mensuradas pelas seguintes métricas e conceitos de valores limiares esperados conforme as filtragens da tabela 4.9.

Tabela 4.9 Heurísticas do *Process Smell Divergent Change*.

Heurísticas	Filtragem
1. Possui alto acoplamento com atividades predecessoras	NPD(A)>=Alto
2. Atividade pouco coesa	RCT(A)<=Baixa

Os valores "Alto"e "Baixo"serão apresentados na próxima seção. A disposição de todos os *process smells* e suas estratégia de detecção se encontram no Catálogo dos *Process Smells* na seção 4.3.

4.2.2 Definição das Estratégia de Valores Limiares

Nesta seção será apresentada a escolha da estratégia de valores limiares para os dez *process smells*. A discussão a seguir terá como principal referência o trabalho de Alves et al. (2010). Este é um trabalho que orientou muitas outras pesquisas em relação a utilização de valores limiares para dados que possuem a distribuição não-normal.

Um valor limiar é um valor de controle, cujas medidas obtida por uma métrica não devem ultrapassar, pois do contrário, poderão causar a redução da qualidade do processo de desenvolvimento de software. O valor limiar imprime um valor aceitável para que uma medida apoie a manutenção da qualidade do processo e suporta a tomada de decisão guiada por métricas.

A escolha da estratégia de valores limiares requer a aplicação de todas as métricas e verificação da distribuição estatística dos dados da medição. De forma simplificada uma distribuição pode ser definida como uma distribuição normal quando os valores coletados

de uma medida tendem para a média entre o conjunto de valores obtidos desta medida. Também de forma simplificada, uma distribuição não-normal pode ser definida quando os valores coletados de uma medida não tendem para a média do conjunto de valores obtidos desta medida. A distribuição de uma medida pode ser visualizada por um histograma e assim compreendida como normal ou não-normal a partir da disposição dos valores em relação a média. A distribuição influencia a escolha de um valor limiar.

Em uma distribuição não-normal não é adequado a utilização de valores limiares por estratégias de estatística descritiva. A estatística descritiva utiliza de valores máximos e mínimos, média e entre outros, porém valores em distribuição não-normal acabam distorcidos da realidade. Em uma distribuição não-normal os dados não estão voltados para a média e sim distribuídos irregularmente entre o conjunto de valores obtidos de uma medida. Por exemplo em um conjunto de cinco valores obtidos de uma mesma medida pode ser encontrado os valores {1, 2, 3, 4, 60}. Para estes dados a média seria quatorze, contudo nenhum dos valores desse conjunto se aproximam do valor da média. Assim, utilizar a média para este conjunto causaria uma segregação dos dados onde a maioria dos valores estariam abaixo da média. Se estes dados fosse observados como uma distribuição não-normal uma outra análise poderia ser aplicada. Seria possível analisar que o valor 60 se distância dos demais e assim aplicado uma forma diferente da estatística descritiva para definir um valor limiar adequado para este conjunto de dados.

Na pesquisa desta dissertação foram constatados que os valores de todas as métricas do catálogo possuíam distribuições não normais. Diante desta constatação a estratégia de definição dos valores limiares partiu da adaptação do estudo proposto por Alves et al. (2010), que atendeu a um *benchmark* de sistemas de software, para estabelecer os valores limiares para a detecção dos *bad smells*. As etapas aplicáveis do método de Alves serão apresentas a seguir. A cada etapa será feita explicação da mesma e apresentado parte de um mesmo exemplo de uso da etapa em questão.

1. **Extração de métricas**. Explicação: calcular as métricas para cada tarefa ou atividade do processo e verificar o peso da tarefa ou atividade em relação ao processo. Exemplo de uso: Neste contexto o peso utilizado por Alves foi o número de linhas de código dos métodos ou classes de um software. Desta forma a menor unidade atribuída no presente estudo foi o "passo", assim o peso correspondeu a quantidade de passos que a tarefa ou atividade possuía. Por exemplo para a tarefa T o valor no número de produto de trabalhos de entrada (NWPIn) foi 5 e seu peso foi 3;

2. **Cálculo da taxa de peso**. Explicação: Calcular o percentual dos pesos das tarefas ou atividades em relação ao peso total do processo. Exemplo de uso: Assim para a tarefa T o valor 3 foi dividido pelo peso total do processo que foi 149;

3. **Agregação das entidades**. Explicação: Agregar os pesos das tarefas ou atividades pelos valores apresentados em cada métrica. Nesta etapa o autor evidência que a soma de todos os pesos das tarefas ou atividades deve atingir o valor 100%. Exemplo de uso: Esta etapa destaca o uso da proporcionalidade. Desta forma toda as tarefas cujo NWPIn foi 5 tem seus pesos somados, a soma dos pesos é divida pelo peso total do processo resultando no valor 0,05%;

4. **Agregação de sistemas**. Explicação: Esta etapa não foi aplicada no presente trabalho. Aqui se normalizam os pesos pelos números de sistemas no *benchmark* e posteriormente são agregados todos os pesos igualmente como aconteceu na etapa anterior, porém entre os sistemas envolvidos. Exemplo de uso: Como a pesquisa desta dissertação utilizou dois subprocessos de um mesmo processo, levando em consideração a comparação de que um sistema esta para um processo, não foi aplicada esta etapa;

5. **Agregação por peso**. Explicação: Agregar os pesos consistem em ordenar de forma crescente os valores das métricas e agregar os respectivos pesos tomando os valores máximos obtidos. Exemplo de uso: Neste caso o valor 5 para a métrica NWPIn alcançou o valor máximo de 98.7% uma vez que acumulou os pesos dos valores predecessores;

6. **Derivação de Limiares**. Explicação: Derivar os valores de limiares se trata de definir o percentual de código que se quer representar, nesta adaptação substituímos para processo. Alves (2010) utilizou de valores em uma escala sugerida pelo SIG *quality model* que considerava o risco de como baixo risco entre 0 e 70% e risco muito alto para valores maiores que 90%. Exemplo de uso: Na pesquisa desta dissertação por sua vez foram atribuídos conceitos de valores limiares (seção 4.2.1) "menor ou igual a baixo" que será substituída pela valores entre menores ou iguais a 70% e o conceito "maior ou igual a alto" para os valores maiores ou iguais a 90% visto a distribuição dos dados mais acumuladas nos menores valores das métricas.

4.3 CATÁLOGO DE PROCESS SMELLS

Nesta seção se encontra o catálogo completo dos *process smell* produzido após todas as etapas anteriormente explicadas.

Tabela 4.10 *Process smell Long Input List*

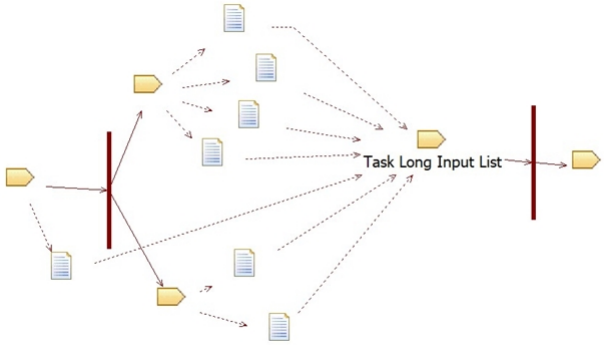
Process Smell	*Long Input List*
Elemento impactado	Tarefa
Descrição	
O *Long Input List* é configurado quando uma tarefa precisa de um grande número de produtos de trabalho de entrada.	
Possíveis Impactos	
A tarefa que possui uma longa lista de produtos de trabalho de entrada pode ser impactada por alguns problemas como: acúmulo de responsabilidades ou ser uma tarefa longa por possuir muitos passos e fluxos. E por sua vez pode prejudicar a compreensibilidade do usuário do processo.	
Representação	
	
Estratégia de detecção	
Heurísticas	1. Tarefa precisa de um grande número de produtos de trabalho de entrada
Métricas e valores limiares	NWPIn(A)>=Alto

Tabela 4.11 *Process smell Feature Envy*

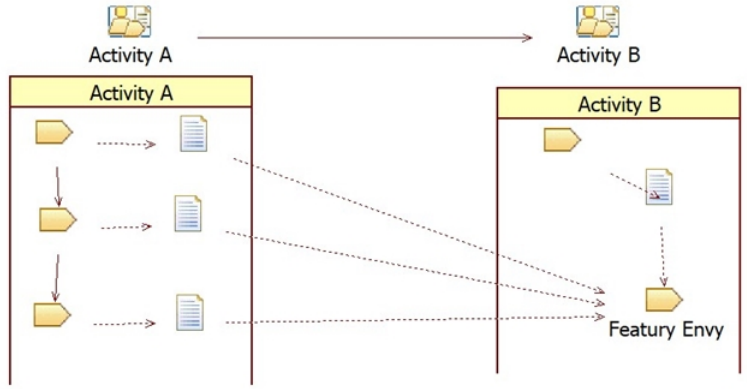
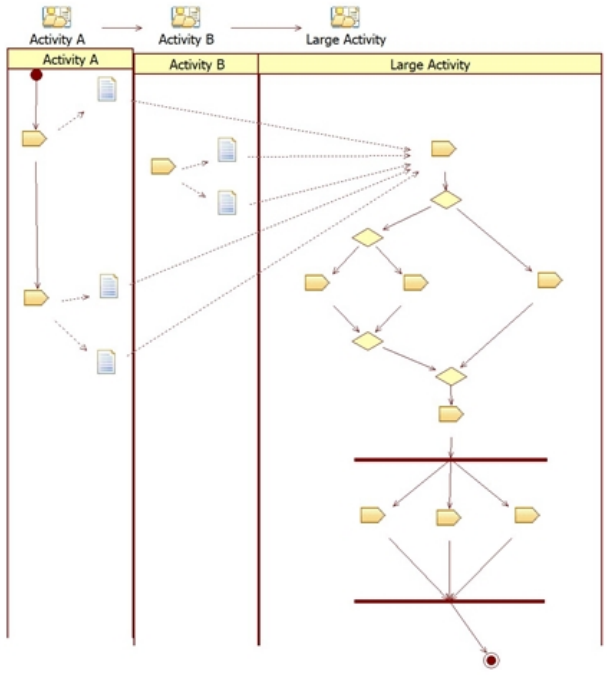
Process Smell	*Feature Envy*		
Elemento impactado	Tarefa		
Descrição			
O *Feature Envy* é representado por uma tarefa que faz uso extensivo de produtos de trabalho de saída de outras atividades. Por esse motivo, talvez essa tarefa deve pertencer a outra atividade.			
Possíveis Impactos			
O *Feature Envy* é reconhecido pela distância que os produtos de trabalho de entrada têm em relação a tarefa que as irá utilizá-los, que normalmente estão em outra atividade e não na atividade a qual a tarefa pertence. Assim, quanto maior for a medida desta distância, menor será a compreensibilidade do executante do processo, principalmente para os casos, no qual o executante precisará entender a tarefa fornecedora.			
Representação			
			
Estratégia de detecção			
Heurísticas	Possui mais que 1/3 do uso de produtos de trabalho de outras atividades	Possui poucas dependências de predecessoras	Possui dependência de poucas outras atividades predecessoras
Métricas e valores limiares	EP1/3(T)=True	NPD(T)<=Baixo	EP(T)<=Baixo

Tabela 4.12 *Process smell Large Activity*

Process Smell	*Large Activity*
Elemento impactado	Atividade
Descrição	
Uma *Large Activity* se caracteriza quando uma atividade é grande por conter muitas tarefas e possuir muitas responsabilidades. Ela centraliza as principais ações do processo, deixando, para outras atividades apenas responsabilidades mais triviais. Além disto, a *Large Activity* utiliza produtos de trabalho destas outras atividades para concluir seus objetivos.	
Possíveis Impactos	
Apesar da sensação de simplificar o modelo usando poucas atividades, uma *Large Activity* possuir muitas tarefas e muitos fluxos de decisão, assim aumenta a complexidade do processo. A inserção de muitas tarefas em uma mesma atividade gera o risco de confundir a responsabilidade da atividade, adicionando tarefas que cuidam de diferentes necessidades do processo. Este fato pode também impactar negativamente a coesão entre as tarefas. Essas duas características podem reduzir a compreensibilidade da atividade. A dependência que uma *Large Activity* tem de produtos de trabalho de entrada insere a possibilidade de impactos negativos caso as atividades as quais a *Large Activity* é dependente mudem. Este fato reduz a capacidade de modicabilidade do processo.	
Representação	
	
Estratégia de detecção	

Heurísticas	1. Possui alto acoplamento com atividades predecessoras	2. Atividade pouco coesa
Métricas e valores limiares	NPD(A)>=Alto	RCT(A)<=Baixa

Tabela 4.13 *Process smell Data Activity*

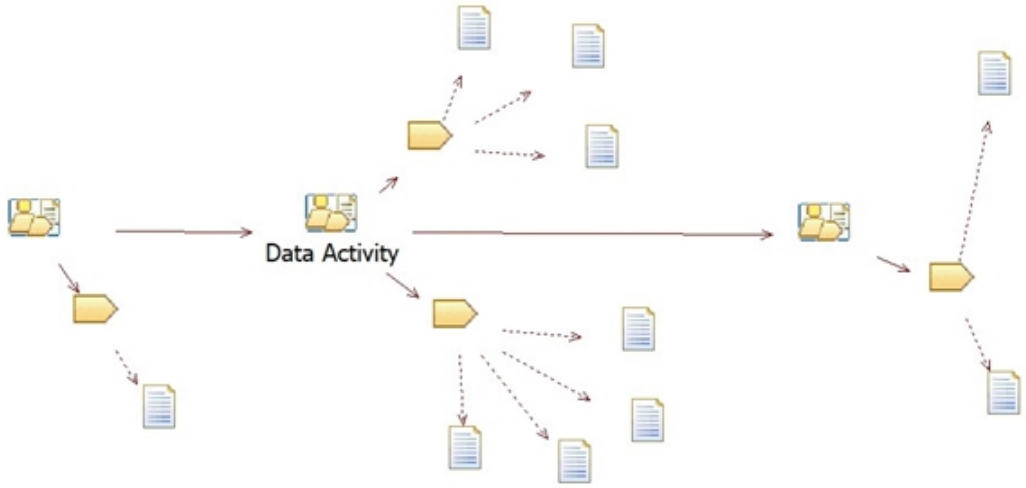
<table>
<tr><td>Process Smell</td><td colspan="2">Data Activity</td></tr>
<tr><td>Elemento impactado</td><td colspan="2">Atividade</td></tr>
<tr><td colspan="3">Descrição</td></tr>
<tr><td colspan="3">Uma atividade se comporta como uma Data Activity quando não possui muitas tarefas, mas fornece bastante produtos de trabalho de saída (outputs).</td></tr>
<tr><td colspan="3">Possíveis Impactos</td></tr>
<tr><td colspan="3">Um agrupamento de muitos produtos de trabalho de saída em uma mesma atividade pode originar tanto um acoplamento disperso (um relacionamento dependente entre muitas atividades) quanto um acoplamento intensivo (um relacionamento dependente entre poucas atividades), dificultando a modificabilidade do processo, uma vez que existem muitas dependências associadas a essa atividade.</td></tr>
<tr><td colspan="3">Representação</td></tr>
<tr><td colspan="3"> 
</td></tr>
<tr><td colspan="3">Estratégia de detecção</td></tr>
<tr><td>Heurísticas</td><td>1. Possui poucas tarefas</td><td>2. A atividade fornece um grande número de produtos de trabalho de saída.</td></tr>
<tr><td>Métricas e valores limiares</td><td>NSTP(A)<=Baixo</td><td>NWPOut(A)>=Alto</td></tr>
</table>

Tabela 4.14 *Process smell Message Chains*

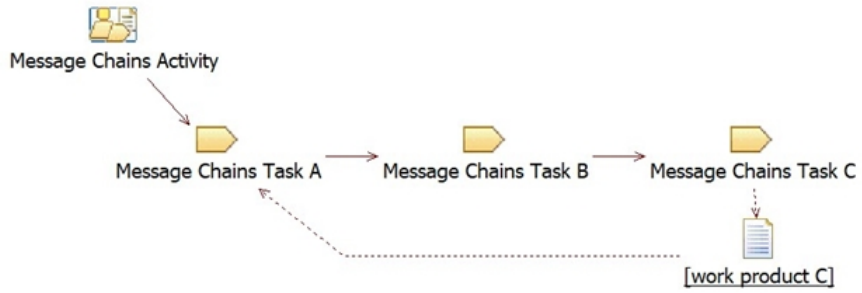
Process Smell	*Message Chains*		
Elemento impactado	Tarefa		
Descrição			
O *Message Chains* define que longas cadeias de trocas de produtos de trabalho entre tarefas podem representar dependências disfarçadas. Uma tarefa precisa de um produto de trabalho de entrada que será gerado por tarefas sucessoras a ela. Assim a tarefa em questão tem que esperar na cadeia de longa tarefas que as outras sejam executadas até que este produto de trabalho de entrada esteja pronto.			
Possíveis Impactos			
Quanto mais longa for a cadeia de tarefas, maior será a dificuldade de leitura do processo uma vez que aumenta a distância entre a tarefa solicitante e a tarefa que entregará os produtos de trabalho de entrada solicitados, afeta a compreensibilidade do processo.			
Representação			
			
Estratégia de detecção			
Heurísticas	1. A tarefa possui dependência de tarefas sucessoras	2. A cadeia de trocas de produto de trabalho entre tarefas ser longa	3. A dependência de tarefas sucessoras deve se apresentar em uma cadeia longa de trocas de produto de trabalho entre tarefas
Métricas e valores limiares	Dependência de tarefas sucessoras >=1	Cadeia longa de trocas de produto de trabalho entre tarefas >=Alto	Cadeia longa de trocas de produto de trabalho entre tarefas >=True (verdadeiro)

Tabela 4.15 *Process smell Shotgun Surgery*

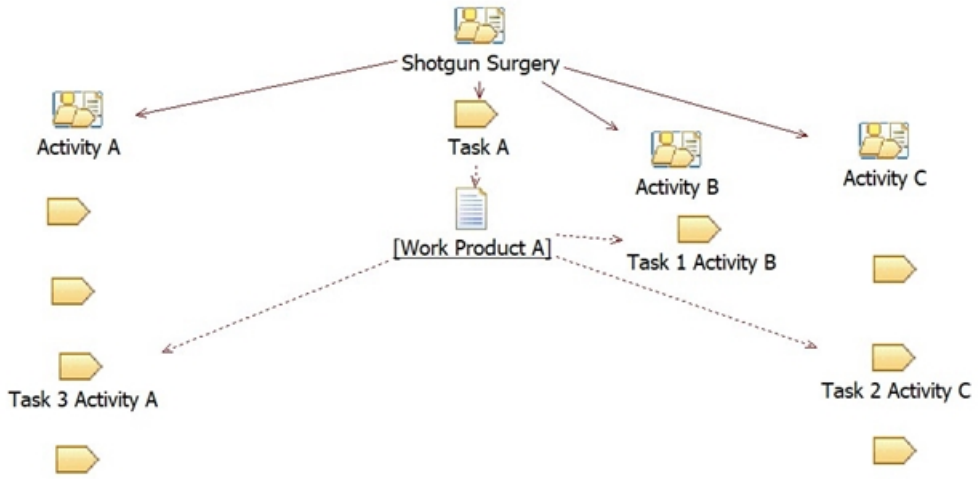
Process Smell	*Shotgun Surgery*
Elemento impactado	Atividade
Descrição	
O *Shotgun Surgery* se caracteriza por uma atividade que ao sofrer uma mudança (em algum produto de trabalho de saída) desencadeará a necessidade de muitas pequenas mudanças em várias outras atividades.	
Possíveis Impactos	
Atividades com *Shotgun Surgery* surgem de um acoplamento disperso no processo, ou seja, atividades dependentes desta atividade estão espalhadas pelo processo. Assim, mudanças estruturais precisarão de maior esforço na rastreabilidade dos pontos que precisam ser ajustados, de forma a reestruturar o processo de forma consistente. Este fato reduz a modificabilidade do processo.	
Representação	
	
Estratégia de detecção	
Heurísticas	1. Atividades com *Shotgun Surgery* surgem de um auto acoplamento eferente disperso
Métricas e valores limiares	NSD(T)>=Alto

Tabela 4.16 *Process smell Work Product Clumps*

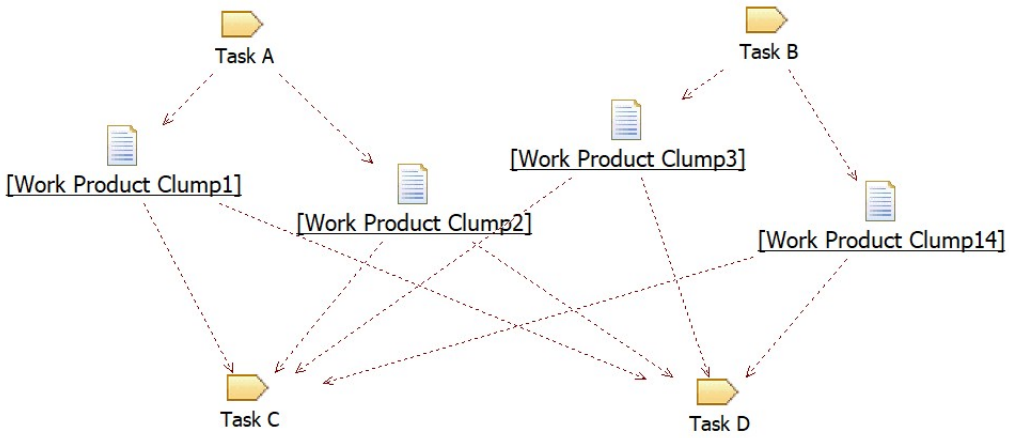
Process Smell	*Work Product Clumps*
Elemento impactado	Produtos de trabalho
Descrição	
O *Work Product Clumps* acontece quando um conjunto de produtos de trabalho de entrada ou de saída de determinadas atividades/tarefas são constantemente vistos juntos. Isto pode indicar que eles devessem pertencer a uma mesma atividade/tarefa.	
Possíveis Impactos	
O *Work Product Clumps* promove a dispersão de um conjunto de produtos de trabalho: eles são necessários para a conclusão de certas tarefas do processo. Essa dispersão reduz a capacidade de modi?cabilidade do processo uma vez que será necessário um constante esforço de rastreabilidade desses produtos de trabalho sempre que houver alguma mudança nesse conjunto.	
Representação	
	
Estratégia de detecção	
Heurísticas	1. Frequência com que uma repetição de conjunto de produtos de trabalhos (dois ou mais) que se apresentaram como entrada em mais de uma tarefa, correspondente a *Frequency of Work Product Clumps* (FRWPC).
Métricas e valores limiares	FRWPC>=1

Tabela 4.17 *Process smell Divergent Change*

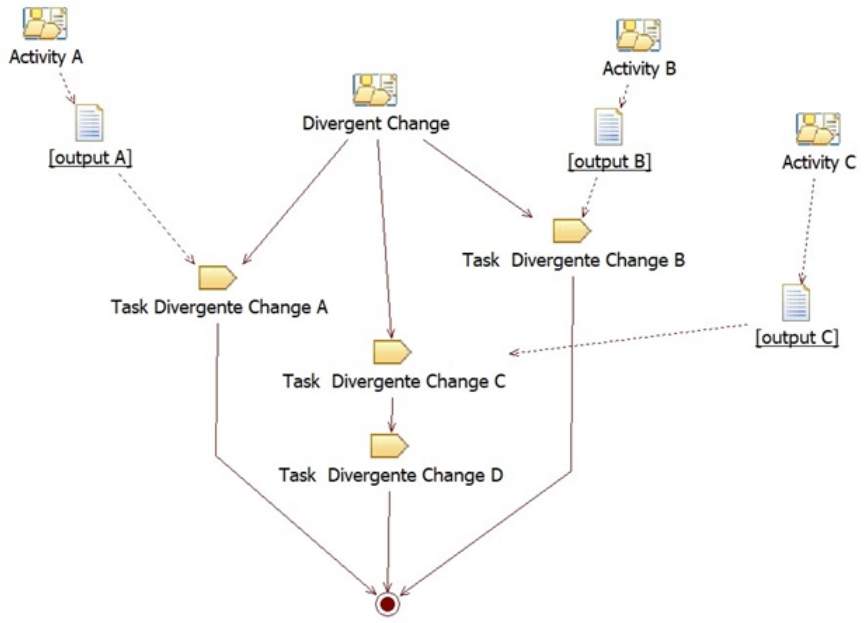
Process Smell	*Divergent Change*	
Elemento impactado	Atividade	
Descrição		
Uma atividade tem como propósito atender a alguma "necessidade"do processo. Um exemplo de "necessidade"em um processo de software pode ser: construir o termo de abertura do projeto; validar um documento de especificação, dentre outras. Assim, o *Divergent Change* ocorre quando uma atividade é configurada para atender a mais de uma "necessidade"do processo, tendo assim, diferentes conjuntos de tarefas para cada necessidade que atenderá.		
Possíveis Impactos		
Essa atividade se encarrega de diferentes propósitos logo, se torna uma atividade pouco coesa, ou seja, as tarefas dessa atividade realizam ações de forma independente das outras tarefas, o que pode prejudicar a compressibilidade dessa atividade.		
Representação		
		
Estratégia de detecção		
Heurísticas	1. Possui alto acoplamento com atividades predecessoras	2. Atividade pouco coesa
Métricas e valores limiares	NPD(A)>=Alto	RCT(A)<=Baixa

Tabela 4.18 *Process smell Brain Task*

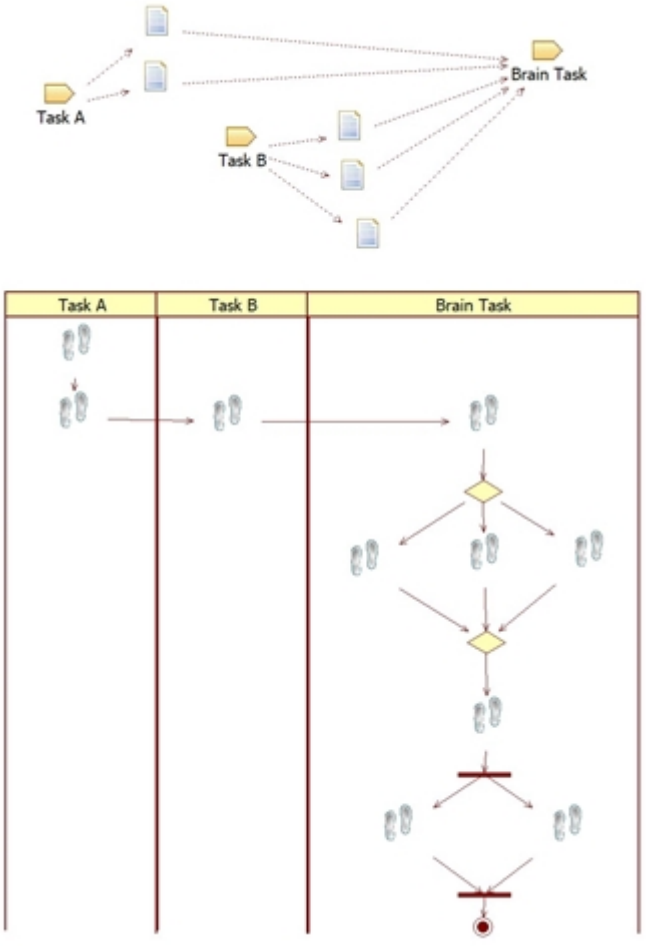
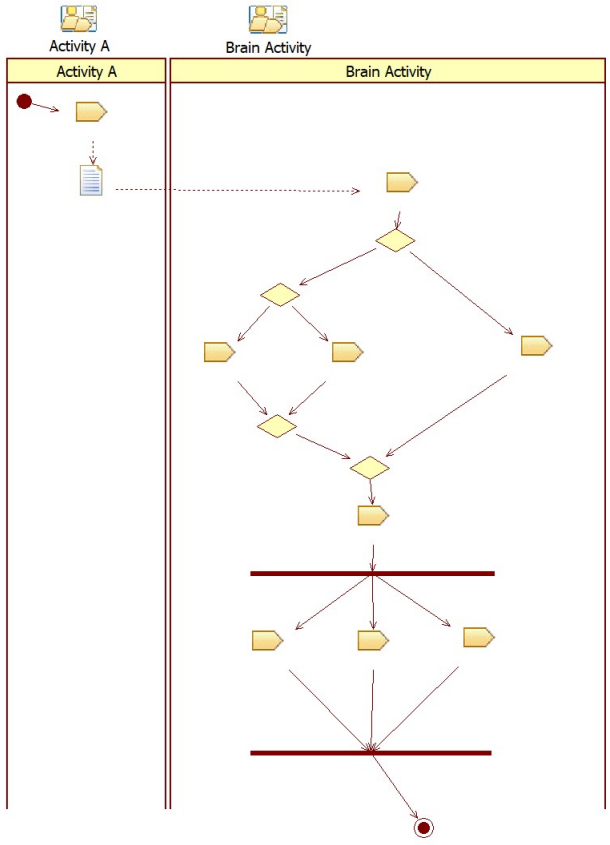
Process Smell	*Brain Task*
Elemento impactado	Tarefa
Descrição	
Uma *Brain Task* é uma tarefa longa que centraliza as ações de uma atividade, possui muitos passos e possui muitos fluxos de decisão para estruturar os passos.	
Possíveis Impactos	
Apesar da sensação de simplificar o modelo usando poucas tarefas, uma *Brain Task* tem muitos passos e muitos fluxos de decisão, o que aumenta a complexidade e reduz a compreensibilidade da tarefa que sofre esse *process smell*.	
Representação	
	
Estratégia de detecção	
Heurísticas	1. Possui muitas tarefas. / 2. Possui muitos fluxos de decisão.
Métricas e valores limiares	NSTP(A)>=Alto / NG(T)>=Alto

Tabela 4.19 *Process smellBrain Activity*

Process Smell	*Brain Activity*
Elemento impactado	Atividade
Descrição	
Uma *Brain Activity* é uma atividade que centraliza as ações de um processo. Essa atividade possui a maior parte das ações relevantes para o processo, possui muitas tarefas e muitos fluxos de decisão para estruturar as tarefas.	
Possíveis Impactos	
Atividades centralizadoras, que concentram muitas tarefas e fluxos de decisão se tornam atividades grandes e complexas, podem reduzir a compreensibilidade do usuário do processo.	
Representação	
	
Estratégia de detecção	

Heurísticas	1. Possui muitas tarefas	2. Atividade pouco coesa	3. Possui muitos fluxos de decisão
Métricas e valores limiares	NSTP(A)=Alto	RCT(A)<=Baixa	NG(T)>=Alto

Capítulo 5

AVALIAÇÃO DA ESPECIFICAÇÃO DOS PROCESS SMELLS

Este capítulo apresenta os resultados da concepção da proposta do catálogo de *process smells* em relação aos objetivos da pesquisa, para tanto foi realizado o estudo de entrevista 1. O capítulo está organizado nas seguintes seções 5.1 com o design do estudo, 5.2 que explica a aplicação, o questionário do estudo e as lições aprendidas com o piloto, 5.3 demonstra os resultados obtidos, 5.4 com as ameaças à validade e 5.5 com as conclusões. O estudo de entrevista 1 teve como objetivo validar a especificação dos *process smells* propostos. Para tanto os *process smells* elaborados foram validados por analistas de processo a partir das suas definições, representações e possíveis impactos. Com as respostas dos participantes foi possível verificar se a proposta de *process smells* era percebida como uma anomalia do processo, bem como compreender o quanto alguns *process smells* se mostraram mais significativos que outros.

5.1 DESIGN DO ESTUDO DE ENTREVISTA 1

O estudo de entrevista foi estruturado com as seguintes características:

Unidade de observação: ponto de vista de analistas de processo

Unidade de análise: Catálogo de *process smells* proposto para processo de software modelados com notação *Software & Systems Process Engineering Meta-Model* (SPEM).

Objetivo: Através de pesquisa exploratória validar o conjunto de *process smells*, para processos especificados com SPEM, validando suas definições, possíveis impactos quanto aos fatores de qualidade do processo compreensibilidade, modificabilidade ou a ambos.

O estudo de entrevista se destinou a responder a seguinte questão de pesquisa:

Process smells podem ser caracterizados como anomalias de processo de software?

Participantes: Doze analistas de processo foram escolhidos por conveniência e a disponibilidade de participação.

Coleta de dados: realização de entrevistas individuais com participantes, onde foi aplicado questionário para verificar se os engenheiros de *process* concordavam com a

definição de cada *smell* e seus impactos negativos para os subatributos de qualidade do processo.

Análise e interpretação dos dados: Estabelecimento de uma taxa de aceitação bem como relacionamento com as percepções dos entrevistados a partir das respostas das questões abertas. Assim, as respostas obtidas foram observadas do ponto de vista quantitativo e qualitativo. O aspecto quantitativo deu-se em relação às taxas de aceitação para cada *process smells*. Quanto ao aspecto qualitativo, as respostas das questões abertas foram relacionadas de modo a justificar as aceitações e rejeições dos *process smells*.

5.2 APLICAÇÃO DO ESTUDO DE ENTREVISTA 1 E FORMATO DO QUESTIONÁRIO

O estudo de entrevista foi aplicado por sessões individuais onde foram realizadas entrevistas com analistas de processo de software, guiadas pelo questionário da pesquisa detalhado na próxima seção. Cada sessão da entrevista foi organizada com uma introdução, perguntas de abertura, questões sobre os *process smells* e questões de fechamento do estudo. Para introdução foram apresentados o objetivo da pesquisa, os elementos do SPEM que compõem a representação dos *process smells* e as definições dos sub atributos de qualidade considerados. As perguntas de abertura trataram de definir o perfil dos participantes. Logo em seguida eram apresentados os *process smells* e as perguntas sobre eles. Assim, para cada *process smells* apresentado foram feitas perguntas baseadas em um questionário composto por respostas dicotômicas (sim/não). Basicamente, as perguntas questionavam sobre definição do *process smell* e possíveis impactos aos subatributos (compreensibilidade e/ou modificabilidade). Por fim, as questões de fechamento do estudo tiveram o objetivo de coletar as percepções dos participantes sobre a proposta do catálogo de *process smells* e sobre o estudo de entrevista aplicado.

5.2.1 Estrutura do Questionário

O questionário foi constituído por três sessões: A primeira foi elaborada com questões para traçar o perfil dos analistas de processo de software entrevistados; a segunda conteve as questões para validação dos *process smells*. Estas questões poderiam receber respostas dicotômicas entre "sim"e "não", sobre a concordância ou não com a definição do *process smell* e o impacto negativo ao sub atributo de qualidade relacionado. Para cada pergunta ainda foi possível informar os motivos que suportam a opinião dada, sendo obrigatório informar o motivo em caso de discordância; por fim, a terceira seção contém as questões de fechamento do estudo de entrevista, buscando entender a percepção que o participante teve sobre a aplicação do questionário, clareza e relevância da proposta. A sessão de questões sobre os *process smells* segue o formato abaixo, conforme exemplo do *Large Activity*. O questionário na íntegra se encontra disponível no Apêndice 2.

1. Quanto a definição do *process smell*

a) Você concorda que uma atividade com muitas tarefas, que possui muitas responsabilidades diferentes e faz uso de muitos produtos de entrada pode ser considerada um *process smell*? [Sim, Não]

b) Comente aqui principalmente caso sua resposta seja negativa. (Opcional para respostas positivas)

2. Verificação dos impactos negativos para a compreensibilidade

a) Você concorda que uma *Large Activity* pode afetar negativamente o atributo de qualidade compreensibilidade? [Sim, Não]

b) Comente aqui principalmente caso sua resposta seja negativa. (Opcional para respostas positivas)

3. Verificação dos impactos negativos para a modificabilidade

a) Você concorda que uma *Large Activity* pode afetar negativamente o atributo de qualidade modificabilidade? [Sim, Não]

b) Comente aqui principalmente caso sua resposta seja negativa.
(Opcional para respostas positivas)

5.2.2 Piloto do Estudo de Entrevista 1

Dadas todas as definições das seções anterior referentes a preparação do estudo de entrevista foi estruturada a aplicação do piloto com dois profissionais. Este piloto possui o intuito de verificar a estrutura proposta e obter *feedback* e oportunidades de ajustes caso necessário antes da aplicação definitiva do estudo. Assim foram alcançadas as seguintes lições:

1. O tempo médio de execução do estudo de entrevista era de 52 minutos;

2. é possível obter dados qualitativos a partir das perguntas abertas ou das perguntas ou considerações dos entrevistados;

3. Quanto a análise qualitativa das respostas abertas foi realizada a gravação da entrevista para que se capturasse de forma completa posicionamento dos participantes e fosse possível fazer o detalhamento das opiniões.

5.3 RESULTADOS DO ESTUDO DE ENTREVISTA 1

Nesta seção serão apresentados os resultados do estudo de entrevista para validação do catálogo de *process smells* contendo os seguintes tópicos: o perfil dos participantes, dados quantitativos sumarizando as respostas dicotômicas, dados qualitativos resultantes das questões abertas e as questões de fechamento. Discute-se também ameaças à validade.

5.3.1 Perfil dos Participantes

Os perfis dos participantes foram analisados em relação aos seus conhecimentos sobre processos de software quanto ao: tipo de experiência, tempo de experiência e quantidade de processos especificados e por fim o fato de possuir certificações relacionadas a processo de software.

Como pode ser visto na figura 5.1, a maioria dos participantes possui experiência profissional, considerando tanto aqueles que têm somente experiência profissional quanto aqueles que têm experiência profissional e acadêmica. A minoria tem apenas experiência

acadêmica e se refere a participantes que são pesquisadores com temas de pesquisa relacionados a processo de software. Quanto ao tempo de experiência e ao número de processos modelados, a maioria dos profissionais têm experiência acima de um ano com especificação de processo e já modelaram mais que dois processos de software (tabela 5.1).

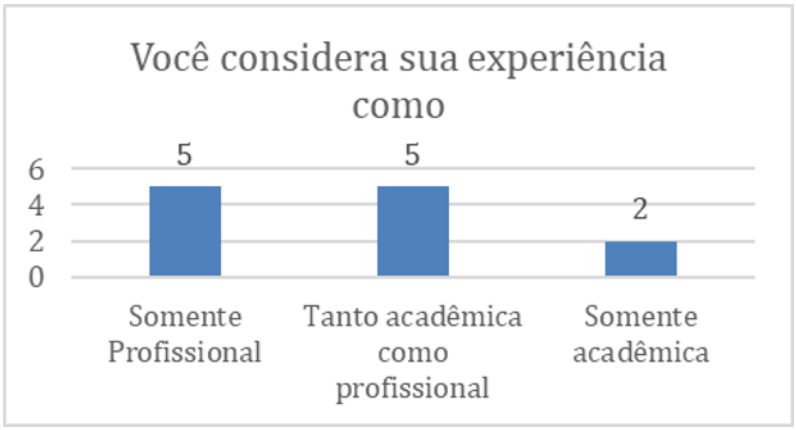


Figura 5.1 Tipos de experiências.

Tabela 5.1 Tempo de experiência e processos modelados.

	Quantidade de Processos			
Tempo de Experiência	**B - Modelei apenas um processo**	**C - Modelei entre 2 à 5 processos**	**D - Modelei entre 6 à 15 processos**	**E - Modelei mais que 15 processos**
F - Acima de 4 anos.		1	3	2
E - Entre 3 à 4 anos.		1		1
D - Entre 2 à 3 anos.		1		
C - Entre 1 à 2 anos.	1			
B - Entre 6 meses e 1 ano.		1		
A - Menos de 6 meses	1			

Quanto às certificações, há participantes com certificados como avaliadores internos de *Capability Maturity Model Integration* (CMMI). Um dos participantes é avaliador CMMI e Melhoria do Processo de Software Brasileiro (MPS.BR). Um participante com certificações Cobit e *Certification in Control Self-Assessment* (CCSA). Sendo assim, dado que metade dos participantes possui alguma certificação (tabela 5.2).

A maioria dos participantes possuem mais de três anos de experiência com processo de desenvolvimento de software. Além da metade dos participantes possuírem certificações de processo atreladas às suas experiências.

5.3.2 Resultados Quantitativos e Qualitativos

As respostas dicotômicas foram a base para a definição de uma taxa de aceitação proposta nesta pesquisa, baseada na razão comparando duas grandezas de uma mesma medida,

Tabela 5.2 Certificações dos participantes.

Certificação	Contagem
Auditor líder - ISO 9001 SGQ	1
CMMI e MPS.BR	1
CMMI e Interno - ISO 9001 SGQ	1
Cobit e CCSA	1
CMMI	2
Não possui	6

calculada pela soma das respostas positivas para cada elemento de cada *process smell* e dividida pelo total de participantes (Taxa de aceitação = soma das respostas positivas de cada *process smell*/total de participantes). Essa taxa foi aplicada para cada um dos três elementos avaliados: definição, compreensibilidade e modificabilidade. As taxas de aceitação para cada *process smell* podem ser vistas na tabela 5.3 nas respectivas colunas Definição, Compreensibilidade e Modificabilidade.

Entre os dez *process smells* apresentados, oito obtiveram apenas taxas de aceitação acima de 75%, como pode ser visto na tabela 5.3. Sendo que o *Shotgun Surgery* recebeu em todas as taxas 100% de aceitação. Apenas o *Brain Activity* e o *Message Chains* obtiveram valores inferiores a 75% nas suas taxas de aceitação. E desta forma ambos foram *process smells* com menor aceitação conforme os especialistas.

Os dados qualitativos apresentam os detalhes das argumentações dos participantes para cada *process smell* e são apresentados a seguir. As avaliações dos dados qualitativos foram organizadas de forma a discutir os prováveis motivos referentes às aceitações e às rejeições dos participantes que originaram os valores das taxas de aceitação de cada *process smells*. Além disso, as respostas abertas apresentam relações diretas com as subpropriedades de qualidade que caracterizam os *process smells* (tamanho, complexidade e acoplamento). As relações entre as subpropriedades de qualidade e *process smells* não estão dispostas no catálogo, elas serão apresentadas nesta seção e orientaram a definição das estratégias de detecção (seção 4.2.1). Os *process smells* estão dispostos de forma decrescente a partir dos totais gerais de aceitação. Por fim, também foram destacadas algumas rejeições dos participantes relacionadas aos papéis e a característica de abstração das atividades do SPEM. Estes itens serão destacados, pois não foram consideradas na concepção dos *process smells* mas influenciaram as respostas de alguns participantes.

5.3.2.1 *Shotgun Surgery*

O *Shotgun Surgery* obteve uma taxa de aceitação de 100% visto que tanto a definição quanto o impacto proposto não receberam nenhuma resposta de rejeição. O *Shotgun Surgery* faz referências a propriedade modularização através da medida de acoplamento. Este acoplamento é disperso, pois se caracteriza por dependências espalhadas no processo em mais de um atividade em relação a atividade *Shotgun Surgery*. Tal acoplamento propõe impacto para o atributo de qualidade modificabilidade. Assim os participantes não discordaram das características deste *process smells*.

Tabela 5.3 Resultados de aceitação dos *Process Smells*.

Process Smell	Definição	Compreensibilidade	Modificabilidade
Brain Activity	0,58	0,58	na
Message Chains	0,67	0,58	na
Data Activity	0,83	na	0,83
Feature Envy	0,83	0,83	na
Long Input List	0,83	0,83	
Work Product Clumps	0,83	na	0,83
Brain Task	0,83	0,92	na
Divergent Change	0,92	0,92	na
Large Activity	0,92	0,92	1,00
Shotgun Surgery	1,00	na	1,00

* na - representam subatributos não aplicáveis ao *process smell*, pois de acordo a definição do *process smell* ele não causará impacto direto para esse subatributo de qualidade.

5.3.2.2 *Large Activity*

O *Large Activity* quanto as aceitações recebeu 92% em relação a sua definição, 92% para a percepção de impacto negativo em relação ao atributo de compreensibilidade e 100% de aceitação para a percepção quanto a impactos negativos ao atributo de modificabilidade. O *Large Activity* compreende três subpropriedades de qualidade. O fato de uma atividade caracterizada como *Large Activity* possuir muitas tarefas faz relação a propriedade de tamanho; já quando se refere a possuir muitos fluxos de decisão apresenta a propriedade de complexidade; e por fim, o ato de utilizar produtos de trabalho de outras atividades aponta para o acoplamento que é uma medida da propriedade modularidade. Neste cenário as subpropriedades de tamanho e complexidade foram definidas na descrição deste *process smell* como impactantes no atributo de qualidade compreensibilidade, enquanto a incidência de acoplamento impactando negativamente o atributo de qualidade modificabilidade.

A rejeição quanto definição foi relacionada ao fato da descrição deste *process smell* não sugerir a quantidade de papéis envolvidos na atividade. Um papel (role) é um elemento

da especificação SPEM que referencia a designação do executante das tarefas do processo. Já a rejeição ao impacto na compreensibilidade propôs que uma atividade por representar uma unidade lógica de trabalho do processo, mesmo com grande tamanho e a alta complexidade, não impactaria negativo quanto à compreensibilidade. Estes tópicos serão discutidos ao final das apresentações dos resultados de cada *process smell*. O impacto negativo quanto a modificabilidade não sofreu nenhuma rejeição assim o acoplamento foi percebido como impactante para este atributo de qualidade. A rejeição geral referente a este *process smells* foi baixa, contudo, destacaram características mais intrínsecas na especificação SPEM que serão discutidas a seguir nesta seção.

5.3.2.3 *Divergent Change*

Em termos de classificação, o *Divergent Change* obteve uma taxa de aceitação de 92%, tanto para sua definição quanto para a percepção sobre seu impacto negativo ao atributo de compreensibilidade. Como a atividade *Divergent Change* atende a mais de uma necessidade do processo, esta tenta atender a mais de uma unidade básica do mesmo, podendo originar tarefas com baixo inter-relacionamento, ou seja, baixa coesão entre as tarefas. Essa baixa coesão foi referenciada como fator de impacto negativo sobre a percepção do atributo de qualidade compreensibilidade. Os participantes, ao rejeitar este *process smell*, fizeram menção ao fato de que as atividades de um processo representam uma unidade lógica de trabalho. Assim, se a atividade consegue atender as necessidades do processo as quais foram designadas para ela, a baixa coesão entre as tarefas não seria um problema, nem geraria impacto negativo para o atributo compreensibilidade. Por fim, se pode perceber que neste estudo o *Divergent Change* obteve uma baixa rejeição.

5.3.2.4 *Brain Task*

O *Brain Task* obteve uma aceitação de 83% para a sua definição e 92% de aceitação na percepção dos participantes por causar impacto negativo para o atributo de qualidade compreensibilidade. A definição do *Brain Task* indica que o mesmo possui muitos passos e possui muitos fluxos de decisão referenciando respectivamente as subpropriedades de qualidade, tamanho e complexidade. Ambas características impactam negativamente o atributo de qualidade compreensibilidade. O *Brain Task* recebeu rejeições referentes a duas situações específicas em relação a especificação SPEM. Na primeira situação os participantes argumentaram que uma tarefa pode ser grande e complexa, uma vez que busca atender uma parte realizável do processo. Assim, não concordaram que o *Brain Task* poderia vir a causar qualquer impacto negativo ao processo. Na segunda situação, especificamente se tratando da propriedade complexidade, foi indicado que seria necessário informar a quantidade de papéis envolvidos na tarefa para melhor percepção da complexidade da mesma *Brain Task*. Contudo, o *Brain Task* obteve uma taxa de aceitação considerável.

5.3.2.5 *Data Activity*

Data Activity obteve uma aceitação de 83% para sua definição, assim como para o impacto negativo à modificabilidade. A definição do *Data Activity* especifica que a

atividade fornece bastante produtos de trabalho de saída (outputs) indicando assim a possibilidade de acoplamento, que por sua vez, impactaria o subatributo de qualidade modificabilidade. Assim algumas das rejeições dos participantes indicam que o acoplamento pode não acontecer uma vez que os produtos de trabalhos gerados podem não ser utilizados por outras atividades. Deste modo seria necessário explicitar o acoplamento deste *process smell* para justificar o impacto negativo ao atributo de modificabilidade. Uma das respostas de rejeição dos participantes pode ser vista nestes trechos "...a entrega de muitos produtos de trabalho não significa necessariamente o aumento do acoplamento entre atividades do processo. Existem produtos de trabalho que podem não ser usados em outras atividades (como documentação, em alguns casos) e, assim, não aumentam o acoplamento." e "por não aumentar necessariamente o acoplamento, acredito que nem sempre gere um impacto na modificabilidade.".

5.3.2.6 *Feature Envy*

O *Feature Envy* ficou com uma taxa de 83% de aceitação, tanto para a definição quanto para o impacto ao atributo de qualidade compreensibilidade. O *Feature Envy* é caracterizado por uma tarefa que faz uso extensivo de produtos de trabalho de saída de outras atividades. Esta característica sugere que o *Feature Envy* está descolado no processo e provavelmente possa ser reposicionado. Por sua vez, esta característica impacta o atributo de qualidade compreensibilidade, devido aos artefatos de entrada pertencerem a outras atividades diferentes da atividade em que a tarefa está, o que requer maior esforço para compreensão. Os participantes não apresentaram muita informação sobre as rejeições quanto ao *Feature Envy*. Apenas foi sinalizado que em alguns casos as dependências deste *process smells* são necessárias na especificação do processo.

5.3.2.7 *Long Input List*

O *Long Input List* ficou com uma taxa de 83% de aceitação igualmente atribuída tanto para a sua definição quanto para o seu impacto negativo ao atributo de qualidade compreensibilidade. O *Long Input List* possui um grande número de produtos de trabalho de entrada. Uma dessas características está relacionada à propriedade de qualidade tamanho e impacta o atributo de qualidade compreensibilidade, pois uma maior quantidade de dados de entrada é mais difícil de compreender do que uma menor quantidade. Para este *process smells* as respostas de rejeição não forneceram informações para compreender o que descaracteriza este o *process smells*. Os participantes que apresentaram rejeição apenas não acreditam que uma lista longa de produtos de trabalho de entrada venha a ser um problema ou mesmo impactar negativamente o atributo de qualidade compreensibilidade. Mesmo com estas rejeições a taxa de aceitação é expressiva.

5.3.2.8 *Work Product Clumps*

O *Work Product Clumps* recebeu uma taxa de 83% de aceitação, igualmente atribuída para a sua definição e para seu impacto negativo ao atributo de qualidade modificabilidade. O *Work Product Clumps* sugere que um conjunto de produtos de trabalho de entrada ou de saída de determinadas atividades/tarefas são constantemente vistos juntos.

Esta característica está relacionada a propriedade de qualidade acoplamento e impacta a modificabilidade. O atributo de modificabilidade é impactado negativamente devido ao esforço necessário para verificar todos os impactos, caso seja necessário modificar ou retirar do processo qualquer um desses produtos de trabalho deste aglomerado. Assim como para o *Long Input List* as respostas de rejeição para este *process smells* não forneceram informações que pudessem descaracterizá-lo. Os participantes apenas não acreditam que um conjunto de produtos de trabalho constantemente visto juntos venha a ser um problema ou que possa impactar negativamente o atributo de qualidade modificabilidade. Assim como o *Long Input List* este *process smells* possui uma taxa de aceitação expressiva.

5.3.2.9 *Message Chains*

O *Message Chains* obteve 67% de aceitação quanto a sua definição e 58% de aceitação quanto ao fato deste *process smells* impactar negativamente o atributo de qualidade compreensibilidade. Este *process smell* define que longas cadeias de trocas de produtos de trabalho entre tarefas podem representar dependências disfarçadas. Uma cadeia longa de trocas de produtos de trabalho proporciona maior esforço de compreensão. Desse modo o atributo de qualidade compreensibilidade é impactado de maneira negativa. Quanto às rejeições em relação a este *process smell*, mais da metade das opiniões compreendem que não há um *process smells*. Neste caso, enquanto as demais opiniões compreendem que o problema do *Message Chains* para processo de *software* tem uma correspondência originada do loop entre as tarefas, o que implica em aumentar a complexidade, e não na troca de produtos de trabalho entre um grande conjunto de tarefas referenciando o acoplamento entre as mesmas, o *Message Chains* recebeu menos de um terço de aceitação dos participantes. Portanto, é um dos *process smell* com menor aceitação no catálogo.

5.3.2.10 *Brain Activity*

O *Brain Activity* recebeu uma taxa de 58% de aceitação. Obteve o mesmo valor para a sua definição, causando impacto de natureza negativa ao atributo de qualidade compreensibilidade. Conforme a definição estabelecida possui "muitas tarefas" e "muitos fluxos de decisão" correspondentes as subpropriedades de tamanho e complexidade, cujas medidas são quantidade de tarefas e quantidade de fluxos de decisão. Entretanto, as respostas dos participantes que discordaram desta definição e por consequência também discordaram do impacto negativo sobre o atributo de compreensibilidade, revelaram que grande quantidade de tarefas ou de fluxos de decisão não prejudicam uma atividade, pois na SPEM, esta representa uma unidade do processo. Desta forma, para os participantes, acumular tarefas e fluxos é uma característica inerente a uma atividade e viria a reduzir de maneira pouco significativa a compreensibilidade da atividade.

Os trechos a seguir representam exemplos da rejeição dos entrevistados quanto ao *Brain Activity* "A atividade por ter um grau de abstração maior ela pode ter muitas subtarefas sem que isso seja um problema de modelagem". ou mesmo por "Mesmo com várias tarefas e vários fluxos a possibilidade de compreensão do processo é melhorada e detalhada, possibilitando assim entendimento rápido do processo." O *Brain Activity*

assim como o *Message Chains* recebeu menos de um terço de aceitação dos participantes sendo os *process smells* do catálogo com menor aceitação.

5.3.2.11 *Brain Activity* vs *Large Activity*

Dentre os resultados avaliados quantos as taxas de aceitação e as respostas qualitativas, pode-se destacar um fato interessante quanto a alta taxa de aceitação do *Large Activity* em relação a baixa aceitação do *Brain Activity*. Estes *process smells* possuem uma estrutura similar, contudo diferem pelo fato do que o *Large Activity*, além de possuir muitas tarefas e fluxos de decisão assim como o *Brain Activity*, possui ainda a relação de acoplamento com atividades ao seu redor. Para compreender mais sobre este fato verificamos as respostas de aceitação de ambos *process smells*. Chegamos a conclusão de que as subpropriedades de qualidade tamanho e complexidade são citadas em ambos como impactantes ao atributo de compreensibilidade. Como o *Large Activity* também é caracterizado pela dependência de produtos de trabalho de outras atividades, os participantes ressaltaram o acoplamento como impactante negativamente no atributo de qualidade modificabilidade, assim como o fato de assumir muitas responsabilidades no processo. Visto que o acoplamento se mostrou um fator entendido como de impacto negativo na qualidade, o *Large Activity* por trazer essa característica, apesar de parecido com o *Brain Activity*, foi mais percebido quanto ao impacto na qualidade do processo.

5.3.2.12 Discussão sobre Características da SPEM

A seguir será apresentada uma breve discussão sobre características específicas do SPEM que influenciaram no posicionamento de alguns participantes durante o estudo de entrevista. Estas características se organizam em dois tópicos: Percepção dos Papéis Relacionados as Atividades ou Tarefas e e Representação de Unidade de Trabalho.

5.3.2.12.1 Percepção dos Papéis Relacionados às Atividades ou Tarefas

Algumas das rejeições em relação ao *process smells*, *Large Activity*, *Brain Activity* e *Brain Task* sugerem que a complexidade depende da quantidade de papéis (role) atribuídas para estas atividades ou tarefas que representam o possível *process smells*. Os papéis representam os realizados dos trabalhos envolvidos nas atividade ou tarefas. A justificativa para estas sugestões é que as ações necessárias a serem feitas na atividade ou tarefa poderão ser divididas entre a quantidade de papéis envolvidos; desta forma a complexidade poderia ser reduzida, uma vez que cada papel assumiria partes do todo a ser feito. A indicação dos participantes é que não há como fazer um bom julgamento sobre a propriedade de qualidade complexidade e o impacto negativo desta sobre o atributo de qualidade compreensibilidade, sem verificar a quantidade de papéis envolvidos na tarefa ou atividade.

Foi indicado ainda que havendo um único papel atribuído a uma atividade ou tarefa certamente seria considerada complexa, uma vez que seria realizada por um só papel. A verificação de papéis pode acrescentar informações para verificação da complexidade e poderá gerar impactos negativos para a compreensibilidade. Porém, ainda não se poderia afirmar que a divisão das ações para mais de um papel reduzisse os impactos negativos sobre a complexidade. É possível que ainda assim os papéis precisem da compreensão do

todo para atuar adequadamente nas suas ações. Este ponto então caberá como trabalho para o desenvolvimento de uma pesquisa futura.

5.3.2.12.2 Representação de Unidade de Trabalho

A SPEM possui os elementos atividade e tarefa que organizam as ações necessárias para executar um processo de desenvolvimento de software. Uma atividade é definida como uma unidade de trabalho que representa o agrupamento de elementos e pode contar com dados de entrada e produzir dados de saída. É uma tarefa que descreve uma unidade de trabalho realizável.

Algumas das rejeições apontam para o fato de uma atividade ser grande e complexa, abstraindo as ações necessárias para se completar uma unidade de trabalho no processo. Esta justificativa indica que as subpropriedades tamanho e complexidade não teriam um impacto negativo no atributo de compreensibilidade, visto que uma atividade poderia conter agrupamentos de elementos para que as suas ações sejam concluídas. A mesma atribuição de unidade de trabalho foi citado para o elemento tarefa, referente ao *process smell Brain Task*, entretanto a especificação SPEM não caracteriza uma tarefa do mesmo modo como o faz para atividade. Então, para este caso, não será atribuída ênfase nesta discussão.

Estas críticas foram apresentadas para os *process smells Large Activity*, *Brain Activity*, *Divergent Change* que atingem atividades do processo.

5.4 AMEAÇAS À VALIDADE

Como ameaças a validade do estudo podemos destacar que apesar de três participantes não conhecerem a SPEM, eles tinham experiência com especificação e modelagem, o que não dificultou seu entendimento da representação visual dos *process smells*. Alguns participantes citaram o desconhecimento do conceito de *bad smells* de código. Isso, porém, não afetou os resultados do estudo de entrevista, dado que os *process smells* e os subatributos de qualidade foram explicados aos participantes antes e durante o estudo.

Quanto à proposição da existência de *process smells* em processo de software, as respostas obtidas ficaram entre relevante, muito relevante e muitíssimo relevante. A duração da realização das entrevistas teve uma média de tempo entre 50 minutos e 1 hora e 20 minutos e houve apenas uma crítica indicando a entrevista como cansativa e com uma sugestão de que esta poderia ter sido feita como um *survey* sem que houvesse a real necessidade de uma entrevista. Contudo, diante da apresentação da pesquisa e esclarecimentos conceituais na maioria das entrevistas, utilizar um *survey* aplicado sem entrevista poderia ocasionar inconsistências das respostas. Quanto a clareza do conteúdo, as respostas foram em maioria entre suficiente, adequado e muito adequado para entender e avaliar os *process smells* propostos.

5.5 DISCUSSÃO

A partir dos resultados obtidos nesta pesquisa, verificamos que os analistas de processo possuem algumas preocupações a serem consideradas na fase de especificação do processo.

A confirmação das definições dos *process smells* e seus impactos sobre os subatributos de qualidade evidenciam esse fato. Até mesmo algumas discordâncias apontam para um refinamento do catálogo e da definição de alguns dos *process smells* propostos e reforçam algumas formas de especificação de processo, que são evitadas pelos analistas de processo.

Por fim, apresentando as questões de finalização e percepções durante a entrevista, compreendemos que o método da pesquisa se mostrou convergente, de acordo aos resultados obtidos. Por se tratar de um estudo exploratório foram obtidas informações de discordâncias em relação ao construto que deverão ser melhor exploradas nos trabalhos futuros.

Capítulo

6

AVALIAÇÃO DAS ESTRATÉGIAS DE DETECÇÃO DOS PROCESS SMELLS

Este capítulo apresenta os resultados da detecção dos *process smells* propostos no catálogo em processos reais em relação aos objetivos da pesquisa, para tanto foi realizado o estudo de entrevista 2. O capítulo está organizado nas seguintes seções 6.1 com o *design* do estudo, 6.2 que explica e as lições aprendidas com o piloto, a forma de aplicação e a estrutura do questionário do estudo, 6.3 a modelagem dos processo reais com a especificação *Software & Systems Process Engineering Meta-Model* (SPEM), 6.4 apresentando os passo para a detecção dos *process smells*, 6.5 com a seleção das ocorrências de *process smells* levadas como estímulos do estudo de entrevista, 6.6 a seção de apresentação dos resultados, 6.7 expondo as ameaças à validade e por fim 6.8 a conclusão.

O segundo estudo de entrevista da pesquisa teve como objetivo realizar a validação dos *process smells* detectados em processos reais utilizados por engenheiros de software. Para tanto, foram estabelecidas estratégias de detecção a partir das características definidas para cada *process smell* do catálogo. E a seguir as métricas capazes de mensurar as características foram selecionadas ou elaboradas com base nos trabalhos de García et. al. (2004) e Khlif (2009). As métricas foram estabelecidas buscando manter a relação com elementos estruturais de modelo do SPEM contidos na representação do *process smell*, assim como refletir os possíveis fatores de qualidade impactados negativamente. Para a aplicação foram selecionados dois subprocessos produtivos de uma filial de empresa desenvolvedora de software[1] e remodelados em SPEM. Estes subprocessos são baseados no processo genérico do tipo espiral. Esta filial, no momento da pesquisa, possuía cerca de cem funcionários que atendiam a projetos de desenvolvimentos de software para clientes estatais e privados. Apesar do quantitativo de funcionários apenas cerca de seis profissionais da equipe de produção atuam de forma indireta com evolução do processo de desenvolvimento de software. A evolução do processo contava com quatro analistas de qualidade que na maioria do tempo estavam dedicados a manutenção do processo.

[1] O nome da empresa foi omitido por questões de confidencialidade de seus processos.

Destes subprocessos foram extraídas métricas que possibilitassem a utilização das estratégias de detecção dos *process smells* do catálogo, assim como estabelecidos valores limiares para as métricas dos processos, utilizando as adaptações do trabalho de Alves et al. (2010). Com estas informações, foi possível aplicar as estratégias detectando ocorrências de possíveis *process smells*. Estes foram avaliados manualmente por profissionais que utilizavam estes subprocessos de forma a obter suas opiniões quanto a eficácia da detecção dos *process smells*. Subsequentemente os profissionais responderam quanto a percepção sobre o quão adequadas foram as estratégias de detecção e os valores limiares aplicados. Assim, com estes resultados foi possível fazer a análise quantitativas e qualitativa referente ao catálogo dos *process smells*.

6.1 DESIGN DO ESTUDO DE ENTREVISTA 2

O estudo foi estruturado com as seguintes características:

Unidade de observação: Ponto de vista de analistas de processo, desenvolvedores e testadores.

Unidade de análise: Avaliação do subprocesso de desenvolvimento e subprocesso de testes, subprocessos mais relevantes no que se refere a entrega de produtos de software, frente aos *process smells* candidatos indicados pela estratégia de detecção.

Fontes de dados: Processo de desenvolvimento de software da empresa XYZ.

Objetivo: Avaliar a eficácia do uso dos *process smells* propostos e suas estratégias de detecção para identificar problemas de compreensibilidade e modificabilidade em processos de software reais de acordo com a percepção de engenheiros de software. A eficácia nesta pesquisa é compreendida como a taxa com que os *process smells* apontados são considerados como problemas pelos engenheiros de software.

O estudo de entrevista 2 se destinou a responder as seguintes questões de pesquisa:

QP1. Os *process smells* detectados impactam negativamente os subatributos de qualidade compreensibilidade ou modificabilidade deste processo?

QP2. As estratégias de detecção destes *process smell* são adequadas para a detecção dos *process smells*?

QP3. Os valores limiares utilizados nas estratégias de detecção são adequados para a detecção dos *process smells*?

Participantes: Foram escolhidos por julgamento cinco profissionais que possuíam maior conhecimento nos processos, participaram de auditorias ou colaboraram para melhoria do processo, segundo dados da empresa XYZ.

Coleta de dados: Entrevistas individuais e aplicação com questionário como guia.

análise, interpretação dos dados: Foi realizada a análise das respostas do estudo de entrevista de modo a obter a taxa de eficácia para cada *process smell* do catálogo proposto. Para tanto foi feita uma avaliação quantitativa e qualitativa das percepções dos entrevistados através das questões fechadas e abertas e das percepções do entrevistador, a partir das notas de entrevista, compreendendo assim os aspectos que colaboram para a eficácia do catálogo.

Entre os dez *process smells* estabelecidos no catalogo apenas oito foram verificados neste estudo de entrevista pois não foram identificas ocorrências nem de *Large Activity*

nem *Message Chains*. Dentre as ocorrências de *process smells* indicados pelas estratégias de detecção foram avaliados apenas um ocorrência de cada *process smell* cujo elemento do processo (atividade, tarefa ou produto de trabalho) era mais significativo para os processos observados. O termo significativo neste contexto representa situações que impactam diretamente a entrega dos produtos de software aos clientes**.** Esta seleção contou com a avaliação de analistas de processo da filial da empresa desenvolvedora de software.

Este estudo de entrevista foi organizado a partir dos seguintes passos:

1. Os subprocessos de desenvolvimento e teste apresentados textualmente foram modelados com SPEM;

2. A detecção dos *process smells* nos subprocessos foram obtidas a partir de uma estratégia semiautomatizada. Primeiramente foram aplicadas as métricas aos elementos do processo. Em seguida foram aplicadas as estratégias para definição dos valores limiares das métricas. E por fim foram aplicadas as estratégias de detecção junto aos respectivos valores limiares identificando assim *process smells* candidatos;

3. Nesta penúltima etapa os *process smells* candidatos foram apresentados aos analistas de processo da empresa XYZ para seleção dos *process smells* indicados que seriam aplicados para validação pelos participantes do estudo;

4. E como última etapa o estudo de entrevista foi aplicado.

A definição sobre o formato do questionário e a forma de aplicação do mesmo foram preparadas para atingir os objetivos e questões de pesquisa do estudo de entrevista.

6.2 APLICAÇÃO DO ESTUDO DE ENTREVISTA 2 E FORMATO DO QUESTIONÁRIO

De modo a validar o *design* e questionário propostos para este estudo de entrevista foi realizado um piloto com um participante analista de processos obtendo alguns pontos de ajuste como:

1. Instruções de localização dos elementos com *process smells* no processo;

2. Melhorar a definições de uma das métricas;

3. Apresentação da tabela de valores das métricas dos elementos avaliados para aplicação dos Valores limiares.

O estudo de entrevista foi aplicado em sessões individuais nais quais os participantes foram entrevistados com um questionário como guia e a execução teve os seguintes passos:

1. Como introdução ao estudo foi mostrado a cada participante a proposta do estudo, os subatributos de qualidade avaliados, os elementos do SPEM utilizados e por fim, foram orientados quanto a forma de navegação nos subprocessos publicados através de páginas web. Os subprocessos foram remodelados com a notação SPEM a partir da ferramenta *Eclipse Process Framework* (EPF), pois antes eram dispostos em documentos *Microsoft Word*.

2. A execução do estudo de entrevista foi estruturada apresentando cada um *process smell* detectado individualmente sendo realizada a sua avaliação em duas etapas:

- **Etapa 1** - Questão inicial

  Nesta etapa era apresentada a questão que indicava o possível *process smells* detectado para avaliação do participante. Esta questão não foi munida de quaisquer informações adicionais sobre as características dos *process smells*, de forma a obter uma avaliação dos participantes direcionada para alguns aspectos do modelo de processo que poderiam impactar negativamente alguma propriedade do processo, porém sem induzir viés na percepção do participante. A questão inicial possuía uma resposta dicotômica sim ou não quanto a percepção sobre o impacto negativo ao atributo de qualidade observado e era sucedida de uma questão aberta sobre o porquê da percepção do participante.

  Exemplo de questão inicial: Analisando a atividade X em relação a quantidade de produtos de trabalho de entrada você acredita que essa atividade reduz a compreensibilidade do processo? [Sim/Não] E por quê?

  Cada questão teve a seguinte estrutura: a) era indicada o elemento do processo para ser avaliado, ressaltando que fosse observada a estrutura deste elemento em relação as características do *process smells*, por exemplo a quantidade de tarefas da atividade. Isto sem comunicar sobre o *process smell* ao participante. Porém as informações que constituem as heurísticas para detecção do *process smell* não foram informadas para evitar o enviesamento, como por exemplo "muitas" tarefas, "baixa" coesão. E por fim questionado se ao observar a estrutura dos elementos era possível perceber impactos negativos ao atributo de compreensibilidade ou de modificabilidade.

- **Etapa 2** - Compreender mais sobre a percepção do participante.

  Esta etapa foi realizada após ao participante responder as questões da etapa 1 onde era exposto aos participantes as definições dos *process smells* e seus possíveis impactos negativos daí então questionado com base nestas informações se a estratégia de detecção e valores limiares das métricas que levaram a indicação de determinado elemento do subprocesso como um possível *process smell* eram adequadas para tanto.

  Exemplo dos elementos apresentados e questões realizadas:

  Configuração do *process smell*

  Process Smell: *Long Input List*

  Elemento impactado: Tarefa

  Descrição

  O *Long Input List* é configurado quando uma tarefa precisa de um grande número de produtos de trabalho de entrada.

  Possíveis Impactos

A tarefa que possui uma longa lista de produtos de trabalho de entrada, pode estar sendo impactada por alguns problemas como: acúmulo de responsabilidades ou ser uma tarefa longa por possuir muitos passos e fluxos. E por sua vez pode prejudicar a compreensibilidade do usuário do processo.

Você concorda que essa estratégia de detecção está correta?

Heurísticas	Tarefa precisa de um grande número de produtos de trabalho de entrada
Métricas e valores limiares	NWPIn(A)>=Alto

Você concorda que este valor limiar está adequado?

NWPIn(T)
5

6.3 MODELAGEM DOS SUBPROCESSOS COM SPEM

Os subprocessos de desenvolvimento e teste da filial da empresa desenvolvedora de software são registrados em documentos *Microsoft Word*. Estes foram lidos e totalmente transpostas para a especificação SPEM com utilização da ferramenta EPF. Para tanto foram estabelecidas algumas premissas: (i) todos as atividades receberam numeração crescente antes dos seus títulos e dispostos pelas ordem com que elas apareciam no processo, (ii) todos as tarefas receberam numeração crescente como subitens (1.1, 1.2...) das atividades as quais elas pertenciam, postos antes dos seus títulos e dispostos pelas ordem com que elas apareciam no processo, (iii) toda tarefa deveria possuir pelo menos um passo, (iv) cada produto de trabalho foi validado na modelagem de forma a ser único, reutilizável, obtendo um identificador (nome) distinto dos demais produtos de trabalho e (v) por fim produtos de trabalho de entrada ou de saída com origem ou destinos externos aos dois subprocessos foram representados desconsiderando essas informações. Após a modelagem realizada, os subprocessos foram revisados por um dos profissionais da empresa para garantir a fidelidade do mesmo, apesar da mudança de representação.

6.4 DETECÇÃO DOS PROCESS SMELLS

O mecanismo de como realizar as estratégias de detecção apesentados na seção 4.2 coletando dados a partir do modelo do processo a ser avaliado foi realizada de forma semiautomatizada. Para tanto algumas escolhas foram feitas. Era possível coletar os dados para obtenção das métricas e valores limiares manualmente, porém acarretaria grande esforço. Assim, partindo dos objetos resultantes da modelagem com EPF, a notação SPEM em *eXtensible Markup Language* (XML) e a publicação do processo em *Hypertext Markup Language* (HTML), foi escolhido o processo publicado em HTML. Esta escolha foi devida a contar com experiências dos pesquisadores com *scripts* de *web scraping* (navegação em documentos HTML) com a linguagem *Python*. Desta forma, um *script* de *web scraping* foi utilizado para percorrer todos os artefatos (páginas HTML) capturando as seguintes informações brutas do processo como nome das atividades, tarefas e seus produtos de

trabalhos de entrada e saída, lista de tarefas que uma atividade possuía e outros.

Os resultados do *script* de *web scraping* foram exportados para uma planilha *Microsoft Excel*. Estes resultados brutos sofreram tratamentos com suporte dos recursos Power Query contidos no Excel que permitem a manipulação de dados. Estes tratamentos de dados consistem em ações de *Extract Transform Load* (ETL), que pode ser ações iniciais para análises de *Business Intelligence* por exemplo. O tratamento dos dados foram convertidos nas medidas de cada atividades e tarefas.

Contudo os recursos do *Excel*, quando se trata de análise de dados, não são tão avançados quanto utilizar uma linguagem de programação. Desta forma, as medidas já calculadas foram tratadas por *scripts* em *Python* para que fossem aplicadas a estratégia de valores limiares descritas na seção 4.2.2. Por fim com as métricas calculadas e os valores limiares estabelecidos foram detectadas entre as atividades ou tarefas aquelas que se caracterizariam como *process smells* usando as estratégias de detecção preparadas segundo as heurísticas e métricas do catálogo de *process smells*.

Tanto o *Message Chains* quanto o *Work Product Clump* foram percebidos desde a definição de estratégias de detecção (seção 4.2.1) que necessitariam cada um de uma estratégia de detecção baseado em algoritmos. Explicar esse fato se mostrou mais adequado nesta seção 4.2.1 por se tratar dos resultados da detecção dos *process smells*.

O *Message Chains* tem na sua definição uma regra que apoia a sua detecção. Esta regra consiste na dependência que uma tarefa tem de esperar por um produto de trabalho de uma outra tarefa que a sucede quanto a posição destas no processo, sendo que ela mesma já forneceu um produto de trabalha para esta tarefa que a sucede. Por exemplo se a tarefa A entregasse um produto de trabalho para a tarefa B e por fim esperasse um produto de trabalho da tarefa B.

Para tanto, foi feita uma verificação geral no processo antes de continuar a elaboração deste algoritmo. Com a manipulação dos dados ainda no Excel foi elaborada para tentar responder a seguinte pergunta: Existe alguma tarefa que receba um produto de trabalho de uma tarefa que a sucede? Assim, para o *Message Chains* foi verificado na fase inicial da preparação do seu algoritmo de detecção que não existiam retro-interações das tarefas com tarefas sucessoras, o que evidenciou a ausência deste *process smell* nos subprocesso avaliados.

Para realizar a detecção do *Work Product Clumps* se faz necessário estabelecer um algoritmo uma vez que sua característica básica é a identificação de um conjunto de produtos de trabalho que são constantemente vistos juntos. Para estabelecer esse algoritmo primeiramente foi necessário definir alguns aspectos como: O que configuraria um conjunto de produtos de trabalho? Quantas vezes seria necessário visualizá-los no processo para se considerar um *process smell*? Para tanto foram estabelecidas algumas premissas.

Cada conjunto de produto de trabalho de entrada é considerado:

1. Onde existam pela menos dois produtos de trabalho de entrada em uma atividade do processo;

2. Um conjunto de produto de trabalho de entrada parte de uma atividade qualquer e deve ser reconhecido em outra atividade mesmo que venha se apresentar como um subconjunto dos seus produtos de trabalho de entrada;

3. Para casos onde no mínimo existam duas repetições do conjunto em mais de uma atividade.

O algoritmo estabelecido tem a seguinte estrutura:

Algoritmo 2: Detecção do *Work Product Clumps*

Entrada: Lista todas as tarefas do processo com mais de 1 produto de trabalho de entrada
Saída: Lista de *Work Product Clumps* detectados
início
WorkProductClumpEmVerificacao;
CodigoDaOrigemEmVerificaco;
criar lista vazia de conjuntos de produtos de trabalho;
repita
Com a lista de conjuntos de produtos de trabalho;
adicionar o código do artefatos da tarefa atual em ordem crescente no atributo código dos artefatos work product clump ;
atribuir o valor 1 no atributo frequência de ocorrência do work product clump;
adicionar o código da tarefa atual no atributo código das tarefas de origem;
até *ter verificado todas as tarefas da lista de tarefas do processo*;
repita
guarde na variável WorkProductClumpEmVerificacao o item atual da lista ;
guarde na variável CodigoDaOrigemEmVerificaco a origem do conjunto de produtos de trabalho do item atual da lista;
repita
se *os código dos artefatos do WorkProductClumpEmVerificacao forem iguais aos códigos do conjunto de produtos de trabalho atual & a variável CodigoDaOrigemEmVerificaco for diferente da origem atual* **então**
acrescente +1 ao atributo frequência de ocorrência da variável WorkProductClumpEmVerificacao;
adicione o código da tarefa atual também como atributo código das tarefas de origem da variável WorkProductClumpEmVerificacao;
remova o item atual da lista de conjuntos de produtos de trabalho;
até *ter verificado todos conjuntos de produtos de trabalho da lista*;
até *ter verificado todos conjuntos de produtos de trabalho da lista*;
repita
guarde na variável WorkProductClumpEmVerificacao o item atual da lista ;
repita
se *o WorkProductClumpEmVerificacao estiver contido no conjunto de produtos de trabalho atual* **então**
acrescente +1 ao atributo frequência de ocorrência da variável WorkProductClumpEmVerificacao;
adicione o código da tarefa atual como atributo código das tarefas envolvidas da variável WorkProductClumpEmVerificacao;
até *ter verificado todos conjuntos de produtos de trabalho da lista*;
até *ter verificado todos conjuntos de produtos de trabalho da lista*;

Observações sobre o algoritmo: 1 - A medida da frequência de ocorrência da

variável recebeu a sigla FRWPC que corresponde a *Frequency of Work Product Clumps*. 2 - estes conjuntos de produtos de trabalho são comparados se forem menores ou iguais em tamanho para que possam estar totalmente contidos nos demais conjuntos de produtos de trabalho de atividades diferentes da(s) sua(s) atividade(s) de origem e são verificados individualmente par a par, pois a disposição de demais conjuntos maiores que os verificados podem camuflar o *Work Product Clumps* tendo um produto de trabalho diferente entre um ou outros do conjunto que é um *clump* (aglomeramento).

Na tabela 6.1 pode ser visto um exemplo o resultado obtido pelo algoritmo 2. Cada linha da tabela evidência uma ocorrência de um *Work Product Clump*. Na primeira coluna são apresentados os códigos os produtos de trabalho que compões a ocorrência detectada. A segunda coluna apresenta o código da tarefa ou tarefas que originam a ocorrência. Sendo mais de uma tarefa de origem significam que estas fornecem como saída os mesmo produtos de trabalho. A terceira coluna apresenta a frequência de ocorrência do *Work Product Clump*. E a última coluna apresenta o código das tarefas que também utilizam o mesmo *Work Product Clump*.

Tabela 6.1 Exemplo de resultados do algoritmo de detecção do *Word Product Clump*.

Código dos artefatos *Work Product Clump*	Código das tarefas de origem	Frequência de ocorrência do *Work Product Clump*	Código das tarefas envolvidas
7 e 8	9	4	20, 62 e 64
1, 2 e 3	1 e 83	3	73

Com base no algoritmo apresentado anteriormente os possíveis *Word Product Clump* foram identificados.

Em relação a detecções dos *process smells* um impasse pôde ser percebido no uso da estratégia de definição de valores limiares. Uma vez que os valores das métricas dos elementos do processo não era bem distribuídas os valores máximos alcançados na etapa 5 da estratégia de definição de valores limiares (seção 4.2.2) - agregação de peso em alguns momentos os valores de 70% e 90% não eram alcançados dificultando a obtenção de um valor limiar. De modo a estabelecer um ajuste para este cenário, isto ocorrendo foram para o valor 90% aceito o primeiro menor valor logo abaixo de 90% e para os casos de 70% eram aceitos o primeiro maior valor logo acima de 70%. Esse ajuste promoveu um estreitamento dos valores limiares para o centro das medidas o que favorecia a continuidade do estudo.

Como resultado das detecções se pôde concluir que os *process smells Message Chains* e *Large Activity* não foram detectados. Em relação ao *Message Chains* foi rapidamente identificado que os subprocesso avaliados não possuíam características para identificação deste *process smell* como foi explicado anteriormente. Quanto ao *Large Activity* foi verificado que os valores limiares para detecção deste *process smell* não se encontrava em nenhuma das atividades dos dois subprocessos avaliados.

Tabela 6.2 Distribuição dos *Process Smells* detectados por processo.

Processo de Teste	Processo de Desenvolvimento
Divergent Change	*Shotgun Surgery*
Brain Task	*Featury Envy*
Data Activity	*Long Input List*
Brain Activity	*Work Product Clumps*

6.5 SELEÇÃO DOS PROCESS SMELLS

Para os *process smells Featury Envy* (16) e *Long Input List* (14), *Shotgun Surgery* (5), *Brain Task* (4) os quais foram detectados nas quantidades indicadas nos parenteses, foi estabelecido um critério de seleção de uma única ocorrência a ser utilizada no estudo de entrevista 2. O critério de seleção foi designar que um analista de processo da empresa XYZ, indicasse apenas um entre as ocorrências detectadas com base na maior relevância para o processo. Relevância neste contexto é entendida como situações críticas do processo por estarem relacionadas a entregas que devem ser feitas aos clientes internos ou externos da empresa. Assim, estas não devem deixar de serem feitas mesmo que complexas ou diante adversidades. Para fins de manter a quantidade de incidências entre os dois subprocesso em equilíbrio, sendo quatro *process smell* para cada, foi solicitada aos analistas que fossem priorizados os *process smells Featury Envy* e *Long Input List*, *Shotgun Surgery* para o subprocesso de desenvolvimento. A tabela 6.2 mostra a distribuição dos *process smells* por subprocesso.

6.6 RESULTADOS DO ESTUDO DE ENTREVISTA 2

Nesta seção serão apresentados os resultados do estudo de entrevista para verificar a aplicabilidade de *process smells* em processos reais de software através da validação dos impactos percebidos pelos participantes quanto aos *process smell*, assim como a validação das estratégias de detecção.

6.6.1 Perfil dos Participantes

Cinco profissionais foram envolvidos no estudo de entrevista sendo dois analistas de teste, dois analistas de desenvolvimento e um analista de processo. Quanto ao tempo de experiência com os processos da empresa apenas um dos analistas de testes tinha apenas um ano na empresa os demais tinham acima de 3 anos. Sobre o conhecimento quanto a temas como *bad smells*, dívida técnica ou tópicos similares, três indicaram ter conhecimento razoável (reconheço alguns padrões) enquanto os outros dois indicaram ter pouco conhecimento sobre o assunto. Em relação ao perfil dos participantes o fato de terem sido escolhidos por julgamento já caracterizou a qualificação dos mesmo para fornecer as respostas para o estudo de entrevista 2.

6.6.2 Resultados Quantitativos e Qualitativos

Os resultados do estudo de entrevista 2 apresentam a análise dos dados extraídos a partir da análise das entrevistas realizadas. De forma a facilitar a compressão dos dados as respostas de concordância ou discordância foram quantificadas e os detalhes quanto aos posicionamentos dos participantes relatados.

Inicialmente foram quantificadas as respostas da primeira pergunta do *design* do estudo de entrevista 2 e estabelecida a figura 6.1. Esta pergunta inicial tinha como propósito coletar a opinião dos participantes sobre a presença sim ou não do *process smell* com base na percepção sobre impactos negativos aos subatributos de qualidade compreensibilidade ou modificabilidade. Para tanto era indicada a observação de certas características em determinada estrutura do processo, como por exemplo a quantidade de produtos de trabalho de entrada em uma determinada tarefa. Assim, sem informações adicionais para realizar esta avaliação concordando ou discordando, os participantes indicavam também os fatos que os apoiaram neste posicionamento. A figura 6.1 representa o agrupamento das respostas após todo o ciclo das entrevistas das concordâncias (sim) ou discordâncias (não) por *process smell* identificado pelas estratégias de detecção.

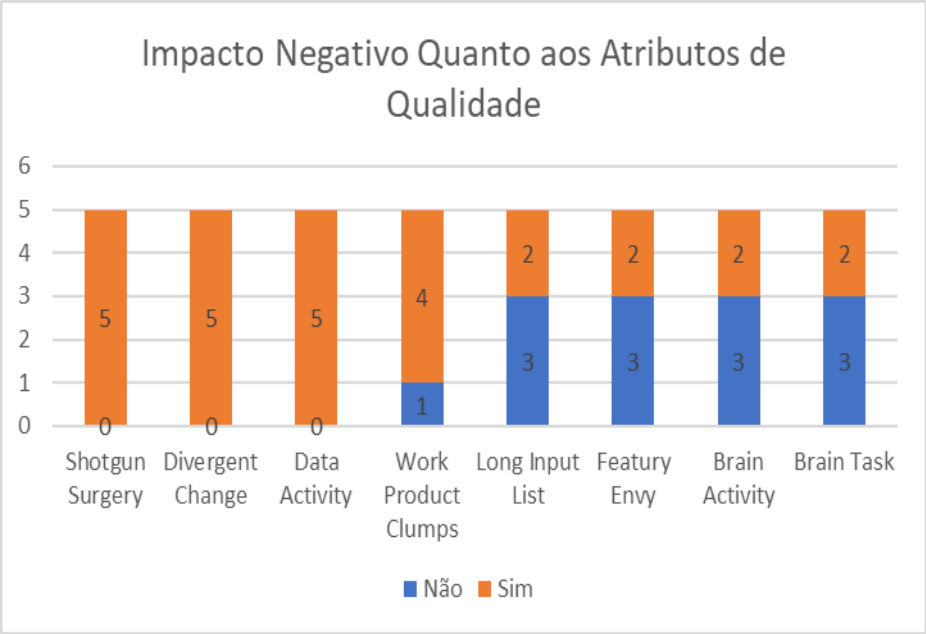


Figura 6.1 Impactos negativos percebidos para cada *Process Smell*

Conforme a figura 6.1 ordenada pelas maiores concordâncias os três primeiros *process smells* se destacam com aceitação dos cinco participantes. Já o *Work Product Clumps* obteve quatro aceitações. Enquanto os últimos quatro *process smells* obtiveram aceitação de dois participantes. O *process smells Shotgun Surgery*, *Divergent Change*, *Data Activity* e *Work Product Clumps* impactam negativamente o subatributo de modificabilidade enquanto os demais impacta negativamente a compreensibilidade.

Este resultado conta com a mudança de três opiniões de discordância para con-

cordância após a apresentação das descrições, possíveis impactos e navegação no processo que detectou os *process smells*. Para ambos os casos os participantes relataram não ter direcionado a atenção corretamente para todos os pontos sugeridos na questão ou então não ter interpretado na verificação do processo modelado em SPEM como a determinada característica do *smell* poderia ser detectada. Sendo para um mesmo participante foi modificada sua opinião quanto ao *Data Activity* e *Divergent Change* já para outro participante aconteceu para o *Shotgun Surgery*.

Na figura 6.1 os *process smells* aceitos por todos os participantes quanto ao fato de serem percebidos por contribuir para causar a redução nos subatributos de qualidade não significam que os mesmos não receberam críticas quanto as heurísticas das estratégias de detecção ou mesmo aos valores limiares. Deste modo para realizar este detalhamento serão apresentados a seguir as análises de cada *process smell*, conforme a ordem da figura 6.1 que culmina nos resultados de maior aceitação e menores quantidades de críticas sucessivamente.

Análises das respostas do estudo de entrevista separada por cada *process smell*

Durante as entrevistas cada *process smell* teve suas heurísticas de detecções e valores limiares validados pelos participantes e estes dados serão apresentados sumarizados a partir de gráficos de barra seguidos de uma discussão quanto ao resultados. O gráfico de barras apresenta a sumarização referente as opiniões de aceitação (sim – se houve concordância e; não – se houve discordância) e na discussão sobre os dados serão abordados os seguintes aspectos sobre cada *process smell:*

Aspectos Quantitativos

1 – Apresentação quantitativa dos resultados considerando as questões objetivas quanto a aceitação dos impactos negativos do *process smell*, das heurísticas e dos valores limiares;

Aspectos Qualitativos

1 – Discussão da Questão Inicial – Tem como proposito avaliar o posicionamento dos participantes frente a questão aberta da primeira questão de cada *process smell* que oportunizou a justificativa sobre a aceitação ou não dos impactos negativos ao atributo de qualidade observado. Neste discussão serão verificadas as opiniões que concordaram ou não com os impactos negativos dos *process smell*. Aos casos de aceitação é possível que as respostas confirmam as características ou proponham características diferentes ao *process smell* ou mesmo aconteça ambos os casos. Enquanto para as discordância serão verificados os motivos que levaram a este posicionamento pelos participantes. A figura 6.2 a seguir representa a análise a ser realizada neste item.

2 – Discussão sobre heurística de detecção e valores limiares – Tem como propósito avaliar a concordância ou discordância quanto a sua heurística de detecção e valor limiar;

A seguir serão apresentados os resultados das análises dos *process smells*. Em alguns momentos das análises dos *process smells* serão citadas elementos do processo como atividade, tarefas ou produtos de trabalho sem atribuição de nomes, devido a confidencialidade dos dados do processo da filial da empresa desenvolvedora de software. Assim os elementos do processo serão citados por numeração como: atividade 1, atividade 2, tarefas 1 ou outros números.

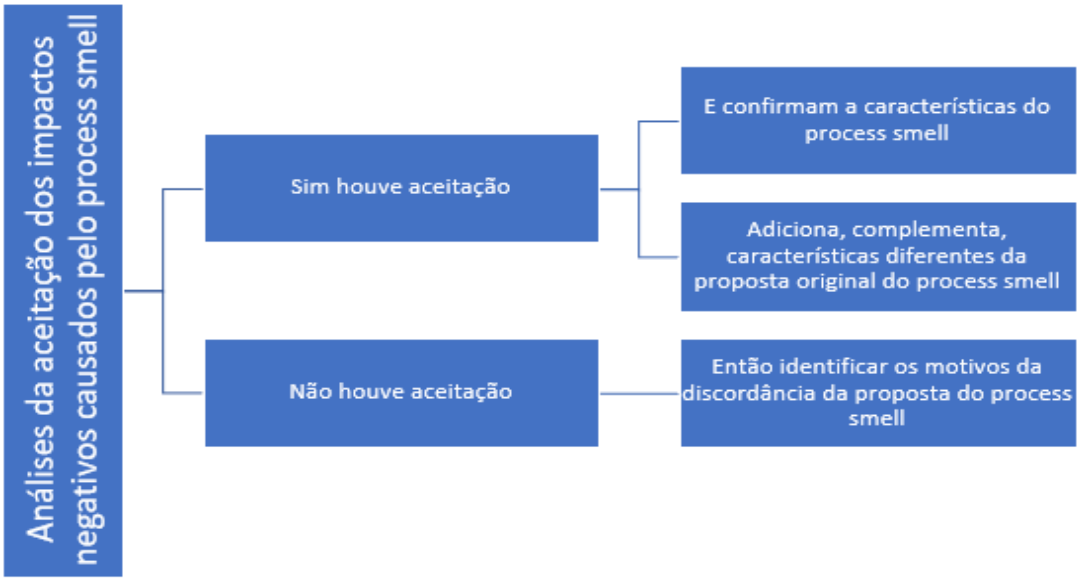


Figura 6.2 Análises da aceitação dos impactos negativos causados pelo *process smell*.

6.6.2.1 *Shotgun Surgery*

A. *SHOTGUN SURGERY* - ASPECTOS QUANTITATIVOS

A análise será iniciada pelo *Shotgun Surgery*, pois este *process smell* além de todos participantes concordarem com o impacto negativo para a modificabilidade, não recebeu nenhuma discordância quanto a sua heurística de detecção "atividades com *Shotgun Surgery* surgem de um acoplamento disperso no processo" ou mesmo ao seu valor limiar. A métrica NSD(A) (*Number of Sucessor Dependent*) identificou na atividade 1 do processo de codificação o valor 12. Este valor foi considerado alto por todos os participantes.

B. *SHOTGUN SURGERY* - ASPECTOS QUALITATIVOS

B.1 Discussão da Questão Inicial do *Shotgun Surgery*

A seguir a questão inicial do estudo de entrevista em relação ao *Shotgun Surgery* que orienta esta primeira parte da análise qualitativa.

> QUESTÃO INICIAL: "PD1 - Analisando a estrutura da "Atividade 1" quanto ao acoplamento que as outras atividades têm em relação ela, você acredita que a Atividade 1 pode colaborar para redução da modificabilidade do processo?
> Você concorda com a pergunta acima?"

B.2 Respostas que "Confirmam as características da proposta do *Shotgun Surgery*"

Os participantes, na sua totalidade, concordaram com a definição do *Process Smell*

Shotgun Surgery. Eles apresentaram principalmente algumas considerações quanto a redução da modificabilidade ocasionada pelo *Shotgun Surgery* que confirmam a descrição contida no catálogo. Estas confirmações podem ser percebidas dos trechos a seguir "Como todas as atividades subsequentes (e relacionadas) sofrerão impactos diante uma modificação na Atividade 1, o fator modificabilidade será impactado." e "se ela sofre alteração as outras podem sofrer retrabalho".

Na primeira afirmação o possível esforço para modificações dado o acoplamento é verificado. O acoplamento é retratado pelas termos "atividades subsequentes (e relacionadas)". O que confirma a principal característica do *Shotgun Surgery* que se representa por uma atividade que sofre uma mudança e desencadeará impactos nas várias atividades que dependem dela. A segunda percepção reforça a ideia que o *Shotgun Surgery* promove impacto na modificabilidade, pois o participante usa o termo "retrabalho". Ele se referiu as mudanças necessárias que deverão acontecer nas atividades dependentes caso a atividade com *Shotgun Surgery* seja modificada.

Uma terceira consideração foi quanto as dependências percebidas entre as demais as atividades sucessoras a atividade 1. No seguinte comentário pode ser identificada esta a percepção: "Na compreensão da especificação existem muitas dependências de atividades". Quando este participante confirma o impacto na modificabilidade enfatizando as dependências percebidas. A dependência entre atividades é uma caraterística do *Shotgun Surgery*. Como este *process smell* é resultado de um acoplamento disperso uma característica inerente é ter atividades dependentes que estão espalhadas pelo processo. Assim este comentário também confirma características desse *process smell*.

B.3 Respostas que "Propõe características diferentes da proposta do *Shotgun Surgery*"

Na discussão foram apresentadas duas sugestões diferentes a proposta inicial mas que poderiam reforçar a detecção do *Shotgun Surgery*. A primeira seria avaliar a posição em que a atividade se encontrava no processo, sendo que atividades mais ao início do fluxo do processo identificadas com estes *process smell* poderiam ser mais críticas quanto aos impactos negativos para a modificabilidade. E a segunda sugestão, complementar a primeira, seria avaliar a cadeia de dependência a partir das atividades que dependem da atividade com *Shotgun Surgery*. Isto porque, uma vez que aconteçam a mudança na atividade *Shotgun Surgery* é possível que exista uma mudança em cadeia para as atividades além das atividades diretamente dependentes do *Shotgun Surgery*. Assim, o impacto total do *Shotgun Surgery* poderia ser avaliado. Estas sugestões indicam que seria possível imprimir maior criticidade ao *Shotgun Surgery* apoiando decisões de ajustes no processo.

B.4 Discussão sobre heurística de detecção e valor limiar do *Shotgun Surgery*

Conforme citado na seção de apresentação dos dados quantitativos tanto heurística a como o valor limiar do *Shotgun Surgery* obtiveram total aceitação pelos participantes. A única heurística do *Shotgun Surgery* definida por "atividades com *Shotgun Surgery* surgem de um acoplamento disperso no processo" reflete que para ele ocorrer é necessário que várias atividades venham a depender de uma mesma atividade e esta assim será uma

atividade com *Shotgun Surgery*. Todos os participantes puderam confirmar a existência do acoplamento eferente disperso, ou seja, muitas atividades dependentes da atividade 1 e espalhadas pelo processo. O termo "espalhadas" se refere ao fato de não haver a concentração do acoplamento em poucas atividades e sim em doze atividades conforme verificado pela métrica NSD(A).

6.6.2.2 *Divergent Change*

A. *DIVERGENT CHANGE* - ASPECTOS QUANTITATIVOS

O *Divergent Change* pode ser considerado o segundo *process smell* mais bem aceito entre os identificados. Analisando as entrevistas foi verificado que todos participantes concordaram com o fato deste *process smell* reduzir a compreensibilidade. Assim como também concordaram que as heurísticas "Possui alto acoplamento com atividades predecessores" e "Atividade pouco coesa" que suportam a detecção deste *process smell*. Já quanto aos valores limiares, a métrica que identifica a primeira heurística *Number of Preccessor Dependent* detectou para a atividade avaliada cinco atividades as quais ela dependia (NPD(A) = 5)(figura 6.3). Este valor limiar recebeu três concordâncias e duas discordância, considerando, assim que apenas dois participantes não acreditam que o fato de uma atividade ser dependente de cinco atividades predecessoras seja um problema. O segundo valor limiar representa a heurística "Atividade pouco coesa" medida pelo *Ratio of Cohesion Task* que recebeu o valor o seguinte valor percentual RCT(A)=0,83. Por sua vez, este valor foi completamente aceito pelos participantes como um valor baixo para coesão da atividade avaliada.

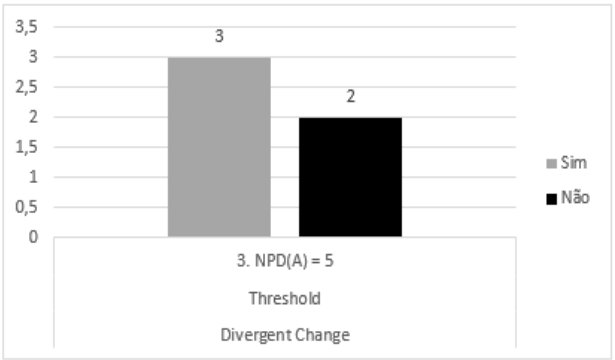


Figura 6.3 Aceitação do valor limiar da métrica NPD(A) do *Divergent Change*.

B. *DIVERGENT CHANGE* - ASPECTOS QUALITATIVOS

B.1 Discussão da Questão Inicial do *Divergent Change*

A seguir a questão inicial do estudo de entrevista em relação ao *Divergent Change*

que orienta esta primeira parte da análise qualitativa.

> QUESTÃO INICIAL: "PT1 - Analisando a estrutura da "Atividade 5" quanto a coesão entre as tarefas e dependências das atividades predecessoras, você acredita que a Atividade 5 pode colaborar para redução da compreensibilidade do processo? Você concorda com a pergunta acima?"

B.2 Respostas que "Confirmam as características da proposta do *Divergent Change*"

Os participantes concordaram totalmente quanto a redução da compreensibilidade ocasionada pelo *Divergent Change*. As considerações apresentadas concordam com a descrição do *process smell*. Os participantes perceberam como principal problema a baixa coesão da atividade enquanto o fato de possuir dependências de atividades predecessoras foi pouco destacado. As percepções sobre a coesão podem ser verificadas por estes trechos: "A atividade realiza situações de forma independente das outras tarefas já traçadas, o que pode prejudicar a compressibilidade dessa atividade." ou em "reduz a compreensibilidade de entendimento do fluxograma, pois o fluxo de atividade não exibe uma ligação entre todas as tarefas, algumas atividades são independentes uma da outra" e também neste comentário "Agrupar tarefas com diferentes propósitos/objetivos podem dificultar a compreensão, tornar algo simples e rápido em complexo e demorado, aumentar o esforço da execução da atividade além do estimado, trazendo prejuízo no entendimento do projeto".

O primeiro comentário o participante indica que "A atividade realiza situações de forma independente das outras tarefas já traçadas". Neste caso ele se refere ao fato das tarefas da atividade apresentarem independência entre si devido a não realizarem expressiva troca de produtos de trabalho entre si. Com o mesmo sentido o segundo comentário traz "atividade não exibe uma ligação entre todas as tarefas" e novamente é reforçada a baixa coesão da atividade. Assim como no último comentário "Agrupar tarefas com diferentes propósitos/objetivos podem dificultar a compreensão" que retrata a independência das tarefas por não realizar trocas de produtos de trabalho entre si. Estes comentário afirma uma característica da descrição do *Divergent Change* determinado por ser uma "atividade se encarrega de diferentes propósitos, desta forma se torna uma atividade pouco coesa".

Este é o comentário que envolveu a relação de dependência de atividades predecessoras "Foi identificado que existe uma baixa coesão entre as tarefas e, portanto, há uma grande dependência de artefatos de outras atividades". Neste caso, o participante ao perceber a baixa coesão da atividade, realizou a verificação de que os produtos de trabalho eram fornecidos diretamente de atividades predecessores e não a partir da troca de produtos de trabalhos das tarefas entre si. Essa característica se define no *Divergent Change* pois ele "ocorre quando uma atividade é configurada para atender a não somente uma "necessidade"do processo". No contexto de processo este cenário pode ser evidenciado pela quantidade de atividades predecessores que a atividade possui. Isto associado ao não ter uma boa coesa imprimi que as atividades predecessoras solicitaram diferentes respostas para a atividade *Divergent Change*.

B.3 Discussão sobre heurística de detecção e valor limiar do *Divergent Change*

As heurísticas estabelecidas para o *Divergent Change* foram completamente aceitas e os participantes não fizeram comentários adicionais. Assim pode se verificar que a heurística da atividade possuir "alto acoplamento com atividades predecessores" apesar de não ter sido percebido como característica mais evidente do *process smell* ao ser apresentada ao participantes foi validada sua influência para detectar este *process smell*. Esta heurista potencializa o acontecimento deste *process smell* uma vez que a partir de muitas atividades predecessoras diferentes "necessidade" do processo são solicitadas a uma mesma atividade. A segunda heurística "Atividade pouco coesa" já havia sido percebida desde a questão inicial assim foi rapidamente confirmada pelos participantes. Este fato conclui que as diferentes "necessidades" do processo solicitadas a uma mesma atividade, precisam de dados e fluxos diferentes o que ocasiona a baixa coesão a atividade que atende diferentes "necessidades" do processo.

Apesar das heurísticas serem totalmente aceitas o mesmo não acontece com os valores limiares. A Atividade 5 que foi detectado com *Divergent Change* foi medida pela métrica RCT(A) e obteve o valor 0,83%. Este valor estava bem abaixo do valor limiar de 70% e foi aceito por todos os participantes, porém dois participantes fizeram um mesmo comentário sobre este valor. Eles consideraram que na tabela da função de distribuição o valor atingido pelo valor limiar 70% para RCT(A) foi 6,7% considerado um valor muito baixo podendo não ser completamente representativo para se definir baixa coesão em uma atividade. Os participantes citaram que considerariam que até 30% de RCT(A) ainda poderia ser considerado um percentual para baixa coesão, valor este que está além dos 6,7%. Neste cenário, na tabela da função de distribuição o valor limiar 88,6% alcançou 33,3% do RCT(A). Sendo assim este valor foi considerado ideal para ajuste ao valor limiar 70% aplicado no estudo de entrevista para caracterizar valores baixos segundo estudo do Alves et al. (2010). A tabela da função de distribuição do RCT(A) é apresentada para facilitar a compreensão destes fato (tabela 6.3).

Quanto ao valor limiar para a métrica NPD(A) dois participantes discordaram conforme foi apresentado no gráfico da análise quantitativa. Ambos se posicionaram desconsiderando que o valor cinco fosse um valor que alto quanto a dependências predecessoras para uma atividade. E por sua vez acreditaram que o valor nove, logo posterior, que representava noventa e quatro porcento das valor de NPD(A) do processo seria um valor adequado para ser considerado alto. Este valor limiar foi ajustado usando a estratégia apresentada no capítulo de construção da segunda etapa do catálogo de *process smells*. Assim o resultado obtido devido a adaptação aplicada, mesmo compreendida pelos participantes contou com essas duas discordância quanto ao valor aplicado.

De forma a explicar as duas discordância será utilizada a tabela de distribuição do NPD(A). Na tabela 6.4 foi obtido um valor limiar de 94% correspondendo ao valor de 9 atividades predecessores dependente. Contudo a estratégia aplicada ajusta os valores limiares altos que correspondem aos limites iguais ou maiores que 90% para o primeiro menor valor de valor limiar disponível uma vez que o valor 90% não seja contemplado na tabela de distribuição. Este adaptação visa não extrapolar os valores mensuráveis para a detecção de um *process smell*. Usando a tabela como exemplo é possível que

Tabela 6.3 Distribuição da Métrica RCT(A).

RCT(A)	NSTPWieghtCum %
0	30.2
0.83	53.0
1.8	66.4
6.7	71.8
19.1	79.2
33.3	88.6
66.7	91.3
100	100.0

Tabela 6.4 Distribuição da Métrica NPD(A)

NPD(A)	NSTPWieghtCum %
0	8.1
1	18.1
2	31.5
3	53.7
5	63.8
9	94.0
21	100.0

acumuladamente um valor limiar avance imediatamente de 63.8% para 100%. Para isso bastava que o último valor a ser acumulado fosse o valor de atividades predecessores não existindo também o valor 21 que por fim acumula o valor de 100%. Este cenário tornaria a detecção inviável, assim utilizar o primeiro menor valor disponível ao valores limiares altos com limites iguais ou maiores que 90% se mostrou razoável.

Assim o posicionamento dos dois participantes sugere que sejam usados os valores logo superiores ao valor limiar para a detecção dos *process smells*. Neste caso conforme apresentado na tabela de distribuição do NPD(A) o valor logo disponível possui um valor limiar de 94% e corresponde ao valor nove de atividades dependentes predecessores. Esse seria o valor ideal conforme a percepção dos participantes e por este motivo eles discordaram.

6.6.2.3 *Data Activity*

A. *DATA ACTIVITY* - ASPECTOS QUANTITATIVOS

O *Data Activity* recebeu aceitação de todos os participantes quanto ao fato de apoiar a redução da modificabilidade do processo, assim como de ser detectável pelas heurísticas "1. Possui poucas tarefas" e "2. Atividade que fornece um grande número de produtos de trabalho de saída". Quanto aos seus valores limiares ambos foram questionados. A

métrica número de passos da atividade *Number of Steps NSTP(A-Tasks)*, relacionada a heurística 1, mediu o valor 11 para a atividade avaliada e recebeu três concordância e duas discordância (figura 6.4). Enquanto a métrica número de produtos de trabalho de saída *Number of Work Product Out NWPOut(A)*, relacionada a heurística 2, mediu o valor 12 e recebeu a concordância de quatro participantes considerando este um valor alto. Mas recebeu a discordância de um participante que considera 12 um valor baixo (figura 6.5).

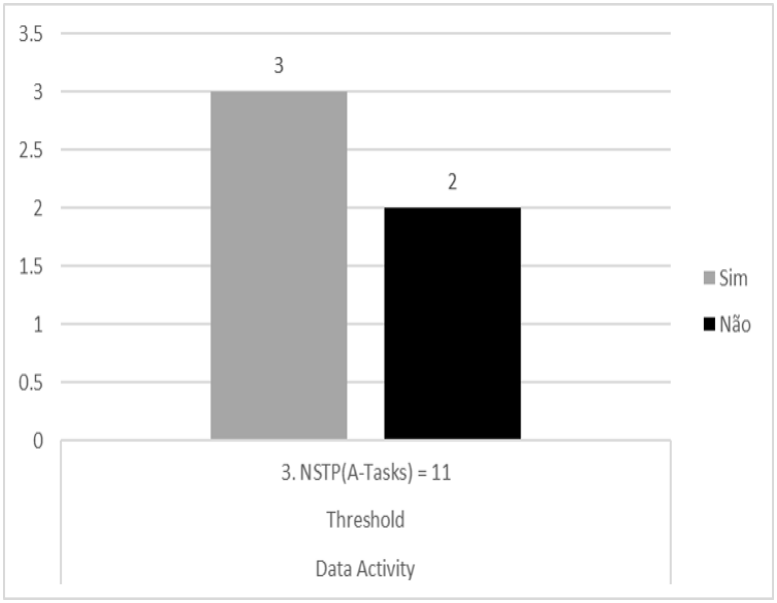


Figura 6.4 Aceitação dos valores limiares da métrica NSTP(A-Tasks) e NWPOut do *Data Activity*.

B. *DATA ACTIVITY* - ASPECTOS QUALITATIVOS

B.1 Discussão da Questão Inicial do *Data Activity*

A seguir a questão inicial do estudo de entrevista em relação ao *Data Activity* que orienta esta primeira parte da análise qualitativa.

> QUESTÃO INICIAL: "Analisando a estrutura da "Atividade 4." quanto a quantidade de produtos de trabalho de saída e a quantidade de tarefas. Você acredita que a Atividade 4 pode colaborar para redução da modificabilidade do processo?
> Você concorda com a pergunta acima?"

B.2 Respostas que "Confirmam as características da proposta do *Data Activity*"

Na questão inicial não foram feitos comentários em relação a quantidade de tarefas da atividade. As respostas quanto ao impacto negativo para a modificabilidade no processo se concentraram na percepção sobre as possíveis dependências que uma atividade com muitos produtos de trabalho de saída pode proporcionar. Alguns trecho das entrevistas

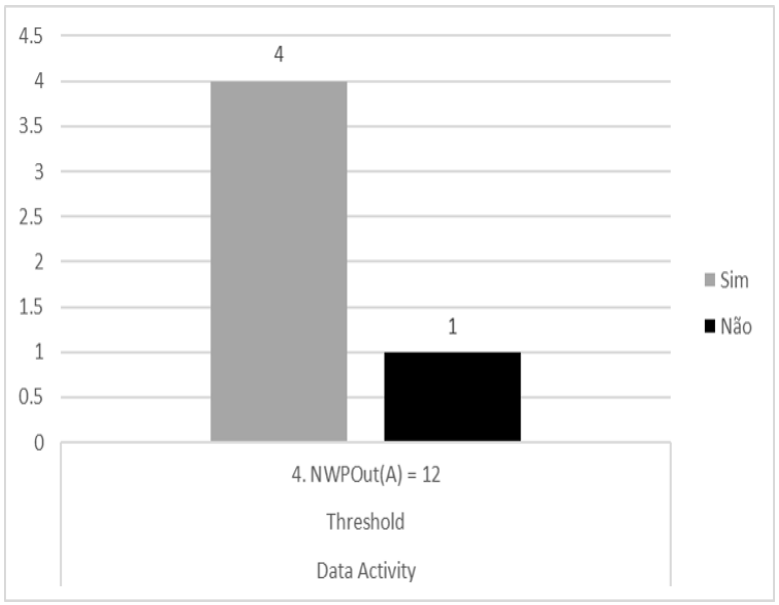


Figura 6.5 Aceitação dos valores limiares da métrica NSTP(A-Tasks) e NWPOut do *Data Activity*.

que sugerem estes resultados são: "caso seja necessário atualizar alguns entregáveis na saída do processo me parece que causará retrabalho de replicação e ou perda de informação caso seja desatento na mudança que está ocorrendo", "No momento da alteração de um dos produtos de trabalho que são entradas de outras tarefas, as tarefas dependentes deverão ser revisadas para garantir se não houve impacto e se vai ser necessário efetuar também alteração na tarefa subsequente" e em "Acredito que a quantidade de produtos de saída dificulta a capacidade de mudança do processo, porque variáveis como impacto e esforço apresentarão valores altos sendo custoso a alteração nessa atividade".

No primeiro comentário o participante faz referência ao "retrabalho" e a "perda de informação". Quanto ao "retrabalho" ele reflete que em um caso de mudança, as possíveis dependências relacionadas aos produtos de trabalho fornecidos pelas atividade *Data Activity* precisarão ser revisitadas. Esta é uma característica deste *process smells* que implica no seu impacto para a modificabilidade. Desta forma existirá um esforço considerável para realizar modificações quando necessário, uma vez que são muitos produtos de trabalhos fornecidos a muitas atividades. Em relação a "perda de informação" é um possível risco de ao acontecer uma mudança nos produtos de trabalhos nem todas as modificações necessárias sejam realizadas. Este fato impacta a modificabilidade, pois uma vez que é difícil de verificar todos os pontos de modificação, alguns destes podem ser esquecidos ou não ser modificados corretamente. Assim o processo pode adquirir inconsistências. A possibilidade de inconsistências no processo e destacada também pelo segundo comentário que destaca a necessidade da revisão das atividades em casos de mudanças de forma a garantir a consistência do processo.

No terceiro comentário o participante reforça a dificuldade de modificabilidade destacando que em virtude da "quantidade de produtos de saída" haverá o "impacto" negativos

Tabela 6.5 Distribuição da Métrica NSTP(A-Tasks).

NSTP(A-Tasks)	NSTPWieghtCum %
1	17,4
2	22,1
3	32,2
4	38,3
6	49,7
7	63,8
11	77,2
16	100,0

para realizar as mudanças, assim como maior "esforço" para realizá-las. Neste comentário são reforçados efeitos dos impactos negativos para a modificabilidade do processo. Apesar destes efeitos não estarem diretamente caracterizados nos impactos do *process smells* são possibilidades bastante condizentes com o *Data Activity*.

B.3 Discussão sobre heurística de detecção e valor limiar do *Data Activity*

Conforme apresentado na seção dos aspectos quantitativos tanto a heurística "1. Possuir poucas tarefas" quanto a "2. Atividade que fornece um grande número de produtos de trabalho de saída" foram aceitas por todos os participantes. Esta primeira heurística foi estabelecida para considerar o fato da atividade *Data Activity* não ser uma atividade relevante no processo e não houve questionamento dos participantes quanto a ela. A segunda heurística também aceita por todos os participantes trata da principal característica do *Data Activity* que é o fato da atividade fornecer um grande número de produtos de trabalho.

Em relação aos valores limiares que suportaram estas heurísticas por sua vez não se mostraram totalmente a contento de todos os participantes. A métrica NPST(A-Tasks) recebeu duas considerações que o valor onze é um valor alto para ser valido como uma atividade com poucas tarefas (tabela 6.5). Um dos participantes indicou que o valor baixo deveria ser caracterizado como menor do que cinco. Já outro participante disse que ajustaria o valor do valor limiar para 50% o que alcançaria conforme a tabela de distribuição da métrica a quantidade de seis ou sete tarefas. O valor onze é o valor limite para o valor limiar da métrica NPST(A-Tasks) o que significa que valores iguais ou abaixo de onze não considerados valores baixo. O único valor que sucede o onze é o valor dezesseis que é o máximo de tarefas contida nas atividades do processo. O onze corresponde ao valor limiar de 77,2% e obteve a concordância dos três outros participantes que não fizeram nenhum comentário contrário a este valor.

Quanto ao valor limiar para a métrica NWPOut(A) foi obtido o valor 12 e este foi aceito pela maior dos participantes, exceto por um deles. Os quatro participantes que aceitaram este valor não fizeram qualquer comentário sobre o valor. O participante que discordou fez a comparação da quantidade de tarefas da atividade avaliada em relação a quantidade de produtos de trabalho fornecido, assim indicou que havia apenas um produto de trabalho a mais do que a quantidade de tarefas da atividade. Por este motivo

não acreditou que havia um grande número de produtos de trabalho nesta atividade. Este participante falou que consideraria a partir de dezesseis produtos de trabalho em uma atividade um valor alto para quantidade de produtos de trabalho de saída. O valor doze obtido na atividade avaliada é o segundo maior valor de produtos de trabalho de saída de uma atividade só sendo inferior a quantidade de vinte e um produtos de trabalho que também se apresentou no processo em questão.

6.6.2.4 *Work Product Clumps*

A. *WORK PRODUCT CLUMPS* - ASPECTOS QUANTITATIVOS

O *Work Product Clumps* recebeu a concordância de quatro participantes quanto ao seu impacto negativo para a modificabilidade enquanto apenas um dos participantes discordou deste fato (figura 6.6). A única heurística para este *process smell* foi definida pela a frequência a qual ocorre a repetição do conjunto de produtos de trabalhos (dois ou mais) como entrada em mais de uma tarefa. Esta heurística obteve a concordância de todos os participantes. Em relação ao valor limiar assim como para a primeira questão quatro participantes concordaram, enquanto um discordou do valor de referência para a métrica de *Frequency of Work Product Clumps* (FRWPC) que detectou duas repetições para o conjunto de produtos de trabalho avaliado (figura 6.7).

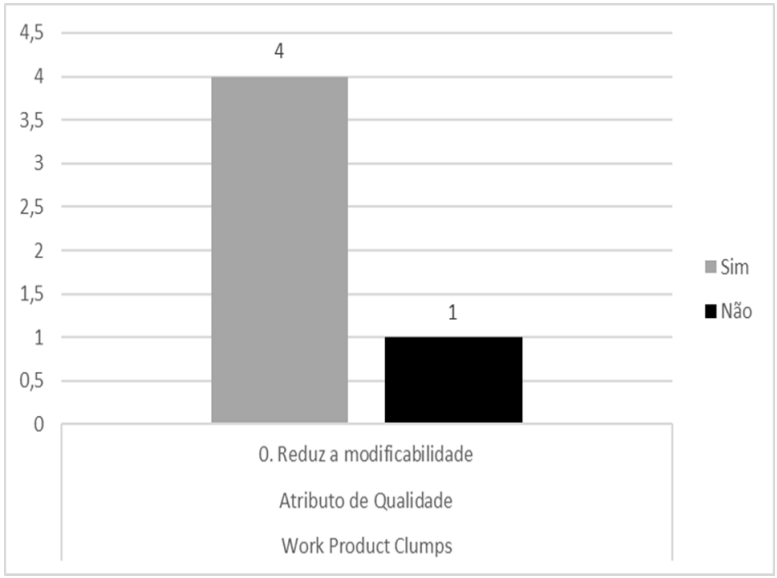


Figura 6.6 Aceitação do impacto a modificabilidade causada pelo *Work Product Clumps*.

B. *WORK PRODUCT CLUMPS* - ASPECTOS QUALITATIVOS

B.1 Discussão da Questão Inicial do *Work Product Clumps*

A seguir a questão inicial do estudo de entrevista em relação ao *Work Product Clumps*

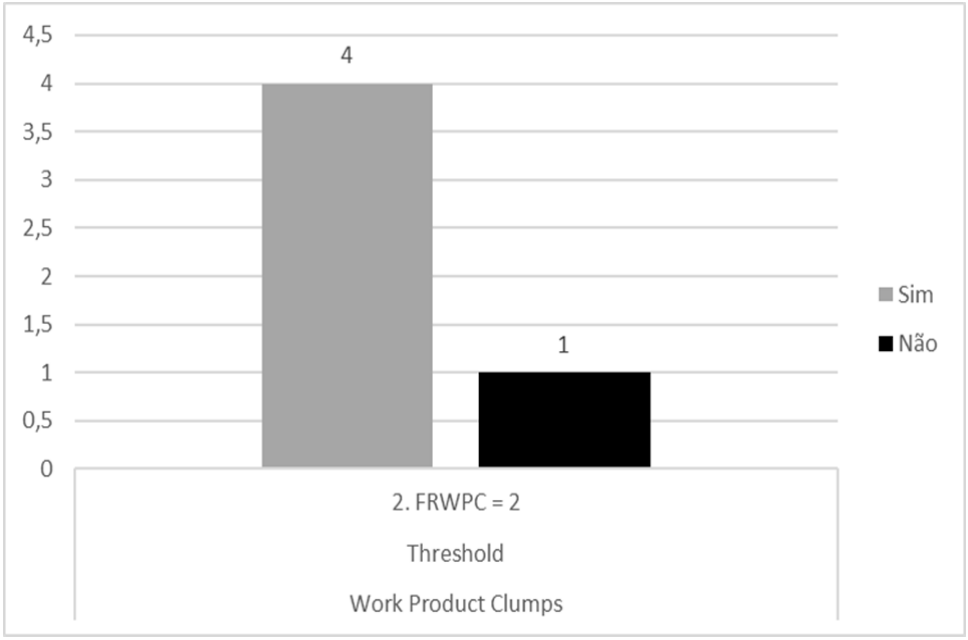


Figura 6.7 Aceitação do valor limiar da métrica FRWPC do *Work Product Clumps*.

que orienta esta primeira parte da análise qualitativa.

> QUESTÃO INICIAL: "PD4 - Analisando os produtos de trabalho "Padrão de desenvolvimento" "Especificação" e "Padrão de Interface" dispostos juntos como produtos de trabalho de entrada nas tarefas "Tarefa 1.1" e "Tarefa 7.1", você acredita que pode eles podem colaborar para redução da modificabilidade do processo?
> Você concorda com a pergunta acima?"

B.2 Respostas que "Confirmam as características da proposta do *Work Product Clumps*"

Quatro participantes concordaram com o impacto negativo para a modificabilidade do processo causado pelo *Work Product Clumps*. As percepções deles foram as seguintes: "Concordo porque a entrada de produtos de trabalho relacionadas às duas tarefas aumentara o esforço realizado caso caso algum produto seja modificado", "No processo fica claro que vai gerar dificuldade para realizar atualização quando necessária nas tarefas mencionadas" e "terá um grande esforço na rastreabilidade desses produtos para uma mudança".

O comentário do primeiro participante retrata o impacto negativo para a modificabilidade quando os mesmo produtos de trabalhos são entradas de duas tarefas e estes precisem ser modificados. O *Work Product Clumps* é um *process smells* característico pelos impactos gerado por aglutinados de produtos de trabalho utilizados em muitas tarefas. Assim a percepção deste participante conseguir alcançar características da definição deste *process smell*. O segundo comentário por sua vez complementa o primeiro, uma vez que dá enfase na dificuldade de atualização das tarefas que tem como entrada produtos de trabalho de um *Work Product Clumps*. Ambos comentário reforçam o esforço

de modificabilidade característico deste *process smell*.

O terceiro comentário utiliza a expressão "esforço de rastreabilidade" e vai de encontro ao impacto proposto no catálogo para o *Work Product Clumps*. Estes "esforço de rastreabilidade" é um impacto do *Work Product Clumps* por que este *process smell* promove uma dispersão dos mesmos produtos de trabalho em diferentes pontos do processo (tarefas) faz necessário que o esforço adicional para localizar (rastrear) estes pontos em caso de mudança. Esta fato trás a possibilidade de inconsistência no processo se todas as mudanças não ocorrerem adequadamente.

B.3 Identificação os motivos da discordância da proposta do *Work Product Clumps*

O único participante que discordou do impacto negativo do do *Work Product Clumps* para o atributo de modificabilidade não apresentou nenhuma justificativa específica. Este participante somente não considera que um aglomerado de produtos de trabalho possa impactar negativo a modificabilidade do processo. Ele compreende como um fato comum ao processo de software a utilização de um mesmo conjunto de produtos de trabalho em mais de uma tarefa do processo.

B.4 Discussão sobre heurística de detecção e valores limiares do *Data Activity*

Para este *process smell* a heurística de verificar a frequência a qual ocorre a repetição do conjunto de produtos de trabalhos (dois ou mais) como entrada em mais de uma tarefa foi aceita por todos os participantes. Neste caso por se tratar de um aglomerado de produtos de trabalhos os participantes consideram adequado a captura da frequência de repetição como uma forma de verificar a ocorrência destes *process smell*.

Quanto ao valor limiar primeiramente foi explicado aos participantes que a definição deste valor limiar foi alcançado a partir de um algoritmo que calculou a frequência da repetição dos produtos de trabalho que eram entradas em comum para mais de uma tarefa. Apresentando ao participantes que esta medida de frequência não poderia ser capturada com a medição de todo o processo, como ocorreu para as tabelas de distribuição de frequência, mas somente para as ocorrências de *Work Product Clumps*, eles compreenderam e não criticaram esta implementação. Assim para o conjunto avaliado a métrica FRWPC resultante do algoritmo identificou duas tarefas que recebiam os mesmo três produtos de trabalho. E quatro participantes concordaram com este valor limiar.

O participante que discordou expôs a seguinte percepção: "O valor 2 representa pouca frequência para gerar muito impacto. Acredito que deveria considerar frequência a partir do valor 10". Este posicionamento em reforça o comentário de discordância quanto a questão inicial do estudo de entrevista. Na visão deste participantes situações recorrentes de processo de software precisariam de uma alta quantidade de repetições para que um impacto negativo fosse percebido. Diferentes dos demais valores limiares o valor dois para a métrica FRWPC não foi obtido a partir de uma escala, assim é compreensível que o participante utilize a sua experiência para determinar valores que ele considere altos e baixos.

6.6.2.5 *Long Input List*

A. *LONG INPUT LIST* - ASPECTOS QUANTITATIVOS

O *Long Input List* foi avaliado por dois participantes como um *process smell* que apoia a redução da compreensibilidade do processo enquanto os três outros participantes não concordaram com esta possibilidade (figura 6.8). Já quanto a única heurística que é definida por "Uma tarefa que precisa de um grande número de produtos de trabalho de entrada" mensurada pelo métrica Number of Work Product Input NWPIn(A) e quanto ao valor limiar desta métrica com valor cinco, todos os participantes concordaram.

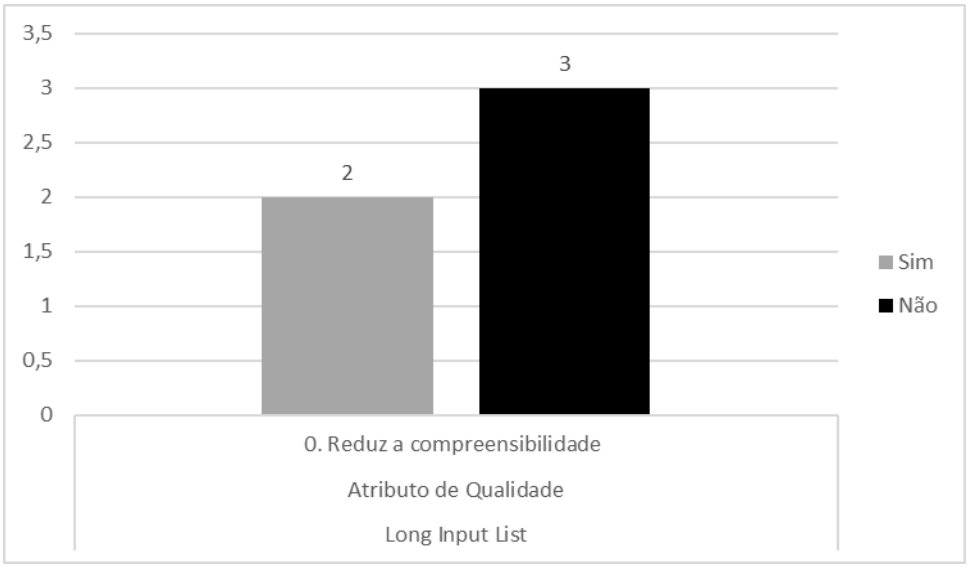


Figura 6.8 Aceitação do impacto a compreensibilidade causada pelo *Long Input List*.

B. *LONG INPUT LIST* - ASPECTOS QUALITATIVOS

B.1 Discussão da Questão Inicial do *Long Input List*

A seguir a questão inicial do estudo de entrevista em relação ao *Long Input List* que orienta esta primeira parte da análise qualitativa.

> QUESTÃO INICIAL**:** "PD3 - Analisando a estrutura da "Tarefa 6.2" quanto a quantidade de produtos de trabalho de entrada, você acredita que a Tarefa 6.2 pode colaborar para redução da compreensibilidade do processo? Você concorda com a pergunta acima?"

B.2 Respostas que "Confirmam as características da proposta do *Long Input List*"

Como citado na seção dos aspectos quantitativos o *Long Input List* recebeu a concordância de três participantes. Eles apresentam entre seus posicionamentos as seguintes

frases: “No momento que são utilizados os inputs como norteadores para o desenvolvimento da tarefa pode colaborar para reduzir a compreensão” e “A tarefa possui uma grande quantidade de produtos de entrada que podem reduzir a compreensibilidade do processo”.

A percepção dos participantes que concordaram com a redução da compreensibilidade do processo causada pelo *Long Input List* está associada ao fato do de existir um alto volume de informações a serem analisada para executar a tarefa. Em ambos os casos os participantes fazem referências a grande quantidade de produtos de trabalho de entrada das tarefas, principal característica que faz o *Long Input List* proporcionar impacto negativo para a compreensibilidade.

B.3 Respostas que “Adiciona, complementa, características diferentes da proposta original do *Long Input List*”

Mesmo concordando com o impacto do *Long Input List* um dos participantes fez uma ressalva sobre o fato de quanto maior for a experiência do profissional que venha a realizar a tarefa não haverá tanto impacto negativo para a compreensibilidade mesmo que exista um grande número de produto de trabalho de entrada. Esta característica não foi cogitada na elaboração do catálogo, pois o objetivo é avaliar o processo a partir da medição dos seus elementos estruturais para identificar os *process smells*.

B.4 Identificação os motivos da discordância da proposta do *Long Input List*

Os três participantes que não concordaram com a redução da compreensibilidade do processo causada pelo *Long Input List* sinalizaram o fato de que se os produtos de trabalho convergem para a execução da tarefa não haverá impacto negativo. Estes são trechos destacam esta opinião: “Apesar da quantidade de produtos de trabalho de entrada todos são relevantes para a execução da atividade e continuidade do processo.” e “Não atrapalha a compreensão se os produtos de trabalho de entrada são relativos”. O fato dos produtos de trabalho de entrada terem relevância e relação entre si não foi capturado durante a elaboração do catálogo. Como o foco do catálogo está na medição dos elementos estruturais do processo, aspectos semânticos ou categóricos que pudessem distinguir se os produtos de trabalho tem a mesma finalidade não foram verificados. Para este cenário seria necessário o uso de outras técnicas de diferem da detecção de *bad smells* de código utilizada como base para a elaboração desta pesquisa.

Um dos participantes ressaltou que deveria ser verificada a qualidade dos produtos de trabalho de entrada, pois artefatos com baixa qualidade ou incompletos sim ocasionam dificuldade para compreensão e execução das tarefas. Assim como no cenário anterior este tipo de avaliação não fez parte do escopo desta pesquisa. Para tanto seriam necessário o uso de técnicas de verificação de corretude, completude ou outras técnicas de avaliação da qualidade dos produtos de trabalho do processo.

B.5 Discussão sobre heurística de detecção e valores limiares do *Long Input List*

Em relação a heurística “Uma tarefa que precisa de um grande número de produtos

de trabalho de entrada" todos participantes concordaram. Um dos participantes fez um comentário reafirmando a sua concordância com a heurística. Ele sinalizou que a tarefa com *Long Input List* pode sofrer o acúmulo de responsabilidades uma vez que possui um grande número de produtos de trabalho que podem se destinar a diferentes finalidades. A partir desta observação o participante acabou reforçando um dos impactos do *Long Input List* definidos no catálogo. O fato de ter muitos produtos de trabalho de entrada pode acontece de diferentes fatores e um destes pode ser atribuir mais de uma responsabilidades para uma mesma tarefa e desta forma os produtos de trabalho entrantes precisam ser aumentados a cada responsabilidade a mais que a tarefa tenha que atender.

Em relação ao valor limiar alcançado pela métrica *Number of Work Product Input NWPIn(A)* que verificou o valor de cinco produtos de trabalho de entrada para a tarefa avaliada, todos os participantes concordaram com este valor. Durante a validação dos valores limiares um participante fez uma consideração que não havia sido avaliada durante a elaboração do catálogo. Este participante falou que em relação a especificação SPEM no que tange aos tipos de produtos de trabalho, os "artefatos" como entradas de uma tarefa são mais significativos do que os "resultados" (outcomes). Uma vez que "resultados" não são tangíveis, assim uma vez compreendido uma determinada informação, padrão ou outras características intangíveis, está não precisará de uma nova interpretação ou aquisição pelo executante da tarefa, apesar de ser um insumo para a execução correta da tarefa. Esta possibilidade não foi avaliada anteriormente, pois leva em considerações aspectos que extrapolam os elementos estruturais do processo em SPEM. Esta sugestão envolve aspectos humanos como capacidade de retenção de informação, corretude da informação retida, entre outras aspectos.

6.6.2.6 *Feature Envy*

A. *FEATURE ENVY* - ASPECTOS QUANTITATIVOS

O *Feature Envy* recebeu uma avaliação similar ao *Long Input List* onde dois participantes concordaram com o impacto negativo para a compreensibilidade do processo enquanto os outros três não concordaram com esta possibilidade (figura 6.9). Não houve discordância quanto as heurísticas "Possui mais que 1/3 do uso de produtos de trabalho de outras atividades", "Possui poucas dependências predecessoras" e "Possui dependência de poucas atividades predecessoras". Nem mesmo houve discordância dos valores limiares que representam as heurísticas e que são medidas respectivamente pelas métricas *External Predecessor 1/3* (EP1/3(T)) e *Number of predecessor dependent* NPD(T) e *External Predecessor* EP(T).

A métrica EP1/3(T) verifica se mais de um terço da quantidade produtos de trabalhos utilizados na tarefa são externos a atividade a qual ela pertence. Quanto a tarefa avaliada o valor limiar para essa métrica obteve valor positivo pelo fato de existir mais de um terço dos produtos de trabalhos usado pela tarefa sendo originados de outras atividades a qual ela não pertencia. Este fato pôde ser facilmente verificado pelos participantes, assim não

houve discordâncias.

A métrica NPD(T) identificou apenas uma tarefa que fornecia produtos de trabalho para a tarefa avaliada. Desta forma também foi aceita pela totalidade dos participantes, pois o valor um é o valor mais baixo possível para este valor limiar. Por fim a métrica EP(T) assim como a NPD(T) identificou o valor um o valor mais baixo possível para este valor limiar, assim não houve discordância.

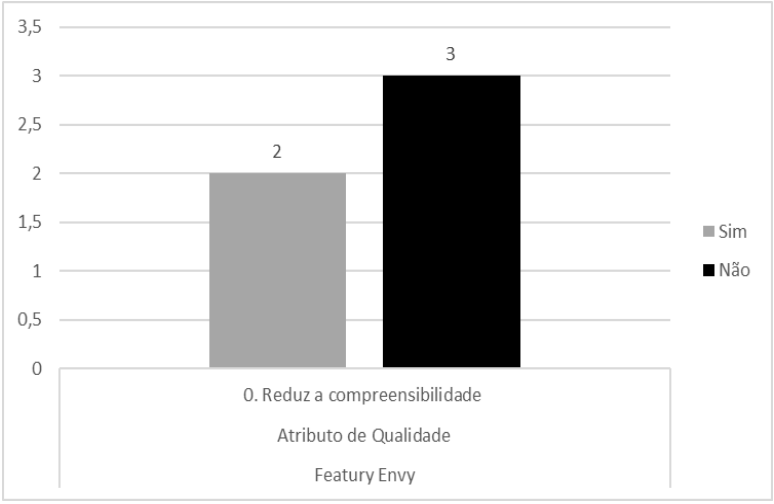


Figura 6.9 Aceitação do impacto a compreensibilidade causada pelo *Feature Envy*.

B. *FEATURE ENVY* - ASPECTOS QUALITATIVOS

B.1 Discussão da Questão Inicial do *Feature Envy*

A seguir a questão inicial do estudo de entrevista em relação ao *Feature Envy* que orienta esta primeira parte da análise qualitativa.

> QUESTÃO INICIAL: "PD2 - Analisando a estrutura da "Tarefa 2.1" quanto a forma com que ela faz uso de produtos de trabalho, você acredita que a Tarefa 2.1 pode colaborar para redução da compreensibilidade do processo?
> Você concorda com a pergunta acima?"

B.2 Respostas que "Confirmam as características da proposta do *Feature Envy*"

Os dois participantes que concordaram com o *Feature Envy* apresentaram os seguintes comentários: e "A distância entre um produto de trabalho com relação a tarefa pode dificultar a compreensão no processo" e "O fluxo do processo demonstra as dependências e relações do processo de codificação que são acessadas durante as realização das atividades desenhada no fluxo apresentado. Por tanto pode atrapalhar na compreensão"

O primeiro comentário apresenta a ideia da distância entre o produto de trabalho e a tarefa que irá utilizá-lo. Este comentário reflete um dos impactos propostos no catálogo para o *Feature Envy*. Este *process smell* se refere a tarefas que invejam dados de tarefas de outras atividades, assim os produtos de trabalho entrada veêm de tarefas de outras

atividades proporciona a distância entre a origem e o destino. Esta distância aumenta o esforço para a compreensão da tarefa uma vez que o usuário do processo precisará acessar outras atividades do processo para compreender a tarefa por completo com seus produtos de trabalho de entrada. O segundo comentário trata esta relação entre a distância da origem dos produtos de trabalho de entrada e as tarefas com os termos "dependências e relações" e ao final também confirma sua percepção de impacto para a compreensibilidade.

B.3 Identificação os motivos da discordância da proposta do *Feature Envy*

Os três participantes que discordaram que ocorreu uma redução da compreensibilidade a partir da tarefa detectada com *Feature Envy*. Para tanto eles apresentaram os seguintes comentários: "Discordo, pois a origem do documento é de uma tarefa correlata a tarefa que utiliza o documento", "De acordo com meu entendimento, o desenvolvimento de software é um processo dinâmico e as relações entre os componentes é esperada" e "Discordo porque o fluxo que compõem a tarefa é muito simples e compreensivo".

Em relação ao primeiro comentário o participante cita "documento" se referindo ao produto de trabalho que promove o acoplamento entre as duas tarefas em questão. Quando faz referência a "tarefa correlata" o participante evidenciou que as tarefas apesar de estarem em atividades diferentes procuravam atender uma mesma necessidade do processo. Assim para ele o fato da tarefa avaliada fazer uso de mais de um terço de produtos de trabalho de uma tarefa de outra atividade não é um *process smell* desde que as tarefas desempenhem uma mesma finalidade. Esta mesma opinião pode ser assumida para o segundo comentário que enfatiza que "as relações entre os componentes é esperada". Componentes neste caso foi a expressão usada para sinalizar os elementos do SPEM, especificamente as tarefas observadas.

O último comentário destaca que ao avaliar a tarefa e as relações com o produto de trabalho de entrada não identificou de que forma a compreensibilidade pudesse ser afetada negativamente, visto que ele compreendeu com clareza a tarefa e as suas relações. Neste caso o participante não forneceu mais informações sobre a sua percepção.

Revisando os comentários um e dois é expresso pela percepção dos participantes que o *Feature Envy* é afetado pelo grau com que a tarefas acopladas procuram atender uma mesma necessidade do processo. O fato deles perceberem que apesar de estarem em atividades diferentes era necessário que a tarefa avaliada obtivesse mais de um terço de produto de trabalho de uma tarefa de outra atividade, pois ela atenderia a uma necessidade similar à da tarefa à qual lhe forneceu os produtos de trabalho de entrada. Um cenário como este não foi avaliado no escopo do catálogo, pois não lida com a medição de elementos estruturais do SPEM. Possivelmente técnicas de avalição semântica ou de categorização pudessem tratar estes cenários, contudo esta avaliação não fez parte do escopo da pesquisa.

B.4 Discussão sobre heurística de detecção e valores limiares do *Feature Envy*

As heurísticas definidas assim como os valores limiares foram totalmente aceitos no que diz respeito a caracterização e detecção do *Feature Envy*. Desta forma foi possível verificar que a definição do *Feature Envy* conseguiu ser refletidas pelas suas heurísticas e

valores limiares, apesar que inicialmente compreender essas heurísticas de forma conjunta se mostrou mais difícil.

Com exceção da heurística "Possui mais que 1/3 do uso de produtos de trabalho de outras atividades" ou outras "Possui poucas dependências predecessoras" e "Possui dependência de poucas atividades predecessoras" são um tanto semelhantes. Essa semelhança necessitou de mais detalhamento, orientação durante as entrevistas, assim como foram mais avaliadas pelos participantes. A primeira heurística foi fácil de ser verificada, visto que a tarefa avaliada era a única da atividade e não possuía mais que três produtos de trabalho de entrada vindo de outra atividade.

Quanto as duas heurísticas que faltam algumas explicações foram dadas aos participantes. A heurística "Possui poucas dependências predecessoras" foi explicado que deveria ser observada a quantidade de tarefas cujo a tarefa observada possui dependência. Esta dependência se refere a receber produtos de trabalho. O que se difere na heurística "Possui dependência de poucas atividades predecessoras" é que se leva em consideração apenas as tarefas dependentes de outras atividades diferente da atividade a qual a tarefa observada pertence.

Estas heurísticas constituem o comportamento do *Feature Envy* da seguinte forma:

1. "Possui poucas dependências predecessoras": Esta heurística indica que a tarefa por ter poucas dependências possivelmente necessite de produto de trabalho de entrada com origem mais concentrada em certa parte do processo.

2. "Possui mais que 1/3 do uso de produtos de trabalho de outras atividades": Como vimos na heurística anterior a tarefa se concentra em dados de certa parte do processo, neste momento esta segunda heurística verifica se os dados que a tarefa necessita se concentram mais em outra atividade da qual ela não pertence. Este fato começa a caracterizar a "inveja" por buscar mais dados em outras atividades.

3. e "Possui dependência de poucas atividades predecessoras" por fim nesta heurística é verificado se existe a busca de dados em atividades predecessoras mais específicas, visto que se observa se existe dependência de poucas atividades predecessoras. Esta verificação sustenta que a "inveja" está direcionada de fato a uma certa parte do processo que não é a parte em que a tarefa se encontra.

Explicado tudo isso a validação seguia com mais fluidez e por sua vez resultou como já apresentado antes na aceitação da totalidade dos participantes quanto as heurísticas e valores limiares deste *process smell*.

6.6.2.7 *Brain Activity*

A. *BRAIN ACTIVITY* - ASPECTOS QUANTITATIVOS

Na validação do *Brain Activity* dois participantes concordaram enquanto três discordaram sobre a possibilidade de redução da compreensibilidade do processo causada por

esse *process smell* (figura 6.10). As heurísticas "Possui muitas tarefas", "Atividade pouco coesa" e "Possui muitos fluxos de decisão" foram todas aceitas por todos os participantes. E quanto aos valores limiares que identificam as respectivas heurísticas, a métrica número de tarefas (*Number of Steps NSTP*(A-Tasks)) identificou onze tarefas para a atividade avaliada, este valor limiar recebeu quatro concordância e uma discordância (figura 6.11). Já a métrica taxa de coesão das tarefas da atividade (*Ratio of Cohesion Tasks* (RCT(A)) identificou o valor de 1,82 referente a baixa coesão da atividade. Este valor limiar foi aceita pelos cinco participantes. E por fim a métrica número de fluxos de decisão (*Number of Gateway* (NG(A)) identificou três fluxos de decisão, recebeu três concordâncias e duas discordâncias (figura 6.12).

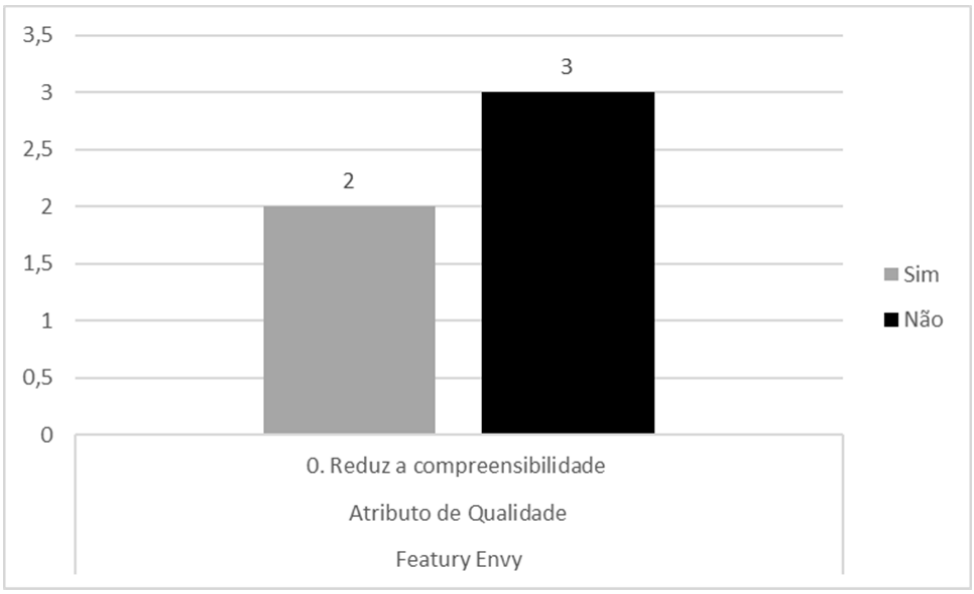


Figura 6.10 Aceitação do impacto a compreensibilidade causada pelo *Brain Activity*.

B. *BRAIN ACTIVITY* - ASPECTOS QUALITATIVOS

B.1 Discussão da Questão Inicial do *Brain Activity*

A seguir a questão inicial do estudo de entrevista em relação ao *Brain Activity* que orienta esta primeira parte da análise qualitativa.

> QUESTÃO INICIAL: "PT4 - Analisando a estrutura da "Atividade 4" em relação a quantidade de fluxos de decisão e de tarefas e quanto as relações estabelecidas entre as suas tarefas você acredita que a Atividade 4 pode colaborar para redução da compreensibilidade do processo?
> Você concorda com a pergunta acima?"

B.2 Respostas que "Confirmam as características da proposta do *Brain Activity*"

Apenas dois participantes que concordaram com *Brain Activity* apoia a redução da compreensibilidade do processo. Os seguintes comentários atestam sua confirmações:

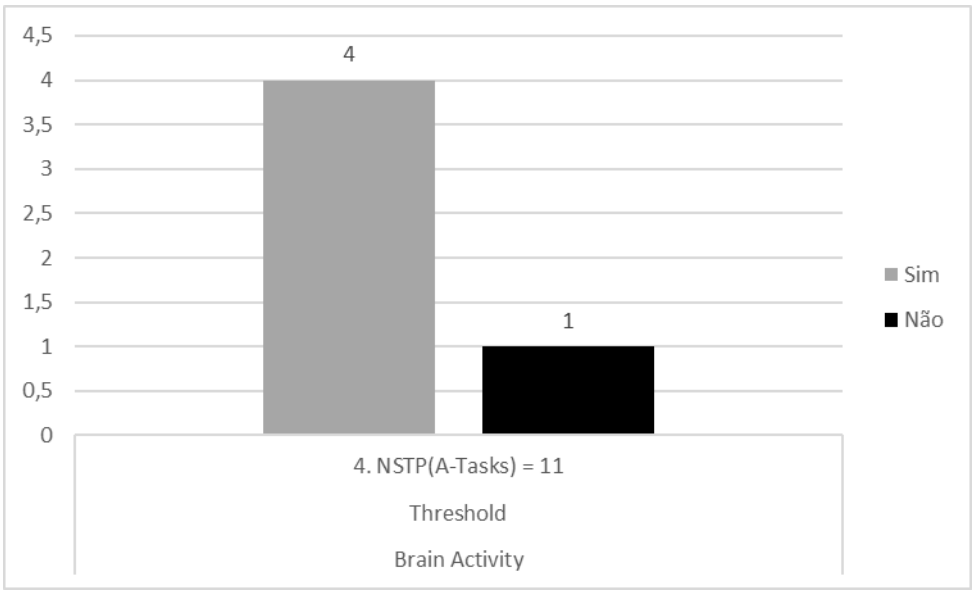


Figura 6.11 Aceitação do valor limiar da métrica NSTP(A-Tasks) do *Brain Activity*.

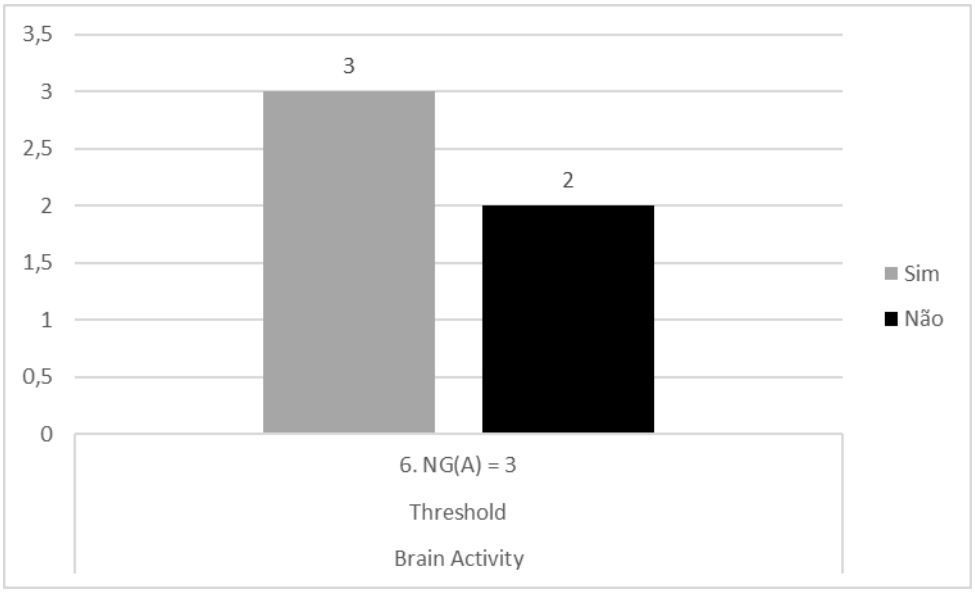


Figura 6.12 Aceitação do valor limiar da métrica NG(A) do *Brain Activity*.

"Avaliando a quantidade, muitas tarefas tornam essa atividade grande e complexas e podem reduzir a compreensibilidade" e "...porque a quantidade de fluxos de decisões e de tarefas que compõem a "Atividade 4" dificulta a processo".

Em ambos os comentários os destaques indicados foram quanto a grande quantidade de tarefas e fluxos de decisão na atividade avaliada. Estas são duas das características principais do *Brain Activity* descritas no catálogo. Contudo as relações estabelecidas entre as tarefas não foi citada nos comentários, o que sugere que foram menos percebidas pelos participantes neste momento.

B.3 Respostas que "Adiciona, complementa, características diferentes da proposta original do *Brain Activity*"

Apesar da aceitação do impacto negativo para a compreensibilidade uma ressalva foi sinalizada por um dos participantes. O participante fez referencia sobre a experiência dos profissionais indicando que quanto maior a experiência menor deve ser o impacto percebido quando a atividade possuir muitas tarefas e muitos fluxos. Como já citado anteriormente este aspecto não foi contemplado no escopo da pesquisa, pois extrapola a proposta que é medir as subpropriedades estruturais do SPEM para apoiar a detecção dos *process smells*.

B.4 Identificação os motivos da discordância da proposta do *Brain Activity*

Três participantes discordaram que a atividade avaliada proporcionou algum impacto negativo para a compreensibilidade. A seguir alguns trechos expostos pelos participantes: "Acredito que o fluxo está compreensível", "A compreensão da atividade não é impactada" e "De acordo com a análise da atividade 4, verifiquei que, neste caso, as tarefas estão bem subdivididas entre os respectivos executantes e, portanto, não houve sobrecarga / impacto na compreensão do papel de cada um dentro da atividade como todo".

Os dois primeiros comentários são bem similares diferentes do terceiro comentário que é mais específico. Os dois primeiros reportam não perceber impactos negativos na compreensibilidade da atividade visto que compreenderam de forma clara o fluxo da atividade avaliada. O terceiro comentário retrata que a distribuição das tarefas da atividade entre os diferentes papéis executores favorece uma boa compreensibilidade da atividade. Neste caso o participante durante a entrevista questionou a organização e agrupamentos das tarefas para compreender por que haviam diferentes fluxos mas ao final da atividade apenas uma única tarefa integrava os fluxos existentes. Ao compreender que cada fluxo era destinado a um papel envolvido na atividade o participante conclui que o fluxo era compreensível visto que nem todos os papeis fariam todas as tarefas.

B.5 Discussão sobre heurística de detecção e valores limiares do *Brain Activity*

As heurísticas deste *process smell* foram todas aceitas pelos participantes. As heurísticas do *Brain Activity* que são "Possui muitas tarefas", "Atividade pouco coesa" e "Possui muitos fluxos de decisão" procuram concluir o que caracterizaria uma atividade centralizada e que atende a maior parte das ações relevantes para o processo. A atividade para atender a maior parte das ações relevantes do processo terá muitas tarefas para alcançar

os resultados necessários, assim como terá muitos fluxos para organizar as tarefas e ainda possuirá baixa coesão visto as diferentes ações que ela atende não precisarão interagir entre si.

Um dos participantes propôs que mais uma heurística fosse adicionada para a detecção do *Brain Activity*. A sugestão aponta para a avaliação dos papeis envolvidos em uma atividade verificando se a distribuição destes nas tarefas favorece ou não o impacto negativo na compreensibilidade. Este é o comentário apresentado pelo participante: "Com relação às heurísticas utilizadas, acrescentaria a avaliação da distribuição das tarefas para os executantes, ou seja, se uma atividade possui muitas tarefas, mas estão bem distribuídas para os seus respectivos executantes o fator compreensibilidade não seria tão impactado em comparação à muitas tarefas executadas somente por um dos papéis existentes no processo".

O comentário anterior faz observação para uma nova característica possível de ser acrescentada para a detecção do *Brain Activity*. A primeira referência sobre uma possível avaliação da quantidade de papéis para o *Brain Activity* foi feita por um dos especialistas em processo de software no primeiro estudo de entrevista. Esta avaliação não foi incorporada na pesquisa para que pudesse ser feita a validação do formato inicial de detecção do *Brain Activity* e com base nos resultados obtidos avaliar a elaboração e incorporação de uma métrica para medir essa característica em trabalhos futuros.

Quanto aos valores limiares a métrica NSTP(A-Tasks) indicou o valor de onze tarefas em uma mesma atividade como um valor alto. Este valor recebeu a concordância de quatro participantes. O participante que discordou considerou o valor 11 baixo e considerou que a partir de dezesseis tarefas ele acreditaria ser um valor alto. Isto sendo que dezesseis foi o valor máximo de tarefas em uma atividade nos processos avaliados. Usar valores máximos normalmente não é uma opção muito utilizada por estratégias que aplicam parâmetros estatísticos para definição de valores limiares. Os valores máximos costuma ser restritivos além de possivelmente serem valores fora dos limites das métricas verificadas (outliers). A estratégia aplicada foi configurar para manter o valor limiar alto em valores iguais ou logo inferiores ao valor de 90% para as métricas utilizadas. Assim o valor dezesseis não foi um valor que satisfizesse esta regra, pois o mesma alcançou 100% da tabela de distribuição de frequência para a métrica NSTP(A-Tasks).

A métrica RCT(A) detectou o valor limiar 1,82 para a coesão da atividade avaliada o que foi totalmente aceito como um valor de baixa pelos participantes. Para avaliar a baixa coesão, característica que não havia recebido destaque na questão inicial, todos os participantes navegaram na atividade em busca de quantas trocas de produtos de trabalho aconteciam entre as tarefas da atividade avaliada, ao perceber que eram poucas em relação a quantidade de tarefas aceitaram tranquilamente esse valor limiar.

E a métrica NG(A), onde o número de fluxos de decisão da atividade foi três, recebeu a concordância de três participantes. Dos dois participantes que discordaram, um considerou que o valor a ser considerado como um alto número de fluxo de decisão deveria ser acima de três fluxos de decisão, enquanto o outro acreditava que deveria ser a partir do valor cinco ou seis. Apesar do opinião destes participantes o valor três representou o maior valor de fluxos de decisão nas atividades dos processos avaliados. Assim para esta pesquisa não seria possível aplicar qualquer ajuste na estratégia de definição deste valor

limiar utilizado para contemplar este valores sugeridos pelos participantes uma vez que superavam os valores disponíveis na especificação do processo.

6.6.2.8 *Brain Task*

A. *BRAIN TASK* - ASPECTOS QUANTITATIVOS

A validação do *Brain Task* concentrou a maior quantidade de pontos os quais os participantes discordaram. No que diz respeito a redução da compreensibilidade do processo apenas dois participantes concordaram e três discordaram. Em relação as heurísticas "Possui muitos passos" e "Possui muitos fluxos de decisão" todos os participantes concordaram com ambas. E quanto aos dois valores limiares deste *process smell*, para ambos, três participantes concordaram e dois discordaram (figura 6.13). Os valores limiares foram medidos pelas métricas número de passos da tarefa (*Number of steps* (NSTP(T)) (figura 6.14) e número de fluxos de decisão (*Number of Gateways* (NG(T)) (figura 6.15) que detectaram respectivamente quatro passos e três de fluxos de decisão para a tarefa avaliada.

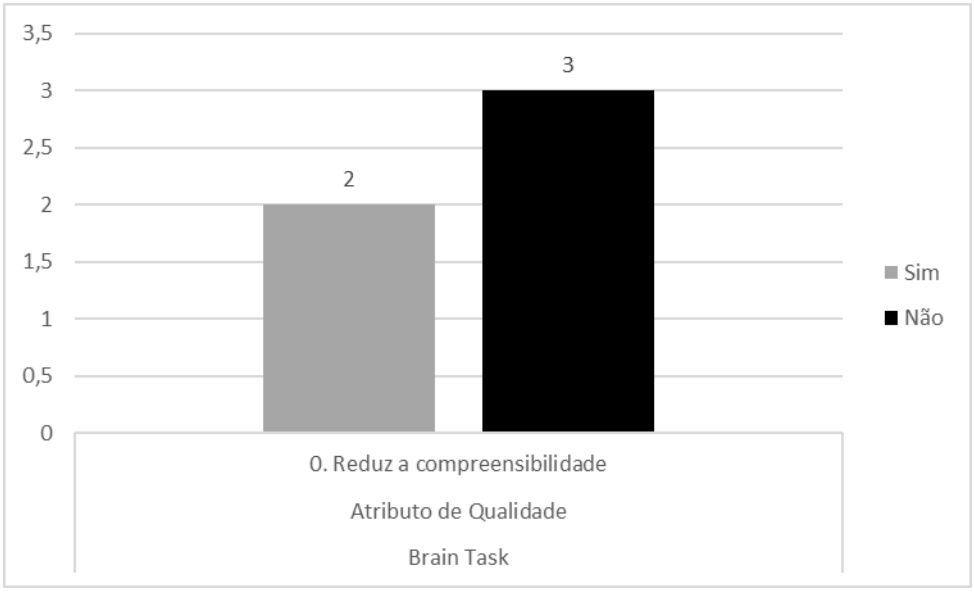


Figura 6.13 Aceitação do impacto a compreensibilidade causada pelo *Brain Task*.

B. *BRAIN TASK* - ASPECTOS QUALITATIVOS

B.1 Discussão da Questão Inicial do *Brain Task*

A seguir a questão inicial do estudo de entrevista em relação ao *Brain Task* que orienta esta primeira parte da análise qualitativa.

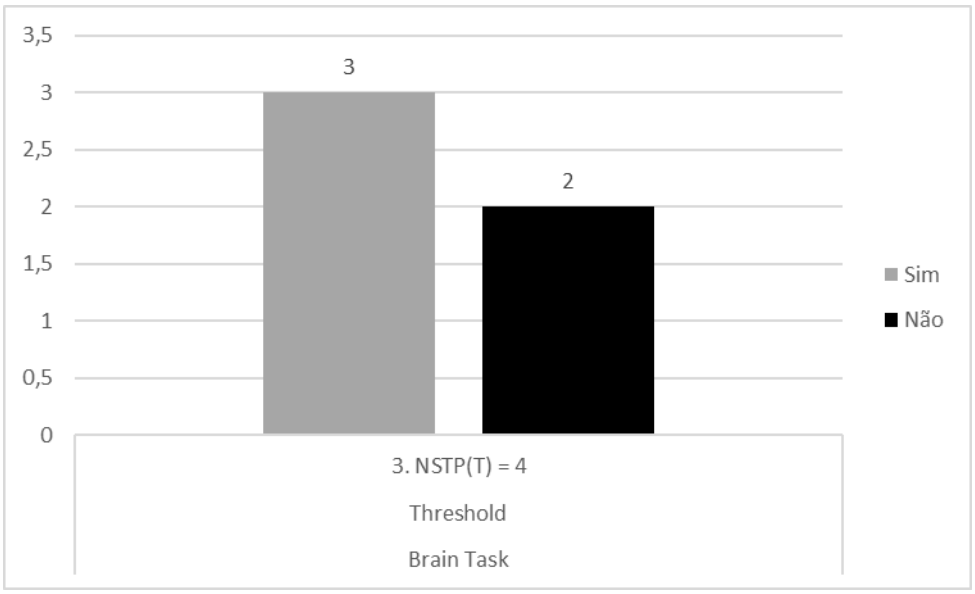


Figura 6.14 Aceitação do valor limiar da métrica NSTP(T) do *Brain Task*.

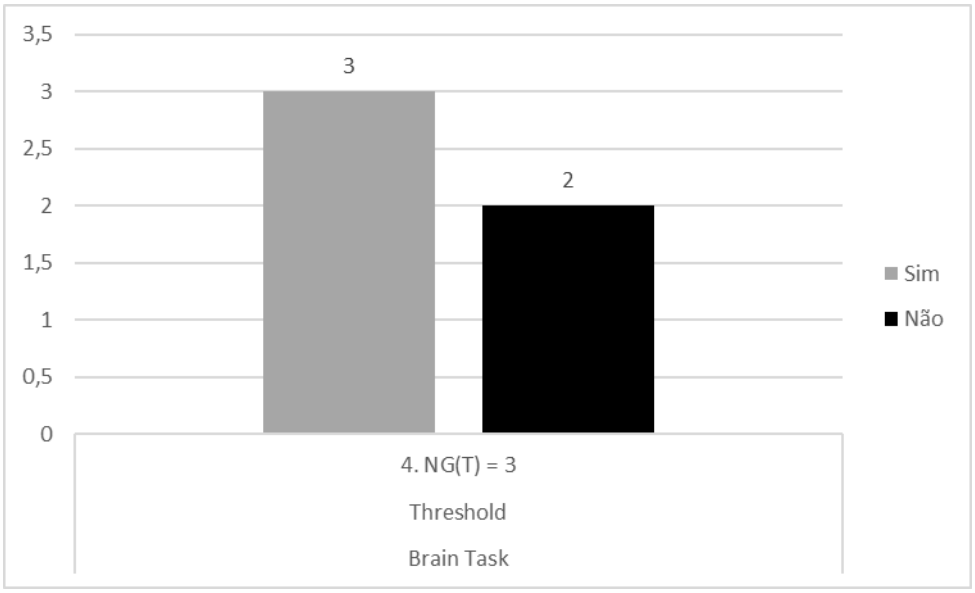


Figura 6.15 Aceitação do valor limiar da métrica NG(T) do *Brain Task*.

QUESTÃO INICIAL: "PT2 - Analisando a estrutura da "Tarefa 3.3" em relação a quantidade de fluxos de decisão e passos, você acredita que a Tarefa 3.3 pode colaborar para redução da compreensibilidade do processo?
Você concorda com a pergunta acima?"

B.2 Respostas que "Confirmam as características da proposta do *Brain Task*"

Sobre a redução da compreensibilidade do processo os dois participantes concordaram com esse impacto negativo. Seus comentário foram: "por ter muitas interações impacta negativamente na realização do processo" e "por ter muitos fluxos de decisão e muitos passos os executantes podem deixar de fazer algum ação importante". Ambos participantes compararam a tarefa avaliadas com outras tarefas do processo e ao perceber que está possuía mais passos e mais fluxos de decisão do que as demais expressaram que esta tarefa poderia não ser realizada corretamente em virtude do maior esforço para compreensão da mesma. Estes comentários afirma duas das principais características do *Brain Task* que são possuir muitos passos e muitos fluxos de decisão. Desta forma a quantidade de fluxos de decisão e de passos pode impactar em execuções equivocadas pelos participantes. Apesar dos impactos deste *process smell* não estarem descritos desta forma no catálogo a execução falha é um resultado inerente a problemas de compreensão de uma tarefa a ser executada.

B.3 Identificação os motivos da discordância da proposta do *Brain Task*

Três participantes discordaram por não verificarem dificuldades para a compreensão da tarefa avaliada. Seus comentários foram: "Não verifico impacto na compreensibilidade da tarefa verificada", "Discordo porque os fluxos de decisão possuem características simples e fáceis de compreender os passos que serão realizados" e "Discordo, pois, no caso identificado (tarefa 3.3) os fluxos de decisão tratam-se mais de pré-requisitos para a realização da tarefa, do que fluxos encadeados que deverão ser executados ao longo da tarefa". Os três participantes nos seus comentários indicaram principalmente que os fluxos de decisão não comprometiam a compreensão da tarefa. Para este validação nenhum comentário enfatizou que a quantidade de passos da tarefa fosse um fator que impactasse a compreensibilidade.

B.4 Discussão sobre heurística de detecção e valores limiares do *Brain Task*

As heurísticas "Possui muitos passos" e "Possui muitos fluxos de decisão" desse *process smell* assim como a de outros foram totalmente aceitas pelos participantes. Essas duas heurísticas tentam configurar uma tarefa centralizadora na atividade. Sendo assim possuir muitos passos e muitos fluxos de decisão são características de tarefas que assumem o papel de entregar os principais resultados de uma atividade. Esta caracterização foi bem aceita até mesmo pelos participantes que não acreditaram que a tarefa detectada como um *Brain Task* realmente gerasse impactos negativos para a compreensibilidade do processo.

Quanto aos valores limiares medidos pelas métricas NSTP(T) e NG(T) sendo respectivamente quatro passos e três de fluxos de decisão, receberam igualmente a a concordância

de três participantes e discordância de dois participantes. Estes valores limiares puderem ser facilmente verificados pelos participantes e ambos se caracterizaram como os maiores valores de passos e de fluxos de decisão para as tarefas dos processos avaliados. Assim os três participantes que concordaram não fizeram nenhum comentário. Os dois participantes que discordaram destes valores acreditam que estes valores respectivamente, não eram valores que pudessem ser considerados altos. Porém não souberam indicar valores que poderiam ser ideais. Conforme a estratégia de definição dos valores limiares utilizada não é possível estabelecer valores limiares além dos valores limites dos processos avaliados, assim não seria possível neste caso fazer ajustes a estratégia de definição dos valores limiares para estabelecer valores maiores dos que os que foram alcançados para definir os valores limiares deste *process smell*.

6.7 AMEAÇAS À VALIDADE

Quanto as ameaças a validade podem ser destacados quatro aspectos. O primeiro se trata fato do estudo de entrevista ser longo. Esta característica pode ter comprometido a avaliação dos profissionais devido ao cansaço dado ao longo tempo de execução que duraram em média entre 3 a 4 horas. O segundo aspecto está relacionada a seleção dos *process smells* feita pelo analista de processo onde a percepção de atividade crítica pode ter impactado em obter cenários cujo o *process smell* melhor se caracterizaria no processo, se tornando mais evidente e mais fácil de ser validado. Contudo, o critério de seleção foi bem aceito pelos entrevistados, assim como emprega o esforço de validação ao elementos que mais colaboram para um bom resultado do processo. Ainda sobre este ponto o que não foi avaliado é se a percepção de elementos críticos do processo vinda dos analistas de processo se aproxima da percepção dos entrevistados.

O terceiro aspecto lida com o conhecimento técnico associado para validação manual, como reconhecer uma situação de acoplamento entre tarefas e atividades, a coesão entre tarefas, entre outros. Estes conhecimentos se mostram intrínsecos e para esta pesquisa foi necessário compartilhá-los durante a entrevista o que não é trivial quando já não se tem estes hábito ou prática. E por fim o uso da escala dos valores limiares acompanhando o trabalho proposto por Alves et al. (2010) mesmo levando em consideração a ausência de uma distribuição não normal não se mostrou familiar ao conhecimento dos participantes e em alguns casos os valores alcançados não pareceram estar a contento. Este fato pode ter influenciado algumas das respostas.

6.8 DISCUSSÃO

Concluído o estudo conseguimos verificar que todos os *process smells* receberam certa eficácia (grau de aceitação) e nenhum deixou de eficaz, mesmo que parcialmente (tabela 6.6). Os quatro *process smells* mais críticos com maior significância receberam 100% de aceitação e um único com um valor um pouco menor com 80%. Foram estes respectivamente *Shotgun Surgery*, *Divergent Change*, *Data Activity* e *Work Product Clumps*. Os demais, menos significativos, receberam 40% de aceitação sendo *Long Input List*, Featury

Tabela 6.6 Eficácia dos *Process Smells*.

Process Smell	Eficiência (%)
Shotgun Surgery	100
Divergent Change	100
Data Activity	100
Work Product Clumps	80
Long Input List	40
Featury Envy	40
Brain Activity	40
Brain Task	40

Envy, *Brain Task* e *Brain Activity*.

Apesar de alguns desses *process smells* serem menos significativos nenhum participante discordou das heurísticas proposta para detecção. Este fato pode ser verificado a partir das análises individuais dos *process smells*, assim parece que por menos significativo que sejam, eles foram detectado corretamente. Por fim da mesma forma que as heurísticas, as métricas aplicadas tiveram boa aceitação. As críticas neste sentido foram direcionadas ao valores limiares de detecção do *process smells*. Estas críticas podem ter ocorrido em virtude da estratégia de definição de valores limiares escolhida, onde apesar de se tentar evitar a estatística descritivo uma vez que a distribuição dos dados se apresentou de forma não normal, manteve os valores percentuais do estudo de Alves et al. (2010) ou mesmo da adaptação aplicada ao centro dos valores quando não foram alcançados os percentuais exatos entre os medidas do processo. Estes por sua vez não foram aplicados a um estudo preliminares antes da aplicação por não fazerem parte dos objetivos gerais do estudo. Apesar do desconforto de alguns participantes, baseado nos resultados da pesquisa este fato não foi prejudicial a qualidade do estudo.

O estudo também revelou que a prática de detecção de *bad smells* em processo como foi exposto não é uma prática comum durante a definição e melhoria de processo no cenário dos participantes. Este fato favoreceu a identificando algumas dificuldades para alcançar este formato de avaliação, mas que foi facilitada pelo presente estudo. Alguns exemplos são a dificuldade para a coleta das métricas e a dificuldade de fazer uma análise detalhada do processo devido ao alto volume de informação existente. Alguns participantes citaram que os conceitos utilizados para alcançar a detecção dos *process smells* não pareciam tão triviais e nem mesmo fáceis de serem capturadas, contudo os dados disponíveis e a navegação na especificação feita com SPEM favoreceram a validação manual.

Capítulo 7

AVALIAÇÃO GERAL DO CATÁLOGO

A realização deste trabalho foi organizada conforme a metodologia para elaboração do catálogo de *process smells*. Esta metodologia foi dividida nas fases de especificação do *process smells* e especificação das estratégias de detecção dos *process smells*. Ambas as fases foram validadas por estudos de entrevista. De modo a consolidar os resultados destes dois estudos de entrevista frente aos objetivos do trabalho este capítulo foi organizado.

Os resultados a seguir buscaram trazer respostas para as perguntas de pesquisa da dissertação: Se é possível e como identificar *bad smells* em modelos de processo de software? Para tanto foram realizadas análises conjuntas dos resultados dos dois estudos de entrevista, de forma a capturar a percepção teórico-prática dos analistas de processo e a percepção cotidiana dos engenheiros de software que atuam no estabelecimento e execução de seus processos de desenvolvimento de software. Estes resultados não apresentam uma generalização da população envolvida com desenvolvimento de software, mas buscam caracterizar as convergências ou divergências apresentadas pela população avaliada dada a natureza qualitativa e exploratória da pesquisa (JANSEN, 2010). Deste modo o fenômeno estudado, os *process smells*, puderam ser evidenciados e se fazia necessário compreender como estes são percebidos durante a especificação e atuação dos engenheiros de software com os seus respectivos processos de desenvolvimento de software. As análises quanto aos achados voltados especificamente para os *process smells* levam em consideração apenas os *process smells* que puderam ser avaliados nos dois estudos de entrevista.

Este capítulo está organizado em três seções. Na seção 7.1 com os resultados em relação ao objetivo geral, na seção 7.2 com os resultados dos objetivos específicos e por fim a seção 7.3 com um resumo sobre as ameaça a validade do estudo.

7.1 OBJETIVO GERAL

O objetivo geral deste trabalho foi especificar um catálogo de *process smells* para apoiar a identificação de anomalias em processo de software especificados com *Software & Systems Process Engineering Meta-Model* (SPEM). Após realizada a pesquisa diferentes contribuições obtidas indicam que um catálogo de *process smells* apoiam a identificação

de anomalias no processo de software. Os resultados da pesquisa apresentam indícios da preocupação com a qualidade da modelagem do processo de software pelos analistas de processo, assim como da percepção de qualidade do processo pelos executantes do processo, engenheiros de software em geral. Entretanto, a verificação da qualidade do processo de forma antecipada a sua execução, como foi verificado nesta dissertação, ainda precisa de estratégias e ferramentas de apoio para ocorra de modo que não proporcione o esforço demasiado de medição e detecção dos *process smells*.

As concordâncias e discordância de todos elementos avaliados como a definição e impactos possíveis, a percepção de redução aos subatributos de qualidade, das heurísticas, métricas e seus valores limiares relacionados em nenhum caso foram negadas completamente, em menores quantidade receberam baixa aceitação e não menores do que 40%. Estes são indícios para justificar que existe situações onde *process smell* podem ser detectados no processo de software.

7.2 OBJETIVOS ESPECÍFICOS

Esta seção apresenta os resultados do estudo em relação aos objetivos específicos estabelecidos.

7.2.1 Verificar a aplicabilidade dos bad smells clássicos de design de código orientado à objetos em processo de software e elencar e especificar os process smells;

A avaliação quanto a aplicabilidade de *bad smells* para processo de *software* levou em consideração as respostas das questões de finalização dos estudos de entrevista, assim como as respostas de concordância aos impactos negativos aos subatributos de qualidades que foram reconhecidas através dos *process smells*. Para tanto foi verificada a percepção de relevância tanto dos analistas de processo no primeiro estudo de entrevista (figura 7.1), quanto a dos profissionais do segundo estudo de entrevista. Apesar de não ser utilizado um método estatístico como correlação por não haver uma relação de causa e efeito, foi comparada as semelhanças das percepções de relevância para os dois estudos de entrevista. E dadas as percepções de relevâncias, estas foram associadas com as taxas de aceitação dos *process smells*.

A avaliação foi iniciada pelas respostas da questão em relação a relevância percebida quanto a proposta de *process smells* para o primeiro estudo de entrevista (figura 7.1). Essa questão utilizou uma escala com sete itens baseada na escala Likert, em seus valores assumiram como menor valor "pouquíssimo relevante" e maior valor "Muitíssimo relevante". Nesta questão a maioria das opiniões dos analistas de processo considerou esta proposição como relevante ou até mesmo muitíssimo relevante. Desta forma se apresenta uma totalidade das percepções em concordância com a proposta, onde a maioria dos analistas de processo (42%) acredita que a proposta seja muitíssimo relevante. Este posicionamento suporta que oito entre os dez *process smells* propostos obtiveram taxas de aceitação acima de 75% no primeiro estudo de entrevista (tabela 5.3). Ainda analisando a tabela 5.3 mesmo após a realização do segundo estudo de entrevista, onde não puderam

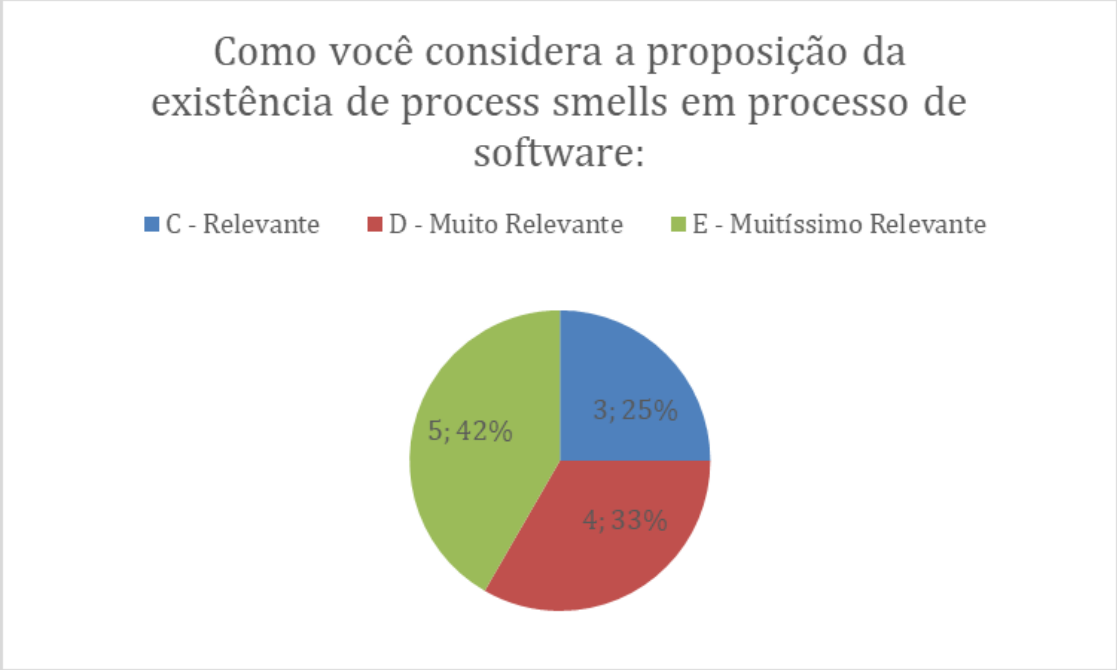


Figura 7.1 Relevância da proposta da existência de *Process Smells*.

ser validados os *process smells Message Chains* e *Large Activity*, ao descartar estes dois *process smells*, se pode verificar que apenas o *process smells Brain Activity* tem taxas de aceitação inferiores a 60%, enquanto os sete demais tem taxas de aceitação acima de 80%.

No panorama que considera as respostas do segundo estudo de entrevista foram realizadas duas questões uma quanto a relevância (figura 7.2) e a outra quanto à adequação das estratégias de detecção dos *process smells* (figura 7.3). Para a avaliação sobre a proposta de detecção de *process smells* a maioria dos participantes (80%) considerou a proposta relevante. Já quanto a eficiência das estratégias de detecção as respostas se agruparam entre adequada (60%) e muito adequada (40%) (figura 7.3). Por fim, foi feita a análise da eficácia (taxa de aceitação) do segundo estudo de entrevista com base na primeira questões onde os cinco participantes deveriam concordar ou discordar dos oito *process smells* (tabela 6.6). Nesta tabela percebemos que metade dos *process smells* tem uma aceitação entre 80% a 100%. E a outra metade tem a taxa de aceitação em 40%. Contudo as heurísticas de detecção foram totalmente aceitas. E quanto aos valores limiares houve uma taxa de aceitação de mais de 80%. Em resumo é possível avaliar que para o segundo estudo de entrevista a aplicabilidade dos *process smells* tem alta relevância. Quanto a eficácia foram verificados valores altos (80% a 100%) e moderados (40%), mas todos tem alguma significância. Em contrapartida a detecção dos *process smells* é validada pela total concordância com as heurísticas e alta aceitação dos valores limiares.

Recapitulando, nos conjuntos de respostas avaliados temos apenas respostas positivas quanto a percepção de relevância do estudo, onde para os *process smells* a eficácia do segundo estudo de entrevista e as taxas de aceitação no primeiro estudo de entrevista

possuem metade ou mais da metade dos valores significativos conforme a percepção dos participantes.

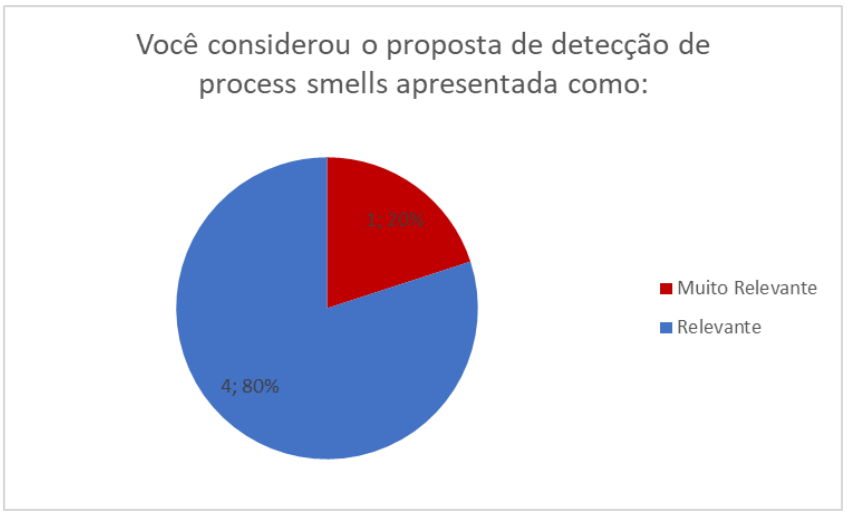


Figura 7.2 Relevância da proposta da existência de *Process Smells*.

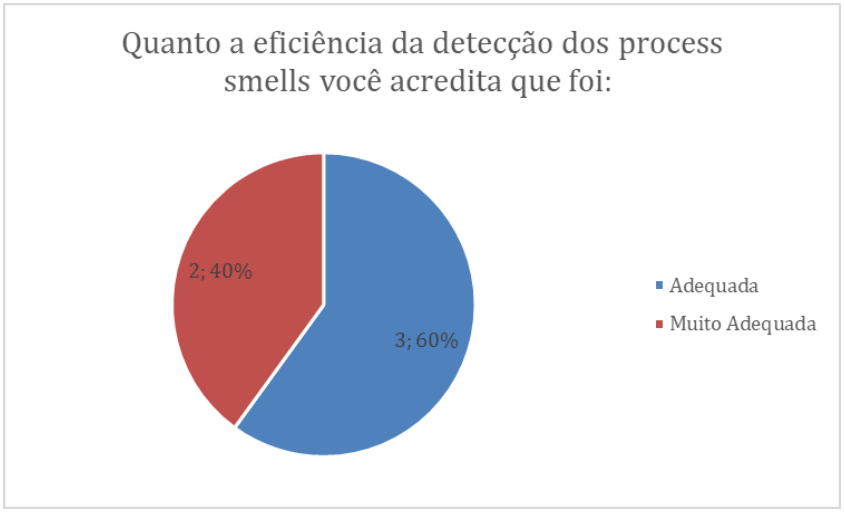


Figura 7.3 Eficiência da detecção de *Process Smells*.

Avaliando as respostas dos dois estudos de entrevista a partir do ponto de vista de cada perfil de participantes, se pode ver que os especialistas de processo foram mais otimistas quanto a proposta em relação aos profissionais do segundo estudo de entrevista mais moderados. Os especialistas aceitaram mais os *process smells* a partir do ponto de vista do *design* do catálogo, enquanto os profissionais envolvidos com a prática apresentam pouco mais da metade das respostas aceitando os *process smell* nos seus processos de trabalho. Desta forma podemos considerar que existe aplicabilidade aos *bad smells* de *design* de código orientado a objeto uma vez adaptados para a realidade de processo de

software. E como visto na seção de resultados da avaliação do catálogo quatro *process smells*, o *Shotgun Surgery*, *Divergent Change*, *Data Activity* e *Work Product Clumps*, entre os demais, foram os mais aceitos.

Nesta seção vale ressalta que entre os relatos dos participantes do segundo estudo de entrevista foi possível capturar a percepção que esta proposta trazia uma nova perspectiva de avaliação do processo que eles não conheciam. Apesar de trabalharem com o processo a certo tempo não tinha tido a oportunidade de avaliar o processo do ponto de vista da identificação de arranjo dos elementos do processo que reduzissem a qualidade percebida para os subatributos de qualidade como proposto pelos *process smells*. Além disso, os desenvolvedores e o testadores mais familiarizados com os conceitos de *bad smells* ou *code smells* e dívida técnica apresentaram melhor interação no estudo, fazendo mais proposições e questionamentos que os demais participantes. Os resultados gerais apresentados nesta seção imprimem que existem, em sua maioria, aspectos positivos quanto a aplicabilidade do catálogo de *process smells* baseado em *bad smells* do paradigma orientado a objeto. Desta forma foram validados tanto a adaptação dos *process smells* propostos, como a detecção e percepção de possíveis impactos negativos aos subatributos de qualidade que podem afetar um processo real de desenvolvimento de software.

7.2.2 Estabelecer as estratégias de detecção dos process smells

De modo a compreender se as estratégias de detecção proposta deram suporte a predição dos *process smells* foi realizada uma verificação das relações entre métricas estruturais de elementos de processo de software e os fatores de qualidade que favoreceram a indicação de *process smells*. Neste verificação foram consideradas apenas as respostas do estudo de entrevista 2. No estudo de entrevista 2 todas as heurísticas foram aceitas. Esta aceitação em totalidade evidência que as métricas propostas se relacionam com os subatributos de qualidade, uma vez que os sintomas se baseiam nas heurísticas e orientam quais métricas deve ser aplicadas para detecção dos *process smells*. Contudo nem todos os valores limiares aplicados para as métricas foram aceitos. Assim foram verificadas as taxas de aceitação dos valores limiares das métricas de modo a compreender como estas se relacionam com as estratégias de detecção dos *process smells*.

As taxas de aceitação dos valores limiares das métricas foram calculadas a partir das respostas de concordância dos participantes frente ao total de respostas possíveis. Sendo que o estudo de entrevista 2 teve cinco participantes, então como ilustração caso somente três participantes dos cinco aceitassem o valor limiar de uma determinada métrica a taxa de aceitação desse valor limiar seria de 60%, representando 3/5 do conjunto de aceitações possíveis.

A tabela 7.1 foi elaborada de modo a apoiar essa verificação. Na tabela podemos ver todo o esquema das estratégias de detecção dos *process smells* organizada na primeira coluna pelos subatributos de qualidade impactados. Na segunda coluna as subpropriedades ou medidas das subpropriedades que influenciam a percepção de qualidade dos subatributos da primeira coluna. As subpropiedades são mensurada pela métricas dos respectivos *process smells* vistos na terceira coluna. Na quarta coluna o percentual de

Tabela 7.1 Fatores, Aceitação e Métrica dos *Process Smells*.

Atributo	Propriedade	Process Smell / Métrica	% Aceitação Process Smell	% Aceitação Métrica
Modificabilidade		**Shotgun Surgery**	100	—
	Acoplamento	NSD(A)	—	100
Compreensibilidade		**Divergent Change**	100	—
	Coesão	RCT(A)	—	100
	Acoplamento	NPD(A)	—	60
Modificabilidade		**Data Activity**	100	—
	Tamanho	NWPOut(A)	—	80
	Tamanho	NSTP(A-Tasks)	—	60
Modificabilidade		**Work Product Clumps**	80	—
	Tamanho	FRWPCA	—	80
Compreensibilidade		**Long Input List**	40	—
	Tamanho	NWPIn(A)	—	100
Compreensibilidade		**Featury Envy**	40	—
	Acoplamento	EP1/3(T)	—	100
	Acoplamento	NPD(T)	—	100
	Acoplamento	EP(T)	—	100
Compreensibilidade		**Brain Task**	40	—
	Tamanho	NSTP(T)	—	60
	Complexidade	NG(T)	—	60
Compreensibilidade		**Brain Activity**	40	—
	Coesão	RCT(A)	—	100
	Tamanho	NSTP(A-Tasks)	—	80
	Complexidade	NG(A)	—	60

aceitação do *process smells* de modo a compreender o quanto o *process smells* foi percebido por influencias negativamente o atributo de qualidade e por fim a última coluna apresenta a taxa de aceitação dos valores limiares das métricas.

As considerações para esta tabela foram feitas observado o percentual (%) de aceitação do *process smell* diante do percentual (%) das taxas de aceitação dos valores limiares das métricas. Apesar de que alguns *process smell* não tenham sido percebidos como tão significativos quanto outros alcançando 40% de aceitação, os valores limiares utilizadas para detecção foram mais bem aceitos na composição da detecção de que alguns *process smell*, visto que nenhum recebeu uma taxa de aceitação inferior a 60%. Assim, mesmo os participantes não considerando que alguns *process smells* sejam tão significativos eles concordaram que os valores limiares estão entre uma moderada (60%) ou boa (100%) adequação para realizar a detecção de tal *process smell*.

As observações iniciais foram feitas no grupo de *process smells* que tem taxa de aceitação entre 100% a 80% correspondendo ao *Shotgun Surgery*, *Divergent Change*, *Data Activity* e *Work Product Clumps*. O *Shotgun Surgery* obteve total aceitação, assim como o thresold da métrica de acoplamento NSD(A). Ao *Divergent Change* é confirmada a proposição dos participantes nos resultados do segundo estudo de entrevista de que a coesão dada pela métrica RCT(A) é mais percebida pelo seu impacto negativo do que o acoplamento verificado pela métrica NPD(A), está com 60% de aceitação do seu thresold. O *Data Activity* como foi dito em análises anteriores obteve atenção dos participantes

em relação a quantidade de produto de trabalho de saída representada pela métrica NWOut(A) que fornece mais explicação deste *process smell* do que o thresold da métrica quantidade de tarefas da atividade representada por NSTP(A-Tasks). Quanto ao *Work Product Clumps* tanto sua aceitação quanto o thresold da única métrica que o compõe FRWPC obtiveram 80%.

Os demais *process smells* ficam com a taxa de 40% de aceitação sendo *Long Input List*, *Feature Envy*, *Brain Task* e o *Brain Activity*. O *Long Input List* e o *Feature Envy* obtiveram respectivamente para os valores limiares das suas métricas NWPIn(A) e EP1/3(T), NPD(T) e EP(T) o valor de 100% de aceitação. Observado o *Brain Task* e o *Brain Activity* podemos verificar que o número de fluxos forneceu pouca explicação. E o tamanho em NSTP(T) não explica tanto no *Brain Task*, o que se justifica, pois, as tarefas no processo não apresentaram um valor maior do que quatro passos. Diferente da percepção no *Brain Activity* onde NSTP(A-Tasks) alcança 80% visto que as atividades possuem uma quantidade maior de tarefas. Por fim o RCT(A) somente do *Brain Activity* com 100% de aceitação do valor limiar mostra como visto no *Divergent Change* que a coesão fornece boa explicação sobre o atributo compreensibilidade.

De modo a observar as métricas (incluindo os seus valores limiares) mais aceitas como fator que impactam negativamente os subatributos de qualidade foram elaborados dois gráficos. Nestes gráficos as métricas estão organizadas de acordo as que obtiveram maior percentual de aceitação. O primeiro gráfico faz relação ao atributo de compreensibilidade (figura 7.4) e no segundo gráfico ao atributo de modificabilidade (figura 7.5). As métricas com 100% significam que foram totalmente aceitas em todas as vezes que foram aplicadas nas detecções dos *process smells*. Isto acorreu para RCT(A), EP(T), EP1/3(T), NPD(T), NWPIn(A) no primeiro gráfico e apenas e NSD(A) no segundo. As métricas NSTP(A-Tasks) do primeiro gráfico e FRWPC e NWPOut(A) do segundo gráfico obtiveram 80%. Por fim todas as demais métricas obtiveram 60% de aceitação. A NSTP(A-Tasks) se repete nos dois gráficos assim foi observada tanto para compreensibilidade obtendo 80% de aceitação quando para a modificabilidade obtendo 60% de aceitação.

Assim verificamos que existem relações sobre como as métricas e seus valores limiares são percebidas em relação aos subatributos de qualidade compreensibilidade e modificabilidade, apesar de que, nem toda concordância ou discordância com os valores limiares refletir totalmente a aceitação ou rejeição dos *process smells*, algumas dessas influências ajudaram a explicar os elementos mais destacados na percepção de impacto negativo aos subatributos de qualidade indicados pelos engenheiros de software.

7.2.3 Validar se o catálogo consegue indicar process smells em modelos de processo de software

Considerando as respostas do segundo estudo de entrevista, as estratégias de detecção foram totalmente aceitas, o que inclui tanto a filtragem como a composição.

Os participantes fizeram poucas sugestões em relação as estratégias de detecção. Estas sugestões indicaram ajustes em alguns *process smells* ou a adição de heurísticas. Em relação a adição de heurísticas, os participantes indicaram elementos que apesar de estarem na descrição do *process smell* não receberam uma heurística para a estratégia de

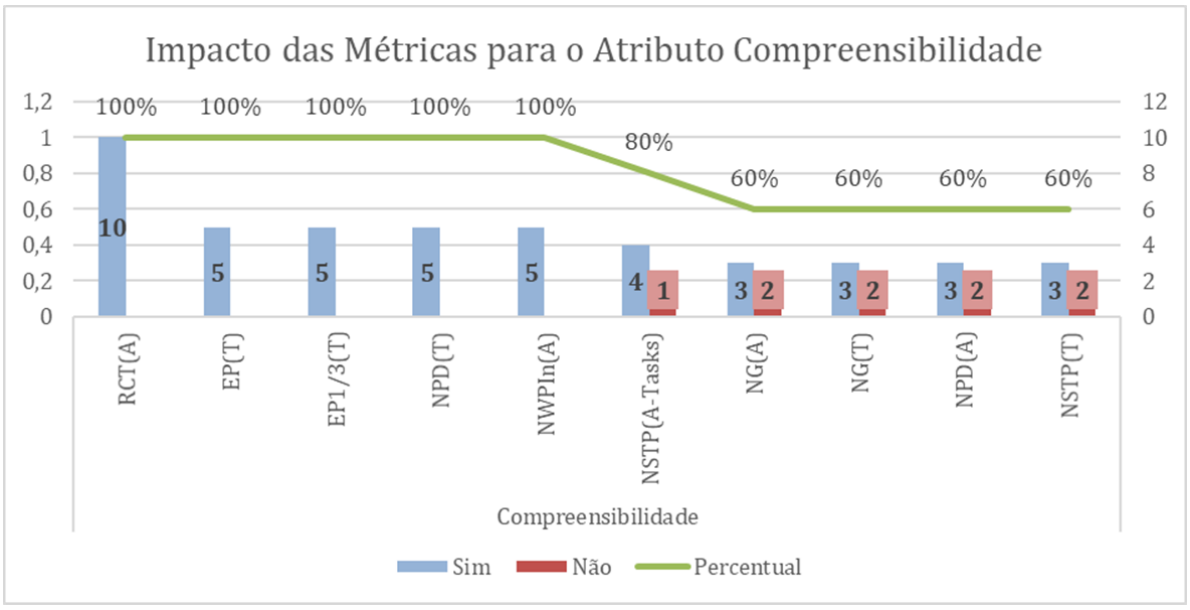


Figura 7.4 Impacto das métricas para o atributo de compreensibilidade.

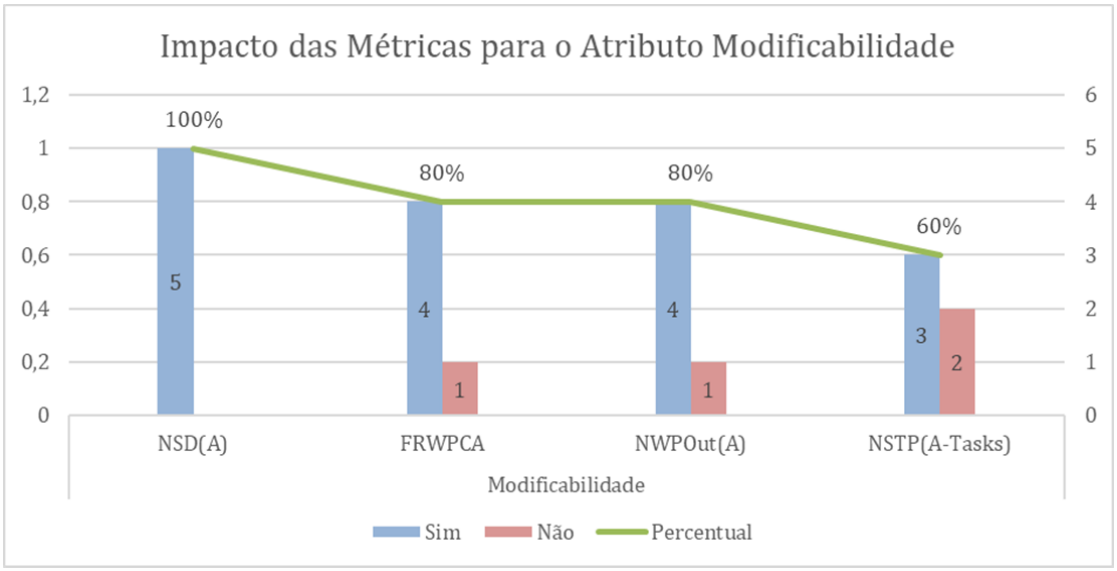


Figura 7.5 Impacto das métricas para o atributo de modificabilidade.

detecção.

Para esta adição de heurísticas dois casos foram sugeridos. Foi sugerido que o *Feature Envy* poderia ter alguma heurística para detectar a distância em que os produtos de trabalhos que serão utilizados em relação a tarefa. Já o *Data Activity* poderia utilizar a heurística de acoplamento aferente para verificar as dependências reais ocasionadas em outras atividades.

Por fim em relação aos valores limiares os participantes tiveram duas críticas principais. A primeira foi quanto aos valores que eles não concordaram mas não apresentaram uma explicação detalhada das suas crenças. A segunda crítica foi quanto aos valores limiares obtidos pela adaptação das estratégias de detecção. Para este casos alguns participantes indicaram que deveriam ser utilizados os valores máximos obtidas pelas métricas para definir os valores limiares. Contudo esta abordagem reduziria ou até inviabilizaria a identificação de *process smells*.

Apesar das críticas de forma geral nenhum valor limiar recebeu uma aceitação menor que 60%, o que imprime que existiu adequação para a detecção dos *process smells*.

Revisando de forma geral os resultados do catálogo proposto, a utilização do método de detecção de *bad smells* do paradigma de orientação à objetos proposta por Marinescu (2004) se mostrou aceitável para a detecção de *process smells*.

7.3 AMEAÇAS A VALIDADE

Em relação as ameaças gerais quanto a proposta desta dissertação leva em considerações os dois estudos de entrevista realizados, assim serão observados dois aspectos que podem ser ameaças para este tipo de estudo.

Como se tratou de um estudo exploratório ambos cenários demandaram tempo consideração de interação dos participantes que pode ter ocasionado certa fadiga. Contudo os profissionais envolvidos costumam participar de baterias extensas de trabalho frente a ajustes do processo em seus respectivos ambientes de trabalho, assim é esperado que o tempo na pesquisa não tenha causado alto grau de fadiga aos participantes.

O segundo aspecto que pode oferecer ameaça ao estudo é o fato da interação com conhecimento técnico específico proposto na pesquisa. Neste caso múltiplos conhecimentos como em relação aos subatributos e propriedades de qualidade, relações de causa e efeito, elementos e medidas estruturais de modelo de processo de software especificado com SPEM, valores limiares, e outros foram tratados ao mesmo tempo, além de requerer as experiências dos profissionais. Para amenizar estes impactos os dois estudos de entrevista contaram com uma instrução inicial com as principais informações para orientar a livre interação dos participantes com os artefatos do estudos, além de ser possível retirar eventuais dúvidas durante a execução dos estudos.

Capítulo

8

CONSIDERAÇÕES FINAIS

Concluímos nesta pesquisa que *bad smells* podem ser aplicáveis para processos de *software*. Foi possível perceber que alguns *process smells* se mostraram mais significativos que outros, o que não é um cenário incomum para o contexto da Engenharia de Software. Este fato acontece em situações como a escolha de metodologia, priorização de requisitos, seleção de padrões de projetos que são constantemente mais ou menos valorados de acordo com o contexto em questão. Assim como as evidência de projetos de *software* consideram que erros em requisitos ocasionam alto custo ou até o fracasso de um projeto, avaliar um processo de *software* forma antecipada foi o objetivo da proposta desta dissertação visando evitar impactos negativos para o desenvolvimento de software a partir de um processo com qualidade reduzida.

Recapitulando os resultados dos *process smells* podem ser destacadas algumas características quanto aqueles percebidas como mais significativas. Os *process smells Shotgun Surgery*, *Data Activity* e *Work Product Clump* relacionados a impactos negativos à modificabilidade foram mais percebidos como significativo pelos participantes. Provavelmente, a propriedade estrutural modularidade definida pela medida de acoplamento avaliada nestes *process smells* foi compreendida como uma característica crítica ao avaliar impactos negativos para processo. O alto acoplamento quando da necessidade de ajustes, mudanças ou melhorias no processo pode proporcionar demasiado esforço para rastrear, avaliar impactos e realizar as modificações mantendo o processo integro. Já o *Divergent Change* relacionados à compreensibilidade também se apresentou significativo porém impactando negativamente a compreensibilidade. Para este *smell*, a propriedade complexidade definida pela medida de coesão foi o elemento mais percebido. A baixa coesão entre elementos do processo foi percebida e evidenciada pelos participantes como fator que implica negativamente a compreensibilidade. A coesão indica o quanto as tarefas trocam produtos de trabalho entre si, assim tarefas que atendem a diferentes propósitos tendem a ter baixa coesão o que dificultou a compreensibilidade dos participantes em relação aos elementos avaliados no processo.

A detecção de *process smells* se tratando de uma forma de avaliação da qualidade do processo pode apoiar algumas conclusões. Esta forma de avaliação observa a qualidade do processo do ponto de vista do usuário uma vez que se aproxima no modelo

de avaliação por múltiplas perspectivas. Assim subatributos como compreensibilidade e modificabilidade podem ser avaliados refletindo necessidade cotidianas dos usuários, diferente dos demais modelos de avaliações com critérios preestabelecidos como nas avaliação do produto frente ao processo, avaliação por modelos de avaliação ou avaliação através especificações formais. Apesar de ter focos diferentes quanto a avaliação a detecção de *process smells* não parece competir com outras formas de avaliação de qualidade de processo, possivelmente até sejam complementares visto que juntam expandem os aspectos de qualidade a serem avaliados.

A detecção de *process smells* pode apresentar alguns benefícios. A definição dos *process smells* partem de problemas já conhecidos na Engenharia de Software como alto acoplamento, baixa coesão, alta complexidade e tamanho demasiado dos elementos de software. Como o processo de software também pode ser considerado como software (OSTERWEIL, 1987) estes problemas podem afetar o processo impactando negativamente a sua compreensibilidade e modificabilidade. Assim aplicar a detecção de *process smells* pode contribuir para redução desses problemas.

Os resultados obtidos nesta pesquisa podem ter diferentes formas de uso. Algumas abordagens listas a seguir não impendem quaisquer outras formas de interpretação dos dados. O catálogo foi especificado considerando elementos básicos da especificação *Software & Systems Process Engineering Meta-Model* (SPEM). Partindo desse pressuposto esperasse que realizar a detecção de *process smells* em outros processos especificados com SPEM seja mais facilmente aplicável. Entretanto, não significa que outras especificações não possam se utilizar deste catálogo dadas as devidas adaptações ou diferentes interpretações.

Outra forma de uso dos *process smells* é a partir do suporte fornecido pelo catálogo. Mesmo caso não sejam aplicadas as estratégias de detecção, o catálogo serve de referência para profissionais que especificam processos reconhecer arranjos dos elementos do processo que se evitados podem melhorar a facilidade de compreender e modificar o processo.

Os resultados desta pesquisa também podem servir como ponto de partida para:

1. basear-se, expandir ou reestruturar o levantamento sobre como vem sendo detectados e quais as taxonomia dos *bad smells* (tabela A.1);

2. elaborar novos *process smells*;

3. aproveitar a comparações entre elementos de código orientado a objeto e SPEM ou mesmo o catalogo atual para apoiar adaptação dos *process smells* para demais formas de especificações de processos de software;

4. avaliar a aplicabilidade dos *process smells* em outros modelos de processos além do modelo de software, como por exemplo a modelos de negócio como os especificados com *Business Process Modeling Notation* (BPMN).

Esta pesquisa não foi realizado com intuito de avaliar modelos de processos genéricos como espiral, *Scrum*, *Rational Unified Process* (RUP) entre outros. Intencionalmente a proposta teve o intuito de avaliar processo reais de desenvolvimento de software quanto a ocorrência de *process smells*. Processos reais refletem as necessidades cotidianas de uma

determinada empresa para alcançar a produção do software, enquanto modelos genéricos propõem princípios, boas práticas ou padrões que orientam uma forma de produzir software sugerindo quais são os benefícios ao adotar tal abordagem. De modo para processos reais era esperado que os participantes conseguissem apresentar mais rapidamente as suas percepções partindo das experiências práticas.

8.1 TRABALHOS FUTUROS

Por fim esta pesquisa apresenta algumas contribuições que podem ser úteis para trabalhos futuros. Como trabalhos futuros percebidos durante a realização da pesquisa podem ser destacadas: a extensão da validação da pesquisa; a evoluções no catálogo de *process smells*; uma definição de valores limiares mais adequadas para processo de software; estabelecer uma forma de detecção automatizada dos *process smells* e o estabelecimento de estratégias de refatoração dos *process smells*.

O estudo sobre *process smells* pode ganhar maior validação dado que sejam realizadas replicação desse estudo. A replicação de estudos costumam estender a pesquisa. No caso dos *process smells* algumas extensões que poderiam ser alcançadas como o aumento da validação das definições e impactos negativos dos *process smells* aos atributos de qualidade. Para tanto a realização de replicação com diferentes configurações, por exemplo maior quantidade de participantes, maior quantidade de processos ou por diferentes contextos colaborariam para uma maior validação dos *process smells*.

A extensão dos *process smells* pode estabelecer trabalhos futuros com diferentes aspectos.

1. identificar e avaliar a incorporação de novas características específicas de processo de software na definição dos *process smells* existentes como a quantidade de papéis envolvidos nas atividades e tarefas indicar pelos participantes durante os estudos de entrevista.

2. avaliar a viabilidade de elaborar *process smells* com elementos mais especializados do SPEM. Um exemplo para este trabalho seria verificar se o uso do elemento variabilidade (*variability element*) se assemelha a estrutura dos *bad smells* relacionados a propriedades específicas do código orientado a objeto como herança e especialização. Este estudo poderia revisar o primeiro critério de exclusão de *bad smells* para adaptação para *process smells* aplicado na pesquisa desta dissertação.

3. revisar o segundo critério de exclusão de *bad smells* para adaptação para *process smells* aplicado na pesquisa desta dissertação. Avaliar e compreender a viabilidade para adaptar *process smells* cujo a detecção se utilize de avaliação semântica poderia agregar novos itens ao catálogo.

4. elaborar novos *process smells* levando em consideração as características específicas ou elementos que ainda não tiverem sido caracterizados em alguns *process smells* baseado na especificação SPEM.

O catálogo foi especificado a partir da adaptação de *bad smells* de código porém é possível que sejam estabelecidos novos *process smells* a partir de características de modelos genéricos de processo de software. Um trabalho futuro seria investigar a elaboração destes *process smells* originados da avaliação de modelo genérico de processo como espiral, *Scrum* ou outros, uma vez que estes possuem características particulares entre si. Avaliando as características peculiares desses modelos genéricos talvez seja possível identificar configurações que promovam algum aspecto que deduza a qualidade do processo frente a atributos como compreensibilidade ou modificabilidade.

Estabelecer estratégias para obter valores limiares não é uma tarefa trivial. Algumas pesquisas têm se dedicado para alcançar valores limiares cada vez mais precisos considerando diferentes perspectivas do contexto ao qual se aplicam (ALVES; YPMA; VISSER, 2010) (SÁNCHEZ-GONZÁLEZ et al., 2012) (FONTANA et al., 2015). Desta forma um trabalho futuro possível seria a determinação de estratégias para obter valores mais adequadas para processo de software. Alguns aspectos que poderiam ser incluídos neste trabalho seria obter a concordância dos valores limiares a partir de múltiplas perspectivas dos usuários (desenvolvedores, testadores, gestores) e verificar se os valores limiares seriam influenciados pelos modelos genéricos dos quais os processos reais se baseiam.

A elaboração de uma ferramenta para detecção automatizada seria um trabalho futuro que colaboraria facilitando a detecção de *process smells*. A preparação dos *scripts* semi automatizados contou além de algumas interações manuais, com o uso de diferentes recursos. Web scraping (extração de conteúdo *Hypertext Markup Language* (HTML)) com *Python* para obter os dados das publicação do processo em HTML. *Power Query*, recurso de análise de dados do *Excel* para extrair as métricas dos elementos estruturais do SPEM (passos, tarefas, atividade e produtos de trabalho). Estratégias de análise de dados com *Python* para estabelecer os valores limiares e realizar das detecções dos *process smells*. Todos estes recursos podem ser traduzidos em uma ferramenta para a detecção de *process smells*. Neste trabalho uma possibilidade seria realiza a leitura do arquivo base em *eXtensible Markup Language* (XML) resultante da especificação de processo com SPEM utilizando a ferramenta *Eclipse Process Framework* (EPF). Esta ferramenta proporcionaria a facilitação no uso da detecção de *process smells* em processo reais sem agregar o esforço de usar todos os recursos citados.

A refatoração é a estratégia que procura descaracterizar os *bad smells* de um código fonte. O reconhecimento dos *process smells* é um passo para a melhoria da qualidade do processo. Contudo a refatoração proporciona a remoção da causa da redução da qualidade do processo. Desta forma a realização de um trabalho futuro para estabelecer estratégia de refatoração de *process smells* proporcionaria um segundo passo para o alcance da melhoria do processo de *software*.

Os trabalhos futuros apresentados não inviabilizam a possibilidade de serem estabelecidos outros trabalhos futuros, porém esses foram os trabalhos mais percebidos ao decorrer do percurso da pesquisa.

REFERÊNCIAS BIBLIOGRÁFICAS

ALVES, T. L.; YPMA, C.; VISSER, J. Deriving metric thresholds from benchmark data. In: IEEE. *2010 IEEE International Conference on Software Maintenance*. [S.l.], 2010. p. 1–10.

BOEHM, R. M. B.; STEECE, B. Cost estimation with cocomo ii. *Prentice-Hall*, 2000.

BRAVO, F. M. et al. A logic meta-programming framework for supporting the refactoring process. *Master's Thesis, Vrije Universiteit, Brussel*, Citeseer, 2003.

CARNEIRO, G. d. F. et al. Identifying code smells with multiple concern views. In: IEEE. *2010 Brazilian Symposium on Software Engineering*. [S.l.], 2010. p. 128–137.

CHIDAMBER, S. R.; KEMERER, C. F. A metrics suite for object oriented design. *IEEE Trans. Softw. Eng.*, IEEE Press, v. 20, n. 6, p. 476–493, jun. 1994. ISSN 0098-5589. Disponível em: ⟨https://doi.org/10.1109/32.295895⟩.

CHIDAMBER, S. R.; KEMERER, C. F. A metrics suite for object oriented design. *IEEE Transactions on software engineering*, IEEE, v. 20, n. 6, p. 476–493, 1994.

CORRADINI, F. et al. A guidelines framework for understandable bpmn models. *Data & Knowledge Engineering*, Elsevier, v. 113, p. 129–154, 2018.

DEURSEN, A. V. et al. Refactoring test code. In: *Proceedings of the 2nd international conference on extreme programming and flexible processes in software engineering (XP)*. [S.l.: s.n.], 2001. p. 92–95.

FLICK, U. *Introdução à pesquisa qualitativa-3*. [S.l.]: Artmed editora, 2008.

FONTANA, F. A.; BRAIONE, P.; ZANONI, M. Automatic detection of bad smells in code: An experimental assessment. *Journal of Object Technology*, v. 11, n. 2, p. 5–1, 2012.

FONTANA, F. A. et al. Automatic metric thresholds derivation for code smell detection. In: IEEE PRESS. *Proceedings of the Sixth international workshop on emerging trends in software metrics*. [S.l.], 2015. p. 44–53.

FOWLER, F.; KENT, G. Bad smells in code. *Refactoring: Improving the design of existing code*, p. 75–88, 1999.

FOWLER, M. *Refactoring: improving the design of existing code*. [S.l.]: Addison-Wesley Professional, 1999.

GARCÍA-BORGOÑON, L. et al. Software process modeling languages: A systematic literature review. *Information and Software Technology*, Elsevier, v. 56, n. 2, p. 103–116, 2014.

GARCÍA, F. et al. Maintainability of software process models: an empirical study. In: IEEE. *Ninth European Conference on Software Maintenance and Reengineering*. [S.l.], 2005. p. 246–255.

GARCÍA, F. et al. Integrated measurement for the evaluation and improvement of software processes. In: SPRINGER. *European Workshop on Software Process Technology*. [S.l.], 2003. p. 94–111.

GARCÍA, F.; RUIZ, F.; PIATTINI, M. Definition and empirical validation of metrics for software process models. In: SPRINGER. *International Conference on Product Focused Software Process Improvement*. [S.l.], 2004. p. 146–158.

GENVIGIR, E. C.; FILHO, F. L.; SANT'ANNA, N. Modelagem de processos de software através do spem-software process engineering metamodel-conceitos e aplicação. *III WORCAP, São Jose dos Campos, SP*, 2003.

GREILER, M.; DEURSEN, A. V.; STOREY, M.-A. Automated detection of test fixture strategies and smells. In: IEEE. *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation*. [S.l.], 2013. p. 322–331.

GRUBEL, D. E. Process modeling language specification. 1995.

ISO. Standard. *ISO 9000: international standards for quality management*. Geneva, CH: [s.n.], 2000.

JANSEN, H. The logic of qualitative survey research and its position in the field of social research methods. In: *Forum Qualitative Sozialforschung/Forum: Qualitative Social Research*. [S.l.: s.n.], 2010. v. 11, n. 2.

JÚNIOR, Á. F. de B.; JÚNIOR, N. F. A utilização da técnica da entrevista em trabalhos científicos. *Revista Evidência*, v. 7, n. 7, 2012.

KHLIF, W. et al. Quality metrics for business process modeling. In: WORLD SCIENTIFIC AND ENGINEERING ACADEMY AND SOCIETY (WSEAS). *Proceedings of the 9th WSEAS international conference on Applied computer science*. [S.l.], 2009. p. 195–200.

KROEGER, T. A. *Understanding the characteristics of quality for software engineering processes*. Tese (Doutorado), 2011.

KROEGER, T. A.; DAVIDSON, N. J.; COOK, S. C. Understanding the characteristics of quality for software engineering processes: A grounded theory investigation. *Information and Software Technology*, Elsevier, v. 56, n. 2, p. 252–271, 2014.

LANZA, M.; MARINESCU, R. *Object-oriented metrics in practice: using software metrics to characterize, evaluate, and improve the design of object-oriented systems*. [S.l.]: Springer Science & Business Media, 2007.

LIU, W.; HU, Z.; LIU, H. An automatic approach to detecting and eliminating lazy classes based on abstract syntax trees.

MACÍA, I.; SANT'ANNA, C.; STAA, A. v. Detectando problemas de design em diagramas de classes: Um estudo experimental. In: *V Experimental Software Engineering Latin American Workshop (ESELAW2008)*. [S.l.: s.n.], 2008.

MANTYLA, M.; VANHANEN, J.; LASSENIUS, C. A taxonomy and an initial empirical study of bad smells in code. In: IEEE. *Software Maintenance, 2003. ICSM 2003. Proceedings. International Conference on*. [S.l.], 2003. p. 381–384.

MARCONI, M. d. A.; LAKATOS, E. M. et al. *Técnicas de pesquisa*. [S.l.]: São Paulo: Atlas, 2002.

MARINESCU, R. Detection strategies: Metrics-based rules for detecting design flaws. In: IEEE. *Software Maintenance, 2004. Proceedings. 20th IEEE International Conference on*. [S.l.], 2004. p. 350–359.

MARTICORENA, R.; LÓPEZ, C.; CRESPO, Y. Extending a taxonomy of bad code smells with metrics. In: *7th ECCOP International Workshop on Object-Oriented Reengineering (WOOR)*. [S.l.: s.n.], 2006. p. 6.

MOHAGHEGHI, P.; DEHLEN, V.; NEPLE, T. Definitions and approaches to model quality in model-based software development–a review of literature. *Information and Software Technology*, Elsevier, v. 51, n. 12, p. 1646–1669, 2009.

MOODY, D. L. Theoretical and practical issues in evaluating the quality of conceptual models current state and future directions. *Data & Knowledge Engineering*, Elsevier, v. 55, n. 3, p. 243–276, 2005.

OCA, I. M.-M. de et al. A systematic literature review of studies on business process modeling quality. *Information and Software Technology*, Elsevier, v. 58, p. 187–205, 2015.

OMG. Software and systems process engineering metamodel specification 2.0. 2008.

OSTERWEIL, L. Software processes are software too. In: *Proceedings of the 9th International Conference on Software Engineering*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1987. (ICSE '87), p. 2–13. ISBN 0-89791-216-0. Disponível em: ⟨http://dl.acm.org/citation.cfm?id=41765.41766⟩.

PALOMBA, F. Textual analysis for code smell detection. In: IEEE PRESS. *Proceedings of the 37th International Conference on Software Engineering-Volume 2*. [S.l.], 2015. p. 769–771.

PALOMBA, F. et al. Detecting bad smells in source code using change history information. In: IEEE PRESS. *Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering*. [S.l.], 2013. p. 268–278.

PALOMBA, F. et al. Do they really smell bad? a study on developers' perception of bad code smells. In: IEEE. *2014 IEEE International Conference on Software Maintenance and Evolution*. [S.l.], 2014. p. 101–110.

PARNIN, C.; GÖRG, C.; NNADI, O. A catalogue of lightweight visualizations to support code smell inspection. In: ACM. *Proceedings of the 4th ACM symposium on Software visualization*. [S.l.], 2008. p. 77–86.

REGULWAR, G. B.; TUGNAYAT, R. M. Bad smelling concept in software refactoring. *Jawaharlal Darda Institute of Engineering & Technology, MIDC, Lohara, Yavaymal (MS), INDIA*, 2012.

ROMPAEY, B. V. et al. On the detection of test smells: A metrics-based approach for general fixture and eager test. *IEEE Transactions on Software Engineering*, IEEE, v. 33, n. 12, p. 800–817, 2007.

RUIZ-RUBE, I.; DODERO, J. M.; COLOMO-PALACIOS, R. A framework for software process deployment and evaluation. *Information and Software Technology*, Elsevier, v. 59, p. 205–221, 2015.

SÁNCHEZ-GONZÁLEZ, L. et al. A study of the effectiveness of two threshold definition techniques. IET, 2012.

SANTOS, E.; SUZANA, R.; SANT'ANNA, C. A catalogue of bad smells for software process. In: . [S.l.: s.n.], 2018. p. 1–10.

SILINGAS, D.; MILEVICIENE, E. Refactoring bpmn models: From 'bad smells' to best practices and patterns. *2007 BPM & Workflow Handbook*, Future Strategies Inc., p. 125, 2007.

SILVA, F. A. das D. et al. Um modelo de simulação de processos de software baseado em agentes cooperativos. 1999.

SOMMERVILLE, I. Software engineering 9th edition. *ISBN-10*, v. 137035152, 2011.

TYRRELL, S. The many dimensions of the software process. *Crossroads*, v. 6, n. 4, p. 22–26, 2000.

UMESH, I.; SRINIVASAN, G. *A study on bad code smell*. [S.l.]: IV, 2015.

VANDERFEESTEN, I. et al. Quality metrics for business process models. *BPM and Workflow handbook*, Citeseer, v. 144, p. 179–190, 2007.

VANSUETENDAEL, N.; ELWELL, D. *Software Quality Metrics*. [S.l.], 1991.

WEBER, K. et al. Melhoria de processo do software brasileiro (mps. br): um programa mobilizador. In: *Proceedings of the XXXI Conferencia Latinoamericana de Informatica (CLEI 2006). Santiago, Chile: agosto.* [S.l.: s.n.], 2006.

ZHANG, M. et al. Improving the precision of fowler's definitions of bad smells. In: IEEE. *2008 32nd Annual IEEE Software Engineering Workshop.* [S.l.], 2008. p. 161–166.

Apêndice

A

LEVANTAMENTO DOS BAD SMELLS

Este apêndice apresenta a lista de artigos resultantes do levantamento feito nesta pesquisa sobre como os *bad smells* vinham sendo detectados e quais eram as taxonomia sobre *bad smells*.

Tabela A.1: Levantamento de taxonomias e formas de detecção de bad smells

Taxonomia *Bad Smells* OO	**Fonte**
Brain Class	(LANZA; MARINESCU, 2007)
Brain Method	(LANZA; MARINESCU, 2007)
Dispersed Coupling	(LANZA; MARINESCU, 2007)
Incomplete Library Class	(BRAVO et al., 2003)
Intensive Coupling	(LANZA; MARINESCU, 2007)
Significant Duplication	(LANZA; MARINESCU, 2007)
The Bloaters - -(God) Large Class	(LANZA; MARINESCU, 2007)
The Bloaters - -(God) Large Class	(FOWLER, 1999)
The Bloaters - -Long Parameter List	(FOWLER, 1999)
The Bloaters - -Long Parameter List	(BRAVO et al., 2003)
The Change Preventers - -Divergent Change	(BRAVO et al., 2003)
The Change Preventers - -Divergent Change	(PALOMBA et al., 2013)
The Change Preventers - -Divergent Change	(CARNEIRO et al., 2010)
The Change Preventers - -Parallel Inheritance Hierarchies	(REGULWAR; TUGNAYAT, 2012)
The Change Preventers - -Shotgun Surgery	(FONTANA et al., 2015)

continua na próxima página

Tabela A.1 – Levantamento de taxonomias e formas de detecção de bad smells

Taxonomia *Bad Smells* OO	**Fonte**
The Bloaters - -Data Clumps	(BRAVO et al., 2003)
The Bloaters - -Data Clumps	(REGULWAR; TUGNAYAT, 2012)
The Bloaters - -Long Method	(FOWLER, 1999)
The Bloaters - -Long Method	(PALOMBA, 2015)
The Bloaters - -Primitive Obsession	(UMESH; SRINIVASAN, 2015)
The Bloaters - -Primitive Obsession	(FOWLER, 1999)
The Bloaters - -Primitive Obsession	(FOWLER, 1999)
The Couplers - - Inappropriate Intimacy	(FOWLER, 1999)
The Couplers - -Feature Envy	(LANZA; MARINESCU, 2007)
The Couplers - -Inappropriate Intimacy	(BRAVO et al., 2003)
The Couplers - -Message Chains	(PARNIN; GÖRG; NNADI, 2008)
The Couplers - -Message Chains	(BRAVO et al., 2003)
The Couplers - -Middle Man	(PARNIN; GÖRG; NNADI, 2008)
The Couplers - -Middle Man	(BRAVO et al., 2003)
The Couplers - -Middle Man	(REGULWAR; TUGNAYAT, 2012)
The Dispensables - - Lazy Load	(LIU; HU; LIU,)
The Dispensables - - Lazy Load	(FOWLER, 1999)
The Dispensables - -Data Class	(LANZA; MARINESCU, 2007)
The Dispensables - -Dead Code	(FONTANA; BRAIONE; ZANONI, 2012)
The Dispensables - -Dead Code	(REGULWAR; TUGNAYAT, 2012)
The Dispensables - -Duplicate Code	(BRAVO et al., 2003)
The Dispensables - -Duplicate Code	(REGULWAR; TUGNAYAT, 2012)
The Dispensables - -Duplicate Code	(BRAVO et al., 2003)
The Dispensables - -Speculative Generality	(PALOMBA et al., 2014)
The Dispensables - -Speculative Generality	(ZHANG et al., 2008)

Apêndice

B

QUESTIONÁRIO DO ESTUDO DE ENTREVISTA 1

Apêndice 2 Questionário do Estudo de Entrevista 1

Este apêndice apresenta a seguir o questionário aplicado no estudo de entrevista 1.

Bem-vindo ao questionário sobre *process smells*, que faz parte da minha pesquisa de mestrado em Ciências da Computação no PGCOMP UFBA.

Este questionário tem como objetivo receber opiniões de concordância ou não, sobre *process smells*, uma proposta de situações em geral, que acreditamos que prejudicam a qualidade de processos de software. O público alvo do questionário são profissionais que atuem de alguma forma com gestão de processo de software.

De forma geral, ao final do trabalho, esperamos definir um catálogo com caracterização e estratégia de detecção de *process smells*. Nossa expectativa é que este catálogo possa contribuir para a redução, ainda na fase de modelagem do processo (design), dos impactos causados pelos *process smells* que normalmente só são percebidos durante a execução do processo (enactment).

Perfil do Participante

Nesta seção fale um pouco sobre você.

1 - Você tem alguma experiência com processo de software? * Favor escolher apenas uma das opções a seguir:

A - Sim

B - Não

2 - Você já trabalhou executando alguma fase de um processo de software definido por uma empresa? * Favor escolher apenas uma das opções a seguir:

A - Sim

B - Não

3- Quantos processos de software você já modelou/especificou? (Antes de responder Veja as Dicas logo abaixo das Opções de Resposta) * Escolha uma das seguintes respostas:

Favor escolher apenas uma das opções a seguir:

A - Nunca modelei nenhum processo

B - Modelei apenas um processo

C - Modelei entre 2 à 5 processos

D - Modelei entre 6 à 15 processos

E - Modelei mais que 15 processos

DICAS

Entenda modelagem e especificação como qualquer atividade que tenha como finalidade descrever e registrar de modo a formalizar um conjunto de atividades inerentes a qualquer parte do desenvolvimento de software. Neste caso independe o meio utilizado, considere: diagrama, documentação textual, especificações como BPM, SPEM, UML, entre outras formas.

4 - Quanto tempo de experiência você tem atuando com modelagem de processo? * Só responder essa pergunta sob as seguintes condições: Resposta foi maior que 'Nunca modelei nenhum processo' na questão 3. Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A- Menos de 6 meses

B - Entre 6 meses e 1 ano.

C - Entre 1 à 2 anos.

D - Entre 2 à 3 anos.

E - Entre 3 à 4 anos.

F - Acima de 4 anos.

5 - Você conhece a especificação de processo de software e sistemas SPEM da Object Management Group? (Antes de responder Veja as Dicas logo abaixo das Opções de Resposta)* Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A - Não conhece nenhum processo(s) modelado(s) com SPEM

B - Conhece superficialmente o assunto sobre processo(s) modelado(s) com SPEM

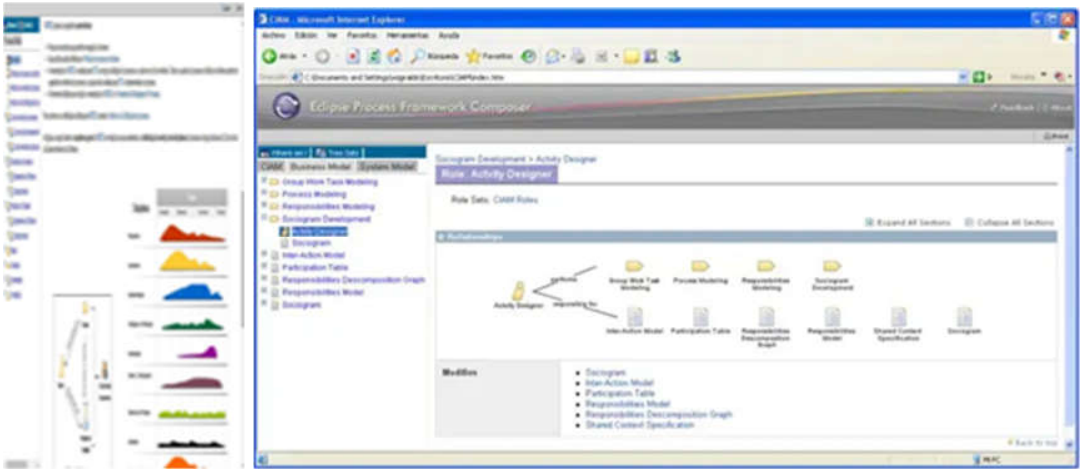
C - Já leu algum(s) processo(s) modelado(s) com SPEM

D - Já apoiou a modelagem de algum(ns) processo(s) com SPEM

E - Já modelou/espeficicou algum(ns) processo(s) com SPEM

DICAS

Para facilitar o reconhecimento, veja as seguintes imagens que representam a especificação ou ferramentas que atuam com o SPEM.



(https://www.researchgate.net/profile/Orlando_Avila-Garcia/publication/242284818/figure/fig2/AS:393258376155137@1470771511303/Figure-2-SPEM-Software-Process-Engineering-Metamodel-20-model-describing-our.png)

Introdução

Nesta seção serão apresentadas algumas definições e orientações para apoiar sua participação neste estudo de entrevista.

Este trabalha tem como perspectiva avaliar situações sobre processo de desenvolvimento de software e o modelo que o representa, que correspondam ao processo utilizado no dia-a-dia organizacional. Não tem como propósito avaliar as estruturas de "modelos de processos gerais, ou paradigmas de processos" como waterfall (modelo cascata), modelo espiral, modelos iterativos e incrementais, modelos ágeis e entre outros.

Definição de *Process Smell*

Process smell são resultados das decisões tomadas durante as fases como: concepção, modelagem e a produção do modelo do processo que incidem resultados sobre elementos do processo em particular que venham a reduzir a qualidade de fatores esperados para um processo. Os *process smells* podem incidir na redução da qualidade de diferentes fatores inseridos no processo, propriamente dito, ou que são associados ao processo.

Entretanto para este trabalho serão verificados somente os impactos aos seguintes subatributos de qualidade:

A - Compreensibilidade avalia a facilidade com que o executante do processo consegue compreender o processo;

B - Capacidade de modificabilidade reflete a facilidade em que o processo pode ser modificado. Este atributo de qualidade envolve a avaliação de esforço e impactos necessário para realizar uma mudança no processo.

Especificação SPEM

SPEM 2.0 é uma especificação construída para conter os elementos mínimos necessário para representar processos de desenvolvimento de software e sistemas. Esta especificação se destina a conseguir representar uma grande quantidade de modelos de desenvolvimento e processos, com diferentes representações, fatores culturais, formalismos e estilos. O SPEM compõe também um framework conceitual que apoia modelar, documentar, apresentar, gerenciar, interagir tendo como base métodos e processos de desenvolvimento. E é composto por elementos, que colaboram entre si, como atividades, regras, produtos de trabalho. (OMG; NOTATION, 2008).

Atividade

Uma atividade é uma unidade básica de trabalho dentro de um Processo e representa um agrupamento de elementos como tarefas, Papéis, Milestones, etc.

Tarefa

Uma tarefa é uma unidade de trabalho atribuível, que define trabalho executado por papéis específicos. Uma tarefa é associada aos produtos de trabalho de Entrada (Input) e Saída (Output). A granularidade de uma definição de tarefa geralmente é de algumas horas a alguns dias. Geralmente, afeta um ou apenas um pequeno número dos produtos de trabalho. Uma tarefa tem um propósito claro em que as funções de execução alcançam um objetivo bem definido. Ele fornece uma explicação completa do passo-a-passo de todo o trabalho que precisa ser feito para alcançar esse objetivo.

Passos

Um Passo define um seção e trabalho que é usada para organizar uma tarefa em partes ou subunidades de trabalho. Um Passo descreve uma parte significativa e consistente do trabalho geral descrito para uma tarefa. A coleção de passos definidos para uma tarefa representa todo o trabalho que deve ser feito para atingir o objetivo geral de desenvolvimento da tarefa. Os passos podem estar organizados sob a forma de "fluxos" de trabalho, assim, nem todos precisam necessariamente ser executados quando é realizada um tarefa no Processo.

Produto de Trabalho (Artefato, Entregável, Resultados)

O produto de trabalho é um elemento que é usado, modificado e produzido ou consumido pelas tarefas. Existem três tipos de produtos de trabalho: Resultados, Entregáveis e Artefatos. Os Resultados definem produtos de trabalhos intangíveis ou que não são formalmente definidos. Os Entregáveis empacotam outros produtos de trabalhos. E um Artefato é um produto de trabalho tangível que pode ser composto de outros artefatos.

Os produtos de trabalho são considerados como de Entrada (Input) ou de Saída (Output). Os de Entrada são insumos que precisam estar concluídos junto a execução da tarefa que necessita destes insumos. Já os produtos de trabalho de Saída (Outpus) são resultantes de uma tarefa.

Observação: Para este trabalho serão considerados produtos de trabalho na sua forma essencial, não abstraindo os tipos citados, levando apenas em consideração se eles se tratam de produtos de trabalho de Entrada (Input) ou de Saída (Output).

Associação/Transição entre atividades e/ou tarefas

Esta associação reflete o fluxo em que as atividades e/ou tarefas do processo são executadas.

Associação/Transição entre atividades/tarefas para produtos de trabalho

Esta associação reflete o fluxo em que são realizadas as atividades/tarefas gerando ou sendo associadas aos produtos de trabalho do processo.

Bifurcação/União concorrente

A representação de bifurcação/união caracteriza fluxos paralelos. Essa representação é iniciada e finalizada por uma linha, que possui uma única entrada e mais de uma saída, contendo no interior ações que ocorrem paralelamente e os fluxos concorrentes são sincronizados, assim cada um aguarda a conclusão dos demais fluxos para que a união entregue o resultado em um mesmo momento.

Inicialização

A inicialização representa o começo das interações do processo.

Conclusão

A conclusão representa o final das interações do processo.

Decisão/ramificação

Representa os caminhos alternativos que o fluxo de decisões em que o processo pode seguir com base em alguma expressão booleana, exemplo sim ou não para algum estado ou ação verificada.

Instruções

As perguntas a seguir devem ser avaliadas da seguinte forma:

Ler o enunciado que explica determinado *process smell* proposto

Opinar, se realmente, cada *process smell* proposto, pode ser caracterizado como um problema sobre o ponto de vista de processo de software.

C1

Questão 1

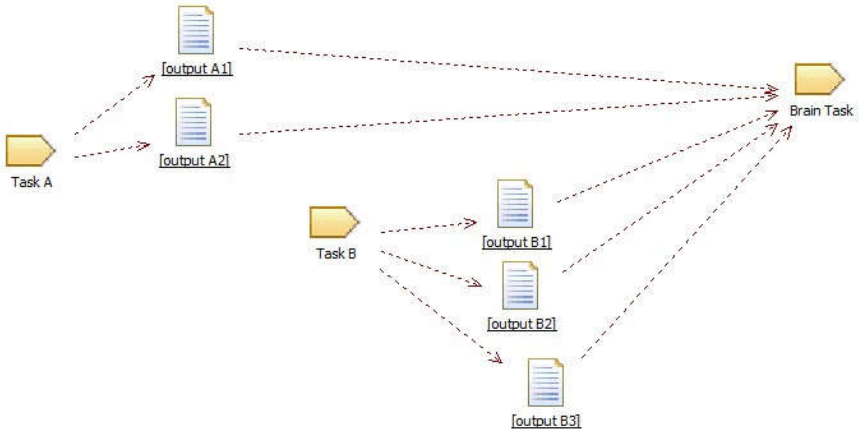
Process Smells: Brain Task

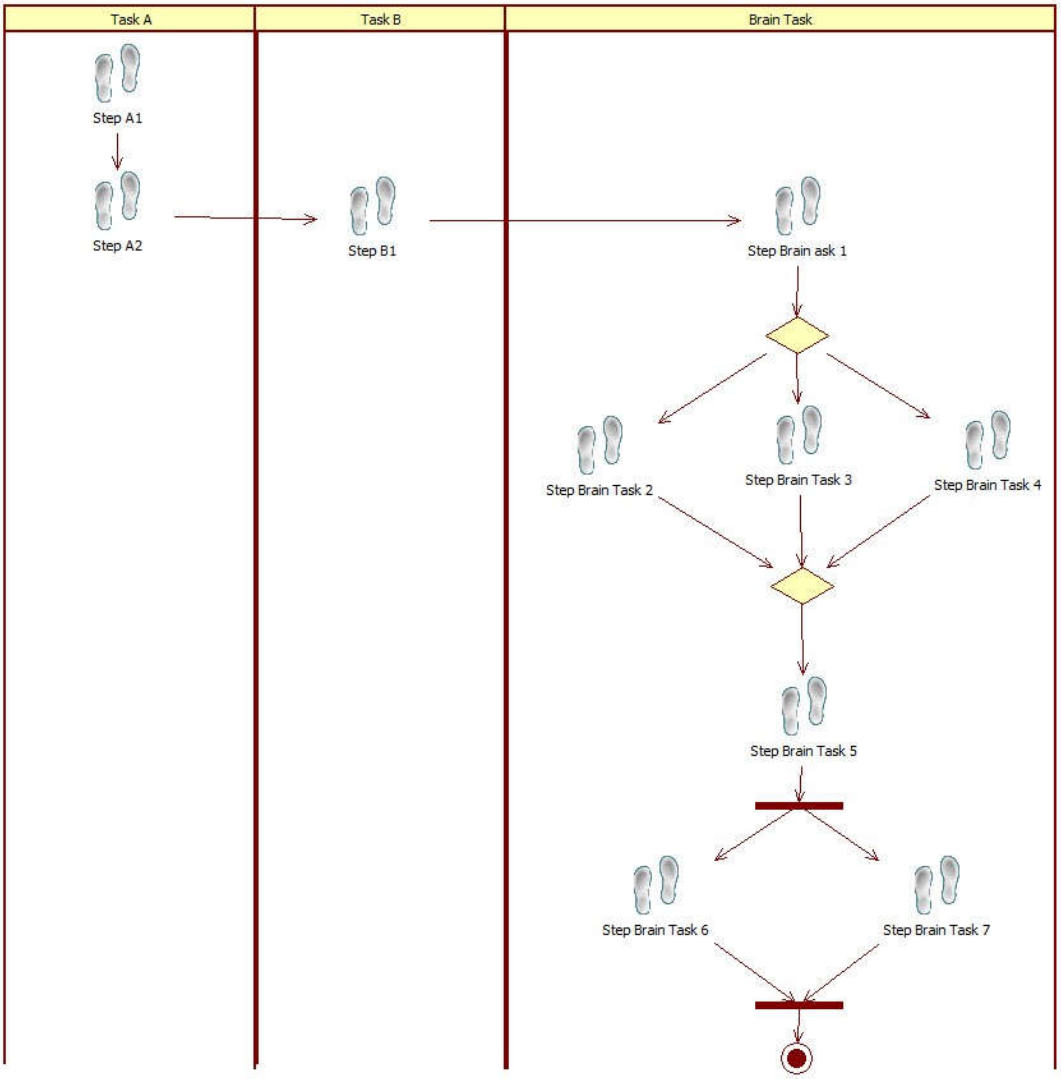
Elemento impactado: Tarefa

Descrição

Uma Brain Task é uma tarefa longa que centraliza as ações de uma atividade, possui muitos passos e possui muitos fluxos de decisão para estruturar os passos.

Representação


[output A1]
[output A2]
Task A
Brain Task
Task B
[output B1]
[output B2]
[output B3]


Task A
Task B
Brain Task
Step A1
Step A2
Step B1
Step Brain ask 1
Step Brain Task 2
Step Brain Task 3
Step Brain Task 4
Step Brain Task 5
Step Brain Task 6
Step Brain Task 7

Possíveis Impactos

Apesar da sensação de simplificar o modelo usando poucas tarefas, uma Brain Task tem muitos passos e muitos fluxos de decisão, o que aumenta a complexidade e redução a compreensibilidade da tarefa que sofre esse *process smell*.

Você concorda uma tarefa que possui muitos passos e possui muitos fluxos pode ser considerada um *process smells*?

* Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A - Sim

B - Não

Caso negativo comente aqui sua escolha.

Você concorda com esse *process smells* afeta negativamente o atributo de qualidade compreensibilidade?

* Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A - Sim

B - Não

Caso negativo comente aqui sua escolha.

C2

Questão 2

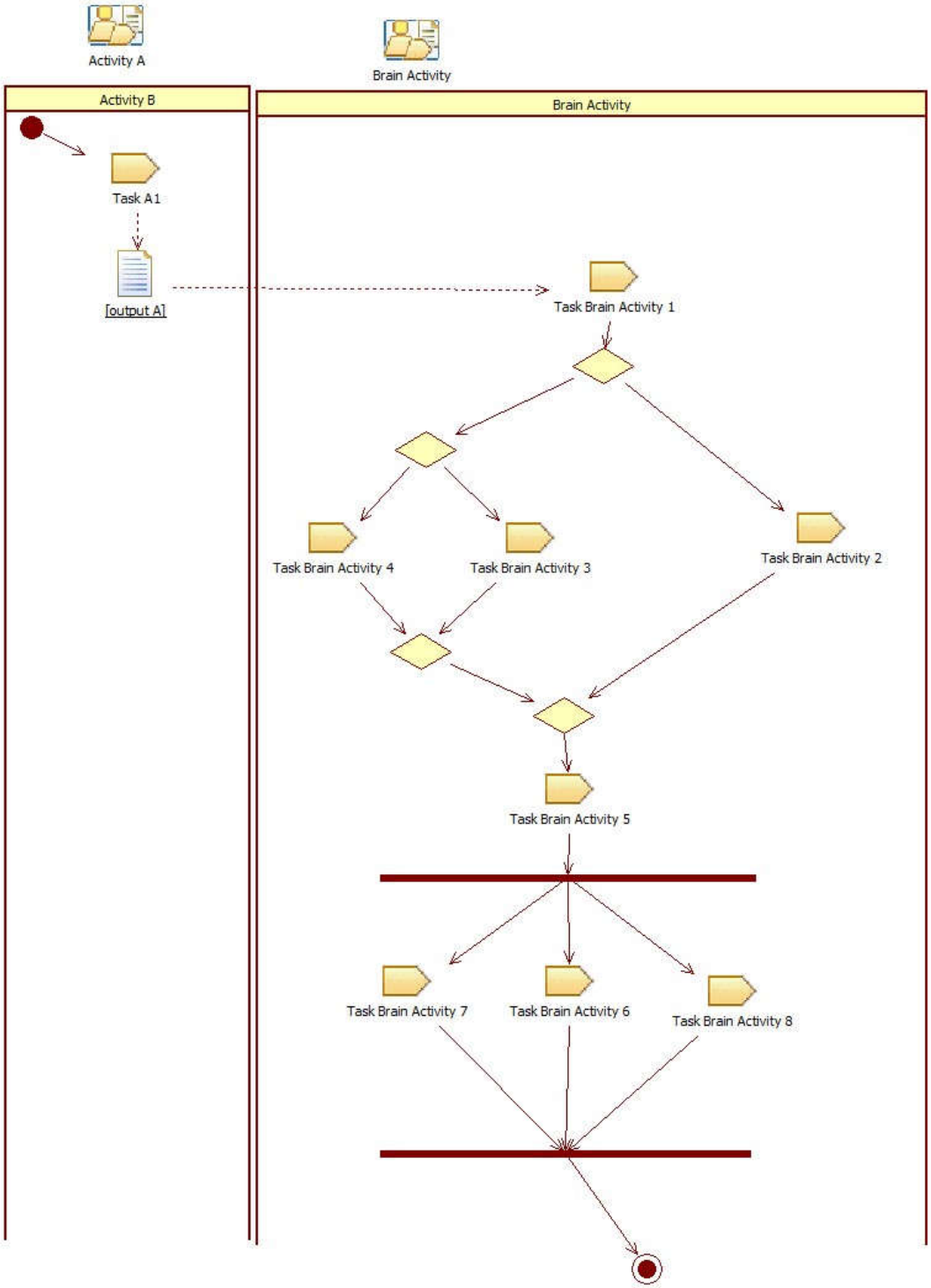
Process smells: Brain Activity

Elemento impactado: Atividade

Descrição

Uma Brain Activity é uma atividade que centraliza as ações de um Processo. Essa atividade possui a maior parte das ações relevantes para o processo. Possui muitas tarefas e muitos fluxos de decisão para estruturar as tarefas.

Representação



Possíveis Impactos

Atividades centralizadoras que concentram muitas tarefas e fluxos de decisão se tornam atividades grandes e complexas e podem reduzir a compreensibilidade do usuário do processo.

Você concorda que uma atividade que possui muitas tarefas e possui muitos fluxos de decisão pode ser considerada um *process smells*?

* Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A - Sim

B - Não

Caso negativo comente aqui sua escolha.

Você concorda com esse *process smells* afeta negativamente o atributo de qualidade compreensibilidade?

* Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A - Sim

B - Não

Caso negativo comente aqui sua escolha.

C3

Questão 3

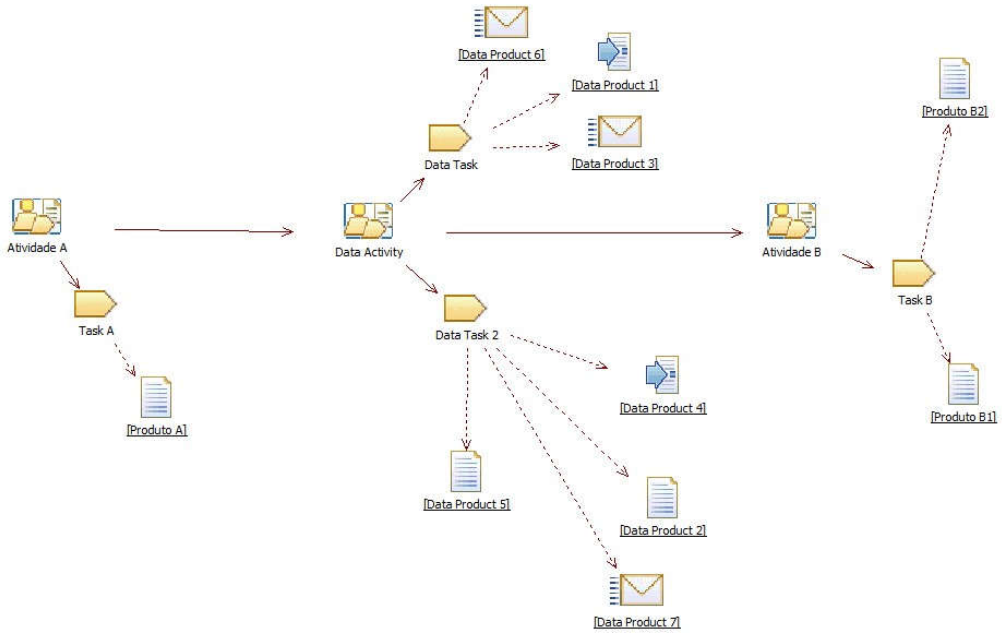
***Process Smells*:** Data Activity

Elemento impactado: Atividade

Descrição

Uma atividade se comporta como uma Data Activity quando não possui muitas tarefas, mas fornecem bastante produtos de trabalho de saída (Outputs).

Representação



Possíveis Impactos

Um agrupamento de muitos produtos de trabalho de Saída em uma mesma atividade pode originar tanto um acoplamento disperso (um relacionamento dependente entre muitas atividades) tanto um acoplamento intensivo (um relacionamento dependente entre poucas atividades), dificultando a capacidade de modificação do processo, uma vez que existem muitas dependências associadas a essa atividade.

Você concorda que uma atividade que não exerce uma função principal no processo e entrega bastante produtos de trabalho de saída podem ser considerada um *process smells*?

* Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A - Sim

B - Não

Caso negativo comente aqui sua escolha.

Você concorda com esse *process smells* afeta negativamente o atributo de qualidade capacidade de modificação?

* Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A - Sim

B - Não

Caso negativo comente aqui sua escolha.

C4

Questão 4

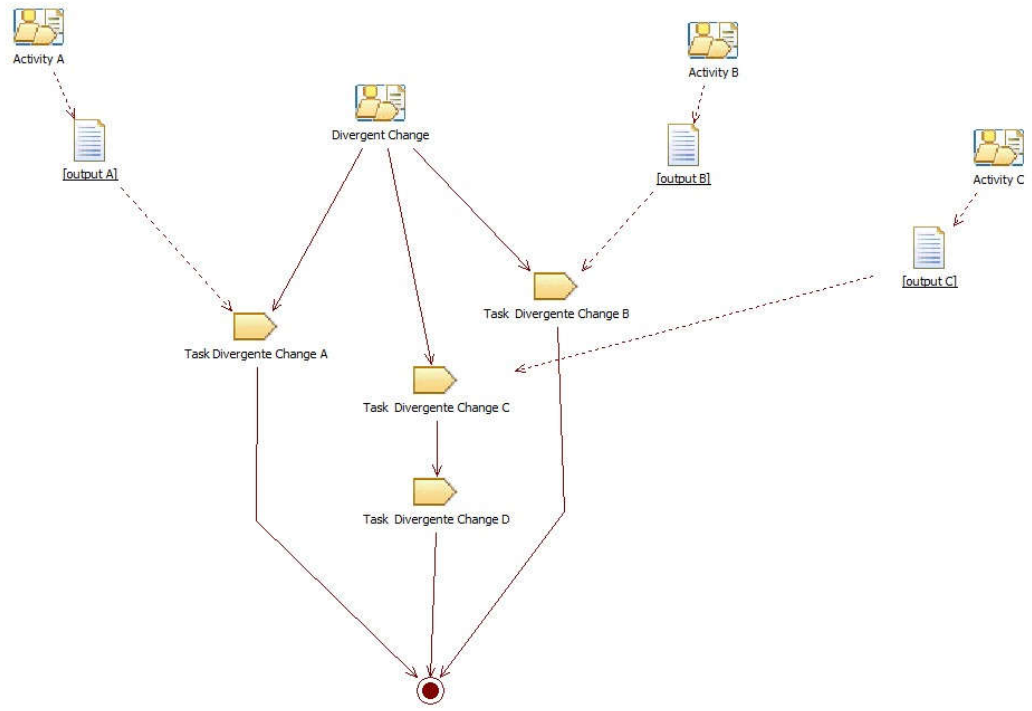
***Process Smells*:** Divergent Change

Elemento impactado: Atividade

Descrição

Uma atividade tem como propósito atender alguma "necessidade" do processo. Um exemplo de "necessidade" em um processo de software pode ser: construir o Termo de Abertura do projeto; validar um documento de especificação; entre outras. Assim, o Divergent Change ocorre quando uma atividade é configurada para atender a não somente uma "necessidade" do processo, tendo assim, diferentes conjuntos de tarefas para cada necessidade a qual atenderá. Em consequência, sempre que qualquer dessas "necessidades", que essa atividade atende sofrer mudanças, a parte das tarefas responsáveis por atendê-la também mudará, enquanto as outras tarefas que não fazem parte desse conjunto não sofrerão nenhuma mudança por este mesmo motivo.

Representação



Na representação acima, cada tarefa Divergent Change está respondendo a uma "necessidade" das atividades A, B e C respectivamente. Se uma dessas atividades mudar, a tarefa Divergent Change respectiva também terá que mudar.

Possíveis Impactos

Essa atividade se encarrega de diferentes propósitos, desta forma se torna uma atividade pouco coesa, ou seja, as tarefas dessa atividade realizam ações de forma independente das outras tarefas, o que pode prejudicar a compressibilidade dessa atividade.

Você concorda que uma atividade que é definida para atender a não somente uma "necessidade" do processo pode ser considerada um *process smells*?

* Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A - Sim

B - Não

Caso negativo comente aqui sua escolha.

Você concorda com esse *process smells* afeta negativamente o atributo de qualidade compreensibilidade?

* Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A - Sim

B - Não

Caso negativo comente aqui sua escolha.

C5

Questão 5

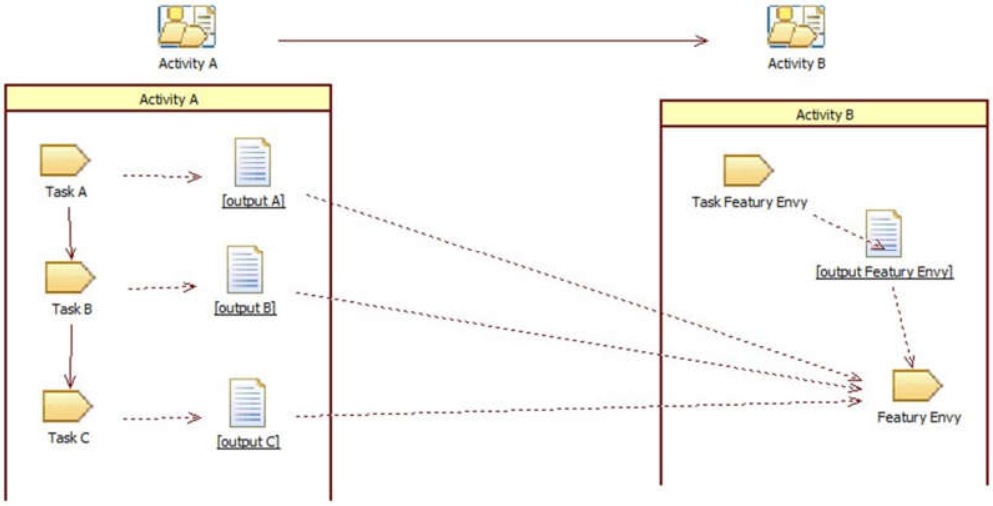
***Process Smells*:** Feature Envy

Elemento Impactado: tarefa

Descrição

O Feature Envy é representado por uma tarefa que fazem uso extensivo de produtos de trabalho de saída de outra atividade. Por esse motivo, talvez essa tarefa deve pertencer a outra atividade.

Representação



Possíveis Impactos

O Feature Envy é reconhecido pela distância que os produtos de trabalho de entrada têm em relação a tarefa que as irá utilizá-los, que normalmente estão em outra atividade e não na atividade a qual a tarefa pertence. Assim, quanto maior for a medida desta distância, menor será a compreensibilidade do executante do processo, principalmente para os casos, onde o executante terá que entender a tarefa fornecedora.

Você concorda que uma tarefa que faz uso extensivo de produtos de trabalho de saída de outra atividade pode ser considerada um *process smells*?

* Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A - Sim

B - Não

Caso negativo comente aqui sua escolha.

Você concorda com esse *process smells* afeta negativamente o atributo de qualidade compreensibilidade?

* Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A - Sim

B - Não

Caso negativo comente aqui sua escolha.

C6

Questão 6

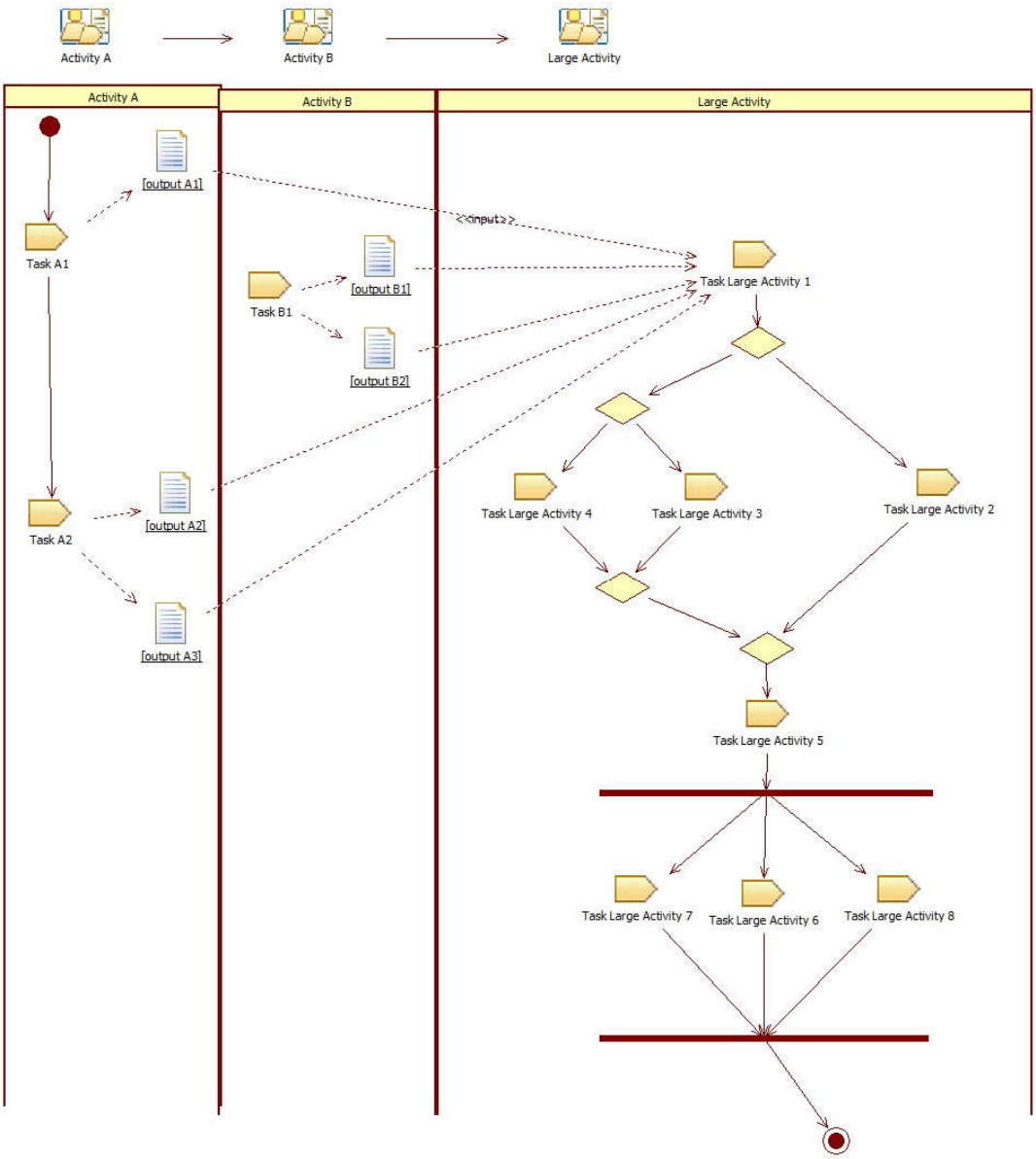
***Process Smells*:** Large Activity

Elemento impactado: Atividade

Descrição

Uma Large Activity se caracteriza quando uma atividade é grande, por conter muitas tarefas, e possui muitas responsabilidades. Ela centraliza as principais ações do Processo, deixando outras atividades com responsabilidades mais triviais e utilizando produtos de trabalho destas atividades para concluir seus objetivos.

Representação



Possíveis Impactos

Apesar da sensação de simplificar o modelo usando poucas atividades, uma Large Activity possui muitas tarefas e muitos fluxos de decisão, assim aumenta a complexidade. A inserção de muitas tarefas em uma mesma atividade gera o risco de confundir a responsabilidade da atividade, adicionando tarefas que cuidam de diferentes necessidades do processo. Este fato pode também

impactar negativamente a coesão entre as tarefas. Essas duas características podem reduzir a compreensibilidade da atividade.

Por fim, a dependência que uma Large Activity têm de produtos de trabalho de entrada insere a possibilidade de impactos negativos caso as atividades quais a Large Activity é dependente mudem. Este fato reduz a capacidade de modificabilidade do processo.

Você concorda que uma atividade com muitas tarefas, que possui muitas diferentes responsabilidades e faz uso de muitos produtos de entrada pode ser considerada um *process smells*?

* Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A - Sim

B - Não

Caso negativo comente aqui sua escolha.

Você concorda com esse *process smells* afeta negativamente o atributo de qualidade compreensibilidade?

* Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A - Sim

B - Não

Caso negativo comente aqui sua escolha.

Você concorda com esse *process smells* afeta negativamente o atributo de qualidade capacidade de modificação?

* Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A - Sim

B - Não

Caso negativo comente aqui sua escolha.

C7

Questão 7

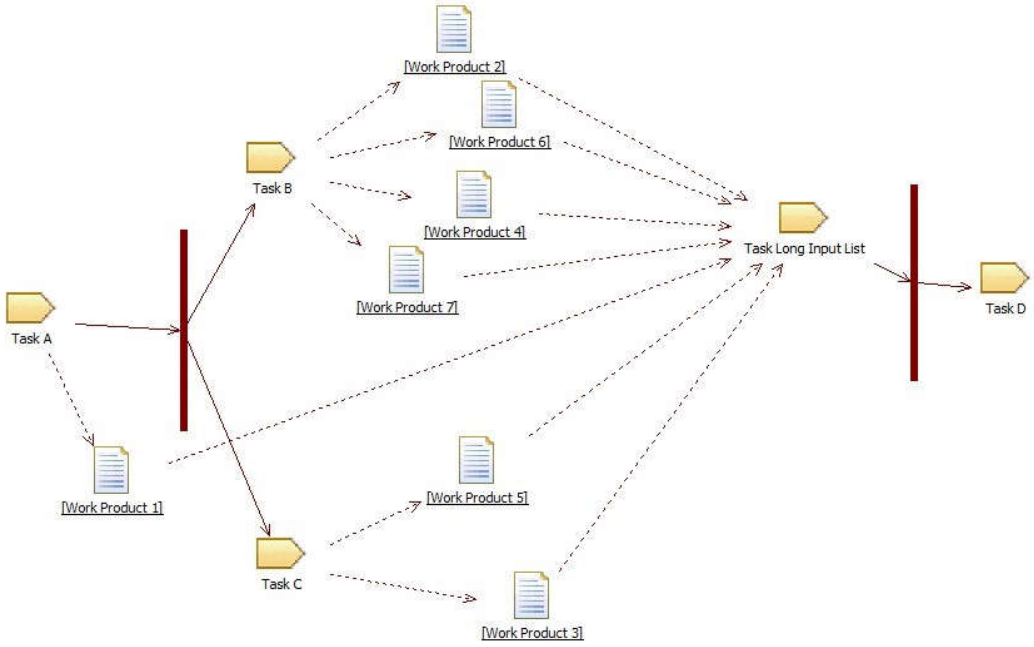
Process Smells: Long Input List

Elemento Impactado: Tarefa

Descrição

O Long Input List é configurado quando uma tarefa precisa de um grande número de produtos de trabalho de entrada.

Representação



Possíveis Impactos

A tarefa que possui uma longa lista de produtos de trabalho de entrada, pode estar sendo impactada por alguns problemas como: acúmulo de responsabilidades ou ser uma tarefa longa por possuir muitos passos e fluxos. E por sua vez pode prejudicar a compreensibilidade do usuário do processo.

Você concorda que uma tarefa que possui uma longa lista de produtos de trabalho de entrada pode ser considerada um *process smells*?

* Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A - Sim

B - Não

Caso negativo comente aqui sua escolha.

Você concorda com esse *process smells* afeta negativamente o atributo de qualidade compreensibilidade?

* Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A - Sim

B - Não

Caso negativo comente aqui sua escolha.

C8

Questão 8

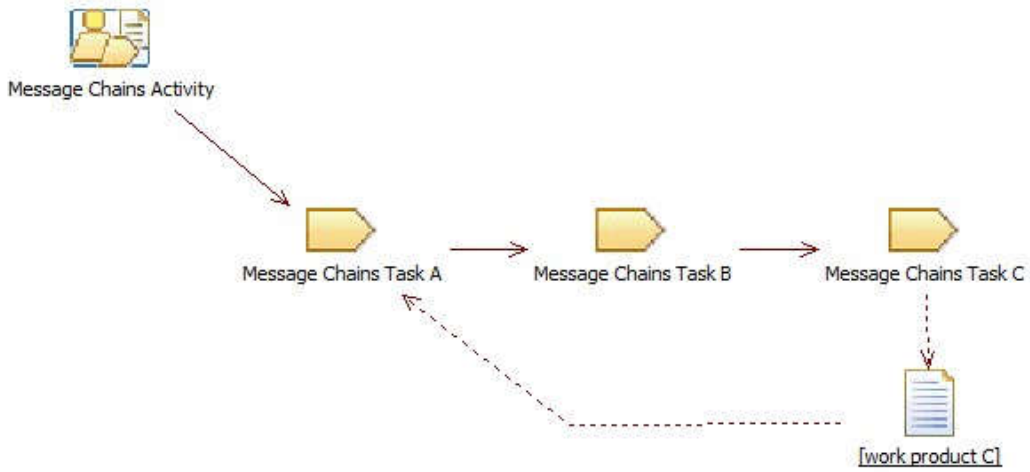
***Process smells*:** Message Chains

Elemento impactado: Tarefa

Descrição

O Message Chains define que longas cadeias de trocas de produtos de trabalho entre tarefas que podem representar dependências disfarçadas. Uma tarefa precisa de um produto de trabalho de entrada que para ser gerado precisa esperar por uma cadeia de outras tarefas serem executadas até que este produto de trabalho de Entrada esteja pronto.

Representação



Possíveis Impactos

Quanto mais longa for a cadeia de tarefas, maior será a dificuldade na legibilidade do processo, uma vez que aumenta a distância entre a tarefa solicitante e a tarefa que entregará os produtos de trabalho de entrada solicitados, afetando a compreensibilidade do processo.

Você concorda que uma cadeia longa de tarefas pode ser considerada um *process smells*?

* Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A - Sim

B - Não

Caso negativo comente aqui sua escolha.

Você concorda com esse *process smells* afeta negativamente o atributo de qualidade compreensibilidade?

* Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A - Sim

B - Não

Caso negativo comente aqui sua escolha.

C9

Questão 9

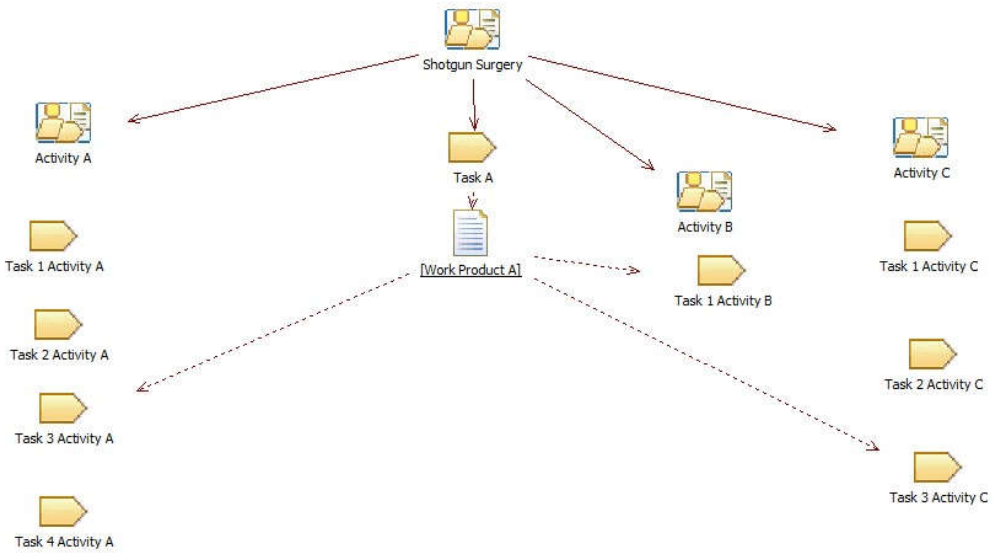
***Process Smells*:** Shotgun Surgery

Elemento impactado: Atividade

Descrição

O Shotgun Surgery se caracteriza por uma atividade que quando sofre uma mudança (em suas tarefas ou passos) desencadeará a necessidade de muitas pequenas mudanças em várias outras atividades.

Representação



Possíveis Impactos

Atividades com Shotgun Surgery ocasionam de acoplamento disperso no processo, ou seja, atividades dependentes que estão espalhadas pelo processo. Assim mudanças estruturais precisarão de maior esforço na rastreabilidade dos pontos que precisam ser ajustados, de forma a reestruturar o processo de forma consistente. Este fato reduz a capacidade de modificação.

Você concorda que uma atividade que ao passar por mudanças podem desencadear mudanças em muitas outras atividades pode ser considerada um *process smells*?

* Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A - Sim

B - Não

Caso negativo comente aqui sua escolha.

Você concorda com esse *process smells* afeta negativamente o atributo de qualidade capacidade de modificação?

* Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A - Sim

B - Não

Caso negativo comente aqui sua escolha.

C10

Questão 10

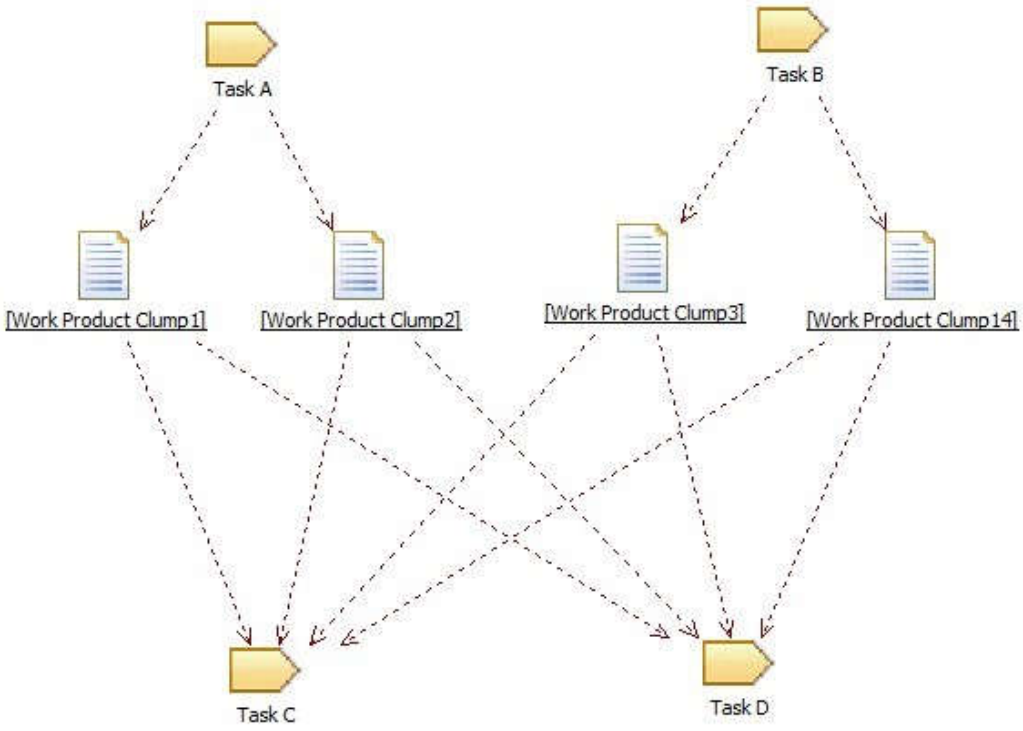
***Process Smells*:** Work Product Clumps

Elemento impactado: produtos de trabalho

Descrição

O Work Product Clumps acontece quando um conjunto de produtos de trabalho de entrada ou de saída de determinadas atividades/tarefas, são constantemente vistos juntos. Isso pode indicar que eles devessem pertencer a uma mesma atividade/tarefa.

Representação



Possíveis Impactos

O Work Product Clumps promove dispersão de um conjunto de produtos de trabalho, todos necessário para a conclusão de certas tarefas do processo. Essa dispersão reduz a capacidade de modificabilidade do processo, uma vez que será necessário um constante esforço de rastreabilidade desses produtos de trabalho dispersos sempre que houver alguma mudança nesse conjunto.

Você concorda que uma situação onde produtos de trabalho que deveriam estar agrupados estão dispersos pode ser considerado um *process smells*?

* Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A - Sim

B - Não

Caso negativo comente aqui sua escolha.

Você concorda com esse *process smells* afeta negativamente o atributo de qualidade capacidade de modificação?

* Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A - Sim

B - Não

Caso negativo comente aqui sua escolha.

Finalização

Nas perguntas a seguir expresse sua opinião sobre o trabalho apresentado.

1 - O questionário apresentou de forma clara o conceito de *process smells* e os *process smells* propostos? *

Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A - Não ficou clara a definição de *process smells* nem os *process smells* propostos

B - Ficou clara somente a definição de *process smells*

C - Ficou parcialmente claro os *process smells* propostos

E - Ficou claro todos os *process smells* propostos

D - Ficou claro tanto a definição, como os *process smells* propostos

2 - Em relação ao tema e conteúdo você considera que o questionário foi:

Escolha uma das seguintes respostas:Favor escolher apenas uma das opções a seguir:

A - Muito abrangente para entender e avaliar os *process smells* propostos

B - Abrangente para entender e avaliar os *process smells* propostos

C - Suficiente para entender e avaliar os *process smells* propostos

D - Extenso/cansativo para entender e avaliar os *process smells* propostos

E - Muito extenso/cansativo para entender e avaliar os *process smells* propostos

3 - Como você considera a proposição dos *process smells* em processo de software: *

Escolha uma das seguintes respostas: Favor escolher apenas uma das opções a seguir:

A - Pouquíssimo relevante

B - Pouco relevante

C - Relevante

D - Muito Relevante

E - Muitíssimo Relevante

4 - Acrescente abaixo quaisquer considerações que tenha pensado sobre este trabalho.

Por favor, coloque sua resposta aqui:

Obrigado pela sua participação.

Logo que tivermos resultados da pesquisa vocês serão participados.

Se houver alguma questão que não foi abordada por neste questionário e você considere relevante, ou se quiser indicar alguma pessoa dentro do perfil do público alvo para participar da pesquisa, pode entrar em contato por email com Edison Santos.

(mailto:ed_black1@hotmail.com).

Enviar questionário

Obrigado por ter preenchido o questionário.

Apêndice

C

QUESTIONÁRIO DO ESTUDO DE ENTREVISTA 2

Apêndice 1 Questionário do Estudo de Entrevista 2

Este apêndice apresenta a seguir o questionário aplicado no estudo de entrevista 2.

Definição de *Process Smell*

***Process Smell* são resultados das decisões tomadas durante as fases como concepção, especificação e a modelagem do processo que incidem resultados sobre elementos do mesmo que venham a reduzir a qualidade de fatores esperados para um processo.**

Subatributos de qualidade de processo de software

A - Compreensibilidade: reflete a facilidade com que o executante do processo consegue compreender o processo;

B - Modificabilidade: reflete a facilidade em que o processo pode ser modificado. Este atributo de qualidade envolve a avaliação de esforço e impactos necessário para realizar mudanças no processo.

Conceitos

coesão - representa as relações dos elementos do mesmo tipo e localização (exemplo tarefas de uma mesma atividade) entre si e acontecem a partir da troca de produtos de trabalho. Elementos coesos tendem a ter um propósito similar ou idêntico.

acoplamento - faz referência às relações de dependências entre elementos, acontecem quando uma atividade/tarefa recebe ou entrega produtos de trabalho para outras atividades/tarefas.

PT1 - Analisando a estrutura da “Atividade 5. Realizar testes funcionais e/ou integrados em funcionalidades com interface” quanto a coesão entre as tarefas e dependências das

atividades predecessoras, você acredita que a Atividade 5 pode colaborar para redução da compreensibilidade do processo?

Localização da atividade:

Processo de Testes -> Atividade 5. Realizar testes funcionais e/ou integrados em funcionalidades com interface

Você concorda com a pergunta acima?

Configuração do *process smell*

Elemento impactado: Atividade

Descrição

Uma Atividade tem como propósito atender alguma "necessidade" do processo. Um exemplo de "necessidade" em um processo de software pode ser: construir o Termo de Abertura do projeto; validar um documento de especificação; entre outras. Assim, o Divergent Change ocorre quando uma Atividade é configurada para atender a não somente uma "necessidade" do processo, tendo assim, diferentes conjuntos de Tarefas para cada necessidade a qual atenderá.

Possíveis Impactos

Essa atividade se encarrega de diferentes propósitos, desta forma se torna uma atividade pouco coesa, ou seja, as tarefas desta atividade realizam ações de forma independente das outras tarefas, o que pode prejudicar a compressibilidade dessa atividade.

Você concorda que essa estratégia de detecção está correta?

Heurísticas	1. Possui alto acoplamento com atividades predecessoras	2. Atividade pouco coesa
Métricas e valores limiares	NPD(A)>=Alto (90%)	RCT(A)<=Baixa (70%)

Você concorda que estes valores limiares estão adequados?

NPD(A)	RCT(A)
5	0,83

Valores Limiares (Função de Distribuição de Frequência)

NPD(A)	NSTPWieghtCum %
0	8.1
1	18.1
2	31.5
3	53.7
5	63.8
9	94.0
21	100.0

RCT(A)	NSTPWieghtCum %
0	30.2
0.8	53.0
1.8	66.4
6.7	71.8
19.1	79.2
33.3	88.6
66.7	91.3
100	100

PT2 - Analisando a estrutura da "Tarefa 3.3 Criar casos de testes ou não, caso não sejam enviados pelos clientes" em relação a quantidade de fluxos de decisão e passos, você acredita que a Tarefa 3.3 pode colaborar para redução da compreensibilidade do processo?

Localização da tarefa:

Processo de Testes -> Atividade 3. Elaborar casos de teste relevantes -> Tarefa 3.3 Criar casos de testes ou não, caso não sejam enviados pelos clientes

Você concorda com a pergunta acima?

Configuração do *process smell*

Elemento impactado: Tarefa

Descrição

Uma Brain task é uma Tarefa longa que centraliza as ações de uma Atividade, possui muitos Passos e possui muitos Fluxos de Decisão para estruturar os Passos.

Possíveis Impactos

Apesar da sensação de simplificar o modelo usando poucas tarefas, uma Brain Task tem muitos passos e muitos fluxos de decisão, o que aumenta a complexidade e redução a compreensibilidade da tarefa que sofre esse *process smell*.

Você discorda que essa estratégia de detecção está correta?

Heurísticas	1. Possui muitos passos.	2. Possui muitos fluxos de decisão.
Métricas e valores limiares	NSTP(T)>=Alto (90%)	NG(T)>=Alto (90%)

Você discorda que estes valores limiares estão adequados?

NSTP(T)	**NG(T)**
4	3

Valores Limiares (Função de Distribuição de Frequência)

NG(T)	NSTPWieghtCum
0	49.0
1	85.9
2	97.3
3	100.0

NSTP(T)	NSTPWieghtCum
1	20,1
2	69,8
3	91,9
4	100,0

PT3 - Analisando a estrutura da “Atividade 4. Preparar ambiente de teste” quanto a quantidade de produtos de trabalho de saída. Você acredita que a Atividade 4 pode colaborar para redução da modificabilidade do processo?

Localização da atividade:

-> Processo de Testes -> Atividade 4. Preparar ambiente de teste

Você concorda com a pergunta acima?

Configuração do *process smell*

Elemento impactado: Produtos de trabalho

Descrição

Uma atividade se comporta como uma Data Activity não possui muitas tarefas, mas fornece bastante produtos de trabalho de saída (outputs).

Possíveis Impactos

Um agrupamento de muitos produtos de trabalho de saída em uma mesma atividade pode originar tanto um acoplamento disperso (um relacionamento dependente entre muitas atividades) quanto um acoplamento intensivo (um relacionamento dependente entre poucas atividades), dificultando a modificabilidade do processo, uma vez que existem muitas dependências associadas a essa atividade.

Você concorda que essa estratégia de detecção está correta?

Heurísticas	1. Possui poucas tarefas	2. A atividade fornece um grande número de produtos de trabalho de saída.
Métricas e valores limiares	NSTP(A)<=Baixo (70%)	NWPOut(A)>=Alto (90%)

Você concorda que estes valores limiares estão adequados?

NSTP(A-Tasks)	**NWPOut(A)**

11	12

Valores Limiares (Função de Distribuição de Frequência)

NSTP(A-Tasks)	NSTPWieghtCum %
1	17,4
2	22,1
3	32,2
4	38,3
6	49,7
7	63,8
11	77,2
16	100,0

NWPOut(A)	NSTPWieghtCum %
1	6,7
2	16,8
3	25,5
4	35,6

5	38,3
7	51,0
8	56,4
9	63,8
12	77,2
21	100,0

PT4 - Analisando a estrutura da “Atividade 4. Preparar ambiente de teste” em relação a quantidade de fluxos de decisão e de tarefas, você acredita que a Atividade 4 pode colaborar para redução da compreensibilidade do processo?

Localização da atividade:

Processo de Testes -> Atividade 4. Preparar ambiente de teste

Você concorda com a pergunta acima?

Configuração do *process smell*

Elemento impactado: Atividade

Descrição

Uma Brain Activity é uma Atividade que centraliza as ações de um Processo. Essa Atividade possui a maior parte das ações relevantes para o processo. Possui muitas Tarefas e muitos Fluxos de Decisão para estruturar as Tarefas.

Possíveis Impactos

Atividades centralizadoras que concentram muitas tarefas e fluxos de decisão se tornam atividades grandes e complexas e podem reduzir a compreensibilidade do usuário do processo.

Você discorda que essa estratégia de detecção está correta?

Heurísticas	1. Possui muitas tarefas	2. Atividade pouco coesa	3. Possui muitos fluxos de decisão
Métricas e valores limiares	NSTP(A)>=Alto (90%)	RCT(A)<=Baixa (70%)	NG(T)>=Alto (90%)

Você discorda que estes valores limiares estão adequados?

NSTP(A-Tasks)	**RCT(A)**	**NG(A)**
11	1,82	3

Valores Limiares (Função de Distribuição de Frequência)

NSTP(A-Tasks)	NSTPWieghtCum %
1	17.4
2	22.1
3	32.2
4	38.3
6	49.7
7	63.8
11	77.2
16	100.0

RCT(A)	NSTPWieghtCum %
0	30.2
0.8	53.0
1.8	66.4
6.7	71.8
19.1	79.2
33.3	88.6

66.7	91.3
100	100

NG(A)	NSTPWieghtCum %
0	62.4
1	76.5
2	79.2
3	100.0

PD1 - Analisando a estrutura da “Atividade 1. Analisar Atividades” quanto ao acoplamento que as outras atividades têm em relação ela, você acredita que a Atividade 1 pode colaborar para redução da modificabilidade do processo?

Localização da atividade:

Processo de Codificação -> Fluxo de Construção -> Atividade 1. Analisar Atividades

Você concorda com a pergunta acima?

Configuração do *process smell*

Elemento impactado: Atividade

Descrição

O Shotgun Surgery se caracteriza por uma Atividade que sofre uma mudança (em suas Tarefas ou Passos) que desencadeará a necessidade de muitas pequenas mudanças em várias outras Atividades.

Possíveis Impactos

Atividades com Shotgun Surgery ocasionam de acoplamento disperso no processo, ou seja, atividades dependentes que estão espalhadas pelo processo. Assim mudanças estruturais precisarão de maior esforço na rastreabilidade dos pontos que precisam ser ajustados, de forma a reestruturar o processo de forma consistente. Este fato reduz a modificabilidade.

Você discorda que essa estratégia de detecção está correta?

Heurísticas	1. Atividades com Shotgun Surgery ocasionam auto acoplamento eferente disperso
Métricas e valores limiares	NSD(A)>=Alto (90%)

Você discorda que este valor limiar é adequado?

NSD(A)
12

Valores Limiares (Distribuição de Frequência Acumulada)

NSD(A)	NSTPWieghtCum
0	8,1
1	18,1
2	31,5
3	53,7
4	86,6
9	94,0
12	100,0

PD2 - Analisando a estrutura da “Tarefa 2.1 Construir o componente utilizando os padrões e práticas adequadas” quanto a forma com que ela faz uso de produtos de trabalho, você acredita que a Tarefa 2.1 pode colaborar para redução da compreensibilidade do processo?

Localização da tarefa:

Processo de Codificação -> -Fluxo de Reuso > 2. Construir/Adequar componente para o reuso -> Tarefa 2.1 Construir o componente utilizando os padrões e práticas adequadas

Você concorda com a pergunta acima?

Configuração do *process smell*

Elemento impactado: Tarefa

Descrição

No Featury Envy uma Tarefa que fazem uso extensivo de Produtos de Trabalho de Saída de outras Atividades. Por esse motivo, talvez essa Tarefa deve pertencer a outra Atividade.

Possíveis Impactos

O Featury Envy é reconhecido pela distância que os produtos de trabalho de entrada têm em relação a tarefa que as irá utilizá-los, que normalmente estão em outra atividade e não na atividade a qual a tarefa pertence. Assim, quanto maior for a medida desta distância, menor será a compreensibilidade do executante do processo, principalmente para os casos, onde o executante terá que entender a tarefa fornecedora.

Você discorda que essa estratégia de detecção está correta?

Heurísticas	Possui mais que 1/3 do uso de produtos de trabalho de outras atividades	Possui dependências de atividades predecessoras	Possui dependência de poucas de outras atividades predecessoras
Métricas e valores limiares	EP1/3(T)=True (Existe)	NPD(A)<=Baixo (70%)	EP(T)<=Baixo (70%)

Você discorda que estes valores limiares estão adequados?

EP1/3(T)	NPD(T)	EP(T)
Existe	1	1

Valores Limiares (Distribuição de Frequência Acumulada)

NPD(T)	NSTPWieghtCum %
0	51.7
1	85.2
2	91.3
3	95.3
4	96.6
5	98.7
8	100.0

EP(T)	NSTPWieghtCum%
0	66.4
1	86.6
2	93.3

3	98.0
4	98.7
8	100.0

PD3 - Analisando a estrutura da “Tarefa 6.2 Codificar unidades de implementação” quanto a quantidade de produtos de trabalho de entrada, você acredita que a Tarefa 6.2 pode colaborar para redução da compreensibilidade do processo?

Localização da tarefa:

Processo de Codificação -> 1. Fluxo de Construção -> Atividade 6. Construir aplicativo -> Tarefa 6.2 Codificar unidades de implementação

Você concorda com a pergunta acima?

Porque?

Configuração do *process smell*

Elemento impactado: Tarefa

Descrição

O Long Input List é configurado quando uma Tarefa precisa de um grande número de Produtos de Trabalho de entrada.

Possíveis Impactos

A tarefa que possui uma longa lista de produtos de trabalho de entrada, pode estar sendo impactada por alguns problemas como: acúmulo de responsabilidades ou ser uma tarefa longa por possuir muitos passos e fluxos. E por sua vez pode prejudicar a compreensibilidade do usuário do processo.

Você discorda que essa estratégia de detecção está correta?

Heurísticas	Tarefa precisa de um grande número de produtos de trabalho de entrada
Métricas e valores limiares	NWPIn(T)>=Alto (90%)

Você discorda que estes valores limiares estão adequados?

NWPIn(A)
5

Valores Limiares (Distribuição de Frequência Acumulada)

NWPIn(T)	**NSTPWieghtCum %**
1	66.4
2	85.2

3	90.6
4	93.3
5	98.7
8	100.0

PD4 - Analisando os produtos de trabalho “Padrão de desenvolvimento” “Especificação” e “Padrão de Interface” dispostos juntos como produtos de trabalho de entrada nas tarefas “Tarefa 1.1 Identificar o que é necessário para construir o componente e indicar forma de desenvolvimento” e “Tarefa 7.1 Retirar dúvida durante codificação”, você acredita que pode eles podem colaborar para redução da modificabilidade do processo?

Localização dos produtos de trabalho:

Padrão de desenvolvimento, Especificação e Padrão de Interface

Obs.: Estes produtos de trabalho se apresentam como entradas das duas tarefas

-> Processo de Codificação -> Fluxo Construção -> Atividade 1. Analisar Atividades -> Tarefa 1.1 Identificar o que é necessário para construir o componente e indicar forma de desenvolvimento e

-> Atividade 7. Esclarecer dúvidas durante codificação -> Tarefa 7.1 Retirar dúvida durante codificação

Você concorda com a pergunta acima?

Porque?