MAPEAMENTO ENTRE REPRESENTAÇÕES DE PROCESSO E PROJETO DE SOFTWARE

Renata Mesquita da Silva Santos

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientador : Toacy Cavalcante de Oliveira

Rio de Janeiro

Dezembro de 2019

MAPEAMENTO ENTRE REPRESENTAÇÕES DE PROCESSO E PROJETO DE SOFTWARE

Renata Mesquita da Silva Santos

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientador: Toacy Cavalcante de Oliveira

Aprovada por: Prof. Toacy Cavalcante de Oliveira, D.Sc.
Prof. Guilherme Horta Travassos, D.Sc.
Prof. Geraldo Bonorino Xexéo, D.Sc.
Prof. Adriano Bessa Albuquerque, D. Sc.
Prof. Paulo Sérgio da Conceição de Alencar, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
DEZEMBRO DE 2019

Santos, Renata Mesquita da Silva

Mapeamento entre Representações de Processo e Projeto de Software/ Renata Mesquita da Silva Santos. – Rio de Janeiro: UFRJ/COPPE, 2019.

XV, 227 p.: il.; 29,7 cm.

Orientador: Toacy Cavalcante de Oliveira

Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2019.

Referências Bibliográficas: p. 161-172.

1. Processo de Software. 2. Projeto de Software. 3. Mapeamento. I. Oliveira, Toacy Cavalcante de. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Dedico este trabalho primeiramente à DEUS,
à minha mãe Edir e a Tiago (in memorian) que
estará para sempre em meu coração.

Agradecimentos

A Deus por acreditar que Ele é e sempre será a base de tudo. Meu sustento!

À minha querida mãe, por todo o apoio dado, não somente neste trabalho, mas em todas as minhas escolhas e iniciativas, acadêmicas, pessoais e profissionais; e a minha prima-irmã Carla Beatriz, pelo incentivo e apoio em momentos necessários. Aos meus queridos pais de coração, Arilise e Clóvis, que com muito carinho e apoio sempre estiveram ao meu lado me incentivando. À minha amiga-irmã Aline, minha amiga de toda vida, sempre ao meu lado, na alegria e na tristeza.

A Fábio, meu presente, que me apoiou na reta final e me deu forças para vencer mais essa etapa da minha vida.

Aos meus amigos, que compreenderam meus momentos de ausência e apoiaram as minhas escolhas. São tantos amigos que seria injusto citar nomes.

Aos amigos dessa caminhada de doutorado, dos grupos de pesquisa da COPPE, em especial aos membros do meu grupo de pesquisa PRISMA, e do grupo de pesquisa em Engenharia de Software Experimental, por todas as contribuições para o desenvolvimento pessoal e troca de conhecimentos. Não posso deixar de destacar os amigos Helvio, Luciana, Maria Alcileia, Raquel e Ulisses que não me permitiram desistir. Essa tese também é de todos vocês!

A todos os amigos e colegas de trabalho do Instituto Federal Fluminense que direta ou indiretamente contribuíram para a realização deste trabalho, em especial à equipe da Diretoria de Tecnologia da Informação e Comunicação. Aos meus bolsistas de iniciação científica Isaac, Leandro, Diego, Bianca, Rafael, José Gustavo e Matheus pelas contribuições e dedicação aos projetos de pesquisa conduzidos ao longo do doutorado.

Aos meus orientadores Toacy Oliveira e Fernando Brito e Abreu, que aceitaram o desafio de me orientar. E que desafio! Foram muitas reuniões e muito aprendizado! Obrigada por tudo e por tanto apoio durante esses anos, principalmente no período em que nem eu mesma acreditava que seria possível concluir! Toda compreensão, atenção e colaboração foram fundamentais para essa nossa conquista!

Aos professores da linha de Engenharia de Software do PESC, em especial ao professor Guilherme Travassos pela contribuição ao meu aprendizado e formação como pesquisadora.

Aos funcionários da secretaria do PESC, pelo apoio sempre presente, especialmente ao Gutierrez da Costa, Ricardo César e Cláudia e Solange.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

MAPEAMENTO ENTRE REPRESENTAÇÕES DE PROCESSO E PROJETO DE SOFTWARE

Renata Mesquita da Silva Santos

Dezembro/2019

Orientador: Toacy Cavalcante de Oliveira

Programa: Engenharia de Sistemas e Computação

As organizações de desenvolvimento de software buscam continuamente melhorar seus processos de desenvolvimento e manutenção de software, já que estes estão relacionados à qualidade dos produtos de software resultantes. A instanciação de elementos do domínio do processo de software em elementos do domínio do projeto de software envolve o uso de processos de software para apoiar o desenvolvimento de software. Por meio de estudos experimentais, foi possível observar que os processos de software não são explicitamente identificados em planos de projetos e/ou nos registros de execução das tarefas. Desta forma, o objetivo desta tese é explicitar o mapeamento entre as perspectivas de processo e projeto de software. Para apoiar este mapeamento é proposto um conjunto de operações de instanciação, como *link*, *split* e *merge*, que permitem mapear explicitamente os elementos utilizados para modelar o processo de software e os elementos usados para executar os projetos de software, reduzindo o distanciamento entre eles. Neste sentido, as operações de instanciação pretendem promover a transição entre as perspectivas, por meio do mapeamento entre as representações de processo e projeto de software. Foram realizados estudos para avaliar os benefícios do mapeamento entre as perspectivas na identificação do processo de software executado e na identificação de não conformidades entre o processo de software de referência ao longo do projeto de software e sua execução. Os resultados destes estudos apresentaram indícios dos benefícios do mapeamento na identificação do processo de software executado e de não conformidades.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.).

MAPPING BETWEEN PROCESS AND SOFTWARE PROJECT REPRESENTATIONS

Renata Mesquita da Silva Santos

December/2019

Advisor: Toacy Cavalcante de Oliveira

Department: Computer Science and Systems Engineering

Software development organizations continually seek to improve their software development and maintenance processes, since they are related to the quality of the resulting software products. Instantiating concepts from the software process domain into concepts of the software project domain, involves using software processes to support software development. After executing empirical studies, it was observed that software processes are not explicitly identified in project plans and/or task execution records. Thus, the purpose of this thesis is to make explicit the mapping between software process and software project perspectives. For this mapping this thesis proposes a set of instantiation operations, such as link, split and merge, which allow developers to explicitly map the elements used to model the software process to the elements used to execute the software projects. In this sense, instantiation operations aim to foster a smooth transition between perspectives, by means of mapping between process and software project representations. To evaluate the benefits of mapping perspectives in identifying the executed software process and identifying nonconformities between the reference software process throughout the software project and its execution, studies were conducted. Their results provide initial evidence of the benefits of mapping in reducing the complexity of executed software process models, in identifying the executed software process and nonconformities.

Sumário

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE SIGLAS

ALM	Application Lifecycle Management
BPMN	Business Process Modeling and Notation)
CFC	Control Flow Complexity
CMMI	Capability Maturity Model Integration
DCBD	Descoberta de Conhecimento em Banco de Dados
ECaM	Extended cardoso metric
ECyM	Extended cyclomatic metric
GQM	Goal, Question, Metric
LOC	Line of Code
MPS.BR	Melhoria do Processo de Software Brasileiro
OMG	Object Management Group.
PDM	Product Data Management
PICO	Population, Intervention, Comparison e Outcome
PMBOK	Project Management Body of Knowledge
PMI	Project Management Institute
PSEE	Process-centered Software Engineeringn Environments
SCM	Software Configuration Management
SIGAEPCT	Sistema Integrado de Gestão Acadêmica da Educação Profissional e Tecnológica
SPEM	Software Process Engineering Metamodel
SPO	Software Process Ontology
TAM	Technology Acceptance Model

1 Introdução

Este capítulo apresenta a motivação, contexto, problema e as questões de pesquisa, os objetivos a serem alcançados, a metodologia, contribuições esperadas e a organização deste trabalho de tese.

1.1 Motivação e Contexto

Organizações de desenvolvimento de software buscam constantemente o desenvolvimento de produtos de software com qualidade. Para isso, visam a melhoria de seus processos, uma vez que a qualidade do produto de software pode ser influenciada pela qualidade do processo que é utilizado em seu desenvolvimento e manutenção (OSTERWEIL, 1987; CUGOLA; GHEZZI, 1998; FUGGETTA, 2000). Esforços têm sido dedicados para o aumento da eficiência e efetividade do processo de desenvolvimento e manutenção dos produtos de software, com destaque para a criação de padrões de qualidade de processo, tais como a norma ISO/IEC 12207:2008 (ISO/IEC, 2008) e os modelos de maturidade CMMI - *Capability Maturity Model Integration* (CMMI PRODUCT TEAM, 2010) e MPS.BR - Melhoria do Processo de Software Brasileiro (SOFTEX, 2016a).

Processos de software são considerados importantes para o setor de desenvolvimento de software, pois orquestram atividades, pessoas e informações envolvidas no desenvolvimento do mesmo. Segundo Pressman (2010), processos de software organizam de forma lógica as diversas atividades técnicas e gerenciais que envolvem pessoas, métodos, ferramentas e artefatos, e restrições que possibilitem disciplinar, sistematizar e organizar o desenvolvimento e manutenção de produtos de software. Desta forma, os processos impõem consistência e estrutura a um conjunto de atividades (PFLEEGER; ATLEE, 2010).

Segundo Diebold e Scherr (2017), a descrição do processo de desenvolvimento é um dos pré-requisitos para a melhoria sistemática do processo de software. A definição de processos padrão permite que, a partir da execução dos projetos, a organização adquira conhecimento sobre seus processos e que as melhorias implementadas possam ser percebidas por toda a organização (SOFTEX, 2016a). Este processo de software padrão pode ser adaptado para se adequar às particularidades do contexto de cada projeto (MARTÍNEZ-RUIZ *et al.*, 2012; KALUS; KUHRMANN, 2013).

Tipicamente, os processos de software padrão são usados para orientar as equipes de desenvolvimento durante a execução dos projetos e como base para o planejamento e monitoramento dos mesmos (BASTARRICA *et al.*, 2017). A instanciação de elementos do domínio do processo de software em elementos do domínio do projeto de software envolve o uso de processos de software para apoiar as atividades de desenvolvimento de software (DERNIAME; KABA; WASTELL, 1999; OMG, 2008).

De acordo com Reis (2003), a instanciação do processo modifica a especificação do processo, acrescentando informações detalhadas sobre os prazos, agentes e recursos utilizados de cada atividade definida no processo. A partir da instanciação, pode-se dar início à execução do processo de software, no qual as tarefas modeladas são realizadas tanto pelos desenvolvedores quanto automaticamente (REIS, 2003). Desta forma, as atividades definidas no processo de software geralmente são instanciadas, executadas e registradas na forma de tarefas em ferramentas de gerenciamento de projetos (BASTARRICA *et al.*, 2017). Estas ferramentas são capazes de registrar informações sobre a execução de processos nos projetos, como registros de data e hora, identificação de tarefas e as partes interessadas envolvidas na execução de uma atividade. Em consequência, as ferramentas são um apoio importante para a execução do processo de software durante os projetos de desenvolvimento.

Quando a execução dos projetos segue consistentemente a definição do processo, os resultados esperados tendem a ser produzidos dentro da qualidade, custo e tempo planejados (MEIDAN *et al.*, 2018). No entanto, durante a execução, os elementos de processo definidos podem ser alterados para, por exemplo, atender a situações não previstas, relacionadas a fatores, como prazo e gerenciamento de recursos (LAURENT *et al.*, 2014). Portanto, é impossível impor uma aderência rígida a um conjunto de regras e restrições (muitas vezes rigorosamente) predefinidas (FUGGETTA; DI NITTO, 2014), dando origem a um conjunto de inconsistências (ou "desalinhando"), como a não execução de uma atividade prevista ou a execução fora da ordem prevista de atividades.

Apesar da definição de um processo de desenvolvimento que deve ser seguido pelos projetos de desenvolvimento, é comum ver gerentes de projetos adotando seus próprios processos de desenvolvimento de software devido a preferências pessoais, experiências anteriores ou mesmo pressões de tempo e custo, e também por treinamento inadequado, falta de comprometimento e falta de comunicação entre os membros da equipe (LEMOS *et al.*, 2011). Neste sentido, desvios em relação ao processo de software padrão podem ocorrer durante a execução dos projetos, o que

em última análise, pode descaracterizar a proposta do processo inicial. De acordo com Rui *et al.* (2014) os desvios e inconsistências são principalmente resultados de processos incompletos e/ou da interação humana. Como resultado da ocorrência dos desvios, a qualidade dos produtos de software, tempo de entrega e os custos podem ser afetados (SMATTI; NACER, 2014).

Fuggetta e Di Nitto (2014) abordam que em automação e suporte de processo de software, é essencial fazer das inconsistências um "cidadão de primeira classe", ajudando a equipe de desenvolvimento a visualizá-las, monitorá-las e gerenciá-las, em vez de "lutar" para erradicá-las. Neste sentido se faz necessário a implementação de mecanismos que permitam identificar não conformidades que ocorrem durante a condução dos projetos, de forma a possibilitar o direcionamento de ações a partir desta identificação. Técnicas de análise de conformidade verificam se o comportamento observado registrado durante a execução corresponde a um comportamento modelado, neste sentido, este tipo de análise é crucial, porque muitas vezes as execuções de processos se desviam dos processos estabelecidos (LEONI; MAGGI; VAN DER AALST, 2012). Problemas e não conformidades, relacionadas à correção ou melhoria dos processos executados, são identificados durante a avaliação de Garantia da Qualidade do processo e se originam quando há desvios entre o esperado e o realizado (SOFTEX, 2016b).

Para realizar a identificação de não conformidades se faz necessário ter acesso ao processo que efetivamente está sendo executado nos projetos. A verificação de conformidade é essencial para assegurar que todos os projetos sejam realizados de acordo com o padrão, o que é importante para melhorar a qualidade dos processos dos projetos e aumentar o nível de maturidade do processo (HE *et al.*, 2009).

Os métodos de avaliação de processos existentes, embora amplamente utilizados, têm limitações tais como: dependência das competências dos avaliadores; grande quantidade de esforço e recursos necessários; subjetividade para analisar dados e julgar a implementação de práticas; baixa confiança na amostragem e sua representatividade (VALLE; SANTOS; LOURES, 2017). A análise de conformidade manual tende a ser demorada e propensa a erros, especialmente para o processo grande e complexo (HE *et al.*, 2009). Conforme observado por Chen, Hoi e Xiao (2011), normalmente, os avaliadores externos não podem acessar diretamente a informação devido a preocupações de segurança e privacidade e as avaliações são baseadas em avaliação subjetiva, além de que as avaliações de processos geralmente requerem avaliadores experientes para compreender processos específicos de software e preparar perguntas relevantes para entrevistas.

Conforme destacado por Cerdeiral e Santos (2018) para alcançar alta maturidade no processo de desenvolvimento de software, as organizações de software precisam analisar seus dados históricos e entender e controlar seus subprocessos críticos para obter conhecimento sobre sua estabilidade estatística, limites de desempenho e capacidade, e estabelecer metas de melhoria organizacional viáveis para esses subprocessos. Neste sentido, o acesso a dados do processo a partir das tarefas de projeto pode possibilitar, aos gerentes de projeto, o suporte a tomada de ações de implantação e/ou melhorias nos modelos de processo de software utilizado como referência nas organizações, contribuindo assim para a maturidade do processo.

1.2 Problema e Questão de Pesquisa

As organizações de desenvolvimento de software, que reconhecem a importância do processo para a qualidade do produto de software desenvolvido, definem um processo, o qual é utilizado como referência ao longo dos projetos de desenvolvimento. A cada projeto é feito o planejamento das tarefas, momento em que é criado um plano do projeto. Este plano é carregado para uma ferramenta de gerenciamento de tarefas, onde as tarefas são criadas, a partir desta etapa de criação do plano do projeto, dar-se-á início à execução. O registro destes dados em ferramentas de gerenciamento de projetos é classificado como sendo registros de execução (LEMOS *et al.*, 2011). Na execução as tarefas criadas são efetivamente realizadas por ferramentas ou por desenvolvedores.

A dificuldade em se identificar inconsistências entre os domínios de processo e projetos pode ser ilustrada na Figura 1, que exemplifica o problema utilizando representações tipicamente usadas em Engenharia de Software. Na Figura 1 à esquerda tem-se a Perspectiva de Processo, relacionada à definição do processo de software, especificando detalhadamente suas atividades, artefatos, papéis e relacionamentos entre estes. Geralmente, a Perspectiva de Processo é materializada na forma de um modelo de processo ou em linguagem natural. Os modelos de processo são representados utilizando linguagens de modelagem de processo, tais como SPEM (S*oftware Process Engineering Metamodel*) (OMG, 2008) e BPMN (*Business Process Modeling and Notation*) (OMG, 2014). À direita tem-se a Perspectiva do Projeto relacionada à instanciação e à execução do processo, onde as atividades previstas na definição do processo, em geral, são registradas na forma de tarefas em ferramentas que são utilizadas ao longo dos projetos de desenvolvimento de software. Como destacado no CMMI (CMMI PRODUCT TEAM, 2010) a

implementação e gerenciamento do processo definido para o projeto são tipicamente descritos em um plano de projeto.

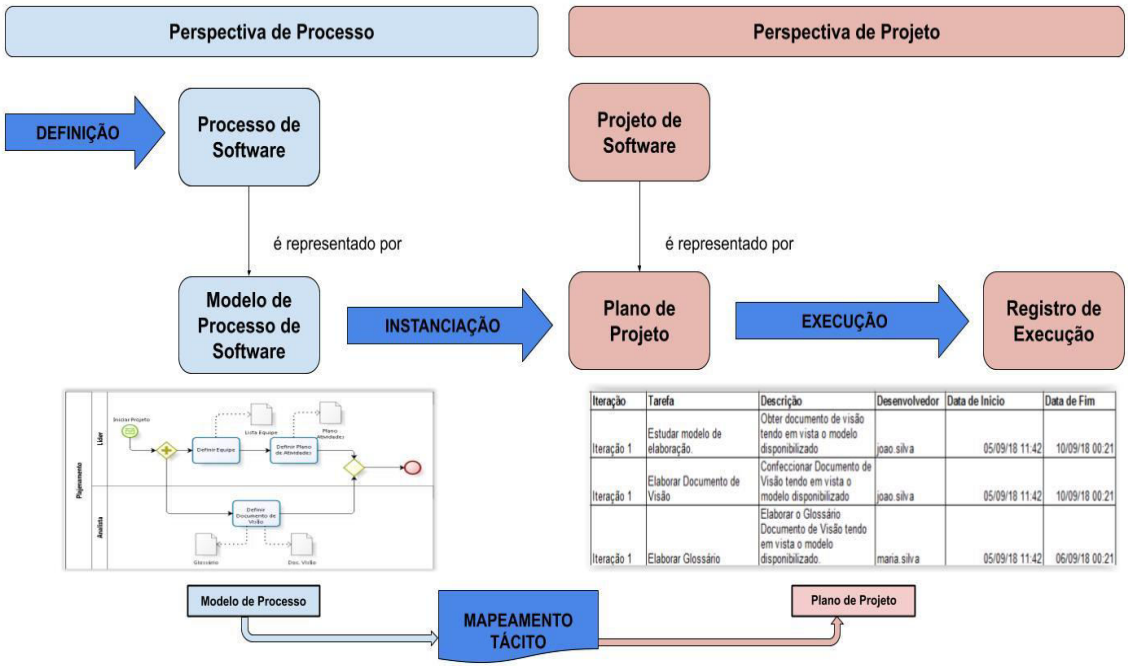


Figura 1 - Perspectivas de Processo e Projeto. Fonte: A Autora.

Embora um plano de projeto tenha que inevitavelmente materializar alguns conceitos, se não todos, prescritos pelos processos de software associados (DERNIAME *et al.*, 1999), existe a dificuldade de entendimento e visualização das atividades de desenvolvimento de software, definidas no processo de software, em um plano de projeto (WU; SIMMONS, 2000; BASTARRICA *et al.*, 2017).

Neste sentido, apesar do reconhecimento da importância do processo de software para a qualidade do produto e dos esforços para sua definição, existe uma lacuna entre as perspectivas de processo e projeto, que impacta na identificação do processo de software efetivamente seguido ao longo dos projetos. Sendo assim, a problemática que se evidencia nesta pesquisa é que os **processos de software não são explicitamente identificados em planos de projeto** e, portanto o alinhamento entre eles não é trivial.

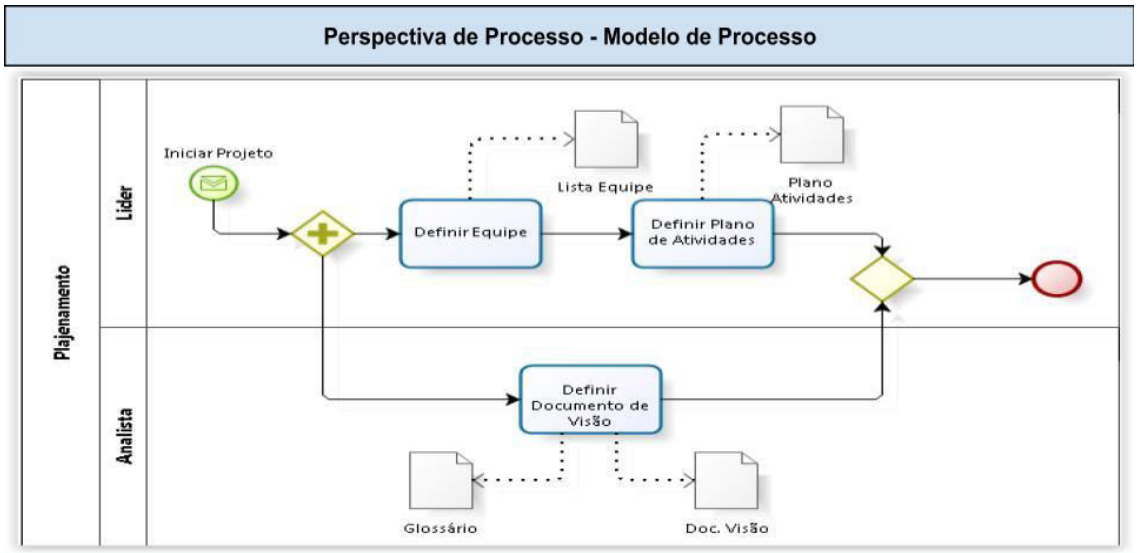
A Ontologia de Processo de Software (Software Process Ontology – SPO) desenvolvida originalmente em Falbo e Bertollo (2009) foi construída com o objetivo de estabelecer uma conceituação comum para as organizações de software falarem sobre processo de software (BRINGUENTE; FALBO; GUIZZARDI, 2011). Apesar, de a SPO considerar a relação entre processo de software e o projeto de software (BRINGUENTE; FALBO; GUIZZARDI, 2011; SEON, 2019), nem sempre tornar esse mapeamento explícito é trivial.

De acordo com Valle, Santos e Loures (2017), devido à crescente utilização de sistemas de informação para apoiar a execução do processo, informações detalhadas sobre a implementação de processos são registradas como registros de eventos, *logs* de transações, entre outros. Conforme observado por Ferreira, Szimanski e Ralha (2013) na prática, muitas vezes acontece de os registros de execução serem criados sem uma conexão explícita com a atividade do processo que está sendo executada, além do fato de que cada atividade do processo pode gerar mais de uma tarefa. Os autores destacam que existe claramente uma lacuna entre os processos como são definidos e os registros como são gerados durante a execução do processo nos projetos. Neste sentido, Ruiz-Rube, Dodero e Colomo-Palacios (2015) destacam que as ferramentas de suporte para projetar, verificar, validar, implantar e avaliar processos, muitas vezes, não fornecem mecanismos para incluir definições explícitas dos processos, o que causa uma falta significativa de consistência entre os modelos de processo e a implantação nas ferramentas (RUIZ-RUBE; DODERO; COLOMO-PALACIOS, 2015).

De acordo com Leoni e Marrella (2017), às vezes, as atividades registradas (tarefas do projeto) não podem ser correspondidas a nenhuma das atividades definidas no modelo (atividades do processo). Conforme observado por Ferreira, Szimanski e Ralha (2013) na prática, muitas vezes acontece que os registros de execução que podem ser capturados pelas ferramentas são de baixa natureza, como, por exemplo, "*O agente X enviou uma mensagem M ao agente Y*".

Neste sentido, pode não ser fácil mapear os conceitos definidos no domínio de processo nos projetos, ou seja, identificar uma correspondência entre as atividades do processo nos planos de projeto e nos registros de execução não é uma tarefa simples (BASTARRICA *et al.*, 2017). Quando um processo de software definido é instanciado para um projeto de software sem assistência adequada, os conceitos de processos podem ser perdidos, principalmente considerando que durante a instanciação do processo, as atividades podem ser divididas, combinadas, removidas e adicionadas na forma de tarefas. Por exemplo, uma atividade presente no modelo de processo pode ser considerada complexa demais para ser realizada por um único desenvolvedor, ao considerar a experiência da equipe, impondo assim que várias tarefas sejam criadas no plano de projeto e vários desenvolvedores sejam alocados a essas. Este comportamento é ilustrado na Figura 2, onde durante a instanciação do processo, considerando a atividade do processo "*Definir Documento de Visão",* foram criadas três tarefas no plano de projeto: *Estudar modelo de elaboração, Elaborar Documento de Visão e Elaborar Glossário*. Desta forma, identificar se o projeto está em conformidade com a definição do processo exige um olhar atento do grupo de

Garantia da Qualidade. Levando-se em consideração que processos têm dezenas e até centenas de atividades e projetos têm um número parecido de tarefas, a verificação de conformidade para identificação de inconsistências é uma tarefa difícil.



Perspectiva de Projeto - Plano de Projeto

Iteração	Tarefa	Descrição	Desenvolvedor	Data de Inicio	Data de Fim
Iteração 1	Estudar modelo de elaboração.	Obter documento de visão tendo em vista o modelo disponibilizado	joao.silva	05/09/18 11:42	10/09/18 00:21
Iteração 1	Elaborar Documento de Visão	Confeccionar Documento de Visão tendo em vista o modelo disponibilizado	joao.silva	05/09/18 11:42	10/09/18 00:21
Iteração 1	Elaborar Glossário	Elaborar o Glossário Documento de Visão tendo em vista o modelo disponibilizado.	maria.silva	05/09/18 11:42	06/09/18 00:21

Figura 1 - Perspectivas de Processo e Projeto – Modelo de Processo e Plano de Projeto. Fonte: A Autora.

Na literatura identificamos os *Process-centered Software Engineering Environments*, ou PSEEs, que se destacam como uma ferramenta relacionada ao gerenciamento de processos de software. Como por exemplo, SPADE (BANDINELLI; DI NITTO; FUGGETTA, 1996), Taba Workstation (MONTONI *et al.*, 2006) e WebAPSEE (LIMA *et al.*, 2006). Estes PSEEs apoiam a definição de processos padrão organizacionais e a adequação desses processos a projetos específicos, com o objetivo de aumentar o controle e melhorar a qualidade dos produtos de software. Embora os PSEEs tenham em sua estrutura base o modelo de processo, a falta de uma linguagem de modelagem de processo padrão impede sua ampla aplicação (CHOU; LI, 2014). Além disto, cabe ressaltar que de acordo com Mohammed, Redouane e Bernard (2007) um importante problema encontrado nos PSEEs é que os processos de desenvolvimento de software estão sujeitos à evolução dinâmica

permanente e sem gerenciar a evolução da execução do processo, os PSEEs estão condenados a falhar em sua adoção na indústria de desenvolvimento de software.

Além disso, o Gerenciamento de Ciclo de Vida de Aplicativos (*Application Lifecycle Management* – ALM) é uma classe de produtos originários da indústria de software convencional e da comunidade de código aberto, que fornece suporte para as várias fases de um ciclo de vida de desenvolvimento de software (FUGGETTA; DI NITTO, 2014). Exemplos de ferramentas ALM são Microsoft Visual Studio Application Lifecycle Management[1], IBM Rational solutions for CLM[2] e MyLyn[3]. Embora não pareçam estar explicitamente conectados à pesquisa de processo de software, eles fornecem mecanismos para automatizar algumas tarefas (normalmente, compilar e testar) e conectar tarefas de gerenciamento a atividades de desenvolvimento de software (FUGGETTA; DI NITTO, 2014).

No entanto, de acordo com Bastarrica *et al.* (2017) as empresas usam ferramentas de gerenciamento de projetos para planejar, monitorar o progresso, simplificar relatórios e fornecer visibilidade da situação do projeto, permitindo que os gerentes de projeto mantenham o controle e, assim, tomem decisões com base nos dados reais de execução. Neste cenário, ferramentas de gerenciamento de projetos como *Redmine*[4] *Jira*[5] e *MSProject*[6], entre outras, são usadas para fornecer dados em tempo real sobre o desempenho e o progresso do projeto (BASTARRICA *et al.*, 2017).

Nesta tese, pretende-se apresentar um mecanismo para mapear as perspectivas de processo e projeto, com base em operações explícitas de instanciação que capturam a lógica de mapeamento entre os elementos utilizados para representar os modelos de processos e os planos de projetos. A falta de um mapeamento explícito entre as perspectivas do processo e do projeto dificulta a identificação de não conformidades, bem como do modelo de processo efetivamente executado nos projetos. Sendo assim, este mapeamento deve ser realizado entre os elementos utilizados para modelar o processo de software e os elementos usados para executar os projetos de software, minimizando assim o distanciamento entre as perspectivas, pois cria vínculos entre o processo e o projeto. Este mapeamento é efetuado por meio da aplicação de operações de instanciação, durante a criação de tarefas ao longo do projeto de software. As operações representam de forma explícita

---

[1]http://msdn.microsoft.com/library/fda2bad5(VS.100).aspx
[2]http://pic.dhe.ibm.com/infocenter/clmhelp/v4r0m1/index.jsp
[3]http://www.eclipse.org/mylyn/
[4]https://www.redmine.org/
[5]https://br.atlassian.com/software/jira
[6]https://products.office.com/pt-br/project/project-and-portfolio-management-software

o mapeamento entre os elementos do modelo de processo de software e as tarefas criadas ao longo do projeto de software.

Considerando que os registros gerados, ao longo dos projetos de software, são uma fonte de dados sobre os processos realizados nas organizações, apresenta-se a seguinte questão de pesquisa:

Como explicitar o mapeamento entre as perspectivas de processo e projeto de software?

Neste sentido, é conjecturado que "**A utilização de operações de instanciação torna explícito o mapeamento entre as representaçãoes de processo e projeto de software**".

1.3 Objetivos

Para responder a questão de pesquisa delineou-se como objetivo geral: **explicitar o mapeamento entre as perspectivas de processo e projeto de software**.

Para alcançar o objetivo geral, foram propostos os seguintes objetivos específicos:

- Evidenciar a ocorrência do problema no qual os processos de software não são explicitamente identificados em planos de projeto, por meio de estudos experimentais;
- Definir um mecanismo de apoio ao mapeamento entre as perspectivas de processo e projeto;
- Definir um conjunto de orientações para a realização do mapeamento entre as perspectivas no contexto de projetos de software;
- Viabilizar a identificação do processo de software executado, por meio da estruturação dos registros de execução;
- Viabilizar a identificação de não conformidades, a partir do mapeamento entre as perspectivas;
- Oferecer apoio ferramental ao mapeamento entre as perspectivas; e
- Realizar estudos experimentais para avaliar a solução definida e seus benefícios.

1.4 Metodologia de Trabalho

As etapas dessa pesquisa foram delineadas com base no caminho para um trabalho de pesquisa definido em (WAZLAWICK, 2009). A partir da definição do tema pesquisa a ser investigado as seguintes etapas foram conduzidas:

1. **Revisão *ad hoc* da literatura:** Uma vez definido como tema de pesquisa o mapeamento das perspectivas de processo e projeto de software foi realizada uma revisão *ad hoc* da literatura, que permitiu identificar trabalhos relevantes para a fundamentação do problema. Nesta revisão da literatura identificamos trabalhos que discutem a ocorrência de desvios durante os projetos de desenvolvimento de software como: Melli (1988); Cugola *et al.* (1995); Cugola *et al.* (1998); Mohammed *et al.* (2007); Silva *et al.* (2010); Yong e Zhou (2010); Silva *et al.* (2013); Smatti e Nacer (2014) e Rui *et al.* (2014). Outros trabalhos destacam que manter o plano do projeto baseado no modelo de processo não é trivial, como em: Frappier e Richard (2004); Killisperger *et al.* (2011); Cuadrado-Garcia *et al.* (2011); Friedrich e Bergner (2011) e Hummel e Heinrich (2013). Além disso, foi realizada uma fundamentação teórica para entendimento do ciclo de vida de processos de software, bem como o levantamento da definição de processos, selecionando autores por conveniência, para dar apoio à identificação de um conjunto de características de processo.
2. **Realização de estudos exploratórios:** O passo seguinte foi a realização de estudos exploratórios. Para isto foi realizada uma pesquisa de opinião com profissionais que atuam em projetos de desenvolvimento de software. Esta pesquisa de opinião teve como objetivo investigar se as características de processo, identificadas na literatura, estão presentes nos projetos de desenvolvimento de software. Também objetivamos investigar quais ferramentas são utilizadas, ao longo dos projetos de desenvolvimento de software, para identificar as estruturas que armazenam os dados de processo. Um estudo foi realizado a partir do conjunto de ferramentas identificadas. Detalhes dos resultados dessa pesquisa de opinião e do estudo de ferramentas podem ser encontrados no Capítulo 3, bem como no Apêndice A e B desta tese. Contudo é importante antecipar que a relação do projeto com o processo não é explícita, apesar de uma grande quantidade de dados ser registrada nas ferramentas, ao longo dos projetos de desenvolvimento de software. A revisão informal da literatura e a pesquisa de opinião levaram à identificação de alguns fatores que distanciam a definição do processo de sua execução, sendo

necessário implementar mecanismos que permitam o mapeamento entre o projeto de software (execução) e o processo de software (definição).

3. **Revisão da literatura:** A revisão se deu por meio de uma busca estruturada, que objetivou a aquisição de conhecimento relacionado ao estado da arte sobre propostas que tratam o problema trata nesta tese. O objetivo do mapeamento foi *analisar ferramentas, métodos, processos e técnicas de mapeamento entre o processo de software definido para o projeto e o processo de software executado* **com o propósito de** *caracterizar* **com respeito a** *capacidade de realizar a representação explícita do mapeamento*, **do ponto de vista de** *engenheiros de software*, **no contexto de** *projetos de desenvolvimento de software utilizados como referência para apresentar as ferramentas, métodos, processos e técnicas*. A abordagem de *snowballing* também foi realizada como forma de aumentar a abragência da revisão. A revisão da literatura possibilitou a identificação dos trabalhos relacionados a esta pesquisa. Estes trabalhos podem ser subdivididos em três tipos de mecanismos de mapeamento: Correspondência (*Matching*), Alinhamento (*Alignment*) e Mineração (*Mining*).
4. **Definição e Estruturação da Solução:** A partir da definição do objetivo de explicitar o mapeamento entre as perspectivas de processo de software e projeto de software, iniciou-se a elaboração da solução proposta nesta tese. A partir dos estudos exploratórios foi possível concluir que os processos de software não são explicitamente identificados em planos de projeto. Desta forma, a falta de representação de informações da perspectiva de processo na perspectiva de projeto, ou seja, a falta de mapeamento explícito entre elas, pode impactar na identificação de não conformidades entre o processo definido para o projeto e o processo efetivamente executado, bem como na identificação do processo de software executado. Foi possível observar um comportamento que sugere que a instanciação, nestes casos, foi feita sem o registro do vínculo entre as atividades do processo de software e as tarefas do plano de projeto. A solução proposta define um conjunto de Operações de Instanciação para realizar o mapeamento entre as perspectivas de processo e projeto. As operações realizam o mapeamento explícito entre as perspectivas de processo e projeto, por meio de uma estrutura de representação que realiza o registro da atividade do processo no plano de projeto. Como forma de apoiar a definição e estruturação das operações de instanciação, entrevistas foram realizadas com gerentes de projeto e um estudo de observação em uma reunião de planejamento de tarefas. Também foi realizada uma busca

estruturada sobre operações de instanciação. Adicionalmente, um apoio ferramental foi desenvolvido para dar suporte ao mapeamento entre as perspectivas.

5. **Demonstração dos Benefícios na Identificação do Processo Executado e na Identificação de não Conformidades:** Conforme destacado por Baier *et al.* (2018), o mapeamento dos eventos produzidos (tarefas e registros de execução) com as atividades de um determinado modelo de processo é essencial para verificação de conformidade e entendimento dos resultados da mineração de processos. Neste sentido, nesta pesquisa são definidas orientações para apoiar a demonstração dos benefícios do mapeamento na identificação do processo de software executado e na identificação de não conformidades. A identificação do processo é realizada com a aplicação de técnicas de mineração de processos, considerando como entrada a base de dados de projeto gerada a partir da aplicação do mapeamento entre as perspectivas. Métricas de complexidade são utilizadas para demonstrar quantitativamente os benefícios na redução da complexidade dos modelos descobertos a partir de registros mapeados. A verificação de conformidade é realizada para identificar as não conformidades entre as perspectivas de processo e projeto. Não conformidades relacionadas às atividades previstas executadas, atividades previstas não executadas, e atividades não previstas executadas.
6. **Avaliação dos Benefícios na Identificação do Processo Executado e na Identificação de não Conformidades:** Estudos foram conduzidos para avaliar o benefício do mapeamento entre as perspectivas de processo e processo em apoiar a identificação do processo de software executado e de não conformidades. A avaliação seguiu as orientações definidas, que tratam da identificação do processo de software executado e de não conformidades. O estudo utilizou três bases de dados de projetos de desenvolvimento de software obtidas de duas empresas de desenvolvimento de software.

A viabilidade de utilização da metodologia foi analisada ao longo do desenvolvimento do trabalho com a realização de pequenos estudos de viabilidade das partes que o compõem.

1.5 Contribuições

Esta tese tem como principais contribuições:

- Consolidação de evidências da falta de mapeamento entre as perspectivas de processo e projeto;
- Revisão da literatura, por meio de uma busca estruturada, provendo informações relevantes sobre o mapeamento entre o processo de software e o projeto de software;
- Um conjunto de Operações de Instanciação para explicitar nos planos de projeto os elementos dos modelos de processo de software;
- Orientações para o mapeamento entre as perspectivas de processo e projeto no contexto de projetos de software;
- Orientações para demonstrar o benefício do mapeamento na identificação do processo de software executado e de não conformidades;
- Avaliação do benefício do mapeamento na identificação do processo de software executado e de não conformidades, utilizando bases de dados de projetos executados por empresas de desenvolvimento de software; e
- Ferramenta de apoio ao mapeamento entre as perspectivas, considerando a importação do modelo de processo em BPMN e a execução do projeto. A execução do projeto compreende a criação e finalização das tarefas.

1.6 Organização do Texto

Este trabalho está organizado em nove capítulos. O Capítulo 1, de Introdução, apresenta o contexto e o problema, os objetivos a serem alcançados e a organização do texto.

O Capítulo 2 apresenta a Fundamentação Teórica, com a descrição dos conceitos fundamentais sobre processos de software. As perspectivas de processo e projeto de software são discutidas, assim como uma breve apresentação da área de mineração de processos de software e de análise de conformidade. O Capítulo 3 apresenta os estudos exploratórios realizados para identificar a ocorrência do problema. No Capítulo 4 é apresentada a revisão da literatura, realizada por meio de uma busca estruturada, que objetivou a aquisição de conhecimento relacionado ao estado da arte sobre propostas que tratam o problema tratado nesta tese.

O Capítulo 5 apresenta a solução proposta para realização do mapeamento entre as perspectivas de processo e projeto. O mapeamento é realizado por meio das Operações de Instanciação. Neste capítulo as operações são descritas em detalhes. O Capítulo 6 descreve orientações para identificação do processo de software executado

e de não conformidades, a partir de registros de execução mapeados. O Capítulo 7 apresenta o apoio ferramental desenvolvido para dar suporte ao mapeamento entre as perspectivas, com a descrição das funcionalidades implementadas para apoiar a utilização da solução. O Capítulo 8 trata de apresentar os estudos conduzidos para avaliar o benefício do mapeamento entre as perspectivas de processo e processo. Esse capítulo apresenta os estudos conduzidos em bases de projetos de desenvolvimento de software de empresas.

O Capítulo 9 apresenta as conclusões, as principais contribuições alcançadas com o desenvolvimento deste trabalho e as limitações. Além disso, a enumeração de possíveis trabalhos futuros descreve desdobramentos da pesquisa.

2 Fundamentação Teórica

Neste capítulo são apresentados os fundamentos e conceitos básicos necessários para entendimento desta tese. Desta forma, os conceitos fundamentais sobre processos de software serão descritos. A área de Mineração de Processos de Software é descrita com o objetivo de apresentar os conceitos do processo de descoberta de conhecimento aplicado a processos de software. São apresentadas métricas de complexidade de modelos de processos de software. Por fim são apresentados os conceitos relacionados a análise de conformidade.

2.1 Processos de Software

Processos cada vez mais fazem parte da realidade das pessoas e das organizações, seja ao executar ações para alcançar um objetivo, seja no fornecimento de serviço ou criação de um produto. De acordo com Derniame e Kaba (1999) a qualidade do produto não pode ser simplesmente assegurada pela inspeção do produto ou pela realização do controle da qualidade estatística, porque a qualidade não é somente relacionada ao produto, mas também pela organização e produção do processo que é executado.

Os processos são importantes porque impõem consistência e estrutura a um conjunto de atividades (PFLEEGER; ATLEE, 2010). A estrutura do processo guia ações que permitem examinar, entender, controlar e melhorar as atividades que compõem o processo.

Esforços têm sido dedicados para o aumento da eficiência e efetividade do processo de desenvolvimento e manutenção dos produtos de software, com destaque para a criação de padrões de qualidade de processo, tais como a norma ISO/IEC 12207:2008 (ISO/IEC, 2008) e os modelos de maturidade CMMI - *Capability Maturity Model Integration* (CMMI PRODUCT TEAM, 2010) e MPS.BR - Melhoria do Processo de Software Brasileiro (SOFTEX, 2016a).

Para se disseminar a utilização de um processo em uma organização, é primordial que este processo esteja bem documentado e de acordo com Campos e Oliveira (2011), a modelagem de processos tornou-se um importante mecanismo para a compreensão do comportamento dinâmico das organizações. Existem várias notações ou métodos disponíveis para a modelagem de processos, e muitas outras ferramentas que utilizam estas notações (CAMPOS; OLIVEIRA, 2011). Pode-se

destacar a Notação de Modelagem de Processo de Negócios (BPMN - *Business Process Modeling Notation*).

O modelo BPMN define um diagrama de processo que é baseado em fluxogramas para a descrição das operações de trabalho. Esta é uma rede gráfica composta principalmente de atividades (trabalho) e setas (fluxo) (CAMPOS; OLIVEIRA, 2011). Atualmente, também em sua versão 2.0, o BPMN conta com uma notação extremamente rica em termos semânticos e um metamodelo que a define, sendo bastante difundido na prática de modelagem de Processos de Negócio (OMG, 2014).

Processos de software descrevem as diversas fases necessárias para produzir e manter um produto de software (PFLEEGER; ATLEE, 2010). Devido à importância da descrição de Processos de Desenvolvimento de Software, é requerido que estes tenham uma organização lógica das diversas atividades técnicas e gerenciais que envolvem agentes, métodos, ferramentas e artefatos, e restrições que possibilitem disciplinar, sistematizar e organizar o desenvolvimento e manutenção de produtos de software (PRESSMAN, 2010).

Um processo de desenvolvimento de software pode ser visto como um guia para o desenvolvimento e deve estabelecer: as atividades a serem realizadas durante o desenvolvimento de software, sua estrutura e organização (decomposição e precedência de atividades); os artefatos requeridos e produzidos por cada uma das atividades do processo; os procedimentos (métodos, técnicas e normas) a serem adotados na realização das atividades; os recursos necessários para a realização das atividades, dentre eles recursos humanos, recursos de hardware e recursos de software, incluindo as ferramentas a serem utilizadas para apoiar a aplicação dos procedimentos na realização das atividades; e roteiros para a elaboração dos principais documentos (artefatos) gerados no desenvolvimento (FALBO, 2000).

Um modelo de processo é caracterizado por conjunto de elementos de processo (atividades, artefatos, recursos e procedimentos), como visto em (VAN DER AALST, 2011; DUMAS *et al.*, 2013; WESKE, 2012) e também por um conjunto de restrições ao comportamento dos elementos e ao modelo de forma geral. Neste trabalho, os elementos de processo, e as restrições relacionadas a eles e ao modelo de processo, serão chamados de Características de Processo de Software. O levantamento das definições de processo, para definição deste conjunto de características, será apresentado na subseção a seguir.

2.1.1 Definições de Processos de Software

Processo é uma palavra com origem no latim *processus*, e de acordo com *Cambridge Dictionary Online* (CAMBRIDGE, 2015), processo é uma série de ações ou eventos realizados para fazer alguma coisa ou para alcançar um determinado resultado; uma série de mudanças que ocorrem naturalmente; ou também um método de fazer alguma coisa. A ISO/IEC 9000:2005 (ISO/IEC, 2005) define processo como um conjunto de atividades inter-relacionadas ou interativas que transformam entradas em saídas.

O termo processo é usado em muitos contextos diferentes (MÜNCH *et al.*, 2012). Na ISO/IEC 12207:2008 (ISO/IEC, 2008) é destacado que um projeto de software é dependente de vários resultados, produzidos por processos de negócio da organização. Neste sentido, para identificar um conjunto de características de processo de software, serão consideradas definições de processo relacionadas ao contexto de negócio e de software, uma vez que estes dois contextos são relacionados.

No contexto de negócios, destacam-se as definições de Eriksson e Penker (2000); Weske (2012) e Dumas *et al.* (2013). Eriksson e Penker (2000) definem processo como atividades realizadas dentro de um negócio, no qual o estado dos recursos do negócio muda. Estes autores também destacam que os processos descrevem como o trabalho é feito dentro do negócio, e que são regidos por regras. De acordo com Weske (2012), processos de negócios consistem em atividades, as quais a execução coordenada compreende algum objetivo do negócio. De acordo com Dumas *et al.* (2013), processo é uma cadeia de eventos, atividades e decisões. Dumas *et al.* (2013) também definem processos de negócio como uma coleção de eventos, atividades e pontos de decisão inter-relacionados, que envolvem uma série de atores e objetos, que conduzem coletivamente a um resultado, que é de valor para pelo menos um cliente.

No contexto do processo de software, ressaltam-se as definições de Osterweil (1987), Feiler e Humphrey (1993), Fugetta (2000), Falbo e Bertollo (2009), Pfleeger e Atlee (2010) e Münch *et al.*, (2012). Osterweil (1987) apresenta como noção elementar que um processo é uma abordagem sistemática para a criação de um produto ou a realização de alguma tarefa, para realizar o trabalho ou alcançar um objetivo em uma forma ordenada.

De acordo com Feiler e Humphrey (1993), processo é um conjunto de passos parcialmente ordenados destinado a atingir uma meta. Um processo de software pode ser definido como um conjunto coerente de políticas, estruturas organizacionais,

tecnologias, procedimentos e artefatos que são necessários para conceber, desenvolver, implementar e manter um produto de software (FUGETTA, 2000).

Falbo e Bertollo (2009) propuseram uma Ontologia de Processo de Software com o objetivo de definir um vocabulário comum no contexto de Processos de Software. Ontologias são usadas para descrever compromissos ontológicos para um conjunto de agentes (humanos ou aplicações de software), ou seja, acordos para se utilizar um vocabulário compartilhado de forma coerente, de modo que esses agentes possam se comunicar sobre o domínio em questão (BERTOLLO *et al.*, 2006). Um conjunto de propriedade de processo, elencadas a seguir foram definidas a partir de uma ontologia proposta por Falbo e Bertollo (2009):

- Uma atividade pode ser decomposta em outras atividades, chamadas subatividades;
- Atividades usam e produzem artefatos;
- Algumas atividades dependem da realização de outras atividades, chamadas pré-atividades;
- Artefatos podem ser decompostos em subartefatos;
- Alguns elementos são necessários para a realização de uma atividade, tais como as pessoas, hardware e software, estes elementos são chamados de recursos;
- Durante a definição de processo de software, é importante definir como as atividades serão realizadas, portanto, procedimentos devem ser estabelecidos para serem adotados na realização das atividades;
- Existem vários tipos de procedimentos;
- Métodos definem um conjunto de passos a serem executados em uma atividade;
- Ferramentas descrevem como usar uma ferramenta de software na realização de uma atividade;
- *Templates* são modelos a serem seguidos quando se prepara um artefato em uma atividade;
- Um método é um procedimento sistemático que define um fluxo de atividades (um conjunto de passos) e heurísticas para desempenhar uma ou mais atividades;
- Um modelo de documento visa estabelecer uma forma padrão para a preparação de alguns artefatos (neste caso, um documento);

- Um processo é definido para estabelecer uma abordagem sistemática para o desenvolvimento ou manutenção de software e pode ser decomposto em atividades ou em outros processos, chamados subprocessos;
- Os processos podem ser compostos por outros processos, chamados subprocessos e atividades;
- Um processo de software pode interagir com outros processos;
- Um modelo de ciclo de vida pode ser utilizado como referência;
- Um modelo de ciclo de vida estrutura as atividades do projeto em fases (ou macro-atividades), estabelecendo uma abordagem para organizar esses macro-atividades; e
- Um modelo de ciclo de vida define um conjunto de macro-atividades (ou fases) que um processo de desenvolvimento deve apresentar.

Pfleeger e Atlee (2010) afirmam que um processo é composto por um conjunto de etapas ordenadas que envolvem uma série de atividades, restrições e recursos com o intuito de produzir uma saída e descrevem um conjunto de características que todo processo deve apresentar:

- O processo prescreve todas as atividades principais;
- O processo utiliza recursos, sujeito a um conjunto de restrições, e produz produtos intermediários e finais;
- O processo pode ser composto de subprocessos que estarão relacionados de alguma forma. O processo pode ser definido como uma hierarquia de processos, organizados de forma que cada subprocesso possa ter o seu próprio modelo de processo;
- Cada atividade do processo tem um critério de entrada e de saída, de forma que se saiba quando a atividade é iniciada e terminada;
- As atividades são organizadas em uma sequência determinada;
- Todo processo tem um conjunto de princípios que explicam os objetivos de cada atividade;
- Restrições ou controles podem ser aplicados a uma atividade, recurso ou produto.

Münch *et al.* (2012) destacam que algumas características típicas dos processos de software, tais como: executados no mundo real; usualmente transforma um ou mais produtos de entrada em um ou mais produtos de saída por meio do consumo de outros produtos (por exemplo, orientações); podem ser executados por seres humanos ou por máquinas, ou pelos dois juntos; e podem ser refinados por subprocessos, os quais podem também ser refinados.

Esta subseção apresentou as definições que embasam a definição do conjunto de características de processo de software utilizadas nesta tese. A subseção a seguir apresenta as perspectivas de processo e projeto de software.

2.1.2 Perspectivas de Processo e Projeto de Software

Dentro de uma empresa ou de um domínio de aplicação, os processos de diferentes projetos tendem a seguir padrões comuns, ou porque as melhores práticas são informalmente reconhecidas ou porque a comunidade os elevam à situação de padrões. Assim, faz sentido tentar capturar esses pontos em comum em uma representação de processo, que descreve esses recursos comuns e promove a homogeneidade cultural da comunidade (DERNIAME; KABA, 1999).

Processos são definidos caso a caso, levando em consideração características específicas do projeto em questão, incluindo as tecnologias envolvidas, o tipo de software a ser desenvolvido, o domínio da aplicação e a equipe de desenvolvimento (ROCHA; MALDONADO; WEBER, 2001).

Em Rocha, Maldonado e Weber (2001), é apresentado um modelo para definição de processos de software composto de três níveis: definição de um processo padrão, especialização do processo padrão e instanciação para projetos específicos.

De acordo com Eman, Melo e Drouin (1997), o processo padrão descreve os elementos fundamentais que devem ser incorporados em qualquer processo definido na organização e as relações entre esses elementos, como sequência e interfaces. O Processo de Software Padrão deve ser definido a partir de padrões de qualidade de processo, tais como a norma ISO/IEC 12207:2008 (ISO/IEC, 2008) e de modelos de maturidade CMMI (CMMI PRODUCT TEAM, 2010) e MPS.Br (SOFTEX, 2016a).

Tendo em vista que tipos de software diferentes possuem características distintas, o processo de software padrão da organização poderá ser especializado para considerar algumas classes de software (sistemas embarcados, aplicações Web), paradigmas (por exemplo, Orientado a Objetos – OO) ou domínios de aplicação (BERTOLLO; SEGRINI; FALBO, 2006). O processo de software especializado mais indicado para um determinado contexto pode, ainda, ser instanciado para um projeto específico (BERTOLLO; SEGRINI; FALBO, 2006), uma vez que cada projeto da organização possui características específicas, tais como, modelos de ciclo de vida, características do projeto, características da equipe, disponibilidade de recursos, requisitos de qualidade do produto, entre outros (ROCHA; MALDONADO; WEBER, 2001). Resultam como produtos destes níveis propostos por Rocha, Maldonado e Weber (2001) processos de software em diferentes níveis de abstração.

Derniame e Kaba (1999) destacam que processos podem ser representados em níveis de detalhes, nos quais normalmente as descrições mais genéricas, como os modelos de ciclos de vida, são, em grande parte, informais, e no outro extremo, os processos de apoio operacional precisam de uma descrição formal executável em um nível detalhado, e no meio, há uma série de níveis intermédios. Os seguintes níveis são apresentados por Derniame e Kaba (1999):

- Um ciclo de vida é uma representação livre, informal de um processo de software. Exemplos bem conhecidos são os modelos de ciclo de vida Cascata e Espiral. O propósito destes modelos é tratar de questões metodológicas globais, por exemplo, especifique antes da codificação (a abordagem em cascata) ou avaliar os riscos antes de especificação (abordagem espiral). Essas representações são aplicadas a muitas empresas e domínios de aplicação.
- Um modelo de processo genérico também é uma representação abstrata, que pode ser usado em muitos projetos e organizações similares, compartilhando propriedades e características comuns. Um exemplo de modelo de processo genérico pode ser uma representação formal de um ciclo de vida, ou de um processo de reengenharia de software.
- Um modelo de processo customizado é mais detalhado, e é geralmente derivado de um modelo genérico levando em consideração as características locais específicas, por exemplo, o domínio do aplicativo. Normalmente existem diferentes níveis de personalização, relacionado ao aumento do nível de detalhe e formalidade.
- Um modelo de processo executável é o nível mais detalhado de customização, que define o processo a seguir em um projeto específico, sendo destacado que tem a perspectiva de ser uma representação pronta para ser carregada em uma máquina, a fim de fornecer suporte automático para pessoas envolvidas no processo de desenvolvimento.
- Finalmente, um modelo de processo de execução é um modelo de processo em vigor, suportando o desenvolvimento real, para isso, é mantido algum tipo de representação do estado atual de desenvolvimento.

Um modelo de processo é caracterizado por conjunto de elementos de processo (atividades, artefatos, recursos e procedimentos), como visto em Dumas *et al.* (2013), Van Der Aalst (2011) e Weske (2012), e também por um conjunto de restrições ao comportamento dos elementos e ao modelo de forma geral.

Os modelos de processo de software prescrevem e formalizam um conjunto de elementos e a maneira que esses elementos se inter-relacionam no desenvolvimento

de software. Uma vez definidos, estes modelos permitem às organizações de desenvolvimento de software analisar, melhorar e otimizar a forma como as pessoas executam suas tarefas de desenvolvimento de software (SILVA *et al.*, 2013). Sendo assim, a definição explícita de processos representa um papel chave nas principais iniciativas de melhoria de processos de software (RUIZ-RUBE; DODERO; COLOMO-PALACIOS, 2015), tornando, segundo Diebold e Scherr (2017), a descrição do processo de desenvolvimento um dos pré-requisitos para a melhoria sistemática do processo de software. Para tanto, existem notações ou métodos disponíveis para a modelagem de processos, e ferramentas que suportam estas notações (CAMPOS; OLIVEIRA, 2011), tais como SPEM (*Software Process Engineering Metamodel*) (OMG, 2008) e BPMN (*Business Process Modeling and Notation*) (OMG, 2014). Estas notações são definidas por meio de um metamodelo e fornecem representações para conceitos que são os elementos de processos de software, como atividades, artefatos, papéis e os relacionamentos entre eles.

Modelos de processo de software, muitas vezes, não se destinam a transmitir uma compreensão de todo o processo, mas apenas de um ponto de vista particular de interesse, exemplos típicos são (DERNIAME; KABA, 1999):

- o modelo de atividade tem como foco os tipos, estrutura e propriedades das atividades do processo e suas relações: este ponto de vista é relevante, por exemplo, para o gerente de projeto para fins de planejamento;
- o modelo do produto (artefatos) descreve os tipos, estrutura e propriedades dos itens de um processo de software; este ponto de vista pode ser de interesse também para o usuário, por exemplo, para compreender que tipo de documentação será entregue com o sistema de software;
- o modelo de recurso descreve os recursos necessários ou fornecidas ao processo, é relevante do ponto de vista gerencial;
- o modelo do papel descreve um conjunto peculiar de recursos, ou seja, as habilidades dos executores e as responsabilidades que desempenham, é relevante para a organização e para o pessoal de garantia da qualidade, como também para outros executores.

O uso de processos de software para apoiar o desenvolvimento de software envolve a instanciação de elementos do domínio do processo de software em elementos do domínio do projeto de software (DERNIAME; KABA; WASTELL, 1999; OMG, 2008). De acordo com Reis (2003), as organizações frequentemente adotam estratégias específicas para instanciar processos, através de modelos de custos e ferramentas de gerência de projetos para obter melhores resultados econômicos.

Reis (2003) destaca que as ferramentas de gerência de processos diferem dos mecanismos convencionais de escalonamento de projetos, com respeito a questões específicas, que reforçam a necessidade de suporte adequado à fase de instanciação, tais como:

- Os processos de software não são completamente definidos antes da execução, já que execução e modelagem são tarefas que podem ocorrer simultaneamente. Portanto, a instanciação pode ocorrer durante a execução e depende de decisões rápidas para não prejudicar o andamento do processo;
- As características das pessoas, incluindo habilidades e experiências passadas, assim como informações relativas aos recursos, como métricas e disponibilidade, influenciam de forma decisiva na realização do processo. Portanto, o conhecimento sobre essas informações deve estar disponível para que a instanciação seja realizada adequadamente;
- A natureza dinâmica dos processos pode influenciar a alocação de recursos e pessoas. Os gerentes frequentemente devem realocar processos em decorrência de modificações dinâmicas. Por isso, a escolha de estratégias de alocação deve levar em consideração o estado corrente da execução do processo assim como a disponibilidade atual e futura (esperada) dos recursos e agentes.

Segundo Reis (2003), a instanciação do processo modifica a especificação do processo, acrescentando informações detalhadas sobre os prazos, agentes e recursos utilizados por cada atividade definida no processo. Como destacado no CMMI (CMMI PRODUCT TEAM, 2010) a implementação e gerenciamento do processo definido para o projeto são tipicamente descritos em um plano de projeto. Um plano de projeto é um documento que define como o projeto será executado, monitorado e controlado (PMI, 2013). O desenvolvimento do plano do projeto deve levar em consideração as necessidades atuais e previstas, os objetivos e requisitos da organização, clientes, fornecedores e usuários finais (CMMI PRODUCT TEAM, 2010). A partir da instanciação, pode-se dar início à execução do processo de software, no qual as tarefas modeladas são realizadas tanto pelos desenvolvedores quanto automaticamente (REIS, 2003). Durante a execução, os elementos de processo definidos podem ser alterados para, por exemplo, atender a situações não previstas, relacionadas a fatores, como prazo e gerenciamento de recursos (LAURENT *et al.*, 2014).

Dessa forma, a partir dos elementos definidos no processo do software, as tarefas são criadas no plano do projeto. Assim, as atividades definidas no processo de software geralmente são instanciadas, executadas e registradas na forma de tarefas

em ferramentas de gerenciamento de projetos (BASTARRICA *et al.*, 2017). Essas ferramentas são capazes de registrar informações importantes sobre a execução de processos, como registros de data e hora, identificação de tarefas e as partes interessadas envolvidas na execução de uma atividade. O registro destas informações em ferramentas de gerenciamento de projetos é classificado como sendo registros de execução (LEMOS *et al.*, 2011).

A partir do exposto, é possível identificar a existência de duas perspectivas: a Perspectiva de Processo e a Perspectiva de Projeto. A Perspectiva de Processo está relacionada à definição do processo de software e geralmente é representada na forma de um modelo de processo ou em linguagem natural, esta trata da representação e organização de elementos de processo, bem como a definição de tecnologias e melhores práticas de desenvolvimento de software requeridas, ao passo que a Perspectiva do Projeto descreve um plano de projeto. A Perspectiva do Projeto está relacionada à instanciação e a execução do processo, onde as atividades previstas na definição do processo, em geral, são registradas na forma de tarefas em ferramentas que são utilizadas na condução de projetos de desenvolvimento de software.

A próxima seção descreve a área de Mineração de Processos de Software.

2.2 Mineração de Processos

De acordo com Cardoso e Machado (2008), a quantidade de dados disponíveis vem crescendo assustadoramente nos últimos anos e vários fatores contribuíram para esse incrível aumento. O baixo custo de armazenagem pode ser visto como uma das principais causas do surgimento dessas enormes bases de dados. Para que o conhecimento seja descoberto de forma eficiente, é realizado um processo chamado descoberta de conhecimento em banco de dados (DCBD – Descoberta de Conhecimento em Banco de Dados) (GOLDSCHMIDT; PASSOS, 2005).

As etapas para Descoberta de Conhecimento consistem em: selecionar os dados, preprocessá-los para retirar dados duplicados ou inconsistentes, transformá-los nos formatos adequados conforme algoritmos e ferramentas utilizadas, submeter os dados transformados à mineração propriamente dita, e, após, fazer a interpretação e validação dos resultados (FAYYAD *et al.*, 1996). O ponto central do processo de descoberta é a etapa de Mineração (GOLDSCHMIDT; PASSOS, 2005). A Mineração de Dados é amplamente difundida, porém, quando o resultado final desta etapa é a geração de Processo, esta etapa é chamada de Mineração de Processos.

Mineração de Processos é uma área de investigação, relativamente recente, que se posiciona não só entre Inteligência Computacional e Mineração de Dados, mas também entre a modelagem e análise de processos (VAN DER AALST, 2011).

O objetivo da Mineração de Processos é a utilização dos dados dos registros de eventos para extrair processos relacionados às informações, por exemplo, descobrir automaticamente um modelo de processo por observação dos eventos registrados por alguns sistemas de empresas (VAN DER AALST, 2011).

Destacam-se dois fatores principais para o crescente interesse em Mineração de Processos: i) cada vez mais eventos são registrados, facilitando assim a criação de históricos de processos e, ii) existe a necessidade de melhorar processos de negócio em ambientes competitivos e de rápida evolução (VAN DER AALST *et al.*, 2011).

A ideia base da Mineração de Processos é descobrir, monitorar e melhorar processos reais por meio da extração de conhecimento a partir de registros de eventos usualmente disponíveis nos sistemas de informação existentes. O ponto inicial para a Mineração de Processos é um registro de eventos (VAN DER AALST *et al.*, 2011), que são dados brutos escondidos em todo tipo de fonte de dados (VAN DER AALST, 2011).

De acordo com Van Der Aalst (2011) uma fonte de dados pode ser um arquivo simples, uma planilha eletrônica, um registro de transação ou uma tabela de banco de dados. Contudo, não se pode esperar que todos os dados sejam de uma única fonte de dados bem estruturada. A realidade é que dados de eventos são tipicamente espalhados por diferentes fontes de dados e frequentemente alguns esforços são necessários para coletar os dados relevantes (VAN DER AALST *et al.*, 2011).

De acordo com Van Der Aalst *et al.* (2011) os três tipos de Mineração de Processo são descritos por:

- Descoberta: uma técnica deste tipo usa exclusivamente um conjunto de registros de eventos para produzir um modelo, isto é, não requer qualquer tipo de informação à *priori*.
- Conformidade: técnicas deste tipo comparam modelos de processo existentes com um conjunto de registros de eventos do mesmo processo. Verificação de conformidade pode ser usada para verificar se o processo tal como descrito nos registros de eventos está em conformidade com os modelos e vice-versa.
- Extensão: técnicas deste tipo complementam ou aperfeiçoam um modelo de processo existente usando informação acerca de um processo descrito em um conjunto de registros de eventos.

Ao utilizar a Mineração de Processos, muitos tipos de informação podem ser coletados sobre o processo, tais como o fluxo de controle, o desempenho, a

informação organizacional e padrões de decisão, e todas essas perspectivas podem ser utilizadas em um modelo que é capaz de representar o processo nos aspectos que forem necessários (LEMOS *et al.*, 2011).

Na área de processos de software, abordagens de mineração de processos são importantes no sentido de permitir a mineração e descoberta de conhecimento a partir de repositórios de ambientes que armazenem informações sobre processos (RUBIN *et al.*, 2007), a qual é chamada de Mineração de Processo de Software. Serão descritos nas subseções seguintes em detalhes, os conceitos relacionados à Mineração de Processos de Software, a Ferramenta para Mineração de Processos - ProM e Registros de Eventos.

2.2.1 Mineração de Processos de Software

Mineração de Processos de Software, uma linha de pesquisa de processos de software, tem como objetivo tornar explícito em termos de um ou mais modelos os processos de software, a partir dos dados armazenados nos Sistema de Gerência de Configuração, a fim de ajudar engenheiros a identificar, melhorar o entendimento, analisar, otimizar e executar seus processos de software (RUBIN *et al.*, 2007).

De acordo com Rubin *et al.* (2007), durante o processo de desenvolvimento de software são criados, alterados, atualizados e revisados todo tipo de documentos e arquivos e para lidar com essa grande quantidade de dados, documentos e arquivos, engenheiros usam diferentes tipos de Sistema de Gerenciamento de Documentos: Sistema de Gerência de Dados de Produtos (PDM); Repositórios; e Sistemas de Gerência de Configuração (SCM).

A capacidade do uso de repositórios de software para deduzir informações sobre o projeto de software tem sido pesquisada no domínio de Mineração de Repositório de Software (RUBIN *et al.*, 2007). Um repositório pode então ser definido como uma base de dados compartilhada de informações sobre artefatos de engenharia produzidos e consumidos por uma empresa (BERNSTEIN; DAYAL, 1994).

De acordo com Kagdi *et al.* (2007) os repositórios mantêm uma riqueza de informações e proporcionam uma visão única do caminho real evolutivo levado para realizar um sistema de software e incluem fontes como as informações armazenadas nos sistemas de controle de versão, por exemplo, o Sistema de Controle de Versões (CVS), Sistemas de Controle de *Bugs*, por exemplo, o *Bugzilla,* e arquivos de comunicação, por exemplo, *e-mail*.

Muitas vezes, esses dados existem para toda a duração de um projeto e podem representar milhares de versões com anos de detalhes sobre o

desenvolvimento e incluem coisas como versões individuais do sistema, as mudanças e os metadados sobre as alterações (por exemplo, quem fez a alteração, por que a mudança foi feita, quando a mudança foi feita, etc.) (KAGDI *et al.*, 2007).

2.2.2 Ferramenta para Mineração de Processos: ProM

O *framework* ProM foi desenvolvido com o objetivo de apoiar a Mineração de Processos, concentrando as mais recentes e avançadas técnicas e algoritmos da área (VAN DER AALST *et al.*, 2011). A ProM é a mais comum e popular ferramenta de mineração de processos (VAN DER AALST, 2011). É uma ferramenta de código aberto e extensível, ou seja, pode ser melhorada por meio da criação de novos *plugins* (SAYLAM; SAHINGOZ, 2014).

O *framework* ProM oferece uma variedade de algoritmos e suporta a mineração de processo em um sentido mais amplo, uma vez que pode ser usado para descobrir processos, identificar gargalos, analisar as redes sociais, verificar regras de negócio, e além disso, a ProM fornece interfaces para extrair informações de diferentes fontes, incluindo Sistemas de Gerência de Configuração, tais com CVS e Subversion (RUBIN *et al.*, 2007).

O componente central do processo de mineração é seu algoritmo (SAYLAM; SAHINGOZ, 2014). Estes autores destacam alguns algoritmos que serão descritos a seguir:

- *Alpha Algorithm*: Ele fornece uma boa perspectiva para o mundo da mineração de processo, embora tenha alguns problemas de ruídos e frequência, e seu resultado não é muito realista;
- *Heuristic Mining*: O algoritmo o *Heuristic Mining* (mineração heurística) (VAN DER AALST, 2011) tem o objetivo de usar a frequência dos eventos e a sequência para gerar o modelo de processo. Dessa forma os caminhos que não são frequentemente percorridos não são incorporados no modelo (VAN DER AALST, 2011). Ele se concentra na perspectiva de fluxo de controle e cria um modelo de processo em formato rede heurística para um determinado registro de eventos;
- *Social Network Analysis*: Sociometria é um método de apresentação em forma gráfica e matriz que se refere a dados relativos às relações interpessoais, os dados referentes às relações interpessoais também estão escondidos, e estão nos registros de eventos; entre outros.

2.2.3 Registros de Eventos (*Event Log*)

O princípio base da mineração de processos é descobrir, monitorar e melhorar processos reais por meio da extração de conhecimento a partir de registros de eventos usualmente disponíveis nos sistemas de informação existentes (VAN DER AALST *et al.*, 2011). Por meio destes registros de eventos é possível identificar a forma como os processos foram estruturados, como estes estão relacionados e de que maneira as pessoas estão envolvidas, permitindo, assim, a criação do conhecimento organizacional e, por conseguinte, um maior controle gerencial (GRECO *et al.*, 2008).

Em Van der Aalst *et al.* (2011) é destacado que os eventos devem estar relacionados com os elementos do modelo de processo, porém em alguns casos, pode não ser trivial estabelecer essa relação, por exemplo, um evento pode referir-se a duas atividades diferentes ou não ser claro a qual atividade se refere. Tais ambiguidades necessitam ser removidas, a fim de interpretar os resultados da Mineração de Processo. Além do problema de relacionar eventos a atividades, há ainda o problema de relacionar eventos com instâncias do processo, isto é comumente referido como correlação de eventos (VAN DER AALST *et al.*, 2011).

O formato específico lido pelo framework ProM (VAN DONGEN *et al.*, 2005), chama-se XES (*Extensible Event Stream*). XES é um padrão baseado em XML para registros de eventos e seu objetivo é fornecer um formato de reconhecimento geral para o intercâmbio de dados de registros de eventos entre ferramentas e domínios de aplicação (VERBEEK *et al.*, 2011). A próxima seção apresenta as métricas de complexidade de modelos de processos.

2.3 Métricas de Complexidade de Modelo de Processos

Métricas de processo é qualquer tipo de medida relacionada a um processo, e permitem quantificar os atributos dos processos (CARDOSO, 2008). De acordo com Vanderfeesten *et al.* (2007), a complexidade mede a simplicidade e capacidade de compreensão de um projeto.

Em Cardoso (2008) é definido como complexidade o grau em que um processo é difícil de analisar, entender ou explicar. Ele destaca que complexidade pode ser caracterizada pelo número e complexidade das interfaces de atividades, transições, ramificações condicionais e paralelas, a existência de *loops*, funções, categorias de atividades, os tipos de estruturas de dados e outras características do processo. É assumido por Lassen e van der Aalst (2009) que a complexidade do modelo afeta diretamente a legibilidade e a qualidade do modelo. Um modelo simples e extremamente compreensível geralmente é resultante de uma baixa complexidade,

enquanto, se a complexidade de um modelo é alta, a compreensibilidade deste modelo é baixa.

De acordo com Cardoso (2008) não existe uma métrica única que possa ser usada para medir a complexidade de um processo. Ele destaca as quatro perspectivas principais de complexidade podem ser identificadas: complexidade da atividade, complexidade do fluxo de controle, complexidade do fluxo de dados e complexidade dos recursos:

- *Complexidade da atividade.* Essa visão sobre complexidade simplesmente calcula o número de atividades que um processo possui.
- *Complexidade do fluxo de controle*. O comportamento do fluxo de controle de um processo é afetado por construções como divisões, junções, loops e pontos finais e iniciais.
- *Complexidade do fluxo de dados*. A complexidade do fluxo de dados de um processo aumenta com a complexidade de suas estruturas de dados, o número de parâmetros formais das atividades e os mapeamentos entre os dados das atividades.
- *Complexidade de recursos*. Atividades em um processo precisam acessar recursos durante suas execuções.

Cardoso *et al.* (2006) discute métricas de complexidade para software e sua aplicabilidade para modelos de processos de negócios. Eles propõem a adaptação de algumas das métricas de código fonte, mais conhecidas e amplamente utilizadas, à análise de processos de negócios: número de linhas de código (*LOC – Line Of Code*) (KALB, 1990) e complexidade ciclomática *McCabe* (MCBABE; BUTER, 1989).

De acordo com Cardoso *et al.* (2006) se uma atividade de processo é visualizada como uma declaração de um programa de software, pode-se derivar uma métrica muito simples (métrica M1) que apenas conta o número de atividades (NOA) em um processo de negócios: *M1: NOA = Número de atividades em um processo*. Outra adaptação da métrica LOC é exibir não apenas as atividades como instruções do programa, mas também levar em conta os elementos do fluxo de controle do processo (ou seja, estruturas de controle). Elementos de controle afetam a sequência de execução das atividades. Dois tipos de métricas podem ser projetados dependendo da estrutura do processo (CARDOSO *et al.*, 2006). Quando os processos são bem estruturados, pode-se simplesmente contar as estruturas de controle correspondentes às divisões, pois é sabido explicitamente que uma junção correspondente de saída. Tendo essas características em mente, Cardoso *et al.* (2006) projetaram uma segunda métrica (M2), que conta as atividades e controla os elementos de um processo: *M2: NOAC = Número de atividades e elementos de controle em um processo*. Quando os

processos não são bem estruturados, as divisões não precisam corresponder a uma junção correspondente. Esses processos geralmente são mais difíceis de entender e resultam frequentemente em erros de projeto. Para processos que não são bem estruturados, Cardoso *et al.* (2006) projetaram uma terceira métrica (M3) que conta o número de atividades e o número de divisões e junções de um processo: *M3: NOAJS = Número de atividades, junções e divisões em um processo*.

Cardoso (2008) propôs a métrica *Control-Flow Complexity* (CFC) para medir a complexidade do fluxo de controle dos modelos de processos de negócios (CARDOSO *et al.*, 2006; CARDOSO, 2008). O CFC é uma adaptação da métrica da complexidade ciclomática de McCabe (McCABE, 1976). A complexidade ciclomática (MCC) de McCabe é uma indicação da complexidade do fluxo de controle de um módulo de programa e foi considerado um indicador confiável de complexidade em grandes projetos de software (WARD, 1989). Enquanto a complexidade ciclomática de McCabe atribui a mesma semântica a todos os nós de decisão, o CFC distingue os vários nós como tendo semânticas diferentes, como divisões AND, divisões XOR, junções OR etc. O CFC é calculado simplesmente adicionando o CFC de todas as construções divididas, conforme Figura 3.

$$CFC_{abs}(P) = \left(\sum_{i \in (\text{XOR-splits of P})} CFC_{XOR-split}\ i\right) + \left(\sum_{j \in (\text{OR-splits of P})} CFC_{OR-split}\ j\right) + \left(\sum_{k \in (\text{AND-splits of P})} CFC_{AND-split}\ k\right) +$$

Figura 3 - Fórmula da *Control-Flow Complexity* (CFC)
Fonte: Cardoso (2008)

De acordo com Cardoso (2008) a complexidade relativa do fluxo de controle para o processo P é calculada da seguinte forma, onde |P| é o número de atividades do processo P: *Quanto maior o valor dos CFC_{abs} (P) e CFC_{rel} (P), maior a complexidade arquitetônica geral de um processo*.

A análise do CFC procura avaliar a complexidade sem a execução direta de processos. A função do CFC é calculada com base na complexidade individual do fluxo de controle das divisões XOR, OR e AND. Salvo indicação em contrário, CFC (P) indica a complexidade absoluta do fluxo de controle (CARDOSO, 2008). A métrica conta as várias *splits* (XOR, OR e AND) na rede e concede a cada uma delas uma certa penalidade (LASSEN; VAN DER AALST, 2009).

A medição de processos de negócios é a tarefa de atribuir números de forma empírica e objetiva às propriedades dos processos de negócios de maneira a descrevê-los (LASSEN; VAN DER AALST, 2009). Propriedades estas relacionadas a estudar e medir incluem complexidade, custo, manutenção e confiabilidade. Lassen e van der Aalst (2009) se concentraram em investigar a complexidade do processo.

Lassen e Van der Aalst (2009) propuseram três métricas de complexidade para uma subclasse de redes de Petri, denominada redes de fluxo de trabalho (*Workflow Nets*). Essas métricas incluem métrica Cardoso estendida (ECaM), métrica ciclomática estendida (ECyM) e métrica de estruturação (SM). O ECaM estende o CFC na medida em que é feito sob medida para oferecer suporte às redes de Petri, enquanto o ECyM estende a métrica ciclomática medindo a acessibilidade do gráfico em vez da estrutura do gráfico, e a métrica SM mede bem a estrutura do design (SZIMANSKI, 2013). As três métricas propostas foram implementadas na ferramenta de análise de processos ProM:

- *Extended cyclomatic metric* (LASSEN; VAN DER AALST, 2009). É uma adaptação da métrica *McCabe's Cyclomatic Number* (McCABE 1976). McCabe não considera o código real, apenas seu comportamento e, no mesmo sentido, podemos estender a métrica ciclomática às redes de *workflow* , não medindo a estrutura real da redes de *workflow* , mas sim o gráfico de acessibilidade. Apenas as redes de *workflow* são redes de máquina de estado e se comportam com um grafo de controle de fluxo, i.e. que não permite a concorrência/simultaneidade. Neste sentido esta métrica foi estendida para redes de *workflow*. A métrica ECyM claramente tenta medir um aspecto que considera quais estados o processo pode estar e quais transições podem ocorrer.
- *Extended cardoso metric* (LASSEN; VAN DER AALST, 2009). É uma generalização e melhoria da Cardoso Metric (CARDOSO, 2006) com o objetivo de trabalhar com *Non-free Choice Petri nets*, onde decisões mais complexas são possíveis. A métrica proposta por Cardoso (2008) é aplicável a linguagens de modelagem com *splits* XOR, OR e AND, é puramente baseada na presença de certas *splits* e *joins* na definição de processo. As redes de Petri têm apenas dois tipos de nós: locais (*place*) e transição (*transition*). No entanto, usando locais e transição, é possível modelar divisões XOR, OR e AND. De fato, usando redes de escolha não livres (*non-free choice nets*), são possíveis opções ainda mais complexas. Portanto, Lassen e van der Aalst (2009) definiram uma versão da métrica para Rede de Petri que generaliza e aprimora a métrica original. A métrica ECaM penaliza cada estado pelo número de

estados sucessores diretos que ela induz. Assim a definição diz que a penalidade para um local *p* é o número de subconjuntos de locais alcançáveis a partir de um local *p*.

- *Structuredness metric* (LASSEN; VAN DER AALST, 2009). Esta métrica tenta capturar melhor a complexidade do modelo conforme é percebido pelos seres humanos. Ele analisa iterativamente a estrutura do modelo e atribui penalidades a construções indesejáveis do ponto de vista da complexidade. Esta métrica foi desenvolvida a partir dos problemas encontrados nas métricas Cardoso Metric (CARDOSO, 2006) e *McCabe's Cyclomatic Number* (McCABE, 1976). A *Cardoso Metric* tem somente o foco na sintaxe do modelo e ignora a complexidade do comportamento. No entanto, isto não resolve o problema uma vez que a complexidade do comportamento resulta da interação entre os componentes de modelagem diferentes. *McCabe's Cyclomatic Number* tem o foco somente no comportamento resultante e ignora a complexidade do modelo em si. Variantes de ambas as métricas têm dois problemas em comum: (i) focam em um único aspecto (comportamento x sintaxe); (ii) não consideram a interação entre os elementos. Esta métrica foi desenvolvida para captar uma visão que reconhece diferentes tipos de estruturas e resultados de cada estrutura, dando-lhe alguns valores de penalidade. A soma destes valores é usado para definir a *Structuredness Metric*.

De acordo com Lassen e Van Der Aalst (2009), as três métricas de complexidade são definidas de maneira diferente e são bons representantes dos vários tipos de métricas de complexidade encontradas na literatura.

Em Szimanski (2013) é selecionado um conjunto de métricas para avaliar modelos hierárquicos. Este modelo hierárquico de Markov pode ser extraído a partir de um *log* de eventos (i.e. uma micro-sequência ou um conjunto de micro-sequências) e uma descrição de alto nível do processo de negócio (i.e. o macro-modelo). O seguinte conjunto de métricas foi selecionado por Szimanski (2013):

- **Número de arcos por nó:** avalia a densidade de um modelo, i.e., através da razão entre o número total de arcos em um modelo de processo para o número total dos seus nós, podemos ter noção da dimensão (complexidade) de um modelo.
- **Densidade relacional:** é um tipo de métrica que além de proporcionar uma visão da dimensão também possibilita a avaliação da intensidade de um modelo de processo.
- **Número de caminhos:** é um tipo de métrica que pode ser importante para medir a complexidade, pois em um modelo que apresenta uma sequência

linear tem a complexidade mínima (independentemente de a sequência ser curta ou longa). Porém, um modelo, que tenha vários caminhos possíveis, tem maior complexidade, pois um modelo em que seja possível ir de qualquer atividade para qualquer outra atividade tem complexidade máxima (independentemente do número de atividades).

- **Tamanho do caminho:** é um tipo de métrica importante para verificar o tamanho dos caminhos possíveis em um modelo.
- **Complexidade ciclomática:** é uma métrica para avaliar a complexidade do comportamento de modelos considerando os loops (ciclicidade) dos caminhos possíveis.
- ***Fan-in/Fan-out*:** é baseada no fluxo de dados entre os diferentes módulos de um programa. A informação pode ser passada entre os módulos de três maneiras: por um módulo ao invocar outro; retornando um resultado do módulo invocado para o módulo que chamou; e por meio da troca de informação através de uma estrutura de dados global.
- **Número de subprocessos**: é uma métrica de modularidade para auxiliar a compreensão de um modelo de processo. Um alto valor para esta métrica indica menor complexidade do modelo.

A próxima seção apresenta conceitos relacionados à Análise de Conformidade de Processos.

2.4 Análise de Conformidade de Processos

Conformidade de processo é o grau de concordância entre o processo de desenvolvimento de software que é realmente realizado e o processo que é definido para ser realizado (SØRUMGÅRD, 1997).

Derniame e Kaba (1999) destacam duas facetas no ambiente de produção de software moderno:

- processo de execução, ou seja, o processo de produção do mundo real, incluindo agentes humanos, ferramentas CASE, entre outro, que abrangem todas as ações destinadas a desenvolver o produto de software, e
- modelo de processo, que é uma representação do mundo real, e captura o estado atual das atividades para guiar ou automatizar partes do processo de produção.

Derniame e Kaba (1999) ainda destacam que o modelo de processo e o processo de execução devem ser perfeitamente alinhados, no sentido de que o estado interno do modelo de processo seja uma representação fiel do real estado de coisas

no mundo real, ou seja, o processo de execução é uma instância do modelo de processo.

Processos de Software são compostos por atividades criativas (modelagem, verificação, comunicação, tomadas de decisões, entre outras), as quais requerem a interação humana para sua realização (BENDRAOU *et al.,* 2007; LAURENT *et al.*, 2014). Neste sentido, durante a execução, os elementos de processo definidos podem ser alterados para, por exemplo, atender a situações não previstas, relacionadas a fatores como prazo e gerenciamento de recursos (LAURENT *et al.*, 2014).

É comum ver gerentes de projeto e desenvolvedores realizarem alterações devido a preferências pessoais, experiências anteriores, pressões de custo e prazo, e também por treinamento inadequado, falta de comprometimento e falta de comunicação entre os membros da equipe (LEMOS *et al.*, 2011).

Um grau de inconsistência entre o processo de execução e o modelo de processo é, portanto, inerente à natureza das coisas e põe o risco de consequências desastrosas para a qualidade do produto final, o que faz com que o modelo de processo coordene as atividades e gerencie o fluxo de informações, e se adapte a qualquer evolução do processo de mundo real (DERNIAME; KABA, 1999).

Apesar da importância reconhecida do Processo de Software para a qualidade do produto (OSTERWEIL, 1987; CUGOLA; GHEZZI, 1998; FUGGETTA, 2000) e dos esforços para sua definição (DERNIAME; KABA, 1999; ROCHA; MALDONADO; WEBER, 2001), desvios ocorrem durante a execução do processo de desenvolvimento de software (SMATTI; NACER, 2014).

Desvios são conhecidos por serem ações realizadas por agentes de processo (gerentes de projeto, analista e desenvolvedores), que não estão descritas ou não são permitidas no modelo do processo. Encontrar soluções para lidar com a ocorrência de desvios tem se tornado mais do que importante, a fim de guiar o desenvolvimento de software (SMATTI; NACER, 2014).

Os desvios e inconsistências são principalmente resultados de modelos de processo naturalmente incompletos e da interação humana (RUI *et al.*, 2014). Como resultado da ocorrência dos desvios, a qualidade dos produtos de software, tempo de entrega e os custos são afetados (SMATTI; NACER, 2014).

Madhavji (1991) *apud* Derniame e Kaba (1999) identifica várias razões que pressionam alterações nos processos de software:

- o processo pode estar com problemas;
- no processo pode estar faltando alguns passos importantes, tornando-o ineficaz;

- os modelos do processo podem ser genéricos e precisam ser personalizados para se obter resultados específico;
- as premissas segundo as quais o modelo foi projetado processo não são mais válidas; e
- a dinâmica da política, dos seres humanos e da tecnologia pode ter um efeito sobre a execução dos processos de software.

Mohammed, Redouane e Bernard (2007) afirmam que um desvio é uma ação executada que não está descrita no processo predefinido ou que viola alguma das restrições expressas no processo. A inconsistência é o estado do processo de software a partir do desvio do processo (MOHAMMED; REDOUANE; BERNARD, 2007).

De acordo com Lemos *et al.* (2011), é importante encontrar automaticamente uma forma de analisar os registros de execução para descobrir o modelo de processo real, para caracterizar tendências e nelas destacar anomalias. A realização de uma Análise de Conformidade de processo de software entre modelos que reflitam informações de planejamento e execução pode evidenciar se existem diferenças entre os modelos.

Sørumgård (1997) destaca que existem várias razões do porquê conformidade de processo é um tópico importante, dependendo do propósito e área de aplicação. Na área do desenvolvimento de software, Sørumgård (1997) afirma que conformidade é um fator da qualidade do processo e destaca três razões primárias para sua importância: garantir uma execução estável do processo, assim alcançando um processo previsível; assegurar a validade dos dados, informações, experiências e conhecimentos, os quais são adquiridos ao longo do desenvolvimento; e garantir que a execução do processo está em conformidade com requisitos específicos, por exemplo, em o caso de credenciamento ou certificação.

De acordo com Sørumgård (1997), na literatura são identificadas três principais maneiras de verificar conformidade em processos de software:

- A realização de entrevistas é uma forma.
- A segunda abordagem é baseada na execução do modelo formal do processo suportada por um sistema computacional. Nesse caso a conformidade pode ser garantida, mais do que medida. Isto porque o processo é executado sob o comando do sistema computacional.
- Uma terceira forma de monitorar processos utiliza algoritmos aplicados a fluxos de eventos do modelo formal e sua execução para comparar as diferenças entre os dois.

No próximo Capítulo serão apresentados os estudos exploratórios conduzidos para fundamentação do problema tratado nesta tese.

3 Estudos Exploratórios

Neste capítulo são apresentados os estudos exploratórios conduzidos para fundamentação do problema tratado nesta tese. A pesquisa de opinião realizada com profissionais e o estudo do conjunto de ferramentas identificadas são descritos. Também será descrito o estudo realizado em duas bases dados de projeto de desenvolvimento de software, relacionando com resultados da revisão ad hoc da literatura. Por último são apresentadas as considerações finais.

3.1 Introdução

Em um estudo preliminar para identificar as variações que ocorrem entre o processo definido e executado (SANTOS; OLIVEIRA; BRITO E ABREU, 2015) realizado em dados de um projeto de desenvolvimento foi possível observar a ocorrência da dificuldade de identificação do processo de software do projeto ao longo dos projetos de desenvolvimento de software. Ao aplicar esta técnica nos registros de execução gerados durante a instanciação e execução do processo de desenvolvimento de software um grande esforço foi necessário para representar a atividade do processo referente a cada tarefa, ou seja, não estava explícita esta informação. Esta dificuldade motivou a pesquisar o problema tratado nesta tese onde os **processos de software não são explicitamente identificados em planos de projeto**, buscando evidências por meio da realização de estudos: pesquisa de opinião, estudo de ferramentas, estudos bases de dados de projeto e revisão *ad hoc* da literatura.

A pesquisa de opinião teve como objetivo investigar se as características de processo, identificadas na literatura, estão presentes ao longo dos projetos de desenvolvimento de software. Também teve como objetivo investigar quais ferramentas são utilizadas, ao longo dos projetos de desenvolvimento de software, para identificar as estruturas que armazenam os dados de execução de processo, ou seja, quais as ferramentas utilizadas ao longo dos projetos. Então foi realizado o levantamento dos dados que são armazenados por cada ferramenta identificada na pesquisa de opinião (Apêndice B). Como forma de reforçar a fundamentação do problema tratado nesta tese, também foi realizado o estudo de duas bases dados de execução de processo, associado a resultados da revisão *ad hoc* da literatura.

Nas seções seguintes serão apresentados os estudos exploratórios conduzidos para apoiar a fundamentação do problema tratado nesta tese.

3.2 Pesquisa de Opinião

Considerando a dificuldade de identificar conceitos de processo de software em projetos de desenvolvimento de software, foi realizada uma pesquisa ao longo do ano de 2015 para entender melhor como os desenvolvedores observam um processo de software e seus conceitos relacionados ao longo dos projetos de desenvolvimento de software. O objetivo foi investigar se as principais características de processo de software identificados na literatura são observadas nos projetos de desenvolvimento de software. Além disso, foram investigadas quais ferramentas são usadas para observar essas características.

Antes de iniciar o planejamento e execução da pesquisa de opinião foi necessário identificar um conjunto de características de processo de software, que seriam investigadas em relação a sua observação ao longo dos projetos de desenvolvimento de software. Nesta pesquisa de tese os elementos de processo, e as restrições relacionadas a eles e ao modelo de processo, serão chamados de Características de Processo de Software. Para a identificação destas características, as definições de processo apresentadas no Capítulo 2 na Subseção 2.1, foram analisadas. O seguinte conjunto de Características de Processo de Software foi identificado:

- O Processo tem em sua composição um conjunto de Atividades (ISO/IEC, 2005; BERTOLLO, 2009; WESKE, 2012; DUMAS *et al.*, 2013).
- O Processo tem em sua composição Eventos (DUMAS *et al.*, 2013).
- O Processo tem em sua composição Pontos de Decisão (DUMAS *et al.*, 2013).
- O Processo é uma coleção inter-relacionada de Atividades, Eventos e Pontos de Decisão (DUMAS *et al.*, 2013).
- O Processo pode ser composto por subprocessos, que estão ligados de alguma forma (FALBO; BERTOLLO, 2009; PFLEEGER; ATLEE, 2010; MÜNCH *et al.*, 2012).
- Um Processo é um conjunto de etapas parcialmente ordenadas (FEILER; HUMPHREY, 1993).
- O Processo estabelece todas as principais atividades do processo (PFLEEGER; ATLEE, 2010).
- O Processo utiliza um Modelo de Ciclo de Vida (Ex. Cascata e Espiral) como referência (FALBO; BERTOLLO, 2009).

- O Processo transforma entradas em saídas (ISO/IEC, 2005; MÜNCH *et al.*, 2012).
- As Regras governam/orientam o Processo (ERIKSSON; PENKER, 2000).
- Algumas Atividades dependem de pré-Atividades (FALBO; BERTOLLO, 2009).
- Uma Atividade pode ser decomposta em sub-Atividades (FALBO; BERTOLLO, 2009).
- As Atividades estão sujeitas a um conjunto de Restrições (PFLEEGER; ATLEE, 2010).
- Cada Atividade do processo tem um critério de entrada e saída, que possibilita saber quando uma Atividade começa e termina (PFLEEGER; ATLEE, 2010).
- As atividades são organizadas em sequência, estando evidente quando uma atividade é realizada em relação a outras atividades (PFLEEGER; ATLEE, 2010).
- As Atividades estão inter-relacionadas com outras Atividades (ISO/IEC, 2005).
- Os objetivos coordenam a execução de uma Atividade (WESKE, 2012).
- Os métodos definem um conjunto de etapas a serem executadas em uma Atividade (FALBO; BERTOLLO, 2009).
- As orientações descrevem como usar uma ferramenta na realização de uma Atividade (FALBO; BERTOLLO, 2009).
- Modelos (*templates*) são adotados para preparar um Artefato em uma Atividade (FALBO; BERTOLLO, 2009).
- As Atividades produzem Artefatos (FALBO; BERTOLLO, 2009; PFLEEGER; ATLEE, 2010).
- As Atividades utilizam Artefatos (OSTERWEIL, 1987; FALBO; BERTOLLO, 2009).
- Artefatos estão sujeitos a um conjunto de Restrições (PFLEEGER; ATLEE, 2010).
- Uma Atividade requer Recursos de Software (FALBO; BERTOLLO, 2009; MÜNCH *et al.*, 2012).
- Uma Atividade requer Recursos de Hardware (FALBO; BERTOLLO, 2009; MÜNCH *et al.*, 2012).
- Uma Atividade requer Recursos de Humanos (FALBO; BERTOLLO, 2009; MÜNCH *et al.*, 2012).
- Os Recursos estão sujeitos a um conjunto de Restrições (PFLEEGER; ATLEE, 2010).

Uma vez identificado o conjunto de características foi dado prosseguimento ao planejamento e execução. No Apêndice A é apresentado o planejamento da pesquisa de opinião, e a execução e discussão dos resultados serão apresentados nas subseções seguintes.

3.2.1 Execução da Pesquisa de Opinião

O público-alvo foi composto por gerentes de projeto, analistas de sistema e desenvolvedores com experiência em projetos de desenvolvimento de software. A pesquisa envolveu quatro países, Brasil, Canadá, Alemanha e Portugal, e resultou em 30 respostas, sendo obtidas 20 respostas na primeira rodada e 10 na segunda rodada. Além das questões de caracterização do participante (identificação e experiência) e sua atual organização de trabalho, foi perguntado quais características eram observadas ao longo dos projetos de desenvolvimento de software em suas organizações. Para as características observadas ao longo dos projetos era solicitado o nome das ferramentas que permitiam aos participantes tal observação.

A cada rodada de execução foi realizada a validação do instrumento com a participação de integrantes do grupo de pesquisa PRISMA. As respostas destas validações não foram consideradas no resultado final. Nosso objetivo foi obter um retorno dos participantes em relação a compreensibilidade do questionário e medir o tempo de preenchimento. A partir de cada validação, foi possível implementar melhorias no instrumento da pesquisa de opinião.

A primeira rodada ocorreu em fevereiro de 2015 e os participantes foram selecionados por conveniência, sendo a população composta por profissionais da rede de contato dos pesquisadores envolvidos, que tinham o perfil de acordo com o definido no planejamento. Após a análise dos resultados identificamos oportunidades de melhoria, a serem implementadas nas próximas rodadas. Em abril de 2015, houve a oportunidade de aplicar o estudo de forma supervisionada, por meio de entrevista, em empresas selecionadas, por amostra de conveniência, em Portugal e no Brasil. Os gráficos de frequência gerados a partir do resultado de cada rodada são apresentados no Apêndice A.

Em cada rodada de execução o contexto e a seleção dos participantes ocorreram de forma diferente, por esta razão, apesar de em geral fazerem parte do planejamento, elas serão descritas junto com a execução de cada rodada a seguir, além dos resultados e discussões.

3.2.1.1 Execução da Primeira Rodada do Estudo

Nesta rodada o estudo foi executado pelo participante, em uma ferramenta de gerenciamento de formulários, para preencher o questionário. Cada participante teve tempo ilimitado para preencher o questionário, preenchendo as informações relacionadas com sua caracterização e de sua organização. Em seguida ele indicou quais características de processo de desenvolvimento de software são observadas durante o projeto de desenvolvimento de software; e por último informou quais ferramentas são utilizadas para observar as características de processo de desenvolvimento durante o projeto de desenvolvimento de software.

A população foi formada por profissionais, da rede de contado dos pesquisadores envolvidos nesta pesquisa de opinião que atuam em projetos de desenvolvimento de software. Os participantes foram convidados a participar por e-mail. O convite de participação continha em anexo o formulário de caracterização que deveria ser preenchido e retornado ao remetente.

Foram enviados 27 convites de participação, e um total de 20 participantes responderam o questionário. Os respondentes foram brasileiros, atuando em Organizações de desenvolvimento de software no Brasil, Canadá e Alemanha.

3.2.1.1.1 Resultados e Discussões

Após o término do período de respostas, foi possível iniciar a etapa de análise dos resultados.

Em relação à questão de pesquisa – *As características de processo de desenvolvimento de software identificadas na literatura de processo são observadas durante o projeto de desenvolvimento de software?*

Ao analisar os gráficos de frequência gerados de cada característica, nos deparamos com a dificuldade de retirar conclusões. A forma como a questão sobre as características foi formulada, utilizando a escala de *Likert* com as opções Concordo Totalmente, *Concordo Parcialmente*, *Discordo Parcialmente* e *Discordo Totalmente*, dificultou a análise.

Em relação à questão – *Quais ferramentas são utilizadas para observar as características de processo de desenvolvimento de software identificadas na literatura de processo*?

Apesar dos problemas encontrados em relação a observância das características, foi possível obter uma relação de ferramentas utilizadas ao longo dos projetos de desenvolvimento de software. A lista de ferramentas apontadas pelos respondentes é apresentada na Tabela 1. Nesta tabela foram omitidas as seguintes

ferramentas que foram citadas pelos participantes apenas uma vez: *Wiki, Netbeans, Kanban Eletrônico, Ferramentas de teste para Python, Ferramentas de teste para Ruby, Sonar, Mylyn, Enterprise Architect, Rational Rose, JudeUML/Astah, MS Project, MS Office, Confluence, ChangePoint, OTTR, Perforce SCM, RTC, Rally, Trello, Pivotal, GLPI, Google Docs, Youtrack, Visual Studio e Sublime*.

Tabela 1 - Ferramentas citadas na primeira rodada do estudo

Ferramenta	Quantidade de vezes que foi citada
GIT	14
Fórum de Discussão	11
E-mail	10
Stack Overflow	10
Github	8
Redmine	5
SVN	5
Maven	5
Jira	4
Gitlab	4
Sourceforge	3
Jenkins	3
Bugzilla	3
Mercurial	3
Google Code	3
Mantis Bug Tracker	2
CVS	2

Nesta rodada não foi possível obter a relação das ferramentas por característica de processo observada, devido à forma como foi formalizada a questão no instrumento. Desta forma o resultado contém ferramentas que não necessariamente suportam a observação das características.

Apesar deste problema descrito acima e da dificuldade de análise da observância das características, foi realizado o estudo de cada ferramenta, para verificar se por meio delas é possível observar as características de processo. Este estudo será apresentado em detalhes na Seção 3.3. Porém cabe destacar que a partir do estudo das ferramentas foi possível concluir que embora a maior parte dos respondentes tenha respondido *Concordo Totalmente* e *Concordo Parcialmente* para a maior parte das características, em algumas delas as ferramentas citadas não apresentam os dados necessários para a observação explícita.

3.2.1.2 Execução da Segunda Rodada do Estudo

Nesta rodada aproveitou-se a oportunidade de uma visita ao ISCTE – Instituto Universitário de Lisboa, em Portugal, para realizar a execução da pesquisa de opinião de forma supervisionada. Esta mesma forma de execução foi executada no Brasil.

De acordo com as oportunidades de melhorias identificadas na primeira rodada, algumas alterações foram feitas no instrumento, tais como: revisão do conjunto de características e reformulação da questão sobre as características, definindo como possibilidades de respostas *Sim*, *Não* e *Às vezes*. Também foi reformulada a questão sobre as ferramentas para que fosse possível informar as ferramentas utilizadas por cada característica de processo observada.

Nesta rodada a forma de aplicação foi por meio de entrevistas, que foram realizadas no Brasil e em Portugal. Cada participante teve tempo ilimitado para preencher o questionário. A aplicação do questionário era iniciada com uma breve apresentação dos objetivos do estudo, e em seguida o questionário era apresentado. Ao longo do preenchimento o participante tinha a possibilidade de consultar o pesquisador aplicador sobre alguma dúvida.

Cada participante preencheu as informações relacionadas à sua caracterização e de sua organização, em seguida ele indicou quais características de processo de desenvolvimento de software são observadas durante o projeto de desenvolvimento de software; e por último informou quais ferramentas são utilizadas para observar as características de processo de desenvolvimento durante o projeto de desenvolvimento de software. A população foi por selecionada por conveniência, sendo composta por respondentes de rede de contato dos pesquisadores envolvidos. Os participantes foram convidados a participar por contato telefônico ou de forma presencial. No total 10 profissionais participaram desta rodada.

3.2.1.2.1 Resultados e Discussões

Após o término do período de respostas, foi possível iniciar a etapa de análise dos resultados.

Em relação à questão de pesquisa – *As características de processo de desenvolvimento de software identificadas na literatura de processo são observadas durante o projeto de desenvolvimento de software?*

Ao analisar os gráficos de frequência gerados de cada característica de processo apresentada ao respondente foi possível observar que:

- não houve unanimidade de observação das características de processo apresentadas pelos respondentes.

- a maioria dos respondentes responderam “Não” ou “Nem Sempre” para as características a seguir:
    - O Processo tem em sua composição Eventos.
    - O Processo tem em sua composição Pontos de Decisão.
    - Cada Atividade do processo tem um critério de entrada e saída, que possibilita saber quando uma Atividade começa e termina.
    - Os objetivos coordenam a execução de uma Atividade.
    - Os métodos definem um conjunto de etapas a serem executadas em uma Atividade.
    - Artefatos estão sujeitos a um conjunto de Restrições.
    - Uma Atividade requer Recursos de Software.
    - Os Recursos estão sujeitos a um conjunto de Restrições.

Em relação a Questão Q2 – *Quais ferramentas são utilizadas para observar as características de processo de desenvolvimento de software identificadas na literatura de processo?*

A lista de ferramentas, respondidas pelos respondentes, por características é apresentada na Tabela 2. De forma geral estas foram as ferramentas respondidas: *AS400, Confluence, Documentação, e-mail, Enterprise Architect, GIT, Github, Google Code, IBM Lotus Notes, Jira, MS Project, Project Libre, Redmine, SVN* e *Trac*.

Tabela 2 - Características por Ferramentas citadas na segunda rodada do estudo

Características	Ferramentas Utilizadas
O Processo tem em sua composição um conjunto de Atividades.	Redmine, Github, MS Project, IBM Lotus Notes, Google Code e Jira
O Processo tem em sua composição Eventos.	Jira
O Processo tem em sua composição Pontos de Decisão.	E-mail, IBM Lotus Notes e Jira
O Processo é uma coleção inter-relacionada de Atividades, Eventos e Pontos de Decisão.	IBM Lotus Notes
O Processo pode ser composto por subprocessos, que estão ligados de alguma forma.	Redmine, IBM Lotus Notes, MS Project, Jira e Enterprise Architect
Um Processo é um conjunto de etapas parcialmente ordenadas.	Redmine, IBM Lotus Notes, Jira e Enterprise Architect
O Processo utiliza um Modelo de Ciclo de	IBM Lotus Notes, MS Project e Jira

Vida (Ex. Cascata e Espiral) como referência.	
As Regras governam/orientam o Processo.	Documentação e Confluence
O Processo de software interage com outros Processos.	Redmine, Github, IBM Lotus Notes, As400 e Jira
Algumas Atividades dependem de pré-Atividades.	Project Libre, Redmine, MS Project, IBM Lotus Notes, MS Project e Jira
Uma Atividade pode ser decomposta em sub-Atividades.	Trac, Redmine, Githud, IBM Lotus Notes, MS Project, Jira e Enterprise Architect
As Atividades estão sujeitas a um conjunto de Restrições.	Documentação, As400, IBM Lotus Notes e Jira
Cada Atividade do processo tem um critério de entrada e saída, que possibilita saber quando uma Atividade começa e termina.	Trac, Project Libre, Redmine, Confluence, MS Project e Enterprise Architect
As atividades são organizadas em sequência, estando evidente quando uma atividade é realizada em relação a outras atividades.	Project Libre, Trac, Redmine, IBM Lotus Notes, Jira, Confluence e MS Project
Os objetivos coordenam a execução de uma Atividade.	Trac, Redmine, IBM Lotus Notes, Confluence e Jira
Os métodos definem um conjunto de etapas a serem executadas em uma Atividade.	SVN, AS 400, IBM Lotus Notes e Confluence
As orientações descrevem como usar uma ferramenta na realização de uma Atividade.	SVN, AS 400, IBM Lotus Notes e Confluence
Modelos (templates) são adotados para preparar um Artefato em uma Atividade.	Documentação, Redmine, AS400, IBM Lotus Notes, Jira e Confluence
As Atividades produzem Artefatos.	SVN, Gitlab, documentação, IBM Lotus Notes, e-mail, Enterprise Architect e Confluence
As Atividades utilizam Artefatos.	SVN, AS 400, IBM Lotus Notes, e-mail, Enterprise Architect, Confluence e GIT
Artefatos estão sujeitos a um conjunto de Restrições.	Documentação, Enterprise Architect, Confluence
Uma Atividade requer Recursos de Software.	Trac e MS Project
Uma Atividade requer Recursos de Hardware.	AS400 e MS Project

Uma Atividade requer Recursos de Humanos.	Trac, Redmine, MS Project, AS400, MS Project
Os Recursos estão sujeitos a um conjunto de Restrições.	AS400 e IBM Lotus Notes

O formato do questionário permitiu observar um comportamento interessante, que embora o respondente respondesse que observava a característica (respondendo *Sim* ou *Nem Sempre*), não necessariamente era respondido a ferramenta correspondente utilizada par tal observação. Este comportamento nos remete à situação que foi observada na primeira rodada, sobre as características de processo, as quais não são observadas explicitamente por meio de ferramentas.

De forma geral, a partir dos resultados obtidos da primeira e segunda rodada, é possível observar em geral as características são observadas ao longo dos projetos de desenvolvimento, embora em algumas situações, o respondente não consiga informar a forma de observação. Este comportamento direciona a interpretação que nem todas as características de processo são explícitas por meio das ferramentas utilizadas durante os projetos de desenvolvimento de software, e que o conhecimento sobre o que acontece durante os projetos de desenvolvimento de software é tácito, ou seja, pessoal e fortemente relacionado a quem está observando. Considerando esta conclusão foi realizado o estudo de cada ferramenta citada como meio de observação das características.

Na seção seguinte será discutido com mais detalhes o estudo das ferramentas.

3.3 Estudo das Ferramentas

As ferramentas são um apoio importante para a execução do processo de software durante os projetos de desenvolvimento de software. Em geral, os membros da equipe de um projeto de desenvolvimento de software passam grande parte do tempo usando ambientes de desenvolvimento integrado (*Integrated Development Environment* - IDEs), navegadores e ferramentas de gerenciamento de projetos. As ferramentas variam de IDEs a editores de código-fonte, automação de compilação e depuradores.

Na pesquisa de opinião realizada foi possível identificar um conjunto de ferramentas que são utilizadas nos projetos de desenvolvimento de software. A partir deste resultado foi realizado o estudo de cada ferramenta citada como meio de observação das características. O objetivo do estudo foi identificar quais dados são armazenados pelas ferramentas na criação e execução das tarefas nos projetos de software.

Em relação às ferramentas foram identificadas as seguintes: *Bugzilla, CVS, GIT, GitHub, GitLab, Google Code, Jazz RTC, Jenkins, JIRA, Mantis, Bug Tracker, Maven, Mercurial, Redmine, Sourceforge, Stack Overflow, SVN, e TFSO*. Para obter os dados que são armazenados por elas, foi necessário ter acesso às ferramentas, quando possível, bem como realizar a análise da documentação de cada uma delas. As seguintes informações de cada ferramenta foram levantadas: *Nome, Descrição da Ferramenta, Estrutura, Tipo de Licenciamento e se possui Recurso de Exportação*. O Apêndice B apresenta os dados que foram levantados. Os seguintes tipos de ferramentas foram identificados: controle de versão, gerenciamento de configuração, repositórios de códigos, controle de *bugs*, controle de tarefas, e recursos tais como, wikis, listas de discussão, documentação, fórum e dados textuais.

Ao analisar cada ferramenta utilizada foi possível observar que as mesmas não suportam o registro de elementos de processo necessários para sua identificação, como a atividade do processo e o fluxo de controle das atividades. Diante do exposto, a partir de cada ferramenta mencionada na pesquisa, que foi analisada, foi possível identificar que elas também não possuem uma conexão explícita entre processos e projetos. Essa conclusão é corroborada por Bastarrica *et al.* (2017), que enfatiza que, embora haja uma infinidade de ferramentas para especificação de processos e planejamento e monitoramento de projetos, não existem ferramentas que integrem ambas as perspectivas. Portanto, foi possível concluir que os desenvolvedores não percebem as características de processo ao longo da execução dos projetos, pois as ferramentas utilizadas não fornecem suporte para esta observação.

Neste sentido, os estudos realizados sobre as ferramentas, demonstram indícios de que as ferramentas utilizadas ao longo dos projetos de desenvolvimento de software, geralmente são agnósticas ao processo de software, ou seja, as mesmas não suportam o registro de elementos de processo necessários para sua identificação.

Ferramentas de gerenciamento de projetos como *Redmine*, *Jira* e *MSProject*, entre outras, são geralmente usadas para planejar um projeto de software (por exemplo, para atribuir tarefas a recursos e agendar prazos), além de registrar informações relevantes sobre sua execução (por exemplo, data de execução de cada tarefa e sua duração). Todas as ações executadas em projetos, suportadas por essas ferramentas, podem ser registradas em arquivos de *log*. Assim, essas ferramentas parecem ser uma boa fonte de dados para encontrar uma conexão entre processos de software e projetos de software.

No entanto, muitos sistemas de gerenciamento de projetos são agnósticos ao processo, assim em seus registros, por exemplo, não há informações registradas sobre a qual processo ou instância de processo uma determinada tarefa está se

referindo (BURATTIN, 2015). Estes fatores impactam diretamente na identificação do processo de software que efetivamente é seguido ao longo dos projetos. Em Santos, Oliveira e Brito e Abreu (2015), foram utilizados os registros de execução de projetos gerados pela ferramenta de gerenciamento de projeto, chamada *Redmine*, para descobrir o processo de software executado por meio de técnicas de mineração de processos. Na literatura foram identificados os *Process-centered Software Engineering Environments*, ou PSEEs. Porém a falta de uma linguagem de modelagem de processo padrão (CHOU; LI, 2014) e falta de suporte ao gerenciamento à evolução dinâmica da execução do processo (MOHAMMED; REDOUANE; BERNARD, 2007) impediram sua ampla aplicação e adoção na indústria de desenvolvimento de software. Importante destacar que na pesquisa de opinião, do total de 30 respostas obtidas durante as rodadas executadas, nenhum participante apontou o uso de PSEEs ao longo dos projetos de desenvolvimento de software. Outro tipo de sistema relacionado inclui os pacotes *Application Lifecycle Management* (ALM). No entanto, eles geralmente não têm uma conexão explícita entre processo e projeto, ou seja, são agnósticos ao processo.

3.4 Estudo das bases de dados de projetos de desenvolvimento de software

A partir da pesquisa de opinião de opinião foi possível concluir que nem todas as características de processo são explícitas por meio das ferramentas utilizadas durante os projetos de desenvolvimento de software, e que o conhecimento sobre o que acontece durante os projetos de desenvolvimento de software é tácito, ou seja, pessoal e fortemente relacionado a quem está observando. E a partir da análise do conjunto de ferramentas foi possível concluir que as mesmas são agnósticas ao processo de software, ou seja, as mesmas não suportam o registro de elementos de processo necessários para sua identificação.

Como forma de reforçar estas conclusões, foi realizado o estudo de duas bases de dados projetos de desenvolvimento de software. A primeira base foi disponibilizada por uma empresa de desenvolvimento de software, nível A no MPS.BR. Esta empresa possui uma ferramenta própria de gerenciamento de tarefas. Os seguintes dados foram disponibilizados: *cd_projeto, cd_tarefa, cd_area, ds_tarefa, dt_cadastro, dt_desejada_concl, tmp_previsto, tmp_gasto, dh_termino, dh_inicio, cd_modulo, dt_desejada_planej, id_entregavel, dh_cadastro_osdh_inicio_exec, dh_inicio_desenv,* e *dh_entregacliente*. A segunda base foi disponibilizada pela gerente do projeto de desenvolvimento de software, chamado SIGAEPCT (FABRO NETO *et al.*, 2012). O

projeto de desenvolvimento deste sistema teve início em 2008, e foi desenvolvido colaborativamente por pesquisadores das Instituições Federais de Ensino envolvidos no projeto, por meio de núcleos de pesquisa geograficamente distribuídos pelo país, utilizando tecnologias de código aberto. A instanciação deste projeto ocorria a partir da criação das tarefas na ferramenta *Redmine*. Os seguintes dados foram extraídos: *Número da tarefa, Situação, Produto, Tipo, Prioridade, Título da Tarefa, Atribuído para, Categoria, Versão, Autor, Início, Data prevista, % Terminado, Tempo estimado, Tarefa pai, Criado em, Alterado em, Núcleo, Coordenação SIGA, SIGAEPCT Versão Afetada, Google Doc,* e *Descrição*. Foram analisadas as duas bases de dados de projetos de desenvolvimento de software, e em nenhuma delas havia o registro de informações que estabelecesse uma ligação com o processo de software utilizado como referência para sua execução.

Desta forma, a partir da análise de bases de dados de projetos de desenvolvimento de software, foi possível observar foi que a instanciação não carrega explicitamente conceitos de processo, ou seja, os processos de software não são explicitamente encontrados em planos de projeto. Para ilustrar melhor o problema abordado, a Figura 4 apresenta dois modelos conceituais que descrevem os elementos do processo de software e do projeto de software. Porém, como pode ser observado pela análise dos modelos, não existe uma ligação explícita entre as perspectivas. Esses modelos estão organizados nas duas perspectivas: a perspectiva de processo e a perspectiva de projeto.

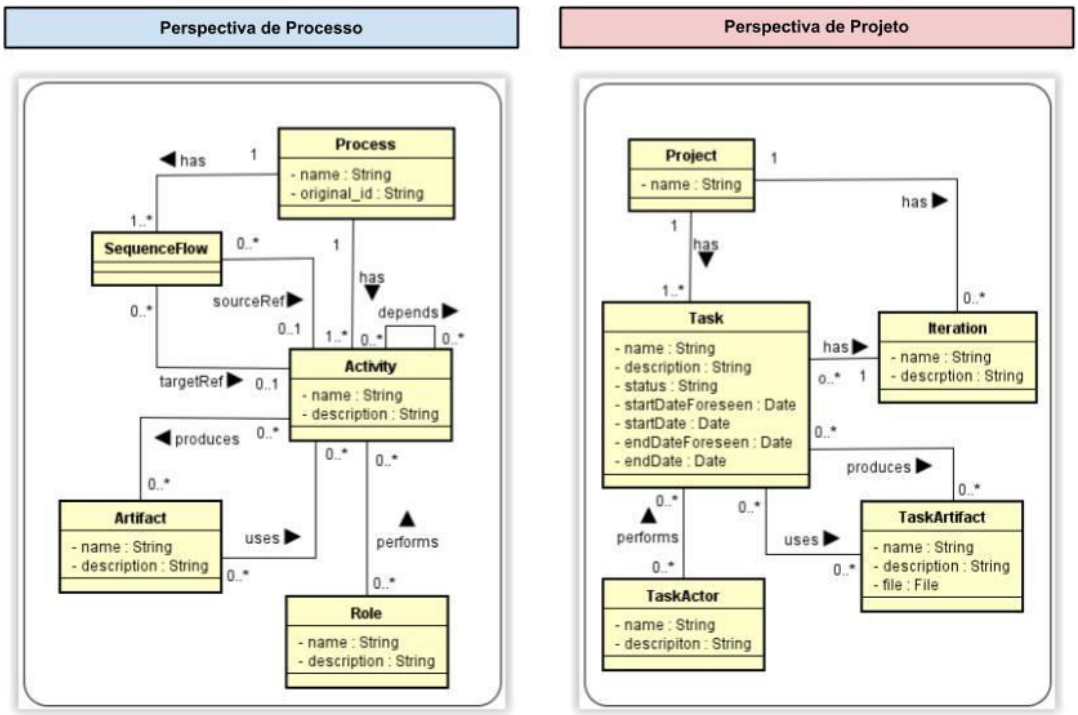


Figura 4 - Metamodelo das Perspectivas de Processo e Projeto. Fonte: A Autora.

A Figura 4 (lado esquerdo) apresenta um trecho de um modelo conceitual de processo, de acordo com os metamodelos SPEM (S*oftware Process Engineering Metamodel*) (OMG, 2008) e BPMN (*Business Process Modeling and Notation*) (OMG, 2014), que são compostos por elementos de processo de software como *Processo, Activity (Atividade), SequenceFlow (Fluxo de Sequência), Artfact (Artefato) e Role (Papel)*. Nos projetos de software os modelos de processo são instanciados, e podem ser representados por meio de planos de projeto. A Figura 4 (lado direito) apresenta o modelo conceitual do projeto de acordo com os conceitos do PMBOK (*Project Management Body of Knowledge*) (PMI, 2017) e no estudo de ferramentas utilizadas ao longo de projetos de desenvolvimento de software, incluindo os seguintes elementos: *Project (Projeto), Task (Tarefa), TaskArtifact (Artefato associado a tarefa), TaskActor (ator associado a tarefa), TaskRole (papel associado a tarefa) e Iteration (Iteração)*.

Embora a Ontologia de Processo de Software (SPO) vise estabelecer uma conceituação comum no domínio processo de software, incluindo processos, atividades, recursos, pessoas, artefatos e procedimentos (RUY *et al.*, 2016) e considere a relação entre processo de software e o projeto de software (RUY *et al.*, 2016), nem sempre tornar esse mapeamento explícito é trivial.

3.5 Revisão *ad hoc* da literatura

Uma revisão *ad hoc* da literatura foi realizada para reforçar as evidências identificadas nos estudos apresentados neste capítulo. Neste sentido, a partir de uma revisão *ad hoc* da literatura realizada foi possível identificar trabalhos, tais como Frappier e Richard (2004), Killisperger *et al.* (2011), Cuadrado-Garcia *et al.* (2011), Friedrich e Bergner (2011) e Hummel e Heinrich (2013), os quais destacam que manter o plano do projeto baseado no modelo de processo não é trivial. Sendo assim, pode não ser fácil mapear conceitos encontrados nos contextos do processo e do projeto. Por exemplo, uma atividade do modelo de processo pode ser considerada muito complexa para ser realizada por um único desenvolvedor ao considerar a experiência da equipe, exigindo, portanto, várias tarefas no plano do projeto. Além disso, é importante destacar que as tarefas criadas no plano do projeto podem ser dispersas por diferentes iterações, de acordo com a metodologia de desenvolvimento adotada. Outro fator observado em projetos de software é a falta de informações sobre a instância do processo à qual uma determinada tarefa pertence. Uma sequência de execução do processo também é chamada de instância do processo (BAIER *et al.*,

2015). Um plano de projeto pode conter tarefas de várias instâncias do mesmo processo de software, por exemplo, conforme relatado por Santos, Oliveira e Brito e Abreu (2015). De acordo com Van der Aalst *et al.* (2011), além do problema de relacionar eventos (registros de execução) às atividades, há o problema de relacionar eventos às instâncias do processo. Sem registrar algumas informações que representam uma conexão entre o processo de software e o projeto de software, a identificação do processo a partir das tarefas não é uma atividade trivial, é demorada, e propensa a erros ao tentar inferir a atividade do processo de cada tarefa após sua execução. De acordo com Baier *et al.* (2018), a suposição de confiar no conhecimento prévio do mapeamento de registros de execução para atividades muitas vezes não é refletida na realidade.

Geralmente, o mapeamento não existe porque (i) o mecanismo de registros dos sistemas captura etapas de baixa granularidade em nível técnico e (ii) como os eventos são registrados, eles raramente são personalizáveis, especialmente nos sistemas legados (BAIER *et al.*, 2018). Desta forma, identificar uma correspondência entre as atividades do processo, nos planos e nos registros de execução não é uma tarefa simples (BASTARRICA *et al.*, 2017).

Neste sentido, foi possível concluir que os processos de software não são explicitamente encontrados em planos de projeto. Desta forma, a falta de representação de informações da perspectiva de processo na perspectiva de projeto, ou seja, a falta de ligação explícita entre elas, pode impactar na identificação do processo de software a partir das tarefas executadas no contexto do projeto.

A seção seguinte apresenta uma discussão sobre a ocorrência de desvios no contexto das execuções do processo, que justificam a dedicação de esforços para a realização do mapeamento das perspectivas, e assim possibilitar as análises relacionadas à identificação do processo de software executado e a verificação de sua conformidade com o processo de software definido como referência ao longo do projeto de software.

3.6 Discussão

Os fatores expostos neste capítulo revelam que, apesar dos esforços dedicados à criação de diretrizes e identificação de práticas recomendadas para o gerenciamento de processos de software, ainda existe uma lacuna entre as perspectivas do processo e do projeto. Embora os processos sejam definidos, por meio de modelos ou descrições textuais, as equipes adotam o modelo de processo mais como um guia de como a execução deve ocorrer, desviando-se deles à vontade

(SILVA *et al.*, 2013; SMATTI; OUSSALAH; NACER, 2016). Na revisão da literatura identificamos trabalhos que destacam a ocorrência de desvios ao longo da condução dos projetos de desenvolvimento de software, tais como: Melli (1988), Cugola *et al.* (1995), Cugola *et al.* (1998), Mohammed *et al.*, (2007), Silva *et al.*, (2010), Yong e Zhou (2010), Silva *et al.* (2013), Smatti e Nacer (2014) e Rui *et al.* (2014).

Desvios são conhecidos por serem ações realizadas por agentes de processo (gerentes de projeto, analistas de sistemas e desenvolvedores), que não estão descritas ou não são permitidas no modelo do processo definido para o projeto (SMATTI; NACER, 2014). De acordo com Silva *et al.* (2013) desvios são comuns em projetos de desenvolvimento de software ou porque os desenvolvedores falham em entender a maneira esperada de executar um processo ou porque decidem executar de outra maneira, assumindo que resultados equivalentes serão obtidos. Além disso, os processos de software são fortemente baseados em atividades criativas (BENDRAOU *et al.*, 2007; LAURENT *et al.*, 2014). Assim, os modelos de processos de software definidos para projetos podem ser alterados em tempo de execução, por exemplo, não executando atividades previstas, adicionando atividades não previstas e alterando a sequência de execução prevista. No entanto, estas alterações não são rastreadas durante os desvios, impossibilitando a avaliação de seus efeitos no sucesso ou fracasso do projeto (SILVA *et al.*, 2010).

Uma vez que os desvios geram inconsistências entre a definição e a execução do processo, se faz necessário mapear as perspectivas, de forma a permitir a identificação do processo que efetivamente é seguido pela equipe do projeto, a assim viabilizar a identificação de não conformidades. Neste sentido, a perspectiva do projeto se apresenta como uma rica fonte de dados do processo, mas a falta de um mapeamento explícito entre as perspectivas dificulta a extração de tais dados.

3.7 Considerações Finais

Este capítulo apresentou os elementos que evidenciaram a existência do problema tratado nesta tese. Os fatores apresentados impactam diretamente na realização de análises da execução dos processos.

Embora as organizações de desenvolvimento de software busquem continuamente melhorar seus processos de desenvolvimento e manutenção de software, uma vez que estas estão diretamente relacionadas à qualidade dos produtos de software resultantes, as investigações conduzidas (pesquisa de opinião com profissionais, a análise de projetos de desenvolvimento de software relacionados à indústria e a revisão da literatura) nos levaram ao entendimento que: i) os

desenvolvedores têm dificuldade de perceber os elementos de processo ao longo da execução dos projetos; ii) as ferramentas utilizadas ao longo dos projetos não suportam o registro de elementos de processo necessários para sua identificação, como a atividade do processo e o fluxo de controle das atividades; e iii) os processos de software não são explicitamente encontrados em planos de projeto. Em nossa análise, estes fatores dificultam o mapeamento entre as perspectivas.

O Capítulo 4 apresenta o mapeamento sistemático conduzido para identificar na literatura trabalhos que visam tratam o problema abordado nesta tese.

4 Revisão da Literatura

Este capítulo apresenta a revisão da literatura, que objetivou a aquisição de conhecimento relacionado ao estado da arte sobre propostas que tratam o problema abordado nesta tese.

4.1 Introdução

O problema tratado nesta tese é que os processos de software não são explicitamente identificados em planos de projeto. Para fundamentar este problema estudos exploratórios foram conduzidos. A partir da realização dos estudos descritos no Capítulo 3 foi possível concluir que o problema tem relação direta com a falta de mapeamento explícito entre as perspectivas de processo e projeto de softwar.

Uma busca estruturada foi realizada com o objetivo de adquirir conhecimento relacionado ao estado da arte de propostas que tratam o problema tratado nesta tese, e assim, identificar os trabalhos relacionados. Este Capítulo apresenta a revisão da literatura sobre a falta de mapeamento entre as perspectivas de processo e projeto. As Seções 4.2 e 4.3 descrevem o planejamento e execução da revisão da literatura, respectivamente. A Seção 4.4, lista os trabalhos relacionados identificados. A Seção 4.5 apresenta as ameaças à validade e para finalizar na Seção 4.6 são apresentadas as considerações finais.

4.2 Planejamento

Seguindo a abordagem GQM (*Goal, Question, Metric*) (BASILI; CALDIERA; ROMBACH, 1994) o objetivo global desta busca estrurada foi:

Analisar ferramentas, métodos, processos e técnicas de mapeamento entre o processo de software definido para o projeto e o processo de software executado;

Com o propósito de caracterizar;

Com respeito à capacidade de realizar a representação explícita do mapeamento;

Do ponto de vista de engenheiros de software;

No contexto de projetos de desenvolvimento de software utilizados como referência para apresentar as ferramentas, métodos, processos e técnicas.

4.2.1 Questão de Pesquisa

Como questão principal tem-se: *Que apoio é oferecido por abordagens existentes (ferramentas, métodos, processos e técnicas) para realizar o mapeamento entre o processo de software definido para o projeto e o plano, bem como entre o processo de software definido e o processo de software executado?*

As seguintes são as questões secundárias:

- **QS1**: Como o mapeamento é realizado (manual / automatizado / semi-automatizado)?
- **QS2:** Quais níveis estão envolvidos no mapeamento (processo – plano/ processo – execução)?
- **QS3:** Quais tipos de problemas são enfrentados pela falta do mapeamento (distanciamento entre os níveis)?
- **QS4:** Quais elementos de processo de software (atividades, fluxo, entre outros) estão envolvidos no mapeamento?
- **QS5:** Qual a importância da realização do mapeamento (argumentação dos autores de cada abordagem em relação a importância do mapeamento)?
- **QS5:** Algum tipo de tratamento é realizado nos artefatos manipulados pelos níveis de definição (processo) e execução (projeto) para a realização do mapeamento (vínculo das informações de definição de processo com os registros de planejamento e execução de projeto)?
- **QS6:** O mapeamento é realizado para suportar identificar o processo de software executado?
- **QS7:** O mapeamento é realizado para suportar a identificação de não conformidade?

4.2.2 Estratégia de Busca e Artigos de Controle

Nesta busca estruturada a abordagem PICO – *Population of interest, evaluated Intervention, Intervention Comparison e expected Outcome* (PAI *et al.*, 2004), para organizar e estruturar a busca a ser realizada:

(P) População (*Population*): Processo de Software aplicados em projeto de desenvolvimento de software

(I) Intervenção (*Intevention*): Técnicas, abordagens, métodos, metodologias, ferramentas de apoio ao mapeamento entre o processo de software definido para o projeto e o plano e também entre o processo de software definido e o processo de software executado,

(C) Comparação (*Comparison*): Não há, pois não se tem como objetivo comparar abordagens, e sim, caracterizá-las.

(O) Resultados (*Output*): Trabalhos sobre o mapeamento entre o processo de software definido para o projeto e o plano e entre o processo de software definido e o processo de software executado.

Baseado na estrutura PICO, os seguintes termos e seus sinônimos foram identificados:

(P) População: *software process, process of software, software development process, software process definition, process model, workflow, lifecycle, life cycle, reference model, standard process, defined process, process description, description of process, modelled process, modelled software process, business process, software development project, software project, project construction, project plan, observed model, observed process, observed software process, executed process, executed software process, real software process, actual software process, real process, actual process, software process deployment, software process execution, process execution, software process instantiation, project execution.*

(I) Intervenção: *technique, technical, approach, method, methodology, tool, procedure, mechanism, research, study.*

(C) Comparação: não aplicável.

(O) Resultados: *mapping, matching, linking, alignment, relation, relationship, adherence, tracing.*

Observação: Como forma de ter uma ampla visão da área de processo, na string abaixo foi incluída na população a termo “business process”.

A base de busca selecionada foi a Scopus, sendo executada a seguinte string de busca:

(TITLE-ABS-KEY (("software process" OR "process of software" OR "software development process" OR "software process definition" OR "process definition’’ OR "process model" OR "workflow" OR "lifecycle" OR "life cycle" OR "reference model" or "reference process" OR "standard process" OR "defined process" OR "process description" OR "description of process" OR "modelled process" OR "modelled software process" OR "business process")) AND TITLE-ABS-KEY (("software development project" OR "software project" OR "project construction" OR "project plan" OR "observed model" OR "observed process" OR "observed software process" OR "executed process" OR "executed software process" OR "real software process" OR "actual software process" OR "real process" OR "actual process" OR "software process deployment" OR "process deployment" OR "software process execution" OR "process execution" OR "project

execution" OR "software process instantiation" OR "process instantiation")) AND TITLE-ABS-KEY (("technique" OR "technical" OR "approach" OR "method" OR "methodology" OR "tool" OR "procedure" OR "mechanism" OR "research" OR "study")) AND TITLE-ABS-KEY (("map*" OR "match*" OR "link*" OR "align*" OR "tracing" OR "relation*" OR "adherence")))

Os seguintes artigos de controle foram identificados:

- Leoni, M., & Marrella, A. (2017). **Aligning Real Process Executions and Prescriptive Process Models through Automated Planning**. Expert Systems with Applications, 82, 162-183.
- Baier, T., Rogge-Solti, A., Mendling, J., & Weske, M. (2015, April). **Matching of events and activities**: an approach based on behavioral constraint satisfaction. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing (pp. 1225-1230). ACM.

4.2.3 Critérios de Inclusão e Exclusão dos Artigos

A seguir são especificados os Critérios de Inclusão (CI) e Critérios Exclusão (CE) que foram adotados para apoiar a seleção dos artigos.

Critérios de Inclusão dos Artigos:

(CI1) Trabalhos que abordam sobre mapeamento de processos de software, apresentando como esta deve ser realizada e/ou apresentam ferramental de apoio para descoberta de não conformidades; OU

(CI2) Trabalhos que abordam níveis de processo que estão envolvidos no mapeamento (processo – plano/ processo – execução); OU

(CI3) Trabalhos que citam os elementos de processo de software estão envolvidos no mapeamento; OU

(CI4) Trabalhos que abordando os tipos de problemas enfrentados pela falta do mapeamento (distanciamento entre os níveis) na identificação de não conformidades; OU

(CI5) Trabalhos que abordam a importância da realização do mapeamento; OU

(CI6) Trabalhos que abordam tipos de tratamentos realizados nos artefatos manipulados pelos níveis para a realização do mapeamento (vínculo das informações de definição de processo com os registros de planejamento e execução de projeto).

Critérios de Exclusão de Artigos:

(CE1) Trabalhos fora da área de computação; OU

(CE2) Trabalhos cujo foco de pesquisa encontra-se em outras áreas da computação que não seja a de Engenharia de Software e Processos de Negócio; OU

(CE3) Abordagens que não sejam aplicadas, especificamente, a processos de software e de negócio; OU

(CE4) Trabalhos que não sejam destinados ao mapeamento (vínculo das informações de definição de processo com os registros de planejamento e execução de projeto); OU

(CE5) Trabalhos indisponíveis; OU

(CE6) Trabalhos não escritos em inglês; OU

(CE7) Artigos publicados por meios que não exigem revisão por pares; OU

(CE8) Artigos que não apresentem estudos de avaliação da abordagem em questão; OU

(CE9) Prefácios e apresentações de *Proceedings* de conferências.

4.2.4 Procedimentos para seleção dos estudos

A seleção dos estudos será realizada em quatro etapas:

- *Etapa 1*

Seleção preliminar das publicações: A seleção preliminar das publicações será realizada por meio da execução da *string* de busca na base pesquisa *Scopus*, selecionada para este estudo.

- *Etapa 2*

Seleção das publicações relevantes - 1º filtro: Para uma seleção inicial das publicações relevantes, os resumos de cada artigo retornado devem ser lidos e avaliados de acordo com os critérios de inclusão e exclusão definidos no planejamento da busca. Em alguns artigos, apenas a leitura dos resumos, pode não ser suficiente, gerando dúvidas se deveriam ou não ser incluídos. Neste caso os artigos devem selecionados para leitura completa.

- *Etapa 3*

Seleção das publicações relevantes - 2º filtro: A seleção final das publicações relevantes dar-se-á por meio da leitura completa dos textos das publicações selecionadas no 1º filtro. Estas publicações também serão avaliadas de acordo com os critérios de inclusão e exclusão.

- *Etapa 4*

Seleção das publicações a partir de *snowballing* – As publicações relevantes resultantes, selecionadas no 2º filtro, serão usadas como "*start set*" da abordagem de snowballing. Snowballing refere-se ao uso da lista de referência de um artigo ou das

citações do artigo para identificar artigos adicionais (WOHLIN, 2014). Neste sentido as referências de cada artigo devem ser analisadas, bem como as citações registradas no Google Acadêmico. Estas publicações também serão avaliadas de acordo com os critérios de inclusão e exclusão.

4.2.5 Campos de Extração e Avaliação de Qualidade

Os campos de extração (C) de dados dos trabalhos selecionados foram definidos visando capturar dados que auxiliem na resposta à questão de pesquisa estabelecida para esta busca estruturada. Estes são listados a seguir:

(C1) Descrição de como é realizada o mapeamento é realizado (manual / automatizado / semi-automatizado);

(C2) Procedimentos (métodos, processos e técnicas) e ferramentas utilizadas;

(C3) Níveis envolvidos no mapeados (processo – plano/ processo – execução);

(C4) Elementos de processo de software estão envolvidos no mapeamento;

(C5) Tipos de problemas são enfrentados pela falta do mapeamento (distanciamento entre os níveis);

(C6) Importância da realização do mapeamento;

(C7) Descrição de como é realizado, caso seja realizado, o tratamento realizado nos artefatos manipulados pelos níveis para a realização do mapeamento (vínculo entre informações da definição do processo e os registros de planejamento e execução de projeto; e

(C8) Suporte a identificação de não conformidade (Sim/Não - Como).

Neste trabalho, será considerado que as fontes dos documentos são confiáveis, e que os trabalhos retornados tenham passado por revisões externas que serviram de filtragem para que tenham qualidade suficiente para contribuir com este mapeamento sistemático.

Em caso de informações conflitantes entre os trabalhos encontrados, será progressivamente atribuída maior relevância aos trabalhos que:

- apresentem soluções para o problema da ausência do mapeamento entre o processo de software definido para o projeto e o plano e entre o processo de software definido e o processo de software executado;
- apresentem uma proposta teórica para mapeamento entre o processo de software;
- contenham resultados de experiências reais das organizações que adotaram o mapeamento de processos de desenvolvimento de software; e
- apresentem evidências da validação dos resultados (tipo de estudo aplicado).

Por questão de tempo e disponibilidade de pesquisadores para participar deste estudo, a avaliação da inclusão ou não dos artigos retornados pela busca estruturada será realizada apenas pela autora desta proposta de tese.

4.3 Execução

De acordo com os procedimentos de seleção dos artigos definidos no planejamento desta busca estruturada, foi realizada a execução da *string* de busca na base pesquisa Scopus, selecionada para este estudo. Nesta etapa foram retornados 885 estudos distintos.

Na etapa seguinte, de seleção das publicações relevantes, os resumos de cada um dos 885 artigos retornados na etapa anterior foram e avaliados de acordo com os critérios de inclusão e exclusão definidos no planejamento da busca. Em alguns artigos, apenas a leitura dos resumos, não foi suficiente, gerando dúvidas se deveriam ou não ser incluídos. No total 81 artigos foram selecionados, e neste caso leitura completa foi realizada. Na etapa seguinte, considerado os critérios de inclusão e exclusão na lista resultante da etapa anterior, 12 artigos foram selecionados. A lista de artigos resultantes após aplicação dos 1º e 2º filtros é apresentada na Tabela 3.

Tabela 3 – Lista de Artigos Selecionados

Ano	**Referência**
2012	LEONI, M.; MAGGI, F. M.; VAN DER AALST, W. M. “Aligning event logs and declarative process models for conformance checking”. In: **International Conference on Business Process Management**. Springer, Berlin, Heidelberg, 2012. p. 82-97.
2012	LEONI, Massimiliano; VAN DER AALST, W. M.; VAN DONGEN, Boudewijn F. “Data-and resource-aware conformance checking of business processes”. In: **International Conference on Business Information Systems**. Springer, Berlin, Heidelberg, 2012. p. 48-59.
2012	ADRIANSYAH, A.; MUNOZ-GAMA, J.; CARMONA, J.; VAN DONGEN, B. F.; VAN DER AALST, W. M. “Alignment based precision checking”. In: **International Conference on Business Process Management**. Springer, Berlin, Heidelberg, 2012. p. 137-149.
2012	ADRIANSYAH, A.; BUIJS, JCA M. “Mining process performance from event logs: the BPI challenge 2012”. In: **Case Study. BPM Center Report BPM-12-15, BPMcenter. org**. 2012.
2013	LEONI, M.; VAN DER AALST, W. M. “Aligning event logs and process models for multi-perspective conformance checking: An approach based on integer linear programming”. In: **Business Process Management**. Springer, Berlin, Heidelberg, 2013. p. 113-129.
2013	FERREIRA, D. R.; SZIMANSKI, F.; RALHA, C. G. “Mining the low-level behaviour of agents in high-level business processes”. **International**

	Journal of Business Process Integration and Management, v. 6, n. 2, p. 146-166, 2013.
2014	BAIER, T.; ROGGE-SOLTI, A.; WESKE, M.; MENDLING, J. “Matching of events and activities-an approach based on constraint satisfaction”. In: **IFIP Working Conference on The Practice of Enterprise Modeling**. Springer, Berlin, Heidelberg, 2014. p. 58-72.
2015	BAIER, T.; ROGGE-SOLTI, A.; MENDLING, J.; WESKE, M. “Matching of events and activities: an approach based on behavioral constraint satisfaction”. In: **Proceedings of the 30th Annual ACM Symposium on Applied Computing**. 2015. p. 1225-1230.
2015	BAIER, T.; DI CICCIO, C.; MENDLING, J.; WESKE, M. “Matching of events and activities-an approach using declarative modeling constraints”. In: **Enterprise, Business-Process and Information Systems Modeling**. Springer, Cham, 2015. p. 119-134.
2017	VAN DONGEN, B.; CARMONA, J.; CHATAIN, T.; TAYMOURI, F. “Aligning modeled and observed behavior: a compromise between computation complexity and quality”. In: **International Conference on Advanced Information Systems Engineering**. Springer, Cham, 2017. p. 94-109.
2017	LEONI, M.; MARRELLA, A. “Aligning real process executions and prescriptive process models through automated planning”. **Expert Systems with Applications**, v. 82, p. 162-183, 2017.
2018	BAIER, T.; DI CICCIO, C.; MENDLING, J.; WESKE, M. “Matching events and activities by integrating behavioral aspects and label analysis”. **Software & Systems Modeling**, v. 17, n. 2, p. 573-598, 2018.

Na última etapa, de seleção das publicações a partir de *snowballing*, as publicações selecionadas no 2º filtro, foram utilizadas como *start set* para aplicação da abordagem. Neste sentido, as referências de cada artigo foram analisadas, bem como as citações registradas no Google Acadêmico. Considerando os critérios de inclusão e exclusão na lista resultante da etapa anterior, 6 artigos foram selecionados. A lista de artigos identificados nesta etapa é apresentada na Tabela 4.

Tabela 4 – Lista de Artigos Selecionados

Ano	**Referência**
2013	BAIER, T.; MENDLING, J. “Bridging abstraction layers in process mining: Event to activity mapping”. In: **Enterprise, Business-Process and Information Systems Modeling**. Springer, Berlin, Heidelberg, 2013. p. 109-123.
2014	BAIER, T.; MENDLING, J.; WESKE, M. “Bridging abstraction layers in process mining”. **Information Systems**, v. 46, p. 123-139, 2014.
2014	FERREIRA, D. R.; SZIMANSKI, F.; RALHA, C. G. “Improving process models by mining mappings of low-level events to high-level activities”. **Journal of Intelligent Information Systems**, v. 43, n. 2, p. 379-407, 2014.
2015	LEONI, M.; MAGGI, F. M.; VAN DER AALST, W. M. “An alignment-based

	framework to check the conformance of declarative process models and to preprocess event-log data". **Information Systems**, v. 47, p. 258-277, 2015.
2016	MANNHARDT, F., LEONI, M., REIJERS, H. A., VAN DER AALST, W. M., & TOUSSAINT, P. J. "From low-level events to activities-a pattern-based approach". In: **International conference on business process management**. Springer, Cham, 2016. p. 125-141.
2019	LEONI, M. "From Low-Level Events to Activities--A Session-Based Approach (Extended Version)". **arXiv preprint arXiv:1903.03993**, 2019.

As publicações identificadas, aresentadas nas Tabelas 3 e 4, podem ser agrupadas em três categorias: Correspondência (*Matching*), Alinhamento (*Alignment*) e Mineração (*Mining*).

Correspondência (*Matching*)

- Baier *et al.* (2014). Matching of events and activities-an approach based on constraint satisfaction.
- Baier *et al.* (2015a). Matching of events and activities: an approach based on behavioral constraint satisfaction.
- Baier *et al.* (2015b). Matching of events and activities-an approach using declarative modeling constraints.
- Baier *et al.* (2018). Matching events and activities by integrating behavioral aspects and label analysis.

Alinhamento (*Alignment*)

- Van Dongen *et al.* (2017). Aligning modeled and observed behavior: a compromise between computation complexity and quality.
- Leoni e Marrella (2017). Aligning Real Process Executions and Prescriptive Process Models through Automated Planning.
- Leoni, Maggi e Van der Aalst (2015) An alignment-based framework to check the conformance of declarative process models and to preprocess event-log data.
- Mannhardt *et al.* (2016). From low-level events to activities-a pattern-based approach.
- Leoni e Van der Aalst (2013). Aligning event logs and process models for multi-perspective conformance checking: An approach based on integer linear programming.

- Leoni, Maggi e Van der Aalst (2012). Aligning Event Logs and Declarative Process Models for Conformance Checking.
- Leoni, Van der Aalst e van Dongen (2012). Data-and resource-aware conformance checking of business processes.
- Adriansyah *et al.* (2012). Alignment Based Precision Checking.
- Adriansyah e Buijs (2012). Mining process performance from event logs.

Mineração (Mining)

- Ferreira, Szimanskie e Ralha (2013). Mining the low-level behaviour of agents in high-level business processes.
- Ferreira, Szimanskie e Ralha (2014). Improving process models by mining mappings of low-level events to high-level activities.
- Baier e Mendeling (2013). Bridging abstraction layers in process mining: Event to activity mapping.
- Baier, Mendeling e Weske (2014). Bridging abstraction layers in process mining.
- Leoni (2019). From Low-Level Events to Activities--A Session-Based Approach (Extended Version).

A próxima seção apresenta os trabalhos relacionados identificados, por meio da busca estruturada realizada, na revisão da literatura.

4.4 Análise dos Trabalhos Relacionados

A partir da revisão da literatura realizada, foi possível identificar trabalhos relacionados que abordam o problema da falta de mapeamento entre as perspectivas do processo e projeto de software. Esses trabalhos podem ser agrupados em três abordagens de solução para o problema: Correspondência (*Matching*), Alinhamento (*Alignment*) e Mineração (*Mining*). Correspondência (*Matching*) está relacionada a mapear registros produzidos durante a execução de um processo com atividades de um determinado modelo de processo. O Alinhamento é baseado no princípio de encontrar um alinhamento do registro de execução e do modelo de processo. Um alinhamento entre um registro de execução e um modelo de processo relaciona as ocorrências de atividades do registro de execução com as etapas de execução do modelo de processo. E em relação à Mineração é proposto descobrir o mapeamento por meio da aplicação de técnicas de Mineração de Processos.

4.4.1 Correspondência (*Matching*)

Em Baier *et al.* (2014), Baier *et al.* (2015a), Baier *et al.* (2015b) e Baier *et al.* (2018), são utilizadas abordagens de Correspondência. A abordagem apresentada em Baier *et al.* (2014) e Baier *et al.* (2015a) fornece meios para auxiliar o analista na identificação do mapeamento entre um modelo de processo e eventos em um registro de execução produzido por sistemas de informação. Essa abordagem consiste em três fases: (i) a primeira fase é automatizada, na qual se cria e resolve um problema de satisfação de restrições (*constraint satisfaction problem*) para reduzir o número de possíveis mapeamentos entre atividades e eventos; (ii) o resultado dessa fase é um conjunto de possíveis mapeamentos entre atividades e registros (o analista é orientado a selecionar o mapeamento correto dos possíveis mapeamentos derivados); e (iii) os mapeamentos derivados são usados para transformar automaticamente um ou muitos registros de execução para refletir as atividades do modelo de processo. Essa abordagem extrai perfis comportamentais do registro e do modelo para criar as restrições e reduzir o número de possíveis mapeamentos com eficiência. A abordagem apresentada em Baier *et al.* (2015b) utiliza restrições declarativas (*declarative constraints*) para apoiar algumas restrições. Declare (VAN DER AALST e PESIC, 2006) é nativamente uma linguagem declarativa de modelagem de processos, na qual é representada fluxos de trabalho usando regras temporais; essas regras visam estabelecer condições específicas para o desempenho de atividades em instâncias de processo.

Em Baier *et al.* (2018) é apresentada uma abordagem semi-automática que mapeia eventos para atividades utilizando *insights* da análise comportamental e análise de rótulo (correspondência baseada em rótulo). Essa abordagem aproveita informações de restrições comportamentais e análise linguística para resolver a complexidade combinatória e identificar o mapeamento entre um modelo de processo e registros de eventos. A abordagem apresentada em Baier *et al.* (2018) usa regras *Declare* derivadas de modelos de processos de negócios existentes e de registros de execução gerados por sistemas para instituir uma conexão entre modelos conceituais de processos e dados de execução operacional. Estes mapeamentos são reduzidos ainda mais usando técnicas do processamento de linguagem natural, que permitem uma correspondência com base em rótulos e fontes de conhecimento externas.

As abordagens apresentadas Baier *et al.* (2014) e Baier *et al.* (2015a) são aplicáveis apenas com relações *um para um* entre registros e atividades e requer pré-processamento para relações *um para muitos*. No entanto, em Baier *et al.* (2018), os

mapeamentos entre registros e atividades podem ser obtidos não apenas da maneira *um para um*, mas também nas relações de *um para muitos*.

4.4.2 Alinhamento (*Alignment*)

O alinhamento entre a execução de um processo e um modelo de processo é uma correspondência entre as atividades registradas nos registros de execução e as atividades permitidas no modelo de processo (LEONI; MARRELLA, 2017). De acordo com Van Dongen *et al.* (2017), confirmar que um modelo de processo está alinhado com as execuções reais do processo é talvez o aspecto mais desejado que um modelo de processo pode ter: modelos de processos alinhados são significativos para as organizações, porque as decisões estratégicas podem ser mais fáceis quando se baseia nos modelos do que nos registros puramente. As abordagens de alinhamento são propostas em Leoni, Maggi e Van der Aalst (2012), Adriansyah *et al.* (2012), Adriansyah e Buijs (2012), Leoni e Van Der Aalst (2013), Leoni, Maggi e Van der Aalst (2015), Mannardt *et al.* (2016), Leoni e Marrella (2017), Van Dongen *et al.* (2017). Estas abordagens destacam que os alinhamentos fornecem uma ferramenta muito poderosa ao relacionar o comportamento observado com o comportamento modelado.

Em Leoni, Maggi e Van der Aalst (2012), Adriansyah *et al.* (2012), Leoni e Van Der Aalst (2013), Leoni, Maggi e Van der Aalst (2015) Leoni e Marrella (2017), a noção de alinhamento foi introduzida para oferecer suporte à verificação de conformidade. Ruiz-Rube, Dodero e Colomo-Palacios (2015) destacam que os alinhamentos são potentes para observar não-conformidades entre o comportamento detectado, conforme registrado nos registros de execução, e o comportamento estabelecido, representado em um modelo de processo. De acordo com Leoni, Maggi e Van der Aalst (2012), o alinhamento fornece diagnósticos sofisticados que identificam onde os desvios ocorrem e qual a gravidade deles. Uma vez que o alinhamento tenha sido realizado, é possível quantificar a conformidade e analisar as diferenças entre modelo e realidade (LEONI; MAGGI; VAN DER AALST, 2012).

Leoni, Maggi e Van der Aalst (2012) propõem para alinhar registros da base de dados de execução e modelos de processo declarativos. Eles usam o algoritmo *A** (VAN DER AALST; ADRIANSYAH e VAN DONGEN, 2012) para encontrar o alinhamento ideal para cada registro da base de dados de execução. Além de simplesmente retornar um alinhamento ideal para cada registro, eles também fornecem ao analista de processos um resumo que permite ter uma visão geral da conformidade do modelo em relação a todos os registros de execução. Em Leoni *et al.* (2012b), é descrita uma abordagem que alinha o modelo e os registros, considerando

as perspectivas de fluxo de controle, dados e recursos. Neste alinhamento, é necessário relacionar os registros, da base de dados de execução, aos elementos do modelo e vice-versa, apresentando assim uma maneira como os registros de execução podem ser reproduzidos no modelo de processo (LEONI; MAGGI e VAN DER AALST, 2012). Essa abordagem também usa o algoritmo A* (VAN DER AALST, ADRIANSYAH e VAN DONGEN, 2012) para encontrar os elementos do processo para cada registro da base de dados de execução. Em Leoni, Maggi e Van der Aalst (2012) é considerada a métrica de aptidão (*fitness*) (um modelo com boa aptidão permite a maior parte do comportamento observado no registro de eventos). Em Adriansyah *et al.* (2012), eles consideram na métrica de precisão (*precision*) (a precisão penaliza um modelo de processo por permitir um comportamento improvável devido ao comportamento observado no registro de eventos). Eles desenvolveram uma abordagem que primeiro alinha o modelo e o registro de eventos. Portanto, esse pré-alinhamento entre registros e modelo torna possível medir a precisão com mais assertividade (ADRIANSYAH *et al.*, 2012). Em Leoni e Van Der Aalst (2013), é apresentada uma técnica que considera dados, recursos e tempo ao verificar a conformidade do processo. A técnica proposta utilizou estratégias para criar alinhamentos de fluxo de controle, mas foram estendidas para incluir outras perspectivas (dados, recursos e tempo). Para estender alinhamentos com outras perspectivas, um problema adicional de ILP (Programação Linear Inteira) é construído e resolvido para cada rastreamento dos registros. A Programação Linear Inteira (ILP) pode ser usada para resolver os chamados problemas de otimização linear (LEONI e VAN DER AALST, 2013). Leoni, Maggi e Van der Aalst (2015) propõem a implementação de uma estrutura para a análise da execução de processos declarativos. Esta estrutura é baseada no princípio de criar um alinhamento de um registro de eventos e um modelo de processo. Leoni, Maggi e Van der Aalst (2015) também usam o algoritmo *A** para encontrar, para cada registro, um alinhamento ideal, isto é, um alinhamento que minimize o custo dos desvios.

Outra abordagem para derivar um alinhamento entre os registros de execução e um modelo de processo é proposta em Leoni e Marrella (2017). Dado um modelo de processo e sua execução registrada em uma base de dados de execução, Leoni e Marrella (2017) ilustraram como o problema de calcular alinhamentos ideais poderia ser formulado como um problema de planejamento no PDDL (*Planning Domain Definition Language*). De acordo com Leoni e Marrella (2017) o PDDL é a linguagem de codificação para tarefas de planejamento e permite representar explicitamente estados do mundo e ações por meio de um domínio de planejamento e instanciar esse

domínio com objetos concretos, um estado inicial e uma especificação do objetivo. Mais informações sobre PDDL em McDermott *et al.* (1998).

Mannhardt *et al.* (2016) propõem um método de abstração supervisionado baseado em padrões comportamentais de atividades que capturam o conhecimento do domínio sobre a relação entre atividades e registros de eventos. Através de um alinhamento entre padrões de atividade e registros de eventos de baixo nível, é obtido um registro de eventos abstratos. Os padrões de atividade codificam suposições sobre como as atividades de alto nível se manifestam em termos de eventos de baixo nível registrados. Eventos no registro de eventos abstratos correspondem a instâncias de atividades conhecidadas pelos especialistas.

Adriansyah e Buijs (2012) usaram alinhamentos entre um modelo de processo e registros de execução para melhorar manualmente os modelos de processos obtidos pelos algoritmos de mineração, e assim projetar informações de desempenho neles. E em Van Dongen *et al.* (2017) é proposto um algoritmo para computar alinhamentos cuja natureza está entre as técnicas apresentadas em Adriansyah *et al.* (2012) e Taymouri e Carmona (2016). Eles apresentaram uma abordagem incremental para calcular alinhamentos para um determinado registro e modelo usando ILP (Programação Linear Inteira).

4.4.3 Mineração (*Mining*)

A mineração de processos consiste em extrair conhecimento dos registros de execução disponíveis nos sistemas de informação atuais (LEONI e MARRELLA, 2017). Esses registros podem ser usados para analisar o processo usando técnicas de mineração de processo para descobrir o processo efetivamente seguido ao longo do projeto, medir a conformidade com um modelo de processo ou aprimorar os modelos existentes com informações de desempenho (VAN DER AALST, 2011). As técnicas de mineração de processos enfrentam um desafio importante: deve ser conhecido o mapeamento dos registros produzidos pelos sistemas de informação com as atividades correspondentes dos modelos de processo (BAIER *et al.*, 2018). As técnicas de Alinhamento e Correspondência apontam para benefícios na área de mineração de processos. As abordagens que utilizam estratégias de mineração são propostas Ferreira, Szimanskie e Ralha (2013), Baier e Mendeling (2013), Ferreira, Szimanskie e Ralha (2014), Baier, Mendeling e Weske (2014) e Leoni (2019).

Em Ferreira, Szimanski e Ralha (2013) é proposta uma abordagem para descobrir esse mapeamento, relacionada à mineração. Esta abordagem proposta por Ferreira, Szimanski e Ralha (2013) identifica uma lacuna entre o alto nível de

abstração em que os processos de negócios são modelados e a natureza de baixo nível dos eventos que são registrados durante a execução do processo. Os autores destacam que a análise isolada dos registros fornece pouca percepção de como um processo está sendo executado em tempo de execução, porque o mapeamento entre eventos de baixo nível e atividades de alto nível está ausente. Eles propõem uma abordagem de mineração para descobrir esse mapeamento. Em Ferreira, Szimanskie e Ralha (2014) é apresentada também uma abordagem para minerar mapeamentos entre os registros de eventos e as atividades do modelo. De acordo com Ferreira, Szimanskie e Ralha (2014) um mapeamento é uma maneira de estabelecer um relacionamento entre cada tipo de evento observado no registro de eventos e uma atividade de alto nível específica definida no processo de negócios. Em Ferreira, Szimanskie e Ralha (2014) a abordagem é capaz de explorar possíveis mapeamentos, selecionar um mapeamento candidato e fornecer sugestões sobre como o modelo de processo pode ser aprimorado para capturar o comportamento observado no registro de eventos.

Baier e Mendeling (2013) e Baier, Mendeling e Weske (2014) desenvolveram uma abordagem que visa abstrair um registro de eventos para o mesmo nível de abstração necessário ao negócio. Portanto, é capturado o conhecimento de domínio sobre mapeamentos de registros de eventos para atividades de maneira formalizada. Um algoritmo para clusterizar eventos para instâncias de atividades foi proposto. Em Baier, Mendeling e Weske (2014) foi desenvolvido um método para lidar com concorrência e lidar com relações *muitos para muitos* entre eventos e atividades. Portanto, foi codificado explicitamente o conhecimento do domínio na função de mapeamento para obter o mesmo nível de abstração usado nas atividades de negócio definidas.

Leoni (2019) apresenta uma técnica onde os registros de eventos são divididos em sessões e cada sessão é abstraída como uma única execução de atividade de alto nível. A abstração é baseada em uma combinação de métodos automáticos de clusterização e visualização. A idéia é que eventos do mesmo registro possam ser agrupados em sessões, de modo que a distância entre o último evento de uma sessão e o primeiro evento da sessão subsequente seja maior que um limite definido pelo usuário. A técnica de clusterização de eventos de baixo nível em atividades de alto nível tem como ponto inicial um registro de eventos, onde todos os eventos são divididos em sessões, que são agrupados em clusteres, sendo criado o registro de eventos abstrato.

As ameaças à validade da revisão da literatura são descritas a seguir.

4.5 Ameaças à Validade

Os resultados apresentados nesta revisão podem ter sido influenciados por certas limitações. Em relação às ameaças internas à validade, deve-se mencionar que nos filtros de avaliação de artigos apenas um pesquisador analisou os resultados, por questão de tempo e disponibilidade de pesquisadores para participar deste estudo.

Além disso, a extração dos dados foi realizada por apenas um pesquisador, o que pode acarretar algum risco de viés. Vale a pena destacar que a *string* de busca pode não conter todas as palavras-chave relevantes, causando a perda de alguns estudos relevantes. No entanto o planejamento, que contém a *string* de busca foi revisada por outro pesquisador. Um fator relevante a ser citado é que os artigos de controle foram devolvidos e analisados ao executar esta busca estruturada, gerando evidências sobre a correção a *string* de busca. Algumas bases de dados eletrônicas, como Springer Link e ACM Digital Library, não foram consideradas neste artigo, foi utilizada somente a base Scopus, portanto, é possível que estudos relevantes não tenham sido indexados por busca. No entanto, acreditamos que o banco de dados da base selecionada foi suficiente para obter noção dos trabalhos que tratam sobre o mapeamento entre as perspectivas.

4.6 Considerações Finais

Este Capítulo apresentou o planejamento e execução da revisão da literatura, realizada por meio de uma busca estruturada, realizada para identificar os trabalhos relacionados ao problema tratado nesta tese. A abordagem de *snowballing* foi realizada como forma de aumentar a abragência da revisão. Após análise dos resultados da execução, os trabalhos identificados foram subdivididos em três categorias: Correspondência (*Matching*), Alinhamento (*Alignment*) e Mineração (*Mining*).

O próximo Capítulo apresenta as Operações de Instanciação que pretendem promover uma transição entre processo e projeto, por meio de um mecanismo de mapeamento entre as perspectivas de processo e projeto. O mecanismo de mapeamento cria vínculos entre as perspectivas, por meio de uma estrutura de mapeamento entre os elementos utilizados para modelar o processo de software e os elementos usados para executar os projetos de software, reduzindo o distanciamento entre eles.

5 Mapeamento entre Processos de Software e Projetos de Software

Este capítulo apresenta a solução proposta nesta tese para mapear as perspectivas de processo e projeto, de forma a possibilitar que processos de software sejam explicitamente identificados nos projetos de software. Um conjunto de operações de instanciação é apresentado. Estas operações pretendem apoiar adequadamente a transição entre as perspectivas de processo e projeto. A partir de uma estrutura de mapeamento entre os elementos utilizados para modelar o processo de software e os elementos usados para executar os projetos de software, objetiva-se minimizar o distanciamento entre processo e projeto, por meio da criação de vínculos entre eles.

5.1 Introdução

Os estudos exploratórios, apresentados no Capítulo 3, evidenciaram o problema tratado nesta tese, no qual **os processos não são explicitamente identificados nos planos de projetos**. Esta tese propõe um mecanismo de mapeamento entre as perspectivas de processo e projeto, como forma de responder a questão de pesquisa que norteou as pesquisas desta tese "**Como explicitar o mapeamento entre as perspectivas de processo e de projeto de software?**".

Neste sentido, esta tese apresenta um mecanismo para mapear elementos de representação de modelos de processos e de planos de projetos com base em operações explícitas de instanciação que capturam a lógica de mapeamento de processo para projeto.

Desta forma, pretende-se realizar a aproximação entre perspectivas, por meio do mapeamento entre os elementos utilizados para modelar o processo de software e os elementos usados para executar os projetos de software. Para efetuar este mapeamento é realizada a aplicação de operações de instanciação durante a criação de tarefas do projeto de software. Essas operações permitem representar de forma explícita o mapeamento entre os elementos do modelo de processo de software e as tarefas criadas ao longo do projeto de software.

Este capítulo apresenta na Seção 5.2 a visão geral da solução definida nesta pesquisa para a realização do mapeamento entre as perspectivas de processo de

software e projeto de software. As Operações de Instanciação são descritas detalhadamente na Seção 5.3 e na Seção 5.4 apresenta uma análise comparativa entre a solução e os trabalhos relacionados apresentados no Capítulo 4. Por último na Seção 5.5 são apresentadas as considerações finais.

5.2 Visão Geral da Solução

Considerando o objetivo específico apresentado no Capítulo 1 na Seção 1.3: **Definir um conjunto de orientações para a realização do mapeamento entre as perspectivas no contexto de projetos de software**; está seção apresenta a visão geral da solução proposta, bem como um fluxo para sua aplicação. A Figura 5 ilustra a visão geral da solução alinhada às perspectivas de processo e projeto de software.

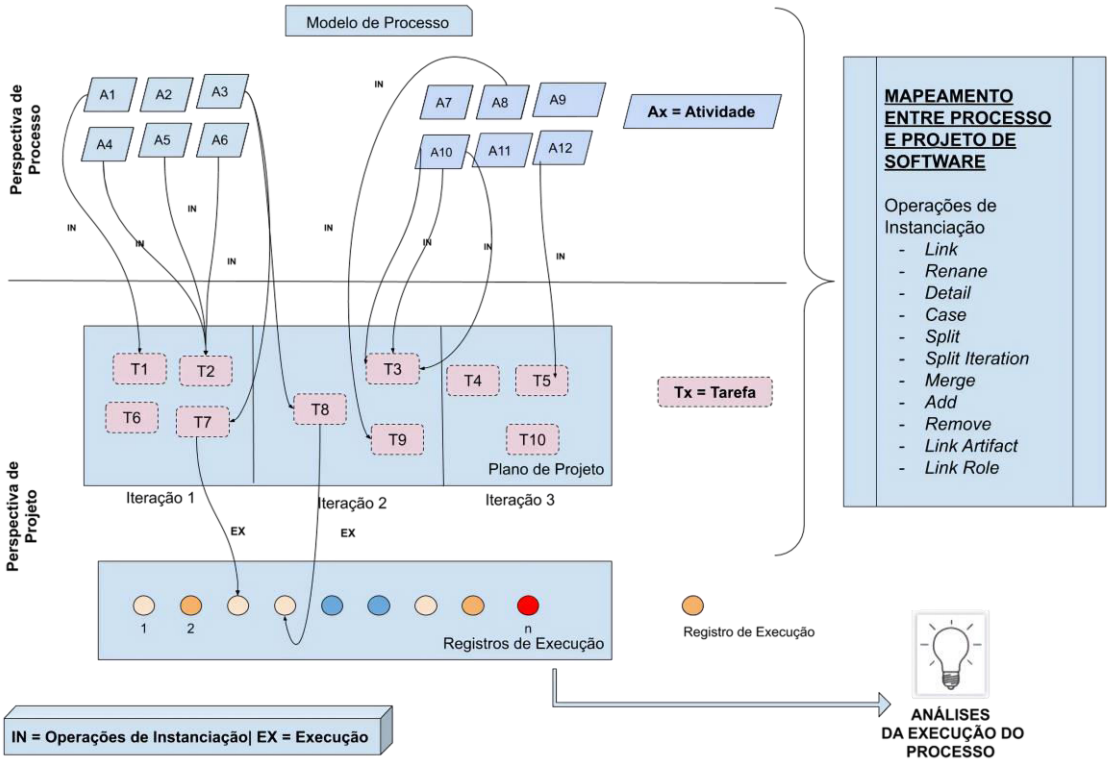


Figura 5 – Visão geral da solução de mapeamento entre as perspectivas. Fonte: A Autora.

Consideram-se como pré-requisitos da solução (i) a definição do modelo do processo de software, composto pela definição das atividades, sequência, papéis e artefatos, e (ii) o planejamento do projeto realizado pelo gerente que envolve a seleção do processo, definição da equipe, cadastro das iterações e instâncias.

Para dar início ao mapeamento o modelo de processo de software deve ser obtido, para que os elementos do processo, tais como atividade, papel e artefato, sejam identificados; e posteriormente serem usados para realizar o mapeamento entre

as perspectivas. O ponto central da solução são as operações de instanciação, que são aplicadas na criação de tarefas, estabelecendo o mapeamento entre representações de processo e projeto de software. Na Figura 5 está representado o mapeamentyo entre atividades do processo (Ax) e tarefas do projeto (Tx). Uma vez criadas as tarefas, as mesmas são executadas, gerando os registros de execução. Por meio das operações de instanciação, é realizada a representação de elementos do processo de software no plano do projeto; e em consequência nos registros de execução. Esta representação é feita durante a instanciação do processo de software definido no projeto de software. A partir desta representação é possibilitado realizar mais facilmente análises da execução dos processos.

A solução proposta se aplica ao contexto de organizações de desenvolvimento de software que possuem um processo de software, o qual é utilizado como referência ao longo dos projetos de desenvolvimento de software. A cada projeto é feito o planejamento das tarefas, sendo criado um plano de projeto. Este plano gerado é carregado para uma ferramenta de gerenciamento de tarefas, onde as tarefas são criadas. A partir desta etapa de criação do plano de projeto, que na área de processos de software é chamada de Instanciação, dar-se-á início à Execução. Ao longo da execução do processo, as tarefas criadas são efetivamente realizadas por ferramentas ou por desenvolvedores. A base de dados de processo, gerada pelo uso de ferramentas de gerenciamento de tarefas, armazenam uma rica fonte de dados.

Quando se aplica à solução para o mapeamento, nesta fonte rica de dados, a mesma pode ser utilizada para a identificação do processo de software efetivamente executado, bem como para a verificação de sua conformidade com o processo de software de referência. Estas aplicações serão discutidas no Capítulo 6.

Algumas questões que podem ser mais facilmente respondidas quando as perspectivas de processos e projetos estão mapeadas são:

I. Qual o modelo de processo de software efetivamente seguido ao longo do projeto de software?
II. Quais atividades previstas no processo de software de referência do projeto foram executadas?
III. Quais atividades previstas no processo de software de referência do projeto não foram executadas?
IV. Atividades não previstas no processo de software de referência do projeto foram executadas?

Estas questões são pertinentes porque ao longo da execução dos projetos desvios podem ocorrer em relação ao processo de referência, e a obtenção destas respostas podem ser úteis para algumas partes interessadas (ex.: gerentes de

projeto), para facilitar a avaliação do processo e subsidiar tomadas de decisão mais eficientes. Essas decisões, tais como implantação e/ou melhorias nos modelos de processo de software, utilizado como referência nas organizações, podem contribuir para a maturidade do mesmo.

A Figura 6 apresenta o processo de aplicação da solução proposta nesta tese.

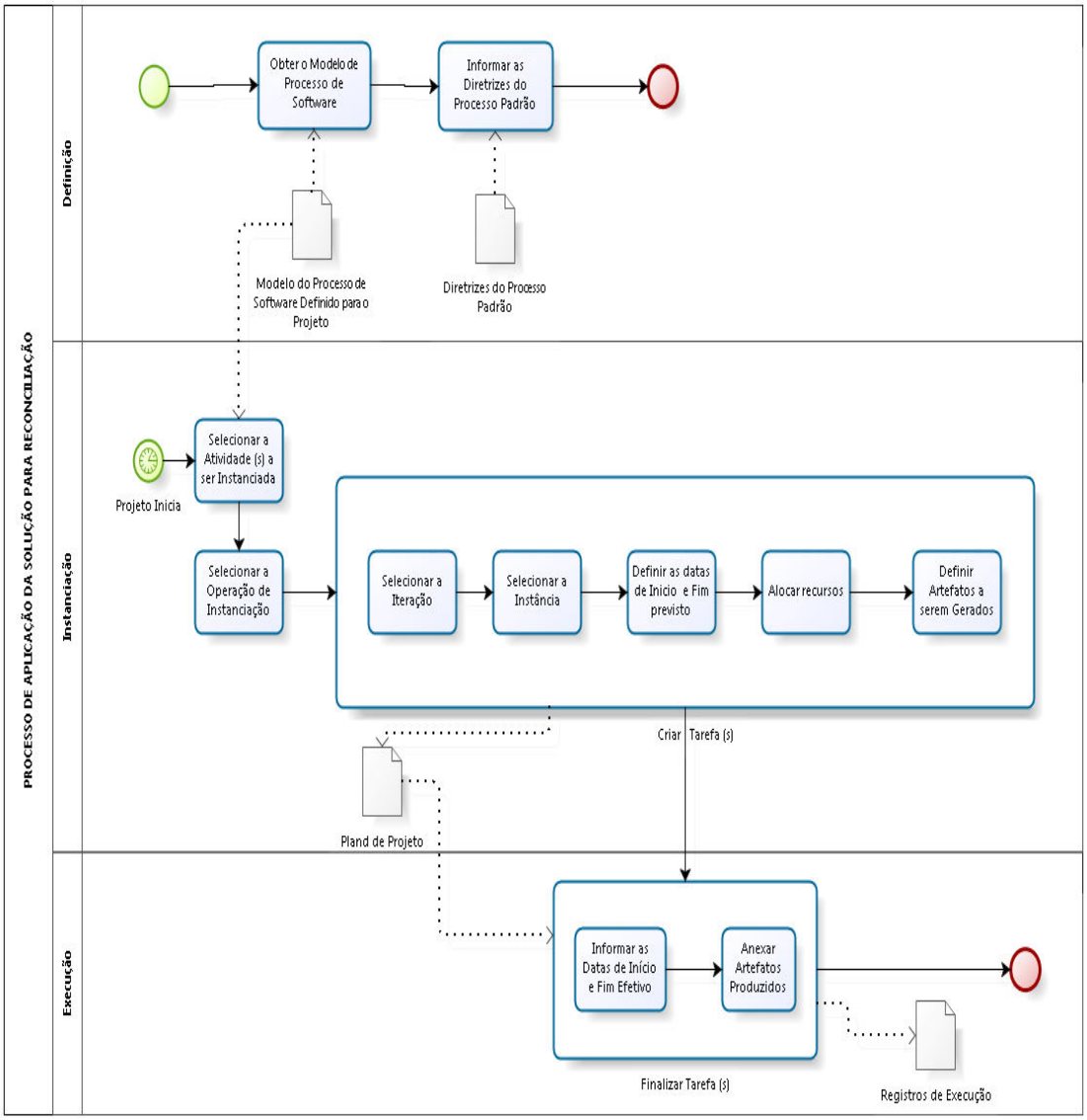


Figura 6 - Processo de Aplicação da Solução. Fonte: A Autora.

O processo de aplicação da solução está dividido em três etapas: *Definição, Instanciação* e *Execução*. A etapa de ***Definição*** envolve as atividades de obtenção do modelo de processo de software definido para o projeto, bem como das diretrizes do processo padrão, quando houver. O modelo do processo de software obtido deve ser usado como guia ao longo do projeto de desenvolvimento de software. A definição do processo, bem como das diretrizes, estão fora do escopo desta solução. As etapas de ***Instanciação*** e ***Execução*** compreendem a criação e finalização de tarefas,

respectivamente. Após a obtenção do modelo de processo, pode ser realizada a criação das tarefas, bem como, a finalização das mesmas. A etapa de ***Instanciação*** compreende as seguintes atividades: *Selecionar a atividade a ser instanciada*, *Selecionar a Operação de Instanciação e Criar Tarefa* (*Selecionar Iteração, Selecionar Instância, Definir as datas de Início e Fim previsto; Alocar recursos e Definir Artefatos a serem gerados*). Ao final o plano de projeto é criado. A etapa de ***Execução*** compreende as atividades: *Finalizar Tarefa* (I*nformar as datas de Início e Fim efetivo* e *Anexar Artefatos Gerados*). Uma vez finalizadas as tarefas, as mesmas podem ser consideradas como registros de execução.

A Tabela 5 apresenta a descrição das atividades da etapa de Definição para obtenção dos modelos de processo de software.

Tabela 5 - Atividades da etapa de Definição

Definição
Obter o Modelo de Processo de Software
Envolve a obtenção do processo de software da Organização no formato de um modelo de processo.
Obter as Diretrizes do Processo de Software
Envolve a obtenção das diretrizes de adaptação do processo de software.

A Tabela 6 apresenta a descrição da etapa de Instanciação visando o mapeamento entre as perspectivas, compreendendo suas atividades e tarefas.

Tabela 6 - Atividades da etapa de Instanciação

Instanciação
Criar o Plano do Projeto
Envolve a criação do plano de projeto que é composto das tarefas do projeto de desenvolvimento de software
Selecionar a Atividade a ser Instanciada
Envolve a seleção da atividade do processo que será instanciada.
Selecionar a Operação de Instanciação
Envolve a seleção da Operação de Instanciação, que apoia a criação de tarefas para a equipe, realizando o vínculo entre um elemento do plano de projeto e um elemento do processo de software.

Criar Tarefa *Envolve a criação de tarefas onde são realizadas as seguintes tarefas:* **Selecionar a Iteração**; **Selecionar a Instância**; **Definir as datas de Inicio e Fim previsto**; **Alocar recursos** e **Definir Artefatos a serem gerados**. Esta atividade gera o **Plano de Projeto**.

A Tabela 7 apresenta a descrição da etapa de Execução visando o mapeamento entre as perspectivas, compreendendo suas atividades e tarefas.

Tabela 7 - Atividades da etapa de Execução

Execução
Finalizar Tarefa *Envolve a finalização de tarefas onde são realizadas as seguintes tarefas:* ***Informar*** **as Datas de Início e Fim efetivo** e **Anexar Artefatos Gerados.** Esta atividade gera os **Registros de Execução**.

Na próxima seção o conjunto de Operações de Instanciação proposto é apresentado em detalhes.

5.3 Operações de Instanciação

Considerando o objetivo específico apresentado no Capítulo 1 na Seção 1.3 – **Definir um mecanismo de apoio ao mapeamento entre as perspectivas de processo e projeto**; um conjunto de operações de instanciação é proposto para representar o mapeamento entre as perspectivas, permitindo assim que os elementos da perspectiva de processo sejam registrados na perspectiva do projeto.

As operações foram definidas a partir da análise de planos de projeto de projetos de desenvolvimento de software, com base na experiência dos pesquisadores envolvidos nesta tese em gerenciamento de projetos de desenvolvimento de software, e considerando o estado da arte da literatura (OMG, 2008, ADRIANSYAH e BUIJS, 2012, MARTINS e SILVA, 2016 e BASTARRICA *et al.*, 2017). Nos planos de projeto analisados, foi identificada manualmente, para cada tarefa do plano do projeto, a atividade correspondente do processo de software definido, com o apoio dos profissionais envolvidos no desenvolvimento do projeto. Por meio desta análise, foi possível identificar três tipos de relacionamentos entre as atividades do processo e as tarefas do projeto: *um para um*, *um para muitos* e *muitos para um*. Neste sentido, as

operações propostas estabelecem relacionamentos destes tipos entre elementos do processo e tarefas do projeto.

As mudanças ocorridas na instanciação introduzem conceitos geralmente não definidos nos modelos de processo, tais como iteração, que impacta no entendimento do processo de software efetivamente executado. Além disso, a instanciação leva em consideração outros aspectos, como tempo, orçamento e alocação de equipe, que podem influenciar se ou quando as tarefas devem ser executadas. Outro fator observado em projetos de software é a falta de informações sobre a instância do processo à qual uma determinada tarefa pertence. Uma sequência de execução do processo também é chamada de instância do processo (BAIER *et al.*, 2015a). De acordo com Van der Aalst *et al.* (2011) além do problema de relacionar registros de execução às atividades do processo, existe o problema de relacionar registros com suas respectivas instâncias do processo. Para mitigar este problema, na aplicação de cada operação é necessário informar a instância do processo. Neste sentido, é desejável implementar mecanismos que permitam que elementos do processo de software definido estejam explícitos durante a instanciação e execução do processo.

As operações de instanciação podem ser usadas ao criar o plano do projeto e podem capturar mapeamentos simples, como a ligação (*link*) entre dois elementos, ou mapeamentos complexos envolvendo vários elementos, como agrupamento (*merge*) ou divisão (*split*). A Figura 7 apresenta o modelo conceitual resultante da aplicação das operações de instanciação. A Figura 7 (lado esquerdo) apresenta um trecho de um modelo conceitual de processo, que está de acordo com os metamodelos SPEM (OMG, 2008) e BPMN (OMG, 2014), composto de elementos de processo de software como *Process*, *Activity*, *SequenceFlow*, *Artifact* e *Role*. Projetos de software são instâncias de modelos de processos e podem ser representados por meio de planos de projeto. A Figura 7 (lado direito) apresenta o modelo conceitual do projeto, que é resultante da instanciação do processo de software e foi desenvolvido, com base nos conceitos do PMBOK (*Project Management Body of Knowledge*) (PMI, 2017) e na análise dos dados das tarefas, que são registrados nas ferramentas utilizadas durante os projetos de desenvolvimento de software, incluindo os seguintes elementos: *Project*, *Task*, *TaskArtifact*, *TaskActor*, *Instance* e *Iteration*.

A aplicação das operações de instanciação pode definir a associação entre *Process* e *Project*, *Process* e *Instance*, *Task* e *Activity*, *Role* e *TaskActor*, e *Artifact* e *TaskArtifact*. Nesse sentido, as operações explicitam o mapeamento entre as perspectivas.

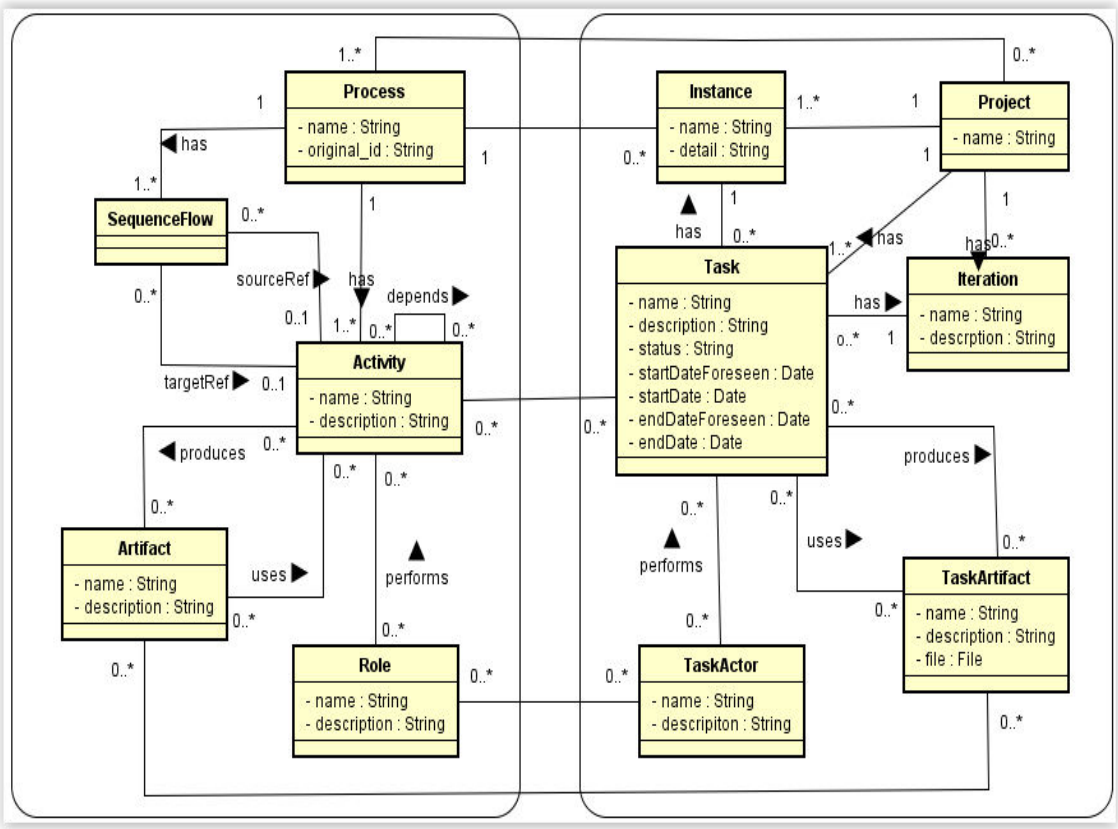


Figura 7 - Mapeamento entre a atividade do processo e a tarefa do projeto. Fonte: A Autora.

O mapeamento entre as perspectivas deve ser introduzido utilizando as operações, apresentadas na Tabela 8, que relacionam as tarefas (projeto) às atividades (processo). Também são relacionados o ator (projeto) e o papel (processo) e artefato produzido (projeto) e o artefato esperado (processo). A atividade do processo de software de referência para o projeto é registrada em cada tarefa, exceto para as operações *Add* (a tarefa criada não possui atividade correspondente associada) e *Remove* (nenhuma tarefa é criada a partir da atividade). O mapeamento entre *Role* e *Actor* e *Artifact* e *TaskArtifact* são introduzidos pelas operações *Link Artifac*t e *Link Role*, respectivamente.

Conforme descrito anteriormente, a aplicação de operações de instanciação estabelece um vínculo entre um plano de projeto (projeto de software) e modelos de processo (processo de software). Então, para usar as operações de instanciação, é necessário que a organização tenha pelo menos um processo de software definido, que é utilizado como referência ao longo dos projetos de software. Gerentes de projeto ou engenheiros de processos responsáveis por projetar e implementar processos, gerentes ou *coaches* que os auxiliem e os orientem, geralmente possuem o conhecimento necessário para garantir a precisão das operações de instanciação,

uma vez que devem conhecer o processo de software definido usado como referência para um determinado projeto.

A Tabela 8 apresenta o nome e descrição de cada operação proposta.

Tabela 8 - Operações de Instanciação Propostas

Nome	Descrição
Link	Estabelece uma ligação entre uma atividade do processo e uma única tarefa do plano de projeto. A tarefa deve ter o mesmo nome da atividade do processo associada.
Detail	Estabelece uma ligação entre uma atividade do processo e uma única tarefa do plano de projeto. O nome da tarefa deve ser mais detalhado que o nome da atividade definida no processo, mas com o mesmo significado.
Rename	Estabelece uma ligação entre uma atividade do processo e uma única tarefa do plano de projeto. O nome da tarefa deve ser diferente do nome da atividade de processo associada.
Case	Estabelece uma ligação entre cada atividade prevista para uma instância do processo e as tarefas do plano de projeto. Cada nome de tarefa pode ser mais detalhado que o nome da atividade, de acordo com a necessidade do contexto do projeto de software. Todas as tarefas serão associadas a uma instância do processo.
Split	Estabelece uma ligação entre uma atividade do processo e duas ou mais tarefas do plano de projeto, em uma iteração. O nome de cada tarefa deve ser diferente do nome da atividade de processo associada.
Split Iteration	Estabelece uma ligação entre uma atividade do processo e duas ou mais tarefas do plano de projeto e as distribui em diferentes iterações. O nome de cada tarefa deve ser diferente do nome da atividade de processo associada.
Merge	Estabelece uma ligação entre duas ou mais atividade do processo e uma única tarefa do plano de projeto.
Add	Cria uma tarefa no plano de projeto que não possui uma atividade prevista no processo de software.
Remove	Uma atividade do processo de software não tem uma tarefa correspondente no plano de projeto.
Link Artifact	Estabelece uma ligação entre um artefato associado a uma atividade de processo e um artefato produzido por uma tarefa do plano do projeto. O artefato produzido por uma tarefa deve ter o mesmo tipo que a atividade de processo associada ao artefato.
Link Role	Estabelece uma ligação entre o papel associado a uma atividade de processo e o ator responsável por uma tarefa do plano do projeto.

Considera-se que ao usar as operações de instanciação propostas, o resultado do mapeamento estabelecido forneça informações que permitam realizar mais facilmente análises dos processos de software efetivamente executados nos projetos

de software, bem como verificar a conformidade entre as perspectivas do processo e do projeto.

Cabe ressaltar que as operações de instanciação apresentadas podem ser confundidas com operações de adaptação, porém se diferenciam pelo tempo em que ocorrem e também pela estrutura de dados resultante. Adaptação de Processo é a ação de adaptar um processo de software definido para encontrar as necessidades específicas de uma organização ou projeto (PEDREIRA *et al.*, 2007), e consiste em excluir, modificar ou adicionar novos elementos e/ou relacionamentos a um processo de software (PEREIRA *et al.*, 2008). Nesse sentido, adaptação de processo é o ato de ajustar as definições e/ou de particularizar os termos de uma descrição de processo geral para derivar um novo processo aplicável em um ambiente alternativo (MARTÍNEZ-RUIZ *et al.*, 2012), ou seja, as operações de adaptação tem como resultado um novo processo que se aplica às necessidades do projeto, e são definidas na perspectiva de processo. Enquanto, as operações de instanciação ocorrem na perspectiva de projeto, considerando informações do contexto do projeto de desenvolvimento, tais como prazo e disponibilidade de recursos.

Cada operação será apresentada em detalhes na seção seguinte.

5.3.1 Descrição detalhada das Operações de Instanciação

Cada Operação de Instanciação está descrita utilizando a estrutura apresentada na Tabela 9. Este formato de descrição foi baseado na forma como os Padrões de Projetos são descritos em Gamma *et al.* (2000).

Tabela 9 - Formato padrão para descrição de cada Operação de Instanciação

Nome	*[Nome da Operação]*
Operação	*[Representação da operação]*
Descrição	*[O que a operação faz* *Que tipo de problema resolve.]*
Motivação	*[Um cenário que ilustre o problema e como o comportamento de cada operação resolvem o problema do cenário de exemplo.]*
Aplicabilidade	*[Pretende-se descrever por meio de respostas para as seguintes questões:* *Quais são as situações em que a operação pode ser aplicada?* *Quais são os exemplos de situações que a operação pode tratar?* *Como você pode reconhecer as situações de aplicabilidade?]*
Participantes	*[Explicação de cada elemento do modelo conceitual envolvidos na aplicação da operação.]*

Relacionamento	*[Tipo de relacionamento entre os participantes envolvidos] [um para um, um para muitos e muitos para um]*
Pseudocódigo	*[Pseudocódigo do funcionamento da operação.]*

Cada operação está descrita em detalhes nas subseções seguintes, utilizando a estrutura apresentada na Tabela 9.

5.3.1.1 Operação Link

Nome

Link

Operação

task.createLink (iteration: Iteration, instance: Instance, project: Project, processActivity: Activity)

Descrição

Estabelece uma ligação entre uma atividade do processo e uma única tarefa do plano de projeto. A tarefa deve ter o mesmo nome da atividade do processo associada.

Motivação

Atividades do processo de software definido para o projeto são instanciadas, porém a falta de vínculo entre o processo e o plano de projeto dificulta a extração de informações do processo ao longo do projeto de software. A criação de tarefas com o mesmo nome da atividade e a representação da atividade do processo associada à tarefa do plano de projeto ajuda na redução do distanciamento entre processo e projeto.

Aplicabilidade

Esta operação se aplica à situação na qual se pretende criar uma tarefa no plano de projeto igual à atividade prevista no processo de software definido. Nesta situação a atividade prevista atende às necessidades em sua descrição e propósito. Esta operação cria uma tarefa no plano de projeto com o mesmo nome de atividade de processo e registra a atividade do processo definido.

Participantes

Activity: Atividade do processo de software definido para o projeto

Task: Tarefa do projeto presente no plano de projeto

Relacionamento

Um para Um entre *Activity* e *Task*.

Pseudocódigo

```
function task.createLink(iteration: Iteration, instance:
Instance,  project: Project, processActivity: Activity)
begin
      self.iteration = iteration
      self.instance = instance
      self.project = project
      self.processActivity = processActivity
      self.name = processActivity.name
      self.defineTask(task_description,task_status,
      task_startDateForeseen, task_startDate,
      task_endtDateForeseen, task_endDate)
      return task
end
```

5.3.1.2 Operação Detail

Nome

Detail

Operação

task.createDetail (iteration: Iteration, instance: Instance, project: Project, processActivity: Activity, task_name_detail: String)

Descrição

Estabelece uma ligação entre uma atividade do processo e uma única tarefa do plano de projeto. O nome da tarefa deve ser mais detalhado que o nome da atividade, mas com o mesmo significado.

Motivação

As atividades previstas no processo de software podem ser definidas em alto nível, sendo necessário um maior detalhamento em sua instanciação no projeto de software. A criação de tarefas com o nome da atividade mais detalhado e a representação da atividade do processo associada à tarefa do plano de projeto ajuda na redução do distanciamento entre processo e projeto.

Aplicabilidade

Esta operação se aplica à situação na qual se pretende instanciar uma atividade do processo de software, porém no contexto do projeto de software, a tarefa a ser criada deve ter o nome da atividade prevista mais detalhado. Nesta situação, a atividade prevista atende às necessidades em sua descrição e propósito, sendo necessário um melhor detalhamento. Esta operação cria uma tarefa no plano de projeto com o nome mais detalhado que o nome da atividade de processo, mas com o mesmo significado, e registra a atividade do processo de referência.

Participantes

Activity: Atividade do processo de software definido para o projeto

Task: Tarefa do projeto presente no plano de projeto

Relacionamento

Um para Um entre *Actitivy* e *Task*.

Pseudocódigo

```
function task.createLink (iteration: Iteration, instance:
Instance, project: Project, processActivity: Activity,
task_name_detail: String)
begin
      self.iteration = iteration
      self.instance = instance
      self.project = project
      self.processActivity = processActivity
      self.name = processActivity.name + task_name_detail
      self.defineTask(task_description, task_status,
task_startDateForeseen,
      task_startDate, task_endtDateForeseen, task_endDate)
      return task
end
```

5.3.1.3 Operação Rename

Nome

Rename

Operação

task.createRename (iteration: Iteration, instance: Instance, project: Project, processActivity: Activity, task_name: String)

Descrição

Estabelece uma ligação entre uma atividade do processo e uma única tarefa do plano de projeto. O nome da tarefa deve ser diferente do nome da atividade de processo associada.

Motivação

As atividades previstas no processo de software podem ser definidas em alto nível e não descrevem especificamente o que se pretende realizar na tarefa do projeto, sendo necessário renomeá-las em sua instanciação no projeto de software. A criação de tarefas com o nome diferente da atividade associada, para se adequar ao contexto do projeto e a representação da atividade do processo associada à tarefa do plano de projeto ajuda na redução do distanciamento entre processo e projeto.

Aplicabilidade

Esta operação se aplica à situação na qual se pretende instanciar uma atividade do processo de software, porém no contexto do projeto de software, a tarefa a ser criada deve ter o nome diferente. Nesta situação a atividade prevista atende às

necessidades em seu propósito, porém o nome da tarefa deve refletir mais especificamente o que será realizado, sendo necessário definir um nome diferente da atividade do processo associada. Esta operação cria uma tarefa no plano de projeto com o nome da tarefa diferente do nome da atividade de processo e registra a atividade do processo definido.

Participantes

Activity: Atividade do processo de software definido para o projeto

Task: Tarefa do projeto presente no plano de projeto

Relacionamento

Um para Um entre *Activity* e *Task*.

Pseudocódigo

```
function task.createLink (iteration: Iteration, instance:
Instance, project: Project, processActivity: Activity,
task_name: String)
begin
      self.iteration = iteration
      self.instance = instance
      self.project = project
      self.processActivity = processActivity
      self.name = task_name
      self.defineTask(task_description, task_status,
      task_startDateForeseen,
      task_startDate, task_endtDateForeseen, task_endDate)
      return task
end
```

5.3.1.4 Operação Case

Nome

Case

Operação

task.createCase (instance: Instance, project: Project, process_activities: Array)

Descrição

Estabelece uma ligação entre cada atividade prevista para uma instância do processo e as tarefas do plano de projeto. Cada nome de tarefa pode ser mais detalhado que o nome da atividade, de acordo com a necessidade do contexto do projeto de software. Todas as tarefas serão associadas a uma instância do processo.

Motivação

O processo de software prescreve um conjunto de atividades que devem ser instanciadas no contexto de um projeto de software. A criação de tarefas, com o nome da atividade mais detalhado, para cada atividades previstas em uma determinada instância, e a representação da atividade do processo associada à tarefa do plano de projeto ajuda na redução do distanciamento entre processo e projeto.

Aplicabilidade

Esta operação se aplica à situação na qual se pretende instanciar todas as atividades previstas para uma determinada instância do processo de software Nesta situação as atividades previstas atendem às necessidades em seus propósitos. Esta operação cria uma tarefa no plano de projeto, para cada atividade prevista na instância do processo, e registra a atividade do processo definido.

Participantes

Activity: Atividade do processo de software definido para o projeto

Task: Tarefa do projeto presente no plano de projeto

Relacionamento

Um para Um entre cada *Activity* e *Task*.

Pseudocódigo

```
function create_case (instance: Instance, project: Project,
process_activities: Array)
begin
      for each activity in process_activities
      begin
            self.instance = instance
            self.project = project
            self.process_activity = process_activity
            self.iteration = read_iteration()
            self.name = process_activity.name +
      read_task_name_detail()
            self.defineTask(task_description, task_status,
      task_startDateForeseen,
            task_startDate, task_endtDateForeseen, task_endDate)
            return task
      end
end
```

5.3.1.5 Operação Split

Nome

Split

Operação

task.createSplit (iteration: Iteration, instance: Instance, project: Project, processActivity: Activity, task_name: String)

Descrição

Estabelece uma ligação entre uma atividade do processo e duas ou mais tarefa do plano de projeto, em uma iteração. O nome de cada tarefa deve ser diferente do nome da atividade de processo associada.

Motivação

As atividades previstas no processo de software podem ser definidas em alto nível, sendo necessário na instanciação para o projeto de software criar duas ou mais tarefas no plano de projeto, em uma mesma iteração. A criação de tarefas, como o nome adequado ao contexto do projeto de software, e a representação da atividade do processo associada a cada tarefa do plano de projeto ajuda na redução do distanciamento entre processo e projeto.

Aplicabilidade

Esta operação se aplica à situação na qual se pretende criar duas ou mais tarefas, para uma iteração, oriundas de uma única atividade prevista no processo de software definido. Nesta situação a atividade prevista atende às necessidades em sua descrição e propósito, mas o alto nível da definição da mesma e o contexto do projeto, implicam na criação de tarefas mais específicas. Esta operação cria duas ou mais tarefa no plano de projeto, em uma iteração, para uma única atividade do processo, e registra a atividade do processo definido.

Participantes

Activity: Atividade do processo de software definido para o projeto

Task: Tarefa do projeto presente no plano de projeto

Relacionamento

Um para Muitos entre *Activity* e *Task*.

Pseudocódigo

```
function create_split (instance: Instance, iteration: Iteration,
project: Project, process_activity: Activity, task_name: String)
begin
     while
     begin
          self.instance = instance
          self.iteration = iteration
```

```
            self.project = project
            self.process_activity = process_activity
            self.name = task_name
            self.defineTask(task_description, task_status,
      task_startDateForeseen,
            task_startDate, task_endtDateForeseen, task_endDate)
            return task
      end
end
```

5.3.1.6 Operação Split Iteration

Nome

Split Iteration

Operação

task.createSplit(iteration: Iteration, instance: Instance, project: Project, processActivity: Activity, task_name: String)

Descrição

Estabelece uma ligação entre uma atividade do processo e duas ou mais tarefa do plano de projeto e as distribui em diferentes iterações. O nome de cada tarefa deve ser diferente do nome da atividade de processo associada.

Motivação

As atividades previstas no processo de software podem ser definidas em alto nível, sendo necessário na instanciação para o projeto de software criar duas ou mais tarefas no plano de projeto, em diferentes iterações. A criação de tarefas como o nome adequado ao contexto do projeto de software e a representação da atividade do processo associada a cada tarefa do plano de projeto ajuda na redução do distanciamento entre processo e projeto.

Aplicabilidade

Esta operação se aplica à situação na qual se pretende criar duas ou mais tarefas, para iterações distintas, oriundas de uma única atividade prevista no processo de software definido. Nesta situação, a atividade prevista atende às necessidades em sua descrição e propósito, mas o alto nível da definição da mesma e o contexto do projeto implicam na criação de tarefas mais específicas. Esta operação cria duas ou mais tarefa no plano de projeto, em iterações distintas, para uma única atividade do processo, e registra a atividade do processo definido.

Participantes

Activity: Atividade do processo de software definido para o projeto

Task: Tarefa do projeto presente no plano de projeto

Relacionamento

Um para Muitos entre *Activity* e *Task*.

Pseudocódigo

```
function createSplitIteration (instance: Instance, iteration:
Iteration, project: Project process_activity: Activity,
task_name: String)
begin
      while
      begin
            self.iteration = read_iteration()
            self.instance = instance
            self.project = project
            self.process_activity = process_activity
            self.name = task_name
            self.defineTask(task_description, task_status,
      task_startDateForeseen,
            task_startDate, task_endtDateForeseen, ,
      task_endDate)
            return task
      end
end
```

5.3.1.7 Operação Merge

Nome

Merge

Operação

task.createMerge (iteration: Iteration, instance: Instance, project: Project, process_activities: Array, task_name: String)

Descrição

Estabelece uma ligação entre duas ou mais atividade do processo e uma única tarefa do plano de projeto.

Motivação

As atividades previstas no processo de software podem ser definidas um baixo nível de detalhamento, sendo necessário, em sua instanciação no projeto de software, agrupá-las em uma tarefa no plano de projeto. A criação da tarefa com o nome diferente das atividades associadas, para se adequar ao contexto do projeto, e a representação das atividades do processo associadas à tarefa do plano de projeto ajuda na redução do distanciamento entre processo e projeto.

Aplicabilidade

Esta operação se aplica à situação na qual se pretende instanciar duas ou mais atividades do processo de software em uma única tarefa no plano de projeto. Nesta situação, as atividades previstas associadas atendem às necessidades em seu propósito e descrição, porém para o contexto do projeto de software uma única tarefa deve ser criada. Esta operação cria uma tarefa no plano de projeto com o nome diferente das atividades associadas e registra as atividades do processo definido.

Participantes

Activity: Atividade do processo de software definido para o projeto

Task: Tarefa do projeto presente no plano de projeto

Relacionamento

Muitos para Um entre *Activity* e *Task*.

Pseudocódigo

```
function createMerge(iteration: Iteration, instance: Instance,
project: Project, process_activities: Array, task_name: String)
begin
      self.instance = instance
      self.iteration = iteration
      self.project = project
      self.process_activities = process_activities
      self.name = task_name
      self.defineTask(task_description, task_status,
task_startDateForeseen,
      task_startDate, task_endtDateForeseen, , task_endDate)
      return task
end
```

5.3.1.8 Operação Add

Nome

Add

Operação

task.createAdd (iteration: Iteration, instance: Instance, project: Project, task_name: String)

Descrição

Cria uma tarefa no plano de projeto que não possui uma atividade prevista no processo de software.

Motivação

As atividades previstas no processo de software não atendem às necessidades de uma tarefa do plano de projeto. Uma tarefa é criada e não possui a representação da atividade do processo associada.

Aplicabilidade

Esta operação se aplica à situação na qual nenhuma atividade do processo de software atende à tarefa no plano de projeto. Nesta situação, nenhuma atividade prevista atende às necessidades em sua descrição e propósito, sendo necessário criar uma tarefa no plano de projeto sem nenhuma atividade do processo associada. Esta operação cria uma tarefa no plano de projeto e não é registrada uma atividade do processo definido.

Participantes

Task: Tarefa do projeto presente no plano de projeto

Relacionamento

Não se aplica.

Pseudocódigo

```
function task.createAdd(iteration: Iteration, instance:
Instance,
project: Project,, task_name: String)
begin
      self.iteration = iteration
      self.instance = instance
      self.project = project
      self.process_activity = NULL
      self.name = task_name
      self.defineTask(task_description, task_status,
      task_startDateForeseen, task_startDate,
      task_endtDateForeseen, , task_endDate)
      return task
end
```

5.3.1.9 Operação Remove

Nome

Remove

Operação

task.remove (processActivity)

Descrição

Uma atividade do processo de software não tem uma tarefa correspondente no plano de projeto.

Motivação

As atividades previstas no processo de software podem não ser instanciadas no projeto de software. Nenhuma tarefa é associada às atividades previstas.

Aplicabilidade

Esta operação se aplica à situação na qual não se pretende criar tarefas para uma determinada atividade prevista no processo de software. Nesta situação esta atividade prevista não atende às necessidades em sua descrição e propósito, não sendo associada a tarefas no plano de projeto.

Participantes

Activity: Atividade do processo de software definido para o projeto

Relacionamento

Não se aplica.

Pseudocódigo

Não se aplica.

5.3.1.10 Operação Link Artifact

Nome

Link Artifact

Operação

task.createLinkArtifact (artifact: Artifact, project: Project, processActivity: Activity)

Descrição

Estabelece uma ligação entre um artefato associado a uma atividade de processo e um artefato produzido por uma tarefa do plano do projeto. O artefato produzido por uma tarefa deve ter o mesmo tipo que a atividade de processo associada ao artefato.

Participantes

Artifact: Artefato associado a uma atividade do processo de software definido para o projeto

TaskArtifact: Artefato associado a uma tarefa do projeto presente no plano de projeto

Relacionamento

Um para Um entre *Artifact* e *TaskArtifact*.

Pseudocódigo

Não se aplica.

5.3.1.11 Operação Link Role

Nome

Link Role

Operação

task.createLinkRole (role: Role, project: Project, processActivity: Activity)

Descrição

Estabelece uma ligação entre o papel associado a uma atividade de processo e o ator responsável por uma tarefa do plano do projeto.

Participantes

Role: Papel responsável por atividade do processo de software definido para o projeto.

TaskActor: Ator responsável por uma tarefa do projeto presente no plano de projeto

Relacionamento

Um para Um entre *Role* e *TaskActor*.

Pseudocódigo

Não se aplica.

A próxima seção apresenta os estudos conduzidos para apoiar a definição e estruturação das operações de instanciação propostas.

5.4 Estudos Conduzidos

Após a definição das operações descritas, estudos foram conduzidos para apoiar a definição e estruturação, como uma forma de obter a visão prática sobre a ocorrência das operações e assim verificar o processo de criação de tarefas por meio das mesmas.

A partir do conjunto de operações, foi realizado um estudo de observação. Este estudo ocorreu em uma reunião de planejamento de uma nova versão, em uma organização de desenvolvimento de software de médio porte. Para apoiar as operações propostas, também foram realizadas duas entrevistas com gerentes de projeto. As entrevistas foram realizadas com gerentes de projeto e o objetivo foi verificar o processo de criação de tarefas na perspectiva de especialistas e, assim, mapear as operações utilizadas. A fim de identificar trabalhos sobre operações de instanciação, uma busca estruturada da literatura foi realizada.

5.4.1 Estudo de Observação

A técnica de estudo de observação foi escolhida para entender como o planejamento de um novo projeto é feito. O estudo foi realizado em uma reunião de planejamento de uma nova versão de software. A organização possui um processo de software definido, o qual é utilizado com guia ao longo dos projetos.

5.4.1.1 Planejamento do Estudo

Seguindo a abordagem GQM (*Goal, Question, Metric*) (BASILI; CALDIERA e ROMBACH, 1994) o objetivo global do estudo de observação foi:

Analisar as práticas de planejamento de tarefas

Com o propósito de caracterizar

Com respeito ao cadastro das tarefas a serem executadas no projeto

Do ponto de vista de engenheiros de software

No contexto de gerentes, analistas e desenvolvedores participando de uma reunião de planejamento de tarefas para desenvolvimento de uma funcionalidade de faturamento em uma organização de desenvolvimento de software de médio porte.

5.4.1.2 Execução do Estudo

A execução deste estudo de observação se deu de forma presencial, em uma reunião de planejamento de uma *sprint*. Estavam presentes o gerente de projeto, um analista, três desenvolvedores e a pesquisadora desta tese. Ao analista foi atribuída a função de líder da reunião. O objetivo geral da reunião era planejar as atividades de desenvolvimento de uma funcionalidade de faturamento.

O início da reunião se deu com a apresentação dos protótipos de tela da funcionalidade a ser planejada, onde foi descrito todo o módulo de emissão de fatura. A equipe avaliou e discutiu os requisitos apresentados. O analista anotou as dúvidas e sugestões apontadas pela equipe, para ser validado com o cliente. Também foram apresentados e discutidos com a equipe o diagrama de estados, o modelo de domínio, o diagrama de casos de usos e suas descrições. Em seguida foi definido o caso de uso que seria tratado na *sprint*. O gerente de projeto definiu a equipe e iniciou a elaboração da lista de tarefas a serem realizadas no desenvolvimento do caso de uso escolhido. O gerente foi anotando as tarefas em um bloco de notas.

Após realizar todo o levantamento das tarefas, as mesmas foram delegadas aos membros da equipe. Para definir o prazo de cada tarefa, foram utilizados cartões. O gerente fazia a leitura da tarefa e cada membro colocava na mesa o tempo estimado, e o gerente definia o prazo pela média dos prazos estimados por cada

membro. No momento de definir os prazos das tarefas, outras tarefas foram definidas, algumas já definidas foram mais detalhadas, e algumas definidas foram retiradas para serem analisadas. Ao final a lista de tarefas foi entregue ao analista para cadastramento das mesmas em um sistema de gerenciamento de tarefas.

Cabe ressaltar que, durante a reunião a pesquisadora fez anotações das observações sobre a reunião, servindo de base para o relato do estudo e discussão dos resultados descritos a seguir.

5.4.1.3 Discussão dos Resultados

Neste estudo foi possível identificar a ocorrência das operações propostas. A reunião observada era da etapa de desenvolvimento, considerando que os requisitos já estavam definidos.

Observou-se que a criação das tarefas é feita baseada em um processo, porém o processo não é explicitado durante o planejamento. Ao final da reunião a lista de tarefas definidas na reunião foi analisada e foi possível concluir que não existe o mapeamento explícito entre as atividades definidas no processo as tarefas definidas no projeto. Por exemplo, as tarefas "*Incluir o campo situação Fatura*" e "*Alterar o modelo de entidade para incluir associação Parte Envolvida e Empresa*" foram criadas, porém apenas pelo nome da tarefa, não é possível identificar a atividade do processo.

Com o apoio do gerente, o mapeamento foi realizado, e as seguintes relações previstas pelas operações foram observadas: *Uma atividade do processo é criada como uma única tarefa*; *Uma atividade do processo pode se tornar várias tarefas*, *Várias atividades do processo são agrupadas um uma única tarefa*; *Uma atividade do processo é criada com um nome diferente*; *Uma atividade do processo não é criada*; *Uma tarefa criada não tem nenhuma atividade correspondente no processo*; *Uma atividade do processo é criada com mais detalhe*; e *Uma tarefa que representa parcialmente uma atividade correspondente no processo*. As seguintes relações não foram observadas: *Várias atividades relacionadas do processo são distribuídas por meio de várias iterações*; *e Um conjunto de atividades do processo são referenciadas a uma instância do processo no plano de projeto*. A seguinte relação "*Várias atividades relacionadas do processo são distribuídas por meio de várias iterações*" não foi observada porque o planejamento era para uma única iteração, e "*Um conjunto de atividades do processo são referenciadas a uma instância do processo no plano de projeto*", não foi observada porque algumas atividades previstas para a fase de desenvolvimento já haviam sido realizadas.

A seguir um segundo estudo conduzido, no formato de entrevista, será apresentado.

5.4.2 Análise com Especialistas

Entrevistas foram realizadas com gerentes de projeto com o objetivo de compreender o processo de criação de tarefas na perspectiva de especialistas e, assim, mapear as relações entre as tarefas do projeto e as atividades do processo.

5.4.2.1 Planejamento do Estudo

Seguindo a abordagem GQM (*Goal, Question, Metric*) (BASILI; CALDIERA e ROMBACH, 1994) o objetivo global desta entrevista foi:

Analisar as tarefas criadas no projeto

Com o propósito de compreender

Com respeito à relação de adequação entre as tarefas executadas no projeto e as atividades do processo de software definido para o projeto

Do ponto de vista de engenheiros de software

No contexto de projetos de desenvolvimento de software em organizações de médio porte.

O público alvo foram gerentes de projeto (ou algum papel responsável pela criação do plano de projeto) que tenham experiência em criação do plano de projeto em desenvolvimento de software.

Para a caracterização do indivíduo foi perguntado o nível de formação e país, e tempo de experiência em gerenciamento de projetos de desenvolvimento de software. E para caracterização da organização as seguintes informações foram obtidas: quantidade de projetos desenvolvidos nos últimos 5 anos; quantidade de profissionais; tipos de software desenvolvidos; modelo de maturidade, se adotado; e se possui processo de desenvolvimento de software definido. No Quadro 1 é exibida a mensagem de convite para a entrevista.

Quadro 1 - Mensagem de Convite para a Entrevista

"Prezado Respondente, *Estamos identificando potenciais participantes para uma pesquisa de opinião, envolvendo a forma de criação de tarefas para a equipe de desenvolvimento de software. Caso tenha interesse em participar, necessitamos que você responda ao formulário de caracterização em anexo.* *Contamos com uma possível colaboração sua na pesquisa de opinião, onde haverá a oportunidade de agregar conhecimento referente aos temas envolvidos na nossa pesquisa de doutorado, por meio do registro de sua experiência na área de desenvolvimento de software.* *A partir da confirmação de sua participação e envio do formulário de caracterização preenchido, enviaremos o questionário de nossa pesquisa.* *Desde já agradecemos a sua colaboração.* *Renata Mesquita (COPPE/UFRJ- Prisma Group)* *Toacy Oliveira (COPPE/UFRJ- Prisma Group)"*

O tempo estimado para execução da entrevista foi de 20 minutos, e a mensagem (Quadro 1) foi enviada por e-mail para convidar para a participação na entrevista.

5.4.2.2 Execução do Estudo

A entrevista foi realizada de forma não supervisionada, não havendo auxílio pessoal aos participantes, durante o período de aplicação do questionário. As entrevistas foram realizadas com dois gerentes de projeto. Ambos gerentes eram de organizações de desenvolvimento de software de médio porte. Porém, apesar do mesmo porte da organização, as mesmas se diferem, porque uma tem nível A no MPS.BR e a outra não adota modelos de maturidade. O questionário aplicado está presente do Apêndice C.

Os questionários foram respondidos e retornados por e-mail. A partir do retorno as respostas foram analisadas.

5.4.2.3 Discussão dos Resultados

A realização das entrevistas teve como foco principal compreender o processo de processo de tarefas e, assim, mapear as relações entre as tarefas do projeto e as atividades do processo. Cada relação apresentada se referia a uma das operações de instanciação proposta. Nas entrevistas, observou-se que as operações *Detail* (Uma atividade do processo é criada com mais detalhes) e *Split Iteration* (Várias atividades relacionadas do processo são distribuídas por meio de várias iterações) não foram muito bem compreendidas. Isso também foi observado para a operação *Case* (Várias atividades relacionadas são detalhadas e referenciadas a uma instância do processo). Sendo assim, esses resultados direcionaram o refinamento da descrição dessas operações, para permitir uma melhor compreensão para a sua aplicabilidade.

A seguir a revisão da literatura realizada, por meio de uma busca estruturada, será apresentada.

5.4.3 Revisão da Literatura

A fim de identificar os trabalhos, que relatem sobre operações de instanciação aplicadas à criação de tarefas, uma busca estruturada da literatura foi realizada, o qual consiste em uma estratégia de revisão da literatura e será descrita na seção a seguir.

5.4.3.1 Planejamento do Estudo

Seguindo a abordagem GQM (*Goal, Question, Metric*) (BASILI; CALDIERA e ROMBACH, 1994) o objetivo global desta busca estruturada foi:

Analisar ações, operações, padrões, políticas ou regras de instanciação de processo

Com o propósito de caracterizar

Com respeito à criação do plano contendo as tarefas do projeto

Do ponto de vista de engenheiros de software

No contexto de projetos de desenvolvimento de software utilizados como referência para apresentar ações, operações, padrões, políticas ou regras de instanciação de processo.

5.4.3.1.1 Questão de Pesquisa

Quais abordagens (ações, operações, padrões, políticas ou regras) de instanciação de processo são utilizadas para a criação do plano de projeto (criação de tarefas)?

5.4.3.1.2 Estratégia de Busca e Artigos de Controle

Nesta busca estruturada, a abordagem PICO – *Population of interest, evaluated Intervention, Intervention Comparison e expected Outcome* (PAI *et al.*, 2004), foi utilizada para organizar e estruturar a busca a ser realizada. A estruturação adotada está descrita a seguir:

(P) População (*Population*): Processo de Software**.**

(I) Intervenção (*Intervention*): Pesquisas relacionadas à instanciação de processo de software**.**

(C) Comparação (*Comparison*): Não há, pois não se tem como objetivo comparar abordagens, e sim, caracterizá-las.

(O) Resultados (*Output*): Ações, Operações, Regras, Padrões ou Políticas de apoio a criação do plano de projeto.

Baseado na estrutura PICO, os seguintes termos e seus sinônimos foram definidos:

(P) População: *software process, software project, process of software, software development project, software development process, project construction, software development.*

(I) Intervenção: *instantiation, creating task, creating activity, creation of task, creation of activity, creating project plan, creation of project plan, creating plan, creation of plan, deployment, creating schedule, creation of schedule, creating WBS, "creation of WBS.*

(C) Comparação: não aplicável.

(O) Resultados: *action, operating, pattern, policy, rule.*

Base de Busca: Scopus (www.scopus.com)

String de Busca: (TITLE-ABS-KEY (("software process*" OR "software project*" OR "process of software" OR "software development project*" OR "software development process*" OR "reference model" OR "standard process*" OR "defined process*" OR "process description*" OR "description of process*" OR "modelled process" OR "modelled software process*")) AND TITLE-ABS-KEY (("instantiation" OR "creating task" OR "creating activit*" OR "creation of task*" OR "creation of activit*" OR "creating project plan" OR "creation of project plan" OR "creating plan" OR "creation of plan" OR "deployment" OR "creating schedule" OR "creation of schedule" OR "creating WBS" OR "creation of WBS")) AND TITLE-ABS-KEY (("action" OR "operation" OR "pattern" OR "polic*" OR "rule")))

Não foram identificados artigos de controle, antes da execução da busca.

5.4.3.1.3 Critérios de Inclusão e Exclusão dos Artigos

A seguir são especificados os Critérios de Inclusão (CI) e Critérios Exclusão (CE) que foram adotados para apoiar a seleção dos artigos.

Critérios de Inclusão dos Artigos:

(CI1) Trabalhos que abordam sobre instanciação de processo de software, apresentando como esta deve ser realizada; OU

(CI2) Trabalhos que abordam a forma como é realizada a criação do plano de projeto; OU

(CI3) Trabalhos que apresentam ferramental de apoio a criação do plano de projeto.

Critérios de Exclusão de Artigos:

(CE1) Trabalhos fora da área de computação; OU

(CE2) Trabalhos cujo foco de pesquisa encontra-se em outras áreas da computação que não seja a de Engenharia de Software; OU

(CE3) Propostas que não sejam aplicadas a processos de software, especificamente; OU

(CE4) Trabalhos que não sejam destinados a instanciação do processo de software; OU

(CE5) Trabalhos que não apresentam ações, operações, padrões, políticas ou regras utilizadas para a criação do plano de projeto, ou seja, criação de tarefas;

(CE6) Trabalhos indisponíveis; OU

(CE7) Trabalhos não escritos em inglês.

5.4.3.1.4 Procedimentos para seleção dos estudos

A seleção dos estudos foi realizada em três etapas:

- *Etapa 1*

Seleção preliminar das publicações: A seleção preliminar das publicações foi realizada por meio da execução da *string* de busca na base pesquisa *Scopus*, selecionada para este estudo.

- *Etapa 2*

Seleção das publicações relevantes - 1º filtro: Para uma seleção inicial das publicações relevantes, os resumos de cada artigo retornado foram lidos e avaliados de acordo com os critérios de inclusão e exclusão definidos no planejamento da busca. Em alguns artigos, apenas a leitura dos resumos, pode não ser suficiente, gerando dúvidas se deveriam ou não ser incluídos. Neste caso os artigos devem selecionados para leitura completa.

- *Etapa 3*

Seleção das publicações relevantes - 2º filtro: A seleção final das publicações relevantes ocorreu por meio da leitura completa dos textos das publicações selecionadas no 1º filtro. Estas publicações também foram avaliadas de acordo com os critérios de inclusão e exclusão previamente definidos.

Por questão de tempo e disponibilidade de pesquisadores para participar deste estudo, a avaliação da inclusão ou não dos artigos retornados pela busca estruturada foi realizada apenas pela autora deste texto, o que representa uma ameaça a validade.

5.4.3.1.5 Campos de Extração e Avaliação da Qualidade dos Artigos

Os campos de extração de dados dos trabalhos selecionados foram definidos visando capturar dados que auxiliem na resposta à questão de pesquisa estabelecida para esta busca estruturada. Estes são listados a seguir:

(C1) Descrição de como é realizada a instanciação do processo de software;

(C2) Descrição das ações, operações, padrões, políticas ou regras utilizadas para a criação do plano de projeto, ou seja, criação de tarefas; e

(C3) Descrição de como são aplicadas as ações, operações, padrões, políticas ou regras utilizadas para a criação do plano de projeto, ou seja, criação de tarefas.

5.4.3.2 Execução do Estudo

De acordo com os procedimentos de seleção dos artigos definidos no planejamento desta busca estruturada, a primeira etapa consistiu na execução da

string de busca na fonte selecionada. Nesta etapa foram retornados 199 estudos distintos.

Na etapa seguinte, de seleção dos artigos relevantes, os resumos de cada um dos 199 artigos retornados na etapa anterior foram lidos. Nesta etapa, considerado os critérios de inclusão e exclusão, 14 artigos foram selecionados. A lista de artigos resultantes após aplicação dos 1º e 2º filtros é apresentada na Tabela 10.

Tabela 10 – Lista de Artigos Selecionados

Ano	**Referência**
2015	VO, T. T.; COULETTE, B.; TRAN, H. N.; LBATH, R. "Defining and using collaboration patterns for software process development". In: **2015 3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)**. IEEE, 2015. p. 557-564.
2015	VO, T. T.; COULETTE, B.; TRAN, H. N.; LBATH, R. **"**An approach to define and apply collaboration process patterns for software development". In: **International Conference on Model-Driven Engineering and Software Development**. Springer, Cham, 2015. p. 248-262.
2015	KHALIL, R., STOCKTON, D., ALKAABI, M. S., & MANYONGE, L. M. "Investigating the effect of variability in product development life cycle". **International Journal of Product Development**, v. 20, n. 6, p. 495-513, 2015.
2014	KHAIYUM, S.; KUMARASWAMY, Y. S. "Pattern of transformation of failures based on severity in real-time embedded software projects". In: **2014 International Conference on Electronics and Communication Systems (ICECS)**. IEEE, 2014. p. 1-5.
2012	GAFFO, F. H.; BARROS, R. M. GAIA "Risks: A risk management framework". In: **Proceedings of the 25th International Conference on Computer Applications in Industry and Engineering**. 2012. p. 57-62.
2010	LAMERSDORF, A.; MUNCH, J.; TORRE, A. F. D.; SANCHEZ, C. R.; HEINZ, M.; ROMBACH, D. "A rule-based model for customized risk identification in distributed software development projects". In: **2010 5th IEEE International Conference on Global Software Engineering**. IEEE, 2010. p. 209-218.
2010	KILLISPERGER, P.; STUMPTNER, M.; PETERS, G.; GROSSMANN, G.; STÜCKL, T. "A framework for the flexible instantiation of large scale software process tailoring". In: **International Conference on Software Process**. Springer, Berlin, Heidelberg, 2010. p. 100-111.
2009	LIGGESMEYER, P.; HEIDRICH, J.; MÜNCH, J.; KALCKLÖSCH, R.; BARTHEL, H.; ZECKZER, D. "Visualization of software and systems as support mechanism for integrated software project control". In: **International Conference on Human-Computer Interaction**. Springer, Berlin, Heidelberg, 2009. p. 846-855.
2009	LEE, S. W.; GANDHI, R. A.; WAGLE, S. J. "Ontology-Guided Service-Oriented Architecture Composition to Support Complex and Tailorable Process Definitions". **International Journal of Software Engineering and**

	Knowledge Engineering, v. 19, n. 06, p. 791-821, 2009.
2007	FONTOURA, L. M.; PRICE, R. T. “A Framework for Tailoring Software Process”. In: **SEKE**. 2007. p. 63-66.
2002	REIS, C. A. L.; REIS, R. Q.; SCHLEBBE, H.; NUNES, D. J. “A policy-based resource instantiation mechanism to automate software process management”. In: **Proceedings of the 14th international conference on Software engineering and knowledge engineering**. 2002. p. 795-802.
2002	REIS, C. A. L.; REIS, R. Q.; SCHLEBBE, H.; NUNES, D. J. “Resource instantiation policies for software process environments”. In: **Proceedings 26th Annual International Computer Software and Applications**. IEEE, 2002. p. 53-58.
1996	MCGUIRE, E. G. “Factors affecting the quality of software project management: An empirical study based on the Capability Maturity Model”. **Software Quality Journal**, v. 5, n. 4, p. 305-317, 1996.
1991	MADHAVJI, N. H.; SCHAFER, W. “Prism= methodology+ process-oriented environment”. In: **Proceedings. 12th International Conference on Software Engineering**. IEEE, 1990. p. 277-288.

Na terceira e última etapa, foi realizada a leitura do texto completo dos 14 artigos. A seleção dos 3 estudos relevantes (REIS *et al.*, 2002a, REIS *et al.*, 2002b e KILLISPERGER *et al.*, 2010) também foi realizada de acordo com os critérios de inclusão e exclusão.

A busca foi executada em maio de 2017 e re-executada em novembro de 2018. Na re-execução da busca não houve alteração nos trabalhos selecionados. A partir da leitura completa de todos os artigos apresentados na Tabela 10 foi realizada a análise dos dados extraídos para responder as questões de pesquisas definidas no planejamento deste estudo.

5.4.3.3 Discussão dos Resultados

Baseado nas informações extraídas dos artigos selecionados (REIS *et al.*, 2002a, REIS *et al.*, 2002b e KILLISPERGER *et al.*, 2010), não foi possível identificar trabalhos que abordem ações, operações, padrões, políticas ou regras de instanciação relacionados à criação do plano de projeto, ou seja, ao cadastro de tarefas. Muitos trabalhos eram relacionados a padrões de projeto e instanciação de *framework*. Em Reis *et al.* (2002a) e Reis *et al.* (2002b) destacam-se as políticas de instanciação de recursos, que apoiam a alocação de recurso às tarefas criadas durante a Instanciação. Em Killisperger *et al.* (2010) são apresentados operadores de instanciação, que permitem instanciação antes e depois do projeto, porém eles tratam de operações aplicadas na perspectiva de processo (modelo de processo), e não na transição entre

as perspectivas atuando na criação do plano de projeto (lista de tarefas) estabelecendo vínculo entre elas.

A seção seguinte apresenta uma análise comparativa entre a solução descrita neste Capítulo e os trabalhos relacionados identificados na revisão da literatura apresentada no Capítulo 4.

5.5 Análise Comparativa entre a Solução e os Trabalhos Relacionados

A solução para mapear processo de software e processos de software é realizada através da representação de elementos do processo de software no plano do projeto, durante a instanciação. Essa representação estabelece um vínculo entre as perspectivas. Dessa forma, é possível identificar as informações do processo a partir das tarefas e registros de execução, ao longo projeto de software. Durante a revisão da literatura realizada (Capítulo 4), foram identificados trabalhos relacionados que abordam o problema da lacuna entre as perspectivas do processo e do projeto. Estes trabalhos foram agrupados em três categorias de solução para o problema: *Correspondência*, *Alinhamento* e *Mineração*.

As estratégias de Correspondência (*Matching*) propostas por Baier *et al.* (2014), Baier *et al.* (2015a), Baier *et al.* (2015b) e Baier *et al.* (2018) exigem pouco esforço manual para corresponder os registros de execução com as atividades predefinidas no modelo de processo. A solução definida nesta tese também requer esforço manual, mas o mapeamento entre as tarefas e atividades do processo é feito na criação dos registros, devido à complexidade de formalizar heurísticas, para identificar o relacionamento entre o processo de software definido (atividade) e o plano do projeto (tarefa), após o plano de o projeto ter sido criado.

As estratégias de Alinhamento apresentaram soluções de alinhamento entre o registro de eventos e o modelo de processo. Os seguintes trabalhos foram identificados: Leoni, Maggi e Van der Aalst (2012), Adriansyah *et al.* (2012), Adriansyah e Buijs (2012), Leoni e Van Der Aalst (2013), Leoni, Maggi e Van der Aalst (2015), Mannardt *et al.* (2016), Leoni e Marrella (2017), Van Dongen *et al.* (2017). Eles apontaram alguns benefícios desse alinhamento para apoiar a verificação de conformidade e utilizam métricas que avaliam a qualidade de um modelo de processo para representar o comportamento observado (aptidão e precisão). No entanto, as técnicas apresentadas não consideram o problema da falta de mapeamento entre o processo e o projeto (a atividade do processo de referência não é registrada durante a execução do processo), o que dificulta a identificação da atividade para cada registro

da base de dados de execução. Nesse sentido, sem esse vínculo, a aplicação de técnicas de alinhamento pode enfrentar desafios. De acordo com Baier *et al.* (2018), a relação entre eventos e atividades não pode ser facilmente identificada usando uma correspondência simples de cadeias, pois os termos usados nos registros raramente ocorrem nos nomes das atividades do modelo de processo. A solução apresentada nesta tese pretende estabelecer o vínculo entre elementos do modelo de processo e tarefa do plano do projeto através de operações de instanciação. Neste sentido, o mapeamento entre as perspectivas pode ser apresentado como um pré-requisito para a aplicação das abordagens de alinhamento, uma vez torna explícita a ligação entre as perspectivas, por meio do mapeamento entre as representações do processo e projeto de software.

Não foi possível identificar na literatura nenhuma abordagem automática, e a solução apresentada não é uma exceção neste aspecto, pois para estabelecer o mapeamento requer um esforço manual. Apesar do esforço manual necessário, o aumento desse esforço ocorre na criação do plano do projeto (instanciação), uma etapa já prevista e realizada ao longo do projeto de software. Considerando que o mapeamento é realizado por profissionais que devem conhecer o processo, não é necessário alocar recursos adicionais para realizar este mapeamento.

Diferentemente das estratégias apontadas nos trabalhos relacionados, a principal vantagem desta solução é que o mapeamento é estabelecido durante a instanciação (criação do plano do projeto) do processo. Assim, os benefícios do mapeamento podem ser obtidos logo após a instanciação do processo de software. Enquanto, nas estratégias identificadas, o mapeamento é realizado após a execução. Portanto, os benefícios do mapeamento podem ser usados apenas em novos projetos ou em iterações subsequentes do mesmo projeto. Uma vez estabelecido o mapeamento, é possível extrair informações do processo, para que ações como re-planejamento e treinamento da equipe possam ser executadas para ajustar o processo do software de referência ao projeto do software, ou vice-versa, se necessário.

As estratégias identificadas na literatura, o mapeamento ocorre após a execução do processo, sendo assim, perde-se o conhecimento dos profissionais envolvidos na criação do plano do projeto. Assim, o esforço de mapeamento é mais significativo após a execução do processo, conforme discutido por Baier *et al.* (2018). De acordo com Baier *et al.* (2018) independentemente das técnicas empregadas, a pior complexidade do problema de calcular alinhamentos ideais é exponencial em relação à quantidade de comportamento permitido pelos modelos de processo e o tamanho dos registros de execução.

Para finalizar, todas as pesquisas identificadas são da área de processos de negócio. Em nossa revisão da literatura, não identificamos trabalhos que abordem estratégias de mapeamento aplicadas a processos de software. Isto reforça a necessidade de investigar esse problema nos processos de software e na dedicação de esforços na proposição de soluções.

5.6 Considerações Finais

Este Capítulo apresentou a visão geral da solução para mapear as perspectivas de processo e projeto. O processo de aplicação da solução, que compreende as etapas de definição, instanciação e execução, foi descrito. Também foram apresentadas em detalhes as operações de instanciação propostas que implementam mecanismos que permitem que elementos do processo de software definido estejam explícitos durante a instanciação e execução. A partir do estudo de observação e das entrevistas realizadas foi possível identificar a ocorrência das relações propostas por cada operação de instanciação. De maneira geral, com base no estudo de observação e nas entrevistas realizadas, foi possível confirmar que o comportamento proposto por cada operação é observado durante a criação das tarefas. A busca estruturada permitiu-nos identificar que não existem abordagens e tecnologias disponíveis na literatura que possa apoiar aos engenheiros de software a mapear o processo definido para as tarefas do projeto. Uma análise comparativa permitiu identificar o diferencial da solução em relação aos trabalhos relacionados.

Considere-se que o resultado do mapeamento estabelecido, por meio das operações de instanciação, além de permitir o mapeamento entre as perspectivas de processo e projeto, também permite realizar mais facilmente análises dos processos de software efetivamente executados nos projetos de software. Neste sentido, o Capítulo seguinte apresenta orientações para apoiar a demonstração do benefício do mapeamento na identificação do processo de software executado, bem como na verificação de sua conformidade com o processo de software definido.

6 Identificação do Processo de Software Executado e de não Conformidades

Este Capítulo apresenta orientações para demonstrar o benefício do mapeamento entre as perspectivas de processo e projeto na identificação do processo de software executado e na identificação de não conformidades.

6.1 Introdução

Considerando os objetivos específicos apresentados no Capítulo 1 na Seção 1.3 – **Viabilizar a identificação do processo de software executado, por meio da estruturação dos registros de execução** e **Viabilizar a identificação de não conformidades, a partir do mapeamento entre as perspectivas**, este capítulo apresenta orientações que apoiam a demonstração do benefício do mapeamento, entre representações do processo e projeto de software, na identificação do processo de software executado e na identificação de não conformidades. Conforme destacado por Baier *et al.* (2018), o mapeamento dos eventos produzidos (tarefas e registros) e as atividades de um determinado modelo de processo é essencial para verificação de conformidade e entendimento dos resultados da mineração de processos.

De acordo com Van der Aalst *et al.* (2011) as técnicas de Mineração de Processo podem ajudar as organizações a ter um conhecimento mais profundo de seus processos, a partir das tarefas registradas em ferramentas de gerenciamento de projetos, em termos de descoberta e aprimoramento de modelos de processo ou verificação da conformidade da execução com a definição. Porém, conforme observado por Santos, de Oliveira e Brito e Abreu (2015), é difícil realizar a análise do processo de software executado a partir dos registros de execução, embora seja registrada uma grande quantidade de dados ao longo do projeto de desenvolvimento de software, uma vez que informações do processo de software de referência não estão explícitas nos registros de execução. O que torna a análise do processo que efetivamente é seguido ao longo dos projetos uma atividade não trivial, demorada e propensa a erros. De acordo com Baier *et al.* (2018) os modelos descobertos são mais compreensíveis, quando a terminologia, representada nos registros de execução, é de conhecimentos dos profissionais envolvidos no processo.

A Figura 8 apresenta a visão geral da identificação do processo de software executado e de não conformidades considerando a obtenção de registros de execução mapeados com as atividades do processo de referência.

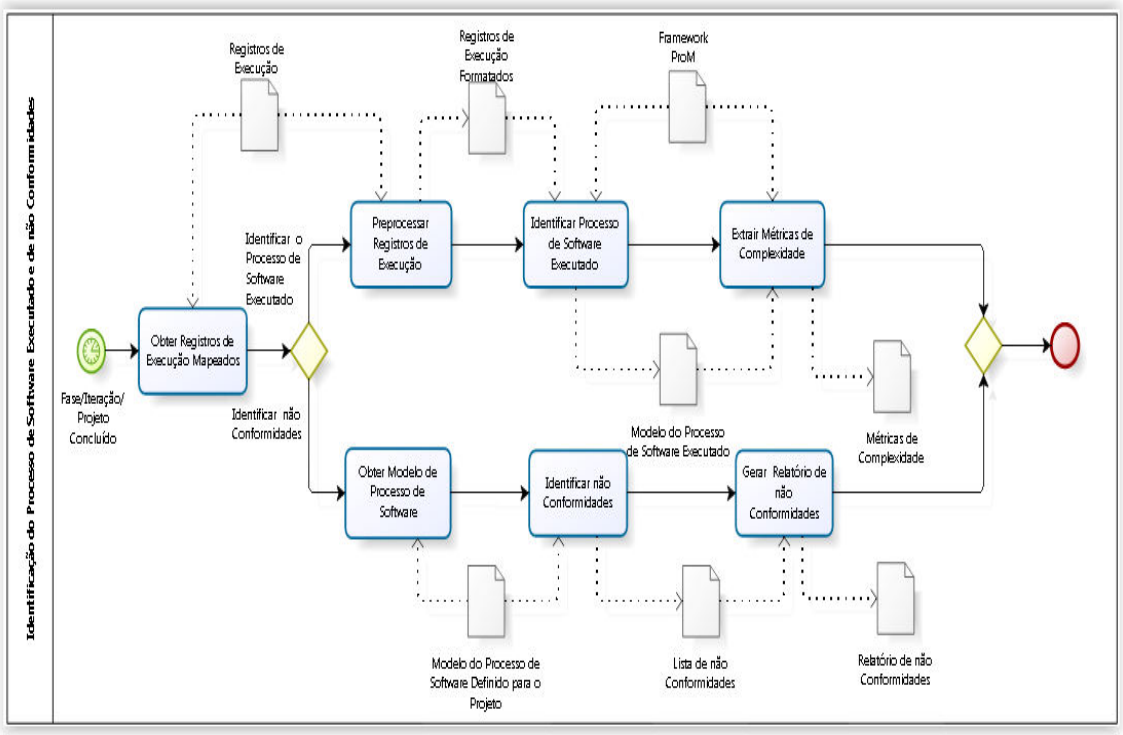


Figura 8 - Visão geral da Identificação do Processo de Software Executado e de não Conformidades. Fonte: A Autora.

A atividade *Obter Registros de Execução Mapeados* envolve a obtenção dos registros de execução, gerados a partir da solução de mapeamento apresentada no Capítulo 5, assim que a execução da iteração ou do projeto é concluída.

A Seção 6.2 apresenta as orientações para a identificação do processo de software executado realizada a partir de bases de dados de execução de projetos mapeados. E a Seção 6.3 apresenta as orientações para a identificação de não conformidade, realizada a partir de bases de dados de execução de projetos mapeados. Por último as considerações finais são apresentadas na Seção 6.4.

6.2 Identificando o Processo de Software Executado

Identificar o processo de software executado é importante, uma vez que desvios podem ocorrer ao longo dos projetos de software. As atividades do processo podem ser divididas, combinadas, removidas e adicionadas na forma de tarefas no plano de projeto; mais de uma tarefa pode ser criada para cada atividade do processo; tarefas podem ser incluídas sem ter uma atividade de processo prevista; e uma tarefa pode estar relacionada a mais de uma atividade do processo.

Uma vez criadas a tarefas, pela aplicação da solução de mapeamento entre as perspectivas, as mesmas podem ser executadas gerando os registros de execução. A representação do mapeamento tem implicações diretas na identificação do processo de software executado, realizada com apoio de técnicas de Mineração de Processos (VAN DER AALST, 2011). A identificação do processo de software executado é beneficiada devido à estruturação dos registros de execução imposta pela aplicação das operações de instanciação.

A área de Mineração de Processo envolve a descoberta do modelo de processo de software sob a ótica de sua execução, para isto, os registros de execução gerados ao longo do projeto de software são utilizados como entrada para as técnicas de mineração de processo. Neste sentido, os registros gerados ao longo da execução dos projetos se apresentam como uma fonte de dados sobre como os processos são realizados nas organizações.

Porém, conforme discutido na Seção 1.2 do Capítulo 1, que descreve o problema tratado nesta pesquisa, e no Capítulo 3 que apresenta estudos exploratórios, existe evidências da falta de mapeamento explícito entre as perspectivas de processo e projeto. Desta forma, a extração de informações sobre os processos a partir dos registros de execução gerados ao longo dos projetos de desenvolvimento de software não é uma atividade trivial.

A partir da aplicação das operações propostas, a extração de informações do processo pode ter sua eficácia atingida positivamente, uma vez que o conhecimento prévio sobre o processo está presente nos registros de execução. Além disso, ao representar os elementos do processo de software no plano de projeto, é permitido o uso de termos conhecidos pelos profissionais envolvidos, facilitando assim a compreensão do processo descoberto. O mapeamento reduz a distância entre o processo de software e o projeto de software e permite uma melhor compreensão da execução dos processos, por meio da representação de elementos do processo nos registros de execução. Assim, a análise da execução dos processos é beneficiada, uma vez que o modelo de processo é descoberto a partir dos registros de execução gerados por meio da solução de mapeamento.

Considerando que o ponto de partida para qualquer atividade de Mineração de Processos são os registros de execução, a qualidade do seu resultado depende diretamente da qualidade dos registros (VAN DER AALST *et al.*, 2011). Neste sentido, o mapeamento aplicado na criação das tarefas do projeto de software torna os registros de execução mais claros e estruturados, impactando diretamente na qualidade dos mesmos. No contexto dessa pesquisa, a estruturação dos registros de

execução se dá pela aplicação das operações de instanciação, que além de estabelecer a ligação entre as perspectivas, também registra a instância do processo.

Como forma de evidenciar os benefícios do mapeamento realizado, métricas podem ser extraídas a patrir dos registros e dos processos descobertos. Neste sentido, métricas de complexidade de modelos de processo são aplicadas para demonstrar quantitativamente o benefício do mapeamento entre as perspectivas na identificação do processo executado.

As seguintes atividades envolvem a identificação do processo de software executado: *Preprocessar Registros de Execução do Processo*, *Identificar Processo de Software Executado*, e *Extrair Métricas de Complexidade*.

A Tabela 11 apresenta a descrição das atividades que englobam a solução de mapeamento para identificação do processo de software executado. Nesta tabela, as atividades são listadas a seguir em negrito, e descrição é apresentada em itálico.

Tabela 11 – Atividades de Identificação do Processo de Software Executado

Identificação do Processo de Software Executado
Preprocessar Registros de Execução do Processo *Envolve o pré-processamento dos registros de execução, realizada para remover duplicatas de registros, inconsistências geradas na gravação, conversão para o formato XES, entre outras ações de limpeza e transformação necessárias nos registros, para que seja possível identificar o processo de software executado, por meio da ferramenta de mineração de processo ProM.* **Identificar Processo de Software Executado** *Envolve a extração do processo de software executado a partir dos registros de execução. Esta extração pode ser realizada pela ferramenta de mineração de processo ProM ou pela ferramenta Disco.* **Extrar Métricas de Complexidade** *Envolve a obtenção de métricas do processo de software executado, que são extraídas dos registros de execução e dos modelos de processos descobertos.*

As atividades de Identificar Processo de Software Executado (Seção 6.2.1) e Extrair Métricas de Complexidade (Seção 6.2.2) serão descritas em detalhes nas seções seguintes.

6.2.1 Identificar Processo de Software Executado

A identificação do processo de software executado será realizada por meio da aplicação de técnicas de Mineração de Processo. O *framework* ProM (VAN DER

AALST, 2011) é a ferramenta de mineração de processo amplamente utilizada para extração do modelo de processo de software. O *framework* de mineração de processo ProM, possui um conjunto de técnicas (algoritmos) que permitem a descoberta de modelos a partir de registros de execução. A ferramenta comercial Disco é frequentemente utilizada por pesquisadores para realizar a análise inicial dos logs através da filtragem, exploração e análise de gargalos (FLUXICON, 2019). Nesta atividade as duas ferramentas podem ser utilizadas com o propósito de identificação do processo de software executado, por meio da aplicação de algoritmos de mineração de processos.

No decorrer do desenvolvimento desta tese um conjunto de algoritmos de mineração de processo foi analisado, utilizado e testado para extrair os modelos de processo de software executado.

Na ferramenta ProM, o algoritmo *Heuristics Mining* (mineração heurística) (VAN DER AALST, 2011) foi selecionado para a descoberta do modelo de processo, a partir de registros de execução mapeados. Ele foi selecionado por ser um algoritmo de mineração prático e aplicável que pode lidar com ruídos e pode ser usado para expressar o comportamento principal excluindo detalhes e exceções, registrados em eventos. Além destes fatores, a ferramenta ProM possui o *plugin Heuristics Miner*, e também fornece suporte para conversão do modelo descoberto para o formato de Rede de Petri e BPMN. O formato Rede de Petri deve ser utilizado como entrada para a extração de métricas de complexidade de modelos de processo. E o formato BPMN é utilizado no apoio ferramental implementado como suporte a solução de mapeamento, conforme será descrito em detalhes no Capítulo 7.

A ferramenta Disco, por outro lado, adota um algoritmo que pode ser considerado uma versão melhorada do *Fuzzy Miner* disponível no ProM (VAN DER AALST, 2016). Uma característica relevante da ferramenta Disco está no fato que, ao importar o arquivo no formato .xes, o modelo descoberto é exibido com uma interface amigável, que torna mais eficiente a análise do mesmo.

Neste sentido, a ProM e a Disco devem ser utilizadas como a ferramenta responsável pela extração do modelo de processo de software executado, e também das métricas de complexidade que serão apresentadas na seção seguinte.

Portanto, no contexto dessa solução, para identificar o processo de software executado, o primeiro passo é importar o arquivo no formato .xes. Este arquivo será importado na ferramenta ProM ou Disco. Na ferramenta ProM, o plugin *Heuristics Miner* foi selecionado neste estudo para ser utilizado. Na Disco, o modelo é exibido assim que ocorre a importação do arquivo.

Esta seção apresentou a atividade de identificação do processo executado. O resultado desta atividade será utilizado para extração das métricas de complexidade.

A atividade de obtenção das métricas será descrita na seção seguinte.

6.2.2 Obter Métricas do Processo de Software Executado

Na seção anterior vimos como identificar o modelo do processo a partir da aplicação de técnicas de Mineração de processos. Nesta pesquisa é conjecturado que a utilização de operações de instanciação pode aumentar da facilidade de mapeamento de modelos de processo e planos de projeto. A facilidade pode ser medida pela demonstração de seus benefícios. Uma forma de avaliar o benefício do mapeamento na identificação do processo de software executado pode ser por meio de métricas de complexidade processo. Neste sentido esta seção apresenta um conjunto de métricas selecionadas para avaliar quantitativamente os benefícios do mapeamento na melhoria do entendimento do processo efetivamente seguido ao longo dos projetos. No Capítulo 2, na Seção 2.3, diversas métricas com foco em modelos de processo foram apresentadas.

Métricas de processo referem-se a tipos de medida relacionada a um processo, e permitem quantificar os atributos dos processos (CARDOSO, 2008). Como se pretende ao mapear as perspectivas facilitar a realização de análises da execução dos processos, métricas de complexidade de processo podem ser utilizadas para demonstrar quantitativamente o potencial do mapeamento para dar apoio a análise do processo de software executado. De acordo com Cardoso (2008) a complexidade do processo pode ser definida como o grau em que um processo é difícil de analisar, entender ou explicar. Neste sentido o aumento do potencial de entendimento dos processos de software, extraídos a partir de registros de execução, pode ser observado pela diminuição da complexidade de tais processos.

Um modelo simples e extremamente compreensível geralmente é resultante de uma baixa complexidade; enquanto, se a complexidade de um modelo é alta, a compreensibilidade deste modelo é baixa. Neste sentido, se o modelo de processo extraído a partir de registros de execução mapeados for menos complexo que o modelo processo extraído a partir de registros não rmapeados, isso quer dizer que as análises podem ser beneficiadas.

As ferramentas utilizadas para mineração de processo também dão suporte a extração de métricas. Para extração das métricas serão usadas a ferramenta Disco (FLUXICON, 2019) e a ferramenta ProM (VAN DER AALST, 2011).

As seguintes métricas foram selecionadas: *número de atividades*, *frequência máxima de ocorrência da atividade*, *extended cardoso metric, números de arcos* e *número de locais.*

Na ferramenta Disco os registros de execução mapeados, no formato xls, ou .xes, são importados e as seguintes métricas podem ser extraídas: *número de atividades e frequência máxima atividade*. E na ferramenta ProM, os registros de execução mapeados no formato .xes, são importados e submetidos a mineração. O modelo minerado, por meio *do plugin Heuristics Miner*, deve ser convertido para o formato Rede de Petri. Esta conversão se dá por meio do *plugin Convert Heuristics net into Petri net* da ProM. O formato Rede de Petri deve ser utilizado como entrada no *plugin Show Petri-net Metrics* da ferramenta ProM para a extração de métricas de complexidade de modelos de processo. A partir do modelo de processo descoberto, no formato de Rede de Petri, as seguintes métricas podem ser extraídas: *extended cardoso metric, número de arcos* e *número de locais.* As métricas selecionadas estão sumarizadas na Tabela 12.

Tabela 12 - Métricas de Complexidade

Métrica	**Explicação**	**Ferramenta**
Número de atividades (CARDOSO, 2006)	Número de atividades	Disco
Frequência máxima	Frequência máxima de atividades nos registros de execução	Disco
extended cardoso metric (LASSEN e VAN DER AALSt 2009)	Métrica para analisar a complexidade do fluxo de controle dos processos de negócios.	ProM
Número de arcos	Número de arcos (*arcs*) no modelo de processo representado em Rede de Petri	ProM
Número de locais	Número de locais (*places*) no modelo de processo representado em Rede de Petri	ProM

Cada métrica selecionada, bem como o relato da importância da métrica para a análise de complexidade, é descrita a seguir:

- **Número de Atividades** (CARDOSO, 2006)**:** é equivalente a métrica mais simples de complexidade para software *Lines of code*. Por esta razão, o número de atividades é uma medida fácil de compreender para o tamanho de um modelo de processo de negócio. Um baixo valor para esta métrica indica menor complexidade do modelo.
- **Frequência máxima:** o modelo de processo descoberto é composto por um conjunto de atividade, e para cada uma delas é armazenada a quantidade de

mínima e máxima de ocorrência nos registros de execução usados para mineração. Esta métrica extrai o maior valor da lista de frequência máxima das atividades. Um alto valor para esta métrica indica menor complexidade do modelo.

- **Extended Cardoso Metric** (LASSEN e VAN DER AALST, 2009)**:** Métrica para analisar a complexidade do fluxo de controle dos processos de negócios. O modelo minerado deve convertido para o formato Rede de Petri para que essa métrica possa ser extraída. Um baixo valor para essa métrica indica menor complexidade do modelo.
- **Número de arcos:** Ao realizar a conversão do modelo de processo descoberto para o formato rede de Petri, o número de arcos (*arcs*) pode ser obtido. Um baixo valor para esta métrica indica menor complexidade do modelo.
- **Número de locais:** Ao realizar a conversão do modelo de processo descoberto para o formato Rede de Petri, o número de locais (*places*) pode ser obtido. Um baixo valor para esta métrica indica menor complexidade do modelo.

Portanto, no contexto dessa solução, para obter os valores das métricas de complexidade *número de atividades e frequência máxima atividade*, deve ser utilizada a ferramenta Disco, sendo possível obtê-las assim que o modelo é descoberto. E para obter os valores das métricas de complexidade: *extended cardoso metric, número de arcos* e *número de locais*, deve ser utilizada a ferramenta ProM, por meio do *plugin Show Petri-net Metrics*.

A seção seguinte apresenta as orientações para a identificação de não conformidades.

6.3 Identificando não Conformidades

A verificação de conformidade é realizada para identificar as não conformidades entre as perspectivas de processo e projeto. Não conformidades estão relacionadas às atividades previstas executada, atividades previstas não executadas, e atividades não previstas porém executadas.

Conformidade de processo é o grau de concordância entre o processo de desenvolvimento de software que é realmente executado e o processo que é definido para ser realizado (SØRUMGÅRD, 1997). Técnicas de análise de conformidade verificam se o comportamento observado registrado em um registro de execução corresponde a um comportamento modelado, neste sentido, este tipo de análise é crucial, porque muitas vezes as execuções de processos se desviam dos processos

estabelecidos (LEONI; MAGGI e VAN DER AALST, 2012). Neste sentido se faz necessário a implementação de mecanismos que permitam identificar não conformidades que ocorrem ao longo dos projetos de desenvolvimento de software, de forma a possibilitar o direcionamento de ações a partir desta identificação.

A Tabela 13 apresenta a descrição das atividades que englobam a solução de mapeamento para identificação de não conformidades. Nesta tabela, as atividades são listadas a seguir em negrito, e descrição é apresentada em itálico.

Tabela 13 – Atividades de Identificação de não Conformidades

Identificação de não Conformidades
Obter o Modelo de Processo de Software
Envolve a obtenção do modelo de processo de software de referência para o projeto.
Identificar as não Conformidades
Envolve a verificação da execução, após a execução da iteração ou do projeto, para identificação de não conformidades. Na identificação de não Conformidades, é verificada a aderência dos processos executados às descrições de processo, padrões e procedimentos.
Gerar Relatório de não Conformidades
Envolve a geração do Relatório de não Conformidades, o qual apresenta as atividades do processo definido obrigatórias não selecionadas durante a execução do processo.

As seguintes atividades envolvem a identificação do processo de software executado: *Obter o Modelo de Processo de Software*, *Identificar não Conformidades*, e Gera*r Relatório de não Conformidades*. A estruturação dos registros de execução se dá pela aplicação das operações de instanciação, que além de estabelecer a ligação entre as perspectivas, também registra a instância do processo.

As atividades de Obter Modelo de Processo de Software (Seção 6.3.1), Identificar não Conformidades (Seção 6.3.2) e Gerar Relatório de não Conformidades (Seção 6.3.3) serão descritas em detalhes nas seções seguintes.

6.3.1 Obter Modelo de Processo de Software

Para a realização da verificação de conformidade o modelo de processo de referência para o projeto deve ser obtido, bem como as diretrizes de adaptação.

A partir desta obtenção, se faz necessário mapear os possíveis caminhos para que seja possível realizar a comparação entre os registros de execução e o modelo de processo. Para isto, a partir do modelo de processo, é gerada uma matriz de

adjacência e posteriormente é executado um algoritmo de listagem de caminhos (*list_path*). O pseudocódigo da geração da matriz de adjacência e do algoritmo de listagem de caminhos está disponibilizado no Apêndice E.

Portanto, no contexto dessa solução, o modelo de processo de processo deve estar no formato .bpmn. Ferramentas de modelagem na notação BPMN geralmente dão suporte a exportação dos modelos para este formato. Detalhes de como usar o apoio ferramental para a obtenção do modelo no Capítulo 7 e no Apêndice D.

Em seguida a obtenção do modelo de processo de software, a atividade de identificação de não conformidades pode ser realizada.

6.3.2 Identificar não conformidades

A identificação de não conformidade é realizada através da comparação entre as atividades previstas no modelo processo de software definido para o projeto, e as atividades que forem identificadas como executadas. Uma atividade é marcada como executada se estiver relacionada a uma ou mais tarefas, que contenham a data de fim da execução registrada.

Esta análise tem como entrada os valores relativos ao projeto e ao processo de software que serão analisados. Os valores relativos ao projeto é a lista de atividades de processo, associadas às tarefas executadas no projeto, e os valores relacionados ao processo é a lista de atividades previstas, por caminho possível no modelo de processo.

Após a obtenção destes valores de entrada é realizada a comparação sendo identificada as atividades previstas executadas, atividades previstas não executadas e atividades não previstas executadas, para cada caminho possível de execução das atividades.

A representação de elementos do processo nas tarefas dos projetos permite que a verificação de conformidade seja realizada de forma semi-automatizada, uma vez que, as informações do processo estão explícitas nos registros de execução

Cabe ressaltar que considerando uma abordagem construtiva, pela aplicação das operações, a verificação de conformidade pode ser realizada após a instanciação (criação de tarefas) ou após a execução (finalização das tarefas).

Na Instanciação o plano de projeto é criado. O plano de projeto criado é composto por um conjunto de tarefas, onde são informadas as datas de início e fim previsto, o ator, artefatos a serem produzidos e a atividade do processo de referência quando possível. No caso da operação *Add*, não é registrada a atividade do processo.

Na Execução as tarefas criadas são realizadas tanto pelos desenvolvedores quanto automaticamente. A partir da Execução são gerados os registros de execução.

Assumindo o encerramento da criação de tarefas, a ***Verificação da Conformidade do Plano de Projeto*** pode ser executada. Nesta atividade ocorre a verificação de conformidade do plano com o modelo de processo a fim de identificar não conformidades. Quando não houver mais alterações a serem feitas no plano a etapa seguinte pode ser realizada. Uma vez executadas as tarefas é possível iniciar a etapa ***Verificação da Conformidade da Execução***. Nesta etapa ocorre a verificação de conformidade dos registros de execução com o modelo de processo a fim de identificar não conformidades.

A realização da ***Verificação da Conformidade do Plano de Projeto*** possibilita que os benefícios do mapeamento, na identificação de conformidades, sejam alcançados ao longo do projeto e não somente ao final, permitindo ao gerente do projeto tomar a decisão sobre as formas de resolução das não conformidades identificadas no decorrer do projeto de software.

A partir de um estudo de observação realizado em uma reunião de planejamento de projeto e das boas práticas recomendadas pelas normas e modelos de maturidade, um conjunto de atividades e tarefas para verificação de conformidade do plano de projeto e da execução foi definido. O Apêndice G apresenta as orientações para verificação de conformidade.

Portanto, no contexto dessa solução, a comparação realizada para identificação de não conformidades, é realizada pelo apoio ferramental. Detalhes de como usar o apoio ferramental está descrito no Capítulo 7 e no Apêndice D. No Apêndice F é apresentado o pseudocódigo da identificação de não conformidade relacionada às atividades.

Em seguida a identificação de não conformidades, um relatório de não conformidade é gerado.

6.3.3 Gerar Relatório de não Conformidades

A análise de conformidade entre o modelo de Processo de Software de Projeto e o Processo de Software Executado, gera uma lista dos elementos de processo, a qual é composta das não conformidades ocorridas ao longo de cada execução do processo. Para representar as não conformidades identificadas é utilizado o mecanismo *localContribution* e *Supression*, da extensão BPMNt (PILLAT; OLIVEIRA; FONSECA, 2012) para destacar as alterações de inclusão e remoção de atividades.

6.4 Considerações Finais

Este Capítulo apresentou orientações para identificar o processo de software executado e identificar não conformidades, a partir de uma base de dados de execução de projetos criada e estruturada por meio da solução de mapeamento entre as perspectivas de processo proposta nessa tese.

O Capítulo seguinte apresenta o apoio ferramental desenvolvido para dar suporte ao mapeamento entre as perspectivas de processo e projeto de software.

7 Apoio Ferramental

Este capítulo apresenta o apoio ferramental desenvolvido para suporte ao mapeamento entre representações de processo e projeto de software. As operações de instanciação definidas foram implementadas visando apoiar, de forma semi-automatizada, o mapeamento entre as perspectivas de processo e projeto.

7.1 Introdução

Considerando o objetivo específico apresentado no Capítulo 1 na Seção 1.3 – **Oferecer apoio ferramental ao mapeamento entre as perspectivas**; um apoio ferramental foi desenvolvido para dar suporte à solução de mapeamento entre as perspectivas de processo e projeto de software.

Conforme apresentado no Capítulo 5 o processo de aplicação da solução proposta nesta tese é composto das seguintes etapas: *Definição* (obtenção do modelo do processo de software), *Instanciação* (criação de tarefas) e *Execução* (finalização de tarefas). Neste sentido, um apoio ferramental foi desenvolvido para dar suporte a estas etapas. A ferramenta também provê suporte à identificação do processo de software executado, por meio da exportação dos registros no formato .xes, bem como a identificação de não conformidades, por meio da geração de relatórios composto das atividades previstas executadas, atividades previstas não executadas e atividades não previstas executadas, para cada caminho possível de execução das atividades.

O apoio ferramental é intitulado *Maptracys*. Este nome foi escolhido para remeter a seu objetivo que é realizar o mapeamento entre as perspectivas, mantendo o registro (trace) deste mapeamento. Uma versão da *MapTracys* pode ser acessada pelo seguinte endereço https://maptracys-test.herokuapp.com/.

7.2 Arquitetura

Esta seção detalha a arquiteura da *Maptracys*. O apoio ferramental foi implentado de acordo com a arquitetura apresentada na Figura 9.

A arquitetura é divida em três camadas:

1. **Camada de Interface do Usúario**: camada onde ocorre toda a interação/visualização das informações produzidas nas demais camadas. Esta camada está diretamente interligada à camada de mapeamento e apresenta o resultado do processamento realizado na camada de

apresentação. A partir dela, é possível que o usuário interaja com todas as funcionalidades disponibilizadas pela ferramenta.

2. **Camanda de Mapeamento**: esta é a camada onde ocorre toda a manipulação dos dados e gerenciamento das funcionalidades da ferramenta. Esta camada é responsável por receber os comandos oriundos da camada de interface do usuário, executar a tarefa, e devolver o resultado. Esta camada de mapeamento e composta por três módulos: Definição, Instanciação e Execução.
3. **Camada de Apresentação**: camada responsável por receber os dados oriundos do banco de dados e da camada de mapeamento, processar os dados para transformá-los em informações mais facilmente compreensíveis ao usuário, converter o resultado em um formato que possa ser lido pelo usuário e retorná-lo para a camada de mapeamento. Nesta camada também são gerados os documentos que são exportados pela ferramenta.

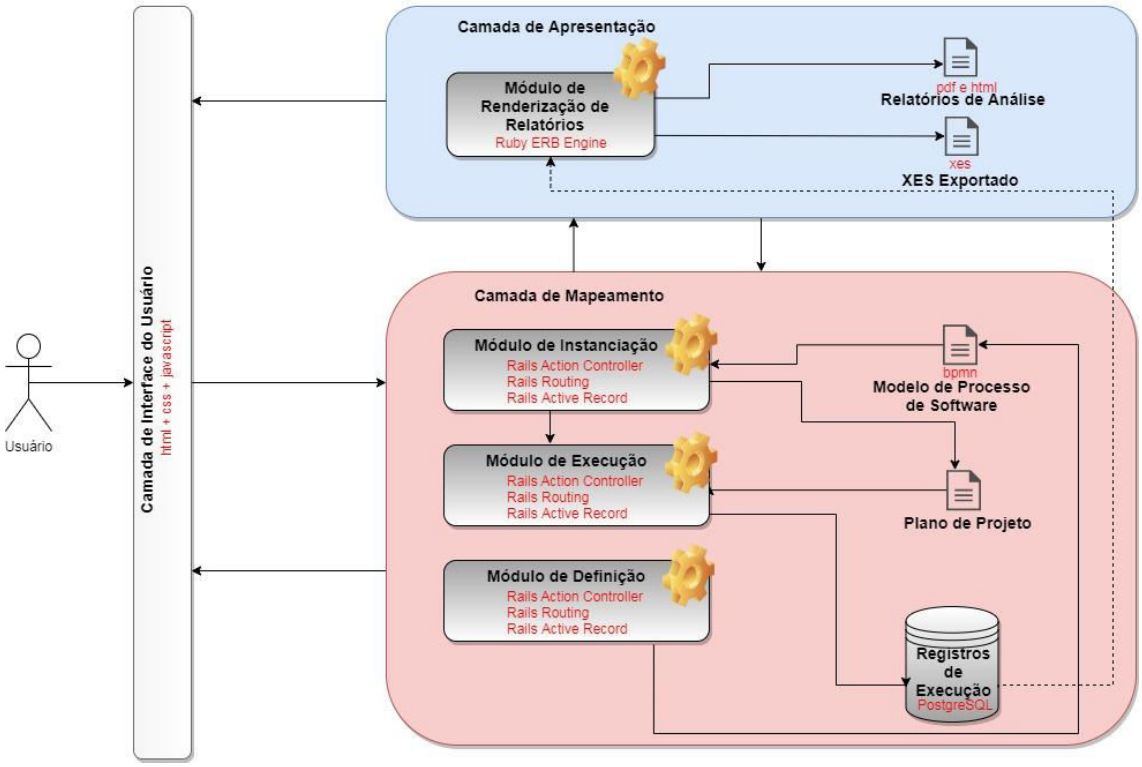


Figura 9 - Arquitetura da Maptracys. Fonte: A Autora.

7.3 Implementação da Maptracys

A Maptracys foi implementada utilizando o framework Ruby on Rails (RUBY ON RAILS, 2019). A Figura 8 exibe as camadas da arquitutera, onde as principais tecnologias utilizadas para desenvolver a ferramenta estão destacadas em vermelho.

A Camada de Mapeamento é composta pelos módulos de Definição, Instanciação e Execução. E a Camada de Apresentação é composta pelo módulo de Renderização de Relatórios.

Cada módulo da Camada de Mapeamento é composto por três níveis: Rails Action Controller (controlador), Rails Routing (roteador) e Rails Active Record (acesso aos dados). O Rails Action Controller recebe a requisição da camada de interface e redireciona esta para um controlador que possua a capacidade de cumprí-la. O Rails Action Controller é responsável por receber a requisição redirecionada pelo Rails Routing, invocar a funcionalidade que resolva a requisição e devolver o resultado. O acesso aos dados (Rails Active Record) é feito através de um DAO (Data Access Object) que se comunica com o banco de dados para obter as informações solicitadas por uma funcionalidade do controlador. A Camada de Apresentação é composta pelo módulo de Renderização de Relatórios. Este módulo processa os dados para transformá-los em informações, convertendo o resultado em um formato que possa ser lido pelo usuário.

Cada módulo será descrito nas subseções a seguir.

7.3.1 Módulo de Definição

No módulo de **Definição** está implementada a funcionaldiade de **Obtenção do Modelo do Processo de Software**. Nesta funcionalidade ocorre a importação do modelo de processo software. Este modelo deve estar no formato .bpmn. A partir do modelo de processo importado é possível visualizar a matriz de adjacência e a lista de caminhos (*List Path*). Estas duas funcionalidades foram desenvolvidas para dar suporte a identificação de não conformidades. Além disso, é possível visualizar a lista projetos (*Project*) associados a este processo, além da lista de atividades (*Activity*), eventos (*Events*) e *gateways* (*Gateways*). No Apêndice E é apresentado o pseudocódigo da geração da matriz de adjacência e listagem de caminhos.

A Figura 10 apresenta a tela que exibe um modelo de processo importado.

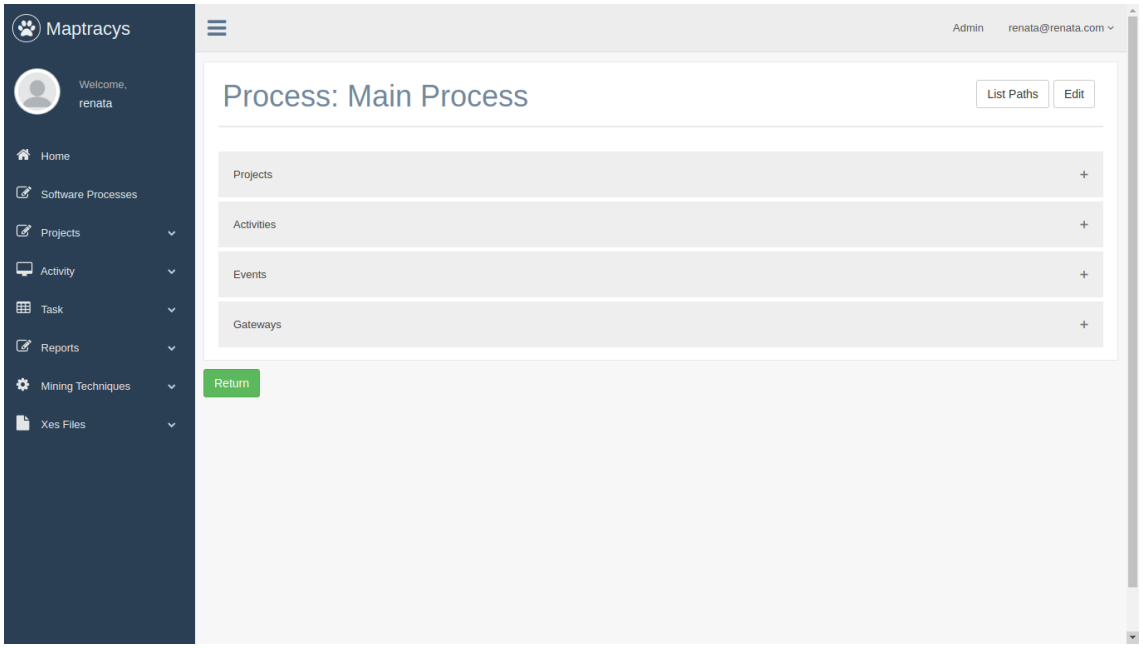


Figura 10 - Tela do modelo de processo importado. Fonte: A Autora.

7.3.2 Módulo de Instanciação

No módulo de **Instanciação** estão implementadas as funcionaldiade de cadastro de projeto, instâncias e iterações, bem como o cadastro de tarefas.

Para apoio ao *Mapeamento entre as perspectivas de processo e projeto de software* as operações de instanciação foram implementadas, sendo aplicadas no cadastro de tarefas. A Figura 11 apresenta a tela da funcionalidade de cadastro do projeto, onde o mesmo deve ser associado a um ou mais processos de software importado.

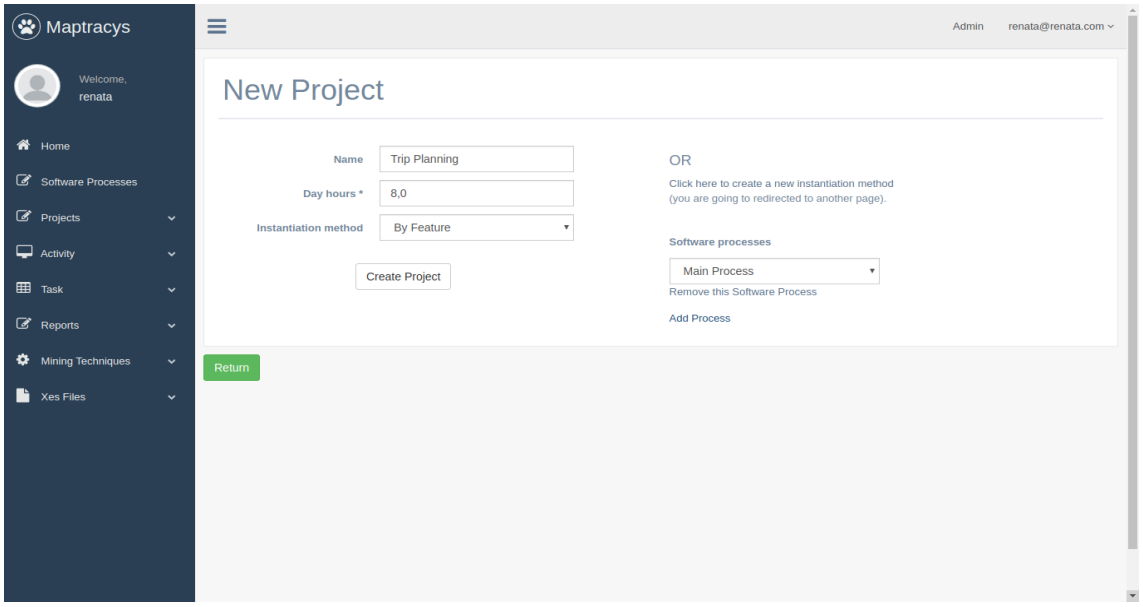


Figura 11 - Tela de cadastro de Projeto. Fonte: A Autora.

As Figuras 12 e 13 apresentam as telas das funcionaldiades de cadastro de Instância e Iterações, respectivamente.

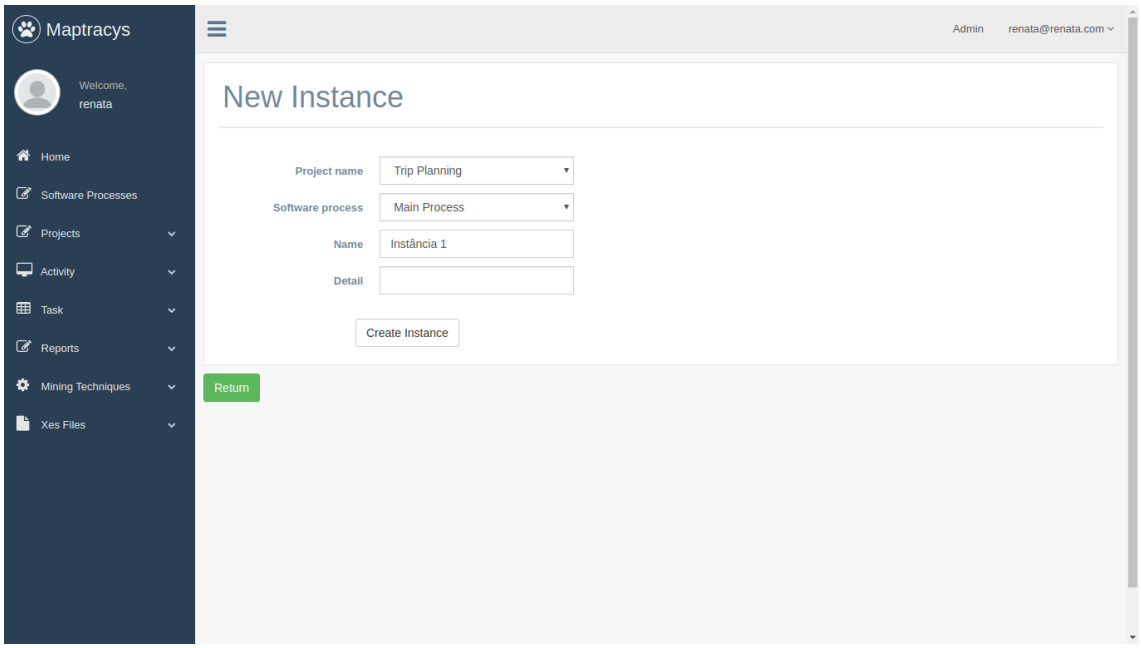


Figura 12 - Tela de cadastro de Instância. Fonte: A Autora.

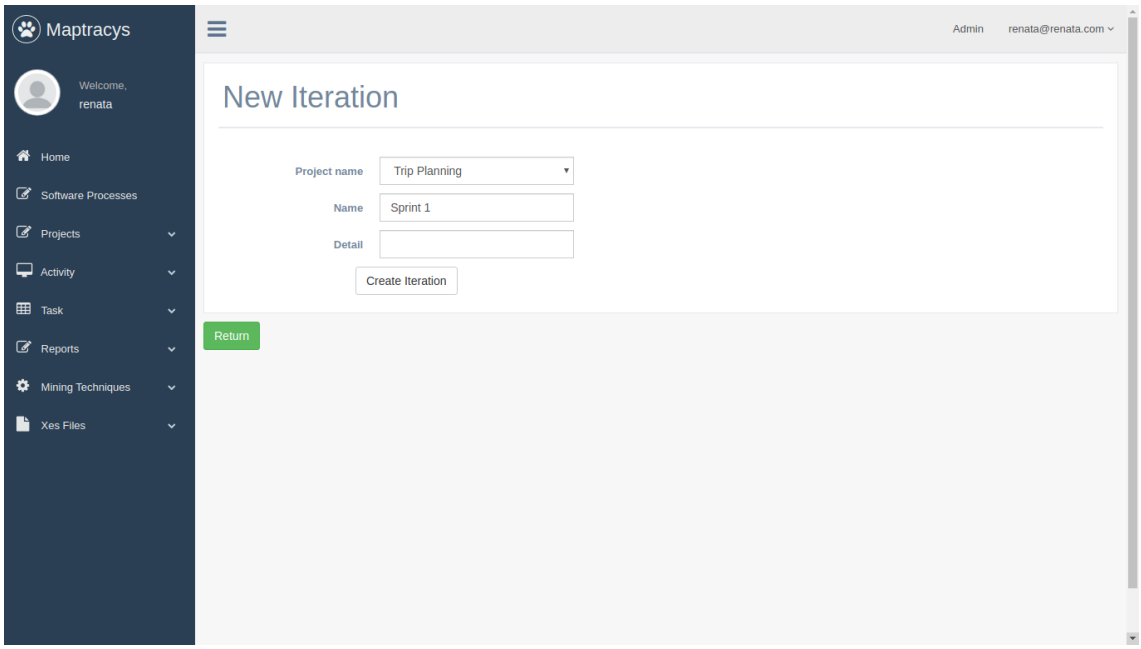


Figura 13 - Tela de cadastro de Iteração. Fonte: A Autora.

A partir da criação do projeto, é possível criar as tarefas e visualizar o andamento das mesmas. A criação das tarefas é realizada por meio das operações de instanciação. As Figuras 14 e 15 apresentam as telas de cadastro de tarefas.

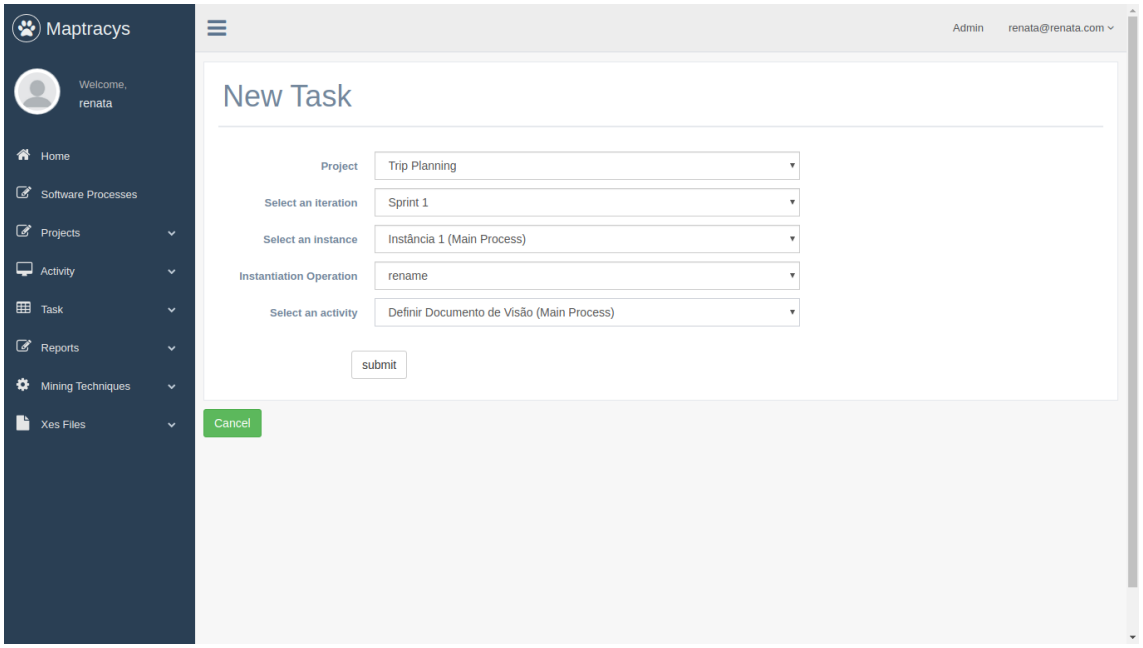


Figura 14 - Tela de cadastro de tarefa. Fonte: A Autora.

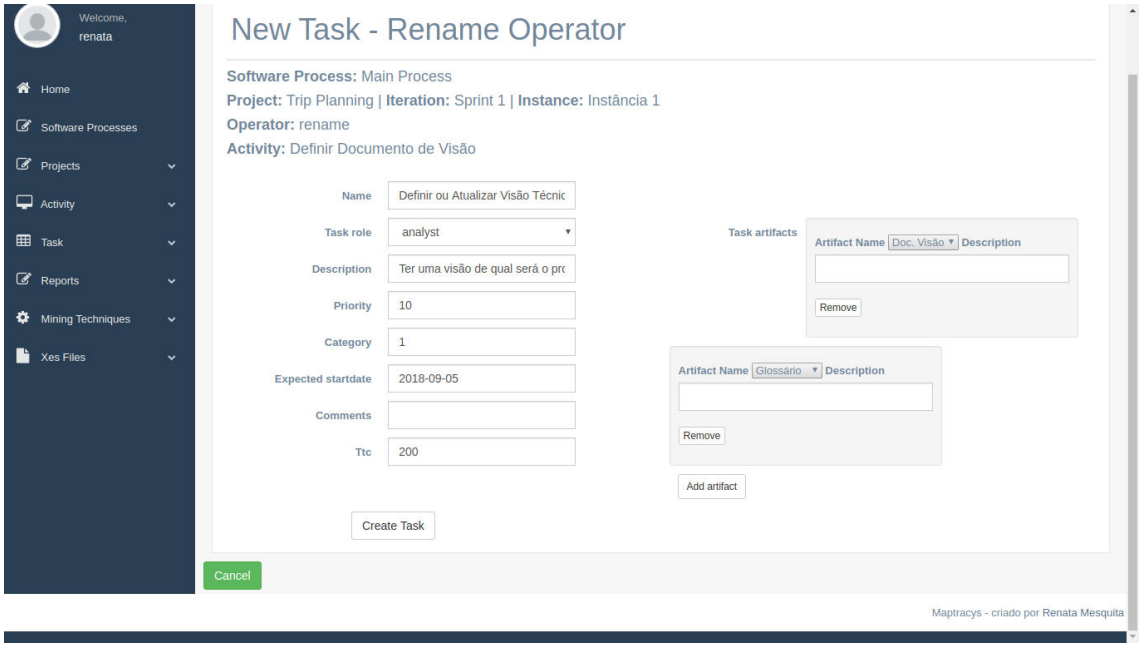


Figura 15 - Tela de cadastro de tarefa. Fonte: A Autora.

A Figura 16 apresenta a tela com um projeto criado.

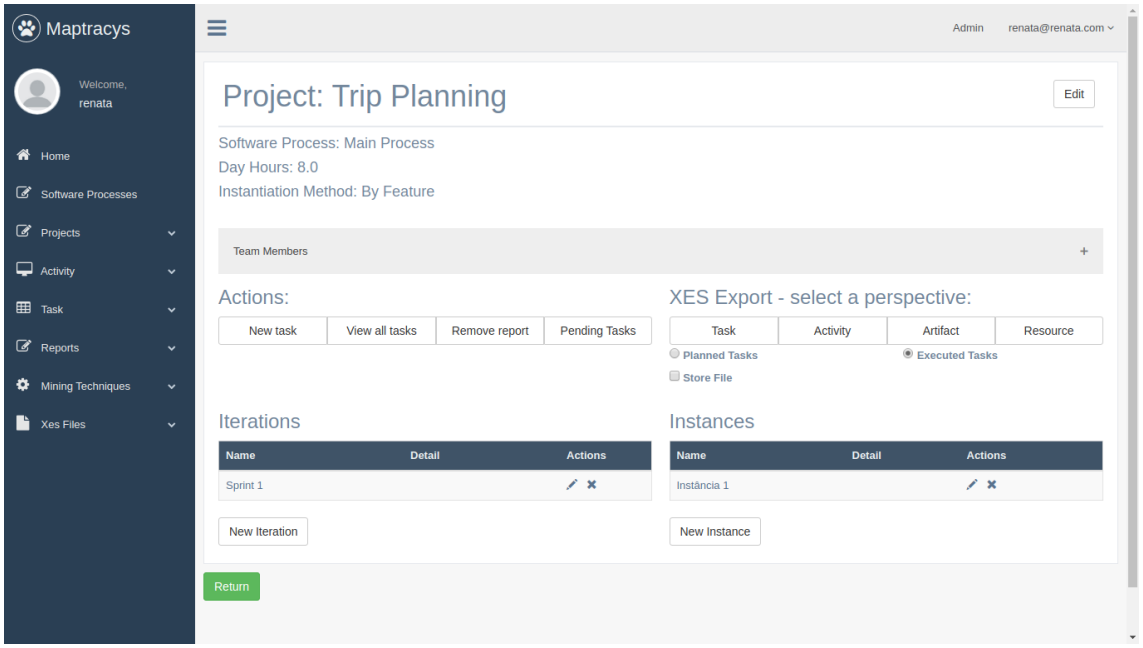


Figura 16 - Tela do projeto criado. Fonte: A Autora.

Como saída deste módulo tem-se uma lista de tarefas, que compõem o plano de projeto. A Figura 17 apresenta a lista de tarefas cadastrada em um projeto.

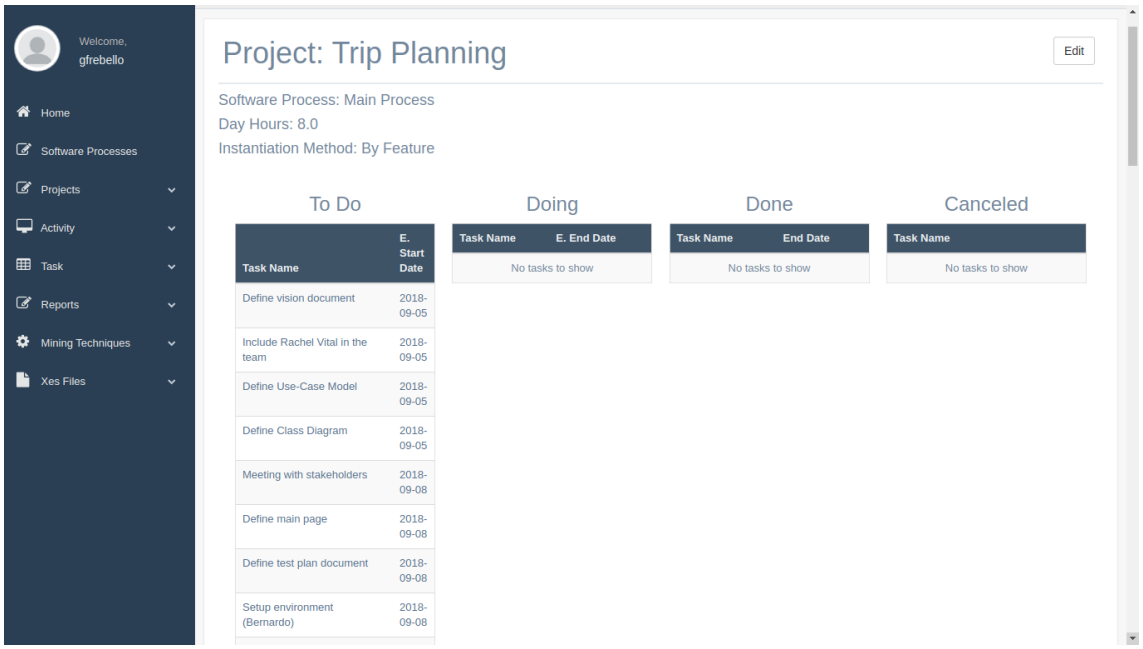


Figura 17 – Lista de tarefas cadastradas no projeto (plano de projeto)

7.3.3 Módulo de Execução

A partir da criação do projeto, é possível criar as tarefas, dar andamento nas tarefas e posteriormente finalizá-las. A Figura 18 apresenta a tela de finalização de tarefas. A partir do registro da data de fim de cada tarefa, as mesmas passam a ser consideradas registros de execução. PostgreSQL (POSTGRESQL, 2019) foi o banco de dados utilizados na ferramenta. A Figura 18 apresenta uma tarefa finalizada.

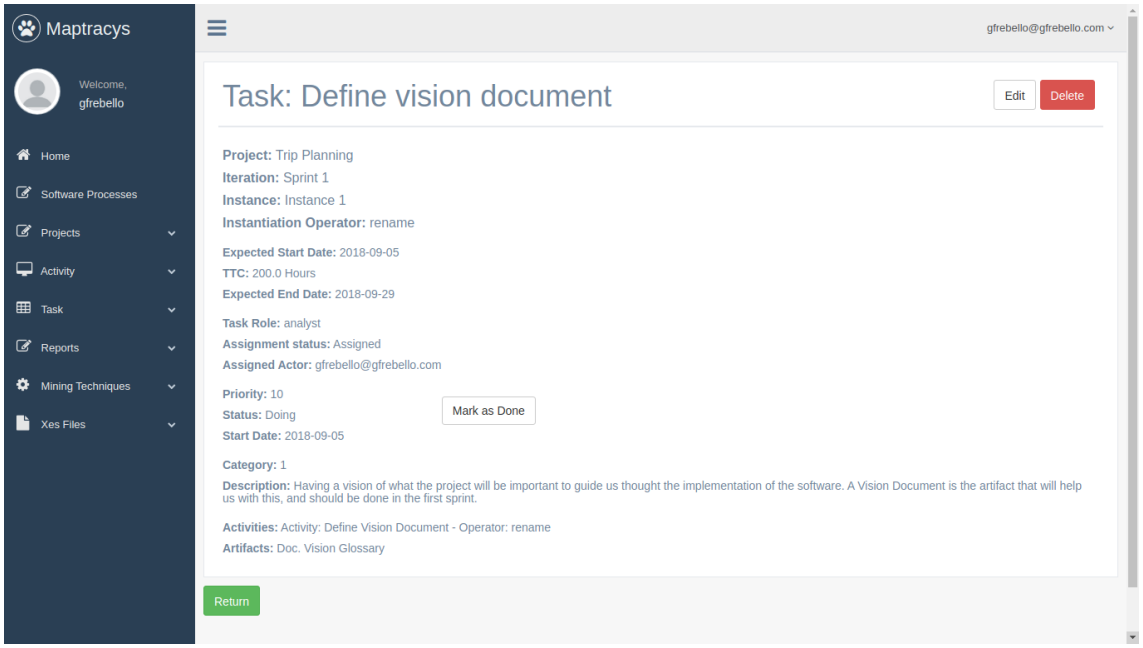


Figura 18 - Tela de tarefa a ser finalizada. Fonte: A Autora.

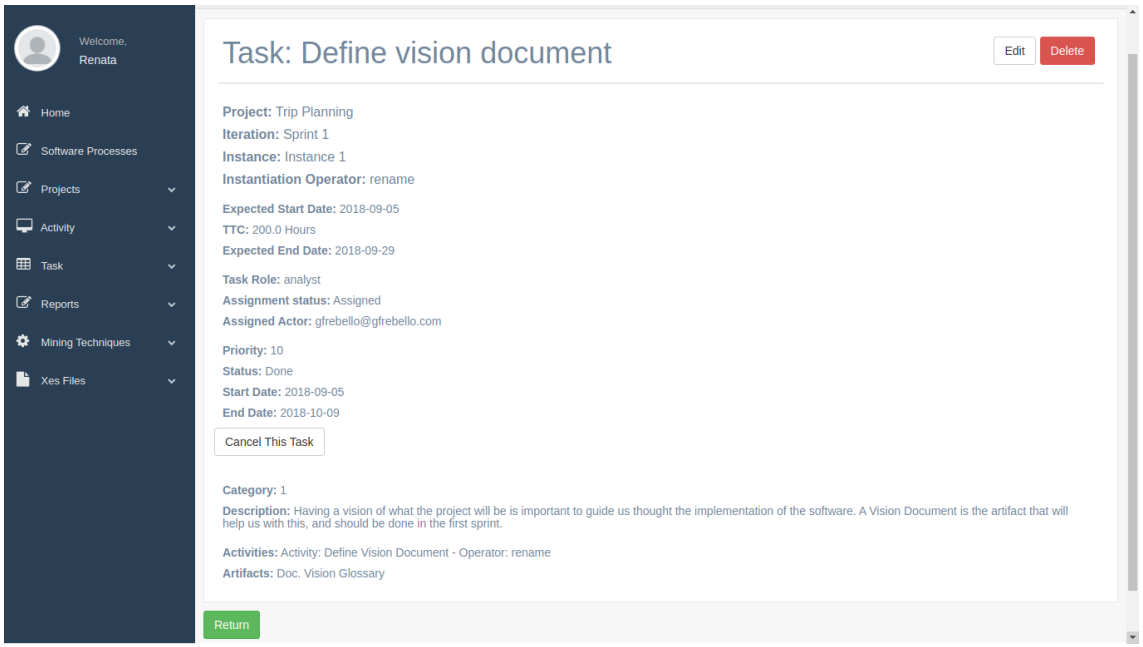


Figura 19 - Tela de tarefa finalizada. Fonte: A Autora.

7.3.4 Módulo de Renderização de Relatórios

O módulo de **Renderização de Relatórios** inclui a geração de relatórios referentes a execução do projeto. Sendo possível gerar relatórios sobre a execução do projeto, como também exportar arquivos no formato .xes. Pela utilização das operações de instanciação os elementos dos processos de software de referência

para o projeto são registrados, e assim podem ser visualizados a partir dos registros gerados durante a execução.

Para suporte a identificação do processo de software executado, o apoio ferramental realiza a exportação dos registros no formato .xes, reconhecido por ferramentas de mineração de processos. Este arquivo pode ser submetido a aplicação de técnicas de mineração de processos.

O arquivo exportado pode considerar as tarefas (Task), atividades (*Activity*), artefatos (*Artifact*) ou recursos (*Resource*). Além disso, pode ser considerada a data de fim prevista (*Foreseen Tasks)* ou a data de fim efetivo (*Executed Tasks*). Utilizando os arquivos exportados é possível identificar os modelos de processo nas seguintes perspectivas: Fluxo de Controle (Atividades), Organizacional (Recursos) ou Informacional (Artefatos), ou seja, modelos de processo relacionados ao fluxo das atividades, uso dos recursos ou entrega dos produtos de software (artefatos). A Figura 19 apresenta a tela de exportação de arquivo no formato .xes.

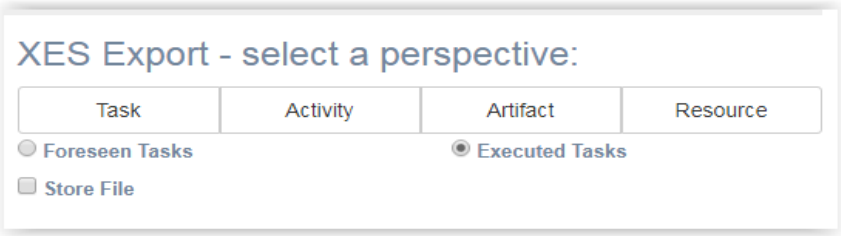


Figura 20 - Tela de exportação de arquivo no formato .xes. Fonte: A Autora.

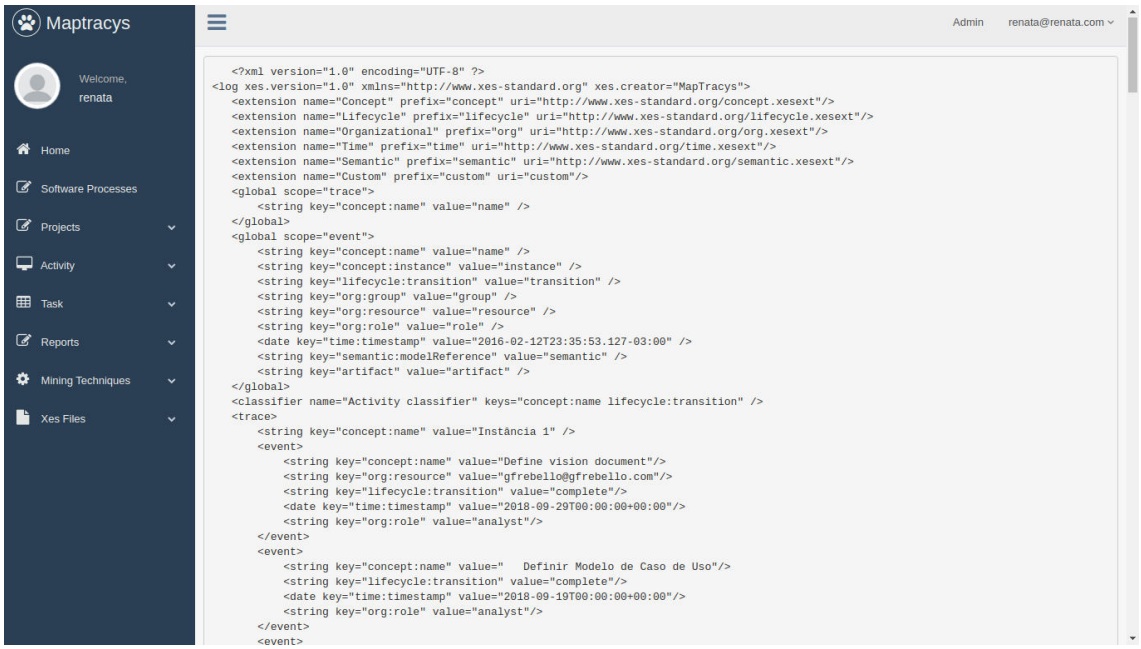


Figura 21 - Tela do arquivo no formato .xes exportado. Fonte: A Autora.

A partir da exportação do arquivo no formato .xes, o mesmo pode ser submetido a ferramentas de mineração de processo, conforme descrito na Seção 6.2.1. A Figura 21 apresenta um arquivo no formato .xes exportado pela ferramenta. Está em desenvolvimento no apoio ferramental o algoritmo o *Heuristic Mining* (mineração heurística) (VAN DER AALST, 2011), o qual tem o objetivo de usar a frequência dos eventos e a sequência para gerar o modelo de processo.

Para a identificação de não conformidade foram implementados relatórios para exibir a lista de atividades previstas no modelo de processo associado ao projeto, as atividades previstas executadas e as atividades previstas não executadas. Esra em desenvolvimento o relatório para análise de sequência, bem como o relatório de oportunidade de melhorias, onde são listadas as tarefas criadas com as operações *Add* e *Merge*. A Figura 22 apresenta a tela inicial do relatório das não conformidades relacionadas às atividades.

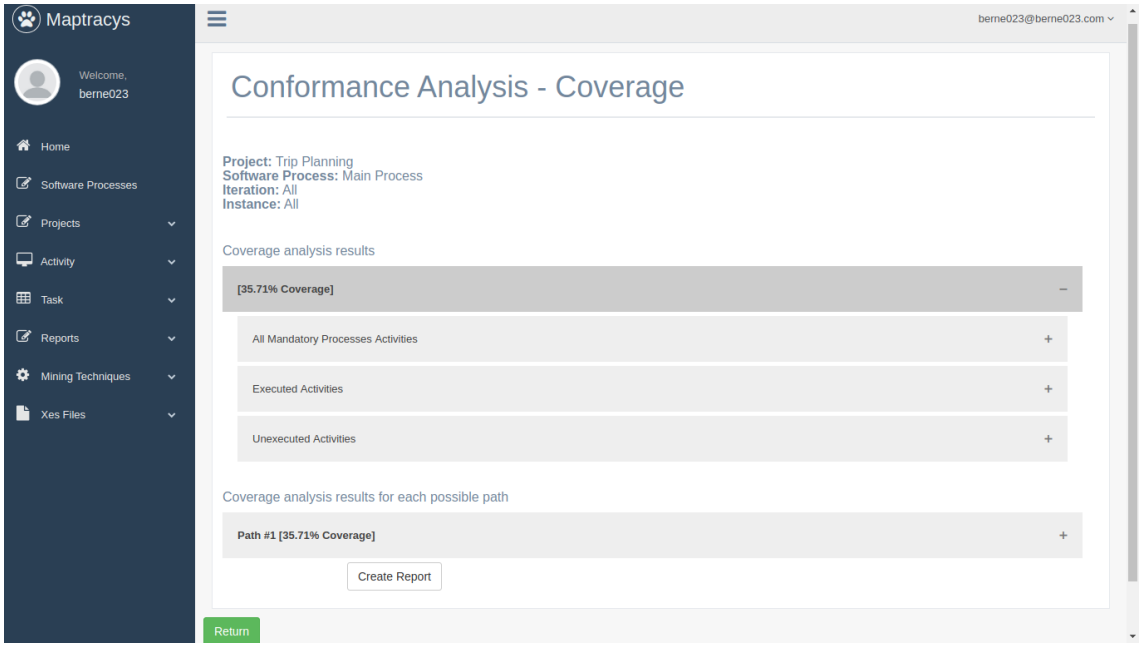


Figura 22 - Tela inicial do relatório das não conformidades relacionadas às atividades.
Fonte: A Autora.

No Apêndice F é apresentado o pseudocódigo da identificação de não conformidade relacionada às atividades.

No Apêndice D é fornecido o manual de uso onde pode ser visualizado como executar as funcionalidades descritas neste capítulo.

7.4 Considerações Finais

Este capítulo apresentou o apoio ferramental, intitulado Maptracys, desenvolvido para dar apoio ao mapeamento entre as perspectivas de processo e projeto de software. A arquitetura da implementada foi apresentada e seus principais módulos foram descritos.

O Capítulo seguinte apresenta os estudos conduzidos como forma de demonstrar os benefícios do mapeamento entre as perspectivas de processo e projeto de software.

8 Avaliação dos Benefícios do Mapeamento na Identificação do Processo de Software Executado e na Identificação de não Conformidades

O objetivo deste Capítulo é apresentar os estudos conduzidos visando avaliar os benefícios do mapeamento entre as perspectivas de processo e projeto de software na identificação do processo de software executado e na identificação de não conformidades.

8.1 Introdução

Este capítulo apresenta os estudos conduzidos para avaliar o benefício da aplicação da solução de mapeamento abordada por esta tese, na identificação do processo de software executado e na identificação de não conformidade. Nesta pesquisa conjecturamos que ao aplicar a estratégia de mapaeamento proposta, às tarefas do projeto, e por consequência, aos registros de execução, passam a possuir informações explícitas do processo de software de referência para o projeto de software. Essas informações são úteis para apoiar a análise da execução dos processos. Portanto, os estudos descritos a seguir tiveram como base as orientações para demonstração dos benefícios do mapeamento entre as perspectivas apresentadas no Capítulo 6.

A Seção 8.2 apresenta o estudo realizado para avaliar o benefício da solução proposta na identificação do processo de software executado e a Seção 8.3 apresenta o estudo realizado para avaliar o benefício na identificação de não conformidade. Por fim, a Seção 8.4 apresenta as considerações finais.

8.2 Identificando o Processo de Software Executado

Esta seção apresenta o estudo conduzido para avaliar o benefício da aplicação da solução de mapeamento abordada por esta tese, na identificação do processo de software executado. Para avaliar este benefício métricas de complexidade dos modelos de processos foram adotadas (ver Subseção 6.2.2). As métricas têm como objetivo demonstrar quantitativamente o benefício do mapeamento entre as

perspectivas na redução da complexidade de entendimento dos processos de software executados, descobertos pela aplicação de técnicas de Mineração de Processo.

O estudo executado utilizou duas bases de dados de projetos de desenvolvimento de software obtidas de uma empresa de desenvolvimento de software. Nessas bases obtidas a estratégia de mapeamento estava aplicada, por meio da representação da atividade do processo de cada tarefa, bem como da instância do processo. Com a utilização destas bases, o modelo de processo de software executado foi descoberto, por meio da aplicação de técnicas de Mineração de Processo, possibilitando assim a extração das métricas que suportam a análise da complexidade dos modelos de processo descobertos. Portanto, a premissa adotada foi: se o modelo de processo descoberto a partir de registros de execução mapeados é menos complexo que o modelo processo extraído a partir de registros não mapeados, isso reforça que a análise da execução dos processos pode ser beneficiada.

A execução do estudo teve como base as orientações para demonstração dos benefícios do mapeamento entre as perspectivas na identificação do processo de software executado apresentadas no Capítulo 6.

A Subseção 8.2.1 apresenta o planejamento do estudo. Na Subseção 8.2.2 a execução do estudo é descrita e na Subseção 8.2.3 os resultados são discutidos. E, a Subseção 8.2.4 discute as ameaças à validade do estudo.

8.2.1 Planejamento do Estudo

Seguindo a abordagem GQM (*Goal, Question, Metric*) (BASILI; CALDIERA e ROMBACH, 1994), o objetivo do estudo foi definido como:

Analisar os modelos de processo de software executados

Com o propósito de caracterizar

Com respeito à redução da complexidade dos modelos de processo de software executados

Do ponto de vista de engenheiros de software

No contexto de projetos de desenvolvimento de software em uma organização de médio porte.

A hipótese definida é "*O mapeamento entre as perspectivas de processo e projeto beneficia a redução da complexidade do modelo de processo de software executado*".

É importante ressaltar que, a perspectiva adotada sobre a existência ou não de benefícios está associada aos valores das métricas adotadas que serão extraídos dos modelos de processos executados.

Na seção a seguir os passos da execução do estudo serão descritos.

8.2.2 Execução do Estudo

A execução do estudo foi realizada de acordo com as atividades para identificação do processo executado apresentadas no Capítulo 6, compreendendo as seguintes atividades: *Obter Registros de Execução do Processo Mapeados*, *Preprocessar Registros de Execução do Processo*, *Identificar Processo de Software Executado*, e *Extrair Métricas de Complexidade*.

Na atividade "**Obter registros de execução mapeados**" a base de execução do projeto expõe o conjunto de tarefas executadas ao longo do projeto, neste sentido a mesma é extraída de ferramentas utilizadas ao longo do projeto para gerenciamento das tarefas. Nas bases utilizadas neste estudo, devido à proximidade dos gerentes de projeto da empresa com o grupo de pesquisa PRISMA, foi possível customizar a ferramenta de gerenciamento de tarefas utilizada por eles para e adicionar o campo chamado "*AtividadeBase*". Este campo foi usado na criação das tarefas, ou seja, na instanciação, para informar a atividade do processo de software definido referente a cada tarefa. Com a representação o campo "*AtividadeBase*", acrescida da informação da instância do processo (a cada versão o processo era instanciado), as bases de registros de execução obtidas foram considerados mapeadas. Caso as operações de instanciação tivessem sido aplicadas na criação das tarefas, a estrutura resultante teria também a representação da atividade do processo de cada tarefa, bem como da instância do processo, possibilitando que tais informações também estivessem representadas de forma explícita. As subseções 8.2.2.1 e 8.2.2.2 provêem mais informações de contexto das bases utilizadas nesse estudo.

Na atividade "**Pré-processar Registros de Execução**" a base de dados obtida foi convertida para o formato .xes. O apoio ferramental utilizado foi a Disco para converter o arquivo em .csv em um arquivo no formato .xes. Esta conversão foi feita para o formato utilizado como entrada para a aplicação de técnicas de mineração de processos.

Na atividade "**Identificar o processo de software executado**" a base dados de execução no formato .xes é submetida a aplicação das técnicas de mineração de processo. Foram extraídos os modelos a partir de registros não mapeados e mapeados. Os modelos partir de registros não mapeados foram descobertos apenas para se ter uma base de comparação para verificar a existência ou não do benefício do mapeamento na redução da complexidade dos modelos de processos descobertos. O apoio ferramental utilizado foi a Disco. Esta ferramenta foi escolhida pela facilidade

de extração dos modelos de processo extraídos e pela qualidade da visualização dos mesmos.

Na última atividade "**Extrair Métricas de Complexidade**" é realizada a extração das métricas, a partir dos modelos de processo de software executado. Métricas estas que apoiam a análise de complexidade dos modelos de processo descobertos. O apoio ferramental utilizado foi a Disco e a ProM. Na ferramenta Disco, as seguintes métricas foram extraídas: *número de atividades* e *frequência máxima de ocorrência da atividade*. Na ferramenta ProM as seguintes métricas foram extraídas: *extended cardoso metric*, *número de arcos* e *número de locais*. Na ferramenta ProM os registros de execução foram importados no formato .xes, minerados usando o *plugin Heuristics Miner*. O modelo minerado foi convertido para o formato Rede de Petri também por meio de um *plugin Convert Heuristics net into Petri net* da ProM. Em seguida, pela aplicação do *plugin Show Petri-net Metrics* as métricas foram extraídas.

A seguir são descritas algumas orientações que apoiam a interpretação das métricas adotadas:

- **Número de Atividades:** um baixo valor para esta métrica indica menor complexidade do modelo.
- **Frequência máxima:** um alto valor para esta métrica indica menor complexidade do modelo.
- **Extended Cardoso Metric:** um baixo valor para essa métrica indica menor complexidade do modelo.
- **Número de arcos:** Um baixo valor para esta métrica indica menor complexidade do modelo.
- **Número de locais:** Um baixo valor para esta métrica indica menor complexidade do modelo.

A seguir são apresentados os resultados de cada passo definido, na execução do estudo, nos projetos: Projeto 1 (Subseção 8.2.2.1) e Projeto 2 (Subseção 8.2.2.2).

8.2.2.1 Base de dados do Projeto 1

O Projeto 1 tem como objetivo o desenvolvimento de um sistema portuário. Este projeto foi realizado em uma empresa de desenvolvimento de software que atualmente possui um quadro de 30 funcionários, sendo a maioria dos desenvolvedores de software. A empresa utiliza métodos ágeis há pelo menos 10 anos. As equipes são pequenas, geralmente, formadas por grupos de 3 a 8 pessoas sendo: analistas de sistemas, desenvolvedores e testadores.

A base de dados obtida deste projeto possui 24 instâncias, com um total de 2741 registros de execução.

Passo 1: **Obter Registros de Execução Mapeados**

Os registros de execução utilizados nesta avaliação foram extraídos da ferramenta Redmine (REDMINE, 2015). Esta ferramenta é utilizada no Projeto 1 para gerenciamento e monitoramento das tarefas que são executadas no desenvolvimento. Os seguintes dados foram extraídos: *Projeto, Tipo, Tarefa pai, Situação, Prioridade, Título, Autor, Atribuído para, Alterado em, Categoria, Versão, Início, Data prevista, Tempo estimado, Tempo gasto, % Terminado, Criado em, Concluído, Tarefas relacionadas, Subprocesso, AtividadeBase, PapelB, Origem, Equipe, Nível de complexidade, Tempo previsto, Data Inicio da Homologação, Data Produção, Chamado, Data Conclusão da Homologação, Responsável pela Homologação e Privado*. Neste projeto a cada versão o processo era instanciado, neste sentido pode-se considerar com a instância do processo o campo "*Versão*". A base de dados obtida estava no formato .csv. Com a representação do campo "*AtividadeBase*", acrescida da informação da instância do processo, as bases de registros de execução obtidas foram considerados mapeadas.

Passo 2: **Pré-processar Registros de Execução**

Após a obtenção dos registros de execução mapeados, no formato .csv, o preprocessamento realizado foi a conversão para o formato .xes foi realizada pelo uso da ferramenta Disco.

Passo 3: **Identificar Processo de Software Executado**

Neste passo o processo de software executado foi identificado a partir dos registros de execução do processo. Esta identificação foi realizada pela ferramenta Disco. A Figura 23 apresenta o modelo de processo descoberto a partir dos registros de execução não mapeados, ou seja, sem considerar as informações do processo nos registros. A Figura 24 apresenta o modelo de processo descoberto a partir dos registros de execução mapeados, ou seja, considerando as informações do processo nos registros de execução. O modelo de processo descoberto a partir dos registros de execução não mapeados foi extraído para ser usado como base de comparação para verificar a existência ou não do benefício do mapeamento na redução da complexidade do modelo de processo obtido.

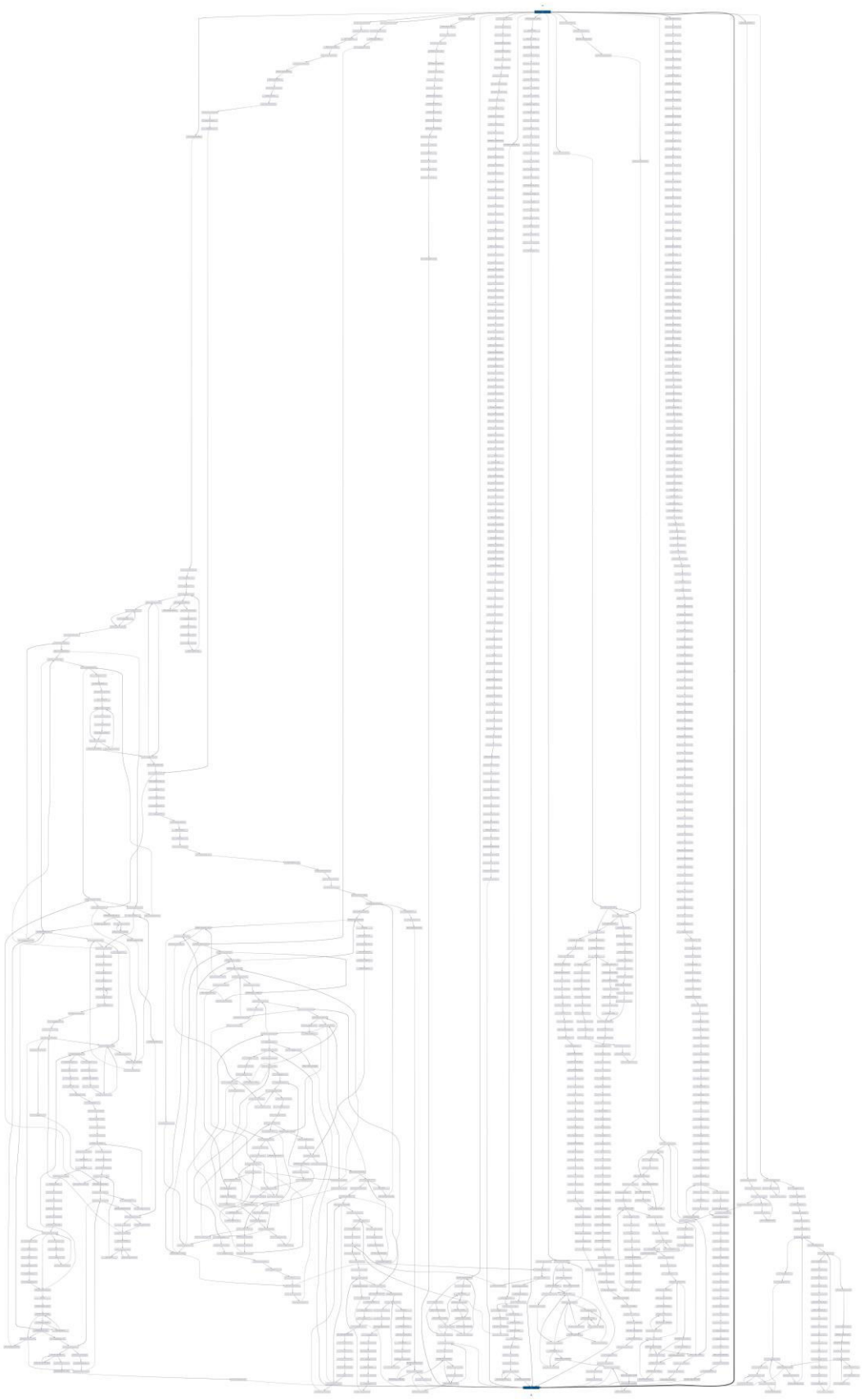


Figura 23 - Modelo de processo descoberto a partir dos registros de execução não mapeados da base de dados do Projeto 1. Fonte: A Autora.

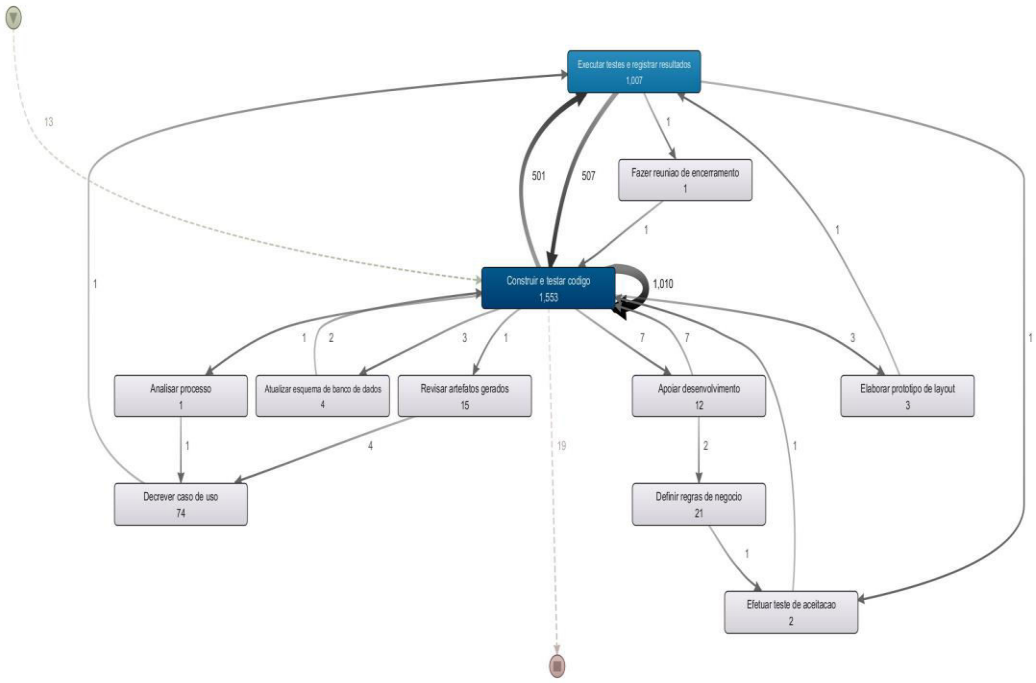


Figura 24 - Modelo de processo descoberto a partir dos registros de execução mapeados da base de dados do Projeto 1

Na ferramenta Disco, no modelo de processo descoberto as atividades com maior frequência são destacadas com coloração mais acentuadas, além de ser destacado em cada linha de transição entre as atividades a informações no número de ocorrência da mesma. No modelo de processo descoberto, da Figura 24, é possível observar que as atividades *Construir e testar código* e *Executar testes e registrar resultados*, apresentam coloração diferenciada das demais.

A Tabela 14 apresenta a relação de atividades, a frequência de ocorrência, bem como a frequência relativa de cada atividade do modelo de processo extraído sob a ótica das atividades do processo (Figura 24).

Tabela 14 - Atividades e Frequência mostrado na Figura 24

Atividade	**Frequência**	**Frequência Relativa**
Construir e testar código	1553	57.67 %
Executar testes e registrar resultados	1007	37.39 %
Descrever caso de uso	74	2.75 %
Definir regras de negócio	21	0.78 %
Revisar artefatos gerados	15	0.56 %
Apoiar desenvolvimento	12	0.45 %
Atualizar esquema de banco de dados	4	0.15 %

Elaborar protótipo de layout	3	0.11 %
Efetuar teste de aceitação	2	0.07 %
Analisar processo	1	0.04 %
Fazer reunião de encerramento	1	0.04 %

Passo 4: **Extrair Métricas de Complexidade**

A partir da identificação do processo de software executado foi possível extrair métricas. As métricas extraídas estão apresentadas na Tabela 15. A tabela apresenta as métricas extraídas do modelo de processo descoberto a partir de registros não mapeados e mapeados, com o objetivo de demonstrar o benefício do mapeamento na identificação do processo de software executado.

As métricas *números de atividades* e *frequência máxima de ocorrência da atividade* foram extraídas pelo uso da ferramenta Disco. E as métricas *extended cardoso metric*, *número de arcos* e *número de locais* foram extraídas pelo uso da ferramenta ProM.

Conforme descrito para extrair as métricas *extended cardoso metric*, *número de arcos* e *número de locais* o modelo deve estar no formato Rede de Petri. Porém, devido a quantidade de instâncias e registros de execução, não foi possível gerar o modelo descoberto pelo uso da ferramenta ProM, por limitação de processamento do equipamento utilizado. Neste sentido consideramos para a extração destas métricas apenas 1 instância.

Tabela 15 – Métricas extraídas dos modelos mostrados nas Figuras 23 e 24

Métricas	**Modelo de processo descoberto a partir de registros não mapeados**	**Modelo de processo descoberto a partir de registros mapeados**
número de atividades	2238	11
frequência máxima atividade	24	1553
extended cardoso metric	323.0	23.0
número de locais	323.0	20.0
número de arcos	646.0	46.0

Conforme pode ser observado na Tabela 15, as métricas extraídas a partir dos registros de execução do Projeto 1 indicam que o mapeamento entre as perspectivas de processo e projeto beneficia a redução da complexidade do modelo de processo de software executado. É possível observar que no modelo de processo descoberto a partir de registros mapeados, os valores das métricas número de atividades, *extended*

cardoso metric, número de locais e são menores, indicando uma redução na complexidade do modelo, quando comparado aos valores das métricas extraídas do modelo de processo descoberto a partir de registros não mapeados. O valor da métrica frequência máxima da atividade foi maior, quando considerando os registros de execução modelo de processo descoberto a partir de registros mapeados. Porém, isso indica redução da complexidade, uma vez que um alto valor para esta métrica indica menor complexidade do modelo.

8.2.2.2 Base de dados do Projeto 2

Este projeto foi realizado em uma empresa de desenvolvimento de software que atualmente possui um quadro de 30 funcionários, sendo a maioria dos desenvolvedores de software. A empresa utiliza métodos ágeis há pelo menos 10 anos. As equipes são pequenas, geralmente, formadas por grupos de 3 a 8 pessoas sendo: analistas de sistemas, desenvolvedores e testadores.

Passo 1: **Obter Registros de Execução Mapeados**

Os registros de execução utilizados nesta avaliação foram extraídos da ferramenta Redmine (REDMINE, 2015). Esta ferramenta é utilizada neste projeto para gerenciamento e monitoramento das tarefas que são executadas no desenvolvimento de um sistema acadêmico. Os seguintes dados foram extraídos: *Tipo, Situação, Título, Tempo estimado, Origem, Tempo gasto, Atribuído para, Categoria, Prioridade, Início, Data prevista, Alterado em, Tarefa pai, PapelB*. Neste projeto a cada versão o processo era instanciado, neste sentido pode-se considerar com a instância do processo o campo "*Versão*". A base de dados obtida estava no formato .csv.

Com a representação o campo "*AtividadeBase*", acrescida da informação da instância do processo, as bases de registros de execução obtidas foram consideradas mapeadas.

Passo 2: **Pré-processar Registros de Execução**

Após a obtenção dos registros de execução mapeados, no formato .csv, o preprocessamento realizado foi a conversão para o formato .xes foi realizada pelo uso da ferramenta Disco.

Passo 3: **Identificar processo de software executado**

Neste passo o processo de software executado foi identificado a partir dos registros de execução extraías da base do processo. Esta identificação foi realizada pela ferramenta Disco. A Figura 25 o modelo de processo descoberto a partir dos

registros de execução não mapeados e a Figura 26 apresenta o modelo de processo descoberto a partir dos registros de execução mapeados. O modelo de processo descoberto a partir dos registros de execução não mapeados foi extraído para ser usado como base de comparação para verificar a existência ou não do benefício do mapeamento na redução da complexidade do modelo de processo obtido.

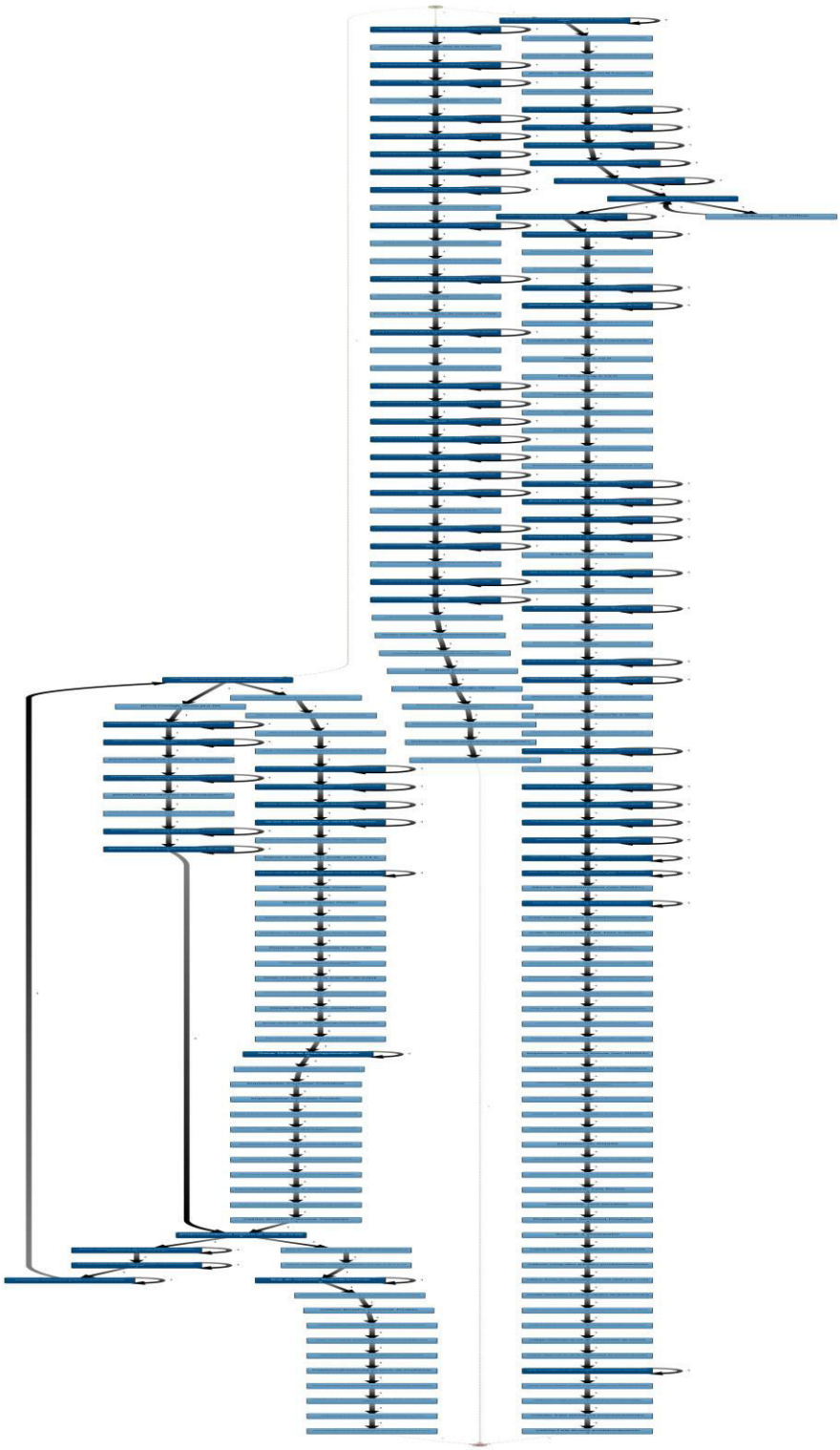


Figura 25 - Modelo de processo descoberto a partir dos registros de execução não mapeados da base de dados do Projeto 2. Fonte: A Autora.

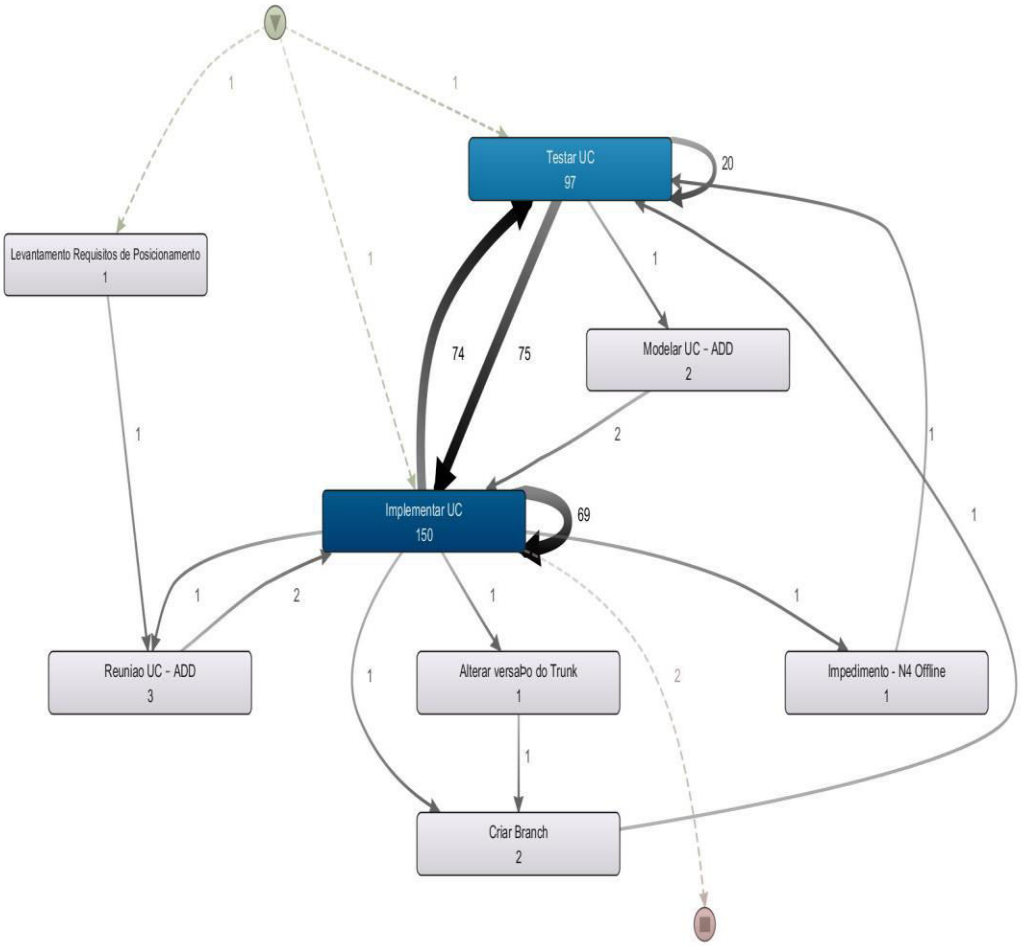


Figura 26 - Modelo de processo descoberto a partir dos registros de execução mapeados da base de dados do Projeto 2. Fonte: A Autora.

Na ferramenta Disco, no modelo de processo descoberto, as atividades com maior frequência são destacadas com coloração mais acentuadas, além de ser destacado em cada linha de transição entre as atividades a informações no número de ocorrência da mesma. No modelo de processo descoberto, da Figura 26, é possível observar que as atividades *Implementar UC* e *Testar UC*, apresentam coloração diferenciada das demais.

A Tabela 16 apresenta a relação de atividades, a frequência de ocorrência, bem como a frequência relativa de cada atividade do processo modelo de processo extraído sob a ótica das atividades do processo.

Tabela 16 - Atividades e Frequência mostrado na Figura 26

Atividade	**Frequência**	**Frequência Relativa**
Implementar UC	150	58.37 %
Testar UC	97	37.74 %
Reunião UC	3	1.17 %
Criar Branch	2	0.78 %
Modelar UC	2	0.78 %
Alterar versão do Trunk	1	0.39 %
Levantamento Requisitos de Posicionamento	1	0.39 %
Impedimento - N4 Offline	1	0.39 %

Passo 4: **Extrair métricas de complexidade**

A partir da identificação do processo de software executado foi possível extrair métricas. As métricas extraídas estão apresentadas na Tabela 17.

Tabela 17 – Métricas extraídas dos modelos mostrados nas Figuras 25 e 26

Métricas	**Modelo de processo descoberto a partir de registros não mapeados**	**Modelo de processo descoberto a partir de registros mapeados**
Número de Atividades	190	8
Frequência Máxima Atividade	2	150
extended cardoso metric	388.0	25.0
número de arcos	776.0	20.0
número de locais	384.0	20.0

Conforme pode ser observado na Tabela 17, as métricas extraídas a partir dos registros de execução do Projeto 2 também indicam que o mapeamento entre as perspectivas de processo e projeto beneficia a redução da complexidade do modelo de processo de software executado. Novamente, é possível observar que no modelo de processo descoberto a partir de registros mapeados, os valores das métricas *número de atividades*, *extended cardoso metric*, *número de locais* e são menores, na qual indica uma redução na complexidade desse modelo, quando comparado aos valores das métricas obtidas a partir de registros não mapeados. O valor da métrica *frequência máxima da atividade* foi maior, quando considerando aos registros mapeados. Conforme mencionado anteriormente, isso indica redução da complexidade, uma vez que um alto valor para esta métrica indica menor complexidade do modelo.

Na próxima seção os resultados apresentados serão discutidos.

8.2.3 Discussão dos Resultados

O estudo conduzido para avaliar os benefícios do mapeamento entre as perspectivas na identificação do processo de software foi executado de acordo com o processo de aplicação apresentado no Capítulo 6. As seguintes atividades foram executadas: *Obter Registros de Execução do Processo Mapeados*, *Pré-processar Registros de Execução do Processo*, *Identificar Processo de Software Executado*, e *Extrair Métricas de Complexidade.*

A partir da obtenção dos registros de execução foi possível identificar o processo de software executado. Ao comparar o modelo de processo descoberto a partir dos registros de execução não mapeados (Figuras 23 e 25) e o modelo de processo descoberto a partir dos registros de execução mapeados (Figuras 24 e 26) é possível observar que a representação de elementos do processo nos registros permite uma melhor compreensão da execução dos processos, uma vez que é possível identificar que as atividades do processo estão explícitas.

Quando se considera para análise do modelo de processo descoberto a partir dos registros de execução não mapeados a visão do processo está implícita, uma vez que não se tem o mapeamento entre cada tarefa com a atividade do processo de referência. Outro fator que impacta no entendimento do processo é o uso de termos conhecidos pelos profissionais envolvidos, uma vez que sem considerar o mapeamento, devido à forma como as tarefas são registradas, seria muito difícil compreender a execução sob a ótica das tarefas do projeto. Na base de dados do Projeto 1, por exemplo, as tarefas estão nomeadas da seguinte forma: "*Ajustar UC Cadastrar IMP (INF03)*, *Mudanças no Envio de Appointments de Retirada"* e "*Ajustar validação de Requisitante e Armador*" e estão relacionadas à atividade do processo nomeada "*Descrever caso de uso"*. Nessa situação, sem a aplicação da solução de mapeamento, que torna explícita a informação do processo, não seria trivial identificar a atividade do processo. Sendo assim, uma vez não estando explícitos os elementos do processo nos registros de execução, a análise da execução dos processos seria muito difícil, como pode ser observado ao comparar os modelos descobertos, nas Figuras 23 e 24 extraídos da base de dados do Projeto 1, e nas 25 e 26 extraídos da base de dados do Projeto 2.

Considerando que o ponto de partida para qualquer atividade de Mineração de Processos são os registros de execução, a qualidade do seu resultado depende diretamente da qualidade dos registros (VAN DER AALST *et al.*, 2011). Desta forma, o

mapeamento afetou diretamente a qualidade dos registros de execução, uma vez que tornou os registros gerados mais claros e estruturados, principalmente por permitir que elementos do processo estivessem explícitos.

Uma forma de avaliar o benefício no entendimento dos processos executados, de forma quantitativa, foi por meio das métricas de complexidades extraídas e apresentadas nas Tabelas 15 e 17. As métricas *números de atividades* e *frequência máxima atividade*, extraída dos registros de execução não mapeados e mapeados, em ambos os projetos demonstram a redução da complexidade. Nas bases de dados do Projeto 1 e 2 houve a redução do número de atividades. Um baixo valor para esta métrica indica menor complexidade do modelo. A métrica *frequência máxima atividade* apresentou em ambas as bases um aumento de valor, quando comparados com registros não mapeados e mapeados. Um alto valor para esta métrica indica menor complexidade do modelo. A redução do número de atividades e o aumento da frequência máxima de ocorrência das atividades são métricas que demonstram maior clareza e estruturação. Neste sentido, a representação das atividades do processo de software no plano do projeto e a definição de sua instância impõem uma estruturação através do mapeamento estabelecido.

Em relação às métricas *extended cardoso metric*, *número de arcos* e *número de locais*, em ambas as bases, observou-se a redução dos valores das métricas, ou seja, o comportamento foi o mesmo quando se comparou o modelo descoberto a partir de registros não mapeados e mapeados. Um baixo valor para estas métricas indica menor complexidade do modelo, conforme apresentado na Subseção 6.2.2.

A hipótese definida no planejamento do estudo foi "*O mapeamento entre as perspectivas de processo e projeto beneficia a redução da complexidade do modelo de processo de software executado*", e considerando os resultados obtidos dos estudos descritos, pode-se concluir que o mapeamento entre as perspectivas beneficia a redução da complexidade do modelo de processo de software executado, quando se observa os valores das métricas obtidas. Esta redução impacta na análise das atividades do processo que estão sendo executadas.

Neste sentido, o mapeamento possibilita que as atividades do processo possam ser explícitas, possibilitando o entendimento da execução, sob a ótica do processo de software. Considera-se que esse entendimento pode ser útil para apoiar os gerentes de projetos nas tomadas de decisões. Além disso, reconhece-se que outros aspectos quantitativos podem ser explorados. Por exemplo, as frequências de ocorrência das atividades (Tabela 14 e 16), permitem analisar a quantidade de esforço direcionada às atividades. Por exemplo, a base de dados do Projeto 1 tem no total

2238 registros de execução, sendo que 1553 destes estão relacionados a atividade Construir e testar código, como pode ser observado na Tabela 14.

Uma vez demonstrado que o mapeamento beneficia a redução da complexidade do modelo de processo executado, aspectos qualitativos podem ser explorados, tais como, o impacto da redução da complexidade na tomada de decisão ao longo dos projetos. Nós conjecturamos que, quando o mapeamento é aplicado, o modelo de processo obtido possa ser utilizado para prover outras informações úteis para orientar a tomada de decisões sobre oportunidades de melhoria no processo de software descoberto, tais como: (i) para identificar as atividades “guarda-chuvas”, ou seja, aquelas associadas ao controle da qualidade; (ii) atividades que apresentam um alto ou baixo nível de granularidade; (iii) atividades que precisam ser excluídas; e a (iv) relação de dependência entre as atividades.

Por exemplo, atividades com alta frequência de ocorrência podem ser analisadas como candidata a adaptação do processo de software padrão da organização, sendo divididas em atividades mais específicas. Outro exemplo seria explorar as dependências identificadas pelas transições presentes nos modelos de processo descobertos. Estudos qualitativos serão explorados em trabalhos futuros para avaliar essa conjectura.

Esta subseção apresentou a discussão dos resultados do estudo conduzido. A subseção seguinte irá discutir as ameaças à validade.

8.2.4 Ameaças à validade

No estudo conduzido as bases utilizadas foram selecionadas por serem as bases disponíveis, e também porque a representação da atividade do processo havia sido realizada na criação de cada tarefa. Neste caso, ocorre a ameaça de validade interna de efeitos de seleção devido ao modo como as bases foram selecionadas. Esta ameaça é difícil de ser contornada pela falta de disponibilidade de bases de dados de projeto, bem como a falta de tempo de profissionais para apoiar o mapeamento das mesmas.

Em relação à ameaça de validade externa, o número de projetos utilizados no experimento é pequeno e os resultados obtidos podem não ser tão representativos para demonstrar os benefícios da solução proposta nesta tese. Para atenuar esse problema, os procedimentos adotados no estudo são apresentados em detalhamento, permitindo a replicação do estudo com um número maior de bases de registros de execução.

8.3 Identificando não Conformidades

Esta seção apresenta o estudo conduzido para avaliar o benefício da aplicação da solução de mapeamento abordada por esta tese, na identificação de não conformidades. A identificação ocorre pela comparação das atividades obrigatórias previstas no modelo de processo de software definido para o projeto e a lista de atividades executadas. A partir desta comparação uma lista de não conformidades é gerada. Um relatório é gerado para análise do gerente de projeto das não conformidades identificadas.

O estudo utiliza a base de dados do projeto de desenvolvimento de software da indústria, chamado SIGA-EPCT (FABRO NETO *et al.*, 2012). Este projeto de desenvolvimento deste sistema teve início em 2008, sendo desenvolvido colaborativamente por pesquisadores das Instituições Federais de Ensino envolvidos no projeto, por meio de núcleos de pesquisa geograficamente distribuídos pelo país, utilizando tecnologias de código aberto. A autora desta tese atuou como membro da equipe de Requisitos e Modelagem neste projeto por cerca 3 anos. Em torno 130 casos de uso foram implementados por uma equipe composta 100 colaboradores. A ferramenta REDMINE foi utilizada pelo projeto SIGA-EPCT para gerenciamento e monitoramento das tarefas que são executadas no desenvolvimento de um sistema acadêmico. Nessa ferramenta, a instanciação e execução do processo ocorrem, a partir da criação e finalização de tarefas (REDMINE, 2015).

A execução do estudo tem como base as orientações para demonstração dos benefícios do mapeamento entre as perspectivas na identificação de não conformidades apresentadas no Capítulo 6.

A Subseção 8.3.1 apresenta o planejamento deste estudo. Na Subseção 8.3.2 a execução do estudo é descrita e na Subseção 8.3.3 os resultados são discutidos. A Subseção 8.3.4 discute as ameaças à validade do estudo.

8.3.1 Planejamento do Estudo

Seguindo a abordagem GQM (*Goal, Question, Metric*) (BASILI; CALDIERA e ROMBACH, 1994), o objetivo do estudo foi definido como:

Analisar as não conformidades identificadas

Com o propósito de caracterizar

Com respeito à capacidade (assertividade) de identificação de não conformidades

Do ponto de vista de engenheiros de software

No contexto de projetos de desenvolvimento de software de um sistema acadêmico desenvolvido colaborativamente por pesquisadores das Instituições Federais de Ensino

A hipótese definida é "*O mapeamento entre as perspectivas de processo e projeto beneficia a identificação de não conformidade*".

Na seção a seguir os passos da execução do estudo serão descritos em detalhes.

8.3.2 Execução do Estudo

A execução do estudo foi realizada de acordo com o processo de identificação de não conformidades apresentado no Capítulo 6. Ele é composto das seguintes atividades: *Obter Registros de Execução Mapeados*, *Obter o Modelo de Processo de Software, Identificar não Conformidades, e Gerar Relatório de não Conformidades*.

Na atividade "**Obter Registros de Execução Mapeados**" a base de execução do projeto expõe o conjunto de tarefas executada ao longo do projeto. Neste sentido a mesma é extraída de ferramentas utilizadas ao longo do projeto para gerenciamento das tarefas. Nas bases utilizadas neste estudo, os registros obtidos não estavam mapeados, sendo necessário realizar ações de mapeamento nos mesmos. Na atividade "**Obter o Modelo de Processo de Software**" o modelo de processo de software definido para o projeto é obtido, contendo a lista de atividades previstas. Na atividade "**Identificar não conformidades**" é feita a comparação entre as atividades previstas e executadas. E na atividade "**Gerar Relatório de não Conformidade**" um relatório contendo as não conformidades identificadas é gerado.

A seguir são apresentados os resultados de cada passo definido, na execução do estudo.

Passo 1: **Obter Registros de Execução Mapeados**

Para a avaliação do benefício do mapeamento na identificação de não conformidade espera-se que os registros de execução estejam mapeados. No caso deste projeto os registros de execução não estavam mapeados, então a partir da base de dados de execução obtida foi realizada o mapeamento entre as perspectivas, por meio da representação da instância do processo, bem como da atividade do processo de software definido referente de cada tarefa, de acordo com a solução proposta. Caso as operações de instanciação tivessem sido aplicadas na criação das tarefas, a estrutura resultante teria também a representação da atividade do processo de cada tarefa, bem como da instância do processo, possibilitando que tais informações

também estivessem representadas de forma explícita. A representação destas informações foi realizada com apoio de profissionais que atuavam no projeto. As seguintes ações foram executadas:

a) Representação da instância de execução do processo

Com o apoio da gerente deste projeto e baseado na experiência da autora desta tese no projeto, foi possível observar, ao analisar a nomenclatura das atividades do processo, que o processo de software de referência para o projeto é centrado em caso de uso. Neste sentido, neste projeto todo o processo é executado por caso de uso, sendo assim, a partir do nome da tarefa foi possível identificar a instância, extraindo do nome do caso de uso. A Tabela 18 apresenta um exemplo.

Tabela 18 - Definição da Instância do processo

Título da Tarefa	**Instância do Processo**
Atualizar diagrama de classe geral SIGA-EDU-CDU-EXTEN-028	SIGA-EDU-CDU-EXTEN-028

Para realizar esta definição foi criada uma nova coluna chamada *Instância do Processo*, no arquivo da base de dados obtida, a qual foi preenchida com o trecho da tarefa referente ao nome do caso de uso (informação do nome da instância do processo).

b) Definir a atividade do processo

Para se obter o mapeamento deve ser representada a informação da atividade do processo referente a cada tarefa. Neste projeto, o título da tarefa é composto pela atividade do processo mais o nome do caso de uso, sendo assim na definição da atividade do processo retirou-se a parte referente ao título do caso de uso. A Tabela 19 apresenta um exemplo.

Tabela 19 - Definição da Atividade do Processo

Título da Tarefa	**Atividade do Processo**
Projetar Tela do SIGA-EDU-CDU-EXTEN-028	Projetar Tela

Uma nova coluna foi criada na base de dados obtida, para receber a informação do nome da atividade do processo. Para casos em que a tarefa não faz referência a nenhuma atividade do processo de software de referência, o campo

atividade do processo é preenchido com o título da tarefa extraído a parte que descreve o título do caso de uso. Tabela 20 apresenta um exemplo.

Tabela 20 - Definição da Atividade do Processo sem Atividade Prevista no Modelo de Referência

Título da Tarefa	**Atividade do Processo**
Revisar diagrama de classe do SIGA-EDU-CDU-EXTEN-028	Revisar diagrama de classe

O resultado é o arquivo obtido no formato .xls, acrescido das colunas referentes a atividade do processo e instância.

Passo 2: **Obter o Modelo de Processo de Software Definido**

O projeto analisado tinha um processo de software de referência para sua execução, planejado para atender às suas necessidades, o qual é formado pelas seguintes fases: Planejamento, Levantamento de Requisitos, Especificação e Projeto, Implementação, Teste e Implantação. Para este estudo foi considerada somente a fase de Especificação e Projeto, que está apresentada na notação BPMN na Figura 26. Esta fase é composta das seguintes atividades: D*escrever Caso de Uso, Revisar Descrição de Caso de Uso, Especificar Relatório, Projetar Tela, Elaborar Diagrama de Classes, Atualizar Diagrama de Classes Geral, Revisar Especificação de Caso de Uso, Definir Casos de Testes, Elaborar Modelo Físico e Atualizar da Banco de Dados*.

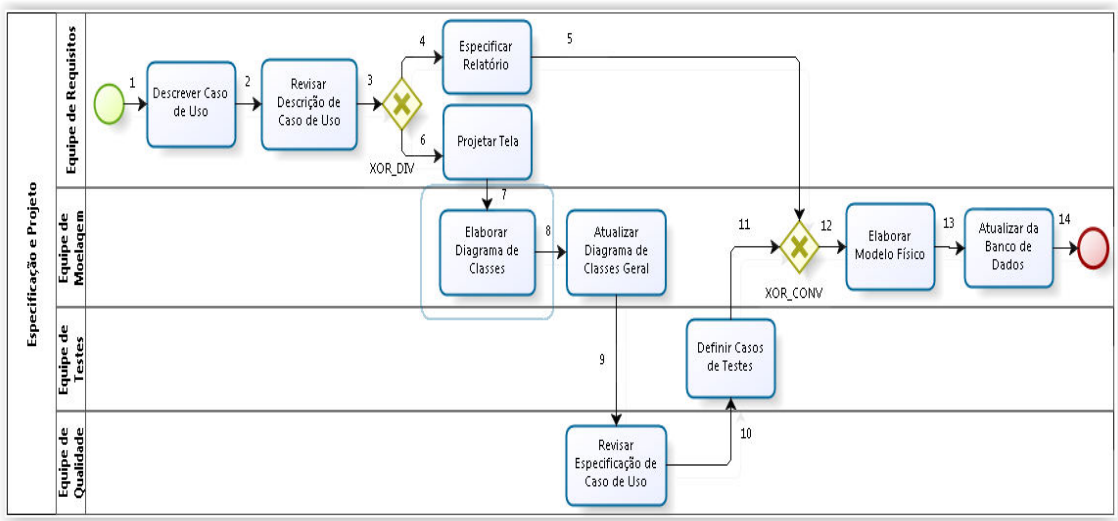


Figura 27 - Modelo de Processo Padrão da Etapa de Especificação e Projeto. Fonte: A Autora.

Ao obter o modelo de processo de software de referência se faz necessário, para a atividade **Identificar não Conformidade.** Ao obter o modelo onde se faz

necessário listar os caminhos possíveis de execução. Para isso, a solução proposta provê a geração de uma matriz de adjacência.

A matriz de adjacência apresentada na Figura 28 foi gerada pela utilização do apoio ferramental desenvolvido para suporte ao mapeamento entre as perspectivas, apresentado em detalhes no Capítulo 7. No Apêndice E é apresentado o pseudocódigo da geração da matriz de adjacência e listagem de caminhos.

```
[0,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0]
[0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0]
[0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0]
[0,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  1,  1,  0,  0,  0]
[0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0]
```

Figura 28 – Matriz de Adjacência gerada a partir do modelo mostrado na Figura 27.
Fonte: A Autora.

A partir da matriz, os seguintes caminhos foram mapeados:

- *Caminho 1*: [Início] [Descrever Caso de Uso] [Revisar Descrição de Caso de Uso] [Especificar Relatório] [Elaborar Modelo Físico] [Atualizar da Banco de Dados] [Fim]
- *Caminho 2*: [Início] [Descrever Caso de Uso] [Revisar Descrição de Caso de Uso] [Projetar Tela] [Elaborar Diagrama de Classes] [Atualizar Diagrama de Classes Geral] [Revisar Especificação de Caso de Uso] [Definir Casos de Testes] [Elaborar Modelo Físico] [Atualizar da Banco de Dados] [Fim]

Passo 3: **Identificar não conformidades**

A identificação de não conformidade foi realizada pela comparação das atividades executadas e as atividades previstas.

As atividades previstas são as seguintes organizadas por caminhos:

- *Caminho 1*: Descrever Caso de Uso, Revisar Descrição de Caso de Uso, Especificar Relatório, Elaborar Modelo Físico e Atualizar Banco de Dados.
- *Caminho 2*: Descrever Caso de Uso, Revisar Descrição de Caso de Uso, Projetar Tela, Elaborar Diagrama de Classes, Atualizar Diagrama de Classes

Geral, Revisar Especificação de Caso de Uso, Definir Casos de Testes, Elaborar Modelo Físico e Atualizar Banco de Dados.

A Tabela 21 apresenta a relação de atividades executadas. Esta relação foi extraída por meio da ferramenta Disco.

Tabela 21 - Atividades Executadas

Atividades	**Frequência**	**Frequência Relativa**
Atualizar Descrição	18	11.04 %
Atualizar Banco de Dados	18	11.04 %
Atualizar diagrama de classe geral	16	9.82 %
Definir Casos de Teste	14	8.59 %
Descrever o CDU	13	7.98 %
Projetar tela	12	7.36 %
Elaborar Modelo Físico	12	7.36 %
Elaborar diagrama de classe	8	4.91 %
Revisar especificação	7	4.29 %
Revisar Descrição	7	4.29 %
Atualizar Projeto de Tela	6	3.68 %
Alterar Especificação	4	2.45 %
Revisão do CDU	4	2.45 %
Alterar Projeto de Tela	3	1.84 %
Atualizar diagrama de classes	3	1.84 %
Refatorar Especificação	3	1.84 %
Analisar dúvida a respeito do CDU	1	0.61 %
Atualizar Banco de Dados	1	0.61 %
Elaborar mapeamento dos vínculos do CDU	1	0.61 %
Alterar casos de Teste	1	0.61 %
Atualizar Relatório	1	0.61 %
Alterar diagrama de classes	1	0.61 %
Implementar Relatório	1	0.61 %
Verificar inconsistências na especificação do CDU	1	0.61 %
Criar Massas de Testes	1	0.61 %
Alterar Regra de Negócio	1	0.61 %
Analisar sugestão do CDU	1	0.61 %
Verificar questionamento do CDU	1	0.61 %
Validar classes do CDU	1	0.61 %
Auxiliar na revisão da descrição do CDU	1	0.61 %
Revisar Projeto Tela	1	0.61 %

Com a relação de atividades previstas e atividades executadas é possível realizar a comparação por caminho e identificar as seguintes listas de não conformidade:

- **Atividades Previstas Executadas**:
  Caminho 1: Descrever Caso de Uso, Elaborar Modelo Físico, Revisar Descrição de Caso de Uso e Atualizar Banco de Dados.

  Caminho 2: Definir Casos de Testes, Descrever Caso de Uso e Projetar Tela, Elaborar Modelo Físico, Elaborar Diagrama de Classes, Revisar Descrição de Caso de Uso, Revisar Especificação de Caso de Uso e Atualizar Banco de Dados

- **Atividades Previstas não executadas**:
  Caminho 1: Especificar Relatório
  Caminho 2: Atualizar Diagrama de Classes Geral

- **Atividades não Previstas executadas**:
  Caminho 1: Atualizar Descrição, Atualizar Banco de Dados, Atualizar diagrama de classe geral, Definir Casos de Teste, Projetar tela, Elaborar diagrama de classe, Revisar especificação, Atualizar Projeto de Tela, Alterar Especificação, Revisão do CDU, Alterar Projeto de Tela, Atualizar diagrama de classes, Refatorar Especificação, Analisar dúvida a respeito do CDU, Elaborar mapeamento dos vínculos do CDU, Alterar casos de Teste, Atualizar Relatório, Alterar diagrama de classes, Implementar Relatório, Verificar inconsistências na especificação do CDU, Criar Massas de Testes, Alterar Regra de Negócio, Analisar sugestão do CDU, Verificar questionamento do CDU, Validar classes do CDU, Auxiliar na revisão da descrição do CDU e Revisar Projeto Tela.

  Caminho 2: Atualizar Descrição, Atualizar Banco de Dados, Atualizar diagrama de classe geral, Revisar Descrição, Atualizar Projeto de Tela, Alterar Especificação, Alterar Projeto de Tela, Atualizar diagrama de classes, Refatorar Especificação, Analisar dúvida a respeito do CDU, Elaborar mapeamento dos vínculos do CDU, Alterar casos de Teste, Atualizar Relatório, Alterar diagrama de classes, Implementar Relatório, Verificar inconsistências na especificação do CDU, Criar Massas de Testes, Alterar Regra de Negócio, Analisar sugestão do CDU, Verificar questionamento do CDU, Validar classes do CDU, Auxiliar na revisão da descrição do CDU e Revisar Projeto Tela.

Tabela 22 - Trecho dos registros de execução não mapeados

Tarefas	Frequência	Frequência Relativa
Atualizar descrição do SIGA-EDU-CDU-REDIA-013	6	3.68 %
Atualizar BD do SIGA-EDU-CDU-REDIA-013	5	3.07 %
Atualizar diagrama de classe geral com o SIGA-EDU-CDU-PELET-061	3	1.84 %
Atualizar descrição do SIGA-EDU-CDU-MATRI-012	3	1.84 %
Refatorar Especificação do SIGA-EDU-CDU-MATRI-012	3	1.84 %
Atualizar diagrama de classes geral do SIGA-EDU-CDU-PELET-060	3	1.84 %
Atualizar descrição do SIGA-EDU-CDU-RACAD-011	3	1.84 %
Elaborar Modelo Físico do SIGA-EDU-CDU-PELET-061	2	1.23 %
Definir Casos de Teste para SIGA-EDU-CDU-INFRA-006	2	1.23 %
Atualizar descrição do SIGA-EDU-CDU-INFRA-006	2	1.23 %
Definir Casos de Teste para SIGA-EDU-CDU-INFRA-003	2	1.23 %
Definir casos de Teste para SIGA-EDU-CDU-INFRA-040	2	1.23 %
Atualizar BD para o SIGA-EDU-CDU-MATRI-012	2	1.23 %
Revisão do SIGA-EDU-CDU-MATRI-012	2	1.23 %
Revisar Descrição do SIGA-EDU-CDU-MATRI-033	2	1.23 %
Atualizar BD do SIGA-EDU-CDU-MATRI-033	2	1.23 %
Definir Casos de Teste para SIGA-EDU-CDU-RACAD-011	2	1.23 %
Alterar Especificação SIGA-EDU-CDU-RACAD-011	2	1.23 %
Atualizar PT do SIGA-EDU-CDU-RACAD-011	2	1.23 %
Atualizar BD para o SIGA-EDU-CDU-REDIA-025	2	1.23 %
Revisar especificação do SIGA-EDU-CDU-REDIA-013	2	1.23 %

A mesma atividade de **Identificar não conformidade**, considerando os registros não mapeados, seria mais demorada e propensa a erros por não ter a informação da atividade do processo. A Tabela 22 apresenta um trecho dos registros de execução não mapeados para exemplificar a dificuldade. No total a base possui 130 registros, por questão de espaço estão omitidas na tabela as tarefas com frequência igual a 1.

Passo 4: **Gerar Relatório de não Conformidades**

Conforme apresentado no passo anterior a atividade **Identificar não conformidade** foi feita pela comparação dos das atividades executadas e as atividades previstas no modelo de processo de software de referência para o projeto e o processo de software executado.

Para representar as não conformidades identificadas foi utilizado o mecanismo *localContribution* e *Supression*, da extensão BPMNt (PILLAT; OLIVEIRA; FONSECA, 2012) para destacar as alterações de inclusão e remoção de atividades. O relatório gerado é apresentado na Tabela 23.

A partir do relatório gerado, o Gerente de Projeto pode ser comunicado das não Conformidades identificadas. Considerando a identificação após a execução, as não conformidades identificadas, podem apoiar tomadas de decisão baseada em dados do processo de software. Caso esta identificação tivesse sido realizada antes da execução, ao fim da instanciação, o gerente de projeto, poderia definir as formas de resolver as não conformidades, como por exemplo: Fazer o plano atender o processo descrito, padrão, procedimento; ou Tomar uma decisão executiva de não satisfazer o processo descrito, padrão, procedimento, caso isso seja necessário, arcando com as consequências deste ato; ou Solicitar alteração o processo descrito, padrão ou procedimento para torná-lo utilizável (eficaz).

Tabela 23 - Relatório da Identificação de não Conformidade

Ação (BPMNt)	Caminho 1	Caminho 2
LocalContribution	Atualizar Descrição, Atualizar Banco de Dados, Atualizar diagrama de classe geral, Definir Casos de Teste, Projetar tela, Elaborar diagrama de classe, Revisar especificação, Atualizar Projeto de Tela, Alterar Especificação, Revisão do CDU, Alterar Projeto de Tela, Atualizar diagrama de classes, Refatorar Especificação, Analisar dúvida a respeito do CDU, Elaborar mapeamento dos vínculos do CDU, Alterar casos de Teste, Atualizar Relatório, Alterar diagrama de classes, Implementar Relatório, Verificar inconsistências na especificação do CDU, Criar Massas de Testes, Alterar Regra de Negócio, Analisar sugestão do CDU, Verificar questionamento do CDU, Validar classes do CDU, Auxiliar na revisão da descrição do CDU e Revisar Projeto Tela	Atualizar Descrição, Atualizar Banco de Dados, Atualizar diagrama de classe geral, Revisar Descrição, Atualizar Projeto de Tela, Alterar Especificação, Alterar Projeto de Tela, Atualizar diagrama de classes, Refatorar Especificação, Analisar dúvida a respeito do CDU, Elaborar mapeamento dos vínculos do CDU, Alterar casos de Teste, Atualizar Relatório, Alterar diagrama de classes, Implementar Relatório, Verificar inconsistências na especificação do CDU, Criar Massas de Testes, Alterar Regra de Negócio, Analisar sugestão do CDU, Verificar questionamento do CDU, Validar classes do CDU, Auxiliar na revisão da descrição do CDU e Revisar Projeto Tela
Suppression	Especificar Relatório	Atualizar Diagrama de Classes Geral

Na próxima seção os resultados apresentados serão discutidos.

8.3.3 Discussão dos Resultados

O estudo conduzido para avaliar os benefícios da ligação explícita entre as perspectivas na identificação de não conformidades foi executado de acordo com o processo de aplicação apresentado no Capítulo 6. As seguintes atividades foram executadas: *Obter Registros de Execução Mapeados*, *Obter o Modelo de Processo de Software, Identificar não Conformidades, e Gerar Relatório de não Conformidades.*

O resultado da ligação estabelecida possui informações que permitem verificar a conformidade entre as perspectivas do processo e do projeto e uma melhor compreensão da execução dos processos. Portanto, é possível considerar que mapear projetos de software com processos de software permite a identificação de não-conformidades, como atividades previstas executadas, atividades previstas não executadas, atividades não previstas executadas, que podem ser confirmadas após uma análise completa da conformidade. Sendo assim as seguintes listas de não conformidades foram extraídas: atividades previstas executadas, atividades previstas não executadas e atividades não previstas executadas, conforme apresentadas no passo 3 da execução do estudo (Subseção 8.3.2)

A partir da obtenção dos registros de execução foi possível identificar as não conformidades. Ao comparar a lista de tarefas previstas no modelo de processo com a lista de atividades executadas apresentadas na Tabela 8.8 observou-se que a representação de elementos do processo nos registros permitiu que as atividades do processo estivessem explícitas, contribuindo assim para a identificação não conformidade.

A identificação de não conformidade ocorreu por meio da comparação entre a lista de atividades previstas e executadas, porém sem considerar o mapeamento esta ação pode ser demorada e propensa a erros, uma vez que não é trivial identificar a atividade correspondente de cada tarefa. Os registros de execução do projeto utilizado neste estudo seguem um padrão de nomenclatura de tarefas bem sugestiva para a identificação da atividade do processo, como por exemplo, a tarefa *Definir Casos de Teste para SIGA-EDU-CDU-INFRA-006* foi associada a atividade do processo *Definir Caso de Teste*. Porém, mesmo neste caso, considerando uma base muito grande, e a influência humana, que por algum motivo poderia não seguir o padrão de nomenclatura, a atividade de identificação de não conformidade seria impactada.

A hipótese definida no planejamento do estudo foi "*O mapeamento entre as perspectivas de processo e projeto beneficia a identificação de não conformidade*", e considerando os resultados do estudo descrito, pode-se concluir que o mapeamento

entre as perspectivas beneficia a identificação, quando se é representada a atividade do processo referente a cada tarefa do projeto.

Alguns aspectos a serem explorado em trabalhos futuros: (i) A redução da dependência dos stakeholders (avaliadores) e subjetividade ao analisar dados do projeto; (ii) O apoio às avaliações em aspectos de segurança, uma vez que a informação está explícita, os avaliadores externos podem não ter tanta necessidade de acessar diretamente a informações sigilosas das organizações; (iii) A identificação de alterações na ordem prevista de execução; e (iv) A identificação de não conformidade por meio de apoio semi-automatizado, a partir de implementação de algoritmos de verificação de conformidade.

O apoio ferramental proposto faz a identificação de não conformidade de forma semi-automatizada, a partir de implementação de algoritmo de verificação de conformidade, apresentado no G, porém não foi usado por não ter uma base populada.

Esta subseção apresentou a discussão dos resultados do estudo conduzido. A subseção seguinte irá discutir as ameaças à validade.

8.3.4 Ameaças à validade

A principal ameaça à validade desse estudo refere-se à manipulação da base de dados obtida, para aplicar o mapeamento entre as perspectivas.

Ameaças relacionadas à validade interna: (i) ameaça de *Instrumentação,* manipulando o objeto de análise (tarefas) devido ao estágio de mapeamento do mesmo; e (ii) ameaça de *Efeitos da Seleção*, porque não foi definido um critério para selecionar o profissional, que apoiou a identificação da atividade de processo correspondente de cada tarefa. A escolha foi feita pela disponibilidade e pelo conhecimento do processo do software, mas outros fatores, como o tempo no projeto, poderiam ter sido considerados. Também vale ressaltar a ameaça à validade de *Conclusão*, pois a não seleção do profissional mais experiente no projeto pode impactar no mapeamento entre tarefa e atividade, causando diferentes variações nas taxas extraídas. Em relação à ameaça de *Validade Externa*, o número de projetos utilizados no experimento é pequeno e os resultados obtidos podem ser uma função desse número. Para atenuar esse problema, o estudo pode ser replicado, considerando um número maior de dados. Por fim, destacamos a ameaça *Interação entre seleção e tratamento*, porque não foi apllicada a seleção aleatória de profissionais que apoiaram o mapeamento, pois a escolha foi por conveniência.

Na seção seguinte são apresentadas as considerações finais desse capítulo

8.4 Considerações Finais

Este capítulo apresentou os estudos conduzidos como forma de avaliar os benefícios do mapeamento entre as perspectivas na identificação do processo de software executado e de não conformidades.

Três bases de dados de projetos de desenvolvimento de software de duas empresas foram usadas nos estudos foram conduzidos. Para a identificação do processo de software executado foram utilizadas duas bases de dados, e para a identificação de não conformidades foi utilizada uma base de dados.

Por meio dos estudos foi possível concluir que a aplicação da solução de mapeamento proposta nesta tese permite um melhor entendimento da execução dos processos nos projetos, uma vez que os benefícios na identificação do processo de software executado e de não conformidades foram observados.

Embora os estudos conduzidos tenham mostrado a viabilidade e benefícios da solução proposta nesta tese para apoiar o mapeamento entre as perspectivas de processo e projeto, outros estudos precisam ser conduzidos para reforçar as evidências obtidas.

9 Conclusão e Trabalhos Futuros

Este capítulo apresenta as contribuições e os resultados da tese, incluindo perspectivas futuras e considerações finais.

9.1 Epílogo

O setor de desenvolvimento de software está amplamente focado no fornecimento de sistemas de software modernos, confiáveis e altamente responsivos, visando atender à crescente demanda dos clientes. Os sistemas de software variam de sistemas de informação complexos (por exemplo, sistemas financeiros) a jogos interativos, aplicativos de assistência médica ou aplicativos móveis. Processo de Software são considerados atores importantes para a indústria de desenvolvimento de software, pois orquestram atividades, pessoas e informações envolvidas no desenvolvimento de software. Eles são usados como base para o planejamento e monitoramento de projetos de software, bem como para orientar as equipes de desenvolvimento durante a execução desses projetos. Porém, as especificações de processo de software definidas para projetos podem ser alteradas em tempo de execução, por exemplo, suprimindo atividades previstas, adicionando atividades imprevistas ou modificando a sequência destas atividades.

Embora as organizações de desenvolvimento de software busquem continuamente melhorar seus processos de desenvolvimento e manutenção de software, uma vez que estas estão diretamente relacionadas à qualidade dos produtos de software resultantes, os estudos exploratórios apresentados no Capítulo 3 (pesquisa de opinião com profissionais, a análise de projetos de desenvolvimento de software relacionados à indústria e a revisão da literatura) nos levaram ao entendimento que:

i) os desenvolvedores têm dificuldade de perceber os elementos de processo ao longo da execução dos projetos;

ii) as ferramentas utilizadas ao longo dos projetos não suportam o registro de elementos de processo necessários para sua identificação, como a atividade do processo e o fluxo de controle das atividades; e

iii) os processos de software não são explicitamente identificados em planos de projeto.

Em nossa análise, estes fatores dificultam o mapeamento entre as perspectivas de processo e projeto de software.

Na revisão da literatura realizada (Capítulo 4) foi possível identificar estratégias de mapeamento entre as perspectivas de processo e projeto de software. Porém as estratégias identificadas realizam o mapeamento após a execução. Portanto, os benefícios do mapeamento podem ser usados apenas em novos projetos ou em iterações subsequentes do mesmo projeto. Cabe ressaltar que não foram identificados trabalhos que abordem estratégias de mapeamento aplicadas a processos de software, todos os trabalhos eram da área de processos de negócio.

Considerando o problema observado e o resultado da revisão da literatura, foi definida a solução para mapeamento entre as perspectivas de processo e projeto de software (Capítulo 5). O cerne da solução são as operações de instanciação propostas, que implementam mecanismos que permitem que elementos do processo de software definido estejam explícitos durante a sua instanciação e execução, por meio da representação dos mesmos no plano de projeto. As operações de instanciação podem ser usadas ao criar o plano do projeto e podem capturar mapeamentos simples, como a ligação (*link*) entre dois elementos, ou mapeamentos complexos envolvendo vários elementos, como agrupamento (*merge*) ou divisão (*split*).

O resultado do mapeamento estabelecido possui informações que permitem verificar a conformidade entre as perspectivas do processo e do projeto e um melhor entendimento da execução dos processos. Neste sentido, foi definido um conjunto de orientações (Capítulo 6) para demonstrar o benefício da solução de mapeamento proposta na identificação do processo de software executado e na identificação de não conformidades. Baseado nestas orientações dois estudos foram conduzido considerando bases de dados de projetos de desenvolvimento de software obtidas de empresas de desenvolvimento de software (Capítulo 7).

O primeiro estudo foi relacionado aos benefícios do mapeamento na identificação do processo de software executado, onde foi possível observar que, por meio dos modelos de processo descobertos e das métricas de complexidade obtidas, o mapeamento entre as perspectivas contribuiu para a redução da complexidade do modelo de processo de software executado. O segundo estudo foi relacionado aos benefícios do mapeamento entre as perspectivas na identificação de não conformidades, onde foi possível concluir que, quando se é estabelecido o mapeamento de forma explícita entre as perspectivas de processo e projeto, pela representação da atividade do processo referente a cada tarefa do projeto, a identificação de não conformidade é beneficiada por esse mapeamento. A identificação de não conformidades realizada pela comparação entre os registros de execução mapeados e modelo de processo de referência para o projeto, pode ser

menos demorada e menos propensa a erros, uma vez que os elementos do processo estão explícitos.

Uma ferramenta foi desenvolvida para prover apoio semi-automatizado ao mapeamento entre as perspectivas de processo e projeto, considerando as etapas da solução desenvolvida, conforme apresentado no Capítulo 7. Suas principais funcionalidades são: (i) importação do modelo de processo do software de referência; (ii) cadastro do projeto do software e sua associação ao processo do software; (iii) criação do plano do projeto por meio das operações de instanciação (cadastro de tarefas); (iv) exportação da tarefas criadas no formato xes, lido pelas ferramentas de mineração de processos; e (v) geração de relatórios de não-conformidades entre o processo e o projeto de software, apresentando para cada caminho permitido no modelo de processo de software:as atividades previstas executadas, atividades previstas não executadas e atividades não previstas executadas.

Em resumo, esta tese:

(i) evidenciou o problema da falta de mapeamento entre as perspectivas de processo e projeto de software;

(ii) definiu uma solução para o mapeamento entre representações de processo e projeto de software, de forma a permitir que o mapeamento explícito seja estabelecido;

(iii) formalizou um conjunto de orientações para demonstrar os benefícios do mapeamento na identificação do processo de software executado e na identificação de não conformidades; e

(iv) forneceu evidências iniciais dos benefícios do mapeamento entre as perspectivas.

9.2 Contribuições e Resultados

A pesquisa e os estudos descritos nesta tese têm as seguintes contribuições:

- Consolidação de evidências da falta de mapeamento entre as perspectivas de processo e projeto por meio dos estudos exploratórios realizados.
- Revisão da literatura, por meio de uma busca estruturada, provendo informações relevantes sobre o mapeamento entre o processo de software e o projeto de software.
- Elaboração de orientações para apoiar o mapeamento entre as perspectivas de processo e projeto no contexto de projetos de software.
- Definição de um conjunto de Operações de Instanciação para explicitar nos planos de projeto os elementos do modelo de processo de software.

- Desenvolvimento de uma ferramenta que pode ser usada para apoiar o mapeamento entre as perspectivas, considerando as etapas de definição (importação do modelo de processo em BPMN), instanciação (criação de tarefas) e execução (finalização de tarefas) do projeto.
- Elaboração de orientações para demonstração do benefício do mapeamento na identificação do processo de software executado e na identificação de não conformidades.
- Avaliação dos benefícios do mapeamento na identificação do processo de software executado e na identificação de não conformidades, utilizando bases de dados de projetos executados por empresas de desenvolvimento de software.

A problemática evidenciada nesta pesquisa foi que os processos de software não são explicitamente identificados em planos de projeto e, portanto o alinhamento entre eles não é trivial. Neste sentido, ao longo da pesquisa de doutorado buscou-se responder a seguinte questão: *Como explicitar o mapeamento entre as perspectivas de processo e de projeto de software?*. Foi possível observar que as orientações de mapeamento elaboradas e as operações de instanciação definidas estabelecem este mapeamento explícito entre as perspectivas e fornecem uma assistência adequada quando um processo de software definido é instanciado para um projeto de software. Assim, permitindo que os conceitos de processos não sejam perdidos nesta transição, principalmente considerando que durante a instanciação do processo, as atividades podem ser divididas, combinadas, removidas e adicionadas na forma de tarefas.

Por meio dos resultados obtidos nos estudos conduzidos ao avaliar os benefícios do mapeamento entre as perspectivas de processo e projeto, em apoiar a identificação do processo de software executado e a identificação de não conformidade, foi possível observar que o mapeamento beneficiou a redução da complexidade dos modelos de processo dos processos executados e também a identificação de não conformidade. Sendo assim, pode-se concluir, pelos benefícios observados, que a utilização das orientações definidas e as operações de instanciação propostas facilitam o mapeamento entre as representações de processo e projeto de software.

Outras contribuições referem-se às publicações realizadas ao longo do desenvolvimento desta pesquisa::

- SANTOS, R. M. S.; OLIVEIRA, T. C.; BRITO E ABREU, F. (2015). Mining software development process variations. In **Proceedings of the 30th Annual ACM Symposium on Applied Computing** (pp. 1657-1660). ACM.

- SANTOS, R. M. S.; OLIVEIRA, T. C. (2015). Descoberta de Conhecimento durante a Execução de Projetos de Desenvolvimento de Software para Apoiar o Alinhamento de Processos. In: **V Workshop de Teses e Dissertação de Software do CBSOFT (WTDSOFT 2015**). Belo Horizonte, Minas Gerais, Brasil.
- SANTOS, R. M. S.; OLIVEIRA, T. C. (2016) Identificação de Processos de Software a partir de Registros de Execução apoiada por Operações de Instanciação. In: **Anais do Simpósio Brasileiro de Qualidade de Software (WTDQS 2016)**, Maceió, Alagoas, Brasil.
- SANTOS, R. M. S. (2018) Reconciling Software Projects to Software Processes. In Sara Eloy, Manuel Alberto M. Ferreira, Maria João Oliveira (Ed.), ISTAR-IUL Winter School 2018 Applied Transdisciplinary Research. (pp.36). Lisboa: Information Sciences, Technologies and Architecture Research Center (ISTAR-IUL).
- SANTOS, R. M. S.; OLIVEIRA, T. C. (2019, Junho) Operações de Instanciação: Mecanismo de Rastreamento entre Processos de Software e Projetos de Software. Relatório Técnico ES-759/19, COPPE/UFRJ, Rio de Janeiro, Brasil. Disponível em: http://www.cos.ufrj.br.
- PILLAT, R. M.; SANTOS, R. M. S.; OLIVEIRA, T. C. (2019) Systematic Literature Review on BPMN-based Process Adaptation Approaches. In: **Proceedings of the XV Brazilian Symposium on Information Systems (SBSI'19)**, Aracaju, Sergipe, Brasil. p. 1.

Ao longo da pesquisa de tese foi possível atuar, em apoio ao orientador desta tese, nas disciplinas de Qualidade de Software, no curso de Engenharia de Sistemas e Computação, na UFRJ. Na atuação como docente relacionada à pesquisa da tese, cabe citar a orientação do seguinte trabalho de conclusão de curso:

- Aniely Martins da Silva, Leonardo Maciel e Luiz Mauro Dias. PREPRO: FERRAMENTA PARA PRÉ-PROCESSAMENTO DE DADOS DE PROCESSO DE SOFTWARE. 2016. Trabalho de Conclusão de Curso. (Graduação em Sistemas de Informação) - Instituto Federal Fluminense. Orientador: Renata Mesquita da Silva Santos.

Além das atividades e publicações acima mencionadas, no período do doutorado a autora desta tese submeteu, a editais internos do Instituto Federal

Fluminense[7], projetos de pesquisa, os quais foram aprovados, sendo concedido um aluno bolsista para cada projeto, por ano. Os projetos são apresentados na Tabela 24, sendo descritos os editais que destacam que desde 2015 houve a condução de projetos de pesquisa relacionados à pesquisa da tese.

Tabela 24 – Projetos de Pesquisa aprovados em Edital

Título do Projeto	**Edital**
Descoberta de conhecimento em base de dados de processo de desenvolvimento de software como ferramenta de apoio aos gerentes de projeto e gerentes de processo	Edital N.º 49/2015 Edital N.º 96/2016 Edital N.º 39/2017
Apoio à garantia da qualidade de processo a partir de cenários de execução de processos de desenvolvimento de software	Edital N.º 39/2017 Edital N.º 51/2018
Análise de conformidade de processos de software apoiada por técnicas de mineração de processos	Edital N.º 51/2018

9.3 Limitações

As limitações foram discutidas ao longo dos capítulos, especificamente quando as ameaças à validade dos estudos foram apresentadas. Cabe ressaltar, a limitação relacionada à dificuldade de realização de estudos experimentais para avaliar a usabilidade e aplicabilidade do recurso de criação do plano do projeto, por meio de operações de instanciação proposta, em projetos de desenvolvimento de software na indústria, bem como a realização de estudos na indústria para avaliar o apoio ferramental desenvolvido.

9.4 Trabalhos Futuros

As operações de instanciação podem facilitar a identificação de não-conformidades, uma vez que a atividade do processo é explicitamente referenciada no plano do projeto, mas o impacto na verificação de conformidades precisam ser exploradas em trabalhos futuros, por meio de estudos experimentais. Pretende-se também evoluir a solução proposta implementando a identificação de não conformidade relacionada a alterações na ordem de execução. A identificação relacionada à sequência deve ser realizada através de uma comparação entre as

[7]www.iff.edu.br

atividades definidas no processo de software definido para o projeto e as atividades que forem identificadas como executadas.

Também pretendemos prover apoio a análise de oportunidades de melhoria identificadas ao longo da execução dos projetos. Além disso, pretendemos conduzir estudos experimentais para observar outras vantagens, como: atualização do processo em tempo de execução (*on-the-fly*), melhoria de processo e gerenciamento de projeto baseado em processos.

Uma outra perspectiva futura, pretende-se formalizar as operações que estabelecem o vínculo entre *Role* e *Actor* e *Artifact* e *TaskArtifact*, para atender às outras operações já estabelecidas entre Activity (processo) e Task (projeto). Para complementar, de forma mais precisa e verificável, pretende-se formalizar as atuais representações do mapeamento em pseudocódigo, com recurso de transformações de metamodelo a metamodelo. Estas tranformações deverão ser baseadas nos metamodelos das perspectivas de processo e projeto.

Por fim, pretende-se realizar um estudo na indústria, com o objetivo de analisar a eficiência e adequação do mapeamento entre as perspectivas de processo e projeto, por meio das operações de instanciações, considerando a metodologia do Modelo de Aceitação de Tecnologia (TAM).

De forma geral, considera-se que mais estudos experimentais devem ser elaborados e executados para verificar as limitações e reforçar os indícios e evidências a cerca da solução proposta, e assim apoiar a evolução da solução.

9.5 Considerações Finais

Esta tese teve como objetivo explicitar o mapeamento entre as perspectivas de processo e projeto de software. Por meio de estudos experimentais, foi possível observar evidências que os processos de software não são explicitamente identificados em planos de projetos e/ou nos registros de execução das tarefas. Neste sentido um conjunto de operações de instanciação foi proposto, para estabelecer o mapeamento explícito entre os elementos utilizados para modelar o processo de software e os elementos usados para executar os projetos de software.

Com a aplicação da solução proposta, as tarefas do projeto, e por consequência, os registros de execução, passam a possuir informações explícitas do processo de software. Neste sentido, o acesso a dados do processo a partir das tarefas de projeto pode possibilitar aos stakeholders do projeto, a tomar decisões sobre ações de implantação e/ou melhorias nos modelos de processo de software utilizado como referência nas organizações, contribuindo assim para a maturidade do processo.

Referências Bibliográficas

ADRIANSYAH, A.; BUIJS, J. C. “Mining process performance from event logs”. In: **International Conference on Business Process Management**. p. 217-218, Springer, Berlin, Heidelberg, 2012.

ADRIANSYAH, A.; MUNOZ-GAMA, J.; CARMONA, J.; VAN DONGEN, B. F.; VAN DER AALST, W. M. “Alignment based precision checking”. In: **International Conference on Business Process Management.** p. 137-149, Springer, Berlin, Heidelberg, 2012.

BAIER, T.; DI CICCIO, C.; MENDLING, J.; WESKE, M. “Matching of events and activities-an approach using declarative modeling constraints”. In: **Enterprise, Business-Process and Information Systems Modeling**, p. 119-134, Springer, Cham, 2015.

BAIER, T.; ROGGE-SOLTI, A.; MENDLING, J.; WESKE, M. “Matching of events and activities: an approach based on behavioral constraint satisfaction”. In: **Proceedings of the 30th Annual ACM Symposium on Applied Computing**. p. 1225-1230, Salamanca; Spain 2015.

BAIER, T.; ROGGE-SOLTI, A.; WESKE, M.; MENDLING, J. “Matching of events and activities-an approach based on constraint satisfaction”. In: **IFIP Working Conference on The Practice of Enterprise Modeling**., p. 58-72, Springer, Berlin, Heidelberg, 2014.

BAIER, T.; DI CICCIO, C.; MENDLING, J.; WESKE, M. “Matching events and activities by integrating behavioral aspects and label analysis”. **Software & Systems Modeling**, v. 17, n. 2, p. 573-598, Springer, 2018.

BAIER, T.; DI CICCIO, C.; MENDLING, J.; WESKE, M. “Matching events and activities by integrating behavioral aspects and label analysis”. **Software & Systems Modeling**, v. 17, n. 2, p. 573-598, Springer, 2018.

BANDINELLI, S.; DI NITTO; E.; FUGGETTA, A. “Supporting cooperation in the SPADE-1 environment”. In: **IEEE Transactions on software engineering, v. 22, n. 12, p. 841-865, 1996**.

BASILI, V.R.; CALDIERA, G.; ROMBACH, D. “The Experience Factory”, **Encyclopedia of Software Engineering.** v. 1, p. 469-476, 1994.

BASTARRICA, M. C.; PEROVICH, D.; MARÍN, J.; RIOSECO, L. “Process-based project management and SPI”. In: **Proceedings of the 2017 International Conference on Software and System Process**. p. 124-133, Paris, France, jul. 2017.

BENDRAOU, R., SILVA, M. A., GERVAIS, M. P. e BLANC, X. “Support for Deviation Detections in the Context of Multi-Viewpoint-Based Development Processes”. In: **CAiSE Forum 2012**. p. 23-31, Paris, France, 2012.

BENDRAOU, R.; SADOVYKH, A.; GERVAIS, M. P.; BLANC, X. “Software process modeling and execution: the UML4SPM to WS-BPEL approach”. In: **33rd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO 2007)**. IEEE, p. 314-321, Lubeck, 2007.

BERNSTEIN, P. A.; DAYAL, U. “An overview of repository technology”. In: **VLDB**. p. 705-713, 1994.

BERTOLLO, G.; SEGRINI, B.; FALBO, R. A. “Definição de Processos de Software em um Ambiente de Desenvolvimento de Software Baseado em Ontologias”. **V Simpósio Brasileiro de Qualidade de Software, SBQS**, v. 6, p. 72-86, Vitória, ES, 2006.

BRINGUENTE, A. C. D. O.; FALBO, R. D. A; GUIZZARDI, G. “Using a foundational ontology for reengineering a software process ontology”. **Journal of Information and Data Management**, v. 2, n. 3, p. 511-511, 2011.

BURATTIN, A. “Obstacles to applying process mining in practice”. In: **Process Mining Techniques in Business Environments**. p. 59-63, Springer, Cham, 2015.

CAMBRIDGE DICTIONARY online. Disponível em: http://dictionary.cambridge.org/pt/dicionario/ingles/process Acesso em: 15 set. 2015.

CAMPOS, A. L. N.; OLIVEIRA, T. C. “Modeling Work Processes and Software Development-Notation and Tool”. In: **ICEIS**. p. 337-343, 2011.

CARDOSO, J. “Business process control-flow complexity: Metric, evaluation, and validation”. **International Journal of Web Services Research (IJWSR)**, v. 5, n. 2, p. 49-76, 2008.

CARDOSO, J.; MENDLING, J.; NEUMANN, G.; REIJERS, H. A. “A discourse on complexity of process models”. In: **International Conference on Business Process Management**. p. 117-128, Springer, Berlin, Heidelberg, 2006.

CATALDO, M.; NAMBIAR, S. “On the relationship between process maturity and geographic distribution: an empirical analysis of their impact on software quality”. In: **Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering**. p. 101-110, ago. 2009.

CERDEIRAL, C. T.; SANTOS, G. “Software project management in high maturity: A systematic literature mapping”. **Journal of Systems and Software**, v. 148, p. 56-87, 2019.

CHAGHROUCHNI, T.; KABBAJ, I. M.; BAKKOURY, Z. “Towards dynamic adaptation of the software process”. In: **2014 9th International Conference on Intelligent Systems: Theories and Applications (SITA-14)**. p. 1-7, IEEE, 2014.

CHEN, N.; HOI, S. C.; XIAO, X. “Software process evaluation: A machine learning approach”. In: **2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)**. p. 333-342, IEEE, 2011.

CHOU, S. C.; LI, C. W. “A Knowledge-based PSEE with the Ability of Project Monitoring”. **International Journal of Information Engineering and Electronic Business**, v. 6, n. 4, p. 1, 2014.

CMMI PRODUCT TEAM, 2010, CMMI® for Development (CMMI-DEV), Version 1.3. Disponível em: http://cmmiinstitute.com/cmmi-models. Acesso em: jul. 2017.

CUADRADO-GARCIA, J. L.; CUADRADO-GALLEGO, J. J.; HERRANZ-MARTINEZ, M. A.; RODRIGUEZ-SORIA, P. “Improve tracking in the software development projects”. In: **2011 Joint Conference of the 21st International Workshop on Software Measurement and the 6th International Conference on Software Process and Product Measurement**. p. 215-220, IEEE, 2011.

CUGOLA, G.; GHEZZI, C. “Software Processes: a Retrospective and a Path to the Future”. **Software Process: Improvement and Practice**, v. 4, n. 3, p. 1, 1998.

CUGOLA, G.; NITTO, E. D.; GHEZZI, C; MANTIONE, M. “How to deal with deviations during process model enactment”. In: **1995 17th International Conference on Software Engineering**. p. 265-265, IEEE, 1995.

CUGOLA, G. “Tolerating deviations in process support systems via flexible enactment of process models”. **IEEE Transactions on Software Engineering**, v. 24, n. 11, p. 982-1001, 1998.

DERNIAME, J. C.; KABA, B. A.; WASTELL, D. (Ed.). **Software Process: Principles, Methodology, and Technology**. Berlin, Heifelber; New York; Barcelona; Hong Kong; London; Milan; Paris; Singapore; Tokyo: Springer, 1999.

DIEBOLD, P.; SCHERR, S. A. “Software process models vs descriptions: What do practitioners use and need?”. **Journal of Software: Evolution and Process**, v. 29, n. 11, p. 1879, 2017.

DOWSON, M.; FERNSTRÖM, C. “Towards requirements for enactment mechanisms”. In: **European Workshop on Software Process Technology**. p. 90-106, Berlin, Heidelberg: Springer, 1994.

DUMAS, M.; LA ROSA, M.; MENDLING, J.; REIJERS, H. A. **Fundamentals of business process management**. v. 1, p. 2. Heidelberg: Springer, 2013

EMAM, K. E.; MELO, W.; DROUIN, J. N. **SPICE: the theory and practice of software process improvement and capability determination**. IEEE Computer Society Press, 1997.

ERIKSSON, H. E.; PENKER, M. Business modeling with UML. **New York**, p. 1-12, 2000.

FABRO NETO, A.; TURCHETTI, R. C.; TROIS, C.; PRIENSNITZ FILHO, W. "Avaliação Qualitativa e Quantitativa entre Ferramentas para Detecção de Vulnerabilidades em Código Fonte para Aplicações Web". In: **Proceeding of 7ª Conferência Ibérica de Sistemas e Tecnologias de Informação**, Madrid, 2012.

FALBO, R. D. A. "A Experiência na Definição de um Processo Padrão Baseado no Processo Unificado". In: **Anais do II Simpósio Internacional de Melhoria de Processo de Software, SIMPROS**, v. 200, pp. 63-74, São Paulo, 2000.

FALBO, R. D. A.; BERTOLLO, G. "A software process ontology as a common vocabulary about software processes". **International Journal of Business Process Integration and Management**, v. 4, n. 4, p. 239-250, 2009.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. "Knowledge discovery and data mining: towards a unifying framework". In: **Proceedings of the Second International Conference on Knowledge Discovery and Data Mining**. p. 82-88, United States 1996.

FEILER, P. H.; HUMPHREY, W. S. "Software process development and enactment: Concepts and definitions". In: **Proceedings of the Second International Conference on the Software Process-Continuous Software Process Improvement**. p. 28-40, IEEE, 1993.

FERREIRA, D. R.; SZIMANSKI, F.; RALHA, C. G. "Mining the low-level behaviour of agents in high-level business processes". **International Journal of Business Process Integration and Management**, v. 6, n. 2, p. 146-166, 2013.

FERREIRA, D. R.; SZIMANSKI, F.; RALHA, C. G.. "Improving process models by mining mappings of low-level events to high-level activities". **Journal of Intelligent Information Systems**, v. 43, n. 2, p. 379-407, 2014.

FLUXICON. Process Mining and Automated Process Discovery Software for Professionals - Disco. 2019. Disponivel em: https://fluxicon.com/disco/. Acesso em: 10 maio 2019.

FRAPPIER, M.; RICHARD, M. "SMP: a process-driven approach to project management". In: **Proceedings of the 37th Annual Hawaii International Conference on on System Sciences.** p. 9 pp, IEEE, 2004.

FRIEDRICH, J.; BERGNER, K. “Formally founded, plan-based enactment of software development processes”. In: **Proceedings of the 2011 International Conference on Software and Systems Process**. p. 199-203, USA, may. 2011.

FUGGETTA, A. “Software process: a roadmap”. In: **Proceedings of the Conference on the Future of Software Engineering**. p. 25-34, 2000.

FUGGETTA, A.; DI NITTO, E. “Software process”. In: **Proceedings of the on Future of Software Engineering**. p. 1-12, 2014.

GALHARDAS, H.; FLORESCU, D.; SHASHA, D.; SIMON, E. “AJAX: An extensible data cleaning tool”. **ACM Sigmod Record**, v. 29, n. 2, p. 590, may. 2000.

GAMMA, E., HELM, R., JOHNSON, R., & VLISSIDES, J. **Padrões de projeto**. Porto Alegre: Bookman, 2000.

GOLDSCHMIDT R.; PASSOS, E. **Data Mining**: um guia prático**.** Rio de Janeiro: Editora Campus, 2005.

GRECO, G.; GUZZO, A.; PONTIERI, L. “Mining taxonomies of process models”. **Data & Knowledge Engineering**, v. 67, n. 1, p. 74-102, 2008.

GÜNTHER, C. W.; ROZINAT, A. Disco: Discover Your Processes. **BPM (Demos)**, v. 940, p. 40-44, 2012.

GUPTA, M.; SUREKA, A. “Nirikshan: Mining bug report history for discovering process maps, inefficiencies and inconsistencies”. In: **Proceedings of the 7th India Software Engineering Conference**. p. 1-10, 2014.

GUPTA, M.; SUREKA, A.; PADMANABHUNI, S. “Process mining multiple repositories for software defect resolution from control and organizational perspective”. In: **Proceedings of the 11th Working Conference on Mining Software Repositories**. p. 122-131, 2014.

HAN, Jiawei; PEI, Jian; KAMBER, Micheline. **Data mining**: concepts and techniques. Amsterdã: Elsevier, 2011.

HE, X.; GUO, J.; WANG, Y.; GUO, Y. “An automatic compliance checking approach for software processes”. In: **Proceeding of 16th Asia-Pacific Software Engineering Conference**. p. 467-474, IEEE, 2009.

HUMMEL, O.; HEINRICH, R. "Towards Automated Software Project Planning - Extending Palladio for the Simulation of Software Processes", In**: Symposium on Software Performance**, pp. 20-29, 2013.

HUO, M.; ZHANG, H.; JEFFERY, R. “An exploratory study of process enactment as input to software process improvement”. In: **Proceedings of the International Workshop on Software Quality**. p. 39-44, 2006.

ISO/IEC, International Organization for Standardization and International Electrotechnical Commission, 2008, “ISO/IEC 12207: Systems and Software Engineering Software Life Cycle Process”, Geneve: ISO, 2008.

ISO/IEC-9000: Quality management systems-Fundamentals and vocabulary, Switzerland: ISO 2005.

KAGDI, H., COLLARD, M. L.; MALETIC, J. I. “A survey and taxonomy of approaches for mining software repositories in the context of software evolution”. **Journal of software maintenance and evolution: Research and practice**, v. 19, n. 2, p. 77-131, 2007.

KALB, G. E. “Counting lines of code, confusions, conclusions, and recommendations”. In: **Briefing to the 3rd Annual REVIC User’s Group Conference**. 1990.

KALUS, G; KUHRMANN, M. “Criteria for software process tailoring: a systematic review”. In: **Proceedings of the International Conference on Software and System Process**. p. 171-180, 2013.

KILLISPERGER, P.; STUMPTNER, M.; PETERS, G.; GROSSMANN, G.; STÜCKL, T. “A framework for the flexible instantiation of large-scale software process tailoring”. In: **International Conference on Software Process**. p. 100-111, Berlin, Heidelberg: Springer, 2010.

KILLISPERGER, P.; STUMPTNER, M.; PETERS, G.; STÜCKL, T. “Priority-Based Constraint Management in Software Process Instantiation”. In: **Information Systems Development**. p. 523-534, Springer, New York, NY: 2011.

KRISHNAN, M. S.; KRIEBEL, C. H.; KEKRE, S.; MUKHOPADHYAY, T. “An empirical analysis of productivity and quality in software products”. **Management science**, v. 46, n. 6, p. 745-759, 2000.

LARMAN, C. Applying UML and patterns: an introduction to object-oriented analysis and design and iterative development. India: Pearson Education, 2005.

LASSEN, K. B.; VAN DER AALST, W. M. “Complexity metrics for Workflow nets”. **Information and Software Technology**, v. 51, n. 3, p. 610-626, 2009.

LAURENT, Y.; BENDRAOU, R.; BAARIR, S.; GERVAIS, M. P. “Alloy4spv: A formal framework for software process verification”. In**: European Conference on Modelling Foundations and Applications**. p. 83-100, Cham: Springer, 2014.

LEMOS, A. M.; SABINO, C. C.; LIMA, R. M.; OLIVEIRA, C. A. “Using process mining in software development process management: A case study”. In: **IEEE International Conference on Systems, Man, and Cybernetics**. p. 1181-1186, IEEE, 2011.

LEMOS, A. M.; SABINO, C. C.; LIMA, R. M. F. e OLIVEIRA, C. A. “Conformance Checking of Software Development Processes Through Process Mining”. In: **Seke**. p. 654-659, 2011.

LEONI, M. “From Low-Level Events to Activities--A Session-Based Approach (Extended Version)”. **arXiv preprint arXiv:1903.03993**, 2019.

LEONI, M.; MAGGI, F. M.; VAN DER AALST, W. M. “Aligning event logs and declarative process models for conformance checking”. In: **International Conference on Business Process Management**. p. 82-97, Berlin, Heidelberg: Springer, 2012.

LEONI, M.; MARRELLA, A. “Aligning real process executions and prescriptive process models through automated planning”. **Expert Systems with Applications**, v. 82, p. 162-183, 2017.

LEONI, M.; VAN DER AALST, W. M. “Aligning event logs and process models for multi-perspective conformance checking: An approach based on integer linear programming”. In: **Business Process Management**. p. 113-129, Berlin, Heidelberg: Springer, 2013.

LEONI, M.; VAN DER AALST, W. M.; VAN DONGEN, B. F. “Data-and resource-aware conformance checking of business processes”. In: **International Conference on Business Information Systems**. p. 48-59, Berlin, Heidelberg: Springer, 2012.

LIMA, A.; FRANÇA, B.; SCHLEBBE, H.; SILVA, M.; REIS, R. Q.; REIS, C. L. “WebAPSEE: Um Ambiente Livre e Flexível Para Gerência de Processos de Software”. In: **VII Workshop de Software Livre. Porto Alegre: Abril**. 2006.

LIMESURVEY. Disponível em: https://www.limesurvey.org/. Acesso em: 15 jul. 2015.

MANNHARDT, F., LEONI, M., REIJERS, H. A., VAN DER AALST, W. M., & TOUSSAINT, P. J. “From low-level events to activities-a pattern-based approach”. In: **International conference on business process management**. Springer, Cham, 2016. p. 125-141.

MARTÍNEZ-RUIZ, T.; MÜNCH, J.; GARCÍA, F.; PIATTINI, M. “Requirements and constructors for tailoring software processes: a systematic literature review”. **Software Quality Journal**, v. 20, n. 1, p. 229-260, 2012.

MARTINS, P. V.; DA SILVA, A. R. “Process and project alignment methodology: A case-study based analysis”. **Computer Science and Information Systems**, v. 13, n. 3, p. 901-925, 2016.

MCCABE, T. J. “A complexity measure”. **IEEE Transactions on software Engineering**, n. 4, p. 308-320, 1976.

MCCABE, T. J.; BUTLER, C. W. “Design complexity measurement and testing”. **Communications of the ACM**, v. 32, n. 12, p. 1415-1425, 1989.

MCDERMOTT, D.; GHALLAB, M.; HOWE, A.; KNOBLOCK, C.; RAM, A.; VELOSO, M.; WELD, D. E WILKINS, D. “**PDDL-the planning domain definition language – version 1.2**” Ecole Nationale Superieure D'ingenieur des Constructions Aeronautiques Adele Howe (Colorado State University) Craig Knoblock, ISI. 1998.

MEIDAN, A.; GARCÍA-GARCÍA, J. A.; RAMOS, I.; ESCALONA, M. J. “Measuring software process: a systematic mapping study”. **ACM Computing Surveys (CSUR)**, v. 51, n. 3, p. 1-32, 2018.

MELLI, R. “Automated software project planning and control”. In: **IFAC Proceedings Volumes**, v. 21, n. 14, p. 1-7, 1988.

MOHAMMED, K.; REDOUANE, L.; BERNARD, C. “A deviation-tolerant approach to software process evolution”. In: **Ninth international workshop on Principles of software evolution: in conjunction with the 6th ESEC/FSE joint meeting**. p. 75-78, 2007.

MONTONI, M.; SANTOS, G.; ROCHA, A. R.; FIGUEIREDO, S.; CABRAL, R.; BARCELLOS, R.; BARRETO, A; SOARES, A; CERDEIRAL, C.; LUPO, P. “Taba workstation: Supporting software process deployment based on CMMI and MR-MPS. BR”. In: **International Conference on Product Focused Software Process Improvement**. p. 249-262, Berlin, Heidelberg: Springer, 2006.

MÜNCH, J.; ARMBRUST, O.; KOWALCZYK, M. e SOTO, M. **Software process definition and management**. Berlin, Heidelberg: Springer Science & Business Media, 2012.

MYERS, Eugene W. “AnO (ND) difference algorithm and its variations”. **Algorithmica**, v. 1, n. 1-4, p. 251-266, 1986.

OMG, Object Management Group. Business Process Model and Notation (BPMN). 2014. Disponível em: http://www.omg.org/spec/BPMN/2.0.2/. Acesso em: 12 dez. 2016.

OMG, Object Management Group. Software & Systems Process Engineering Meta-Model Specification. 2008. Disponível em: http://www.omg.org/spec/SPEM/2.0/PDF. Acessado em: 12 dez. 2016.

OSTERWEIL, L. “Software processes are software too” In Proceedings of the 9th international conference on Software Engineering (pp. 2-13). IEEE Computer Society Press, 1989.

PAI M, MCCULLOCH M, GORMAN JD, PAI N, ENANORIA W, KENNEDY G, PRATHAP THARYAN; COLFORD J. J. “Systematic reviews and meta-analyses: an illustrated, step-by-step guide”. **The National medical journal of India**, v. 17, n. 2, p. 86-95, 2004.

PEDREIRA, O.; PIATTINI, M.; LUACES, M. R.; BRISABOA, N. R. “A systematic review of software process tailoring”. **ACM SIGSOFT Software Engineering Notes**, v. 32, n. 3, p. 1-6, 2007.

PEREIRA, E. B.; BASTOS, R. M.; OLIVEIRA, T. C. “Process tailoring based on well-formedness rules”. In: **SEKE**. p. 185-190, USA, july. 2008.

PFLEEGER, S. L.; ATLEE, J. M. **Software engineering: theory and practice**. India: Pearson Education, 2000.

PILLAT, R. M.; OLIVEIRA, T. C.; FONSECA, F. L. “Introducing software process tailoring to bpmn: Bpmnt”. In: **2012 International Conference on Software and System Process (ICSSP)**. p. 58-62, IEEE, 2012.

PMI, Project Management Institute. A Guide to the Project Management Body of Knowledge (PMBOK) Sixth Edition. Pennsylvania, USA: Project Management Institute, 2017.

PMI, **Project Management Institute**. Um guia do conhecimento em gerenciamento de projetos. Guia PMBOK, 5. ed. Pennsylvania, USA: Project Management Institute, 2013.

PONCIN, W.; SEREBRENIK, A.; VAN DEN BRAND, M. “Process mining software repositories”. In: **2011 15th European Conference on Software Maintenance and Reengineering**. p. 5-14, IEEE, 2011.

POSTGRESQL, 2019. Disponível em: https://www.postgresql.org/docs/. Acesso em: 29 dez. 2019.

PRESSMAN, R. S. **Engenharia de Software**, 6. ed. Porto Alegre: Editora MCGrawHill, 2010.

REDMINE, 2015. Disponível em: http://www.redmine.org/. Acesso em: 13 set. 2015.

REIS, C. A. L.; REIS, R. Q.; SCHLEBBE, H.; NUNES, D. J. “A policy-based resource instantiation mechanism to automate software process management”. In: **Proceedings of the 14th international conference on Software engineering and knowledge engineering**. p. 795-802, 2002.

REIS, C. A. L.; REIS, R. Q.; SCHLEBBE, H.; NUNES, D. J. “Resource instantiation policies for software process environments”. In: **Proceedings 26th Annual International Computer Software and Applications**. p. 53-58, IEEE, 2002.

REIS, C. A. **Uma abordagem flexível para execução de processos de software evolutivos**. 2003. Tese de Doutorado. Curso de Ciência Da Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brasil.

ROCHA, A. R. C.; MALDONADO, J. C.; WEBER, K. C. **Qualidade de software**: teoria e prática. São Paulo: Prentice Hall, 2001.

RUBIN, V.; GÜNTHER, C. W.; VAN DER AALST, W. M.; KINDLER, E.; VAN DONGEN, B. F; SCHAFER, W. “Process mining framework for software processes”. In: **International conference on software process**. p. 169-181, Berlin, Heidelberg Springer, 2007.

RUBY ON RAILS, 2019. Disponível em: https://guides.rubyonrails.org/. Acesso em: 28 dez. 2019.

RUI, Z.; TONG, L.; FEI, D.; QI, M.; LEILEI, L.; YUN, H.; YETING, C. “An Approach to Detecting Software Process Deviations”. In: **2014 International Conference on IT Convergence and Security (ICITCS)**. p. 1-4, IEEE, 2014.

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

RUY, F.B.; FALBO, R.A.; BARCELLOS, M.P.; COSTA, S.D.; GUIZZARDI, G. “SEON: a software engineering ontology network”. In: **European Knowledge Acquisition Workshop**. p. 527-542, Cham: Springer, 2016.

SANTOS, R. M. S.; OLIVEIRA, T. C.; BRITO e ABREU, F. “Mining software development process variations”. In: **Proceedings of the 30th Annual ACM Symposium on Applied Computing**. p. 1657-1660, 2015.

SAYLAM, R. e SAHINGOZ, O. K. “A process mining approach in software development and testing process: a case study”. In: **Proceedings of the World Congress on Engineering**. 2014.

SEON, Software Engineering Ontology Network. Disponível em: http://dev.nemo.inf.ufes.br/seon/. Acesso em: 18 ago. 2019.

SILVA, M. A.; BENDRAOU, R.; BLANC, X.; GERVAIS, M. P. “Early deviation detection in modeling activities of mde processes”. In: **International Conference on Model Driven Engineering Languages and Systems**. p. 303-317, Berlin, Heidelberg: Springer,, 2010.

SILVA, M. A.; BLANC, X.; BENDRAOU, R.; GERVAIS, M. P. Experiments on the impact of deviations to process execution. **Ingénierie des systèmes d'information (2001)**, v. 18, n. 3, p. 95-119, 2013.

SMATTI, M.; NACER, M. A. “Dealing with Deviations on Software Process Enactment: Comparison Framework”. In: **Conference on Advanced Aspects of Software Engineering**. p. 108-115, 2014.

SMATTI, M.; OUSSALAH, M.; NACER, M. A. “Supporting Deviations on Software Processes: A Literature Overview”. In: **ICSOFT**. p. 191-209, Cham: Springer, 2015.

SOFTEX “MPS.BR – Melhoria de Processo do Software Brasileiro – Guia Geral MPS de Software”. Disponível em: http://www.softex.br/mpsbr. Acesso em: 15 jun. 2016a.

SOFTEX “MPS.BR – Melhoria de Processo do Software Brasileiro – Guia de Implementação – Parte 2: Fundamentação para Implementação do Nível F do MR-MPS-SW”. Disponível em: http://www.softex.br/mpsbr. Acesso em: 15 jun 2016b.

SØRUMGÅRD, S. **Verification of process conformance in empirical studies of software development**. 1997. Tese de Doutorado. Ph. D. thesis, Norwegian University of Science and Technology, Noruega.

SZIMANSKI, F. **Melhoria de Modelos de Processo de Negócio com Mineração de Processos e Simulação Baseada em Agentes**. 2013. Tese de Doutorado. Universidade de Brasília, Brasil.

TAYMOURI, F.; CARMONA, J. “A recursive paradigm for aligning observed behavior of large structured process models”. In: **International Conference on Business Process Management**. p. 197-214, Cham: Springer, 2016.

VALLE, A. M.; SANTOS, E. A. P.; LOURES E. R. “Applying process mining techniques in software process appraisals”. **Information and software technology**, v. 87, p. 19-31, 2017.

VAN DER AALST, W. M. **Process mining: discovery, conformance and enhancement of business processes**. Heidelberg: Springer, 2011.

VAN DER AALST, W. M.; ADRIANSYAH, A.; VAN DONGEN, B. “Replaying history on process models for conformance checking and performance analysis”. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, v. 2, n. 2, p. 182-192, 2012.

VAN DER AALST, W. M.; PESIC, M. “DecSerFlow: Towards a truly declarative service flow language”. In: **International Workshop on Web Services and Formal Methods**. p. 1-23, Berlin, Heidelberg: Springer, 2006.

VAN DER AALST, Wil *et al.* “Process mining manifesto”. In: **International Conference on Business Process Management**. p. 169-194, Berlin, Heidelberg; Springer, 2011.

VAN DONGEN, B. F., DE MEDEIROS, A. K. A., VERBEEK, H. M. W., WEIJTERS, A. J. M. M., & VAN DER AALST, W. M. “The ProM framework: A new era in process mining tool support”. In: **International conference on application and theory of petri nets**. p. 444-454, Berlin, Heidelberg; Springer: 2005.

VAN DONGEN, B.; CARMONA, J.; CHATAIN, T.; TAYMOURI, F. “Aligning modeled and observed behavior: a compromise between computation complexity and

quality". In: **International Conference on Advanced Information Systems Engineering**. p. 94-109, Berlin, Heidelberg; Springer, 2017.

VANDERFEESTEN, J.; CARDOSO, J.; MENDLING, H.; REIJERS, A. e VAN DER AALST., W.M.P. Quality metrics for business process models. **BPM and Workflow handbook**, v. 144, p. 179-190, 2007.

VEIGA, G. M.; FERREIRA, D. R. "Understanding spaghetti models with sequence clustering for ProM". In: **International Conference on Business Process Management**. p. 92-103, Berlin, Heidelberg: Springer, 2009.

VERBEEK, H. M. W.; BUIJS, J. C.; VAN DONGEN, B. F.; VAN DER AALST, W. M. "Xes, xesame, and prom 6". In: **International Conference on Advanced Information Systems Engineering**. p. 60-75, Berlin, Heidelberg: Springer, 2010.

VILLELA, K.; SANTOS, G.; MONTONI, M.; BERFGER, P.; FIGUEIREDO, S.; MAFRA, S., ROCHA, A.N., TRAVASSOS, G.H. "Definição de processos em ambientes de desenvolvimento de software orientados a organização". **III Simpósio Brasileiro de Qualidade de Software-SBQS 2004**, p. 4-18, 2004.

WARD, W. "Software defect prevention using mccabe's complexity metric". **Hewlett-Packard Journal**, v. 40, n. 2, p. 64-&, 1989.

WAZLAWICK, R. S. **Metodologia de pesquisa para ciência da computação**. São Paulo: Elsevier Editora, 2009.

WESKE, M "Business process management architectures". In: **Business Process Management**. p. 333-371, Berlin, Heidelberg: Springer, 2012.

WOHLIN, C. "Guidelines for snowballing in systematic literature studies and a replication in software engineering". In: **Proceedings of the 18th international conference on evaluation and assessment in software engineering**. p. 1-10, 2014.

WU, C.; SIMMONS, D.B. "Software Project Planning Associate (SPPA): a knowledge-based approach for dynamic software project planning and tracking". In: **Proceedings 24th Annual International Computer Software and Applications Conference. COMPSAC2000**. p. 305-310, IEEE, 2000.

YONG, Y.; ZHOU, B. "Software process deviation threshold analysis by system dynamics". In: **2010 2nd IEEE International Conference on Information Management and Engineering**. p. 121-125, IEEE, 2010.

APÊNDICE A – Pesquisa de Opinião

Este apêndice apresenta o planejamento da pesquisa de opinião realizada para evidenciar o problema tratado nesta tese. Serão apresentados o protocolo da pesquisa de opinião que descreve seu objetivo, questão de pesquisa, hipóteses, orientações de análise e o instrumento usado para coletar os dados. Ao final os gráficos gerados a partir das respostas obtidas são apresentados.

Objetivo Global

O objetivo global é investigar se as características de processo de desenvolvimento de software, identificadas na literatura, estão presentes ao longo da condução dos projetos de desenvolvimento de software. Também objetivamos investigar quais ferramentas são utilizadas para observar as características de processo de desenvolvimento, com o propósito de identificar as atuais estruturas de armazenamento de dados de processo.

Objetivo do Estudo

Os objetivos específicos desta pesquisa de opinião foram definidos utilizando a abordagem Goal-Question-Metric (GQM) proposta em (BASILI, CALDIERA e ROMBACH, 1994), e são os seguintes:

- ***Analisar*** *as características de processo de desenvolvimento de software c****om o propósito de ca****racterizar c****om respeito a i****dentificação das características de processo de processo ao longo da condução do projeto de desenvolvimento de software* ***do ponto de vista dos*** *Gerentes de Projeto, Analistas e Desenvolvedores* ***no contexto de*** *projetos de desenvolvimento de software.*
- ***Analisar*** *as ferramentas utilizadas para dar apoio ao longo da condução do projeto de desenvolvimento de software* ***com o propósito de*** *caracterizar* ***com respeito ao*** *conjunto de características de processo identificadas nas ferramentas* ***do ponto de vista dos*** *Gerentes de Projeto, Analistas e Desenvolvedores* ***no contexto de*** *projetos de desenvolvimento de software.*

O objeto de estudo inclui as características de processo de desenvolvimento de software, identificadas na literatura, conforme descrito no Capítulo 3.

Objetivo de Medição

O objetivo de medição é, a partir de um conjunto de características de processo de desenvolvimento de software, analisar:

1 Quais características de processo de desenvolvimento de software são observadas durante o projeto de desenvolvimento de software.
2 Quais características de processo de desenvolvimento de software não são observadas durante o projeto de desenvolvimento de software.
3 Quais ferramentas são utilizadas para explicitar as características de processo de desenvolvimento de software que são observadas durante o projeto de desenvolvimento de software.

Questões de Pesquisa

As questões de pesquisa consideradas são:

Q1: As características de processo de desenvolvimento de software identificadas na literatura de processo são observadas durante o projeto de desenvolvimento de software?

Métrica 1: número de características de processo de desenvolvimento de software classificadas como observadas durante o projeto de desenvolvimento de software.

Q2: Quais ferramentas são utilizadas para observar as características de processo de desenvolvimento de software identificadas na literatura de processo?

Métrica 1: número de ferramentas utilizadas para observar as características de processo de desenvolvimento de software durante o projeto de desenvolvimento de software.

Definição de Hipóteses

H0: Todas as características de processo de desenvolvimento de software podem ser observadas durante o projeto de desenvolvimento de software pelos Gerentes de Projeto, Analistas e Desenvolvedores.

H': Uma ou mais características de processo de desenvolvimento de software não podem ser observadas durante o projeto de desenvolvimento de software pelos Gerentes de Projeto, Analistas e Desenvolvedores.

H0: A utilização de ferramentas de apoio ao processo de desenvolvimento software permite que os Gerentes de Projeto, Analistas e Desenvolvedores observem todas as características de processo de desenvolvimento de software durante o projeto de desenvolvimento de software.

H': A utilização de ferramentas de apoio ao processo de desenvolvimento software permite que os Gerentes de Projeto, Analistas e Desenvolvedores observem uma ou mais características de processo de desenvolvimento de software durante o projeto de desenvolvimento de software.

Definição da Instrumentação

A pesquisa de opinião deve ser preenchida por Gerentes de Projeto, Analistas e Desenvolvedores.

Os profissionais participantes devem indicar dois aspectos:

1. Se as características de processo de desenvolvimento de software apresentadas são observadas durante o projeto de desenvolvimento de software.
2. Se ferramentas são utilizadas para explicitar as características de processo de desenvolvimento observadas durante o projeto de desenvolvimento de software.

O questionário foi disponibilizado de forma diferente em cada uma das rodadas executada. Na primeira e terceira rodadas a pesquisa de opinião foi não supervisionada, não havendo nenhum auxílio pessoal aos participantes, apenas instruções descritas no questionário que foi disponibilizado na internet. Na segunda

rodada que ocorreu de forma supervisionada o questionário foi disponibilizado de forma impressa com auxílio pessoal aos participantes.

O questionário está dividido em quatro partes:

- caracterização do participante;
- caracterização da organização;
- identificação das características de processo de desenvolvimento de software observadas durante o projeto de desenvolvimento de software;
- identificação das ferramentas utilizadas para observar as características de processo de desenvolvimento durante o projeto de desenvolvimento de software

Variáveis

As variáveis independentes do estudo são:

- o conjunto de características de processo de desenvolvimento de software extraídas da literatura de processo.

As variáveis dependentes são:

- o conjunto de características de processo de desenvolvimento de software observadas durante o projeto de desenvolvimento e software,
- o conjunto de ferramentas utilizadas para explicitar as características de processo de desenvolvimento de software observadas.

Detalhes de Instrumentação

A instrumentação utilizada neste estudo foi projetada para ser tão simples quanto possível e demandar a mínima quantidade de tempo para os participantes responderem as questões. O instrumento final foi refinado ao longo das rodadas de execução com o objetivo de aplicar o estudo de forma não supervisionada, não havendo nenhum auxílio pessoal aos participantes, apenas instruções descritas no questionário.

O questionário atual está dividido em três partes:

- caracterização do participante e da organização;

- identificação das características de processo de desenvolvimento de software observadas durante o projeto de desenvolvimento de software;
- identificação das ferramentas utilizadas para observar as características de processo de desenvolvimento durante o projeto de desenvolvimento de software

A primeira tela é a tela de boas vindas, que descreve os objetivos da pesquisa, apresenta um resumo das etapas a serem seguidas nos questionários e fornece outras instruções de uso do instrumento:

"*This Survey aims at verifying the software development process features that can be observed during the software development project. The survey also intends to identify what tools have been used to support practitioners (Project Managers, System Analysts and Developers) during the execution of software development projects.*

The survey is being carried out by Renata Mesquita da Silva Santos, Phd. Student at COPPE/UFRJ, under the supervision of Dr. Toacy Oliveira (COPPE/UFRJ), Dr. Fernando Brito e Abreu (ISCTE-IUL), Dr. Guilherme Horta Travassos (COPPE/UFRJ), also with the participation of Rafael Maiani de Mello (Phd Student at COPPE/UFRJ).

The time observed to fill out the survey in our pilot application fit the range of 15-20 minutes, with a few participants needing no more than 30 minutes. Therefore, its execution can be saved and resumed at any time to facilitate your contribution.

The information gathered in this survey will only be used for academic purposes, and results will always be published in their entirety, thus ensuring absolute confidentiality of organizations and individual responses.

Your contribution is very important to support our research and we hope to be able to count on your contribution. If you agree to participate in this survey, you may proceed to the next step.

The survey includes a total of 3 steps: i) participant characterization; ii) the observance of a set of software development process features; iii) and tools used.

There are specific links available on the survey's page to allow you to freely navigate in the questionnaire. Please refrain from using the browsers' navigation button as they will mess up with the survey session and lead to information loss. We thank you for your contribution!"

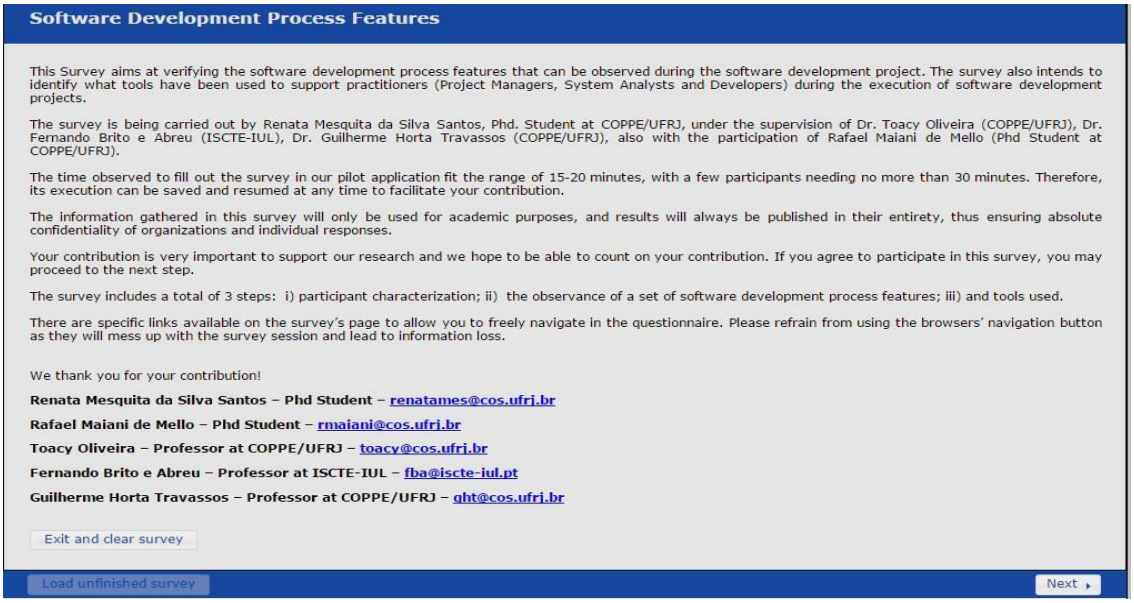


Figura A.1 Tela inicial

As Figuras A.2, A.3 e A.4 apresentam a etapa de caracterização do participante e da organização (onde atualmente o participante atua). A caracterização do participante é organizada em três grupos de questões. O primeiro grupo é apresentado na Figura 3.2 composta por questões relacionadas a informações específicas do participante (e-mail, país, formação e ocupação).

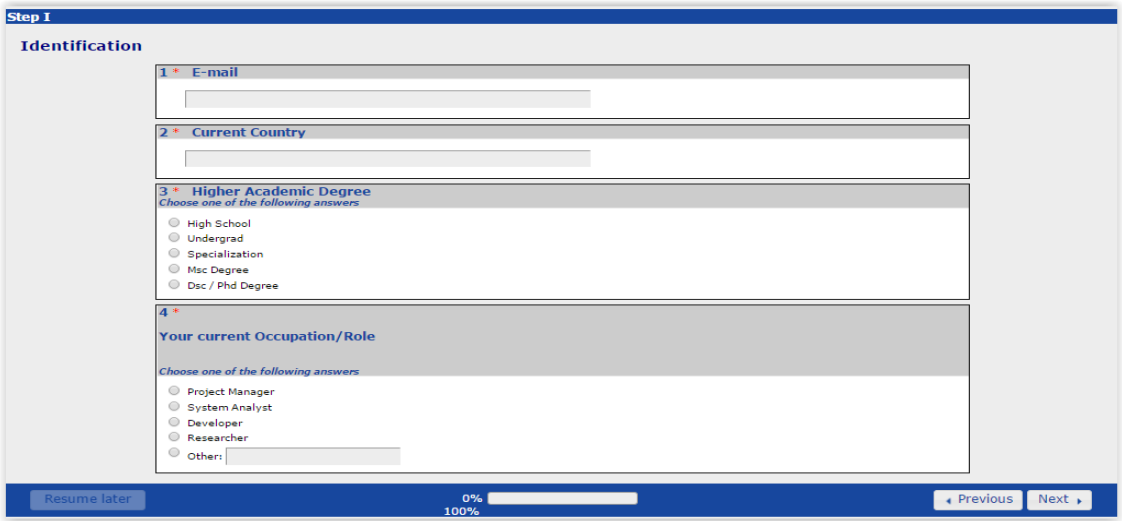


Figura A.2 Etapa I – Informações do participante

O segundo grupo, apresentando na Figura 3.3, é formado por questões sobre a experiência do participante em projetos de desenvolvimento de software processos de desenvolvimento de software. E o terceiro grupo, apresentado na Figura 3.4 é composto de questões sobre a caracterização da atual organização do participante

(país, quantidade de desenvolvedores, modelo de maturidade e qualidade seguido, se processo de desenvolvimento de software é adotado).

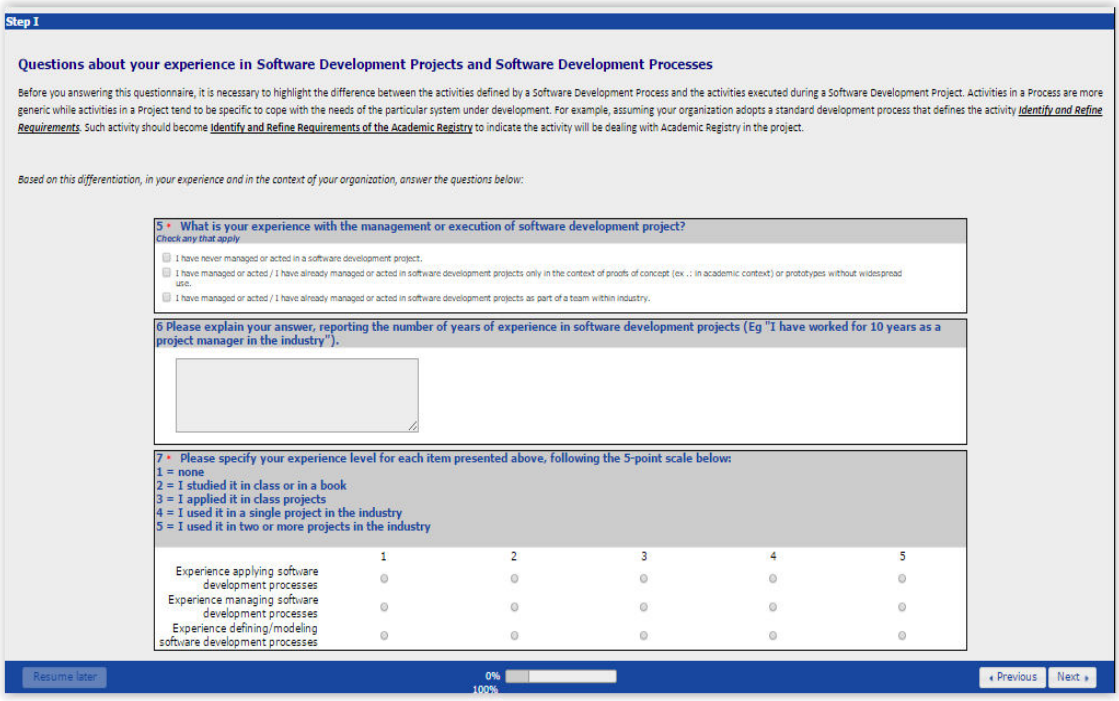


Figura A.3 Etapa I – Experiência do participante

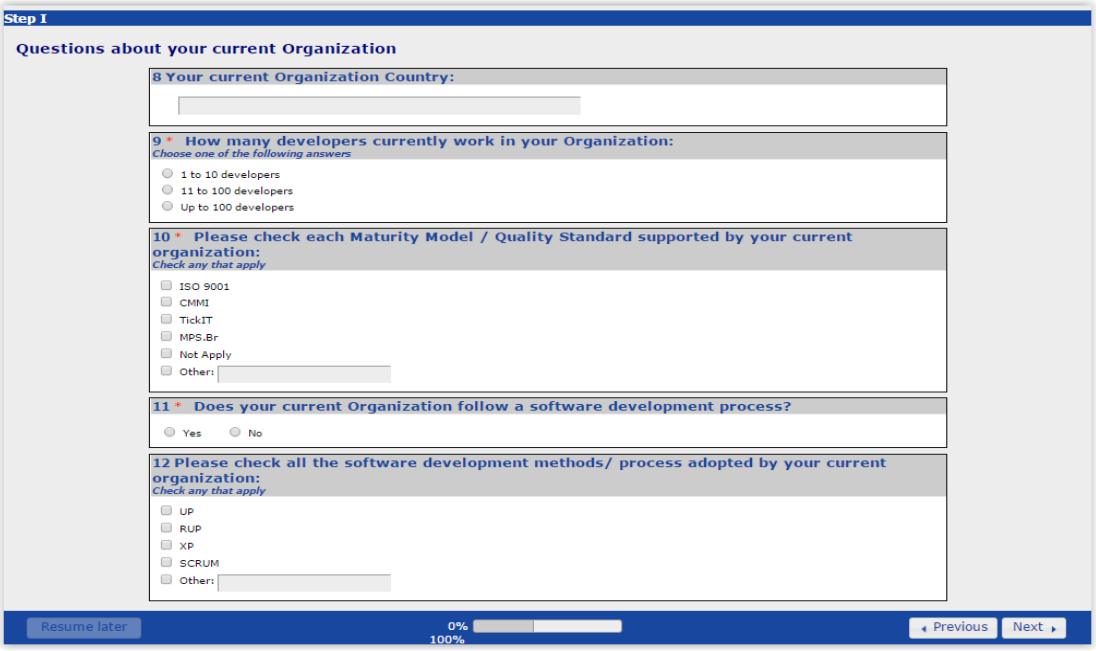


Figura A.4 Etapa I – Caracterização da Organização

A segunda etapa é composta pela questão central desta pesquisa opinião. A Figura A.5 ilustra a tela onde são apresentadas as questões sobre as características de processo extraída da literatura (Capítulo 3). A primeira questão solicita que o participante marque todas as características que são observadas durante o projeto de

desenvolvimento de software e a segunda pretende obter quais características, que não tenham sido mencionadas anteriormente, são observadas ou se gostaria que fossem observadas durante o projeto de desenvolvimento de software. A lista completa será apresentada em seguida.

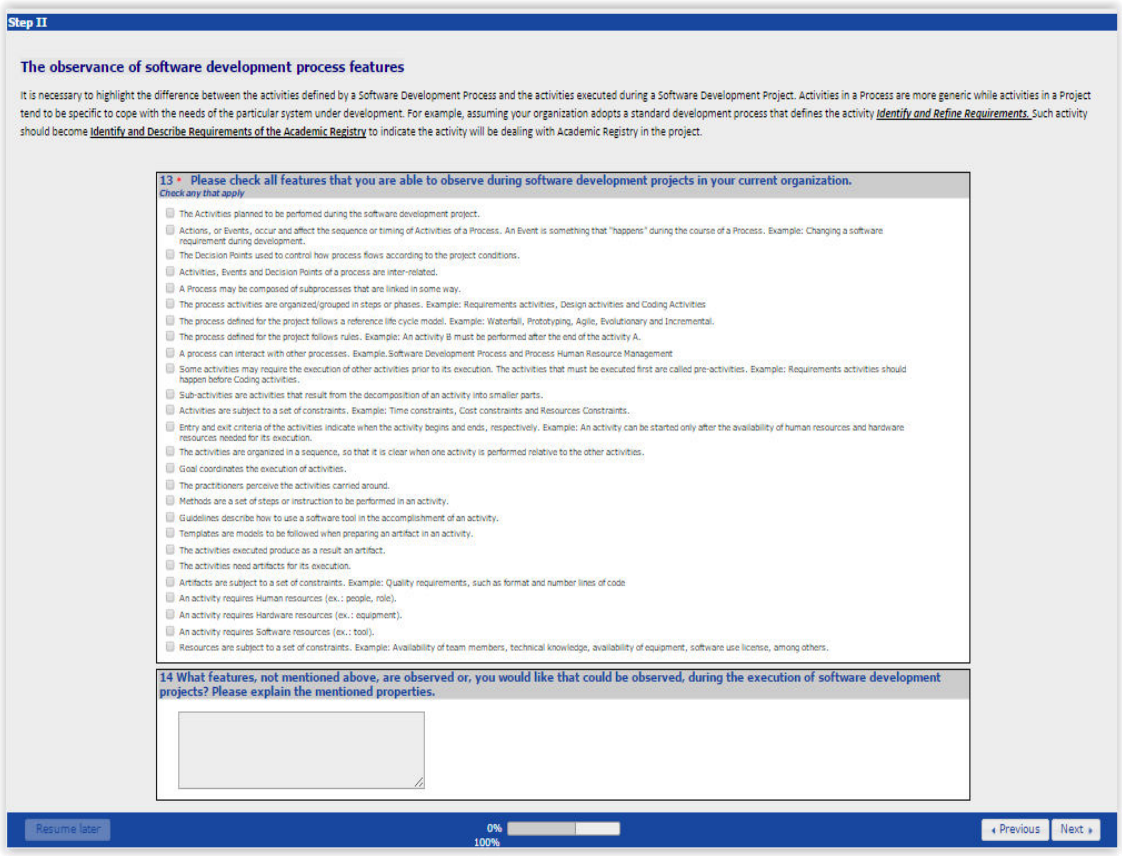


Figura A.5 Etapa II – Observância das Características

Lista de Características de processo de desenvolvimento de software apresentadas na versão atual do questionário:

- *The Activities planned to be perfomed during the software development project.*
- *Actions, or Events, occur and affect the sequence or timing of Activities of a Process. An Event is something that "happens" during the course of a Process. Example: Changing a software requirement during development.*
- *The Decision Points used to control how process flows according to the project conditions.*
- *Activities, Events and Decision Points of a process are inter-related.*
- *A Process may be composed of subprocesses that are linked in some way.*
- *The process activities are organized/grouped in steps or phases. Example: Requirements activities, Design activities and Coding Activities*

- *The process defined for the project follows a reference life cycle model. Example: Waterfall, Prototyping, Agile, Evolutionary and Incremental.*
- *The process defined for the project follows rules. Example: An activity B must be performed after the end of the activity A.*
- *A process can interact with other processes. Example.Software Development Process and Process Human Resource Management*
- *Some activities may require the execution of other activities prior to its execution. The activities that must be executed first are called pre-activities. Example: Requirements activities should happen before Coding activities.*
- *Sub-activities are activities that result from the decomposition of an activity into smaller parts.*
- *Activities are subject to a set of constraints. Example: Time constraints, Cost constraints and Resources Constraints.*
- *Entry and exit criteria of the activities indicate when the activity begins and ends, respectively. Example: An activity can be started only after the availability of human resources and hardware resources needed for its execution.*
- *The activities are organized in a sequence, so that it is clear when one activity is performed relative to the other activities.*
- *Goal coordinates the execution of activities.*
- *The practitioners perceive the activities carried around.*
- *Methods are a set of steps or instruction to be performed in an activity.*
- *Guidelines describe how to use a software tool in the accomplishment of an activity.*
- *Templates are models to be followed when preparing an artifact in an activity.*
- *The activities executed produce as a result an artifact.*
- *The activities need artifacts for its execution.*
- *Artifacts are subject to a set of constraints. Example: Quality requirements, such as format and number lines of code*
- *An activity requires Human resources (ex.: people, role).*
- *An activity requires Hardware resources (ex.: equipment).*
- *An activity requires Software resources (ex.: tool).*
- *Resources are subject to a set of constraints. Example: Availability of team members, technical knowledge, availability of equipment, software use license, among others*.

A última etapa é composta pela questão sobre as ferramentas que são utilizadas para explicitar as características de processo que foram selecionadas na questão sobre a observância das características. O questionário foi configurado de forma que somente as características selecionadas na questão principal desta pesquisa sejam exibidas. Esta configuração teve como objetivo reduzir o tempo de resposta e tornar o questionário mais otimizado. A figura A.6 apresenta a última etapa.

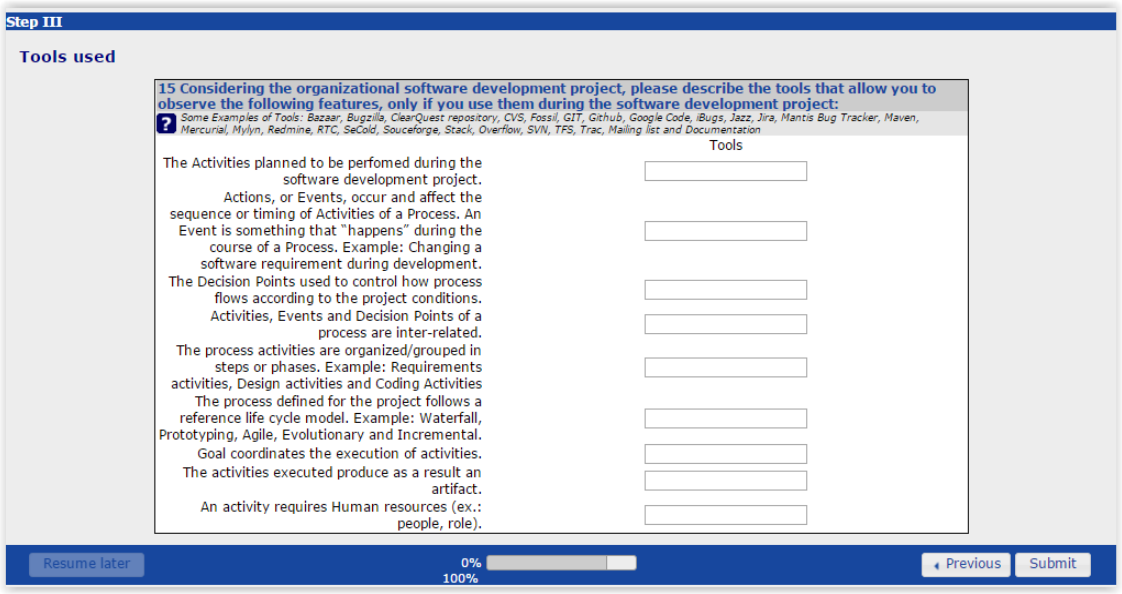


Figura A.6 Etapa III – Ferramentas Utilizadas

Para finalizar, a última tela, apresentada na Figura A.7, agradece ao participante sua colaboração e fornece informações de contato, caso o participante tenha dúvidas ou sugestões.

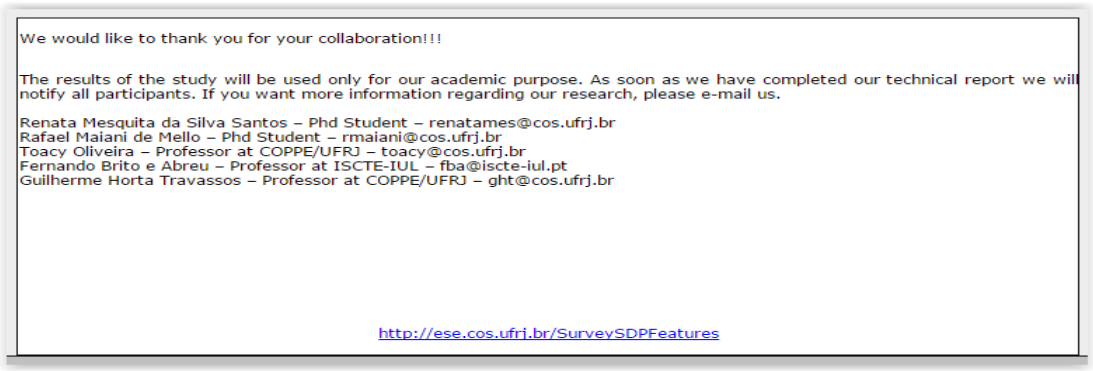


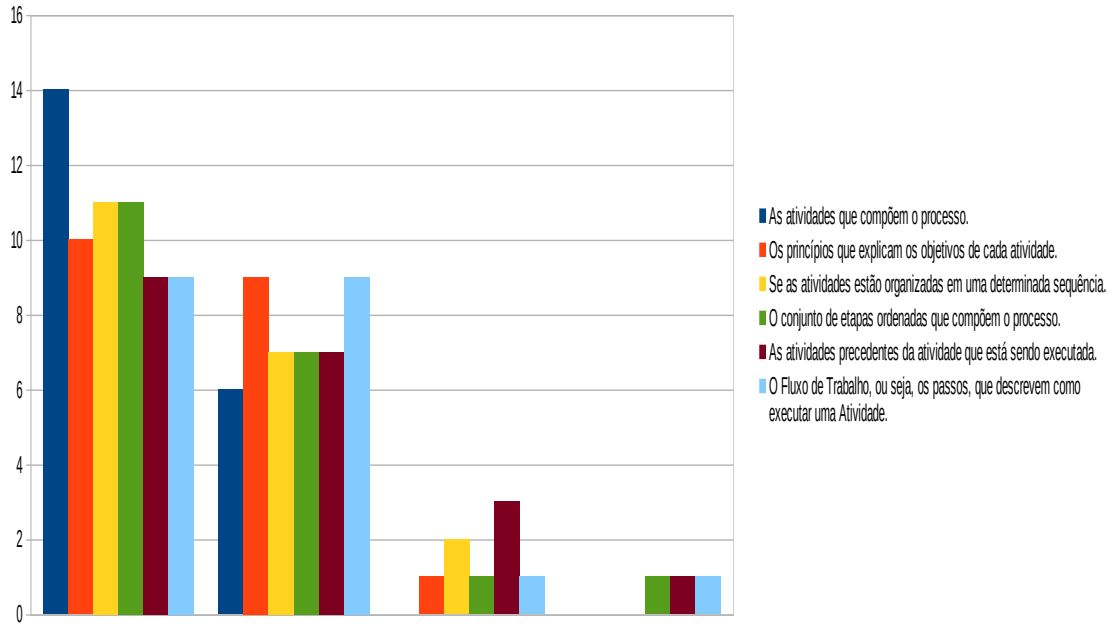
Figura A.7 Tela Final

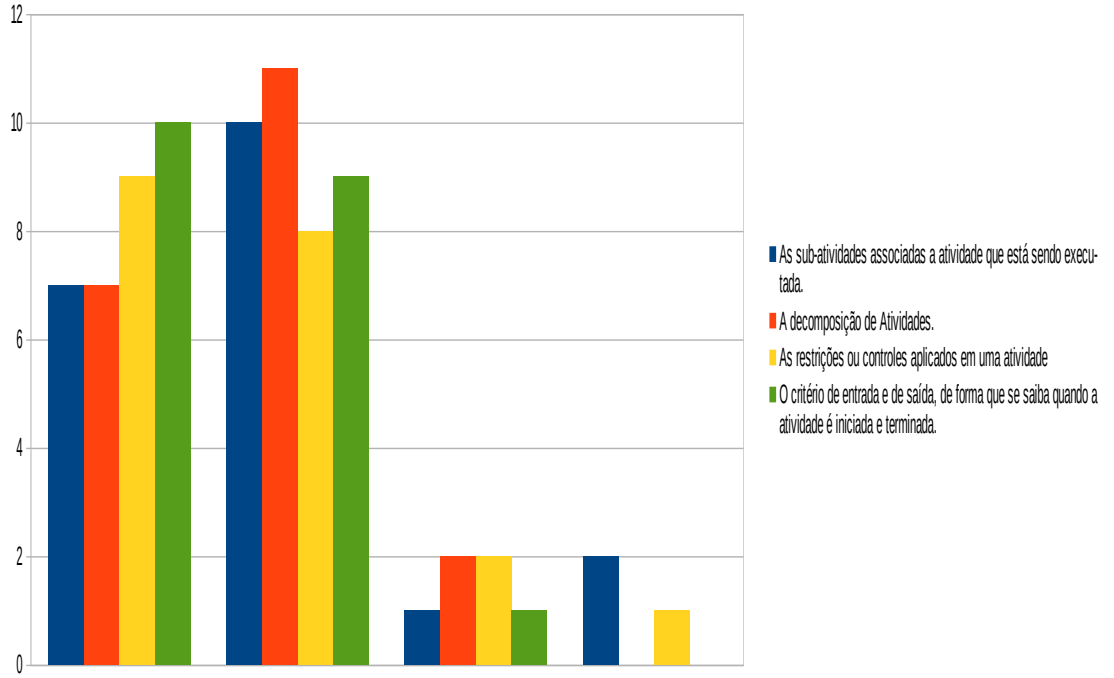
Todas as etapas de planejamento descritas até aqui, incluindo o instrumento apresentado, é baseado na versão atual do questionário. Esta versão é resultado da evolução ocorrida, quando oportunidades de melhorias que foram identificadas em cada rodada de execução. Em cada rodada de execução o contexto e a seleção dos participantes ocorreram de forma diferenciada, por esta razão estas seções, que em geral fazem parte do planejamento, serão descritas em cada rodada a seguir, além dos resultados e discussões.

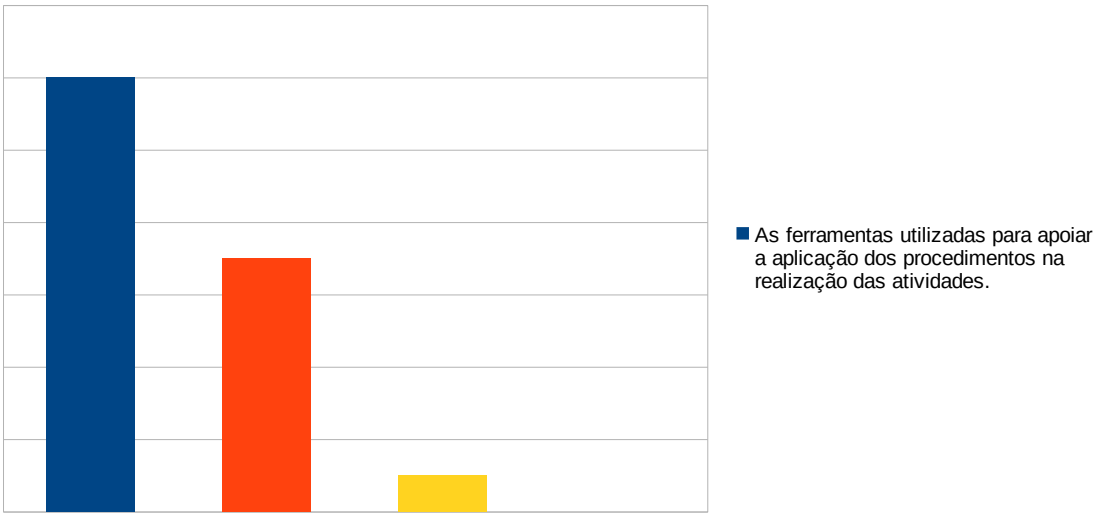
Gráficos gerados a partir da execução de cada rodada

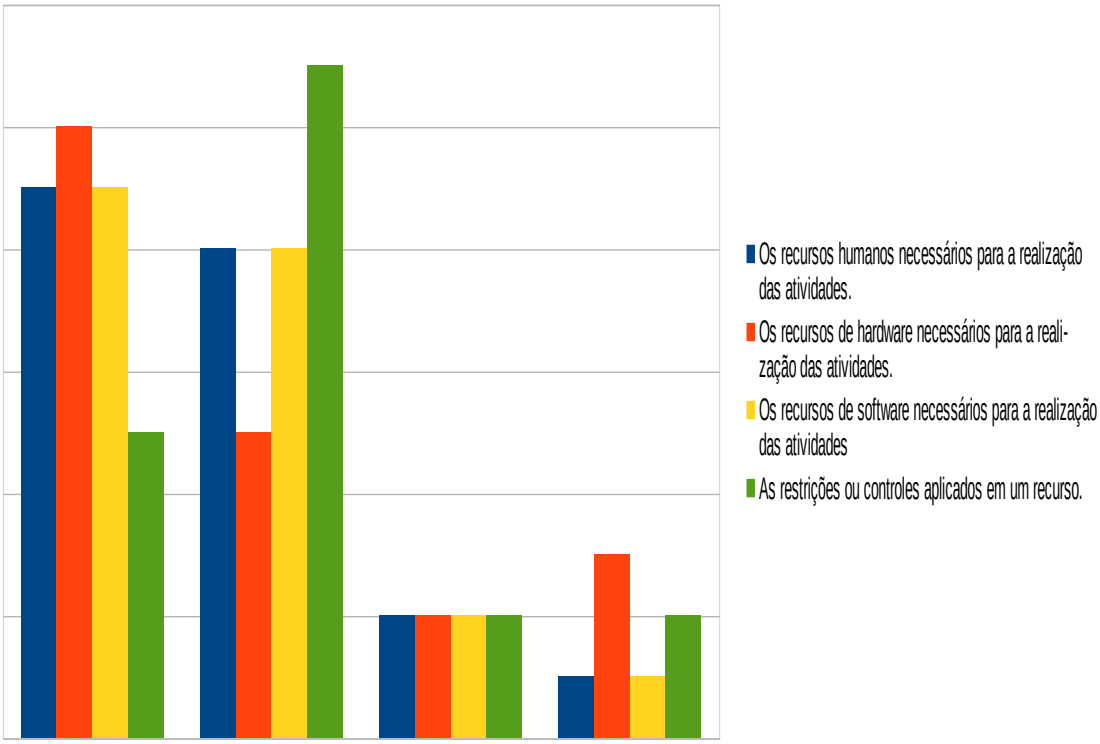
Os gráficos a seguir foram gerados para cada rodada de execução descrita no Capítulo 3.

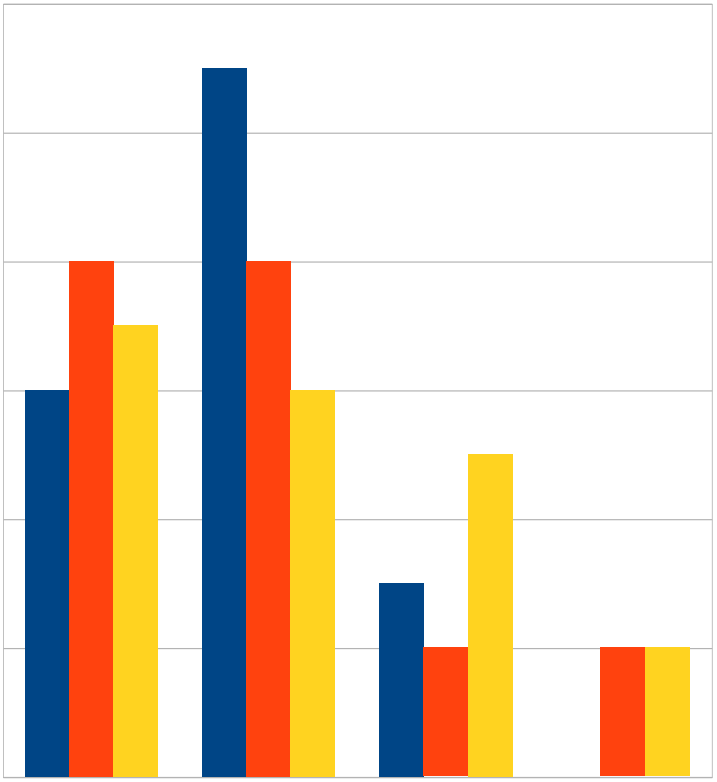
Gráficos Primeira Rodada



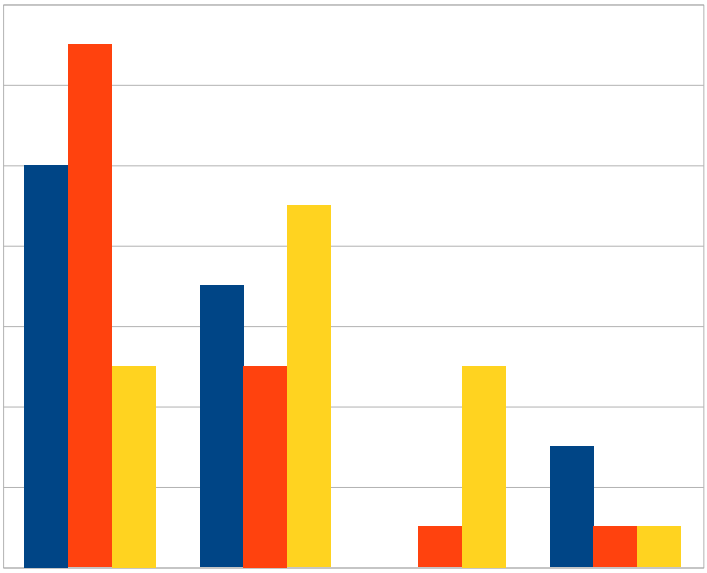



As ferramentas utilizadas para apoiar a aplicação dos procedimentos na realização das atividades.

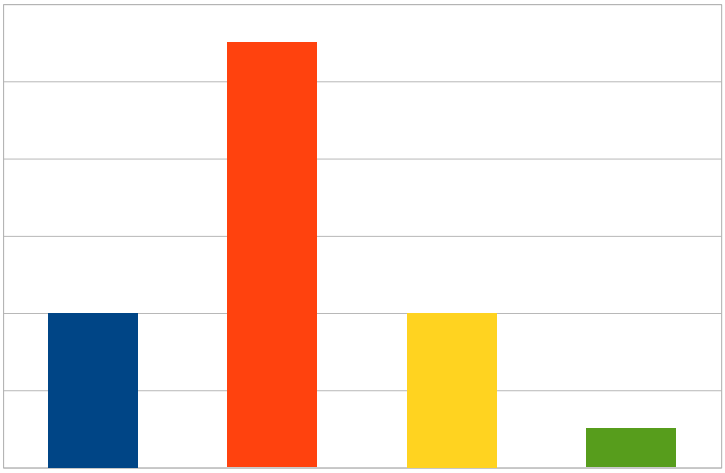

Os recursos humanos necessários para a realização das atividades.
Os recursos de hardware necessários para a realização das atividades.
Os recursos de software necessários para a realização das atividades
As restrições ou controles aplicados em um recurso.



- Os pontos de tomada de decisão.
- O modelo de ciclo de vida estabelecido para o processo.
- Os roteiros estabelecidos para a elaboração dos principais documentos (artefatos) gerados no desenvolvimento.

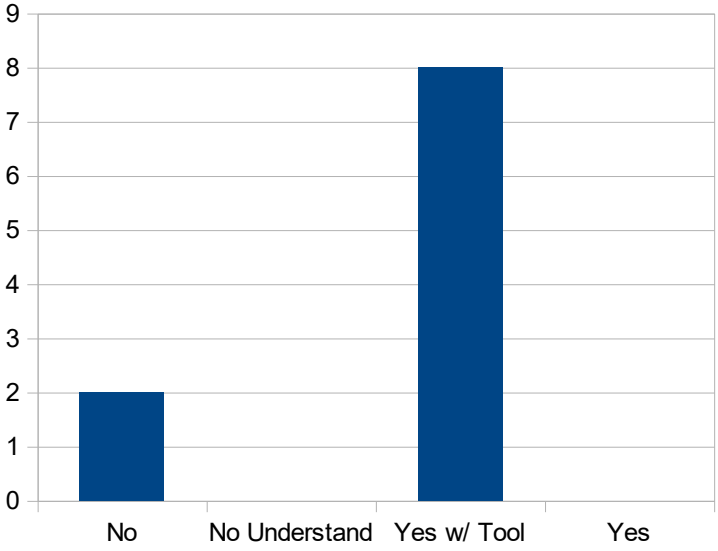


- O(s) artefato(s) consumido(s) necessário(s) em cada atividade.
- O(s) artefato(s) produzido(s) em cada atividade.
- As restrições ou controles aplicados em um artefato.

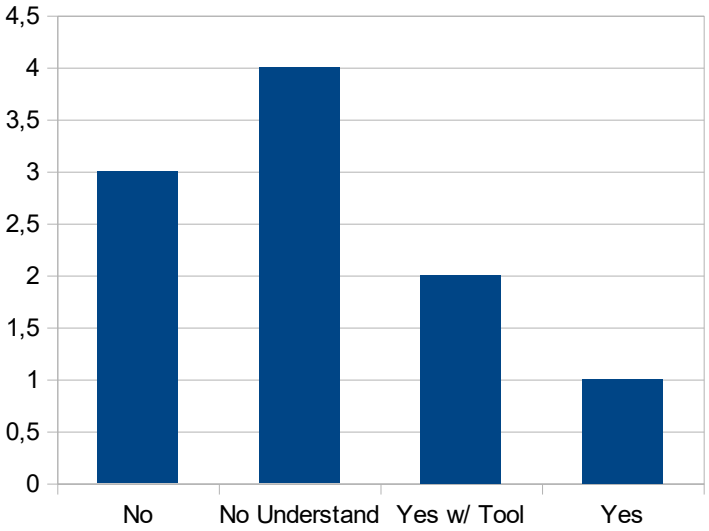


Os eventos que compõem o processo.

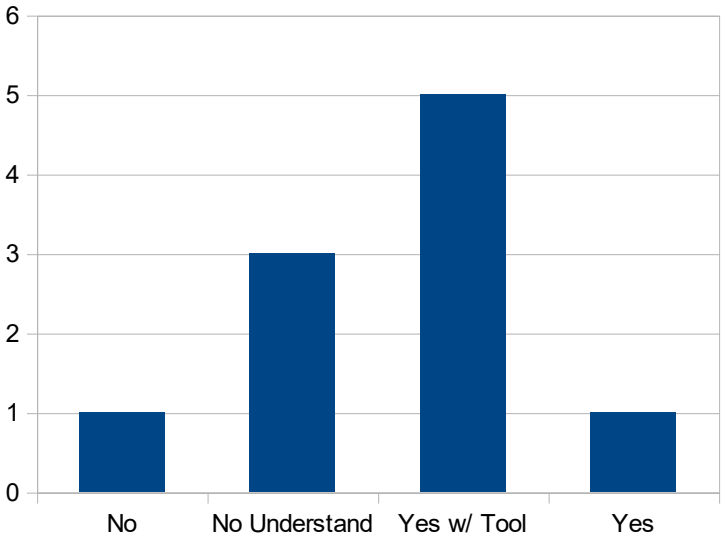
Gráficos Segunda Rodada



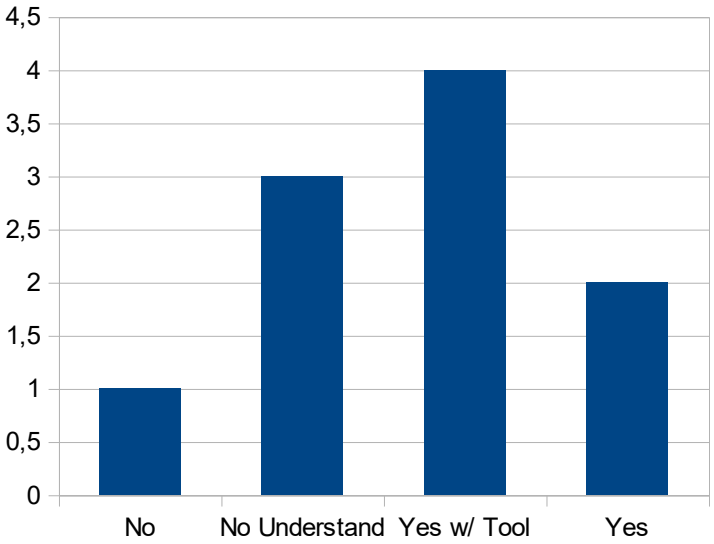
[The Activities planned to be perfomed during the software development project.]



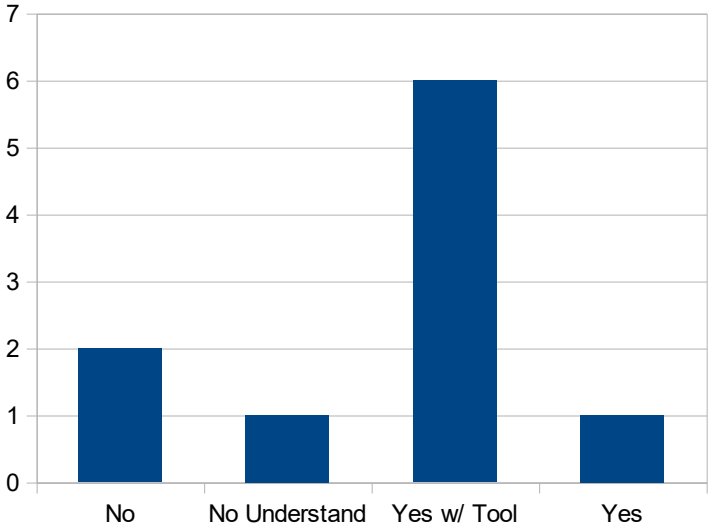
[Actions, or Events, occur and affect the sequence or timing of Activities of a Process. An Event is something that “hap-pens” during the course of a Process. Example: Changing a software requirement during development.]



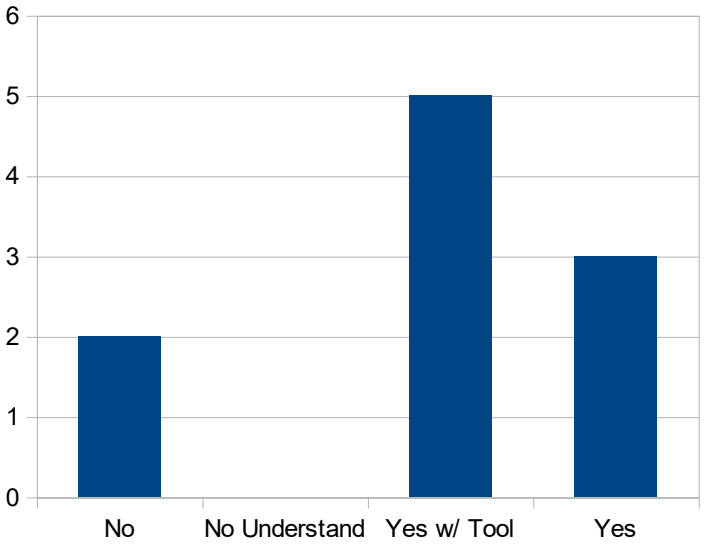
[The Decision Points used to control how process flows according to the project conditions.]



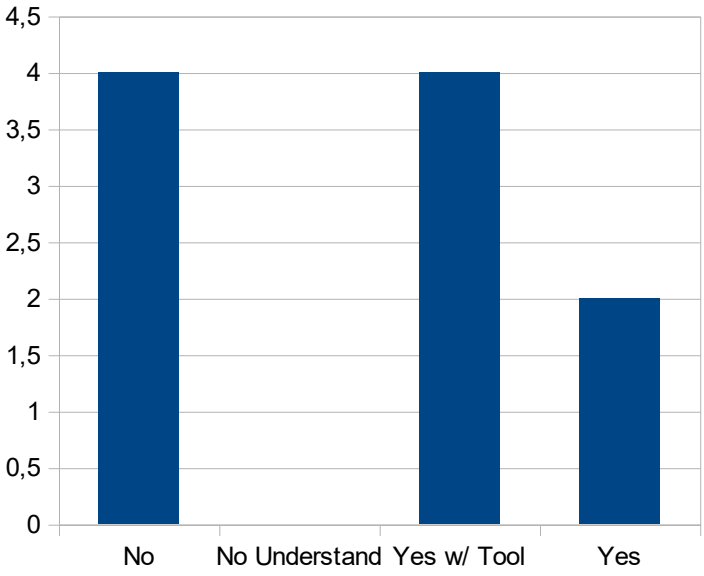
[Activities, Events and Decision Points of a process are inter-related.]



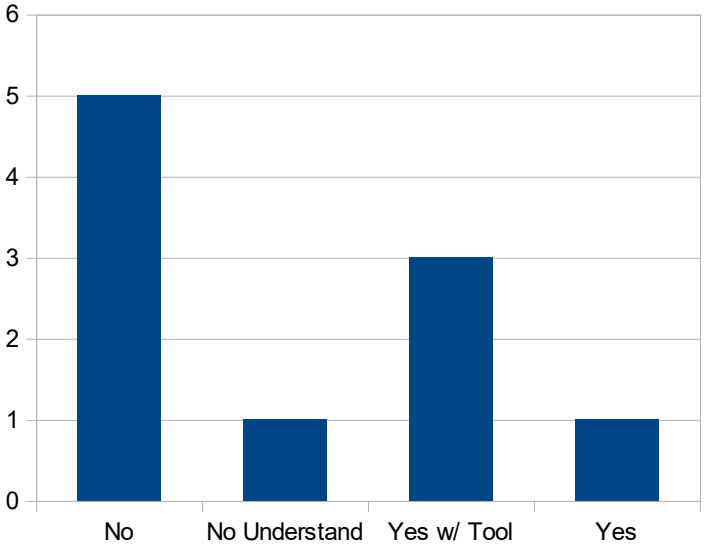
[A Process may be composed of subprocesses that are linked in some way.]



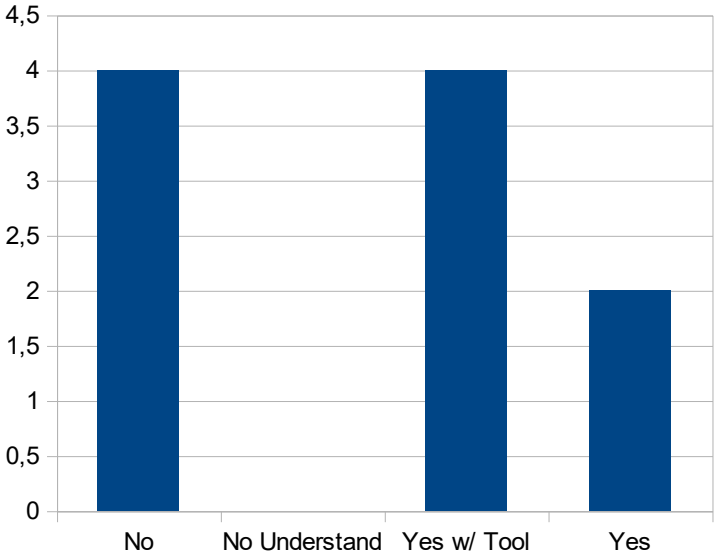
[The process activities are organized/grouped in steps or phases. Example: Requirements activities, Design activities and Coding Activities]



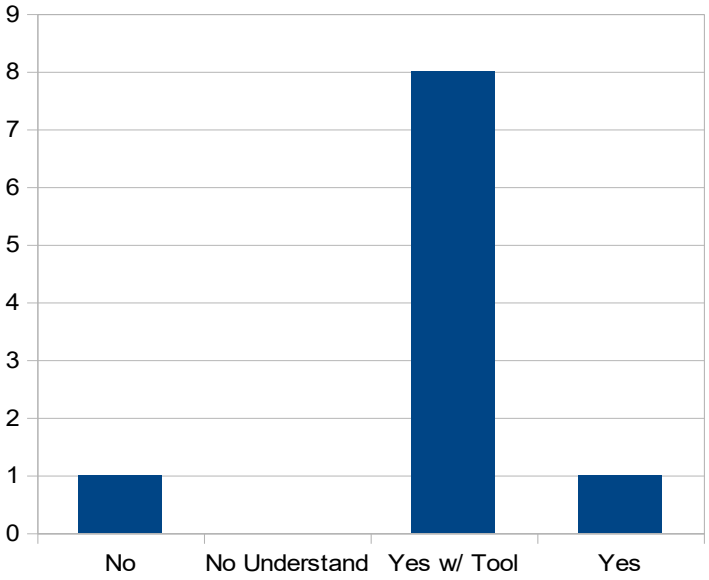
[The process defined for the project follows a reference life cycle model. Example: Waterfall, Prototyping, Agile, Evolutionary and Incremental.]



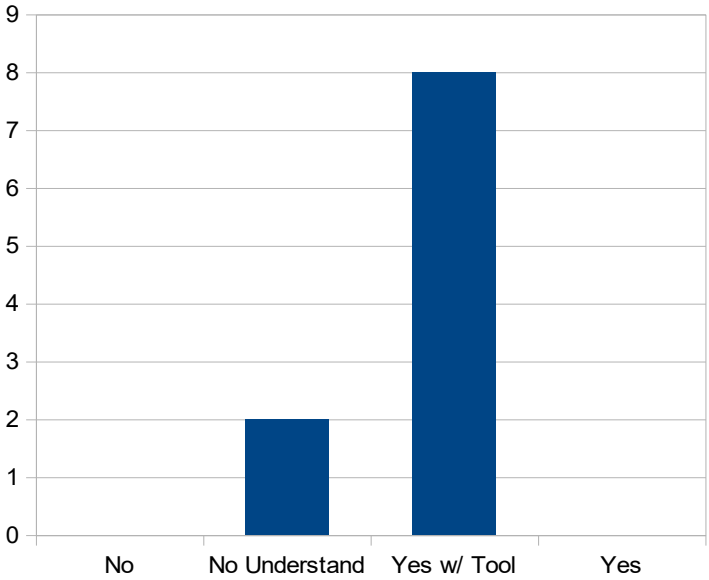
[The process defined for the project follows rules. Example: An activity B must be performed after the end of the activity A.]



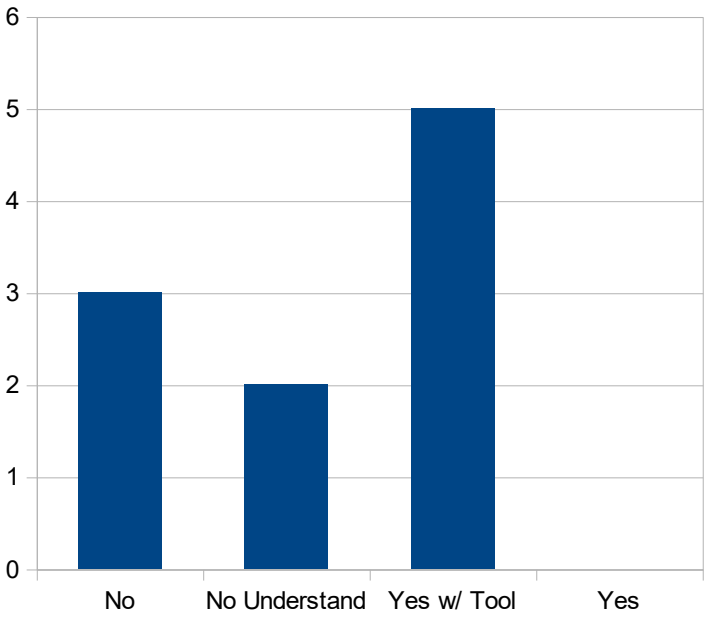
[A process can interact with other processes. Example.- Software Development Process and Process Human Resource Management]



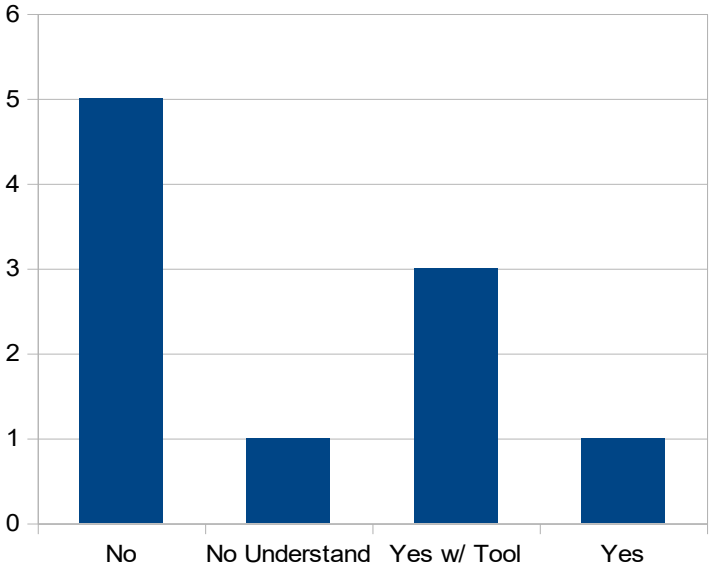
[Some activities may require the execution of other activities prior to its execution. The activities that must be executed first are called pre-activities. Example: Requirements activities should happen before Coding activities.]



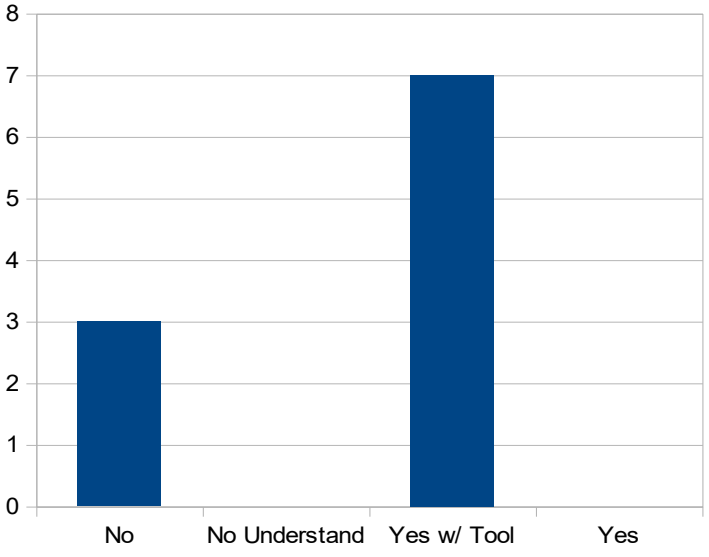
[Sub-activities are activities that result from the decomposition of an activity into smaller parts.]



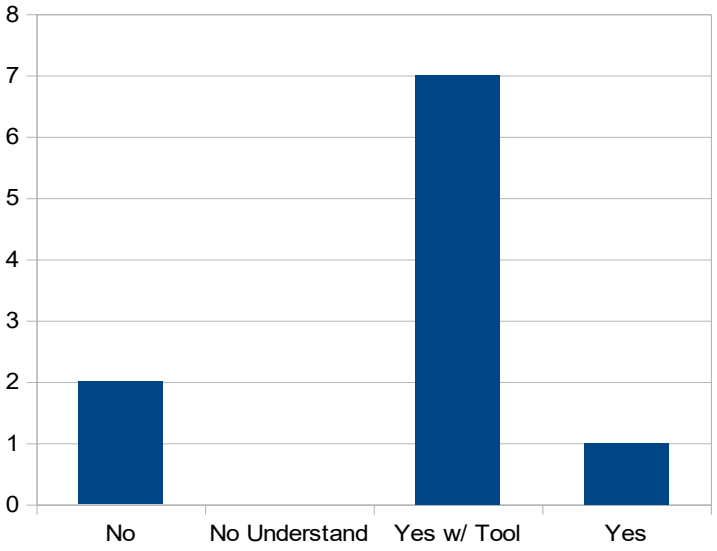
[Activities are subject to a set of constraints. Example: Time constraints, Cost constraints and Resources Constraints.]



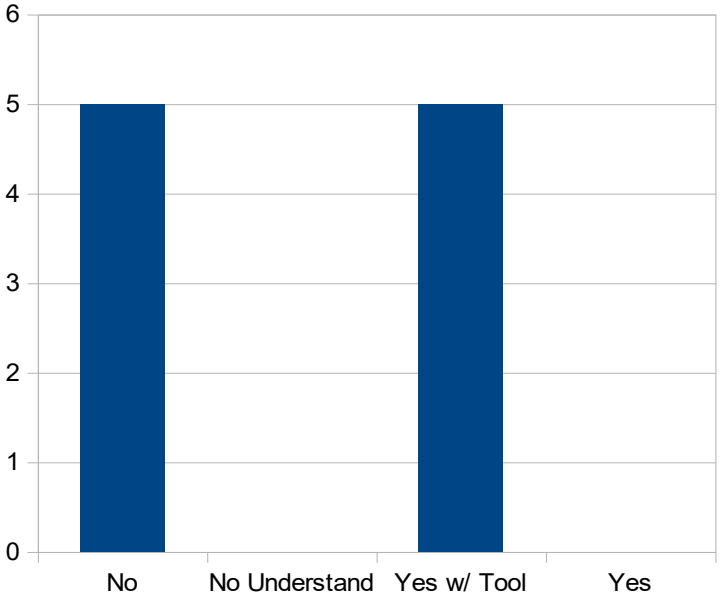
[Entry and exit criteria of the activities indicate when the activity begins and ends, respectively. Example: An activity can be started only after the availability of human resources and hardware resources needed for its execution.]



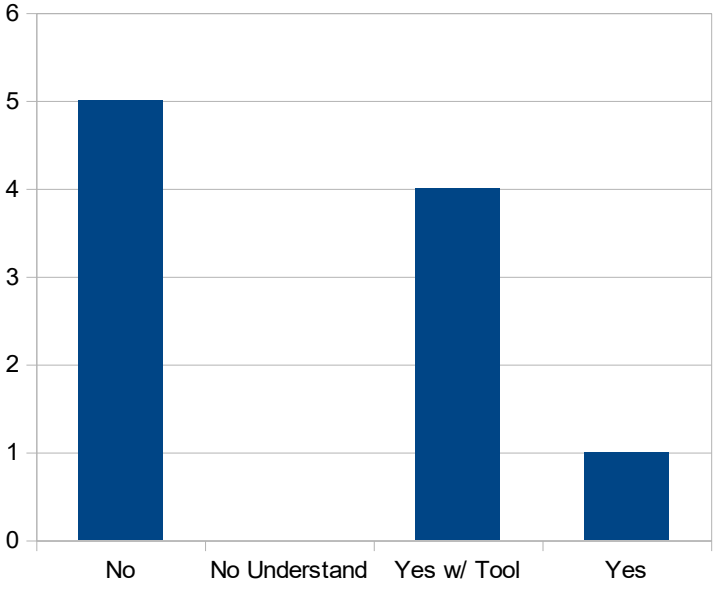
[The activities are organized in a sequence, so that it is clear when one activity is performed relative to the other activities.]



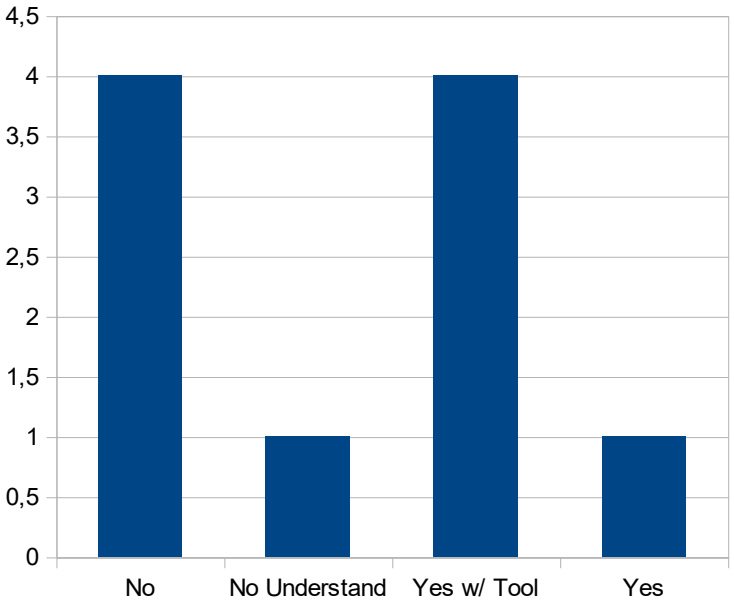
[Goal coordinates the execution of activities.]



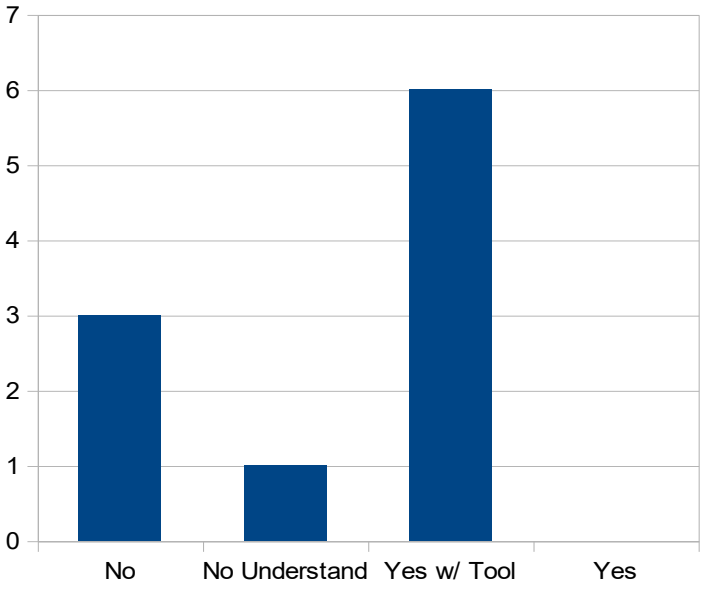
[The practitioners perceive the activities carried around.]



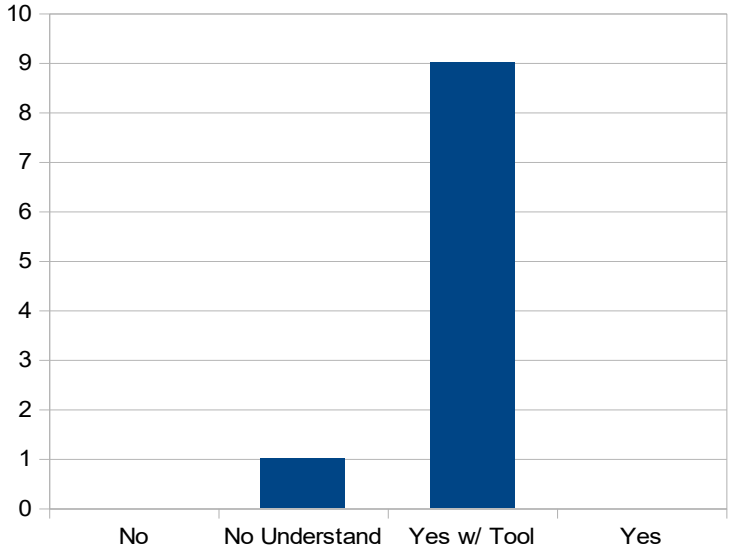
[Methods are a set of steps or instruction to be performed in an activity.]



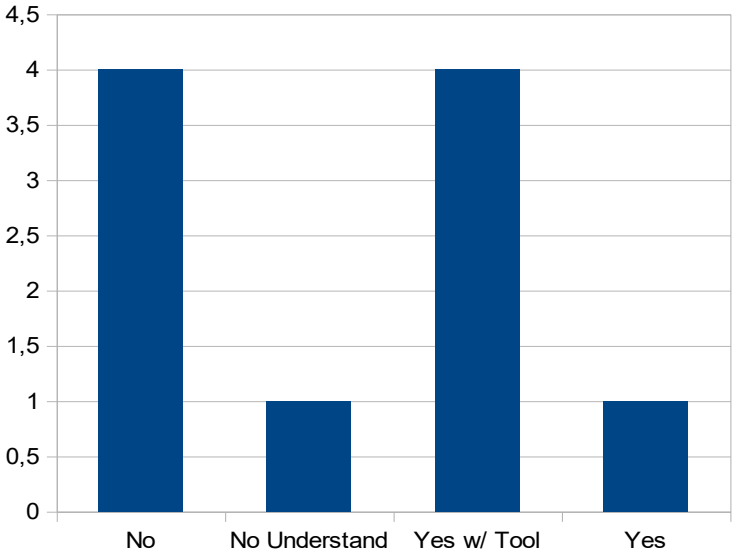
[Guidelines describe how to use a software tool in the accomplishment of an activity.]



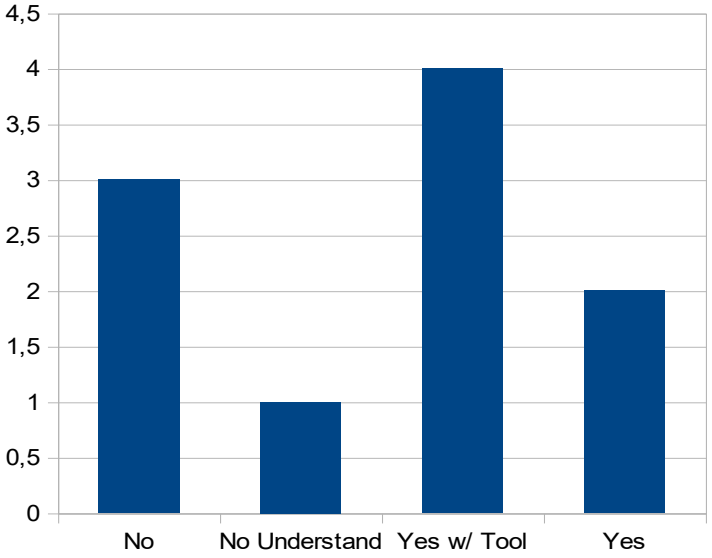
[Templates are models to be followed when preparing an artifact in an activity.]



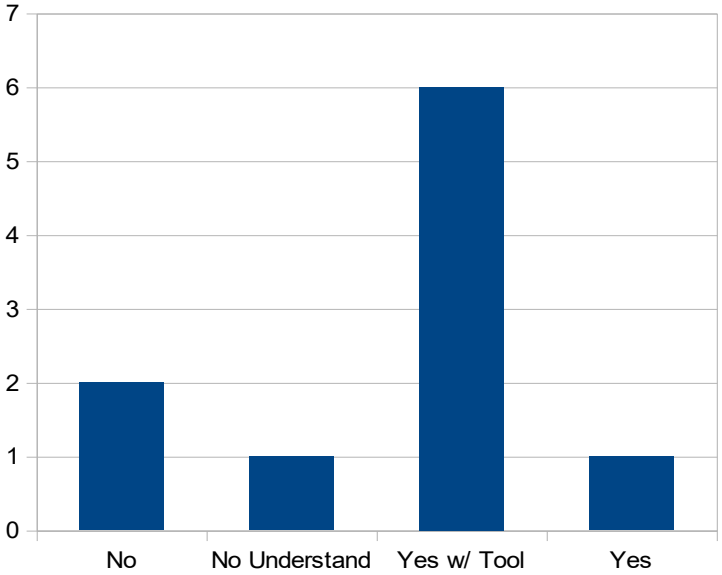
[The activities executed produce as a result an artifact.]



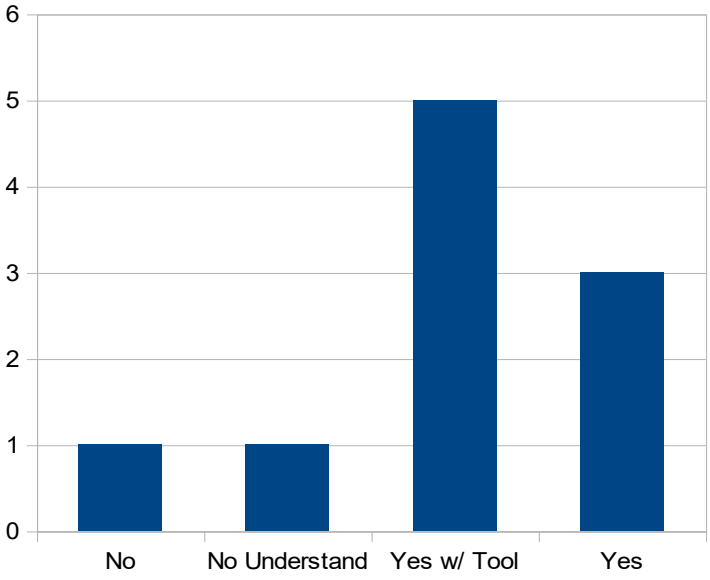
[The activities need artifacts for its execution.]

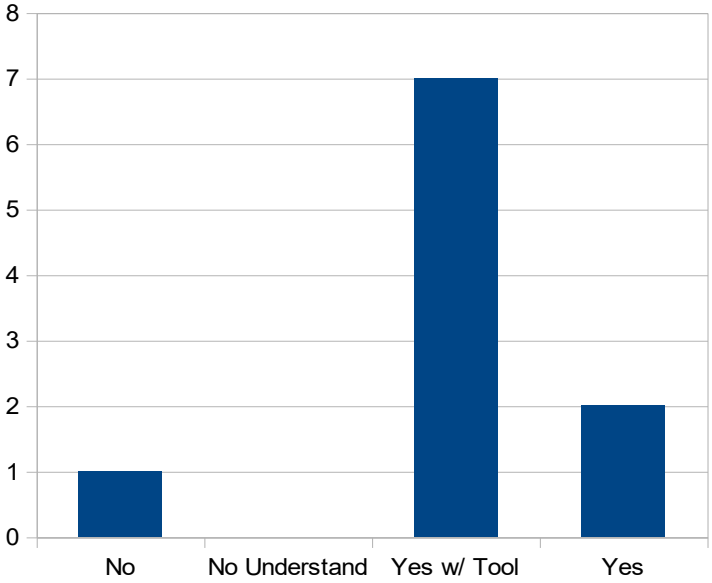


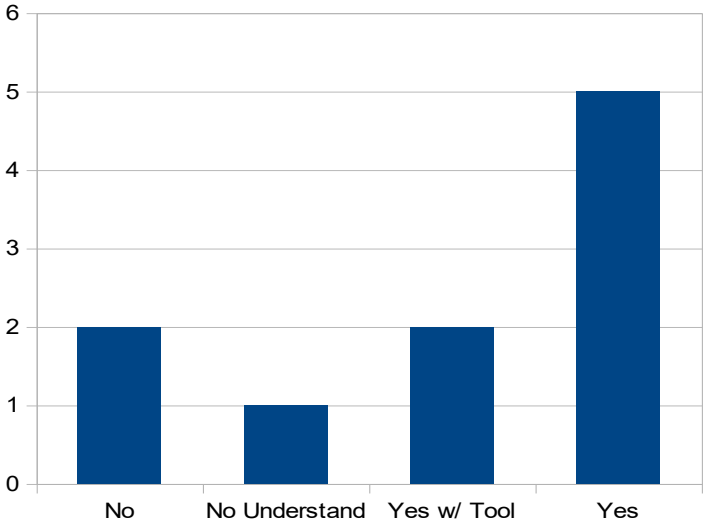
[Artifacts are subject to a set of constraints. Example: Quality requirements, such as format and number lines of code]



[An activity requires Human resources (ex.: people, role).]


6
5
4
3
2
1
0
No
No Understand
Yes w/ Tool
Yes
[An activity requires Hardware resources (ex.: equipment).]


8
7
6
5
4
3
2
1
0
No
No Understand
Yes w/ Tool
Yes
[An activity requires Software resources (ex.: tool).]


6
5
4
3
2
1
0
No
No Understand
Yes w/ Tool
Yes
[Resources are subject to a set of constraints. Example: Availability of team members, technical knowledge, availability of equipment, software use license, among others.]

APÊNDICE B – Lista de Ferramentas

Este apêndice apresenta o levantamento dos dados que são armazenados por cada ferramenta identificada nas execuções da Pesquisa de Opinião (Capítulo 3). Os seguintes dados de cada ferramenta foram registrados: Nome, Descrição, Estrutura, Licenciamento e se possui Recurso de Exportação.

Tabela B.1 – Dados das ferramentas identificadas nas execuções da Pesquisa de Opinião

Nome	**Descrição**	**Estrutura**	**Licenciamento**	**Recurso de Exportação**
Bugzilla	Bugzilla é uma ferramenta baseada em Web e e-mail que dá suporte ao desenvolvimento do projeto Mozilla, rastreando defeitos e servindo também como plataforma para pedidos de recursos.	Produto Reporter Componente Descrição do Componente Versão Severity Hardware Sistema Operacional Sumário Descrição Anexos	Gratuito	Sim
CVS	O CVS, ou Concurrent Version System (Sistema de Versões Concorrentes) é um sistema de controle de versão que permite que se trabalhe com diversas versões de arquivos organizados em um diretório, localizados local ou remotamente, mantendo-se suas versões antigas e os logs de quem e quando manipulou os arquivos.	Versão Commiter Data Modificações	Gratuito	Não
GIT	GIT é um sistema de controle de versão distribuído e um sistema de	Commiter	Gratuito	Não

	gerenciamento de código fonte, com ênfase em velocidade.	email Arquivos Modificados Inserções Remoções Comentário Data Completa		
GitHub	GitHub é um Serviço de Web Hosting Compartilhado para projetos que usam o controle de versionamento Git.	Commiter Arquivos Adições Remoções Título Comentário Labels Milestone Assignee (Atribuir a)	Gratuito ou Pago (Dependendo das funcionalidades utilizadas)	Não
GitLab	O GitLab é um gerenciador de repositório de software. GitLab é similar ao GitHub, mas o GitLab permite que os desenvolvedores armazenem o código em seus próprios servidores, ao invés de em servidores operados pelo GitHub.	Commiter Arquivos Adições Remoções Título Descrição Atribuir a Data de vencimento Etiqueta (Label)	Gratuito (Community) Pago (Enterprise)	Sim
Google Code	Google Code é um site da companhia Google para interesse de programadores em desenvolvimento de softwares. O Google oferece uma variedade de APIs para programas de internet e para desktop para programadores, incluindo produtos como Google AdSense, Google Maps, Google Checkout e Google Toolbar.	D Tipo Status Prioridade Milestone Dono Summary Labels Anexos	Gratuito (Descontinuado)	Não

		Estrelas Aberto em Fechado em Modificado em BlockedOn Blocking Blocked MergedInto Reporte CC Projeto		
Jazz RTC	Rational Team Concert é uma ferramenta de colaboração de equipe para desenvolvimento de software desenvolvido pela marca Rational Software da IBM, lançada 2008. Ele fornece um ambiente colaborativo, n qual as equipes de desenvolvimento de software usam para gerenciar todos os aspectos de seu trabalho, tais como planos, tarefas, controle de revisão, gerenciamento de construção e relatórios.	Summary Tipo Severidade Encontrado em Como foi encontrado Regression (CheckBox) Regression From Sistema Operacional Cliente Data de Criação Criado por Área de Projeto Área de Equipe Filed Against Tags Número APAR Possuído por Prioridade Planejado para Estimativa Correção Tempo Gasto Data de Vencimento Descrição	Pago	Não

		Discussão		
Jenkins	Jenkins é uma ferramenta de integração contínua, de código aberto e escrita em Java.	Projeto Descrição Opções e configurações de Builds Gerenciamento de Código Fonte (CVS, CVS Projectset, Subversion) Arquivos	Gratuito	Sim
JIRA	O Jira é uma ferramenta que permite o monitoramento de tarefas e acompanhamento de projetos garantindo o gerenciamento de todas as suas atividades em único lugar. A ferramenta também apresenta painéis de controle adaptativo, filtros de pesquisa, estatísticas, RSS e email.	Summary (Nome) Tipo Assignee (Cessionário) Reporter Prioridade Data de Vencimento Componentes Descrição Anexos Labels Estimativa Original (Quantidade de Trabalho) Estimativa Restante Comentários	Pago	Sim
Mantis Bug Tracker	Mantis Bug Tracker é uma ferramenta baseada na web que tem como principal função gerenciar defeitos de outros softwares.	Categoria Frequência Gravidade Prioridade Atribuir a Resumo Descrição Passos para Reproduzir Informações Adicionais Arquivos Visibilidade	Pago	Sim

Maven	Maven é uma ferramenta de automação de compilação utilizada primariamente em projetos Java. O projeto Maven é hospedado pela Apache Software Foundation, que fazia parte do antigo Projeto Jakarta. O Maven utiliza um arquivo XML (POM) para descrever o projeto de software sendo construído, suas dependências sobre módulos e componentes externos, a ordem de compilação, diretórios e plugins necessários. Ele vem com objetivos pré-definidos para realizar certas tarefas, como compilação de código e seu empacotamento.	Estrutura do POM (Project Object Model) Project Model Version Group ID Artifact ID Packaging Version Name URL Description	Gratuito	Não
Mercurial	Mercurial é uma ferramenta multiplataforma de controle de versão distribuído para desenvolvedores de software.	Changeset Tag Usuário Data Summary Parent Arquivos (+ ou) Descrição Manifest	Gratuito	Não
Redmine	O Redmine é uma aplicação web flexível para gerenciamento de projeto. Escrito utilizando o framework Ruby on Rails, é multiplataforma e multidatabase. Redmine é um software de código aberto e lançado sobre os termos da GNU General Public License (GPL).	Tipo Título Descrição Situação Tarefa pai Prioridade Atribuído para Início Data prevista Tempo estimado % Terminado Arquivos	Gratuito	Sim

		Observadores		
Sourceforge	SourceForge é um repositório de código fonte baseado em Web. Ele atua como um centro para desenvolvedores gerenciarem projetos livres e de código aberto colaborativamente.	Milestone Status Owner Labels Prioridade Última Atualização Criado Criador Privado Arquivos Análises Wiki Git SVN Mercurial	Não	Sim
Stack Overflow	Stack Overflow é um website que apresenta perguntas e respostas de uma grande quantidade de tópicos de programação de computadores.	Usuário Pergunta Tags Votos Respostas Visitas"	Gratuito	Não
SVN	O SVN funciona de maneira bem simples. Basicamente existe um repositório, o qual armazena os arquivos. Você pode modificálo utilizando um commit. E, para futuros logs, ele armazena informações básicas sobre o commit, como: versão, usuário que fez a modificação, a data completa (datetime) e o que foi modificado.	Identificador Repository_id Revision Committer Committed_on Comments Commit_date	Gratuito	Não
TFS	Um servidor de nível empresarial para que as equipes compartilhem códigos, rastreiem trabalhos e entreguem software. Permite a criação de projetos utilizando ferramentas auxiliares como Agile,	Title Order Work Item Type	Gratuito (Community) Pago	Sim

	CMMI, Scrum, Git, etc.	State Story Points Value Area Iteration Path Tags Colunas Personalizadas	(Enterprise)	

APÊNDICE C – Questionário Aplicado na Entrevista de Verificação das Operações

Este apêndice apresenta o questionário utilizado na entrevista aplicado a gerentes de projeto, com o objetivo de verificar a ocorrência das operações de instanciação no contexto da indústria.

Questionário

Nome:
Organização:

1. A organização possui um processo de software definido?

1.1 Em qual notação o processo de software do projeto é apresentado ao time?

Notação: ( ) UML ( ) SPEM ( ) BPMN ( ) Linguagem Natural (texto)

2. O processo de software definido sofre adaptações para um projeto específico?

3. A partir do processo de software do projeto é feito o planejamento das tarefas necessárias para a condução do projeto de desenvolvimento de software? Se sim, descreva como é realizado. (Ou seja, como é realizado o planejamento de tarefas??)

4. Qual a periodicidade da criação de tarefas?

( ) Somente no início do projeto
( ) Semanalmente
( ) Mensalmente
( ) Por *sprint*
( ) Por versão
( ) A qualquer momento
( ) Outro

5. A organização utiliza algum apoio ferramental para criação de tarefas? Se sim, qual?

6. Quais dados são informados na criação da tarefa?

7. Dados são registrados durante a realização da tarefa? Se sim, quais?

8. Quais dados são informados no encerramento da tarefa?

9. Dentre os dados já informados, quais dados são obrigatórios para a criação e encerramento da tarefa?

10. Na criação das tarefas é realizado algum mecanismo de associação da tarefa com a atividade do processo de software do projeto?

Se sim, como é feito?

Se não, seria importante para o gerenciamento do projeto?

11. Ao analisar um plano de projeto, que contém as tarefas criadas, é possível identificar um conjunto de relações entre as atividades do processo e as tarefas. Assinale abaixo, as ações que você observa que são realizadas durante a criação das tarefas.

a. ( ) Uma atividade do processo é criada como uma única tarefa.
b. ( ) Uma atividade do processo torna-se várias tarefas.
c. ( ) Várias atividades do processo são agrupadas um uma única tarefa.
d. ( ) Uma atividade do processo é criada com um nome diferente.
e. ( ) Um atividade do processo não é criada.
f. ( ) Uma tarefa criada não tem nenhuma atividade correspondente no processo.
g. ( ) Uma atividade do processo é criada com mais detalhes.
h. ( ) Várias atividades relacionadas do processo são distribuídas por meio de várias iterações.
i. ( ) Um conjunto de atividades do processo são referenciadas a uma instância do processo no plano de projeto.
j. ( ) Uma tarefa que representa parcialmente uma atividade correspondente no processo.

12. Outras ações, não apresentadas acima, são observadas durante a criação das tarefas? Se sim, quais?

APÊNDICE D – Manual da Ferramenta

Este apêndice apresenta o manual da ferramenta.

Prefácio

Este pequeno manual se propõe a descrever, da forma mais breve e objetiva possível, um guia, par aque seja, possível executar algumas das mais importantes funcionalidades da ferramenta MapTracys. Uma versão da MapTracys pode ser acessada pelo seguinte endereço https://maptracys-test.herokuapp.com/.

O manual possui apenas três partes distintas, sendo ainda uma versão primária que, posteriormente, poderá vir a ser expandido. Logo de início temos uma a seção ***Introdução*** que falará um pouco sobre a ferramenta e sobre a pesquisa em que esta foi baseada. Na sequência, temos a seção ***Mapeamento entre as perspectivas de processo e projeto***, que trata do ciclo de vida das tarefas, como são criadas e assumem diferentes estados durante a execução das iterações e, consequentemente, do projeto. Então temos a seção ***Mapeamento para Identificação do Processo de Software Executado***, em que é apresentada a forma como a ferramenta faz a identificação dos processos a partir das informações exportadas do projeto. Para finalizar, temos a seção ***Relatórios*** que apresentará todos os possíveis relatórios que a ferramenta executa, como também outras funcionalidades que permeam os relatórios.

Para melhor compreensão das explicações, foram adicionadas imagens que representam momentos citados durante o texto, já que o mesmo possui na maior parte do tempo uma estrutura linear e algorítmica.

Introdução

A ferramenta foi desenvolvida como parte do trabalho da pesquisadora **Renata Mesquita da Silva Santos** para obtenção do grau de Doutora.

A ferramenta visa apoiar de forma semi-automatizada a atividade de Garantia da Qualidade de Processo, a partir de cenários de execução de processo de desenvolvimento de software, por meio da implementação do processo definido.

Para isso foram implementadas diversas funcionalidades que foram descobertas durante as pesquisas na literatura que cobre o assunto de garantia de qualidade. O resultado deste estudo possibilita apoiar os Gerentes de Projeto na realização de ações de Garantia da Qualidade de Processo, a partir de cenários de execução de processo de desenvolvimento de software, possibilitando a identificação de não conformidades entre a execução dos processos e os planos e recursos definidos.

A ferramenta em desenvolvimento é capaz de gerar análises sobre: a cobertura do processo; a identificação de não conformidade entre o processo esperado e efetivamente executado, bem como identificação de alterações na ordem de execução das tarefas e de oportunidades de melhoria do processo. Além dos relatórios citados, ainda é possível identificar processos executados, através dos dados exportados de um projeto executado utilizando a ferramenta.

Mapeamento entre as perspectivas de processo e projeto

Criação da Tarefa

A abordagem de mapeamento entre as representaçoes processos de software e os projetos de software é apoiada pelas operações de instanciação, onde é realizada a representação de elementos do processo de software no plano do projeto, e em consequência nos registros de execução. Esta representação é feita durante a instanciação do processo de software definido no projeto de software. O mapeamento se dá entre modelos de processos e planos de projetos com base em operações explícitas de instanciação que capturam a lógica de mapeamento entre processo e projeto.

Pré-Requisitos:

- A importação de um processo;
- A criação de um projeto;
- Ao menos uma iteração definida; e
- Ao menos uma instância criada.

A seguir são apresentados os passos para execução deste processo e a definição de alguns termos:

- Clique em ***Home*** e então clique no nome do projeto;
- Na página do projeto clique em ***New task***;
- Selecione então:
    - Uma iteração;
    - Uma instância;
    - Uma operação de instanciação; e
    - Uma atividade.

Para finalizar esta etapa, clique em ***Submit***.

- Agora é necessário preencher os dados para:
    - O papel da tarefa;
    - Descrição da tarefa, caso necessário;
    - Prioridade da tarefa, caso necessário;
    - Categoria da tarefa, caso necessário;
    - Data esperada para seu início;
    - Comentários sobre a tarefa, caso necessário;
    - Tempo estimado para sua execução; e
    - Artefatos da tarefa, caso necessário (podendo ser criada um do zero ou importado da atividade vinculada a tarefa).

Após preencher estes dados, clique em ***Create Task***.

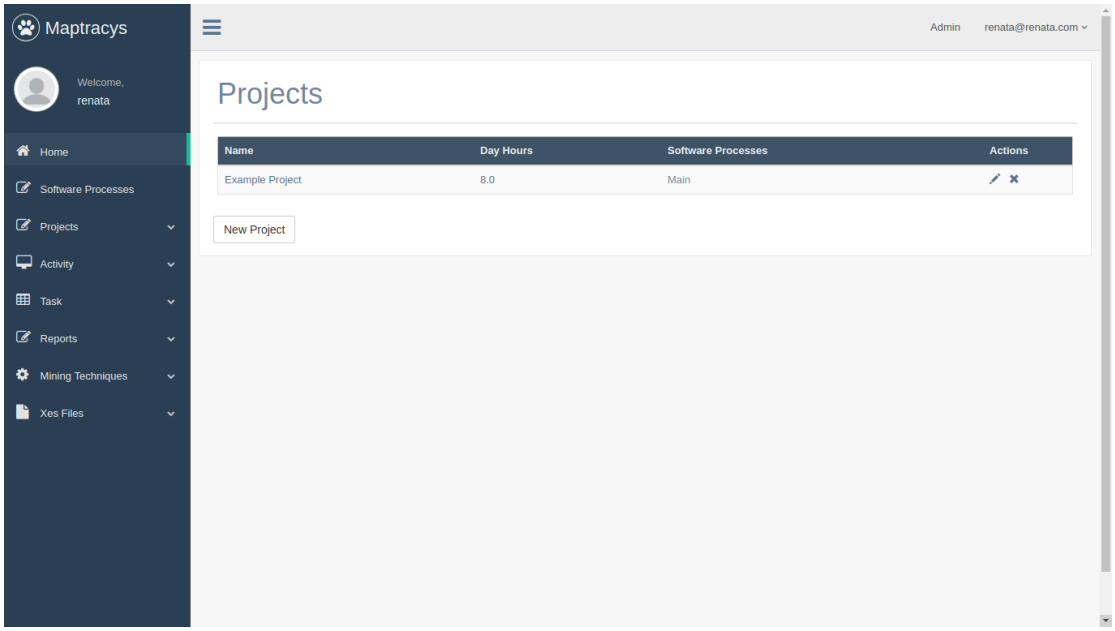


Figura 1: tela de seleção do projeto

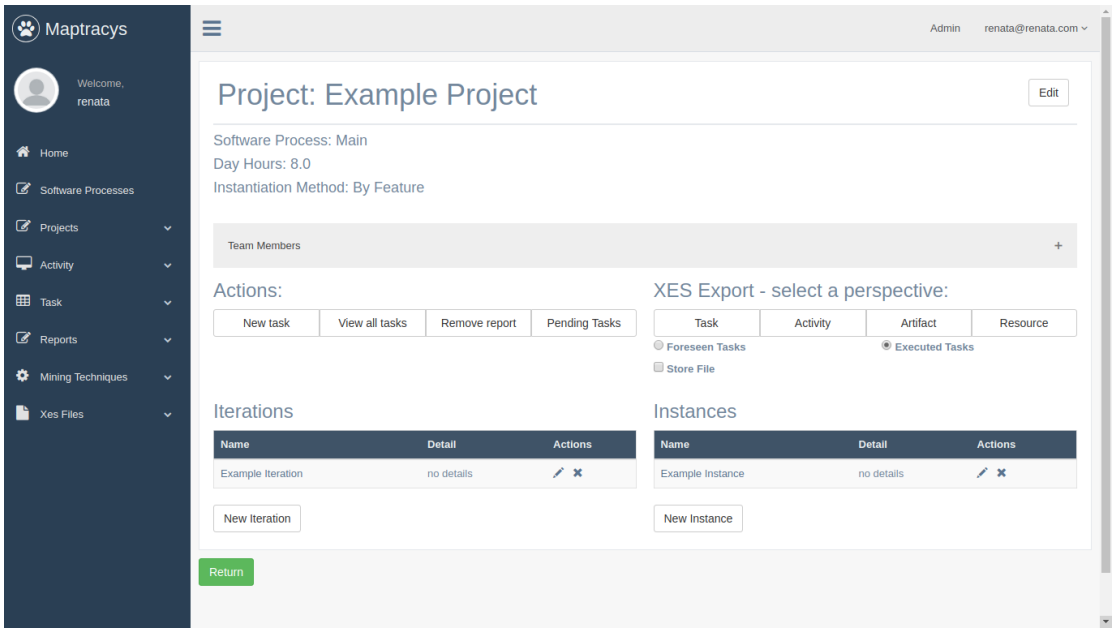


Figura 2: tela de exibição do projeto

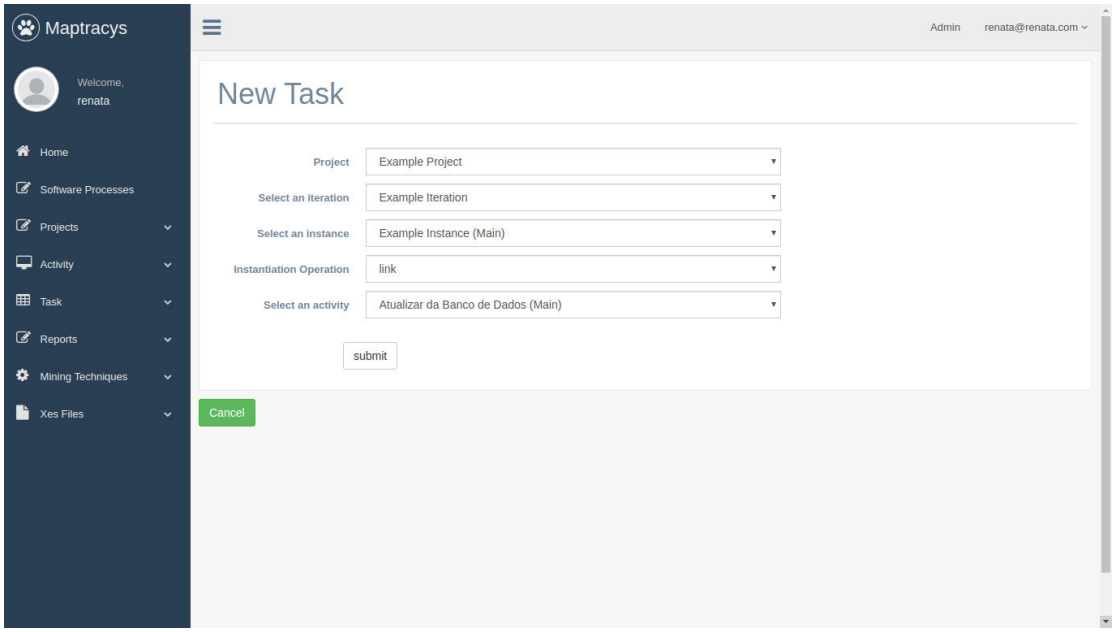


Figura 3: tela inicial de criação de tarefa

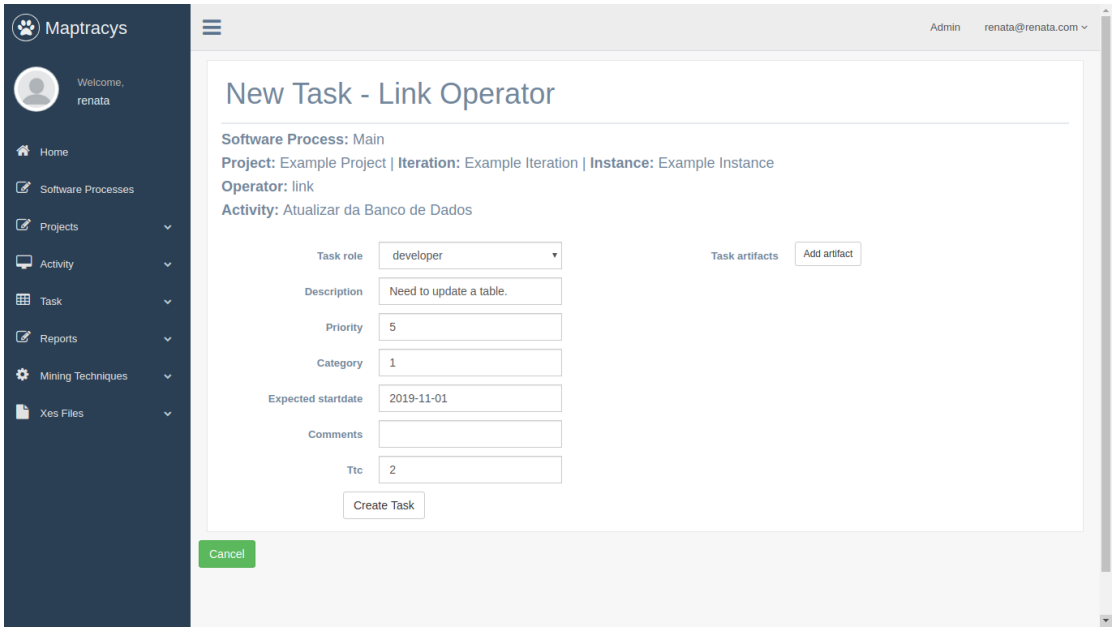


Figura 4: tela final de criação de tarefa

Ao clicar em ***Create Task***, será redirecionado para a página de listagem de tarefas.

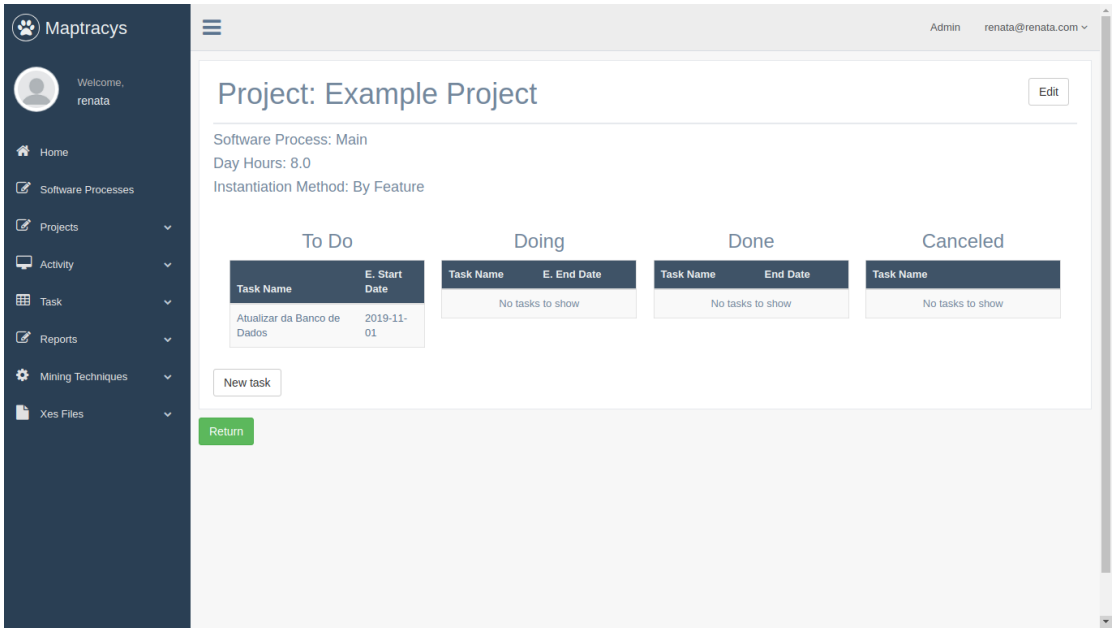


Figura 5: tela de listagem das tarefas de um projeto

Finalização da Tarefa

Na ***figura 5*** é apresentada a página de listagem de tarefas de um determinado projeto. Nesta página, clique no nome de uma tarefa e será redirecionado para a página de exibição da tarefa.

Na página de exibição da tarefa existem diferentes botões de ação, eles são:

- ***Edit:*** para modificar dados referentes a tarefa;
- ***Delete:*** para deletar a tarefa atual;
- ***Take this Task:*** para atribuir a tarefa ao atual usuário;
- ***Assign a Non-User Actor:*** para atribuir a tarefa a um outro usuário;
- ***Cancel This Task:*** para cancelar a tarefa atual.

Ao clicar em ***Take this Task*** ou em ***Assign a Non-User Actor***, será apresentada um pedido de confirmação, confirme e a tarefa terá mudado seu status de ***To-Do*** para ***Doing***, tendo a data atual como data efetiva do início da execução.

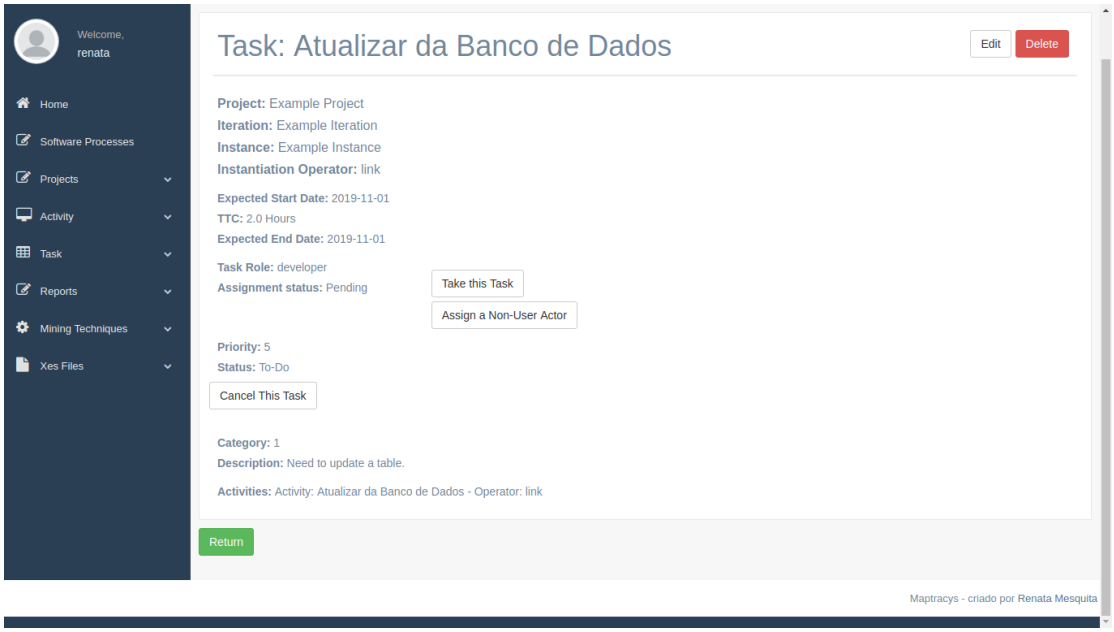


Figura 6: tela de exibição da tarefa (*To-Do*). Fonte: A Autora.

A página agora não terá mais as opções ***Take this Task*** e ***Assign a Non-User Actor***, em seu lugar estará a opção ***Mark as Done***. Clique nesta opção, será solicitada a confirmação da ação, confirme e a tarefa será finalizada, tendo como data de fim efetivo a data atual.

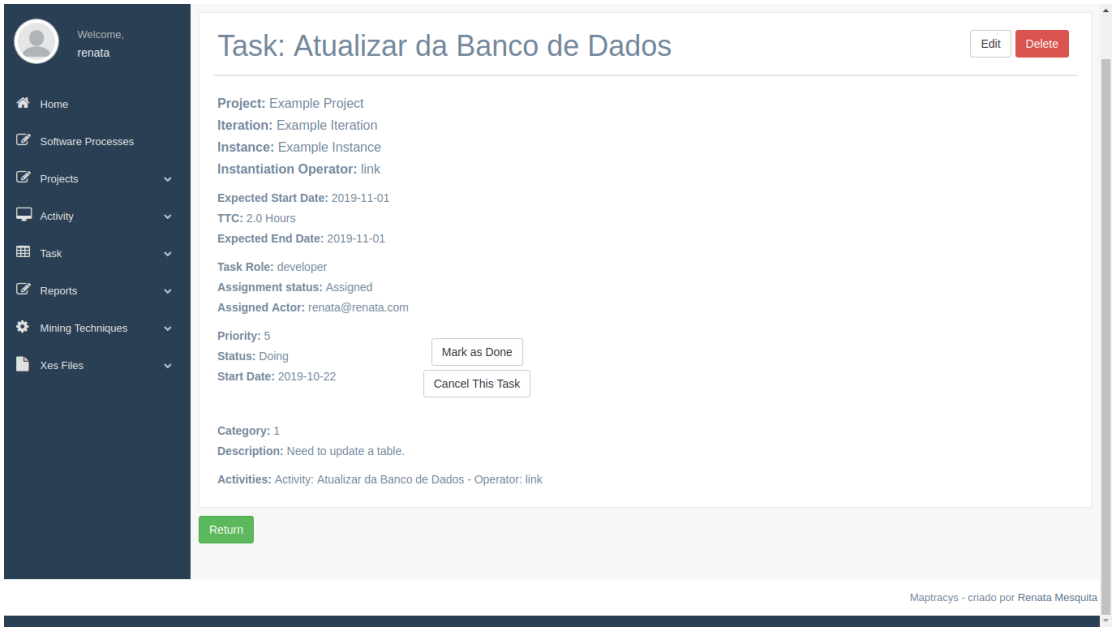


Figura 7: tela de exibição da tarefa (*Doing*). Fonte: A Autora.

Após clicar em ***Mark as Done***, a página será recarregada e é possível observar as informações referentes a tarefa, sendo esta agora identificada como ***Done***.

Figura 8: tela de exibição da tarefa (*Done*). Fonte: A Autora.

Mapeamento para Identificação do Processo de Software Executado

Obter os Registros de Execução do Processo

Na página de exibição do projeto, na seção ***XES Export*** (figura 2), estão listadas as quatro possíveis perspectivas para exportação do processo e abaixo das perspectivas estão dois seletores que identificam o momento esperado para exportação.

XES é um padrão baseado em XML para logs de eventos. Seu objetivo é fornecer um formato geralmente reconhecido para o intercâmbio de dados do log de eventos entre ferramentas e domínios de aplicativo. Seu principal objetivo é a mineração de processos, ou seja, a análise de processos operacionais com base em seus logs de eventos. No entanto, o XES foi projetado para também ser adequado para mineração geral de dados, mineração de texto e análise estatística.

As perspectivas são:

- **Task:** Ao clicar nesta opção, será exportado um arquivo XES contendo as informações referentes a todas as tarefas planejadas ou executadas, de acordo com o seletor escolhido, de cada instância definida no projeto;
- **Activity:** Ao clicar nesta opção, será exportado um arquivo XES contendo as informações referentes a todas as atividades planejadas ou executadas, de acordo com o seletor escolhido, de cada instância definida no projeto;
- **Artifacty:** Ao clicar nesta opção, será exportado um arquivo XES contendo as informações referentes a todos o artefatos definidos para as tarefas planejadas ou executadas, de acordo com o seletor escolhido, de cada instância definida no projeto;

- **Resource:** Ao clicar nesta opção, será exportado um arquivo XES contendo as informações referentes a todos os papéis definidos para as tarefas planejadas ou executadas, de acordo com o seletor escolhido, de cada instância definida no projeto.

Os momentos esperados são:

- **Planned:** Quando esta opção está selecionada, todos os elementos, referentes a opção selecionada na perspectiva, que foram planejados serão exportados;
- **Executed:**Quando esta opção está selecionada, apenas os elementos, referentes a opção selecionada na perspectiva, que foram executados serão exportados.

Ainda é possível, ao invés de exportar o arquivo, selecionando a opção ***Store File***, armazená-lo no banco de dados para uso futuro.

Obter o Processo de Software Executado

Pré-Requisitos:

- Arquivo .xes já importado.

Selecionando a opção ***Mining Techniques*** no menu lateral**,** clique em ***Heuristic Mining***. Após o carregamento da nova página, clique em ***New Heuristic Mining***.

Digite o valor do ***Threshold*** e ***Frequency Wish*** e selecione o arquivo ***XES*** do modelo que se deseja realizar a mineração. Após completar os procedimentos descritos anteriormente, clique em ***Execute*** e será exibida uma nova página contendo a representação gráfica do processo identificado.

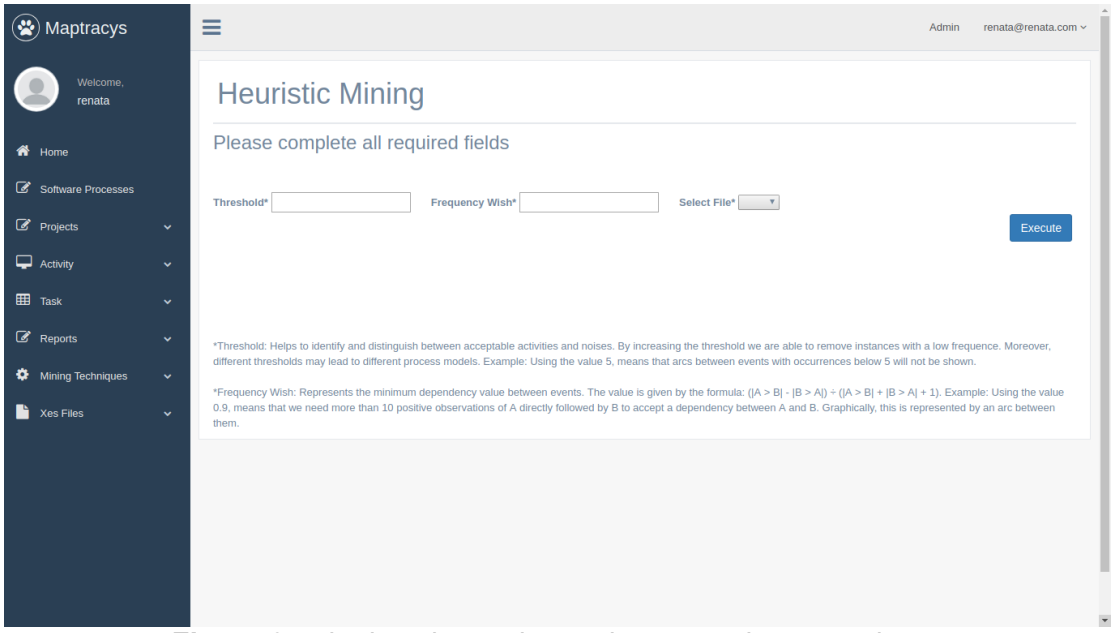


Figura 9: tela de seleção de arquivo para mineração de processo

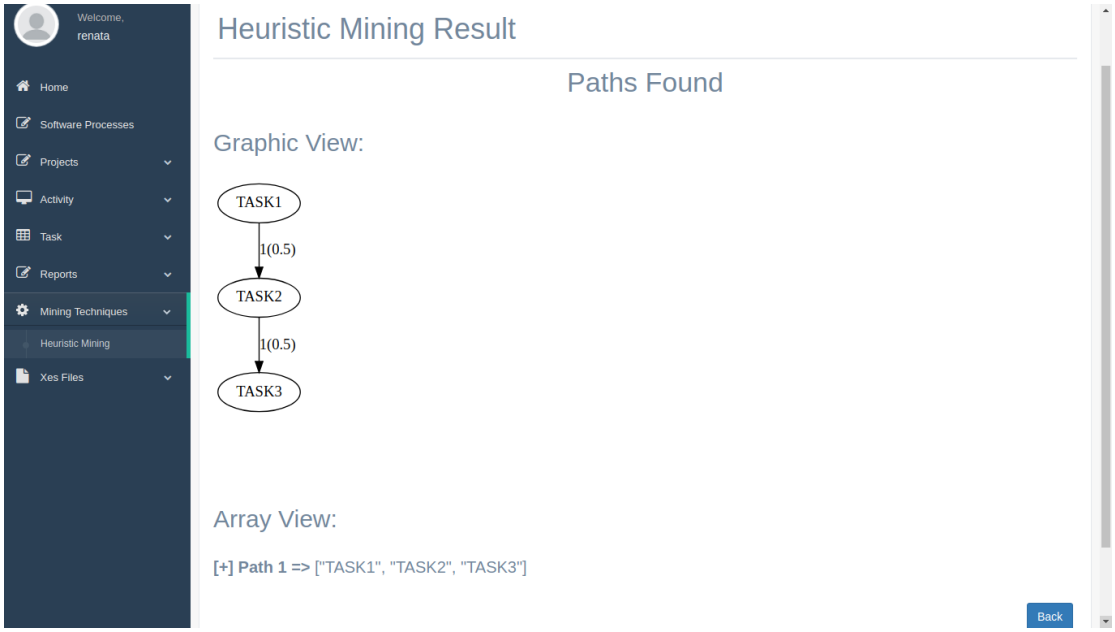


Figura 10: tela de exibição do resultado da mineração

Caso nenhuma representação gráfica seja gerada após a execução dos passos descritos, altere os valores informados do *threshold* e frequência, pois eles estão fornecendo um valor alto de corte, sendo impossível gerar um caminho que satisfaça essas condições.

Definições:

- **Heuristic Mining:** o objetivo do *Heuristic Mining* é que caminhos pouco frequentes não sejam ser incorporados ao modelo de processo fornecido pelo usuário;
- **Threshold:** É o valor mínimo de repetição de dois eventos adjacentes quaisquer;
- **Frequency Wish:** É o valor mínimo (entre 0 e 1) da relação de dependência entre dois eventos quaisquer.

A ferramenta fornece uma tabela de registros, na seção ***Heuristic Mining***, contendo todas as minerações realizadas. Nessa tabela é fornecida o nome do usuário que a executou, o nome do arquivo utilizado, os valores de threshold e da frequência, a data de execução da mineração e o resultado. O usuário pode ainda, caso deseja, deletar o registro da mineração.

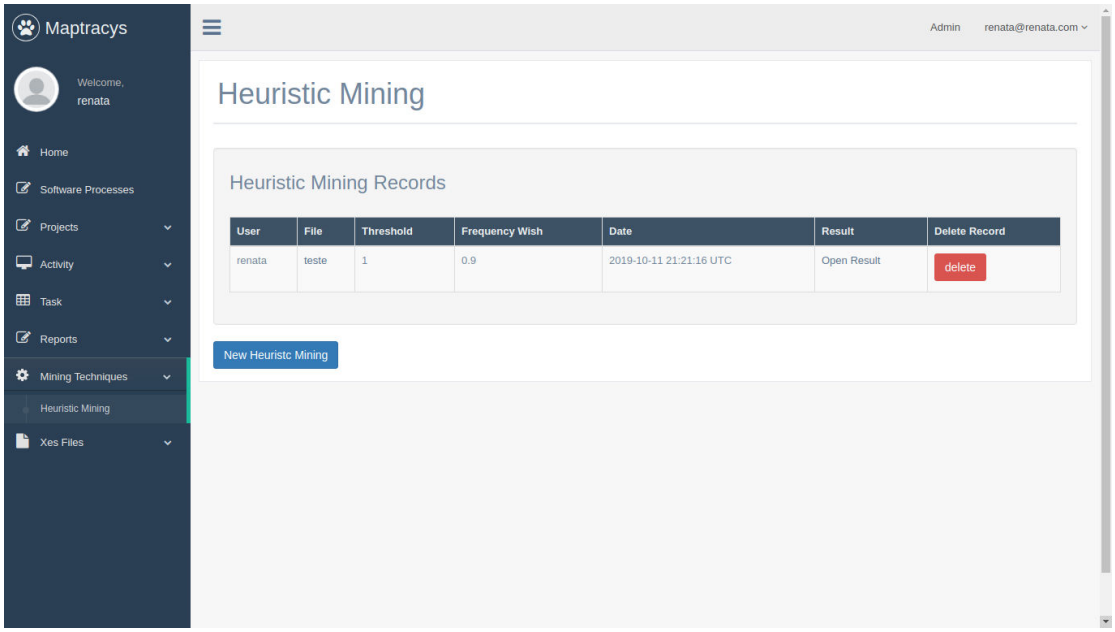


Figura 11: tela contendo a tabela de registro das minerações realizadas

Relatórios

Análise de Cobertura

A Análise de Cobertura é um relatório que tem como retorno o grau de completude do projeto em porcentagem. A análise é dividida em duas partes:

- **A análise de cobertura geral:** considera todas as atividades listadas no processo para sua quantificação e pode ser subdividida em outras três partes:
    - **Todas as Atividades Obrigatórias do Processo:** lista todas as atividades definidas no processo;
    - **Atividades Executadas:** lista todas as atividades do processo que já foram executadas;
    - **Atividades Não Executadas:** lista todas as atividades do processo que não foram executadas; e
- **A análise de cobertura por caminhos:** leva em consideração as atividades divididas em caminhos de execução específicos. Projetos podem possuir diversos caminhos possíveis para que se chegue ao resultado final, este fato que é levado em consideração para elaboração desta parte do relatório. Esta análise ainda pode ser subdividida em três partes:
    - **Atividades Esperadas no Caminho:** lista todas as atividades de um caminho de execução específico;
    - **Atividades Executadas:** lista todas as atividades que já foram executadas em um caminho específico;

Atividades Não Executadas: lista todas as atividades que não foram executadas em um caminho específico.

Para gerar um relatório de cobertura, primeiramente clique em ***Reports***, no menu lateral, após a exibição das opções desta seção, clique em ***Conformance Analysis***. Ao clicar nesta opção, abrirá uma nova página. Na página ***Conformance Analysis*** na primeira aba existem dois campos: ***Project*** e ***Software process***; escolha um projeto e um processo de software dentre os que estão disponíveis e, após isso, clique no botão ***Coverage Analysis***.

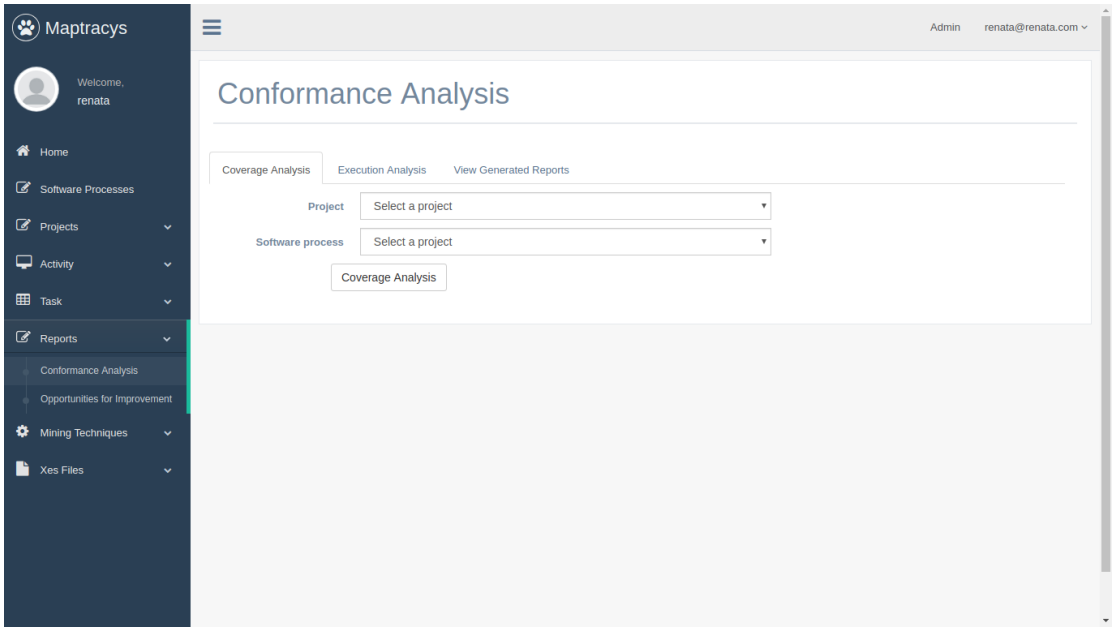


Figura 12: tela de seleção do relatório a ser executado (aba **Coverage Analysis**)

O resultado desse procedimento será a exibição de uma nova página exibindo o relatório da análise de cobertura. Essa página é composta pelo título da página, logo abaixo são exibidos o projeto, o processo, a iteração e a instância que geraram esta análise (neste relatório em específico, são utilizadas todas as iterações e todas as instâncias pertencentes ao projeto) e por último está a exibição dos dados, como explicados nos parágrafos anteriores.

Ao final dos resultados da análise está um botão, cuja função é salvar o estado da análise atual. Ao clicar no botão ***Create Report*** a análise será armazenada para rastreamento da evolução da execução e futura análise manual por parte do gerente de projeto.

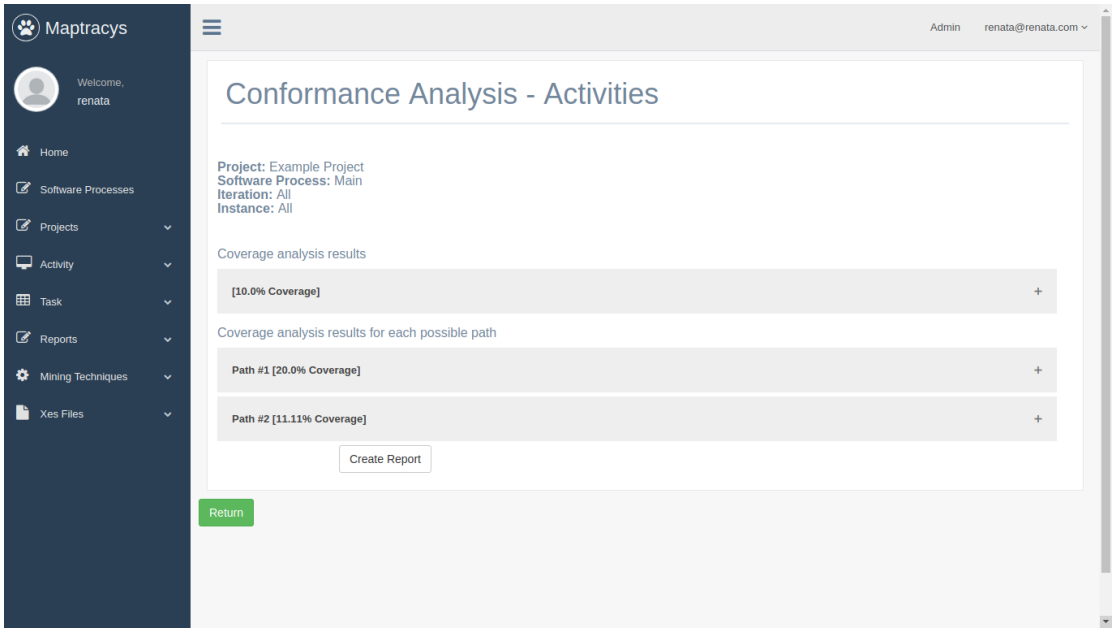


Figura 13: tela de exibição da análise de cobertura

Análise de Ordem

Este relatório é baseado numa análise ponto a ponto das tarefas. Esse relatório avalia para cada tarefa se esta foi executada de acordo com a atividade anterior e com a próxima. O relatório é dividido da seguinte forma:

- Uma tabela contendo o processo de software e a data de início da tarefa;
- **A próxima atividade:** contém uma tabela composta pela próxima tarefa da atividade, pela próxima atividade, pelo nome da próxima tarefa e se a próxima tarefa da primeira coluna está de acordo com o que era esperado; e
- **A atividade anterior:** contém uma tabela composta pela tarefa da atividade anterior, pela atividade anterior, pelo nome da tarefa anterior e se a tarefa anterior da primeira coluna está de acordo com o que era esperado.

Para gerar um relatório de ordem, primeiramente clique em ***Reports***, no menu lateral, após a exibição das opções desta seção, clique em ***Conformance Analysis***. Ao clicar nesta opção, abrirá uma nova página. Na página ***Conformance Analysis***, na segunda aba, existem quatro campos: ***Project***, ***Iteration***, ***Instance*** e ***Software process***; escolha um projeto, uma iteração, uma instância e um processo de software, dentre os que estão disponíveis (neste relatório é possível escolher para que seja executado com base em todas as iterações, todas as instâncias e todos os processos, todos ao mesmo tempo ou combinados de acordo com a necessidade) e após isso, clique no botão ***Order Analysis***.

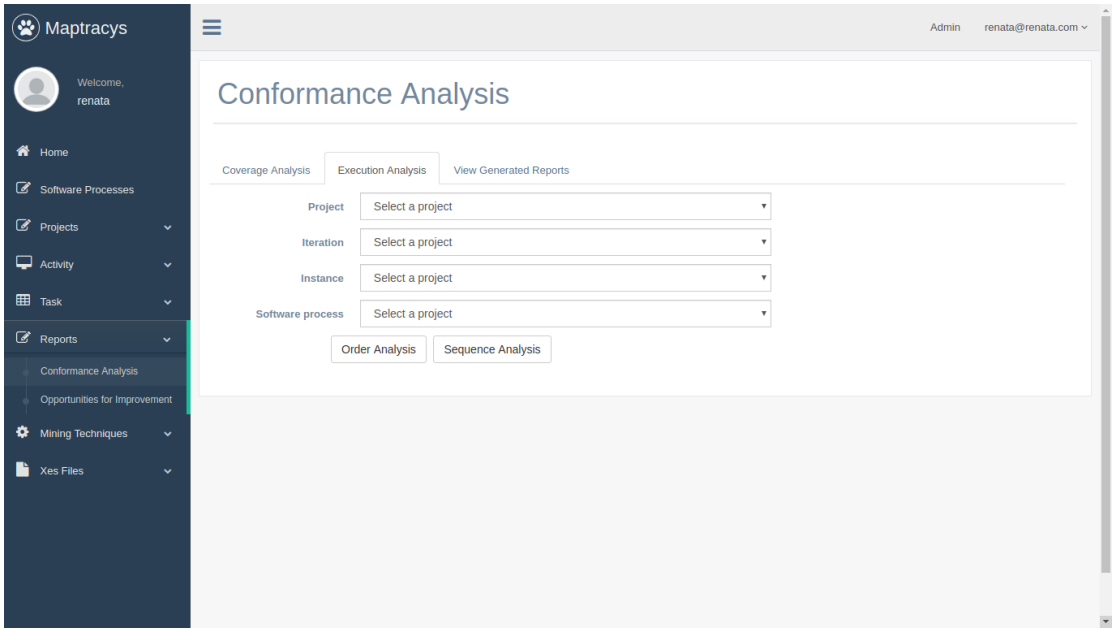


Figura 14: tela de seleção do relatório a ser executado (aba **Execution Analysis**)

O resultado desse procedimento será a exibição de uma nova página exibindo o relatório da análise de ordem. Essa página é composta pelo título da página, logo abaixo são exibidos o projeto, o processo, a iteração e a instância que geraram esta análise (neste relatório podem ser utilizados todos os processos, todas as iterações e todas as instâncias pertencentes ao projeto, assim sendo, dependendo da combinação utilizada para gerar o relatório, em alguns casos não será exibido o nome destes, mas a palavra “All” em seu lugar) e por último está a exibição dos dados, como explicados no tópico anterior.

Ao final da análise principal, é possível clicar no botão ***See Merge Tasks***, isso fará com que seja exibida outra página contendo todas as tarefas do tipo “merge”. A página aberta consiste do nome da página no topo, logo abaixo são apresentados o projeto, o processo, a iteração e a instância em que o relatório é baseado e, por fim, uma tabela contendo o nome de todas as tarefas do tipo merge.

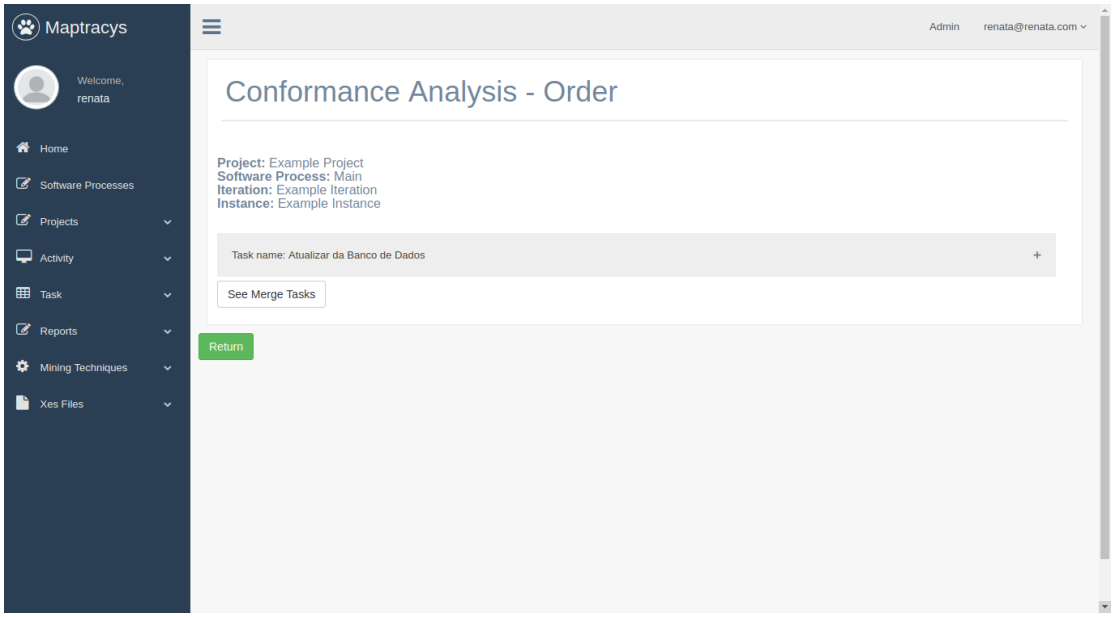


Figura 15: tela de exibição do relatório da análise de ordem

Visualizar Relatórios Gerados

Os relatórios dessa seção são os gerados durante a análise de cobertura, esses relatórios ficam armazenados para que seja possível posteriormente a realização de uma análise de evolução do projeto de acordo com as datas em que os relatórios foram gerados.

Para exibir os relatório gerados, primeiramente clique em ***Reports***, no menu lateral, após a exibição das opções desta seção, clique em ***Conformance Analysis***. Ao clicar nesta opção, será aberta uma nova página. Na página ***Conformance Analysis*** na terceira aba existem dois campos: ***Project*** e ***Software process***; escolha um projeto e um processo de software dentre os que estão disponíveis e, após isso, clique no botão ***Generated Reports***.

O resultado desse procedimento será a exibição de uma nova página uma tabela contendo todos relatórios referentes ao projeto e ao processo selecionados. Essa tabela é composta pelo número de identificação do relatório, pela listagem dos caminhos que são possíveis naquele processo, a data em que foi gerado e as ações possíveis de serem executadas no relatório.

Após realizar o procedimento de exibir relatórios e já na página ***Reports***, basta clicar na ação ***Show*** de um relatório para que este seja exibido.

A página de exibição tem em seu topo a numeração do relatório e a esquerda desta numeração estão dois botões de ação, um para geração do pdf do atual relatório e outro para edição do mesmo. Logo abaixo existem três tabelas. A primeira é a listagem completa dos caminhos e consiste da numeração do caminho e da listagem dos passos daquele caminho. A segunda é a listagem das atividades executadas separadas por caminho e consiste da numeração do caminho, do nome das atividades que foram executadas naquele caminho e se essas atividades estão em conformidade com o que era esperado. A terceira é a listagem das atividades não executadas separadas por caminho e consiste da numeração do caminho, do nome das atividades que não foram executadas naquele caminho e se essas atividades estão em conformidade com o que era esperado.

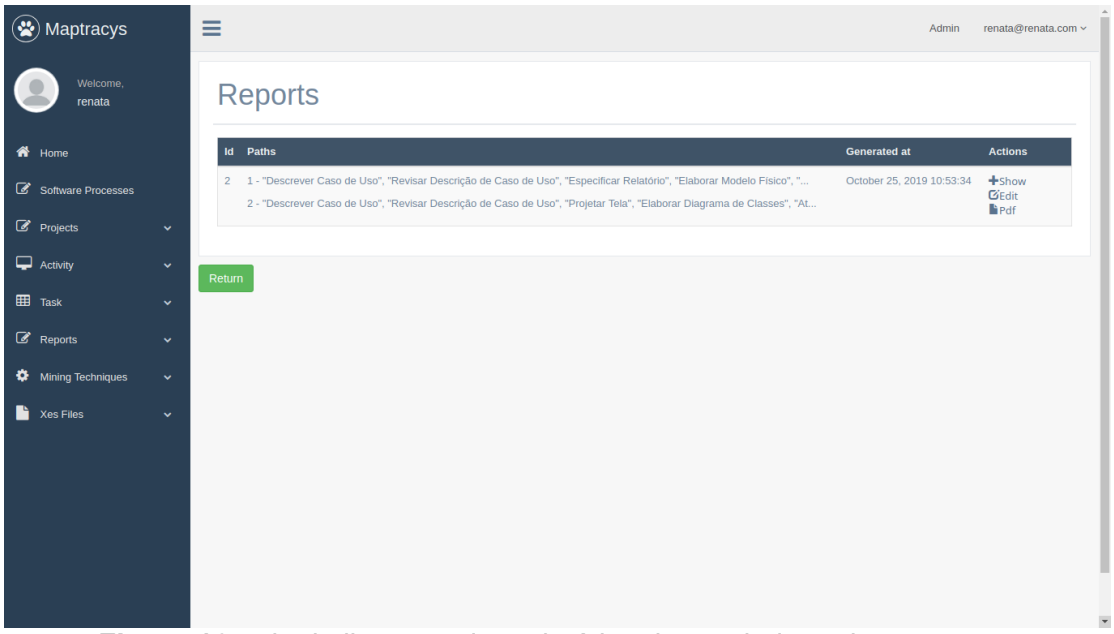


Figura 16: tela de listagem dos relatórios de um dado projeto e processo

Oportunidades de Melhoria

Para exibir possíveis oportunidades de melhoria clique em **Reports** no menu lateral e então clique na opção **Opportunities for Improvement** e será redirecionado para uma nova página. Na nova página entre com as informações para o projeto alvo, a iteração que será avaliada (podendo ser todas) e a instância que será usada (também podendo ser todas) e então clique em ***Show Opportunities***. Ao clicar neste botão, será redirecionado para uma página que listará todas as possíveis tarefas que podem ser adicionadas ao processo.

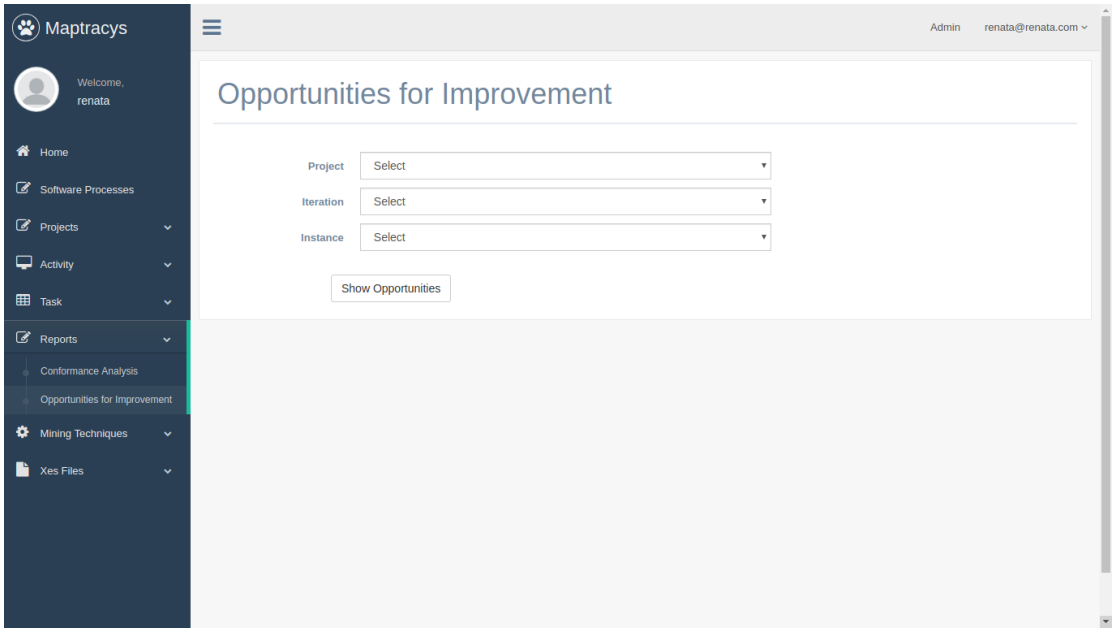


Figura 17: tela de seleção do projeto para exibição das oportunidades de melhoria

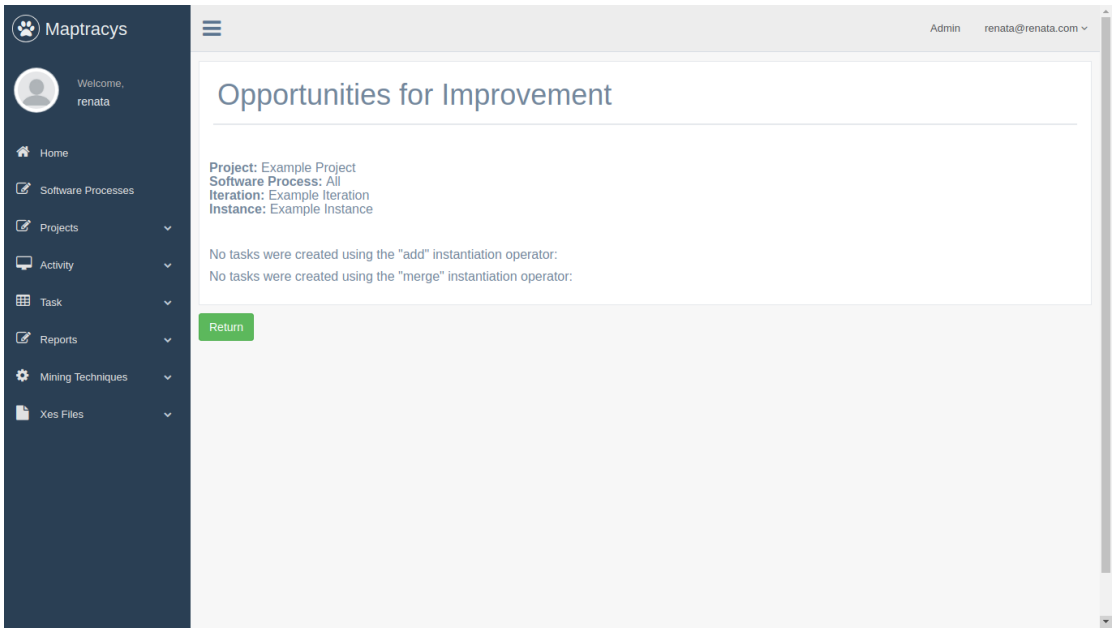


Figura 18: tela de exibição das oportunidades de melhoria de um dado projeto

APÊNDICE E – Listagem de Caminhos (listpath)

Este apêndice apresenta o pseudocódigo da geração da matriz de adjacência e listagem de caminhos.

A função l*ist_paths* é responsável por retornar a listagem de caminhos possíveis a partir de um determinado modelo de processo. O retorno deste método é uma matriz de inteiros, onde cada linha representa um caminho e cada coluna representa um evento, podendo estes eventos serem atividades ou eventos como início (“start”) e fim (“end”) de um caminho.

O Método list_paths possui a seguinte estrutura:

1. É tomada a matriz de adjacência que representa o processo com suas atividades, através da linha *matrix = activities_adjacency_matrix[1]*. O método *activities_adjacency_matrix* possui como retorno um array contendo a listagem de atividades presentes naquele processo e a matriz de adjacência que representa a interação das atividades dentro daquele processo. Obs.: *Cabe ressaltar que, o método activities_adjacency_matrix foi implementado considerando que o modelo de processo importado está representado na linguagem BPMN, sendo um arquivo com extensão .bpm.*

   O método *activities_adjacency_matrix* possui a seguinte estrutura:

   a. O método inicia atribuindo à variável *activities_array* um array contendo todas as atividades do processo, mais os eventos de início (“start”) e fim (“end”), sendo adicionado o evento de início na primeira posição do array e o de fim na última, se o mesmo os possuir. Não será abordado em mais detalhes o funcionamento do método *activities_with_events* por ser algo simplório e autoexplicativo pela breve descrição dada;
   b. Logo na sequência, é tomado o tamanho do *activities_array* e armazenado na variável *number_of_nodes*;
   c. Então, cria-se a matriz (*matrix*) de dimensões iguais a *number_of_nodes* e inicializada com zeros;
   d. Faz-se então um laço através de cada elemento do *activities_array*:
      i. Pega-se o próximo elemento a partir do método *outgoing* do elemento atual do array e atribui a variável *next_element*. O método *outgoing* possui a seguinte sequência de execução:

- Encontra-se as possíveis sequências do elemento, a partir do id do processo de software e do id original do elemento enquanto ainda estava no arquivo .bpm do modelo de processo importado:

- *SequenceFlow.where(software_process_id: self.software_process_id, sourceRef: self.original_id)*;
- Se foram encontradas possíveis sequências para o elemento, então:
    - Faz-se um mapeamento para retornar apenas um array com a referência dos próximos elementos de quando ainda estava no arquivo .bpm e o atribui à variável *sequences*: sequences = find_sequences.map{ |sequence| sequence.targetRef };
    - Então, faz-se uma busca através dos dados de atividades, procurando a partir do id do processo em que o elemento está compreendido e da lista de próximos elementos e atribui o retorno desta busca a variável *activity*: *activity = Activity.find_by(software_process_id: self.software_process_id, original_id: sequences);*
    - Se ao menos uma atividade foi encontrada na busca anterior, retorna-se a variável *activity*;
    - Se não foi encontrada uma atividade. Faz-se uma busca através dos dados de gateways, procurando a partir do id do processo em que o elemento está compreendido e da lista de próximos elementos e atribui o retorno desta busca a variável *gateway*;
    - Se ao menos um gateway foi encontrado na busca anterior, retorna-se a variável *gateway*;
    - Se não foi encontrado um gateway. Faz-se uma busca através dos dados de eventos, procurando a partir do id do processo em que o elemento está compreendido e da lista de próximos elementos e atribui o retorno desta busca a variável *event*;
    - Se ao menos um evento foi encontrado na busca anterior, retorna-se a variável *event*;
    - Se nenhuma das alternativas anteriores se concretizarem como verdadeiras, será retornado o valor nulo.

ii. Se o *next_element* for do tipo *Activity* ou *Event*, então:

- *j* recebe o índice do *next_element* no *activities_array*.
    - Se *j* não for nulo: *matrix[array_index][j] = 1*.

ii. Senão, se *next_element* for do tipo *Gateway*, então:

- É invocado o método *gateway_outgoing*, passando: *matrix, activities_array, next_element e array_index*, nesta sequência;
    - O método *gateway_outgoing* possui uma estrutura parecida com a apresentada a partir do tópico d.i. Para não se tornar redundante, serão apresentadas apenas as diferenças presentes:
        - Ao nvés de *next_element*, em *gateway_outgoing* é usado uma variável chamada *next_elements*. Isto se dá pois, sendo um gateway, existe mais do que uma possível sequência para se seguir;
        - Na sequência, vem a última diferença. É feito um laço através de cada um dos próximos eventos: *next_elements.each*. Dentro

deste laço se repetem os passos apresentados nos pontos d.ii e d.iii.

ii. Por fim, retorna-se um array contendo o array de atividades (*activities_array*) e a matriz de adjacência (*matrix*): *return [activities_array, matrix]*.

2. Atribui-se a o tamanho de *matrix* à *number_of_nodes*;
3. Atribui-se o índice inicial do array à *current_path*;
4. Cria-se um array de array e o atribui a variável *paths*. Ambos arrays possuem tamanho de um elemento, sendo o array interno preenchido com zero: *paths = Array.new(1){ Array.new(1) {0} }*;
5. Enquanto *paths* no índice *current_path* retornar um valor verdadeiro (*while paths[current_path]*. Obs.: Valores verdadeiros em Ruby são todos aqueles diferentes de *false* e *nil*), então:
    a. Invoca-se o método *map_path*, passando, *nesta ordem, matrix, number_of_nodes, paths, current_path*;
        i. Atribui-se a variável *node* o último elemento do caminho atual: *node = paths[current_path].last*;
        ii. Enquanto existir um próximo nó, é executado uma sequência de procedimentos. A identificação de um próximo nó é feita através do método *get_next_node*, que recebe como parêmetros: *matrix*, *number_of_nodes* e *node*. O método *get_next_node* possui a seguinte composição:

- Cria-se um array e o atribui a variável *next_node*;
- Então realiza-se um laço indo de zero até *number_of_nodes*, não incluído, e dá-se o nome do contador de *j*;
    - Se *matrix[node][j] == 1*, adiciona-se *j* ao final do array *next_node*.
- Se o tamanho do array *next_node* for maior ou igual a um, retorna-se o array *next_node*. Senão, retorna-se nulo.

i. Dentro do laço citado no passo a.ii:

- Se o tamanho de *next_node* for igual a um:
    - É adicionado ao final do caminho atual o primeiro elemento do array *next_node*: *paths[current_path] << next_node.first*;
    - Se for encontrado um laço no caminho, encerra-se o laço “while”: *break if paths[current_path].take(paths[current_path].length - 1).include? next_node.first* .
- Senão, se o tamanho de *next_node* for maior que um:
    - Faz-se um laço através de todos os elementos de *next_node*. Dentro deste laço serão criadas cópias do caminho atual, porém serão adicionados um de cada próximo nó em cada uma dessas cópias. O laço tem seu início em um e vai até o tamanho de *next_node*, não incluído:
        - A variável *path* recebe uma cópia do caminho atual (*paths[current_path]*) somado ao elemento atual do laço sobre *next_node* (*next_node[i]*). São removidos os valores repetidos

em sequência presentes em *path*. Isto é feito através do método *strip_duplicates*, que recebe como parâmetro o *path*. O método *strip_duplicates* possui a seguinte estrutura:

- Cria-se um array auxiliar (*aux_list*), vazio, para armazenar os elementos não duplicados;
- Inicializa-se o valor do índice (*index*) com o valor referente ao início de um array;
- Enquanto *index* for diferente do tamanho do array recebido como parâmetro (*list*):
  - Adiciona-se ao final da lista auxiliar o valor de *list* no índice atual;
  - Enquanto *list[index]* não for diferente de *list[index+1]*, *index* é incrementado em um;
  - Incrementa-se *index* em um.
- Retorna-se o array *aux_list*.

- Se não existir outro caminho igual ao representado em *path* no array *paths*, inclui-se *path* ao final de *paths*.

- Adiciona-se ao caminho atual (*paths[current_path]*) o primeiro elemento presente em *next_node*.

- Atribui-se a variável *node* o último elemento presente no caminho atual (*paths[current_path*).

a. Então, incrementa-se em um o valor de *current_path*: *current_path += 1;*

3. Por fim, retorna-se o array de caminhos (*paths*) contendo um exemplar de cada caminho encontrado: *return paths.uniq*.

APÊNDICE F – Pseudocódigo para Identificação de não Conformidade

Este apêndice apresenta o Pseudocódigo para Identificação de não Conformidade.

Tabela F.1 Pseudocódigo da Verificação de Conformidade – Perspectiva de Atividade

1. Inicializa-se os atributos de projeto (project), processo de software (software_process) a partir dos valores que foram passados pelo usuário;
2. O projeto (project) e o processo de software (software_process) são buscados no banco de dados;
3. Todas as atividades executadas (all_executed_activities) são identificadas usando o método *activities_list*, passando o processo de software, da classe *Project*:
    a. É instanciado um array para armazenar as atividades denominado *activity_list*;
    b. Executa-se um laço através de todas as instâncias (instances) do atual projeto;
    c. Dentro do laço de instâncias ocorre um laço para cada tarefa (task) de cada instância;
    d. É testado o estado de atribuição de cada tarefa através do método *assignment_status* da classe *Task*:
        i. A função *assignment_status* retorna verdadeiro (true) se a tarefa possuir uma data de início ou um ator atribuído a ela, em caso contrário retorna nulo (null) que é avaliado como falso (false);
    e. Se o retorno do método *assignment_status* for verdadeiro: busca-se por todas as atividades relacionadas aquela tarefa, filtrando pelo identificador do processo. Essas atividades retornadas são adicionadas ao array *activity_list*;
    f. Por fim é retornado o arrayl *activity_list* retirando-se os valores duplicados;
4. Todas as atividades esperadas (expected_activities) são identificadas usando o método *all_mandatory_processes_activities*, passando o processo de software, da classe *Project*:
    a. É instanciado um array para armazenar as atividades denominado *activities*;
    b. Executa-se um laço através de todos os processos (software_processes) do atual projeto;
    c. Dentro do laço de processos executa-se outro laço. Esse laço interno busca por todas as atividades de cada processo que estão relacionadas ao processo passado para o método, filtrando pelo identificador do processo passado. As atividades retornadas são adicionadas ao array *activities*;
    d. Por fim é retornado array *activities*;
5. É feita uma filtragem no resultado anterior para garantir que apenas instâncias da classe Activity estão presentes;
6. Na sequência é feita uma interseção entre todas as atividades executadas

(all_executed_activities) e todas as atividades esperadas (expected_activities) para se identificar quais são as atividades comuns (common_activities) em ambas: common_activities = all_executed_activities ∩ expected_activities;

7. O resultado de todas as atividades não executadas (all_unexecuted_activities) é tomado. Seu valor é igual a diferença entre as atividades esperadas (expected_activities) e a interseção de todas as atividades executadas (all_executed_activities) e todas as atividades esperadas (expected_activities): all_unexecuted_activities = expected_activities - (all_executed_activities ∩ expected_activities);
8. E então vem a cobertura total (all_coverage) que é o resultado da conta: (length(common_activities) / length(expected_activities)) * 100. O valor final está representado em porcentagem;
9. Cria-se três novos arrays: *common_activities, coverage, unexecuted_activities*;
10. Recupera-se os possíveis caminhos através do método list_paths da classe SoftwareProcess e atribuem-nos ao atributo paths;
11. Toma-se os nomes das atividades usando o método activities_adjacency_matrix da classe SoftwareProcess, pega-se o primeiro índice e o atribui ao atributo paths_activities;
12. Instancia-se um novo array (executed_activities);
13. Tem-se então um laço através de todos os caminhos;
14. Remove-se o primeiro evento (start) de cada caminho;
15. Executa-se outro laço, este é através de cada passo possível em cada um dos caminhos;
16. Dentro do laço mais interno, substitui-se o identificador da atividade por seu nome: paths[i][j] = paths_activities[node];
17. Para finalizar o laço mais interno, faz-se um filtro para retirada de todo e qualquer evento que não seja uma atividade;
18. E então são calculados os valores de atividades não executadas (unexecuted_activities), atividades executadas (executed_activities) e a cobertura (coverage) para cada caminho:
    a. common_activities[i] = all_executed_activities ∩ paths[i];
    b. unexecuted_activities[i] = paths[i] - common_activities[i];
    c. executed_activities[i] = all_executed_activities ∩ paths[i];
    d. coverage[i] = (length(executed_activities[i]) / length(paths[i])) * 100.

APÊNDICE G – Orientações para Verificação de Conformidade

Este apêndice apresenta as orientações para verificação de conformidade.

A partir de um estudo de observação realizado em uma reunião de planejamento de projeto e das boas práticas recomendadas pelas normas e modelos de maturidade, um conjunto de atividades e tarefas para verificação de conformidade do plano de projeto e da execução foi definido. Além do estudo de observação realizado, particularmente, os seguintes modelos de maturidade serviram como base para a descrição das atividades: CMMI (CMMI Product Team, 2010) e MPS.BR (SOFTEX, 2016b).

Neste apêndice estão descritas as orientações para verificação de conformidade a ser realizada após a Instanciação (criação de tarefas) e para verificação de conformidade a ser realizada após a Execução (registros de execução). As descrições envolvem somente uma descrição sucinta no nível de atividades conforme apresentado nas Tabelas H.1 e H.2. Nestas tabelas, as etapas estão realçadas em cinza e suas respectivas atividades são listadas a seguir em negrito. A descrição de cada atividade é apresentada em itálico.

Verificação da Conformidade do Plano de Projeto

Ao concluir a etapa de "*Instanciação*", antes de iniciar a etapa de "*Execução*", pode ser realizada a verificação de conformidade do plano de projeto com o modelo de processo, a fim de identificar não conformidades. O gerente de projeto é comunicado das não conformidades identificadas, sendo de sua responsabilidade tomar a decisão sobre as formas de resolução. Estas formas podem envolver a alteração do plano de projeto ou seguir para a etapa seguinte. A etapa "*Execução*" é iniciada após a equipe de garantia de qualidade verificar a forma como as não conformidades identificadas foram resolvidas. Cabe ressaltar que não é obrigatória uma equipe de garantia de qualidade, mas sim de algum profissional com este papel. A Tabela H.1 apresenta as atividades da etapa de *Verificação da Conformidade do Plano de Projeto*.

Tabela G.1 – Atividades da etapa Verificação da Conformidade do Plano de Projeto

Verificação da Conformidade do Plano de Projeto
Obter o Modelo de Processo de Software *Envolve a obtenção do modelo de processo de software de referência para o projeto.* **Obter o Plano de Projeto** *Envolve a obtenção do plano de projeto.* **Identificar as não Conformidades** *Envolve a verificação do plano criado, antes do início da execução, para identificação de não conformidades. Na identificação de não Conformidades, é verificada a aderência do plano às descrições de processo, padrões e procedimentos.* **Gerar o Relatório de não Conformidades** *Envolve a geração do Relatório de não Conformidades, o qual apresenta as atividades do processo definido obrigatórias não selecionadas na criação do plano, e também as inconsistências na ordem de execução das tarefas no plano.* **Comunicar ao Gerente de Projeto as não Conformidades identificadas** *Envolve a comunicação, ao Gerente de Projeto, das não Conformidades identificadas.* **Definir Forma de Resolver as não Conformidades** *Envolve a definição, por parte do Gerente de Projeto, da forma de resolução das não Conformidades relatadas durante a avaliação da Garantia de Qualidade, como por exemplo: Fazer o plano satisfazer o processo descrito, padrão, procedimento; ou Tomar uma decisão executiva de não satisfazer o processo descrito, padrão, procedimento, caso isso seja necessário, arcando com as consequências deste ato; ou Solicitar alteração o processo descrito, padrão ou procedimento para torná-lo utilizável (eficaz).* **Verificar a correção das não Conformidades** *Envolve a verificação das correções, das não Conformidades, realizadas pelo Gerente de Projeto.*

Verificação da Conformidade da Execução

Ao concluir a etapa de "*Execução*" pode ser realizada a *Verificação de Conformidade da Execução* com o modelo de processo, a fim de identificar não conformidades. Esta verificação pode ocorrer ao fim de uma iteração ou ao fim do projeto. Esta verificação se faz necessária porque durante a execução, os elementos de processo definidos podem ser alterados para, por exemplo, atender a situações

não previstas, relacionadas a fatores, como prazo e gerenciamento de recursos (LAURENT *et al.*, 2014). Nesta etapa são identificadas as não conformidades de remoção de atividade e alteração na ordem do processo, bem como a inclusão de atividades não previstas que podem ser consideradas oportunidades de melhoria. A Tabela H.2 apresenta as atividades da etapa de verificação da conformidade da execução.

Tabela G.2 – Atividades da etapa de Verificação da Conformidade da Execução

Verificação da Conformidade da Execução
Obter o Modelo de Processo de Software *Envolve a obtenção do modelo de processo de software de referência para o projeto.* **Obter os Registros de Execução do Processo** *Envolve a obtenção dos registros de execução, assim que a execução da iteração ou do projeto é concluída.* **Identificar as não Conformidades** *Envolve a verificação da execução, após a execução da iteração ou do projeto, para identificação de não conformidades. Na identificação de não Conformidades, é verificada a aderência dos processos executados às descrições de processo, padrões e procedimentos.* **Gerar o Relatório de Verificação da Execução** *Envolve a geração do Relatório de não Conformidades, o qual apresenta as atividades do processo definido obrigatórias não selecionadas durante a execução do processo, e também as inconsistências na ordem de execução das tarefas realizadas durante a execução.* **Comunicar ao Gerente de Projeto as não Conformidades identificadas** *Envolve a comunicação das não Conformidades identificadas ao Gerente de Projeto.* **Identificar Oportunidades de Melhoria** *Envolve a identificação de tarefas realizadas ao longo da execução do processo que não são prevista e não estão descritas no processo de referência. Estas alterações, quando realizadas em outras execuções do processo padrão, podem ser utilizadas para realizar a alteração do processo descrito, padrão ou procedimento para torná-lo utilizável (eficaz)* **Elaborar Relatório Final** *Envolve a elaboração do relatório final da Garantia da Qualidade que registra as ações realizadas ao longo da instanciação e execução do processo.*