Instituto Alberto Luiz Coimbra de
Pós-Graduação e Pesquisa de Engenharia
COPPE UFRJ

EXECUÇÃO, MONITORAMENTO E REALIMENTAÇÃO DE PROCESSOS DE SOFTWARE UTILIZANDO DADOS DE PROVENIÊNCIA E ONTOLOGIA

Gabriella Castro Barbosa Costa

Exame de Qualificação apresentado 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.

Orientadora: Cláudia Maria Lima Werner

Rio de Janeiro
Setembro de 2015

EXECUÇÃO, MONITORAMENTO E REALIMENTAÇÃO DE PROCESSOS DE SOFTWARE UTILIZANDO DADOS DE PROVENIÊNCIA E ONTOLOGIA

Gabriella Castro Barbosa Costa

Setembro/2015

Orientadora: Cláudia Maria Lima Werner

Programa: Engenharia de Sistemas e Computação

Atualmente, as organizações estão investindo na definição e melhoria de seus processos com o objetivo de melhorar a qualidade de seus produtos e, no contexto de processos de software, isto não é diferente. Assim, uma prática a ser considerada para apoiar a melhoria contínua de processos de software consiste na reutilização do conhecimento adquirido em execuções anteriores deste processo. Tendo sido definidas medidas para avaliar o desempenho de processos de software (tais como tempo, produtividade, esforço, por exemplo), uma análise dos dados de execução do processo, necessária para a obtenção de medidas previamente definidas, pode ser realizada objetivando detectar pontos de melhoria do processo. Uma das formas de captura destes dados de execução do processo, é através de modelos de proveniência. Posteriormente a esta captura, os dados de execução do processo podem ser analisados utilizando mecanismos de derivação de informações, tais como as máquinas de inferência para ontologias, visando obter informações capazes de contribuir com a melhoria do processo. Assim, este trabalho tem como objetivo definir, implementar e avaliar uma sistemática para apoiar a execução, monitoramento e realimentação de processos de software, visando melhorar o desempenho através da utilização de dados de proveniência e ontologias. Uma prova de conceito foi realizada utilizando dois processos de software de empresas nacionais, para avaliar parte da abordagem proposta.

Sumário

1. INTRODUÇÃO 1
1.1 Motivação 1
1.2 Problema e Questão de Pesquisa 2
1.3 Objetivos 3
1.4 Organização do Texto 3
2. PROCESSOS DE SOFTWARE 5
2.1 Introdução 5
2.2 Ciclo de Vida de Processos 6
2.3 Considerações sobre o Capítulo 17
3. PROVENIÊNCIA DE DADOS 18
3.1 Introdução 18
3.2 Modelos de Proveniência de Dados 19
3.2.1 OPM 19
3.2.1 PROV 21
3.3 PROV-Process 22
3.4 Considerações sobre o Capítulo 27
4. REVISÃO DA LITERATURA 29
4.1 Revisão Bibliográfica Inicial 29
4.2 Busca Estruturada 45
4.2.1 Planejamento da Busca Estruturada 45
4.2.2 Execução da Busca Estruturada 49
4.2.3 Agregação dos Resultados Obtidos 51
4.3 Considerações sobre o Capítulo 52
5. PROPOSTA 53
5.1 Fase 1: Definição das Medidas do Processo 54

5.2 Fase 2: Execução e Monitoramento do Processo 57
5.3 Fase 3: Análise da Execução do Processo 66
5.4 Fase 4: Realimentação do Processo 74
5.5 Considerações sobre o Capítulo 75
6. PROVA DE CONCEITO 76
6.1 Planejamento 77
6.2 Execução 77
6.2.1 Processo 1 77
6.2.2 Processo 2 94
6.3 Análise das Hipóteses 109
6.4 Limitações Encontradas 110
6.5 Ameaças à Validade 111
6.6 Considerações sobre o Capítulo 112
7. CONSIDERAÇÕES FINAIS 113
7.1 Contribuições 113
7.2 Atividades Realizadas 113
7.3 Atividades Previstas 114
7.4 Cronograma Proposto 115
REFERÊNCIAS BIBLIOGRÁFICAS 117
APÊNDICE A – Trabalhos Analisados na Revisão Bibliográfica Inicial 126
APÊNDICE B – Informações dos Artigos Incluídos na Busca Estruturada 130

Lista de Figuras

Figura 2.1: Ciclo de Vida de Processos de Negócio (PUNTAR et al., 2009).....6

Figura 2.2: Ciclo de Vida do Reúso de Processos de Software (JØRGENSEN (2000) apud BARRETO (2011)).....9

Figura 2.3: Ciclo de Vida de Processos da Abordagem.....10

Figura 2.4: Modelo de Processo utilizando a notação BPMN (COSTA et al., 2014).....14

Figura 3.1: Nós e dependências causais do OPM.....20

Figura 3.2: Vértices do Modelo PROV.....21

Figura 3.3: Principais Arestas do Modelo PROV (BELHAJJAME et al. 2012).....21

Figura 3.4: Modelo do Banco de Dados da Abordagem PROV-Process.....24

Figura 3.5: Principais classes e propriedades da PROV-O (BELHAJJAME et al., 2013).....27

Figura 4.1: Total de Artigos retornados pelas máquinas de busca.....50

Figura 5.1: Fases da Proposta.....53

Figura 5.2: Diagrama de Tabelas e Relacionamentos do PROV-Process.....58

Figura 5.3: Processo para Requisição de Mudanças no Software.....61

Figura 5.4: Indivíduos da classe ‘Activity’.....63

Figura 5.5: Indivíduos da classe ‘Person’, que é uma subclasse de ‘Agent’.....63

Figura 5.6: Indivíduos da classe ‘Entity’.....64

Figura 5.7: Especificação da Execução da Atividade 1 – Abrir Requisição de Mudança.....64

Figura 5.8: Especificação da Execução da Atividade 2 – Implementação da Solução.....65

Figura 5.9: Especificação dos Agentes.....65

Figura 5.10: Especificação da Entidade DLL - Cálculos.....66

Figura 5.11: Especificação da Entidade DLL – ERP – Exportação de Informações.....66

Figura 5.12: Especificação da Entidade DLL – NF-e v2.0 66

Figura 5.13: Tela do protótipo da EvolTrack-Process (SANTANA, 2005) 67

Figura 5.14: Inferência sobre a Activity ‘Implementação da Solução 2’ 68

Figura 5.15: Associação entre as relações/propriedades do PROV-O 69

Figura 5.16: Inferência sobre a Activity ‘Implementação da Solução 3’ 69

Figura 5.17: Inferência sobre a Activity ‘Implementação da Solução 4’ 69

Figura 5.18: Inferência sobre a Activity ‘Implementação da Solução 5’ 70

Figura 5.19: Inferência sobre a Activity ‘Implementação da Solução 6’ 70

Figura 5.20: Inferência sobre a Activity ‘Implementação da Solução 7’ 70

Figura 5.21: Inferência sobre o Agent ‘anthony.nichols’ 71

Figura 5.22: Inferência sobre o Agent ‘april.sanchez’ 71

Figura 5.23: Inferência sobre o Agent ‘favio.riviera’ 72

Figura 5.24: Inferência sobre o Agent ‘helen.kelly’ 72

Figura 5.25: Inferência sobre o Agent ‘marc.marseau’ 72

Figura 5.26: Inferência sobre a Entity ‘DLL – Cálculos’ 73

Figura 5.27: Inferência sobre a Entity ‘DLL – ERP Exportação de Informações’ 73

Figura 5.28: Inferência sobre a Entity ‘DLL – NF-e v2.0’ 74

Figura 6.1: Processo Principal para requisição e gerenciamento de mudanças no software 78

Figura6.2: Subprocesso 'Descrição Owner' 79

Figura6.3: Subprocesso 'DescriçãoTécnica' 79

Figura6.4: Subprocesso 'Descrição Testes' 79

Figura 6.5: Esquema para Importação da Activity 'Abertura da RDM' 85

Figura 6.6: Esquema para Importação da Activity 'Implementação da Solução' 86

Figura 6.7: Esquema para Importação da Activity 'Alterar RDM para Concluída' 86

Figura 6.8: Gráfico da Medida de Tempo de Execução do Processo 87

Figura 6.9: Gráfico da Medida de Tempo Médio de Execução do Processo 88

Figura 6.10: Gráfico da Medida de Média de Desdobramentos 89

Figura 6.11: Tarefas que influenciaram outras tarefas 90

Figura 6.12: Agentes que influenciaram uma tarefa 91

Figura 6.13: Tarefas e Artefatos manipulados pelo Agente DotNet 91

Figura 6.14: Tarefas em que um determinado artefato foi manipulado 92

Figura6.15: Artefato 'arApuracaoPisCofins03' 93

Figura 6.16: Artefato 'fRelApuracaoPisCofins' 93

Figura 6.17: Tarefa implementação da Solução (id=26) 94

Figura 6.18: Modelo de Fluxo Processo 2 95

Figura 6.19: Gráfico da Porcentagem de erros por módulo do sistema 103

Figura 6.20: Gráfico da Porcentagem de solicitações de novos recursos por módulo do sistema 104

Figura 6.21: Gráfico da Medida de Tempo de Execução do Processo 105

Figura 6.22: Gráfico da Medida de Tempo Médio de Execução do Processo 106

Figura 6.23: Agentes que influenciaram uma tarefa 107

Figura 6.24: Tarefas e Artefatos manipulados pelo Agente Humberto 107

Figura 6.25: Tarefas em que um determinado artefato foi manipulado 108

Figura 6.26: Inferência sobre a tarefa Resolução do Caso (id=11) 109

Figura 6.27: Artefato ‘Versão 2.4.5 Maldivas’ 109

Lista de Tabelas

Tabela 2.1: Principais Elementos BPMN (adaptado de OMG, 2011) 11

Tabela 4.1: Listagem dos trabalhos selecionados de TEIXEIRA (2014) 30

Tabela 4.2: Análise das abordagens com relação aos requisitos propostos 44

Tabela 4.3: Quantidades de trabalhos obtidos de acordo com os requisitos de seleção 44

Tabela 4.4: Totais de Artigos da Busca Estruturada e após leitura do título e resumo 50

Tabela 4.5: Totais de Artigos após leitura completa 51

Tabela 5.1: Medidas Essenciais e suas Relações (adaptado de [PUTNAM e MYERS, 2003]) 54

Tabela 5.2: Informações para a Definição de Novas Medidas (adaptado de ROCHA et al., 2012) .. 55

Tabela 5.3: Exemplo de Medida criada (ROCHA et al., 2012) 56

Tabela 5.4: Relação entre os vértices do PROV e os elementos do BPMN 60

Tabela 5.5: Dados de Execução da Tarefa 62

Tabela 5.6: Artefatos Manipulados por Atores durante a execução 62

Tabela 6.1: Definição da Medida de Tempo de Execução do Processo (TEP) 80

Tabela 6.2: Definição da Medida de Tempo Médio de Execução do Processo (TMEP) 81

Tabela 6.3: Definição da Medida de Porcentagem de Desdobramentos (PD) 81

Tabela 6.4: Exemplo de Dados e execução – Processo 1 – Parte 1 83

Tabela 6.5: Exemplo de Dados e execução – Processo 1 – Parte 2 83

Tabela 6.6: Definição da Medida de Porcentagem de Erros por Módulo do Sistema (PEM) 96

Tabela 6.7: Definição da Medida de Porcentagem de Novos Recursos por Módulo do Sistema(PNRM) 97

Tabela 6.8: Definição da Medida de Tempo de Execução do Processo (TEP) 98

Tabela 6.9: Definição da Medida de Tempo Médio de Execução do Processo (TMEP) 98

Tabela 6.10: Exemplo de Dados e execução – Processo 2 – Parte 1 100

Tabela 6.11: Exemplo de Dados e execução – Processo 2 – Parte 2 100

Tabela 7.1: Cronograma de atividades para esta proposta de tese 116

1. INTRODUÇÃO

Este capítulo tem como objetivo apresentar a motivação para o desenvolvimento do presente trabalho, bem como o problema e a questão de pesquisa a ser tratada no mesmo. Além disso, são apresentados os objetivos a serem alcançados e uma breve descrição da organização do texto.

1.1 Motivação

As organizações têm investido cada vez mais na melhoria da definição, uso e gerenciamento de seus processos, tendo como princípio que a qualidade dos produtos de software está fortemente relacionada à qualidade dos processos de desenvolvimento adotados para construí-los (FUGGETTA, 2000).

Partindo-se da premissa de que processos similares são realizados por organizações de desenvolvimento de software, nota-se a possibilidade de utilização da experiência de projetos anteriores, de forma a estabelecer melhores políticas a serem adotadas em projetos futuros (BARRETO, 2011). Assim, a reutilização do conhecimento adquirido em execuções anteriores do processo apresenta-se como uma prática a apoiar a melhoria contínua do processo.

De forma a possibilitar a melhoria de processos de software, é necessária a definição de medidas associadas a estes processos, viabilizando análises e verificações sobre a necessidade de alterações no processo em questão. A partir destas verificações, os processos podem ser avaliados, alterados / remodelados ou novos processos podem ser concebidos (FERREIRA *et al*., 2008).

Tendo sido definidas medidas para avaliar o desempenho de processos de software (tais como tempo, produtividade, esforço, por exemplo), uma análise dos dados de execução do processo, necessária para a obtenção das medidas previamente definidas, deve ser realizada objetivando detectar pontos de melhoria do processo. Para a captura destes dados de execução do processo, modelos de proveniência, tais como OPM (MOREAU *et al*., 2011) e PROV (GROTH e MOREAU, 2013), podem ser utilizados como uma alternativa para prover um modelo padrão para captura destes dados. Proveniência de dados é definido por BUNEMAN (2001) como a descrição da origem de uma parte de um dado e o processo pelo qual este passou até o armazenamento junto ao banco de dados. A captura da proveniência pode se dar tanto em nível de dado quanto em nível de processo. Ressalta-se, porém, que, para a obtenção dos benefícios das informações de proveniência, é necessário a captura destas, bem como a modelagem e armazenamento de modo integrado para posterior consulta (MARINHO, 2011). Assim, de forma a garantir o armazenamento e posterior captura dos dados de execução dos processos, uma solução seria a utilização de um modelo de proveniência.

Após a captura e armazenamento dos dados de proveniência do processo, uma análise sobre estes pode ser realizada de forma a permitir a melhoria do processo (por exemplo, um menor tempo de execução ou uma maior eficiência dos resultados). Uma possível maneira de analisar os dados de proveniência de processos é por meio do uso de ontologia e dos mecanismos de inferência oferecidos por ela, possibilitando a descoberta de informações estratégicas para a realimentação do processo de software. Um maior detalhamento sobre proveniência de dados, modelos de proveniência OPM e PROV e ontologias pode ser encontrado no Capítulo 3.

Tem-se como proposta deste trabalho o apoio a reutilização da experiência adquirida em execuções anteriores de processos de software, visando aprimorar seu desempenho em futuras execuções, por meio de dados de proveniência e ontologia.

1.2 Problema e Questão de Pesquisa

Em um mundo onde o prazo para entrega de novos produtos no mercado é cada vez mais curto, as empresas são obrigadas a inovar constantemente. O tempo necessário para a execução de todo o ciclo de desenvolvimento de um projeto tornou-se um parâmetro crítico para o desenvolvimento de software. Se uma determinada empresa de desenvolvimento de software não é capaz de entregar o sistema requisitado dentro do prazo necessário, certamente haverá outra empresa que o fará dentro dos requisitos pré-estabelecidos (FERREIRA *et al*., 2008). Além de prazos cada vez mais curtos, há uma necessidade crescente por soluções eficazes que sejam capazes de lidar com a diversidade e contexto dos projetos em que os processos precisam ser aplicados, além da demanda por melhoria contínua dos mesmos, visando a produção de software de qualidade.

Pesquisadores e profissionais da indústria têm, cada vez mais, explorado técnicas para melhorar os processos de software. Deseja-se avaliar se o processo projetado foi, de fato, implementado, e o que ocorreu durante a sua fase de execução. Tem-se como objetivo verificar o quão apropriado foi o processo modelado e quais alterações precisam ser realizados para prover a melhoria do processo de software (HUO *et al*., 2008).

Visando alcançar a melhoria contínua do processo, uma definição formal do mesmo e medidas associadas ao desempenho do processo devem ser criadas. Partindo deste princípio, os processos podem ser avaliados e diagnosticados, a fim de formular hipóteses sobre onde devem ser realizadas as alterações. Posteriormente, novos processos podem ser concebidos e pode-se verificar se as melhorias desejadas foram efetivamente atingidas (FERREIRA *et al*., 2008).

Assim, o problema de pesquisa do presente trabalho é definido a seguir:

> *Como melhorar o desempenho de processos de software*
> *a partir da experiência adquirida em execuções anteriores do mesmo?*

1.3 Objetivos

O objetivo geral deste trabalho consiste em *definir, implementar e avaliar uma sistemática para apoiar a execução, monitoramento, análise e realimentação de processos de software, visando melhorar o desempenho do mesmo, pela utilização de dados de proveniência e ontologias.*

Para atingir o objetivo proposto, foram estabelecidas, inicialmente, quatro etapas principais:

1. Definição de modelos / padrões a serem utilizados para a representação do processo a ser executado, monitorado e realimentado;
2. Definição do procedimento e de mecanismos / apoio ferramental que permitam a execução, monitoramento e extração dos dados de execução de instâncias do processo definido;
3. Definição do procedimento e de mecanismos / apoio ferramental que permitam a análise e avaliação dos dados coletados / métricas relativas à execução do processo; e
4. Definição do procedimento e de mecanismos / apoio ferramental que possibilitem a realimentação do processo a partir das melhorias identificadas com relação ao desempenho do processo previamente executado.

1.4 Organização do Texto

Este trabalho foi dividido em sete capítulos. O presente capítulo, apresenta a motivação, o problema, a questão de pesquisa e os objetivos a serem alcançados, além da organização do texto.

O Capítulo 2 expõe os principais conceitos relacionados a processos de software, bem como seu ciclo de vida, notação para modelagem e uma seção sobre melhoria e medição de processos de software.

No Capítulo 3, tem-se uma introdução sobre proveniência de dados e seus principais modelos (OPM e PROV), além da apresentação da abordagem PROV-Process, que será utilizada nesta proposta para captura e análise dos dados de proveniência de processos de software.

O Capítulo 4 traz uma revisão bibliográfica, realizada a partir da revisão proposta por TEIXEIRA (2014), e uma busca estruturada com foco em soluções / ferramentas utilizadas para execução, monitoramento e avaliação de processos de software.

O Capítulo 5 trata desta proposta de trabalho, apresentando, em linhas gerais, as etapas envolvidas na abordagem de execução, monitoramento e realimentação de processos de software.

No Capítulo 6, é apresentada uma prova de conceito relacionada às três primeiras etapas da abordagem, onde são utilizados dois processos de software da indústria.

Por fim, no Capítulo 7, são detalhadas as principais contribuições deste trabalho, as atividades previstas e as realizadas até o momento, os resultados preliminares já obtidos e os previstos e uma proposta de cronograma a ser seguido.

2. PROCESSOS DE SOFTWARE

Neste capítulo, são apresentados os conceitos relacionados a processos de software. Inicialmente, é feita uma introdução sobre processos de software, seguida por uma seção sobre ciclo de vida de processos propostos na literatura, detalhando cada uma de suas fases. Por fim, na Seção 2.5, tem-se as considerações finais do capítulo.

2.1 Introdução

Falbo (1998) afirma que um processo de software consiste em um conjunto de atividades, artefatos, procedimentos e recursos:

- **Atividades**: Compreendem as tarefas a serem realizadas. Uma atividade requer recursos e pode consumir ou gerar artefatos. Para sua realização, uma atividade pode adotar um procedimento. Uma atividade também pode ser decomposta em subatividades. Além disso, atividades podem depender da finalização de outras atividades.
- **Artefatos**: Compreendem os produtos de software gerados ou consumidos por atividades durante a sua execução.
- **Procedimentos**: Compreendem as condutas estabelecidas e ordenadas para a realização das atividades.
- **Recursos**: Compreendem qualquer fator necessário à execução de uma atividade, mas que não seja um insumo para a mesma. Os recursos podem ser classificados em: recursos de hardware, recursos de software e recursos humanos.

Existem diversas definições para processos de software. Fuggetta (2000) apresenta a seguinte definição: “um conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos necessários para conceber, desenvolver, implantar e manter um produto de software”. Já para Acuna *et al*. (2000), um processo de software pode ser definido como o “conjunto de atividades necessárias para produzir um sistema de software, executado por um grupo de pessoas organizadas de acordo com uma determinada estrutura organizacional e contando com o apoio de ferramentas”. Sommerville (2004) define processo de software como “um conjunto de atividades e resultados associados que produzem um produto de software”. Uma definição objetiva e completa de processo de software que será adotada neste trabalho é a apresentada por Paulk (2009): “um conjunto de atividades, métodos, práticas e transformações que as pessoas usam para desenvolver e manter software e seus produtos associados”.

2.2 Ciclo de Vida de Processos

O ciclo de vida de um processo, também chamado de meta-processo, deve abranger as principais atividades (meta-atividades) envolvidas neste. Processos de negócios[1] possuem um ciclo de vida (Figura 2.1) que inclui seis atividades principais, sendo elas: projeto, modelagem, simulação, execução, monitoramento e, finalmente, a melhoria do processo (HALL *et al*., 2007) *apud* (PUNTAR *et al*., 2009). Estas atividades são brevemente descritas a seguir:

Figura 2.1: Ciclo de Vida de Processos de Negócio (PUNTAR et al., 2009)

1. **Projeto**: Atividade em que começa o ciclo de vida de um processo. Nesta fase são definidos os pontos que serão explorados, os processos que serão mapeados, as pessoas e as áreas que serão envolvidas. Nesta fase, deve-se procurar as respostas para a seguinte pergunta: "Quem faz o que, em que sequência, quais serviços são fornecidos e quais sistemas de software são usados no processo?" (COSTA, 2009).
2. **Modelagem**: Nesta fase são realizados o levantamento e a documentação do processo. O levantamento das informações do processo pode ser realizado através de entrevistas com os atores do processo e/ou através da análise da documentação já existente. A modelagem é considerada como um passo essencial, haja vista que não é possível gerenciar aquilo que não se conhece. O modelo do processo deverá conter todas as suas atividades e executores definidos. Durante esta etapa, os modelos criados deverão ser devidamente armazenados, de forma a possibilitar as adaptações que serão realizadas no mesmo visando a melhoria do processo.

[1] Processos de software podem ser vistos como um processo de negócio onde o domínio em questão é o desenvolvimento de software.

3. **Simulação**: Após a modelagem, é possível simular a execução de um processo e, dessa forma, verificar o fluxo de informações, testar as regras definidas e principalmente medir a eficiência do processo. Esta atividade permite a identificação de possíveis gargalos e permite efetuar possíveis ajustes no processo.
4. **Execução**: Nesta fase é realizada a automação do processo, na qual se utiliza um motor de execução de processo, que basicamente envia e recebe as tarefas que compõem o processo aos seus executores, sejam eles usuários ou sistemas. Os dados relativos à execução, de cada uma das tarefas, devem ser capturados e armazenados nesta etapa para serem utilizados na fase posterior (Monitoramento).
5. **Monitoramento**: Graças à automação do processo, possibilita-se o seu monitoramento e acompanhamento através de estatísticas geradas, em tempo real, pelo próprio sistema de execução do processo.
6. **Melhoria**: Com base nos dados provenientes do monitoramento / acompanhamento da execução do processo, estes podem ser utilizados para a melhoria do processo definido. Ferramentas para extração de novo conhecimento e / ou mineração de dados que utilizam os conceitos de ontologias, redes neurais, árvores de decisão ou visualização de dados, por exemplo, podem ser utilizadas nesta etapa para minerar os dados obtidos e fornecer informações úteis à melhoria do processo. A partir deste ponto, o ciclo pode ser reiniciado, partindo-se do princípio de que o processo inicialmente pensado foi alterado / melhorado.

O ciclo de vida de processos de software apresenta-se bastante semelhante ao ciclo de vida de processos de negócio. Por exemplo, em (REIS, 2003), o ciclo de vida de um processo de software é descrito contendo as seguintes fases (ou atividades): provisão de tecnologia, análise de requisitos, projeto do processo, instanciação do processo, simulação do processo, execução do modelo de processo e avaliação do processo. De acordo com a proposta deste trabalho, foram inseridas neste ciclo as atividades ou requisitos necessários para englobar a captura da proveniência dos dados do processo. A seguir são detalhadas as fases descritas por REIS (2003).

1. **Provisão de Tecnologia**: inclui o fornecimento de tecnologia de suporte à produção de software e de modelos de processo (como as linguagens de modelagem de processo, modelos de processo prontos para reutilização, ferramentas para aquisição, modelagem, análise, projeto, simulação, evolução, execução, monitoração de modelos de processo e captura da proveniência dos dados do processo).
2. **Análise de Requisitos do Processo**: identifica os requisitos para o projeto de um novo processo ou os novos requisitos para um processo existente.

3. **Projeto do Processo**: nesta etapa, as linguagens de modelagem do processo são utilizadas. Essa fase pode ser também descrita como uma etapa de modelagem do processo, que elícita e captura descrições de processos informais, convertendo-as em modelos de processos formais ou instâncias de modelos de processos.
4. **Instanciação do Processo**: altera a especificação do processo produzido pela atividade de **Projeto do Processo** de forma a acrescentar informações detalhadas sobre os prazos, agentes e recursos utilizados por cada atividade definida no processo.
5. **Simulação do Processo**: esta fase permite a verificação e validação dos processos definidos antes de sua execução.
6. **Execução**: esta fase utiliza o processo instanciado e o executa através da invocação de ferramentas para guiar e assistir a realização do processo modelado. Informações e métricas sobre o andamento do processo são coletadas e analisadas durante a sua execução. Esta fase será melhor explorada na Subseção 2.2.1, haja vista que o foco desta proposta de trabalho está centrado na execução de processos de software.
7. **Avaliação do Processo**: esta fase visa prover informações quantitativas e qualitativas que descrevem o desempenho de todo o processo em execução. A avaliação pode ocorrer em paralelo com a execução do modelo de processo e as informações adquiridas podem ser utilizadas nas futuras ocorrências da atividade de análise de requisitos. Esta avaliação de desempenho do processo pode ser realizada com base na utilização dos dados de proveniência, que foram armazenados durante a etapa de execução do processo, em conjunto com o uso de ferramentas inteligentes de extração de conhecimento, tais como máquinas de inferência, que atuarão sobre estes dados de execução.

Existem diversas outras propostas de ciclo de vida de processos de software além da apresentada por (REIS, 2003), tais como a proposta de NGUYEN e CONRADI (1994), que traz uma classificação de meta-processos, compostos por fases e comparam alguns ambientes com relação à utilização das fases propostas - e a proposta de DERNIAME *et al*. (1999) que propõe um ciclo de vida chamado *PROMOTER Reference Model*. Destas propostas, uma que merece destaque é a de (JØRGENSEN, 2000) *apud* (BARRETO, 2011), que apresenta um "ciclo de vida do reúso de processos de software" que leva em consideração que um processo de software e seu respectivo modelo têm uma natureza evolucionária, devido à necessidade de melhoria e correção contínua, assim como a instabilidade do ambiente operacional. Esta proposta de ciclo de vida é apresentada na *Figura 2.2*, onde as setas são etapas e os vértices (em itálico) são estados. A etapa de

"**Modelagem de Processos Visando a Reutilização**" visa definir, através de uma Linguagem de Modelagem de Processos (*Process Modeling Language – PML*), modelos de processos e componentes genéricos e abstratos a serem reutilizados em diferentes contextos. A etapa de "**Recuperação e Adaptação de Processos**" prevê critérios, estratégias e mecanismos para auxiliar um projetista na seleção, adaptação e instanciação de processos genéricos que forneçam soluções para o novo problema a ser solucionado. Já na etapa "**Execução de Processos**" são descritas todas as ocorrências de um modelo de processo desde o início de sua execução, incluindo as modificações referentes à evolução do processo. Na última etapa, a de "Generalização e Avaliação de Processos Encerrados" é realizada a extração de informações, a partir da experiência adquirida em processos bem-sucedidos, para alimentar o repositório de modelos de processos reutilizáveis.

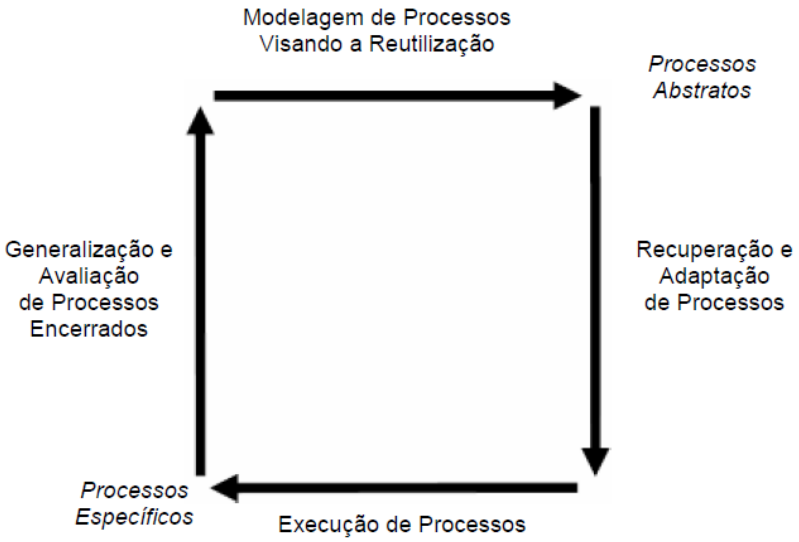


Figura 2.2: Ciclo de Vida do Reúso de Processos de Software (JØRGENSEN (2000) apud BARRETO (2011)).

A partir das definições de ciclo de vida de processos apresentadas anteriormente, foi elaborado o ciclo de vida que será utilizado nesta proposta de trabalho, conforme pode ser visualizado na Figura 2.3.

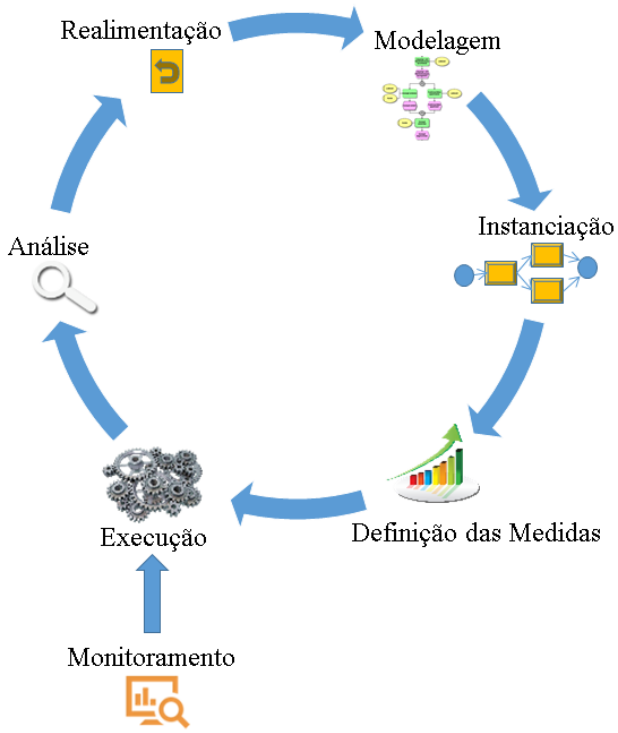


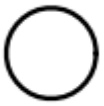
Figura 2.3: Ciclo de Vida de Processos da Abordagem

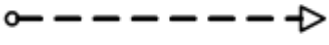
Uma das notações existentes para a modelagem de processos existente é o BPMN (*Business Process Model and Notation*) (OMG, 2011). Esta notação oferece algumas vantagens com relação à notação chamada de SPEM (*Software & Systems Process Engineering Metamodel*) (OMG, 2008), com relação aos formalismos oferecidos que dão suporte à etapa de execução / monitoramento de processos de software. O BPMN foi desenvolvido para facilitar a interpretação humana, técnica e não técnica. Uma vez que os processos são definidos, eles podem ser simulados, executados, monitorados e analisados. De acordo com Pillat *et al*. (2015), embora o SPEM ofereça um grande potencial para a representação de processos de software, este metamodelo não fornece conceitos ou formalismos para a modelagem precisa do comportamento do processo, não podendo ser diretamente mapeado para uma linguagem executável. O SPEM não aborda as fases de simulação, execução, monitoramento e otimização, que são atividades importantes na gestão de processos. Por outro lado, existem no mercado diversas ferramentas baseadas em BPMN (OMG, 2011) para apoiar processos de negócios tanto em sua fase de modelagem quanto nas etapas de execução, monitoramento, análise e melhoria. Estas vantagens apresentadas com relação ao BPMN motivaram a definição desta notação para representar os processos de software desta proposta neste trabalho, conforme será detalhado no Capítulo 5.

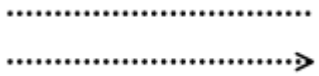
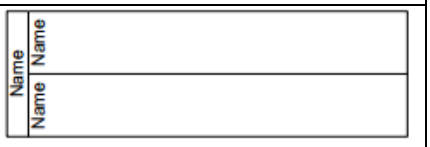
O BPMN (*Business Process Modelling Notation*) (OMG, 2011) é uma notação padrão para modelagem de processos de negócios, com enfoque na análise de domínio e projeto de sistemas de alto nível. A respectiva notação herda elementos de uma série de notações anteriores, combinando à

modelagem de processos de negócio, tais como a linguagem de definição de processo XML (XPDL) (WFMC, 2002) e o Diagramas de Atividades da UML (OMG, 2005). Modelos de processos BPMN são compostos, principalmente por: (i) atividades, denotando eventos de negócios, (ii) itens de trabalho (realizados por seres humanos ou por aplicativos de software) e (iii) itens de controle, que gerenciam o fluxo de controle entre as atividades. A Tabela 5.4 apresenta os principais elementos da notação BPMN.

Tabela 2.1: Principais Elementos BPMN (adaptado de OMG, 2011)

Elemento	**Descrição**	**Notação**
Atividade	Uma atividade é um termo genérico para o trabalho executado por uma empresa em um processo. Uma atividade pode ser atômica (Tarefa) ou não atômica (composta - Subprocesso). As atividades são representadas por retângulos arredondados.	
Evento	Um evento é algo que acontece no decorrer de um processo. Estes eventos podem afetar o fluxo do processo e, geralmente, têm uma causa que o disparam (gatilho) ou um impacto (resultado). Os eventos são círculos com centros abertos para permitir indicadores internos para diferenciar diferentes gatilhos ou resultados. Existem três tipos de eventos: Início, Intermediário e Fim, apresentados a seguir.	
Tipos de Evento	Início: Como o nome indica, o evento de Início indica onde um determinado processo será iniciado. Intermediário: Eventos intermediários ocorrem entre um evento de Início e um evento de Fim. Eles afetam o fluxo do processo, mas não iniciam ou finalizam (diretamente) o processo. Fim: Como o nome indica, o evento de Fim indica o término de um processo.	
Gateway	Um *gateway* é usado para controlar a divergência e a convergência entre as sequências de fluxo em um processo. Assim, ele determinar a ramificação, bifurcação, fusão e união de caminhos.	

Tipos de gateway	Os ícones exibidos dentro do *gateway* indicam o tipo de controle de fluxo do processo, sendo os principais tipos de *gateway*: - *Gateway* Exclusivo: Representa uma condição de fluxo em que apenas um dos caminhos a partir do *gateway* será seguido, de acordo com uma informação a ser testada. Este *gateway* pode ser representado tanto com o losango vazio ou com um "X". Além de realizar separação de fluxos, este *gateway* também pode unificar fluxos distintos em uma única sequência de atividades. Neste caso, o *gateway* exclusivo implica que, dos caminhos que convergem a ele, o primeiro que chegar dará continuidade ao fluxo do processo. - *Gateway* Inclusivo: Representa uma condição de fluxo em que pode haver uma combinação dos caminhos criados a partir do *gateway*, de acordo com uma informação a ser verificada. Este gateway é representado visualmente como o losango com um marcador de círculo dentro dele. Ao atingir este *gateway*, o processo deverá avaliar a condição relacionada, e uma ou mais das saídas do *gateway* poderão ser acionadas. Se este *gateway* for utilizado para realizar a convergência de fluxos, ele garante que todos os fluxos que estiverem em execução sejam concluídos, antes de dar continuidade à sequência de atividades de saída. - *Gateway* Paralelo: Este *gateway* representa a divisão de um fluxo em dois ou mais a serem executados paralelamente. Este *gateway* é representado visualmente como o losango com um marcador de "+" dentro dele. Quando este *gateway* é utilizado para realizar a convergência de fluxos, ele garante que todos os fluxos paralelos sejam concluídos, chegando até ele antes de dar continuidade ao fluxo de saída.	
Fluxo de Sequência	Um fluxo de sequência é usado para mostrar a ordem em que as atividades serão realizadas em um processo.	
Fluxo de Mensagem	Um fluxo de mensagens é usado para mostrar o fluxo de mensagens entre dois participantes do processo que sejam	

	preparados para enviar e receber tais mensagens.	
Associação	Uma Associação é usada para ligar as informações e artefatos com elementos gráficos do BPMN. A ponta da flecha na Associação indica uma direção de fluxo (por exemplo, dados), quando for o caso.	
Piscina (*Pool*)	A piscina (*pool*) é a representação gráfica da colaboração entre participantes. Um participante representa uma entidade específica (por exemplo, uma empresa) e/ou uma entidade genérica (por exemplo, um comprador, vendedor ou fabricante).	
Raia (*Lane*)	Uma raia (*lane*) é uma subpartição dentro de um processo, às vezes dentro de uma piscina (*pool*). As raias são utilizadas para organizar e categorizar as atividades.	
Objeto de Dados	Os objetos de dados fornecem informações para que as atividades possam ser realizadas e/ou armazenam o que é produzido por uma atividade. Estes podem representar um objeto singular ou uma coleção de objetos.	
Mensagem	Uma mensagem é usada para descrever o conteúdo da comunicação entre dois participantes do processo.	

Um exemplo de processo de software definido utilizando a notação BPMN pode ser visualizado na Figura 2.4. Nesta figura, os eventos de Início e Término do processo são representados por circunferências. As tarefas do processo são exibidas pelos retângulos em azul e estas são atribuídas aos papéis definidos na lateral esquerda do processo, tais como Gestor da Demanda, Gerente de Projetos, Desenvolvedor etc. Assim, todas as tarefas incluídas neste processo possuem um nome e um papel, que especifica a função que um ator deverá possuir para ser capaz de realizar a tarefa mencionada. Os *gateways* são representados pelos losangos em amarelo.

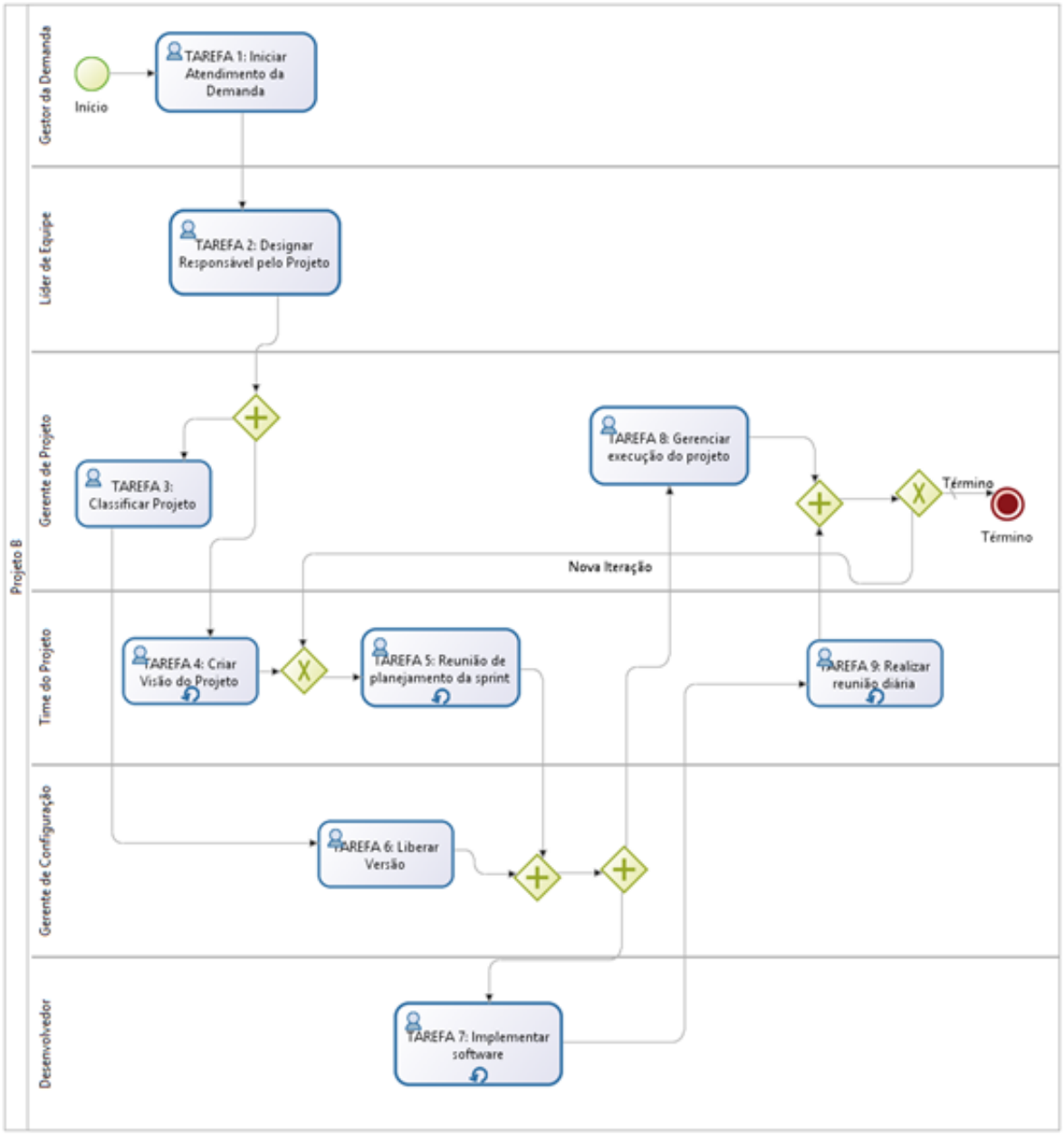


Figura 2.4: Modelo de Processo utilizando a notação BPMN (COSTA et al., 2014)

Após a modelagem do processo, é possível realizar a instanciação do mesmo, acrescentando informações detalhadas sobre os prazos, agentes e recursos a serem utilizados pelas atividades definidas no modelo de processo.

Embora possam parecer corretos do ponto de vista da Engenharia de Software, muitos processos de software podem não ser competitivos no mercado, seja por consumirem tempo ou esforço demais ou por não ser capaz de gerar os produtos com a qualidade esperada pelo usuário. Por este motivo, processos devem estar em constante melhoria (ROCHA *et al.*, 2012).

Diversos *frameworks* que apoiam a implantação de programas de melhoria de processos em organizações de desenvolvimento de software podem ser citados, tais como as normas

internacionais ISO/IEC 12207 – *Systems and Software Engineering – Software Life Cycle Process* (ISO/IEC, 2008) e ISO/IEC 15504 – *Information Technology – Process Assessment* (ISO/IEC, 2003) e o modelo CMMI-DEV – *Capability Maturity Model Integration for Development* (SEI, 2010), além do modelo nacional, o MR-MPS – Modelo de Referência para Melhoria de Processo do Software Brasileiro (SOFTEX, 2011).

A melhoria de processos de software pode estar relacionada aos seguintes objetivos: buscar por níveis mais altos de maturidade e / ou realizar mudanças tendo em vista maior adequação às necessidades da organização ou melhorias no desempenho dos processos (ROCHA *et al*., 2012). De acordo com o contexto deste trabalho, busca-se, através da realimentação do processo, prover uma melhoria no desempenho deste.

De acordo com ROCHA *et al*. (2012), medições são essenciais para a aplicação de melhorias em processos de software, haja vista que fornecem dados que permitem conhecer o seu desempenho. Tais dados possibilitam a detecção de problemas no desempenho e inadequações nos processos.

Um processo de medição poderia ter três atividades principais, de acordo com o MR-MPS (SOFTEX 2011): (1) Definir objetivos e medidas; (2) Especificar procedimentos de coleta, armazenamento e análise; e (iii) Coletar, analisar e reportar medidas.

A fase de execução de processos de software é realizada através da invocação de ferramentas para guiar e assistir a realização do processo, tendo como base para tal o modelo do processo instanciado. A execução de processos deve ser capaz de: (a) **automatizar o processo**: apoio à coordenação de atividades e ativação automática das atividades que podem ser executadas sem intervenção humana; (b) **suportar o trabalho cooperativo**: apoio à cooperação de pessoas trabalhando em um projeto de software durante o ciclo de desenvolvimento; (c) **suportar a monitoração**: prover diferentes visões do estado da execução do processo, permitindo ao gerente de projetos obter informações sobre o andamento das atividades; (d) **suportar o registro do histórico do processo**: coleta de dados da evolução do processo para permitir que o processo possa ser melhorado onde houver necessidade (REIS, 2003).

O mecanismo de execução de processos deve ser construído levando em consideração a semântica da linguagem que foi utilizada para a modelagem do processo. A execução de um processo pode ser realizada a partir do momento em que se obtém um modelo de processo executável (ou instanciado), ou seja, um modelo que descreva o processo com tal nível de detalhe que permita a sua execução por uma máquina. REIS (2003) apresenta a existência de formalismos de representação e paradigmas de análise e execução de processos, porém, ressalta que muitas abordagens não podem ser classificadas em um único paradigma, sendo estas conhecidas como multiparadigma. Estes formalismos são brevemente descritos a seguir:

- **Execução Procedimental**: Considera que processos de software devem ser modelados em paradigmas similares à programação convencional. Um processo é modelado através de instruções passo a passo que descrevem como o processo deverá ser executado e estruturas de controle especificam a ordem dos passos do processo. Neste paradigma, todas as situações do processo precisam ser especificadas antes do início da execução do mesmo.
- **Execução Baseada em Regras**: As atividades do processo neste paradigma são modeladas como regras com pré e pós-condições. O mecanismo de execução assemelha-se às máquinas de inferência de sistemas especialistas. Como desvantagem deste paradigma pode-se citar que, apesar do gerente ter a opção de monitorar a execução das regras, o processo de desenvolvimento de software geral não fica visível e é de difícil monitoração.
- **Execução Baseada em Regras ECA (Evento-Condição-Ação)**: Neste formalismo, diferentemente da execução baseada em regras, existe a possibilidade de detecção de eventos antes da execução da regra, ou seja, o mecanismo de execução verifica a ocorrência de eventos e a pré-condição das regras, e, após isto, pode disparar as regras que satisfazem a condição e o evento, gerando novos eventos a serem tratados. Como desvantagem deste formalismo pode-se citar a dificuldade de controlar a execução e também a dificuldade para entendimento humano deste formalismo.
- **Execução Baseada em Redes de Petri**: Adota o formalismo matemático de Redes de Petri (um grafo direcionado com dois tipos de nodos: lugares e transições - os lugares representam os predicados, objetos ou tipos de recursos e as transições representam os passos do processo e podem ser rotuladas com pesos). Através desse formalismo, é possível representar não-determinismo, visto que um lugar pode ser entrada para várias transições, possibilitando também a representação de paralelismo. Como desvantagem deste formalismo pode-se citar que Redes de Petri ainda são complexas para serem compreendidas por usuários não técnicos.
- **Execução Baseada em Redes de Tarefas**: Consiste em um grafo direcionado que representa a estrutura do processo. Em geral, os nodos representam atividades e os arcos representam o fluxo de controle e de dados entre atividades. A semântica de execução para este paradigma, geralmente, é definida em função dos fluxos de controle, dados e construtores providos pela linguagem a ser utilizada. Assim, este tipo de execução não necessita ser baseada em algum formalismo já existente, o que pode ser visto como uma vantagem deste paradigma.

Tendo como objetivo assegurar que a execução das tarefas de um processo está ocorrendo de acordo com o esperado, estas precisam ser monitoradas / acompanhadas constantemente (MINGHUI *et al*., 2004). Sendo assim, a etapa de monitoramento visa medir o progresso do processo e identificar desvios com relação ao que foi previamente planejado. Durante o monitoramento é possível estabelecer ações corretivas, adotadas para o cumprimento dos objetivos do projeto.

Como exemplos de ferramentas que permitem a execução e o monitoramento de processos que enquadram-se na categoria de BPMS (*Business Process Management Systems*) dentre as diversas existentes, pode-se citar: Bonita Open Solution (BONITASOFT, 2015), Bizagi (BIZAGI, 2015), FLUIG (FLUIG, 2015), Intalio BPMS (INTALIO, 2014), eClarus Business Process Modeler for SOA Architects (ECLARUS, 2015) e jBPM Business Process Management Suite (RED HAT, 2015).

2.3 Considerações sobre o Capítulo

Neste capítulo foram apresentados tanto o conceito de processo de software quanto o ciclo de vida de processos. Um ciclo de vida baseado nos existentes foi proposto e será utilizado na abordagem a ser implementada. Deve-se ressaltar que, visando atingir o objetivo desta proposta de trabalho, durante as etapas de modelagem, execução/monitoramento e análise da execução do processo, podem ser usados modelos de proveniência, tais como OPM (MOREAU *et al*., 2011) e PROV (BELHAJJAME *et al*.2012) como uma alternativa para prover um modelo padrão para captura destes dos dados de execução do processo e, posteriormente, para auxiliar na derivação de informações que possam influenciar na melhoria do desempenho do processo. O Capítulo seguinte apresenta os principais conceitos relacionados ao tema proveniência de dados.

3. PROVENIÊNCIA DE DADOS

Neste capítulo, são apresentados os conceitos relacionados a proveniência de dados e seus principais modelos, sendo eles o OPM (MOREAU *et al*., 2011) e o PROV (BELHAJJAME *et al*.2012). Uma abordagem para utilização do modelo de proveniência PROV para processos de software é descrita na Seção 3.3. Por fim, na Seção 3.4, tem-se as considerações finais do capítulo.

3.1 Introdução

Proveniência de dados é um registro da história da derivação dos dados, que possibilita a reprodutibilidade, interpretação dos resultados e diagnóstico de problemas (LIM *et al.*, 2010).

BUNEMAN (2001) define proveniência de dados como a descrição da origem de uma parte de um dado e o processo pelo qual este passou. A captura da proveniência pode ser feita considerando tanto os dados quanto os processos.

FREIRE *et al*. (2008) relatam que a proveniência pode ser capturada de forma prospectiva e retrospectiva. A forma prospectiva captura os passos a seguir para a geração de um dado produto, possibilitando o registro da especificação de tarefas computacionais, como por exemplo, um conjunto de processos e/ou um *script*. Já a proveniência retrospectiva captura os passos executados por uma tarefa computacional, assim como a informação sobre o ambiente utilizado para derivar um produto específico, ou seja, um *log* detalhado sobre a execução da tarefa.

De acordo com SIMMHAN (2007), a proveniência de processos envolve a descrição das tarefas que fazem parte de um processo. Em determinados processos, para que se tenha a informação dos dados geradores de um dado produto, é também importante que se registre cada dado consumido pelo processo. Assim, na captura da proveniência de um processo, é necessário obter a descrição da execução de todas as tarefas efetivadas para que se tenha a informação relativa ao sucesso ou insucesso destas tarefas durante a execução.

Para obtenção dos benefícios das informações de proveniência, é necessária a captura destas, bem como a modelagem e armazenamento de modo integrado para posterior consulta (MARINHO, 2011). Assim, a utilização da proveniência de dados conta com dois principais modelos propostos na literatura, sendo eles OPM (MOREAU *et al*., 2008) e PROV (BELHAJJAME *et al*. 2012).

Uma das formas para se analisar e verificar a qualidade dos dados gerados por processos de software é através da utilização de técnicas e modelos de proveniência de dados. O uso de proveniência auxilia também na execução dos processos (MILES *et al*., 2011). A aplicação da proveniência de dados não interfere na execução dos processos e ainda permite ao usuário refinar as regras de filtragem aplicadas para coleta dos dados (GHOSHAL *et al*., 2013). Neste contexto, o

presente trabalho visa utilizar os dados de proveniência para auxiliar na melhoria de processos de software e, a seguir, é apresentado um breve detalhamento de como a proveniência de dados pode ser inserida em algumas das etapas do ciclo de vida de processos.

Na fase de modelagem do processo, os modelos criados deverão ser devidamente armazenados, de forma a possibilitar a captura da proveniência prospectiva (proveniência relacionada à especificação do processo). Já durante a fase de execução do processo, o uso de modelos de proveniência, tais como OPM (MOREAU *et al*., 2011) e PROV (BELHAJJAME *et al*., 2012) podem ser utilizados como uma alternativa para prover um modelo padrão para captura destes dados de execução. Posteriormente, na fase de análise do processo, os dados de proveniência armazenados serão utilizados para a derivação de informações que poderão contribuir com a melhoria do processo definido inicialmente.

3.2 Modelos de Proveniência de Dados

A utilização da proveniência de dados conta com modelos propostos na literatura, dentre os quais destaca-se o OPM (MOREAU, 2008) e o PROV (GROTH, 2013). Nas subseções a seguir, estes modelos são detalhados.

3.2.1 OPM

O modelo OPM foi criado de forma a permitir a interoperabilidade dos metadados de proveniência entre sistemas e encontra-se na versão 1.1. Este modelo foi projetado para atender aos seguintes requisitos (MOREAU *et al*., 2011):

- Permitir que informações de proveniência possam ser trocadas entre os sistemas por meio de uma camada de compatibilidade, com base em um modelo de proveniência compartilhado;
- Permitir que os desenvolvedores criem ferramentas que operem em tais ações de modelos de proveniência;
- Definir o modelo de uma maneira precisa à tecnologia a ser utilizada;
- Apoiar uma representação digital de proveniência para qualquer "coisa", produzida, ou não, por sistemas de computador; e
- Definir um conjunto de regras que identificam as inferências válidas e possam gerar gráficos de proveniência.

O OPM utiliza um grafo para representar as informações de proveniência. Neste grafo, são usados vértices chamados de *artefatos* (parte imutável de estado, que pode ter um corpo físico em um objeto físico ou uma representação digital em um sistema de computador), *processos* (ação ou série de ações executadas ou causadas por artefatos, e que resultam em novos artefatos) ou *agentes* (entidade contextual agindo como um catalisador de um processo, permitindo, facilitando, controlando ou afetando sua execução). Para capturar as dependências entre os *artefatos*, *processos* e *agentes*, utiliza-se uma aresta, que representa a dependência causal entre a sua fonte, denotando o efeito e o seu destino, que denota a causa. O OPM possui 5 relações de dependência causal, sendo respectivamente: *Used* (Utilizou), *wasGeneratedBy* (FoiGeradoPor), *wasControlledBy* (FoiControladoPor), *wasStartedBy* (FoiInicializadoPor) e *wasDerivedFrom* (FoiDerivadoDe) (MOREAU *et al*., 2011). Na Figura 3.1, é possível visualizar os vértices e as arestas (dependências causais) existentes no modelo OPM.

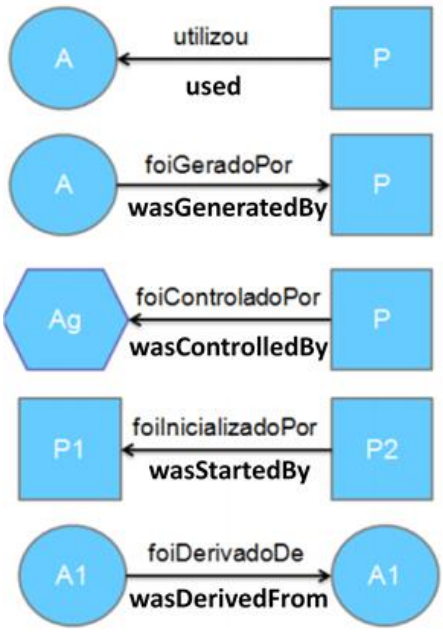


Figura 3.1: Nós e dependências causais do OPM

Recentemente, foi especificado por um grupo de trabalho do W3C, um modelo de proveniência mais genérico, denominado PROV. O PROV apresenta, além das características básicas do OPM, novas construções e alterações. Dois nós existentes no OPM foram alterados e novas relações causais foram criadas (BELHAJJAME *et al*. 2012). Neste trabalho, tanto para o armazenamento quanto para a captura dos dados de proveniência dos processos de software será utilizado o modelo PROV (BELHAJJAME *et al*. 2012), por ser um modelo mais atual, por possuir relações causais que não são explicitadas no OPM (MOREAU *et al*., 2008) e devido à necessidade

de maior especificidade e menor nível de abstração para o armazenamento das informações de proveniência de processos de software. O modelo PROV é detalhado na próxima subseção.

3.2.1 PROV

O PROV (BELHAJJAME *et al.*, 2012) é um modelo de proveniência especificado pela W3C, que objetiva expressar a proveniência de dados pela descrição das entidades, atividades e agentes envolvidos em produzir, entregar ou enunciar um determinado objeto.

O modelo PROV também utiliza um grafo para representar as informações de proveniência, pela utilização de três tipos de vértices (Figura 3.2), e sete tipos de arestas principais (Figura 3.3). O PROV se diferencia do OPM pela nomenclatura de dois vértices e por apresentar novas arestas. O vértice representado por um círculo, no PROV é denominado Entidade e no OPM o mesmo é chamado Artefato. Já o vértice representado por um retângulo é denominado Atividade e no OPM o mesmo é chamado de Processo.

Figura 3.2: Vértices do Modelo PROV

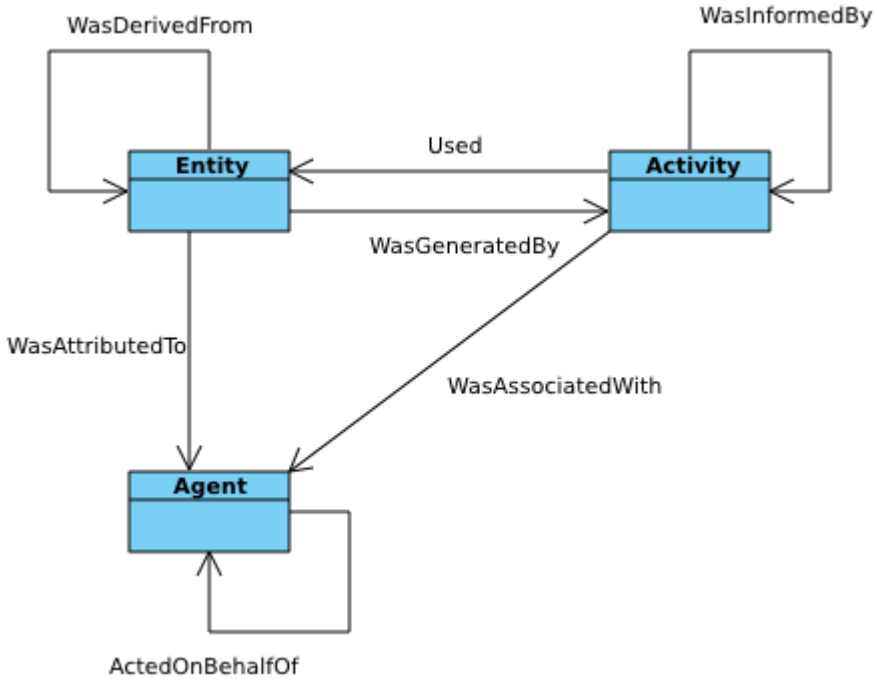


Figura 3.3: Principais Arestas do Modelo PROV (BELHAJJAME et al. *2012)*

De acordo com BIVAR *et al.* (2013), as dependências causais do PROV podem ser descritas da seguinte forma:

- *used*: relaciona atividades e entidades, afirmando que uma atividade usou uma entidade.
- *wasGeneratedBy*: Relaciona entidades a atividades e indica que uma entidade foi gerada por uma atividade.
- *wasAssociatedWith*: Relaciona atividades e agentes, indicando que uma determinada atividade foi associada a um determinado agente.
- *wasAttributedTo*: Relaciona entidades e agentes e indica que uma entidade foi atribuída a um determinado agente.
- *actedOnBehalfOf*: Relaciona agentes indicando que um agente tem autoridade ou responsabilidade por outro agente.
- *wasDerivedFrom*: Relaciona entidades, no sentido de que uma entidade foi originada da outra. Esta derivação tem o caráter evolutivo e não corretivo.
- wasInformedBy: Relaciona atividades, indicando que uma atividade A2 é dependente de outra atividade A1, por meio de uma entidade não especificada, que é gerada por A1 e utilizado por A2.

De forma a possibilitar a captura e posterior recuperação de dados de proveniência de processos de desenvolvimento de software utilizando o modelo PROV, encontra-se em desenvolvimento um trabalho de mestrado que consiste em uma arquitetura que permita o armazenamento e a captura dos dados de execução dos processos, chamada PROV-Process. Pretende-se utilizar a abordagem PROV-Process nesta tese para atingir os objetivos de armazenar os dados relativos aos processos de software a serem executados. A abordagem PROV-Process é brevemente apresentada a seguir.

3.3 PROV-Process

A abordagem PROV-Process consiste em uma arquitetura especificada para captura e armazenamento dos dados de proveniência, utilizando o modelo PROV. Tanto os registros gerados para a especificação dos modelos de processo quanto os dados de execução das tarefas de processos relevantes para a proveniência serão capturados, utilizando o modelo PROV. A partir da sumarização destes dados, utilizando regras de inferência e técnicas de mineração de dados, serão identificados dados relevantes para auxiliar na melhoria do processo. Aos dados capturados, poderão ser aplicadas tecnologias de web semântica, técnicas de visualização ou mesmo análises

estatísticas sobre os registros coletados, visando obter indicações de melhorias para os processos em uso. Dentre as tecnologias a serem utilizadas, destaca-se o uso de ontologias, máquinas de inferência e técnicas de mineração de dados.

O diagrama da base de dados utilizada pela abordagem PROV-Process é apresentado na Figura 3.4., o qual foi criado de acordo com as especificações do PROV-DM (MOREAU *et al*., 2013). PROV-DM é o modelo conceitual de dados que constitui a base W3C de especificações de proveniência (PROV). Distingue estruturas centrais, formando um núcleo de informações de proveniência, a partir do qual é possível estender estruturas para uso mais específico de proveniência. O modelo PROV-DM está organizado em seis componentes, sendo respectivamente: (1) entidades, atividades e o momento em que são criados, utilizados ou terminam; (2) derivações entre entidades; (3) agentes responsáveis por entidades que foram geradas e atividades que ocorreram; (4) noção de conjunto, um mecanismo de apoio à proveniência de proveniência; (5) propriedade de vincular entidades que se referem à mesma coisa; e (6) coleções que formam uma estrutura lógica para os seus membros (MOREAU *et al*., 2013).

O diagrama de tabelas relacionais, disposto na Figura 3.4, destaca as tabelas que fazem parte dos respectivos componentes, onde as mesmas apresentam colorações distintas, representando os componentes citados. A cor rosa, representa o componente 1 (Entidades e atividades), a cor verde, o componente 2 (Derivações). A cor azul denota o componente 3 (Agentes, Responsabilidade e Influência), já a cor amarela representa o componente 5 (Alternativo) e, por fim, a cor roxa representa o componente 6 (Coleções). O componente 4 (Bundles), encontra-se representado no diagrama como atributo da tabela Entity, haja vista que este trata-se de um tipo de entidade.

A tabela Entity possui um identificador (idEntity), conforme especificação do PROV-DM, o qual indica ainda a possibilidade de atributos opcionais. Para este modelo, foram criados os atributos Name e Type, os quais representam, respectivamente, o nome e o tipo da entidade, sendo que o tipo da entidade deve ser preenchido com Plan ou Bundle ou Collection ou EmptyCollection, por tratarem-se de tipos de entidades.

A tabela Activity é composta por um identificador (idActivity), um atributo startTime para registro do início da atividade e um atributo endTime para registro do fim da atividade, conforme especificação do PROV-DM, o qual indica ainda a possibilidade de atributos opcionais. Para este modelo, foi criado o atributo Name para informação do nome da atividade.

A tabela wasGeneratedBy possui um identificador (idWasGeneratedBy) e um atributo Entity_idEntity para identificar uma entidade criada, conforme especificado pelo modelo PROV-DM. Os atributos Activity_idActivity e Time, indicados como opcionais no PROV, também foram inseridos, representando, respectivamente, um identificador para atividade que cria entidade e o tempo de criação da entidade.

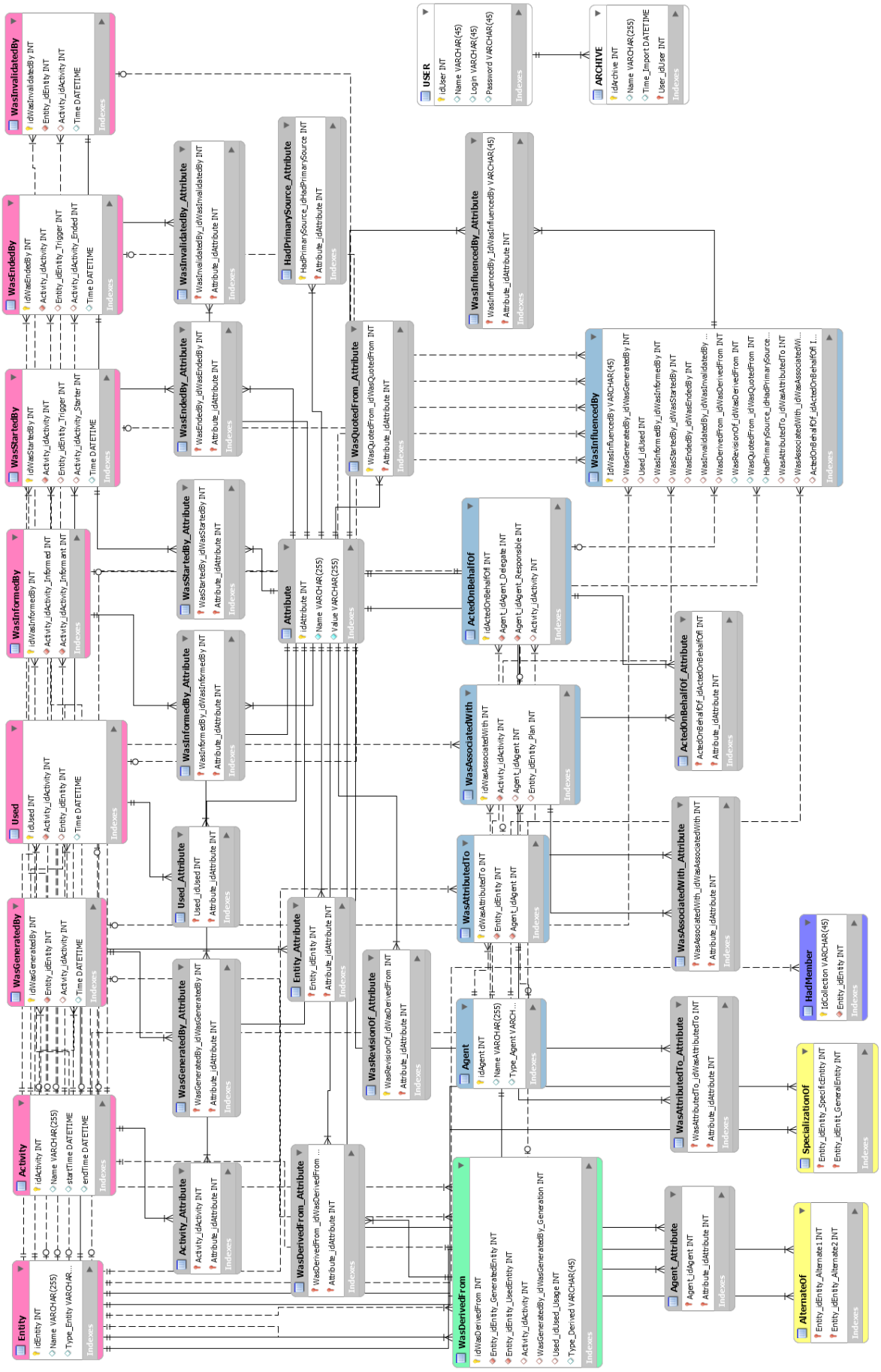


Figura 3.4: Modelo do Banco de Dados da Abordagem PROV-Process

A tabela Used possui um atributo Activity_idActivity para identificação da atividade consumida e um atributo Entity_idEntity para identificação da entidade consumida, conforme definido no PROV. Os atributos idUsed e Time, especificados como opcionais no PROV, constam na tabela e indicam, respectivamente, o identificador e o tempo em que a entidade começou a ser utilizada.

A tabela wasStartedBy possui um atributo Activity_idActivity, para identificação da atividade iniciada, conforme definido no PROV. Os atributos idWasStartedBy, Entity_idEntity_Trigger, Activity_idActivity_Started e Time, especificados como opcionais no PROV, constam na tabela e indicam, respectivamente, um identificador para o início da atividade, um identificador para entidade desencadeando uma atividade, um identificador para atividade que gerou a entidade e o tempo em que a atividade foi iniciada.

A tabela wasEndedBy possui um atributo Activity_idActivity, para identificação da atividade finalizada, conforme definido no PROV. Os atributos idWasEndedBy, Entity_idEntity_Trigger, Activity_idActivity_Ended e Time, especificados como opcionais no PROV, constam na tabela e indicam, respectivamente, um identificador para o final da atividade, um identificador para entidade desencadeando uma atividade final, um identificador para atividade que gerou a entidade e o tempo em que a atividade foi terminada.

A tabela WasInvalidatedBy possui um atributo Entity_idEntity, para identificação de entidades inválidas, conforme definido no PROV. Os atributos idWasInvalidatedBy, Activity_idActivity e Time, especificados como opcionais no PROV, constam na tabela e indicam, respectivamente, um identificador para uma invalidação, um identificador para a atividade que invalidou a entidade e o tempo em que a entidade começou a ser anulada.

A tabela WasDerivedFrom possui um atributo Entity_idEntity_GeneratedEntity, para identificação da entidade gerada pela derivação e um atributo Entity_idEntity_UsedEntity, para identificação da entidade utilizada pela derivação, conforme definição do PROV. Os atributos idWasDerivedFrom, Activity_idActivity, WasGeneratedBy_idWasGeneratedBy_Generation e Used, especificados como opcionais no PROV, constam na tabela e indicam, respectivamente, um identificador para um derivação, um identificador para atividade usando e gerando as entidades acima, um identificador para a geração, envolvendo a entidade gerada e a atividade, e um identificador para o uso que envolve a entidade usada e a atividade. O atributo Type_Derived deve ser preenchido com WasRevisionOf ou WasQuotedOf ou HasPrimarySource, indicando o tipo de derivação, sendo este não obrigatório.

Dando prosseguimento ao modelo, de acordo com a especificação PROV-DM, a tabela Agent possui um atributo idAgent, para a identificação de um agente, um atributo Name, para especificação do nome do mesmo, e um atributo Type_Agent, o qual deve ser preenchido com

Person ou Organization ou SoftwareAgent, para a definição de categoria de agentes, em uma perspectiva de interoperabilidade.

A tabela WasAttributedTo possui um atributo Entity_idEntity, para a identificação de entidade e um atributo Agent_idAgent, que se refere ao identificador do agente a quem a entidade é atribuída, indicando responsabilidade por sua existência, conforme definição do PROV. O atributo opcional, indicado no PROV, foi criado como idWasAttributedTo, o qual trata-se de um identificador para a relação.

A tabela wasAssociatedWith, apresenta o atributo Activity_idActivity, um identificador para a atividade, conforme definido no PROV. Os atributos opcionais, dispostos no PROV e criados nesta tabela, correspondem a idWasAssociatedWith, Agent_idAgent e Entity_idEntity_Plan, indicando respectivamente, um identificador para a associação entre uma atividade e um agente, um identificador para o agente associado a atividade e um identificador para o plano do agente invocado no âmbito desta atividade.

A tabela ActedOnBehalfOf possui um atributo Agent_idAgent_Delegate, para a identificação do agente associado a uma atividade e um atributo Agent_idAgent_Responsabile, para a identificação do agente em nome do qual o agente delegado agiu, conforme definição do PROV. Os atributos opcionais constantes no PROV e criados na tabela, correspondem a idActedOnBehalfOf e Activity_idActivity, indicando, respectivamente, um identificador para o link entre delegado e responsável e um identificador de uma atividade, a qual detém o link delegado.

Na sequência dos componentes do PROV, encontram-se os Bundles, os quais, no modelo desenvolvido, são especificados na tabela Entity, atributo Type_Entity, haja vista que se tratam de tipos de entidade.

A tabela AlternateOf é formada por um auto-relacionamento com a tabela Entity, possuindo o atributo Entity_idEntity_Alternate1, que se trata de um identificador da primeira das duas entidades, e o atributo idEntity_Alternate2, que indica um identificador da segunda das duas entidades.

Assim como a tabela supracitada, a tabela SpecializationOF também se trata de um auto-relacionamento com a tabela Entity, onde o atributo Entity_idEntity_SpecificEntity indica um identificador da entidade que é uma especialização da entidade geral, e o atributo Entity_idEntity_GeneralEntity indica um identificador da entidade que está sendo especializada.

Por fim, a tabela HadMember, é formada pelo atributo idCollection, que se refere a um identificador para coleção de membros, e pelo atributo Entity_idEntity, que indica o identificador de uma entidade que é membro da coleção.

Com os registros de execução do processo, contidos no arquivo importado para o sistema, será possível identificar, por exemplo, os agentes e atividades de um dado processo. Portanto, torna-

se possível a análise do que foi realizado durante a execução de uma determinada atividade em outras instâncias de um mesmo processo, visando identificar a reincidência, ou não, desta tarefa ou tarefas similares a esta. O uso da proveniência aplicar-se-á ao histórico de todas as atividades deste tipo que foram realizadas, verificando o que foi feito e as correções efetuadas. Mediante a identificação das correções já aplicadas e o impacto das alterações em outros fluxos, pode-se identificar a origem do erro para tratamento devido e inferir a ação a ser tomada para evitar reincidências e/ou melhorar o processo como um todo.

Após o armazenamento dos dados de execução do processo na base de dados anteriormente apresentada, estes são exportados como indivíduos para ontologia especificada a partir do modelo PROV-DM, chamada PROV-O (BELHAJJAME *et al.*, 2013), de forma a permitir a inferência sobre estes dados, visando obter informações para a melhoria do processo. Esta ontologia define tanto os vértices do PROV (Agente, Entidade e Atividade), utilizando classes, como as arestas, utilizando propriedades. As principais classes e propriedades da PROV-O são mostradas na Figura 3.5.

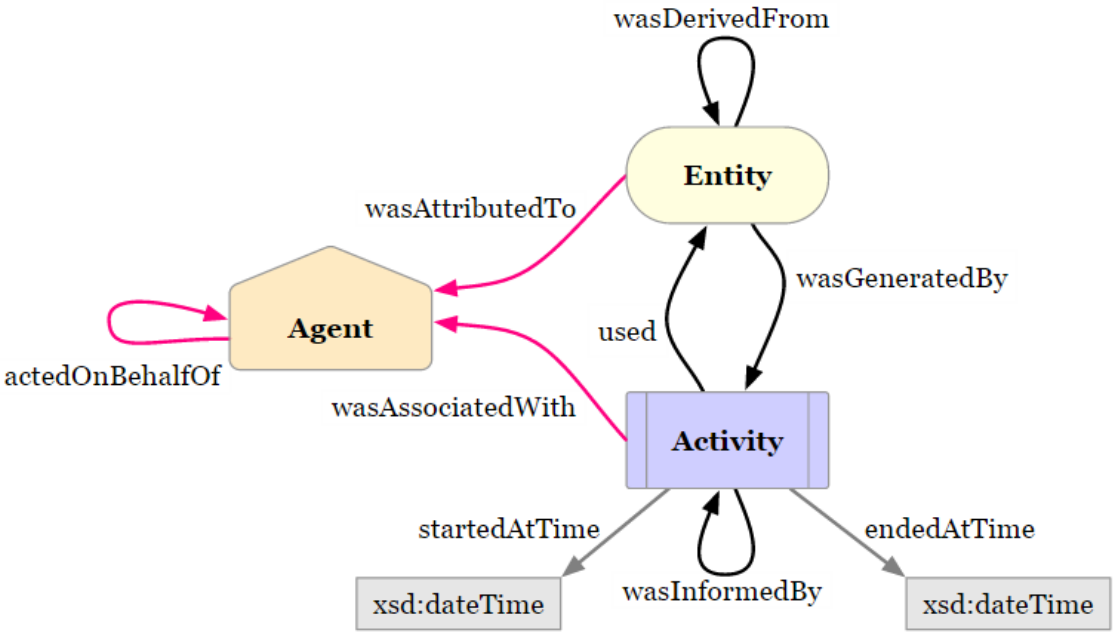


Figura 3.5: Principais classes e propriedades da PROV-O (BELHAJJAME et al., 2013)

3.4 Considerações sobre o Capítulo

Este Capítulo apresentou os principais conceitos relacionados a proveniência de dados que serão utilizados nesta proposta de pesquisa, tais como o modelo PROV, sua respectiva ontologia, chamada PROV-O, e a abordagem PROV-Process, que se encontra em desenvolvimento.

O modelo de proveniência PROV, implementado pela abordagem PORV-Process, será utilizado tanto durante a execução do processo, de forma a permitir a captura e armazenamento dos dados de execução como um mecanismo para auxiliar o gerente do processo no monitoramento do mesmo. Considerando que o monitoramento do processo consiste no rastreamento de sua execução, os dados capturados utilizando os modelos de proveniência, juntamente com a aplicação de técnicas de visualização de software, permitirão que informações relacionadas ao estado do processo possam ser facilmente vistas e entendidas pelo gerente responsável, auxiliando-o na tomada de decisão.

Após a execução do processo, na fase de análise/avaliação deste, a análise dos dados de proveniência, em conjunto com o uso da ontologia PROV-O, permitirá avaliar como o processo foi realizado e o que poderá ser melhorado em uma próxima execução do mesmo, além de permitir uma auditoria detalhada dos dados e das tarefas que foram executadas (CALLAHAN *et al*., 2008).

4. REVISÃO DA LITERATURA

Partindo do ciclo de vida de Processos de Software (Seção 2.2), este capítulo tem como objetivo apresentar uma análise de trabalhos que apresentam / descrevam a(s) fase(s) de execução e/ou monitoramento / avaliação de processos de software.

Esta proposta de tese surgiu a partir do projeto CDSoft (Balanceando Colaboração e Disciplina em Processos de Desenvolvimento de Software)[2] (WERNER *et al*., 2011a), que tem como objetivo definir uma abordagem e desenvolver um ferramental de apoio à composição de processos de software, utilizando uma sistemática baseada em Linha de Processos. Assim, uma seleção inicial de trabalhos relacionados foi realizada, a partir de uma revisão sistemática sobre trabalhos propostos na área de reutilização de processos de software, com foco em abordagens de Linha de Processos de Software. Esta seleção inicial foi realizada visando identificar potenciais trabalhos que poderiam ser utilizados como controle em uma busca estruturada sobre execução e monitoramento de processos de software, de forma geral, e não apenas no contexto de Linha de Processos.

4.1 Revisão Bibliográfica Inicial

Tem-se como objetivo desta revisão bibliográfica inicial a análise de publicações que tratam sobre a execução e o monitoramento de processos, com o propósito de caracterizar abordagens (como ferramentas, métodos, processos e técnicas), com relação à execução/monitoramento destes processos, do ponto de vista de pesquisadores, no contexto acadêmico e da indústria.

Uma seleção inicial de trabalhos foi feita a partir da revisão sistemática realizada por (TEIXEIRA, 2014), que esta aborda uma análise dos trabalhos propostos na área de reutilização de processos com foco em abordagens de linha de processo de software.

A revisão apresentada por TEIXEIRA (2014) traz um total de 47 trabalhos, listados no APÊNDICE A. Tendo como objetivo a seleção dos trabalhos de controle para a busca estruturada, um filtro foi aplicado a estes 47 trabalhos, em conjunto com a própria autora da revisão citada, de forma a identificar os trabalhos relacionados à execução e monitoramento de processos. Foram estabelecidos dois critérios distintos para a seleção dos trabalhos, sendo respectivamente:

[2] Site: http://reuse.cos.ufrj.br/cdsoft/

1. **Trabalhos que abordem execução de processos, apresentando como esta deve ser realizada e/ou ferramentas de apoio para esta etapa do ciclo de vida de processos;**
2. **Trabalhos que abordam sobre monitoramento, avaliação ou melhoria de processos.**

Dos 48 trabalhos apresentados na revisão de TEIXEIRA (2014), foram selecionados apenas aqueles que atendiam, no mínimo, a um dos dois critérios apresentados anteriormente. Neste primeiro filtro, 13 artigos foram selecionados, cujas referências encontram-se listadas na Tabela 4.1. Deve-se ressaltar que alguns dos trabalhos listados no APÊNDICE A foram agrupados nesta etapa, pois, após a leitura dos mesmos, identificou-se que se tratava de um mesmo trabalho, com mais de uma publicação, como é o caso de DAI e LI (2007a) (2007b), DAI et al. (2008) e GARY e LINDQUIST (1999a)(1999b). Já os três primeiros trabalhos, cujo primeiro autor é ALEIXO, F., foram tratados de forma separada, haja vista que, de fato, tratam-se de trabalhos distintos.

Tabela 4.1: Listagem dos trabalhos selecionados de TEIXEIRA (2014)

	Referência
1	ALEIXO et al. (2010)
2	ALEIXO et al. (2012)
3	ALEIXO (2013)
4	AVRILIONIS et al. (1996)
5	BARRETO (2011)
6	COULETTE et al. (2000)
7	DAI e LI (2007a)(2007b) e DAI et al. (2008)
8	GARY e LINDQUIST (1999a)(1999b)
9	HALLERBACH et al. (2008)
10	JAUFMAN e MÜNCH (2005)
11	MARTÍNEZ-RUIZ et al. (2013)
12	SIMMONDS et al. (2012)
13	TORTORELLA e VISAGGIO (1997)

Para a análise dos trabalhos selecionados pelo filtro anterior, cinco questões foram elaboradas com o objetivo de investigar as soluções existentes para execução e avaliação de processos de software.

1. A primeira questão foi elaborada para caracterizar os trabalhos que tratavam especificamente de processos de software e os que tratavam de processos de uma forma geral ou de processos de outras áreas de conhecimento. Esta questão visa verificar se a execução / monitoramento / avaliação de processos de software pode

ser tratada da mesma forma que processos de uma forma geral ou se estas fases, em processos de software, devem ser tratadas diferentemente de processos existentes em outras áreas.

2. A segunda questão foi elaborada visando identificar padrões para a modelagem do processo a ser executado, já que, na maior parte das vezes, a linguagem ou padrão de modelagem utilizado servirá como base para a definição da máquina de execução do processo a ser utilizada.
3. Como terceira questão, buscou-se identificar nos trabalhos as máquinas de execução ou ferramentas propriamente ditas que foram usadas para executar o processo modelado posteriormente.
4. A quarta questão tem por objetivo elicitar como é realizado o monitoramento do processo, se existe alguma maneira sistematizada para tal ou ferramenta específica.
5. A quinta questão visa mapear quais tipos de informações ou métricas de execução são armazenadas durante a execução do processo, visando auxiliar na etapa de avaliação deste processo. Desta forma, as cinco questões elaboradas foram:

1. O processo a ser executado é um processo de software?
2. Como o processo a ser executado foi modelado (usa algum padrão)?
3. Qual a máquina de execução / ferramenta utilizada para execução?
4. Como é realizado o monitoramento do processo?
5. Quais informações/métricas de execução são armazenadas e como se dá o acesso a estas informações relativas à execução do processo?

A seguir são apresentadas as informações extraídas a partir da leitura dos 13 artigos selecionados. Para cada um deles, foram respondidas as cinco questões de pesquisa mencionadas anteriormente.

ALEIXO *et al.* (2010) apresentam uma abordagem para o gerenciamento de variabilidades em processos de software e a derivação automática destes processos, no contexto de famílias de processos de software. É apresentada uma implementação da abordagem proposta usando o processo OpenUP[3] e as ferramentas GenArch[4] e Eclipse Process Framework (ECLIPSE FOUNDATION, 2014).

A execução de processos em LPrS é apenas citada na seção de "Discussões e Novas Perspectivas", onde o autor afirma que a abordagem proposta está sendo refinada, de forma a prover

[3] http://epf.eclipse.org/wikis/openup/

[4] CIRILO, E.; KULESZA, U.; LUCENA, C. A Product Derivation Tool Based on Model-Driven Tecniques and Annotations. Journal of Universal Computer Science, 14, 2008.

"a instalação (*deployment*) de uma versão executável do processo de software dentro de um *workflow engine*". Para isso, sugere uma transformação do tipo "modelo para modelo" da especificação EPF para uma especificação de descrição de *workflows*. Com este objetivo, a ferramenta proposta por ALEIXO *et al*. (2010) foi estendida para gerar especificações de *workflows* a partir de processos EPF, que podem ser executados no jBPM Workflow Engine[5]. Para isso, o autor cita que as transformações foram escritas: (i) na linguagem QVT, para transformação do modelo de processo EPF, gerado pelo GenArch, para um modelo na linguagem de workflow jPDL; e (ii) na tecnologia Acceleo, para permitir a geração de especificações jPDL para formulários JSF que são instalados e executados no jBPM.

Com relação às questões a serem respondidas sobre este trabalho, tem-se:

Trabalho Relacionado 1 - ALEIXO *et al.* (2010)
1. O processo a ser executado é um processo de software?
Sim.
2. Como o processo a ser executado foi modelado (usa algum padrão)?
A modelagem do processo foi realizada no Eclipse EPF.
3. Qual a máquina de execução / ferramenta utilizada para execução?
Para execução dos processos é utilizada o jBPM Workflow Engine.
4. Como é realizado o monitoramento do processo?
Não aborda.
5. Quais informações/métricas de execução são armazenadas e como se dá o acesso a estas informações relativas à execução do processo?
Não aborda.

ALEIXO *et al*. (2012) apresentam um estudo comparativo (qualitativo) entre abordagens de modelagem de linhas de processo de software que utilizam composição e anotação de processos. A abordagem que utiliza composição de processos é representada pelo Eclipse Process Framework EPF Composer (ECLIPSE FOUDATION, 2010) e a abordagem que utiliza anotação é representada pelo GenArch-P (ALEIXO *et al*., 2010). Para análise comparativa das abordagens foram modeladas duas linhas de processos diferentes: uma baseada na família de processos OpenUP e a outra baseada na família processos Scrum. Após a especificação das linhas de processo usando o EPF e o GenArch-P, foi realizado um estudo no qual a abordagem anotativa GenArch-P demonstrou um melhor resultado. Foram levados em consideração os seguintes critérios: (i) rastreabilidade, (ii) detecção de erros, (iii) uniformidade, (iv) quantidade de conceitos, mecanismos e ferramentas necessárias para a adoção da abordagem e (v) gerenciamento sistemático de variabilidade. Para compreender à usabilidade das abordagens para a modelagem de linhas de processos, o autor cita que é fundamental a realização de outros estudos. Também é citado (e proposto como trabalho

[5] http://labs.jboss.com/jbossjbpm/

futuro) que as abordagens de composição e de anotação podem ser combinadas, de forma que os principais pontos fortes de ambas sejam considerados.

Com relação às questões a serem respondidas sobre este trabalho, tem-se:

Trabalho Relacionado 2 - ALEIXO *et al*. (2012)
1. O processo a ser executado é um processo de software?
Sim. Ambos os processos modelados tratavam-se de processos de software.
2. Como o processo a ser executado foi modelado (usa algum padrão)?
A modelagem dos processos foi realizada no Eclipse EPF e no GenArch-P.
3. Qual a máquina de execução / ferramenta utilizada para execução?
Não aborda.
4. Como é realizado o monitoramento do processo?
Não aborda.
5. Quais informações/métricas de execução são armazenadas e como se dá o acesso a estas informações relativas à execução do processo?
Não aborda.

ALEIXO (2013) propõe uma abordagem anotativa para a gerência de variabilidades em processos de software, e também avalia / compara as abordagens composicional e anotativa para a gerência de variabilidades em processos de software. A abordagem anotativa proposta realiza a modelagem e definição de uma LPrS utilizando o EPF (Eclipse Process Framework). Para possibilitar a gerência automatizada de variabilidades dos elementos da linha de processos são especificados os seguintes modelos de variabilidades: modelo de características, modelo de configuração e modelo de processo. As características relevantes para um processo são selecionadas com o auxílio de uma ferramenta de derivação, o GenArch-P, o que possibilita a geração automática de processos de software customizados. O processo customizado também é especificado de forma a permitir sua execução em máquinas de *workflow*.

Segundo o autor, para possibilitar que a especificação de processos pudesse ser executada e que sua execução fosse acompanhada, foi proposta a implantação deste processo em um motor de workflow. Foi aplicada uma transformação de modelo para modelo da especificação EPF para uma especificação de *workflow* correspondente. Com este objetivo, foi implementada uma ferramenta específica para gerar especificações de *workflows* que pudessem ser executados no jBPM Workflow Engine (SALATINO, 2009) a partir de especificações de processos geradas no EPF. As atividades do *workflow* gerado são relacionadas a uma página JSF que contém as informações sobre tal atividade. As páginas JSF permitem que o fluxo do processo seja monitorado, utilizando-se o console Web do jBPM.

Nesta abordagem, o usuário consegue visualizar o estado atual do processo e tem acesso a detalhes a respeito da atividade que precisa ser realizada. Ao realizar cada uma das atividades do processo, o usuário informa o fim desta ao motor de *workflow*. Após isto, o motor de *workflow* solicita ao usuário informações adicionais sobre a atividade finalizada através de um formulário

JSF. Estas informações são armazenadas pelo motor de *workflow* em um banco de dados interno e são relacionadas a uma instância específica do processo, podendo ser recuperadas posteriormente.

Com relação a métricas de processo, ALEIXO (2013) apresenta que, dada a grande variedade de métricas existentes, selecionar quais métricas devem ser aplicadas em um determinado processo de desenvolvimento de software não é uma tarefa trivial. Desta forma, diversas métricas alternativas e complementares podem ser aplicadas ao processo, de acordo com as necessidades de cada projeto. Assim, o autor cita a possibilidade de utilização de "variabilidades nas métricas" em LPrS, possibilitando ao engenheiro de processo derivar um processo contendo métricas alinhadas com as fases, atividades, técnicas e tecnologias que este contém.

Com relação às questões a serem respondidas sobre este trabalho, tem-se:

Trabalho Relacionado 3 - ALEIXO (2013)
1. O processo a ser executado é um processo de software?
Sim.
2. Como o processo a ser executado foi modelado (usa algum padrão)?
A modelagem dos processos foi realizada no Eclipse EPF e no GenArch-P.
3. Qual a máquina de execução / ferramenta utilizada para execução?
Para a execução dos processos é utilizada o jBPM Workflow Engine.
4. Como é realizado o monitoramento do processo?
As informações de execução do processo são armazenadas em um banco de dados interno do próprio jBPM Workflow Engine.
5. Quais informações/métricas de execução são armazenadas e como se dá o acesso a estas informações relativas à execução do processo?
Não aborda. O autor apenas cita que métricas de processos podem ser tratadas como variabilidades da LPrS, já que estas podem variar de acordo com o processo a ser instanciado.

AVRILIONIS *et al*. (1996) tem como foco a construção incremental e a evolução das instâncias do modelo de processo, a fim de apoiar o gerente do projeto durante a execução do processo. É apresentada a abordagem Pynode, que utiliza componentes de processo para definir processos. Esta abordagem permite o desenvolvimento de processos orientado a componentes e também a configuração de um modelo de processo (através da composição dinâmica dos componentes). O comportamento dos componentes é descrito por uma Linguagem de Modelagem de Processos - *Process Modelling Language* (PML) própria e a execução do processo é realizada utilizando um mecanismo que interpreta essa PML.

Na abordagem Pynode, cada componente possui uma interface e um fragmento de modelo de processo. O fragmento de modelo de processo descreve o comportamento do componente usando uma PML específica. Este implementa as funcionalidades, propriedades e serviços, especificados no nível da interface.

Neste trabalho é introduzido o conceito de *visões de execução* (*execution views*), que são compostas por um conjunto de componentes reutilizáveis, que permite o uso e a combinação destes

componentes de forma interativa. Cada visão de execução define como seus componentes são usados e como estes estão conectados a outras visões de execução. A visão de execução é usada para descrever as partes mais dinâmicas do processo, cuja estrutura não é completamente definida no início da execução. Assim, a abordagem Pynode proporciona uma composição dinâmica de componentes em tempo de execução, devido a este mecanismo de visões de execução. Para o monitoramento da execução do processo, o autor sugere o uso de visões de observação (em desenvolvimento). Segundo o autor, uma visão de observação seria uma visão que forneceria suporte, em tempo de execução, para a medição e composição do processo, capturando o conhecimento e as informações necessárias para atingir os objetivos do processo. Embora citado, este mecanismo de visões de observações não foi detalhadamente explicitado, por se tratar de uma etapa em desenvolvimento.

Com relação às questões a serem respondidas sobre este trabalho, tem-se:

Trabalho Relacionado 4 - AVRILIONIS et al. (1996)
1. O processo a ser executado é um processo de software?
Sim.
2. Como o processo a ser executado foi modelado (usa algum padrão)?
O processo é modelado utilizando uma PML específica, porém, esta não é descrita no artigo.
3. Qual a máquina de execução / ferramenta utilizada para execução?
Utilizam uma ferramenta própria. O autor cita que a abordagem Pynode foi implementada como uma extensão do Opsis System, um ambiente que permite a execução e evolução de modelos de processos.
4. Como é realizado o monitoramento do processo?
Não aborda claramente. Para o monitoramento do processo o autor apenas sugere o uso de visões de observação (em desenvolvimento, de acordo com o trabalho).
5. Quais informações/métricas de execução são armazenadas e como se dá o acesso a estas informações relativas à execução do processo?
Não aborda.

BARRETO (2011) apresenta uma abordagem para definição de processos baseada em reutilização, que considera o contexto da alta maturidade e utiliza componentes de processos. Nesta abordagem, é utilizada uma arquitetura de processos que possui variabilidades como uma Linha de Processos de Software. Esta arquitetura de processos pode ser composta por componentes de processo, por atividades ou pela combinação destes. É utilizado um modelo de componentes para a modelagem de variabilidades da LPrS. Neste modelo podem ser adotados dois tipos de componentes: componentes concretos, que não permitem configurações, e componentes abstratos, com definições incompletas que permitem configurações através da implementação por componentes concretos. Neste modelo de componentes de processos, também podem ser definidos componentes mandatórios e componentes opcionais, além da criação de pontos de variação e variantes.

Com relação ao monitoramento do processo, BARRETO (2011) apresenta um componente de processo como uma entidade mensurável (algo que pode ser medido) e, portanto, herda relacionamentos que dizem respeito a medidas, ou seja, um componente de processo pode estar associado a medidas que o quantificam. Cada tipo de entidade mensurável possui um conjunto de elementos mensuráveis que o caracteriza, sendo assim, é possível estabelecer, por exemplo, que componentes de processo são medidos em relação a custo, esforço, entre outros.

BARRETO (2011) utiliza e adapta uma Ontologia de Medição de Software definida por BARCELLOS (2009). Esta ontologia foi dividida em sete subontologias. Uma destas subontologias é a de Comportamento de Processos, que trata da aplicação dos resultados da medição na análise do comportamento de processos. BARRETO (2011) destaca esta subontologia devido ao fato desta incluir diversos conceitos relacionados à alta maturidade de processos, tais como *baselines* de desempenho de processo e capacidade de processo, definidos a seguir.

"*Baselines* de desempenho são definidas em relação a um componente de processo e a uma medida (com a indicação de um limite inferior e superior), em um dado contexto. Quando um componente de processo possuir uma *baseline* de desempenho associada a ele, esse componente será estável para a medida e contexto relacionados" BARRETO (2011). As *baselines* de desempenho de processo são originadas a partir de medidas coletadas durante diversas execuções de um componente de processo, ou seja, deve-se disponibilizar um meio para que as *baselines* de desempenho possam ser registradas através das ferramentas de apoio, de modo a permitir a utilização dessa informação, posteriormente, para a definição de novos processos da linha. Deve-se ressaltar também que o contexto em que uma *baseline* é estabelecida também é importante, uma vez que é necessário saber se esta aplica-se ou não a uma situação específica.

ISO/IEC, 2003

Capacidade de processo é definido na norma ISO/IEC 15504 (2003) como uma caracterização da habilidade de um processo atender a metas de negócio, sejam elas projetadas ou atuais ou projetadas. A capacidade do processo pode ser definida em 6 níveis:

- Nível 0 - Processo incompleto;
- Nível 1 - Processo executado;
- Nível 2 - Processo gerenciado;
- Nível 3 - Processo estabelecido;
- Nível 4 - Processo previsível;
- Nível 5 - Processo em otimização.

Como a abordagem proposta por BARRETO (2011) não oferece nenhum apoio à execução do processo, foi desenvolvida uma versão simplificada da base de medidas e das *baselines* de

desempenho. Estas foram registradas diretamente nas ferramentas de apoio para permitir a utilização dessa informação na definição dos processos.

Com relação às questões a serem respondidas sobre este trabalho, tem-se:

Trabalho Relacionado 5 - BARRETO (2011)
1. O processo a ser executado é um processo de software?
Sim.
2. Como o processo a ser executado foi modelado (usa algum padrão)?
Utiliza um modelo de componentes próprio.
3. Qual a máquina de execução / ferramenta utilizada para execução?
Não aborda.
4. Como é realizado o monitoramento do processo?
Não aborda.
5. Quais informações/métricas de execução são armazenadas e como se dá o acesso a estas informações relativas à execução do processo?
A abordagem utiliza uma base de medidas pré-cadastradas, relacionadas aos componentes, apesar de não oferecer suporte à execução e coleta destas medidas na própria abordagem.

COULETTE *et al*. (2000) apresentam um ambiente chamado RHODES, para engenharia de software, centrado em processos, que permite aos usuários descrever e executar processos. Neste artigo é apresentada uma extensão do RHODES, que suporta ‘componentes complexos’. O autor divide os componentes de processos em ementares e complexos. Componentes elementares são entidades de baixo nível que representam uma unidade de decomposição de processos. Eles são semelhantes às classes em um diagrama de classe UML. Porém, o autor ressalta que componentes elementares não são suficientes para representar processos complexos e, de forma a possibilitar o reúso destes componentes, é preciso de um maior nível de detalhamento para representação dos componentes do processo, que é o que ele chama de componente complexo. Componentes complexos são semelhantes aos pacotes em UML ou às atividades (conjunto de tarefas) em uma notação para modelagem de processos. Eles consistem em um conjunto de componentes elementares e cada componente complexo possui duas partes: uma interface e uma implementação.

O ambiente RHODES é classificado como um PSEE (*Process Centered Software Engineering Environment*). PSSEs foram elaborados para descrever, armazenar e executar processos. Para a modelagem dos componentes de processo no RHODES é utilizada a linguagem *PBOOL+ Process Modeling Language*, uma extensão da linguagem PBOOL (CRÉGUT e COULETTE, 1997) uma linguagem para definição e reutilização de processos executáveis orientada a objetos.

A arquitetura do RHODEs é composta por 5 partes principais: (1) metamodelo, (2) base de componentes; (3) *kernel* de execução; (4) ferramenta para modelagem dos processos; e (5) instâncias do RHODES para os processos modelados.

Nesta abordagem, as tarefas a serem executadas são especificadas em uma árvore e a execução do processo só termina quando todas as tarefas desta árvore são finalizadas, ou seja, quando a tarefa raiz desta árvore é finalizada.

Com relação às questões a serem respondidas sobre este trabalho, tem-se:

Trabalho Relacionado 6 - COULETTE *et al.* (2000)
1. O processo a ser executado é um processo de software?
Sim.
2. Como o processo a ser executado foi modelado (usa algum padrão)?
Utiliza a linguagem *PBOOL+ Process Modeling Language*.
3. Qual a máquina de execução / ferramenta utilizada para execução?
A própria ferramenta RHODEs oferece um *kernel* de execução.
4. Como é realizado o monitoramento do processo?
Não aborda.
5. Quais informações/métricas de execução são armazenadas e como se dá o acesso a estas informações relativas à execução do processo?
Não aborda.

DAI e LI (2007a)(2007b) e DAI *et al.* (2008) definem uma Arquitetura de Processos composta por componentes de processo de evolução e conectores. Componentes de processo de evolução são definidos como fragmentos coesos e consistentes do processo de evolução de software. Esse componente é considerado uma "caixa-preta" sem comunicação direta com outros componentes. Esta comunicação é realizada através de conectores. Conectores são unidades de comunicação entre estes componentes e são representados por condições em Redes de Petri, que são usados para descrever as relações de interação entre os componentes. Para a descrição do processo de evolução e de seus componentes é utilizada a linguagem proposta pelos próprios autores, a EPDCL (*Evolution Process Component Description Language*) e o metamodelo criado para a representação do processo foi baseado em uma extensão de Redes de Petri.

Esta abordagem é dividida em cinco passos: (1) Elicitação e Análise de Requisitos de Processos; (2) Construção da Arquitetura de Processos; (3) Construção dos Componentes de Processos que compõem a Arquitetura; (4) Composição de um Processo Específico e (5) Simulação do Processo. No passo 5, como o processo foi modelado seguindo as regras de execução de Redes de Petri, o processo de execução do software pode ser executado por uma máquina de processo. Durante a execução, os dados ou conhecimentos relevantes para o processo podem ser coletados. O autor não especifica qual a máquina de execução usada ou como estes dados são coletados. No texto é ressaltado ainda que, neste trabalho, a solução possui como foco as fases da arquitetura de composição do processo.

Com relação às questões a serem respondidas sobre este trabalho, tem-se:

Trabalho Relacionado 7 - DAI e LI (2007a)(2007b) e DAI *et al*. (2008)
1. O processo a ser executado é um processo de software?
Sim
2. Como o processo a ser executado foi modelado (usa algum padrão)?
EPDCL (Evolution Process Component Description Language).
3. Qual a máquina de execução / ferramenta utilizada para execução?
Não foi mencionada.
4. Como é realizado o monitoramento do processo?
Não aborda.
5. Quais informações/métricas de execução são armazenadas e como se dá o acesso a estas informações relativas à execução do processo?
Não aborda.

GARY e LINDQUIST (1999a) (1999b) apresentam um *framework* baseado em componentes para a modelagem de processos de software chamado OPC (*Open Process Framework*). Nesta abordagem as informações para a composição de processos são divididas em três partes: esquema de processo (possíveis entidades e relações); estados de processo (transições do comportamento dinâmico do componente: executando, suspenso e abortado) e implementação de processo (representação executável de um componente). Esta representação executável pode ser uma rede de Petri, uma linguagem de programação de processos ou uma ferramenta de processo encapsulada para fornecer a implementação do processo. Esta implementação de um componente é incluída no próprio componente.

Um ambiente para apoiar a abordagem OPC foi implementado em Java. Este ambiente inclui mecanismos para a criação, modificação, extensão, implantação, monitoramento, reutilização e interoperabilidade de componentes.

Com relação às questões a serem respondidas sobre este trabalho, tem-se:

Trabalho Relacionado 8 - GARY e LINDQUIST (1999a)(1999b)
1. O processo a ser executado é um processo de software?
Sim.
2. Como o processo a ser executado foi modelado (usa algum padrão)?
Foi criado um ambiente em Java para apoiar a criação, modificação, extensão, implantação, monitoramento, reutilização e interoperabilidade dos componentes de acordo com a abordagem OPC.
3. Qual a máquina de execução / ferramenta utilizada para execução?
É utilizado o mesmo ambiente citado na questão anterior.
4. Como é realizado o monitoramento do processo?
Não aborda (apesar de citar que a ferramenta desenvolvida realiza o monitoramento dos componentes).
5. Quais informações/métricas de execução são armazenadas e como se dá o acesso a estas informações relativas à execução do processo?
Não aborda.

HALLERBACH *et al*. (2008) apresentam uma abordagem para o gerenciamento de variantes do ciclo de vida do processo, chamada Provop (*PROcess Variants by OPtions*). As variantes do processo podem ser configuradas a partir de um processo de base, ou seja, uma variante específica é derivada a partir do processo de base, através da aplicação de um conjunto de operações de mudança bem definidas para ele. Assim, são definidas operações para adaptação de um modelo de processo base e a especificação de situações de contexto para aplicação de tais operações.

A abordagem Provop fornece suporte para todo o do ciclo de vida do processo e permite a configuração de processos flexíveis, resultando em uma coleção manutenível de variantes do processo. São abordadas 4 fases do ciclo de vida de processos: Modelagem, Instanciação, Execução e Otimização. A fase de execução é realizada logo após a seleção e instanciação do modelo de variantes. Este modelo precisa ser traduzido em um modelo de workflow executável, por exemplo, especificado com WS-BPEL. É citado pelo autor a presença de variáveis de contexto dinâmicas, que exigem a capacidade de alternar entre as variantes durante a execução, porém, tanto esta questão quanto os detalhes relacionados à execução do processo não são apresentados neste trabalho.

A Provop também permite evoluir e aperfeiçoar o processo básico. As opções modeladas são verificadas contra o novo modelo de processo básico. Se uma opção é afetada por mudanças do processo básico, por exemplo, porque um ponto de ajuste foi movido para uma nova posição, esta opção será atualizada no processo base.

Com relação às questões a serem respondidas sobre este trabalho, tem-se:

Trabalho Relacionado 9 - HALLERBACH *et al*. (2008)
1. O processo a ser executado é um processo de software?
Não.
2. Como o processo a ser executado foi modelado (usa algum padrão)?
Notação própria da abordagem PROVOP
3. Qual a máquina de execução / ferramenta utilizada para execução?
Não especificado. Cita apenas que o processo a ser executo é descrito utilizando WS-BPEL
4. Como é realizado o monitoramento do processo?
Não especificado.
5. Quais informações/métricas de execução são armazenadas e como se dá o acesso a estas informações relativas à execução do processo?
Não aborda.

Em JAUFMAN e MÜNCH (2005) é apresentada uma abordagem baseada em LPr para adaptação de processos para projetos específicos chamada EPAc (*Emergent Process Acquisition method* - Método de Aquisição de Processos Emergentes). Um processo emergente é um processo que precisa lidar com a mudança de objetivos e características do contexto, sendo que estes só

podem ser antecipados em um momento muito próximo do início do projeto. Portanto, o processo em si precisa ser altamente adaptável e o apoio para esta adaptação é necessária.

Esta abordagem é dividida em duas partes: (1) definição/utilização de uma linha de processos específica de domínio para fornecer conhecimento sobre o domínio; (2) o processo instanciado é executado e os dados coletados de sua execução são usados para revisar o processo definido. O método foca na fase de engenharia de aplicação, ou seja, na derivação de um processo específico de projeto, de acordo com seus objetivos e suas características de contexto.

Após a execução do processo definido, é realizada uma etapa com o objetivo de entender o processo executado por membros da equipe do projeto, visando identificar o 'delta' entre o que foi prescrito e o que foi realizado. Nesta etapa, é utilizada a ferramenta InStep (MICROTOOL, 2014) para apoiar a coordenação entre os membros da equipe do projeto, registrando o status das atividades realizadas e o momento em que estas atividades são realizadas. Como saída, esta ferramenta fornece um arquivo de *log* de texto. Após isto, uma ferramenta dos próprios autores faz a conversão deste arquivo para um arquivo XML. Este arquivo XML é usado como entrada para as ferramentas ProM (DUSTDAR *et al*., 2004) e InterPol. A ferramenta ProM gera um modelo do processo realizado. A ferramenta Interpol realiza a análise do delta.

Com base nos objetivos do projeto e nos planos de projetos similares, o gerente de projeto define *milestones*. Estes incluem: (1) os artefatos a serem entregues, (2) o tempo para entrega dos artefatos do projeto, (3) a maturidade do artefato entregue, (4) os responsáveis pela entrega do artefato.

Para a adaptação do processo 'em voo' (em tempo de execução), foi utilizado o algoritmo apresentado em REICHERT (2000).

Com relação às questões a serem respondidas sobre este trabalho, tem-se:

Trabalho Relacionado 10 - JAUFMAN e MÜNCH (2005)
1. O processo a ser executado é um processo de software?
Não. É descrito um exemplo de processo no domínio automotivo.
2. Como o processo a ser executado foi modelado (usa algum padrão)?
Não especificado.
3. Qual a máquina de execução / ferramenta utilizada para execução?
Não especificado.
4. Como é realizado o monitoramento do processo?
São utilizadas ferramentas como InStep ProM e InterPol com o objetivo de entender o processo executado por membros da equipe do projeto, visando identificar o 'delta' entre o que foi prescrito e o que foi realizado.
5. Quais informações/métricas de execução são armazenadas e como se dá o acesso a estas informações relativas à execução do processo?
São estabelecidos *milestones* durante a modelagem do processo, como (1) os artefatos a serem entregues, (2) o tempo para entrega dos artefatos do projeto, (3) a maturidade do artefato entregue, (4) os responsáveis pela entrega artefato. Não é tratado no texto como estas informações são armazenadas / acessadas.

MARTÍNEZ-RUIZ *et al*. (2013) apresentam um estudo de caso sobre a utilização do paradigma de Processos Rico em Variantes (VRP – *Variant-Rich Process*) para apoiar a adaptação de processos em Desenvolvimento de Software Global (GSD – *Global Software Development*). É apresentada a abordagem SPRINTT (*Software Process Institutionalization based on Tailoring and sTandardization*), que inclui um ciclo composto por quatro etapas (adaptação, execução, análise e padronização) para transformação e inclusão de variantes de processos. Para a modelagem dos processos e suas variações é utilizada a linguagem vSPEM e, para as atividades de definição e adaptação de processos VRP, é utilizado um *plugin* para o Eclipse EFP chamado vEPF. A versão do 1.0 do vEPF oferece como funcionalidades a execução de engenharia reversa sobre projetos EPF e fornece mecanismos para criar processos ricos em variantes, além de possibilitar a adaptação destes processos. Como trabalho futuro, é proposta a conclusão da segunda versão da ferramenta vEPF, que irá integrar mecanismos de variabilidade transversais e apoiará completamente a adaptação e execução de processos em Desenvolvimento de Software Global. Assim, não foi apresentado, de fato, como se dá a execução e monitoramento do processo modelado, bem como as ferramentas usadas para isto.

Com relação às questões a serem respondidas sobre este trabalho, tem-se:

Trabalho Relacionado 11 - MARTÍNEZ-RUIZ *et al*. (2013)
1. O processo a ser executado é um processo de software?
Sim.
2. Como o processo a ser executado foi modelado (usa algum padrão)?
Utilizando a linguagem vSPEM
3. Qual a máquina de execução / ferramenta utilizada para execução?
Não especificado.
4. Como é realizado o monitoramento do processo?
Não especificado.
5. Quais informações/métricas de execução são armazenadas e como se dá o acesso a estas informações relativas à execução do processo?
Não especificado.

SIMMONDS *et al.* (2012) propõem uma combinação de notações e ferramentas para a formalização de modelos de processos de software em LPrS. De acordo com esta abordagem, uma LPrS possui duas partes principais: (1) um modelo geral de processo e (2) a especificação de quais elementos de processos variam e como podem variar.

É utilizado o Eclipse Process Framework Composer para especificação do processo geral, que utiliza a notação SPEM 2.0. A ferramenta SPLOT foi utilizada para especificar a variabilidade do processo através de um modelo de características. A ferramenta Modisco / AMW foi usada para lidar com as questões de inconsistências entre os dois modelos e estabelecer restrições entre estes. Um estudo de caso na indústria é apresentado com o objetivo de mostrar como essas ferramentas

são usadas para especificar e analisar os modelos de processo de software que incluem variabilidade.

Com relação às questões a serem respondidas sobre este trabalho, tem-se:

Trabalho Relacionado 12 - SIMMONDS *et al.* (2012)
1. O processo a ser executado é um processo de software?
Sim.
2. Como o processo a ser executado foi modelado (usa algum padrão)?
Utiliza a notação SPEM 2.0, através da ferramenta Eclipse Process Framework Composer.
3. Qual a máquina de execução / ferramenta utilizada para execução?
Não aborda.
4. Como é realizado o monitoramento do processo?
Não aborda.
5. Quais informações/métricas de execução são armazenadas e como se dá o acesso a estas informações relativas à execução do processo?
Não aborda.

TORTORELLA e VISAGGIO (1997) apresentam a abordagem CREP (*Characterizing Reverse Engineering Process*), um método para formalizar, caracterizar e avaliar a engenharia reversa de componentes de processo. Esta abordagem é dividida em quatro passos: (1) Construção de um Modelo de Processo; (2) Validação do Modelo de Processo; (3) Modificação do Modelo de Processo; e (4) Caracterização do Componente de Processo. É apresentado um ambiente de apoio à abordagem denominado PROMETHEUS *(PROcess Model Evolution Through Experience Unfolded Systematically*). O formalismo para a modelagem de processos utilizado é Modelador Conceitual de Processo (Process Conceptual Modeler– PCM), uma ferramenta do próprio PROMETHEUS.

Com relação às questões a serem respondidas sobre este trabalho, tem-se:

Trabalho Relacionado 13 - TORTORELLA e VISAGGIO (1997)
1. O processo a ser executado é um processo de software?
Não.
2. Como o processo a ser executado foi modelado (usa algum padrão)?
Utiliza uma ferramenta própria *Process Conceptual Modeler* – PCM.
3. Qual a máquina de execução / ferramenta utilizada para execução?
Não aborda.
4. Como é realizado o monitoramento do processo?
Não aborda.
5. Quais informações/métricas de execução são armazenadas e como se dá o acesso a estas informações relativas à execução do processo?
Não aborda.

De acordo com as informações extraídas dos trabalhos anteriormente detalhados, obteve-se a Tabela 4.2. Todas as abordagens analisadas foram classificadas de acordo com os seguintes requisitos, elaborados a partir das cinco questões de pesquisa que foram analisadas em cada um dos 13 trabalhos selecionados no primeiro filtro:

R1. A abordagem trata de processo de software.

R2. A modelagem do processo utiliza um padrão ou linguagem específica.

R3. A abordagem apresenta o uso de uma máquina de execução de processos.

R4. A abordagem leva em consideração a fase de monitoramento do processo.

R5. A abordagem define e armazena métricas de execução de processos.

Tabela 4.2: Análise das abordagens com relação aos requisitos propostos

√ = atende, ! = atende parcialmente e × = não atende

		R1	R2	R3	R4	R5
1	ALEIXO *et al*. (2010)	√	√	√	×	×
2	ALEIXO *et al*. (2012)	√	√	×	×	×
3	ALEIXO (2013)	√	√	√	√	×
4	AVRILIONIS *et al*. (1996)	√	√	√	!	×
5	BARRETO (2011)	√	√	×	×	√
6	COULETTE *et al*. (2000)	√	√	√	×	×
7	DAI e LI (2007a) (2007b) e DAI et al. (2008)	√	√	×	×	×
8	GARY e LINDQUIST (1999a) (1999b)	√	√	√	×	×
9	HALLERBACH *et al*. (2008)	×	√	×	×	×
10	JAUFMAN e MÜNCH (2005)	×	×	×	√	!
11	MARTÍNES-RUIZ *et al*. (2013)	√	√	×	×	×
12	SIMMONDS *et al*. (2012)	√	√	×	×	×
13	TORTORELLA e VISAGGIO (1997)	×	√	×	×	×

A Tabela 4.3 apresenta a quantidades de trabalhos obtidos de acordo com os requisitos analisados em todos estes trabalhos.

Tabela 4.3: Quantidades de trabalhos obtidos de acordo com os requisitos de seleção

Medida	**Valor**
Quantidade de trabalhos selecionados	13
Quantidade de abordagens que tratam de processos de software	10
Quantidade de abordagens que utilizam um padrão para a modelagem do processo	12
Quantidade de abordagem que utilizam alguma ferramenta para execução de processos	5
Quantidade de abordagens que abordam sobre o monitoramento do processo	2
Quantidade de abordagens que abordam sobre métricas de execução de processos	1

Baseado na análise dos trabalhos selecionados inicialmente percebe-se que, no contexto de processos de software, o número de abordagens que são capazes de apoiar todo o ciclo de vida de processos é limitado. A maioria das abordagens tem como foco a definição e modelagem dos processos software, enquanto as fases de execução, monitoramento e melhoria, apesar de serem citadas, ainda não vem sendo tratadas por estas abordagens.

4.2 Busca Estruturada

Apenas dois dos trabalhos analisados na seção anterior apresentaram a utilização de alguma ferramenta de execução de processos de software e tratam sobre a avaliação/monitoramento desta execução, sendo eles: ALEIXO (2013) e AVRILIONIS *et al*. (1996). Assim, estes dois trabalhos foram selecionados como os potenciais trabalhos de controle que deveriam ser citados em uma revisão da literatura sobre execução e monitoramento de processos de software.

Como a revisão bibliográfica inicial apresentada na seção anterior foi realizada a partir de uma revisão sistemática sobre trabalhos propostos na área de reutilização de processos, existe a possibilidade de não terem sido analisados trabalhos existentes que tem como foco as fases de execução e monitoramento de processos e que NÃO estão relacionados à reutilização de processos. Portanto, para identificar outros trabalhos sobre execução e monitoramento de processos de software e, assim, agregar maior evidência sobre estas etapas do ciclo de vida de processos, uma *Busca Estruturada* (RIBEIRO, 2014), que consistente em uma estratégia de revisão da literatura foi realizada. A busca estruturada foi a estratégia utilizada neste trabalho, tendo em vista que algumas restrições de tempo para execução deste estudo limitam o rigor no planejamento da revisão, considerando que se tem como foco apresentação a aplicação deste trabalho ao contexto de dois projetos da indústria.

4.2.1 Planejamento da Busca Estruturada

4.2.1.1 Objetivo da Busca Estruturada

Seguindo a abordagem GQM (*Goal, Question, Metric*), a busca estruturada desenvolvida neste trabalho possui como objetivo (conforme já mencionado na Seção 4.1), a análise de publicações que tratam sobre a execução e o monitoramento de processos de software, com o propósito de caracterizar abordagens (como ferramentas, métodos, processos e técnicas), com relação à execução/monitoramento de processos, do ponto de vista de pesquisadores, no contexto acadêmico e da indústria.

4.2.1.2 Questões de Pesquisa

Como forma de identificar as abordagens que tratam sobre execução e monitoramento de processos de software, foram definidas as seguintes questões a serem respondidas com a busca estruturada:

Questão Principal:

- Quais soluções ou ferramentas têm sido utilizadas para execução, monitoramento e avaliação de processos de software?

Questão Secundária:

- Quais soluções apresentadas abordam a realimentação do processo com base em dados de execução do mesmo?

4.2.1.3 Estratégia de Busca e Artigos de Controle

Nesta busca estruturada utilizou-se a estrutura PICO (*Population, Intervention, Comparison, Outcome Measure* – População, Intervenção, Comparação e Resultados) para organizar e estruturar a busca a ser realizada:

(P) População (*Population*): Processos de software.

(I) Intervenção (*Intevention*): Pesquisas relacionadas à execução e monitoramento de processos.

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

(O) Resultados (*Output*): Técnicas, abordagens, métodos, metodologias, ferramentas para execução de processos de software.

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

(P) População: *software process, software development*

(I) Intervenção: *process execution, process enactment, process evaluation, process monitoring, process improvement*

(C) Comparação: não aplicável

(O) Resultados: *technique, approach, method, methodology, tool, procedure, mechanism, research, study*

Através da junção dos termos apresentados anteriormente, foi definida uma *string* de busca inicial, a qual foi alterada e refinada gradativamente. Este refinamento teve como objetivo adequar o número de artigos retornados e atender o objetivo desta revisão. Após os devidos refinamentos, a *string* adotada foi:

(("*process execution*" OR "*process enactment*") AND ("*process evaluation*" OR "*process monitoring*" OR "*process improvement*")) AND ("*software process*" OR "*software development*") AND ("*technique*" OR "*approach*" OR "*method*" OR "*methodology*" OR "*tool*" OR "*procedure*" OR "*mechanism*" OR "*research*" OR "*study*")

Esta *string* foi adaptada para cada uma das máquinas de busca utilizadas, de acordo com sua respectiva sintaxe. As bases de pesquisa utilizadas para esta revisão foram definidas de acordo com os seguintes requisitos:

- Ser capaz de usar expressões lógicas ou um mecanismo similar;
- Permitir pesquisas em todo o documento indexado ou em campos específicos deste;
- Estar disponível na instituição do pesquisador.

Assim, com base nos critérios anteriores, foram selecionadas as seguintes bases de pesquisa: Scopus, IEEExplore e Compendex, cujas respectivas *strings* de buscas utilizadas encontram-se a seguir:

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

String de Busca: TITLE-ABS-KEY ((("*process execution*" OR "*process enactment*") AND ("*process evaluation*" OR "*process monitoring*" OR "*process improvement*")) AND ("*software process*" OR "*software development*") AND ("*technique*" OR "*approach*" OR "*method*" OR "*methodology*" OR "*tool*" OR "*procedure*" OR "*mechanism*" OR "*research*" OR "*study*"))

- **Base de Busca: IEEExplore** (www.ieeexplore.ieee.org)

String de Busca: ((("*process execution*" OR "*process enactment*") AND ("*process evaluation*" OR "*process monitoring*" OR "*process improvement*")) AND ("*software process*" OR "*software development*") AND ("*technique*" OR "*approach*" OR "*method*" OR "*methodology*" OR "*tool*" OR "*procedure*" OR "*mechanism*" OR "*research*" OR "*study*"))

- **Base de Busca: Compendex** (www.engineeringvillage.com)

String de Busca: ((("process execution" OR "process enactment") AND ("process evaluation" OR "process monitoring" OR "process improvement")) AND ("software process" OR "software development") AND ("technique" OR "approach" OR "method" OR "methodology" OR "tool" OR "procedure" OR "mechanism" OR "research" OR "study"))

Também foi considerado como critério para definição da *string* de busca a exibição dos artigos de controle pré-definidos, listados a seguir, desde que estes estivessem previamente indexados pela máquina utilizada. Deve-se ressaltar que o primeiro artigo definido como controle

foi definido na revisão bibliográfica inicial, apresentada na Seção 4.1. O trabalho de ALEIXO (2013), embora sugerido como um potencial trabalho de controle não se encontra na lista a seguir por tratar-se de uma tese de doutorado e, por conseguinte, não estar indexado nas bases de pesquisa utilizadas.

Com o objetivo de encontrar outros artigos de controle para compor a busca estruturada, uma busca *ad hoc* foi realizada visando obter outros trabalhos que se enquadrassem no objetivo proposto pela busca estrutura e que pudessem ser utilizados como controle. Assim, foram usados como controle os trabalhos apresentados a seguir.

- **Artigos de Controle**

AVRILIONIS, D., BELKHATIT, N., CUNIN, P.Y., 1996, "A Unified Framework for Software Process Enactment and Improvement". In: *Proceedings of the 4th International Conference on Software Process* (ICSP4), Brighton, UK.

SOYLEMEZ, M., TARHAN, A, 2013, "Using Process Enactment Data Analysis to Support Orthogonal Defect Classification for Software Process Improvement". In Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2013 Joint Conference of the 23rd International Workshop on, IEEE, pp. 120-125.

HUO, M., ZHANG, H., JEFFERY, R., 2006, "A Systematic Approach to Process Enactment Analysis as Input to Software Process Improvement or Tailoring," Software Engineering Conference, APSEC 2006, 13th Asia Pacific , pp.401,410, 6-8 Dec. 2006.

- **Critérios de Inclusão dos Artigos:**

(I1) Trabalhos que abordam sobre execução de processos de software, apresentando como esta deve ser realizada e/ou apresentam ferramental de apoio para esta etapa do ciclo de vida de processos; OU

(I2) Trabalhos que abordam sobre monitoramento da execução de processos; OU

(I3) Trabalhos que abordam sobre avaliação de processos de software com base em dados de execução e/ou métricas previamente definidas.

- **Critérios de Exclusão de Artigos:**

(E1) Trabalhos fora da área de computação; OU

(E2) Trabalhos cujo foco de pesquisa encontra-se em outras áreas da computação que não seja a de Engenharia de Software; OU

(E3) Propostas que não sejam aplicadas a processos de software, especificamente; OU

(E4) Trabalhos indisponíveis; OU

(E6) Trabalhos não escritos em inglês.

Após a execução da *string* de busca nas bases de pesquisa selecionadas, cada artigo retornado (eliminando-se as duplicatas) foi avaliado de acordo com os critérios de inclusão e exclusão de artigos, após a leitura de seus resumos. Artigos que porventura, apenas com a leitura de seus resumos, pudessem levar à dúvida se deveriam ou não ser incluídos no estudos, foram selecionados para leitura completa. A avaliação da inclusão ou não dos artigos retornados pela busca estruturada foi realizada apenas pela autora desta proposta de tese.

4.2.1.4 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 às questões de pesquisa estabelecidas para esta busca estruturada. Estes são listados a seguir:

(C1) Descrição de como é a realizada a execução do processo, quais os procedimentos, ferramentas utilizadas e dados armazenados;

(C2) Descrição de como é realizado o monitoramento do processo, quais os procedimentos, ferramentas utilizadas e dados monitorados;

(C3) Descrição de como é a realizada a avaliação do processo, quais os procedimentos, ferramentas utilizadas e dados avaliados;

(C4) Descrição de como é a realizada a realimentação do processo, quais os procedimentos, ferramentas e dados utilizados.

4.2.2 Execução da Busca Estruturada

A presente busca estruturada foi realizada no decorrer do mês de dezembro de 2014, tendo sido revisada em 12 de janeiro de 2015, com o objetivo de verificar se alguma nova publicação seria retornada pelas máquinas de busca ao executar a mesma *string* de busca, o que não ocorreu. O total de artigos retornado por cada uma das máquinas de busca utilizadas pode ser visualizado na Figura 4.1.

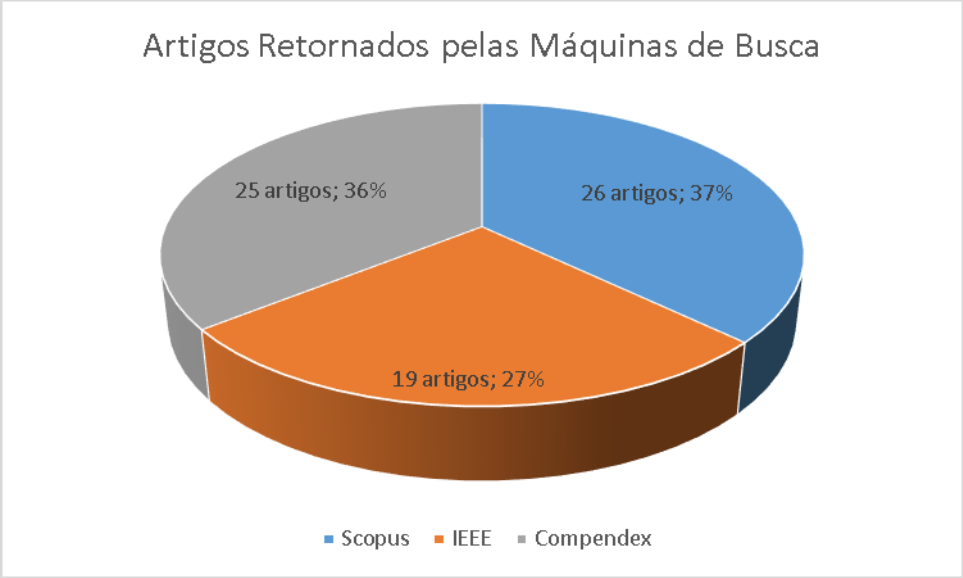


Figura 4.1: Total de Artigos retornados pelas máquinas de busca

Eliminando-se as duplicatas existentes entre os 70 trabalhos retornados, foram obtidos 40 artigos. Após a leitura do título e *abstract* destes 40 artigos e de acordo com os critérios de inclusão e exclusão, foram eliminados 14 artigos e incluídos 25, além da inclusão de um artigo que gerou dúvida sobre sua inclusão/exclusão, pois não ficou claro apenas pela leitura de seu resumo se ele realmente atendia ao primeiro critério de inclusão. Estes resultados podem ser visualizados na Tabela 3.4.

Tabela 4.4: Totais de Artigos da Busca Estruturada e após leitura do título e resumo

Total de Artigos	70
Total de Artigos s/ as Duplicatas	40
Total de Artigos Excluídos	14
Total de Artigos Incluídos	25
Total de Artigos em Dúvida	1

Posteriormente, com a leitura completa dos 25 artigos incluídos e do artigo em dúvida, identificamos que boa parte dos artigos até mencionavam a questão da melhoria de processos de software, porém, esta melhoria não era realizada a partir (ou com base em) dados de execução previamente coletados de execuções anteriores do processo em questão. Portanto, conforme estabelecido no critério de inclusão (I3), foram incluídos nesta fase apenas trabalhos que

abordassem sobre avaliação de processos de software com base em dados de execução e/ou simulação de processos. O resultado final, após a leitura dos artigos é apresentado na Tabela 3.5.

Tabela 4.5: Totais de Artigos após leitura completa

Total de Artigos Excluídos	10
Total de Artigos Duplicados	1
Total de Artigos Incluídos	16

Entre os excluídos, havia uma duplicata com relação ao artigo de controle proposto por HUO *et al*. (2006). Embora com títulos diferentes, tratava-se da mesma proposta/trabalho. Foi mantido o artigo de controle. A seguir é exibida a referência da duplicata que foi excluída:

- HUO, M.; ZHANG, H.; JEFFERY, R, 2006 **An exploratory study of process enactment as input to software process improvement**. In: 2006 4TH INTERNATIONAL WORKSHOP ON SOFTWARE QUALITY, WOSQ 2006, CO-LOCATED WITH THE 28TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE 2006.

Todos os dados extraídos dos 16 artigos incluídos nesta busca estruturada podem ser visualizados no APÊNDICE B – Informações Extraídas dos Artigos Incluídos na Busca Estruturada.

4.2.3 Agregação dos Resultados Obtidos

Com base na análise dos dados extraídos dos trabalhos selecionados, foram elaboradas as respostas as questões de pesquisa definidas durante o planejamento desta busca estruturada, conforme apresentado a seguir:

Questão primária

- Quais soluções ou ferramentas têm sido utilizadas para execução, monitoramento e avaliação de processos de software?

  RESPOSTA: Não há um consenso sobre ferramentas utilizadas nas abordagens para execução, monitoramento ou avaliação de processos de software. Entre as abordagens analisadas, ou é sugerida a utilização de uma ferramenta genérica para gerenciamento de processos, tais como BPMS (*Business Process Management Suite* ou *System*) e PSEEs (*Process centeres Software Engineering Environment*) (KEDJI *et al*., 2011) ou são adotadas soluções proprietárias que atendem às necessidades específicas da abordagem apresentada, tais como Opsis System (AVRILIONIS *et al*., 1996),

GENESIS (AVERSANO *et al*., 2004), PSEE for MDA software processes (MACIEL *et al*., 2009).

Questão Secundária:

- Alguma das soluções apresentadas aborda a realimentação de processos com base em dados de execução do mesmo?

  RESPOSTA: Dos trabalhos analisados, apenas a plataforma GENESIS (AVERSANO *et al*., 2004), permite, de fato, a evolução dos processos, usando uma política de operações de mudança. Nesta plataforma, é possível criar um modelo de processo inicial e executar as suas instâncias, e, em seguida, detalhar o modelo de processo assim que mais conhecimento e informações estiverem disponíveis, como, por exemplo, a adição de novas atividades e transições. As novas instâncias de processo correspondentes são modificadas automaticamente após as alterações.

4.3 Considerações sobre o Capítulo

Este capítulo apresentou tanto uma revisão bibliográfica inicial da literatura como uma Busca Estruturada com o objetivo de analisar soluções existentes que estejam relacionadas, de alguma maneira, a esta proposta de trabalho. Porém, conforme pode ver visto no decorrer deste capítulo, existem diversas abordagens que tratam sobre execução / monitoramento e avaliação de processos, porém, apenas um dos trabalhos analisados aborda a realimentação do processo com base nos dados de execução. Assim, através do que foi apresentado como trabalhos existentes na área de pesquisa que vem sendo estudada, busca-se enfatizar a falta de abordagens que apresentem uma solução completa para execução, monitoramento e realimentação de processos visando a melhoria de seu desempenho, conforme definido nos objetivos da presente proposta de trabalho.

5. PROPOSTA

De acordo com o objetivo proposto no Capítulo 1, de *definir, implementar e avaliar uma sistemática baseada em reutilização para apoiar a execução, monitoramento, análise e realimentação de processos, visando melhorar o desempenho destes, utilizando dados de proveniência e ontologias*, esta proposta de trabalho foi dividida em quatro fases distintas (Figura 5.1), sendo elas:

1. Definição das medidas do processo a serem monitoradas;
2. Execução e monitoramento do processo;
3. Análise da execução do processo; e
4. Realimentação do processo.

Estas quatro fases são apresentadas nas Subseções 5.1 a 5.4.

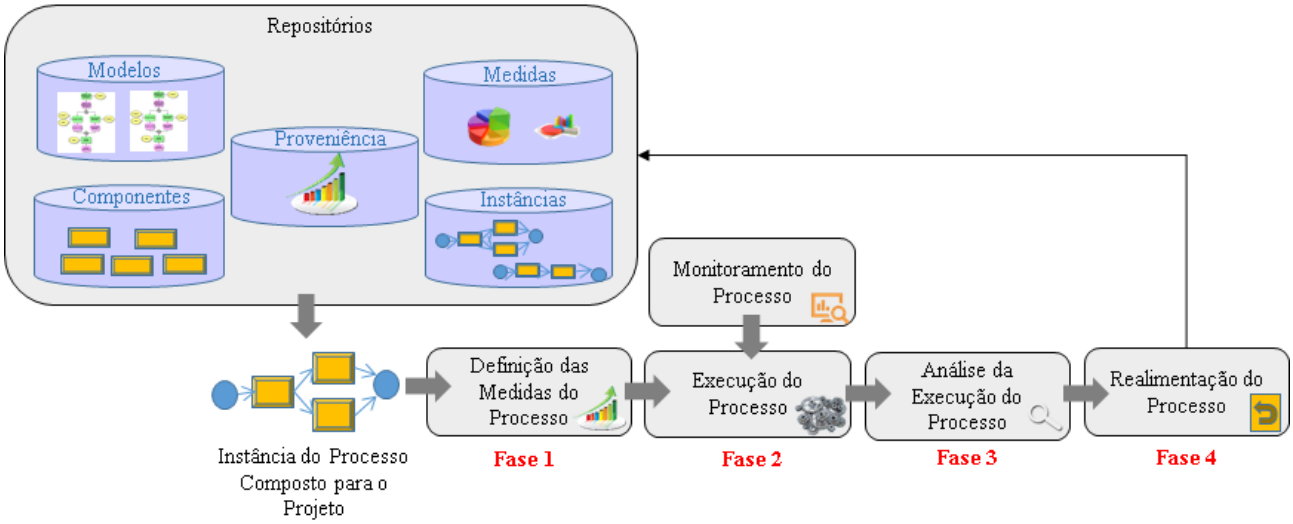


Figura 5.1: Fases da Proposta

A Fase 1 desta proposta, exibida na Figura 5.1, inicia-se após a modelagem do processo e a criação da instância deste processo que será executada. Para a definição do processo de software a ser executado, monitorado e realimentado, é usado um diagrama especificado em BPMN 2.0. Uma introdução sobre a modelagem de processos utilizando BPMN foi descrita na Seção 2.3. Após a definição deste modelo, que será utilizado para o armazenamento da proveniência prospectiva do processo, a execução das quatro fases desta proposta de trabalho pode ser realizada. Cada uma destas quatro fases é apresentada em detalhes nas seções seguintes.

5.1 Fase 1: Definição das Medidas do Processo

Na fase de definição das medidas do processo, deverão ser estabelecidas as medidas a serem coletadas e armazenadas durante a fase de Execução e Monitoramento do Processo (Seção 5.2). Durante essa fase, o gerente de projeto será capaz de definir uma ou mais medidas (medidas base[6] ou medidas derivadas[7]), alinhadas aos objetivos de negócio da organização.

PUTNAM e MYERS (2003) *apud* ROCHA *et al*. (2012) apresentam cinco medidas de software essenciais, a partir das quais podem ser obtidas a maioria das demais medidas utilizadas pelas organizações. Estas medidas, detalhadas a seguir, são interdependentes entre si.

1. **Quantidade de funções (tamanho):** mede-se utilizando linhas de código fonte (*Source Lines of Code* - SLOC) ou pontos por função;
2. **Produtividade:** mede-se por meio de linhas de código fonte ou pontos de função por homem-mês, utilizando dados históricos da organização. Ressalta-se que esta não é uma medida precisa, haja vista que não leva em consideração a produtividade da equipe durante as fases do desenvolvimento que não seja a codificação;
3. **Tempo:** mede-se, normalmente, em horas, dias, semanas ou meses, de acordo com o tamanho do projeto;
4. **Esforço:** medido em homem-horas ou homem-mês; e
5. **Confiabilidade:** mede-se em termos de taxa de defeitos, considerando-se que qualquer produto é entregue, ainda, com a presença de alguns defeitos.

A interdependência entre estas cinco medidas é apresentada na Tabela 5.1. Assim, por exemplo, se queremos um software que seja desenvolvido de forma mais rápida, a diminuição do tempo de desenvolvimento acarretará no aumento do esforço, além de, provavelmente, diminuir sua confiabilidade.

Tabela 5.1: Medidas Essenciais e suas Relações (adaptado de [PUTNAM e MYERS, 2003])

	Tempo	**Esforço**	**Confiabilidade**	**Tamanho**	**Produtividade**
Desenvolvimento Mais Rápido	↓	↑	↓	↓	↑
Desenvolvimento Melhor	↑	↑	↑	↑	↑
Desenvolvimento Mais Barato	↑	↓	↓	↓	↑

Considerando que as medidas de software, para serem efetivas, devem estar alinhadas aos objetivos estratégicos do negócio da organização e atender às necessidades dos gerentes e

[6] Exemplos: prazo estimado para o projeto, prazo real do projeto.
[7] Exemplo: aderência ao prazo do projeto, dada pela razão entre o prazo real do projeto e o prazo estimado para o projeto.

engenheiros de software (ROCHA *et al*., 2012), esta proposta prevê a possibilidade de criação de novas medidas de forma a atender a estes objetivos, durante esta fase de definição de medidas.

A definição de novas medidas nesta abordagem será estabelecida conforme proposto por (ROCHA *et al*., 2012), devendo conter as seguintes informações (Tabela 5.2):

Tabela 5.2: Informações para a Definição de Novas Medidas (adaptado de ROCHA et al., 2012)

1	**Nome:**	Nome da medida
2	**Definição**	Descrição sucinta da medida
3	**Mnemônico:**	Sigla utilizada para identificar a medida
4	**Tipo de Medida:**	Classificação da medida quanto à sua dependência funcional (medida base ou medida derivada).
5	**Entidade Medida:**	Entidade que a medida mede.
6	**Propriedade Medida:**	Propriedade da entidade medida que é quantificada pela medida.
7	**Unidade de Medida:**	Unidade de medida em relação à qual a medida é medida.
8	**Tipo de Escala:**	Natureza dos valores que podem ser atribuídos à medida.
9	**Valores da Escala:**	Valores que podem ser atribuídos à medida.
10	**Intervalo Esperado dos Dados:**	Limites de valores da escala definida de acordo com dados históricos ou com metas estabelecidas.
11	**Fórmula de Cálculo de Medida:**	Fórmula utilizada no procedimento de medição de medidas derivadas, para calcular o valor atribuído à medida considerando-se sua relação com outras medidas ou com outros valores.
12	**Procedimento de Medição:**	Descrição do procedimento que deve ser realizado para coletar uma medida.
13	**Momento da Medição:**	Momento em que deve ser realizada a coleta e registro de dados para a medida.
14	**Periodicidade de Medição**	Frequência de coleta da medida.
15	**Responsável pela Medição:**	Papel desempenhado pelo recurso humano responsável pela coleta da medida.
16	**Procedimento de Análise:**	Descrição do procedimento que deve ser realizado para representar e analisar os dados coletados para uma medida, incluindo as ferramentas analíticas que devem ser utilizadas.
17	**Momento da Análise de Medição:**	Momento em que deve ser realizada a análise de dados coletados para a medida.
18	**Periodicidade da Análise:**	Frequência de análise de dados da medida.
19	**Responsável pela Análise:**	Papel desempenhado pelo recurso humano responsável pela análise da medida.

A Tabela 5.3 apresenta um exemplo de criação de uma nova medida, de acordo com as informações apresentadas na Tabela 5.2.

Tabela 5.3: Exemplo de Medida criada (ROCHA et al., 2012)

1	**Nome:**	Precisão Da Estimativa de Tempo da Fase Especificação e Análise de Requisitos do Sistema
2	**Definição**	Medida utilizada para quantificar a precisão da estimativa de duração da fase Especificação e Análise de Requisitos do Sistema.
3	**Mnemônico:**	PET-EAR
4	**Tipo de Medida:**	Medida Derivada
5	**Entidade Medida:**	Subprocesso Planejamento do Projeto
6	**Propriedade Medida:**	Precisão da estimativa de tempo
7	**Unidade de Medida:**	-
8	**Tipo de Escala:**	Absoluta
9	**Valores da Escala:**	Números reais positivos, utilizando-se uma precisão de duas casas decimais.
10	**Intervalo Esperado dos Dados:**	[0.8, 1.0]
11	**Fórmula de Cálculo de Medida:**	PET-EAR = TE-EAR/TR-EAR Onde: TR-EAR = tempo real despendido em atividades da fase Especificação e Análise de Requisitos do Sistema no período considerado; TE-EAR = tempo estimado para atividades da fase Especificação e Análise de Requisitos do Sistema no período considerado.
12	**Procedimento de Medição:**	Calcular a precisão da estimativa de tempo utilizando a fórmula de cálculo da medida.
13	**Momento da Medição:**	Ao final da semana, na Atividade Registrar Dados para Monitoramento do Projeto.
14	**Periodicidade de Medição**	Semanal.
15	**Responsável pela Medição:**	Analista de Sistemas.
16	**Procedimento de Análise:**	- Representar em um gráfico de controle os valores coletados para a medida no projeto. - Analisar o comportamento do processo em relação à *baseline* de desempenho estabelecida para o processo: (i) Se os valores coletados para a medida encontrarem-se dentro dos limites da *baseline* de desempenho, o processo apresenta comportamento adequado. (ii) Se houver valores fora dos limites da *baseline*, o comportamento do processo não está adequado. É necessário investigar as causas da instabilidade no comportamento do processo e identificar ações corretivas, quando pertinente.
17	**Momento da Análise de Medição:**	Atividade Analisar Desempenho dos Processos do Projeto.
18	**Periodicidade da Análise:**	Semanal
19	**Responsável pela Análise:**	Gerente do Projeto

Conforme apresentado em (ROCHA *et al*., 2012), a coleta e o armazenamento dos dados de medidas sendo realizados de forma automatizada é o ideal para evitar a possibilidade de erros e

omissões que possam ser causados por uma coleta manual destes dados. Assim, nesta proposta de tese será adotado um mecanismo para o estabelecimento das medidas de processo de forma que estas possam ser capturadas de forma automatizada. O gerente do projeto poderá definir as medidas que deverão ser coletadas de acordo com os objetivos do projeto, através de uma ferramenta que disponibilize a função de definição de tais medidas (utilizando um formulário específico para isto, tendo como base os itens propostos na Tabela 5.2) e a incorporação de rotinas ao modelo de fluxo do processo (por exemplo, inclusão de novas tarefas automatizadas) que possibilitarão a captura das informações necessárias a estas medidas.

Posteriormente à definição das medidas que deverão ser realizadas sobre o processo, é necessário definir como os dados a serem utilizados para estabelecer as medições serão coletados e armazenados. Esta coleta é realizada na Fase 2 da abordagem proposta neste trabalho, ou seja, durante a execução e monitoramento do processo de software.

5.2 Fase 2: Execução e Monitoramento do Processo

Posteriormente às adaptações no fluxo de processo, de forma a permitir a coleta dos dados necessários às medidas, o mesmo estará pronto para ser executado em uma ferramenta para execução e monitoramento de processos (vide Seção 2.2.2). Nesta proposta, para a execução dos processos (definidos utilizando a notação BPMN), será utilizada a ferramenta jBPM (RED HAT, 2015).

Após a etapa de execução, as variáveis coletadas para o estabelecimento das medidas serão utilizadas para a geração das mesmas. As medidas geradas serão armazenadas em um repositório, a ser definido, com o objetivo de garantir a segurança e a recuperação futura destas.

Além dos dados necessários para o estabelecimento das medidas definidas na Fase 1, durante esta fase de execução e monitoramento também são capturados os dados da proveniência retrospectiva do processo. Estes dados serão capturados e posteriormente armazenados e analisados através da utilização da abordagem PROV-Process (Seção 3.3), que consiste em uma arquitetura especificada para captura, armazenamento e análise dos dados de proveniência de processos, utilizando o modelo PROV (BELHAJJAME *et al*. 2012).

A abordagem PROV-Process oferece um repositório relacional padrão para armazenamento dos dados de proveniência tanto prospectiva quanto retrospectiva e segue as especificações estabelecidas pelas recomendações do modelo PROV-DM (*The PROV Data Model*), com adaptações voltadas para o armazenamento de dados de proveniência de processos de software. As tabelas do repositório do PROV-Process, para armazenamento da proveniência retrospectiva são exibidas na Figura 5.2.

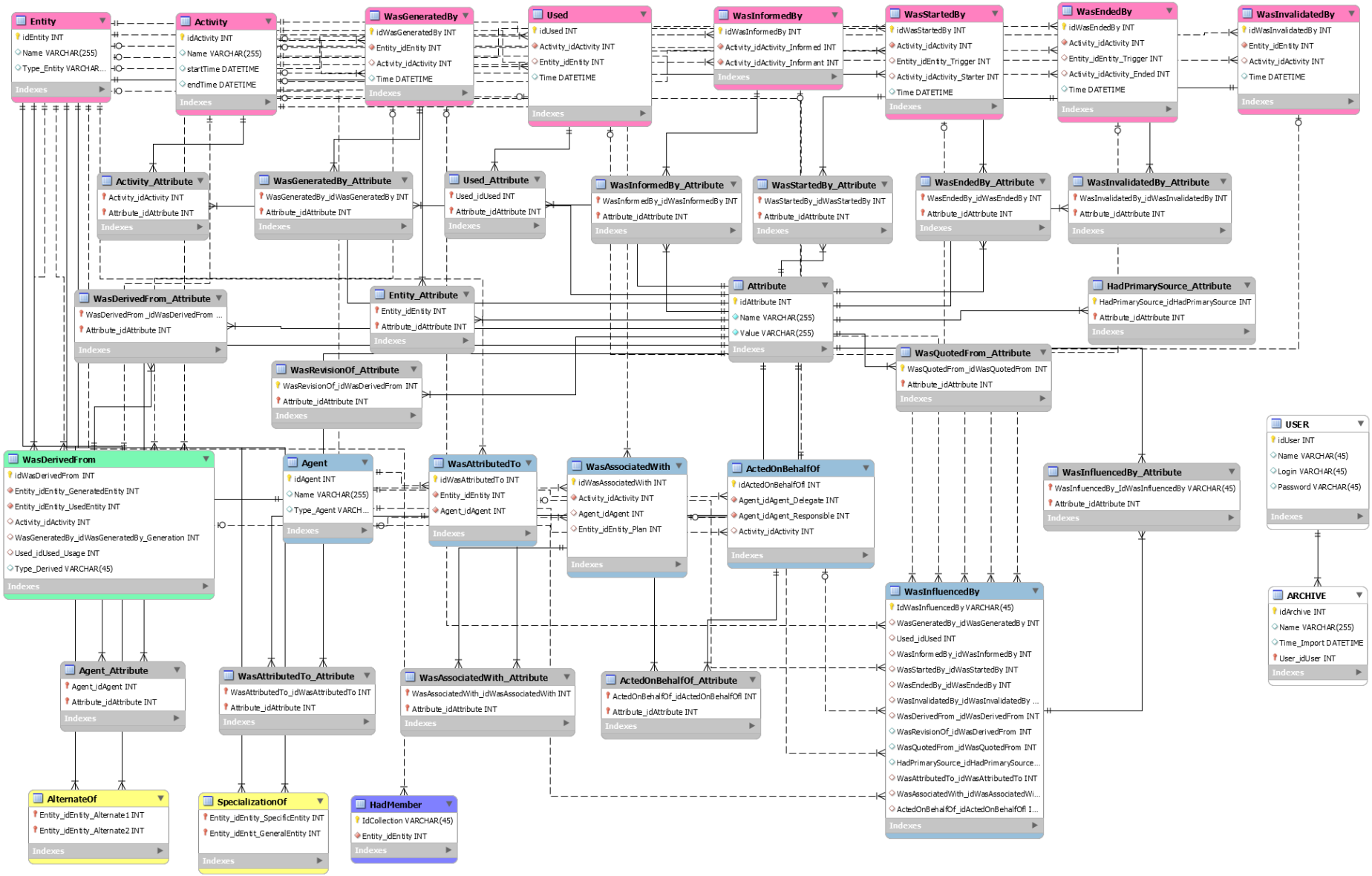


Figura 5.2: Diagrama de Tabelas e Relacionamentos do PROV-Process

Neste repositório, foram modelados como tabelas os três vértices do modelo PROV, sendo elas entidade (*Entity*), atividade (*Activity*), agente (*Agent*), além das relações causais existentes entre estas entidades, como: *wasGeneratedBy, used, wasInformedBy, wasStartedBy, wasEndedBy, wasInvalidatedBy, wasDerivedFrom, wasAttributedTo, wasAssociatedWith, actedOnBehalfOf, alternateOf, specializationOf, hadMember*. Além destas tabelas específicas do modelo PROV, foram criadas duas tabelas extras, *User* e *Archive*. A tabela *User* foi criada especificamente para o controle dos usuários do sistema desenvolvimento, provendo suporte à abordagem e a tabela *Archive* foi criada com o intuito de armazenar informações dos arquivos importados com dados de execução de processos (em formato .csv) na ferramenta criada, já que, atualmente, esta ferramenta possui um importador de dados de execução de processos. Futuramente, porém, espera-se utilizar esta abordagem de forma integrada à proposta desta tese, sem a necessidade de exportar/importar dados das execuções dos processos. Além da criação destas duas tabelas extras, foram inseridos os campos *Started* e *Ended* na tabela que representa a relação *wasAssociatedWith,* com o objetivo de capturar a data/hora de início e de término de uma atividade que foi executada por um agente. Também foram criados os campos *Name* nas tabelas *Agent* e *Entity*, e o campo *Description* na tabela *Activity*, com o objetivo de se ter um identificar textual para os registros inseridos nas mesmas.

Atualmente, a abordagem PROV-Process não possui suporte para armazenamento da proveniência prospectiva (FREIRE *et al*. 2008), assim, as adaptações no modelo de dados da abordagem PROV-Process para o armazenamento deste tipo de proveniência encontram-se em desenvolvimento, tendo como base as extensões propostas por MISSIER *et al*. (2013).

Tem-se como proposta a utilização deste repositório de acordo com critérios pré-estabelecidos e partindo do princípio que os elementos do PROV estão estreitamente ligados aos elementos que podem ser representados através da notação BPMN (alguns exemplos desta relação podem ser visualizados na Tabela 5.4). Deve-se ressaltar que estes critérios foram estabelecidos a partir de uma única base de dados de execução real de um dos processos apresentados no Capítulo 6. Assim, novos critérios poderão ser adicionados aos listados a seguir, a partir do momento em que forem avaliadas outras bases de dados de execução no decorrer do desenvolvimento desta tese.

Tabela 5.4: Relação entre os vértices do PROV e os elementos do BPMN

Representação BPMN		Representação PROV	
Tarefa		Atividade	
Artefato		Entidade	
Ator	-	Agente	

Os critérios estabelecidos para a utilização da abordagem PROV-Process até o momento são:

1. Todas as execuções de um processo específico serão armazenadas na tabela *Archive*, considerando que cada registro nesta tabela representará uma execução distinta de uma instância de processo. O campo *idArchive* conterá o identificador da instância executada.
2. Todas as tarefas executadas em uma instância específica de processo serão armazenadas na tabela *Activity*, sendo sua descrição a mesma do nome da tarefa identificada no modelo de fluxo descrito em BPMN.
3. O executor de uma tarefa será armazenado como registro da tabela *Agent*, além da criação de um registro na tabela *wasAssociatedWith*, representando a relação entre *Activity* e *Agent*. Nesta tabela *wasAssociatedWith* também serão inseridos a data/hora de início e término da execução da *Activity* pelo *Agent*
4. Os artefatos gerados e consumidos pelas atividades durante a execução do processo serão armazenados como registros da tabela *Entity*, sendo criadas as relações de acordo com a ação realizadas sobre o artefato durante a execução, podendo ser *used*, *wasStartedBy*, *wasEndedBy*, *WasGeneratedBy*. A relação *wasAttributedTo* será utilizada para estabelecer que um artefato foi utilizado por um determinado agente.
5. As relações *wasRevisionOf* e *wasDerivedFrom* serão capturadas ao serem criados novos artefatos durante a execução do processo, se estes forem revisões de artefatos existentes ou derivados de artefatos já existentes.

Para exemplificar como funcionará o mecanismo de captura e armazenamento de dados da proveniência retrospectiva com base nos dados de execução do processo, de forma simplificada, foi

definido um processo para tratar alterações em um software[8]. Este processo é composto por três tarefas (i) *Abertura da Requisição de Mudanças no Software*, (ii) *Implementação da Solução* e (iii) *Alterar Requisição de Mudança para Concluída*, conforme exibido na Figura 5.3.

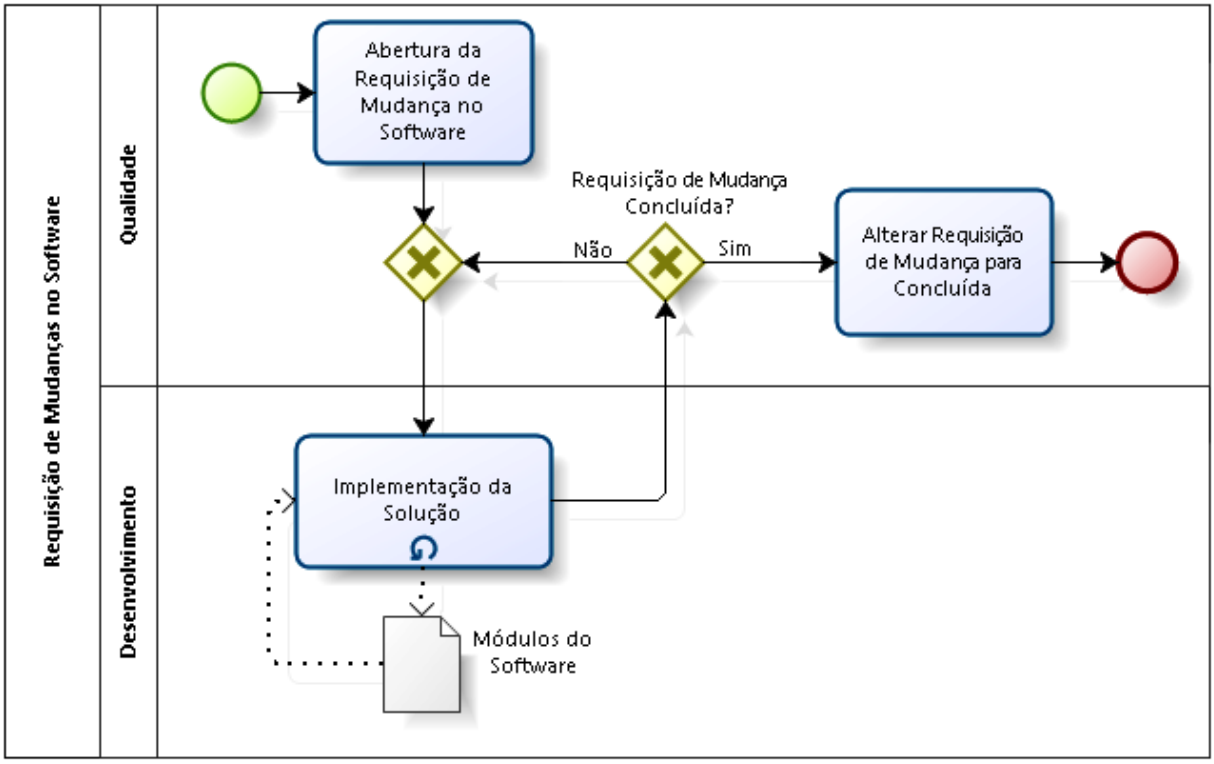


Figura 5.3: Processo para Requisição de Mudanças no Software

De acordo com o processo da Figura 5.3, a tarefa (ii) *Implementação da Solução* pode ou não ser realizada de forma colaborativa, ou seja, um ou mais atores podem atuar nesta tarefa buscando atender o que foi descrito na *Abertura da Requisição de Mudança no Software*. A tarefa (ii) também é uma tarefa que poderá ser realizada mais de uma vez em uma mesma instância deste processo, haja vista que será avaliada, após a sua execução, verificando a necessidade de 're'-execução ou não da mesma.

A partir de uma simulação da execução do processo acima, foram obtidos os dados de execução de uma Requisição de Mudança específica, cuja tarefa de Implementação da solução foi realizada 5 vezes, até que a requisição fosse aprovada e marcada como concluída pela equipe de Qualidade, conforme apresentado nas Tabelas 5.5 e 5.6:

[8] Deve-se ressaltar que uma descrição mais completa e detalhada de todas as etapas desta proposta, aplicadas a exemplos de processos reais da indústria, é apresentado no Capítulo 6.

Tabela 5.5: Dados de Execução da Tarefa

ID	Tarefa Executada	Início	Término	Módulos Manipulados	Ator Envolvido
1	Abrir Requisição de Mudança	2015-05-04 09:00:00	2015-05-04 09:05:00	-	marc.marseau
2	Implementação da Solução	2015-05-04 09:10:00	2015-05-04 10:30:00	DLL - Cálculos	april.sanchez
3	Implementação da Solução	2015-05-04 11:00:00	2015-05-04 14:00:00	DLL - Cálculos	helen.kelly
4	Implementação da Solução	2015-05-04 16:00:00	2015-05-04 16:35:00	DLL - Cálculos DLL - NF-e v2.0	helen.kelly, favio.riviera
5	Implementação da Solução	2015-05-05 09:00:00	2015-05-05 12:00:00	DLL - ERP Exportação de Informações	anthony.nichols
6	Implementação da Solução	2015-05-05 14:35:00	2015-05-05 15:05:00	DLL - Cálculos DLL - ERP Exportação de Informações	favio.riviera
7	Implementação da Solução	2015-05-06 09:10:00	2015-05-06 11:05:00	DLL - ERP Exportação de Informações	april.sanchez
8	Alterar RDM para Concluída	2015-05-06 14:00:00	2015-05-06 14:35:00	-	marc.marseau

Tabela 5.6: Artefatos Manipulados por Atores durante a execução

Artefato Manipulado	Atores que manipularam o Artefato
DLL – Cálculos	april.sanchez, favio.riviera, helen.kelly
DLL - ERP Exportação de Informações	anthony.nichols, april.sanchez , favio.riviera
DLL - NF-e v2.0	favio.riviera, helen.kelly

Após a coleta destes dados, os mesmos foram importados na base de dados relacional da abordagem PROV-Process, de acordo com os cinco critérios estabelecidos para a utilização desta abordagem, conforme mencionado anteriormente.

Além de permitir o armazenamento de dados de proveniência, a abordagem PROV-Process oferecerá como funcionalidade uma interface para que seja realizada a construção de um arquivo OWL (*Web Ontology Language*) que modela a proveniência capturada de um determinado processo na ontologia PROV-O (BELHAJJAME *et al*., 2013). A exportação destes dados se dá por meio da criação do arquivo OWL com base na ontologia PROV-O, adicionando a esta ontologia indivíduos que são representados na base relacional como Atividades, Agentes e Entidades (*Activity, Agent* e *Entity*) além dos relacionamentos entre estes indivíduos, de acordo com os dados de execução do processo.

As Figuras 5.4, 5.5 e 5.6 apresentam, respectivamente, as tarefas do processo que foram representadas como indivíduos da classe *Activity* na ontologia, os atores do processo que foram representados como indivíduos da classe *Person* (uma subclasse da classe *Agent*), na ontologia e os artefatos do processo que foram representados como indivíduos da classe *Entity* na ontologia.

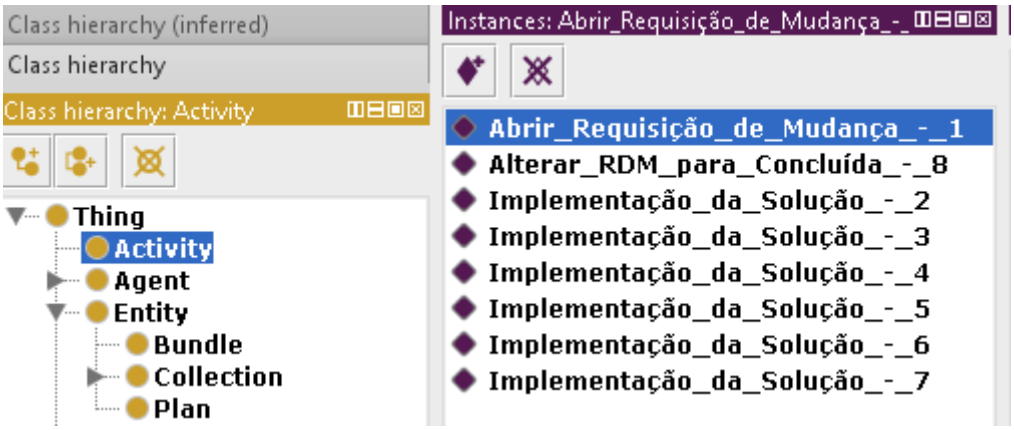


Figura 5.4: Indivíduos da classe 'Activity'

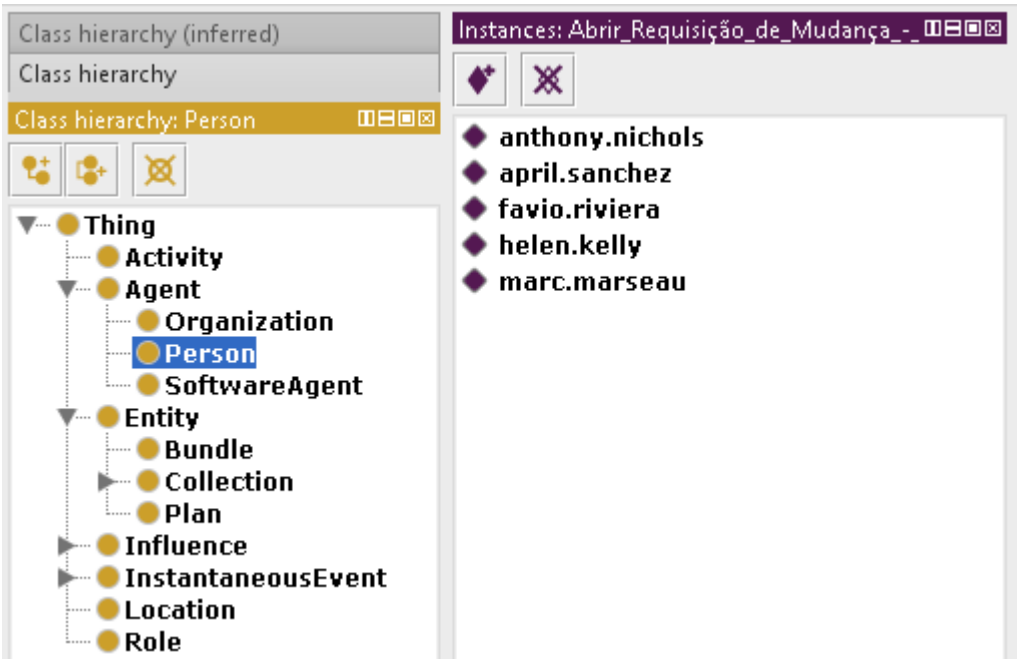


Figura 5.5: Indivíduos da classe 'Person', que é uma subclasse de 'Agent'

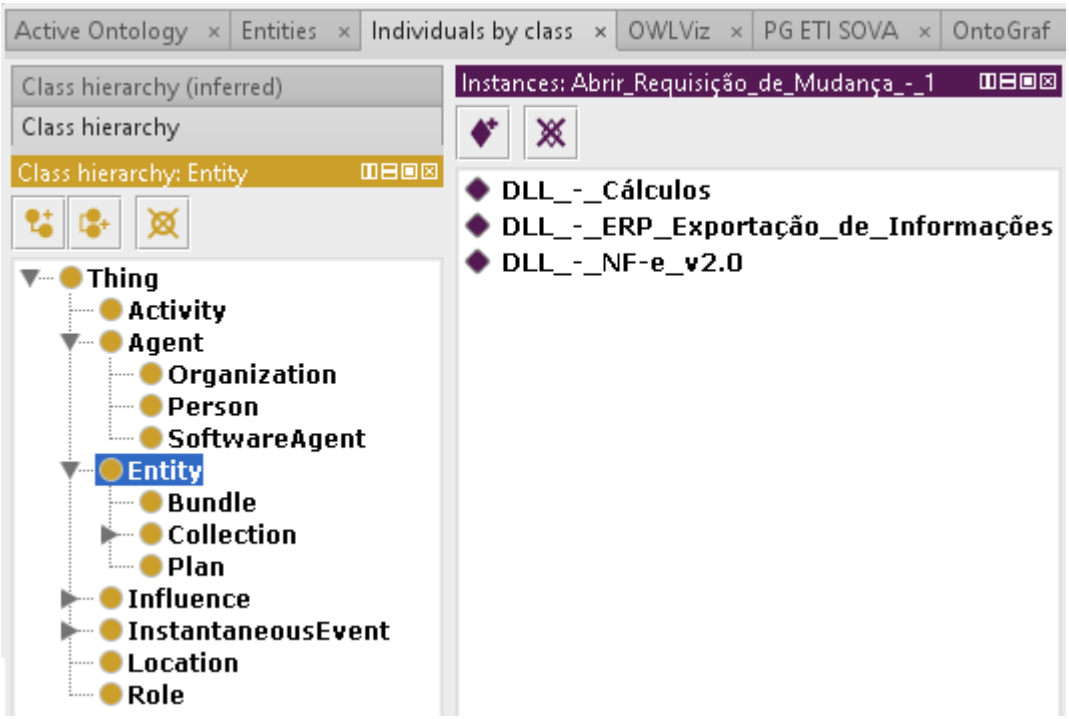


Figura 5.6: Indivíduos da classe 'Entity'

Além dos dados de execução referentes a tarefas, atores e artefatos estarem especificados na ontologia como indivíduos, também foram especificados nesta ontologia os relacionamentos entre estes dados, conforme exibido nas Figuras 5.7 a 5.12.

A Figura 5.7 apresenta que a tarefa *Abrir Requisição de Mudança – 1* foi criada como indivíduo da classe *Activity* e que a mesma está associada ao ator *marc.marseau* (para isto, foi utilizada a propriedade/relacionamento *wasAssiciatedWith* do modelo PROV), já que, de acordo com os dados de execução desta tarefa, a mesma foi realizada pelo ator *marc.marseau*. Também foi relacionado a esta tarefa seu horário de início e de término, utilizando as propriedades (*data properties*) *startedAtTime* e *endedAtTime*.

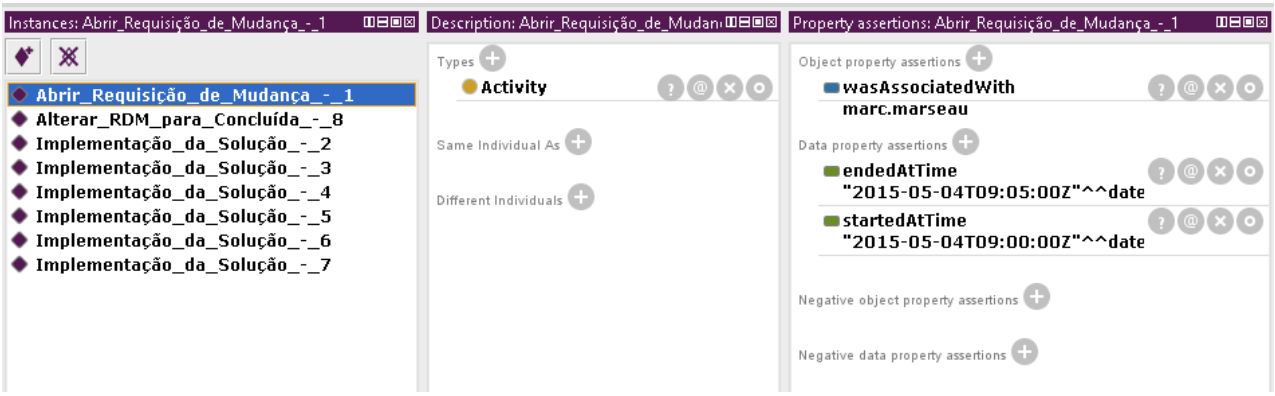


Figura 5.7: Especificação da Execução da Atividade 1 – Abrir Requisição de Mudança

Todas as outras sete tarefas executadas no exemplo citado foram modeladas na ontologia com seus respectivos relacionamentos. Outro exemplo de especificação de tarefa executada pode ser visualizado na Figura 5.8, na qual a primeira execução da atividade *Implementação da Solução – 2* está associada ao ator *april.sanchez* e utilizou o artefato *DLL – Cálculos*, além da especificação de seu horário de início e de término.

Figura 5.8: Especificação da Execução da Atividade 2 – Implementação da Solução

Os atores citados na coluna 'Ator Envolvido' da Tabela 5.6 de dados de execução do exemplo foram todos modelados na ontologia como indivíduos da classe *Person*, que é uma subclasse da classe *Agent*, conforme pode ser visualizado na Figura 5.9.

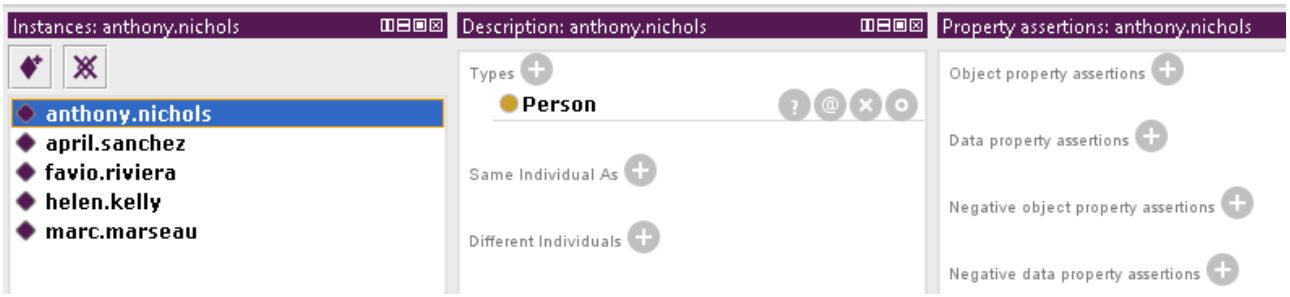


Figura 5.9: Especificação dos Agentes

Os artefatos citados na coluna 'Artefato Manipulado' da Tabela 5.7 de dados de execução do exemplo foram todos modelados na ontologia como indivíduos da classe *Entity*, além de serem estabelecidas suas relações com os atores que os manipularam, conforme pode ser visualizado nas Figuras de 5.10 a 5.12, pela propriedade *wasAttributedTo*.

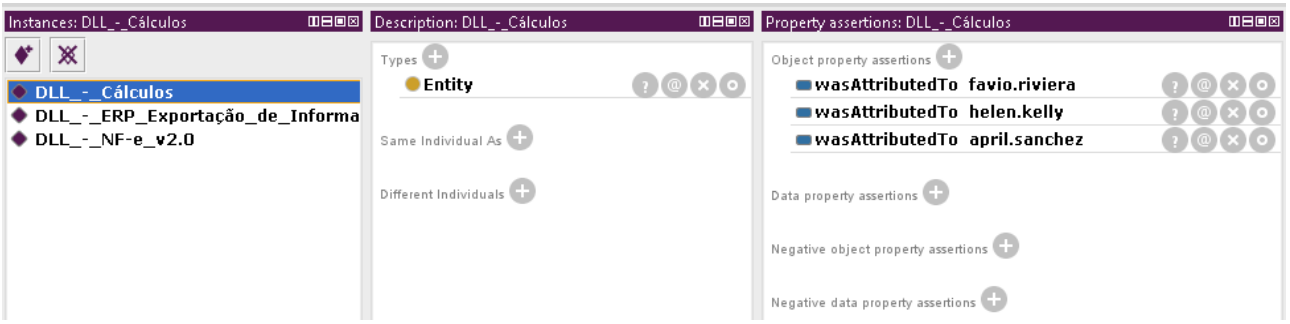


Figura 5.10: Especificação da Entidade DLL - Cálculos

Figura 5.11: Especificação da Entidade DLL – ERP – Exportação de Informações

Figura 5.12: Especificação da Entidade DLL – NF-e v2.0

Após a execução do processo e a coleta tanto das métricas previamente definidas como dos dados de proveniência, passa-se para a Fase 3 desta proposta, que consiste em analisar os dados obtidos durante a Fase 2.

5.3 Fase 3: Análise da Execução do Processo

Esta fase consiste em apresentar ao responsável pelo projeto diferentes mecanismos de análise da execução do processo, considerado dados de execução, métricas e dados de proveniência que foram capturados durante a Fase 2. Como exemplos destes mecanismos, pode-se citar diferentes formas de visualização dos dados de execução, como grafos que representam o fluxo real do processo executado, de forma a possibilitar ao responsável pelo gerenciamento do processo

analisar o que foi, de fato, executado e se este condiz com o fluxo de processo modelado. Um exemplo deste tipo de visualização, proposto por SANTANA (2015), é apresentado na Figura 5.13.

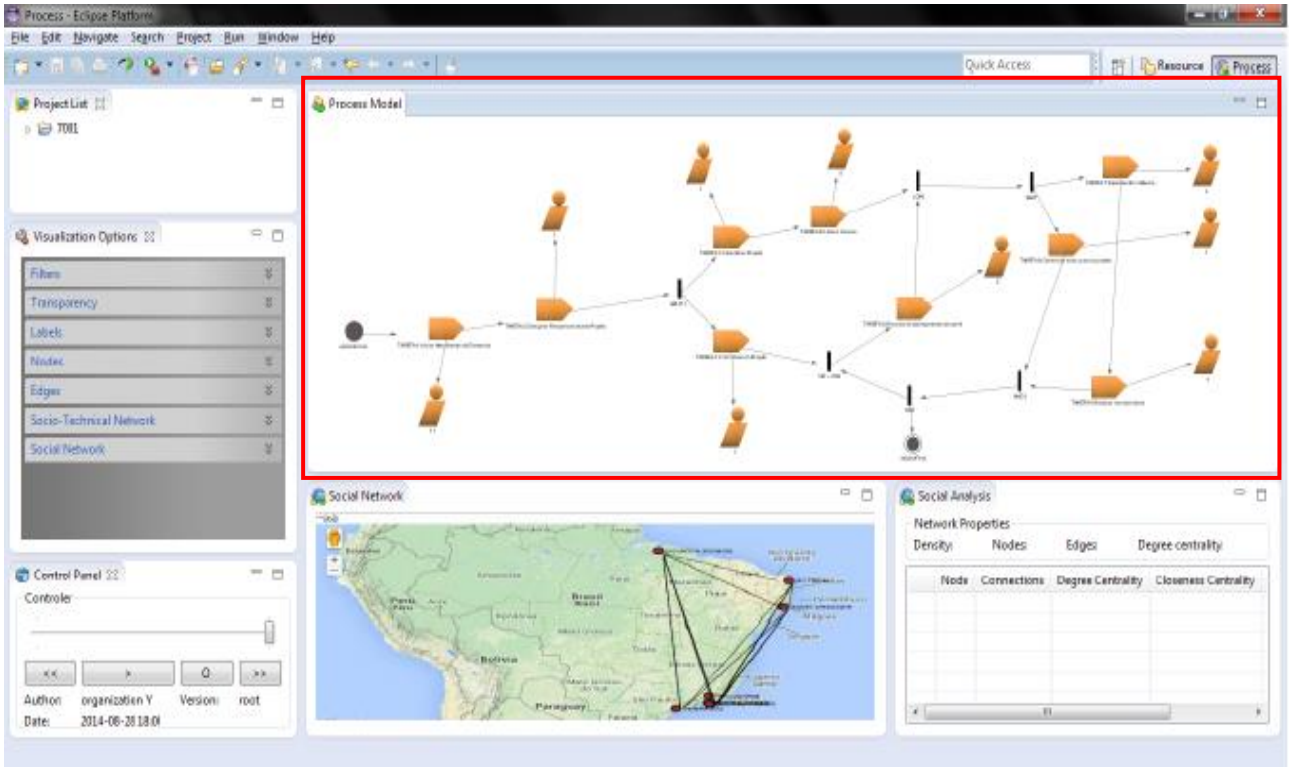


Figura 5.13: Tela do protótipo da EvolTrack-Process (SANTANA, 2005)

Com relação às métricas capturadas, estas poderão ser analisadas através de gráficos gerados com os valores coletados / calculados durante a execução, permitindo ao responsável pelo gerenciamento do processo a verificação dos valores obtidos e uma comparação destes com outras execuções do mesmo processo. Até o presente momento, ainda não foi definido quais gráficos seriam mais adequados para serem utilizados para a análise das métricas estabelecidas.

Com relação aos dados de proveniência capturados, estes poderão ser analisados pelo responsável pelo gerenciamento do processo, em um momento inicial, através de grafos de causalidade gerados através da proveniência retrospectiva capturada. Além disto, como um diferencial desta proposta de pesquisa, pode-se citar a possibilidade de oferecer ao responsável pelo projeto a derivação de informações estratégicas, que até então estavam implícitas sobre a execução do processo. Isto poderá ser realizado por meio dos mecanismos de inferência proporcionados por ontologias em conjunto com máquinas de inferência como HermiT (HERMIT, 2015), FaCT++ (FACT++, 2015) e Pellet (SIRIN, *et al*. 2007) sobre os dados de proveniência capturados durante a Fase 2.

Um exemplo simplificado de como os mecanismos de proveniência podem ser úteis para esta análise dos dados de execução é apresentado a seguir, utilizando o mesmo processo apresentado como exemplo da fase anterior, para tratar alterações em um software (Figura 5.3).

Após a execução da máquina de inferência sobre a ontologia PROV-O com os respectivos indivíduos e relacionamentos criados a partir dos dados de proveniência da execução do processo utilizado como exemplo, foram obtidas informações derivadas dos relacionamentos criados e que não estavam explícitas nos dados de execução de processo. Alguns exemplos destas informações são apresentados a seguir.

Ao utilizar a ferramenta Protégé para a manipulação de ontologia e de inferências realizadas sobre estas, os resultados retornados pelo motor de inferência podem ser visualizados em retângulos tracejados com cor de fundo bege, com seleção em vermelho da Figura 5.14.

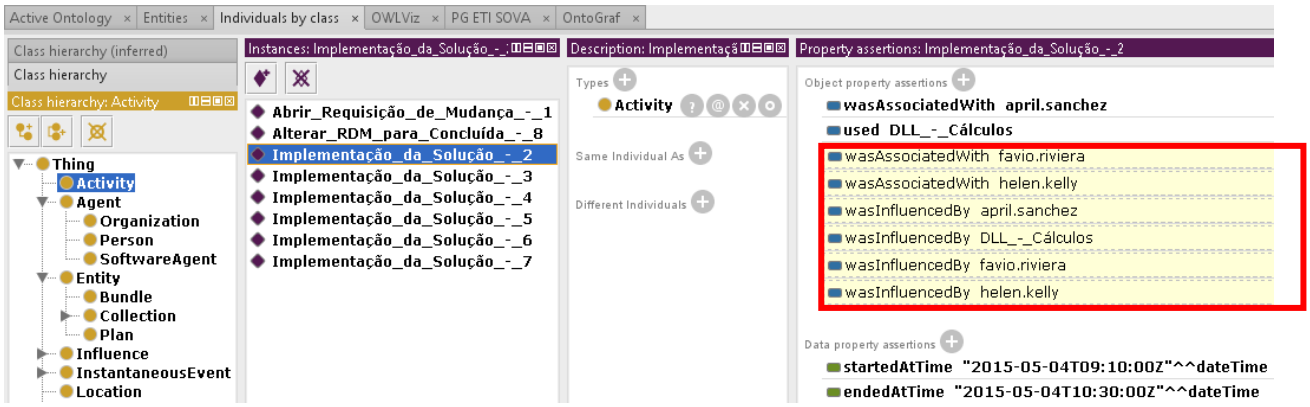


Figura 5.14: Inferência sobre a Activity 'Implementação da Solução 2'

Uma informação que até então estava implícita e foi trazida através das inferências realizadas com relação à tarefa *Implementação da Solução 2* (Figura 5.14) foram as relações (1) *wasAssociatedWith* favio.riviera e (2) *wasAssociatedWith* helen.kelly. Enquanto nos dados de execução do processo estava explícito apenas que, durante a execução da tarefa *Implementação da Solução 2*, apenas o ator *april.sanchez* estava vinculado à mesma, esta inferência traz uma informação estratégica que os atores *favio.riveira* e *helen.kelly* poderiam estar envolvidos na execução desta tarefa também, haja vista que, durante a execução de outras tarefas do processo, estes atores manipularam o mesmo artefato (DLL – Cálculo) que foi usado na tarefa *Implementação da Solução 2*. Assim, em uma próxima execução deste mesmo processo (se esta for realizada pela mesma equipe), o gerente do projeto, com posse desta informação, poderia sugerir que os três atores trabalhassem em conjunto na tarefa *Implementação da Solução 2*, o que poderia, talvez, evitar tantas repetições da tarefa de Implementação de Solução até que a mudança solicitada no sistema seja aprovada. Esta inferência foi realizada a partir de uma das regras de completude da ontologia

utilizada pelo PROV-Process, que foi criada com base na PROV-O. Esta regra define que, conforme a Figura 5.15, se uma atividade utiliza uma entidade e esta entidade foi atribuída a um agente, pode-se inferir que uma atividade está associada a um agente. Inferências com a mesma semântica da descrita anteriormente também foram feitas para as atividades Implementação da Solução 3, 4, 5, 6 e 7, conforme pode ser visualizado nas Figuras 5.16 a 5.20.

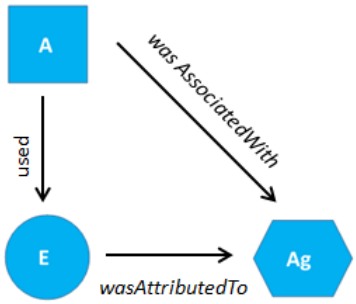


Figura 5.15: Associação entre as relações/propriedades do PROV-O

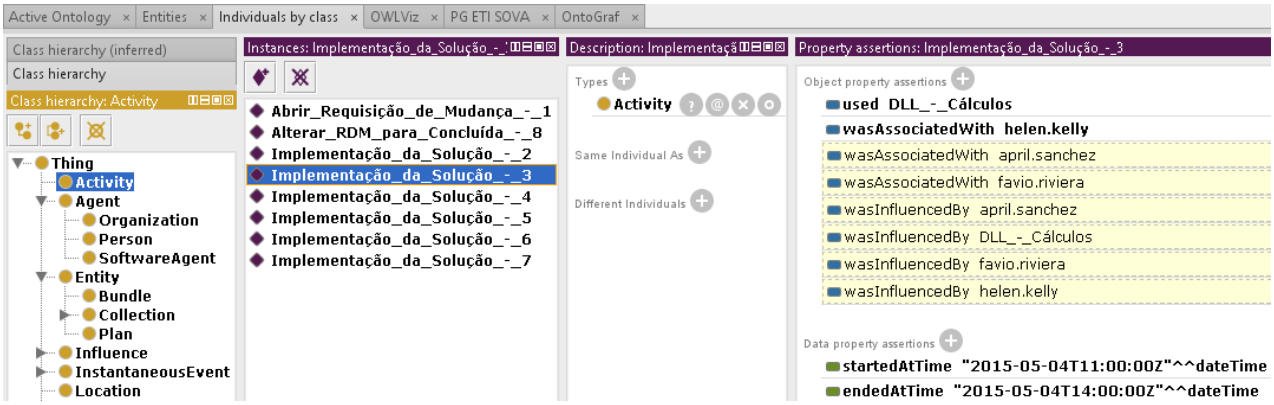


Figura 5.16: Inferência sobre a Activity 'Implementação da Solução 3'

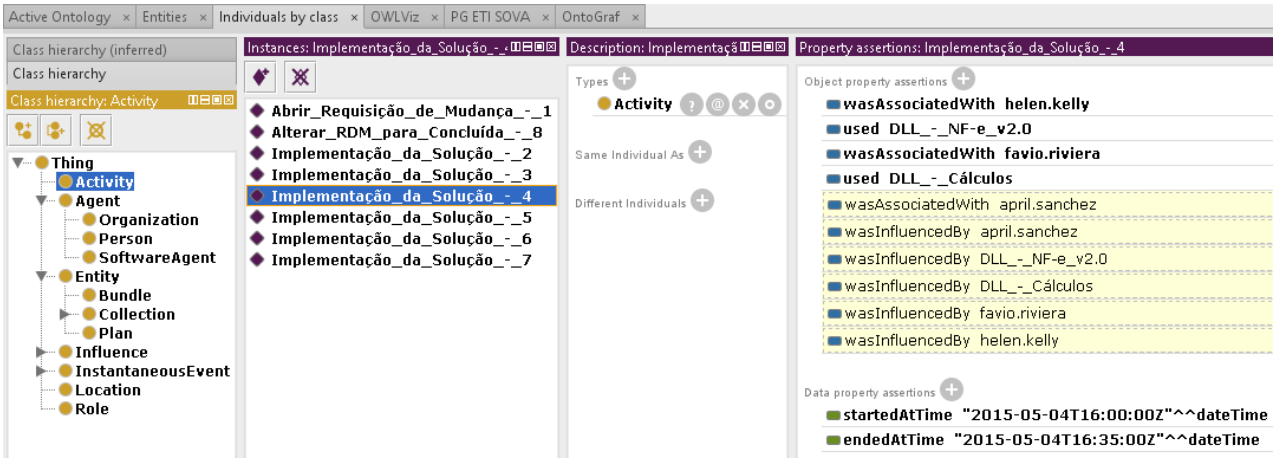


Figura 5.17: Inferência sobre a Activity 'Implementação da Solução 4'

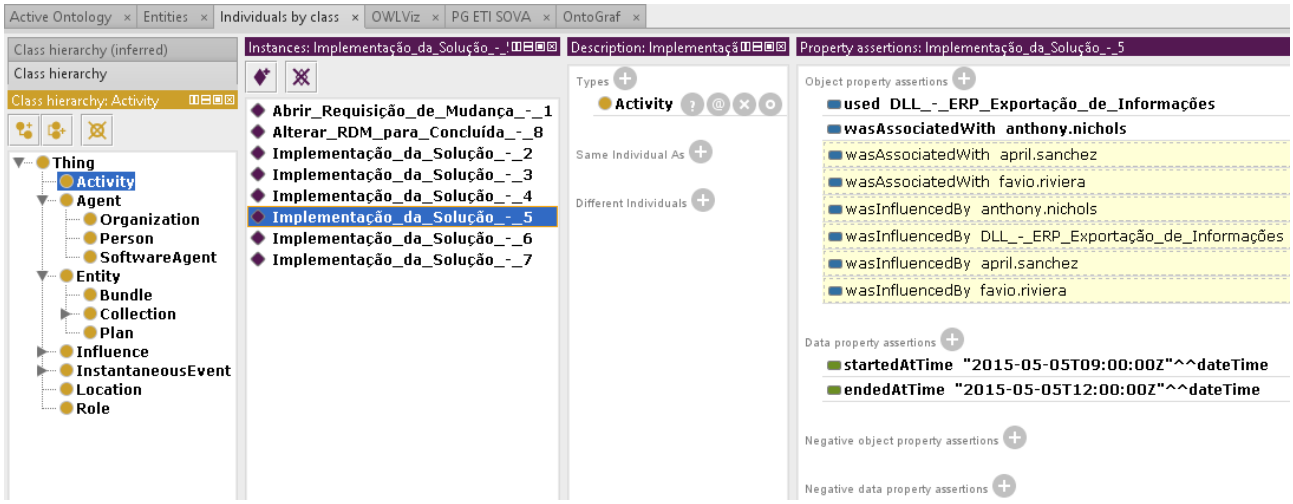


Figura 5.18: Inferência sobre a Activity 'Implementação da Solução 5'

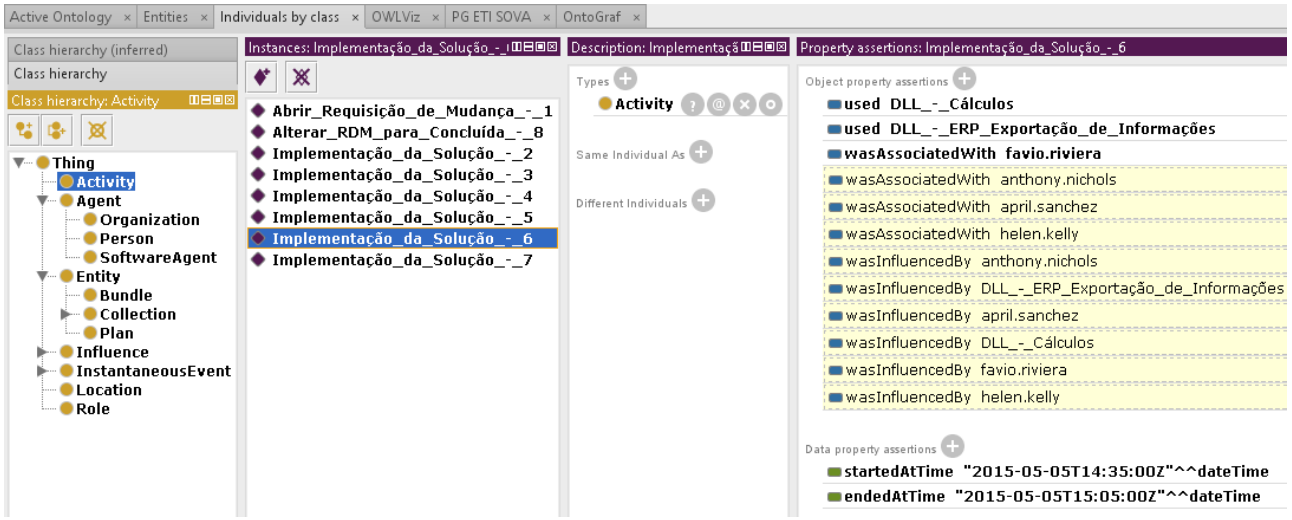


Figura 5.19: Inferência sobre a Activity 'Implementação da Solução 6'

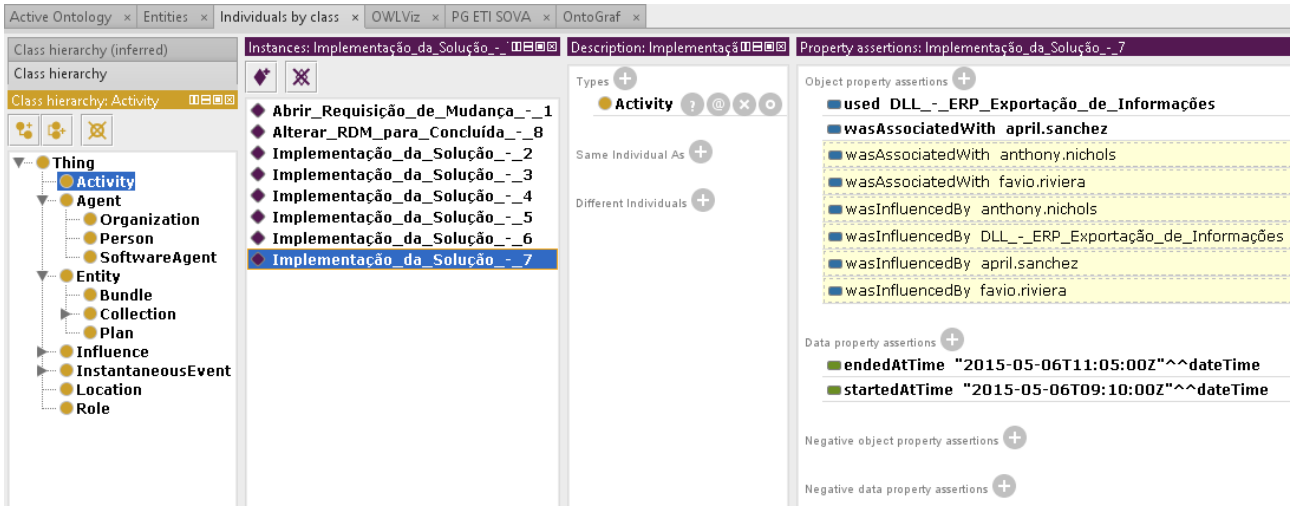


Figura 5.20: Inferência sobre a Activity 'Implementação da Solução 7'

Também foram inferidas informações para os agentes envolvidos na execução do processo, conforme pode ser visualizado nas Figuras 5.21 a 5.25. Embora na importação dos indivíduos para a ontologia os agentes criados não tivessem nenhum relacionamento diretamente ligado a eles, após a execução do mecanismo de inferência obteve-se em quais atividades o agente selecionado atuou e quais entidades ele manipulou durante a execução do processo. Estas informações foram obtidas de forma simples, através do Protégé, apenas clicando no agente desejado, sem a necessidade de realização de expressões de pesquisa elaboradas.

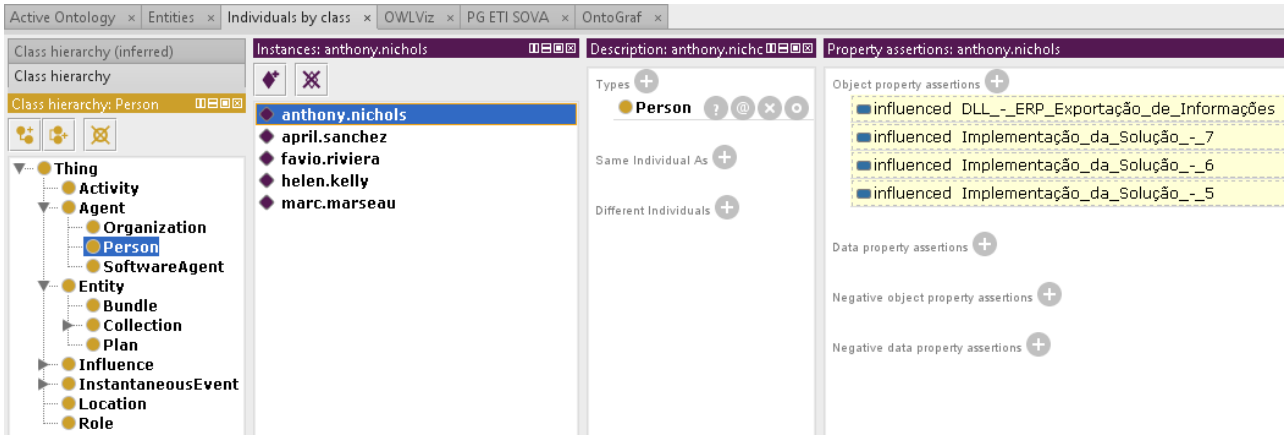


Figura 5.21: Inferência sobre o Agent 'anthony.nichols'

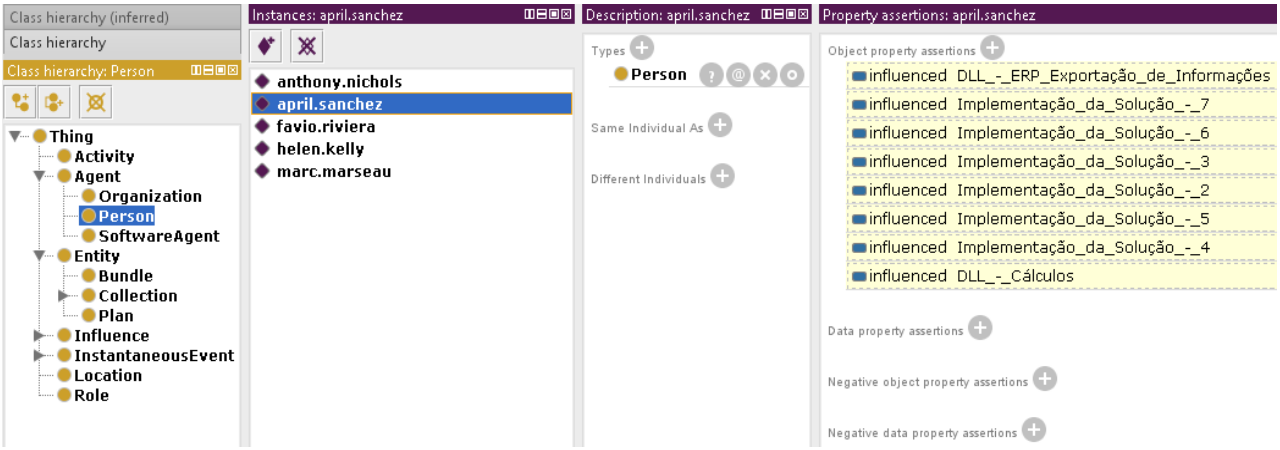


Figura 5.22: Inferência sobre o Agent 'april.sanchez'

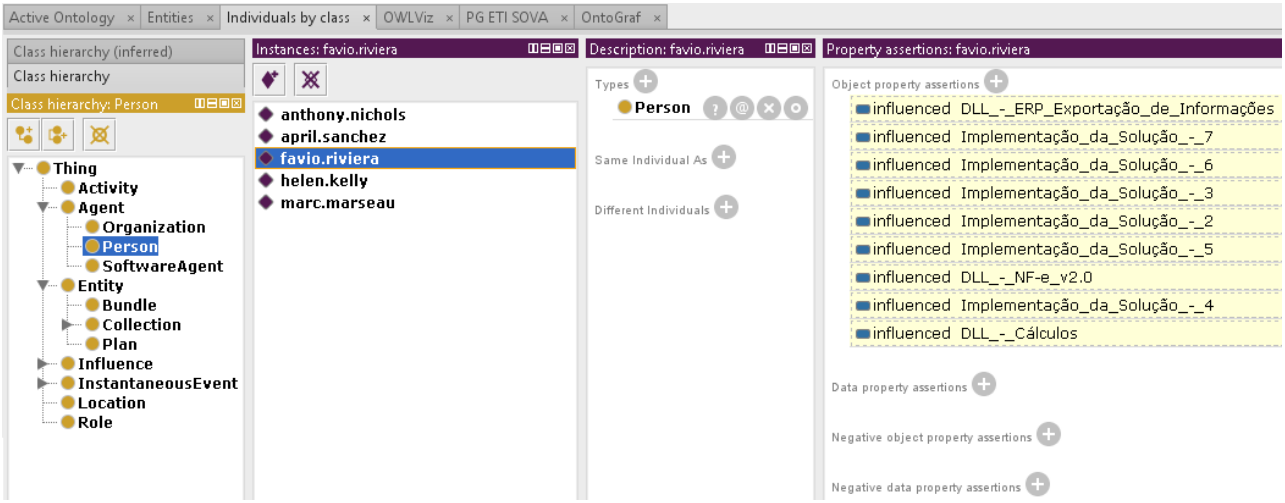


Figura 5.23: Inferência sobre o Agent 'favio.riviera'

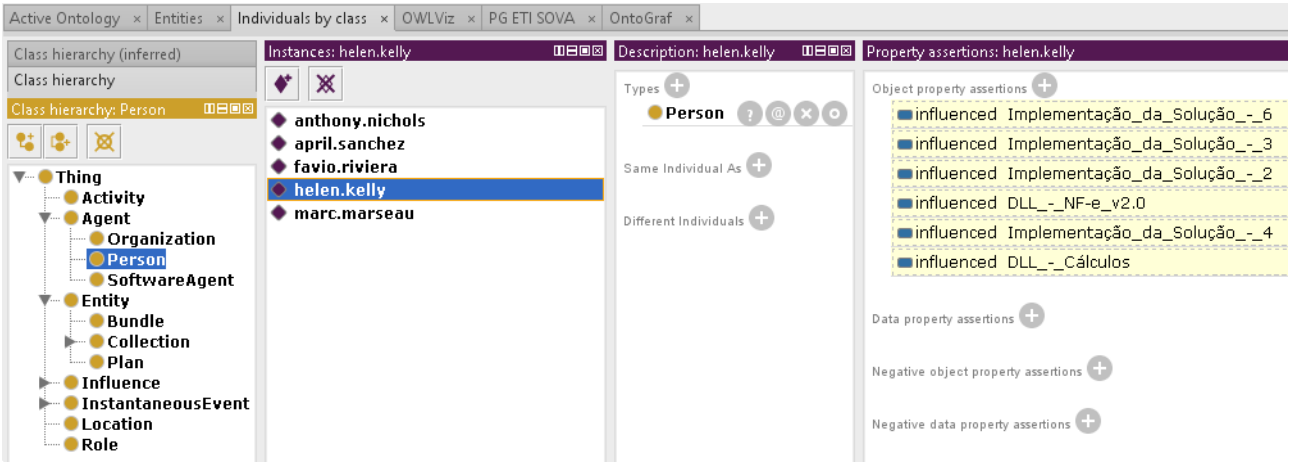


Figura 5.24: Inferência sobre o Agent 'helen.kelly'

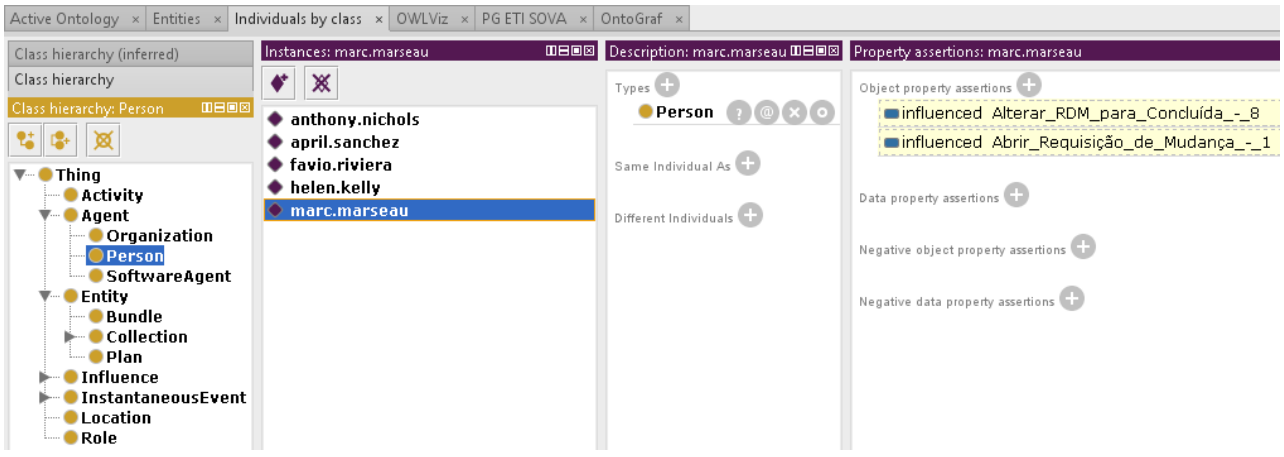


Figura 5.25: Inferência sobre o Agent 'marc.marseau'

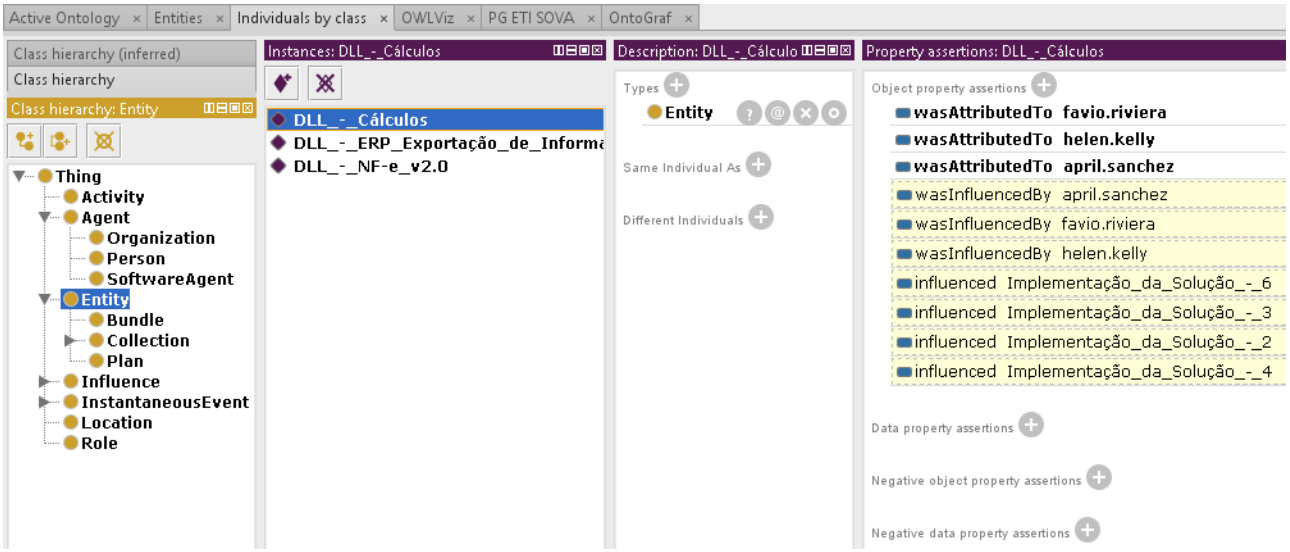


Figura 5.26: Inferência sobre a Entity 'DLL – Cálculos'

Com relação à inferência sobre os indivíduos cadastrados como entidades na ontologia, foram inferidas informações relativas aos agentes que atuaram sobre a entidade escolhida, além de todas as atividades em que esta entidade foi manipulada, conforme pode ser visualizado nas Figuras 5.26 a 5.28. Estas informações também foram obtidas de forma simples, através do Protégé, apenas clicando no agente desejado, sem a necessidade de realização de expressões de pesquisa elaboradas.

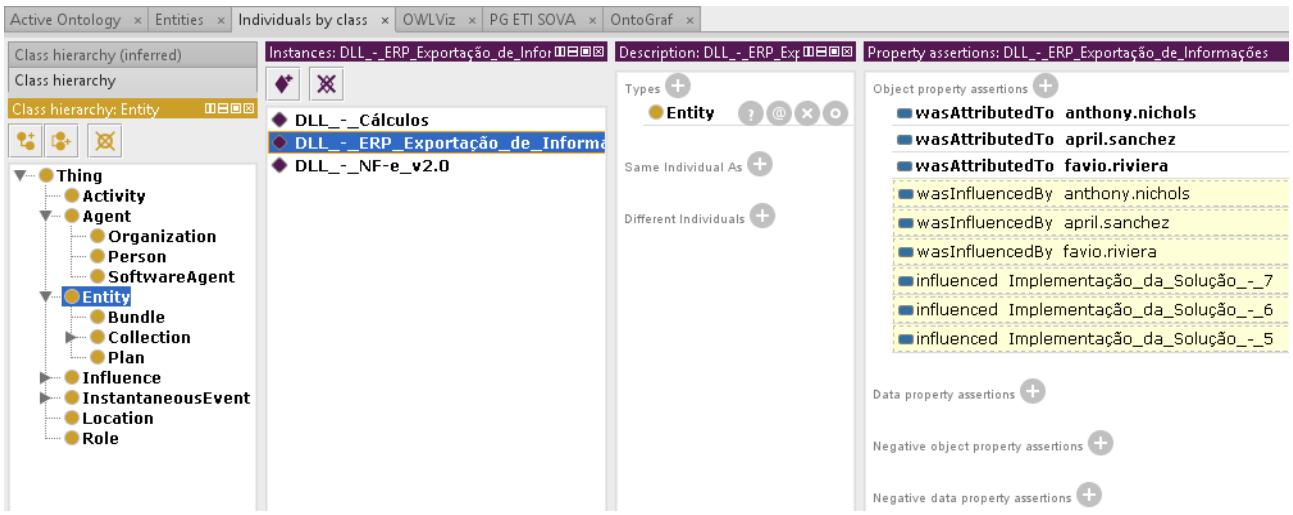


Figura 5.27: Inferência sobre a Entity 'DLL – ERP Exportação de Informações'

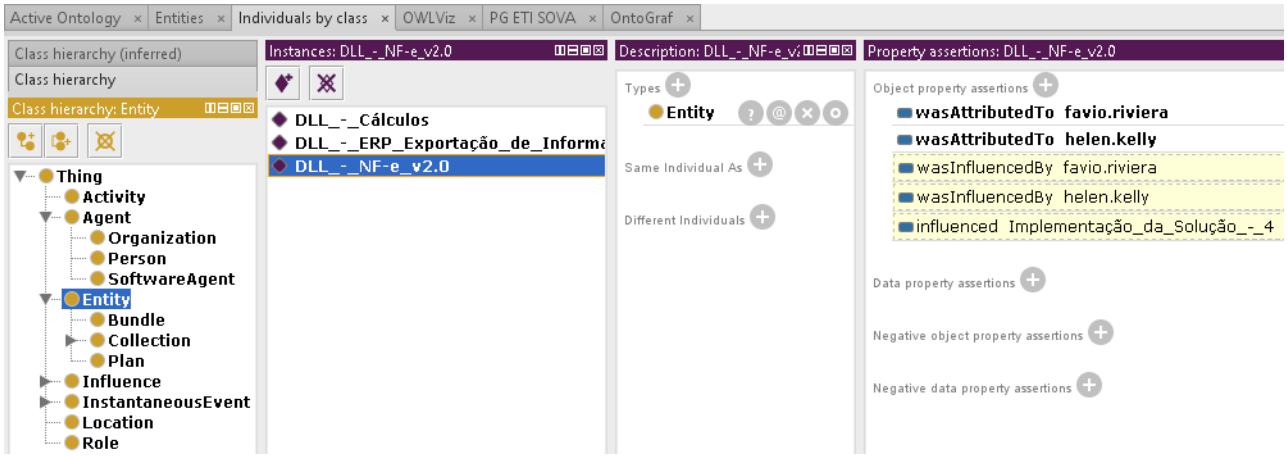


Figura 5.28: Inferência sobre a Entity 'DLL – NF-e v2.0'

Nesta seção, foram apresentados exemplos simples de informações que puderam ser obtidas por meio da especificação de regras utilizando o conceito de *Property Chains* (OWL, 2009) à ontologia utilizada. Durante o decorrer do desenvolvimento desta proposta de tese, toda a ontologia será avaliada, sendo analisada cada uma das entidades e relacionamentos do modelo PROV, com o objetivo de estabelecer novas regras que serão utilizadas para a descoberta de informações até então implícitas, que poderão auxiliar na melhoria do desempenho do processo.

5.4 Fase 4: Realimentação do Processo

Após a execução do processo e a coleta tanto das métricas previamente definidas como dos dados de proveniência e a possibilidade de análise do que ocorreu, de fato, durante a execução do processo, o responsável pelo gerenciamento do processo poderá observar possíveis melhorias e adequações no mesmo. Assim, esta proposta de trabalho prevê a implementação de uma fase de realimentação do processo que permitirá a realização de alterações no mesmo, possibilitando a geração de novas instâncias mais adequadas ao objetivo a ser atingido pela organização.

De acordo com o exemplo de processo apresentado nas seções anteriores deste capítulo para tratar alterações em um software, ao realimentar o processo, além das informações a respeito das métricas coletadas, as informações inferidas poderiam ser apresentadas ao gerente do processo ou ao responsável por criar novas instâncias deste processo. Estas informações podem ser apresentadas no momento da geração da nova instância, como observações a respeito dos membros da equipe que deveriam trabalhar de forma conjunta ou separada. Além disto, podem ser informadas tarefas a serem suprimidas do processo. Assim, estas informações, obtidas com base em dados de execuções anteriores, poderiam ser utilizadas durante a condução do processo e atribuição das tarefas aos

atores, visando otimizar o tempo total para execução do processo e/ou diminuir a quantidade de ‘ciclos’ ou repetições da tarefa relativa à implementação da solução.

Além da possibilidade de acesso às informações geradas/inferidas através dos dados de execução de processos, embora ainda não tenha sido definido em detalhes, espera-se que nesta fase o gerente do processo consiga alterar o processo principal, incluindo ou adaptando tanto tarefas (e seu respectivo fluxo), como papéis e artefatos do processo em questão.

5.5 Considerações sobre o Capítulo

Este Capítulo apresentou as quatro fases da proposta de execução, monitoramento e realimentação de processos, sendo elas: (1) Definição das medidas do processo; (2) Execução e monitoramento do processo; (3) Análise da execução do processo e (4) Realimentação do processo.

A proposta apresentada neste capítulo vem sendo refinada a partir dos exemplos reais da indústria que foram analisados e são apresentados no Capítulo 6. Dessa forma, as fases propostas foram criadas e encontram-se em evolução de acordo com conceitos presentes no referencial teórico e, principalmente, através de observações feitas durante a prova de conceito apresentada no próximo capítulo e do *feedback* recebido dos especialistas dos processos reais apresentados neste estudo.

Deve-se ressaltar que, no contexto da presente proposta de tese, a Fase 4: Realimentação do processo, ainda não foi descrita de forma detalhada e será trabalhada no decorrer dos próximos anos. Portanto, ainda não foi aplicada aos exemplos apresentados no próximo capítulo.

6. PROVA DE CONCEITO

Este capítulo apresenta uma prova de conceito (*Proof of Concept* - PoC), considerando a aplicação das três primeiras fases desta proposta de tese (apresentadas no Capítulo 5). A definição e o planejamento da PoC são apresentados na Seção 6.1. Para a avaliação, foram analisados dois processos da indústria em que foram aplicadas as três primeiras fases desta proposta de tese (apresentadas no Capítulo 5), obtendo, após isto, sugestões de informações que poderiam realimentar o processo principal, de forma a melhorar o desempenho das instâncias executadas a partir do mesmo. Estes dois processos de software são de empresas nacionais de pequeno porte.

O objetivo da PoC foi definido de acordo com a abordagem *Goal/Question/Metric* (GQM) (BASILI, 1992). Segundo a abordagem GQM, os objetivos devem ser definidos conforme o template a seguir:

- Analisar o <objeto do estudo> com a finalidade de <objetivo> com respeito à <foco da qualidade> do ponto de vista de <perspectiva> no contexto de <contexto>.

Assim, foi definido o seguinte objetivo para a PoC realizada:

- Analisar os dados de execução de processos com a finalidade de melhorar o desempenho do processo com respeito às métricas previamente definidas do ponto de vista do gerente do processo no contexto de processos de software.

Com base nesse objetivo, foi formulada a hipótese H_T que gerou a hipótese nula H_{T0} e a hipótese alternativa H_{T1} apresentadas a seguir:

- HT1: O armazenamento e posterior análise de dados de execução de processos de software utilizando um modelo de proveniência de dados em conjunto com uma ontologia é capaz de oferecer informações estratégicas a serem utilizadas para a melhoria do desempenho das próximas instâncias deste processo.
- HT0: O armazenamento e posterior análise de dados de execução de processos de software utilizando um modelo de proveniência de dados em conjunto com uma ontologia **NÃO** é capaz de oferecer informações estratégicas a serem utilizadas para a melhoria do desempenho das próximas instâncias deste processo.

O restante deste capítulo apresenta o planejamento (Seção 6.1) e a execução da prova de conceito (Seção 6.2), além da análise das hipóteses definidas anteriormente (Seção 6.3) e uma análise qualitativa dos resultados obtidos (Seção 6.4). Também são apresentadas as limitações encontradas (Seção 6.5), ameaças à validade (Seção 6.6), e, por fim, tem-se as considerações finais do capítulo.

6.1 Planejamento

Para a realização da PoC, foram utilizados dois processos da indústria. O primeiro trata-se de um processo para requisição e gerenciamento de mudanças nos softwares desenvolvidos por uma empresa brasileira de desenvolvimento de software de gestão empresarial, há 19 anos no mercado brasileiro [CEOSOFTWARE, 2015]. Já o segundo, trata-se de um processo para solicitação e implementação de novos recursos e tratamento de erros em um ERP Contábil da Projetus Tecnologia da Informação Ltda, uma empresa de desenvolvimento de software brasileira, de pequeno porte [PROJETUS, 2015]. Estes processos foram obtidos e modelados através de entrevistas/reuniões e análise de documentação enviada sobre o mesmo.

Para a obtenção do primeiro processo apresentado neste capítulo, foram realizadas diversas reuniões com um aluno de mestrado que estava realizando sua pesquisa na empresa citada e que tinha conhecimento do processo descrito. Estas reuniões tiveram início em setembro de 2014. O processo modelado foi avaliado e aprovado pelo gerente de projetos da empresa em fevereiro de 2015.

Para a obtenção do segundo processo, o gerente de desenvolvimento da empresa enviou-me o fluxo de atividades seguidas no processo em um documento e este foi analisado e remodelado para a notação para modelagem de processos utilizada nesta proposta de tese (BPMN). As dúvidas com relação ao processo especificado no documento enviado foram sanadas pessoalmente, com o próprio gerente, ou via e-mail. Tanto o início das negociações para obtenção deste processo quanto a finalização e a aprovação do mesmo pelo gerente do processo ocorreram em junho de 2015.

6.2 Execução

Para a condução da PoC, foram utilizadas as especificações dos processos anteriormente descritos para a criação de um modelo de fluxo do mesmo, utilizando a notação BPMN (Seção 2.3). O detalhamento completo da aplicação das três primeiras fases desta proposta de tese a cada um destes dois processos é apresentado nas subseções a seguir.

6.2.1 Processo 1

Com base nas especificações obtidas sobre o processo para requisição e gerenciamento de mudanças, foram criados os modelos de fluxo deste processo (Figuras 6.1 a 6.4). A Figura 6.1 apresenta a descrição das atividades e tarefas, bem como o fluxo destas, além dos papéis responsáveis por sua execução. Neste processo, são definidas três atividades ou subprocessos

(*Descrição Owner*, *Descrição Técnica* e *Descrição Testes*), cujo detalhamento é apresentado nas Figuras 6.2, 6.3 e 6.4.

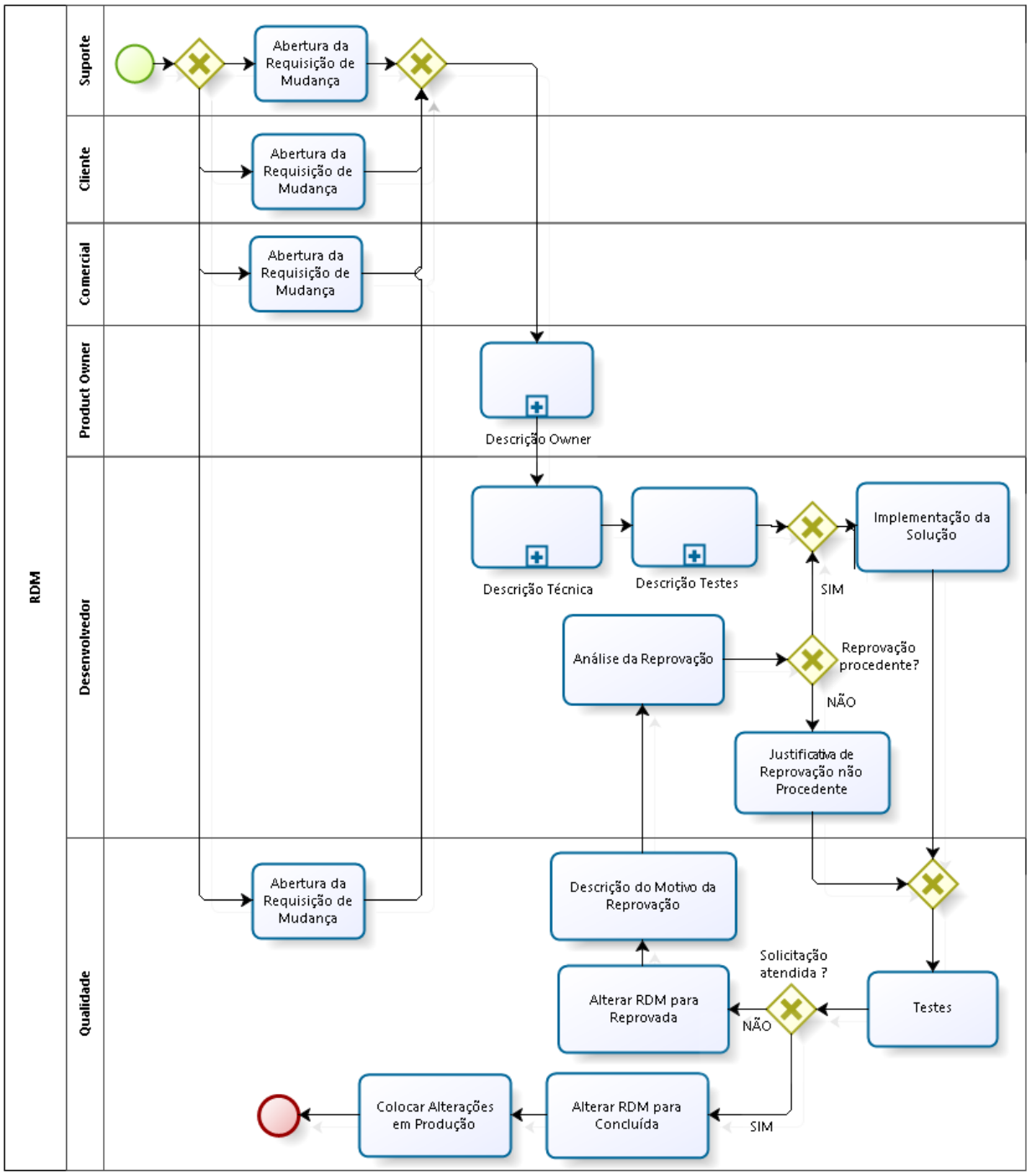


Figura 6.1: Processo Principal para requisição e gerenciamento de mudanças no software

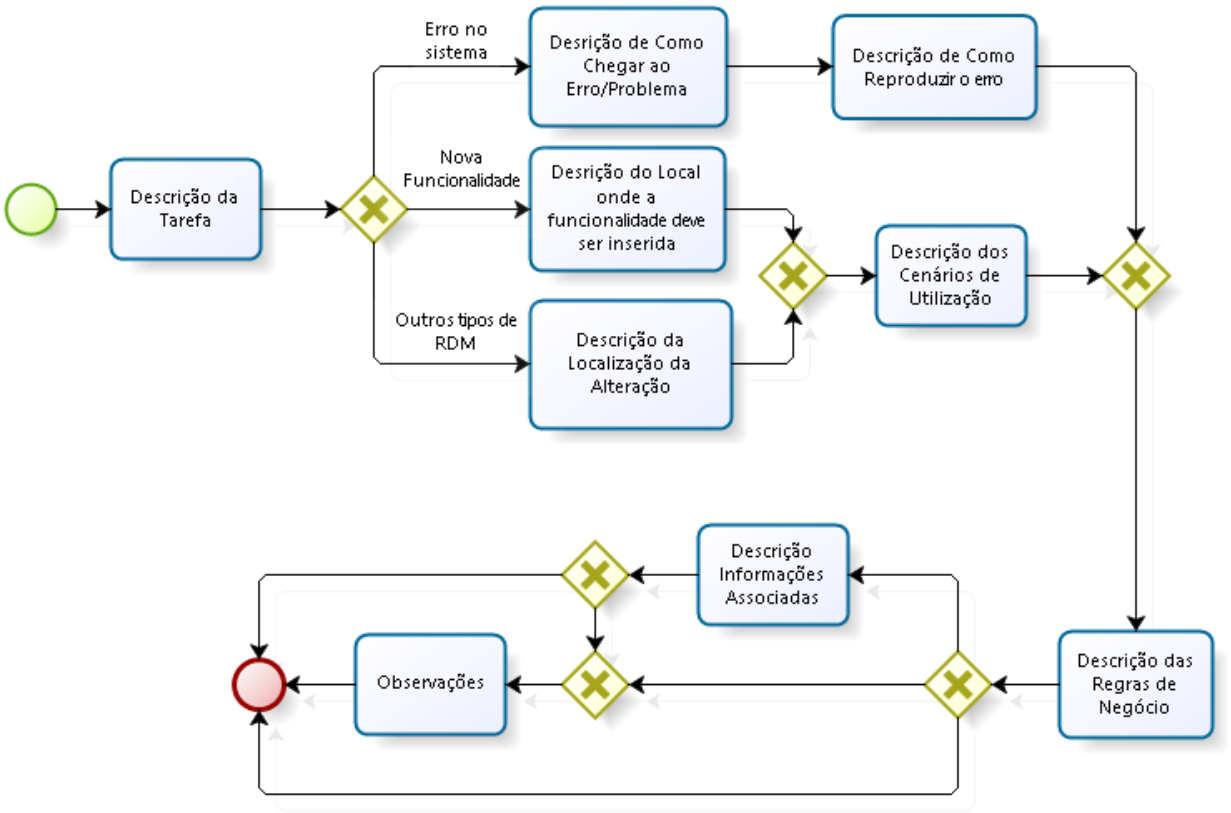


Figura6.2: Subprocesso 'Descrição Owner'

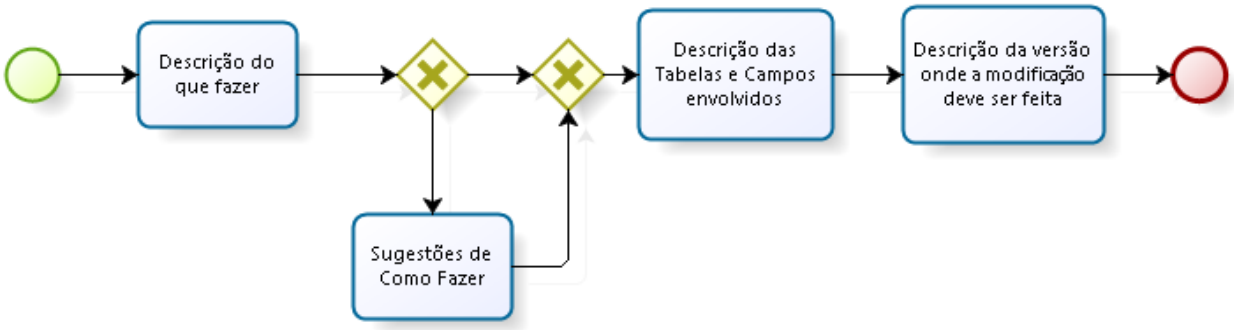


Figura6.3: Subprocesso 'DescriçãoTécnica'

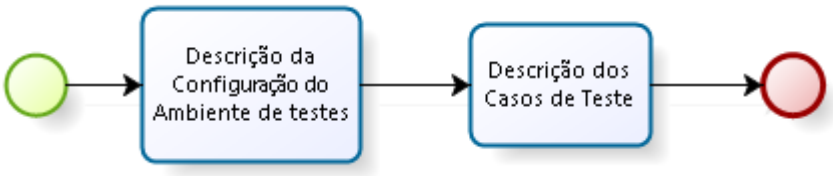


Figura6.4: Subprocesso 'Descrição Testes'

Após a definição dos modelos apresentados anteriormente, foram desenvolvidas as fases propostas no Capítulo 5, conforme descrito nas subseções a seguir.

6.2.1.1 Fase 1: Definição das medidas do processo a serem monitoradas

Para o processo de requisição e gerenciamento de mudanças nos softwares desenvolvidos, proposto nesta seção, foram estabelecidas três medidas a serem coletadas e armazenadas durante a fase de Execução e Monitoramento do Processo, sendo elas: (1) Tempo de duração da execução do processo, (2) Tempo médio de duração das execuções do processo, e (3) Porcentagem de execuções do processo que foram criadas a partir de uma RDM anterior (desdobramento). Estas medidas foram definidas conforme proposto na Seção 5.1 e são detalhadas a seguir (vide Tabelas 6.1 a 6.3).

Tabela 6.1: Definição da Medida de Tempo de Execução do Processo (TEP)

1	**Nome:**	Tempo de Execução do Processo
2	**Definição**	Medida utilizada para quantificar a duração da execução do processo de Requisição de Mudanças.
3	**Mnemônico:**	TEP
4	**Tipo de Medida:**	Medida Base
5	**Entidade Medida:**	Processo de Requisição de Mudanças
6	**Propriedade Medida:**	Tempo
7	**Unidade de Medida:**	Horas
8	**Tipo de Escala:**	Absoluta
9	**Valores da Escala:**	Números reais positivos, utilizando-se uma precisão de duas casas decimais.
10	**Intervalo Esperado dos Dados:**	-
11	**Fórmula de Cálculo de Medida:**	TEP = TF – TI Onde: TF = tempo final (data e hora de finalização) do processo; TI = tempo inicial (data e hora de inicialização) do processo.
12	**Procedimento de Medição:**	Calcular o tempo de duração da execução do processo, usando a fórmula de cálculo da medida.
13	**Momento da Medição:**	Ao iniciar e finalizar qualquer execução de uma instância do processo.
14	**Periodicidade de Medição**	Sempre que for inicializada e finalizada uma execução do processo.
15	**Responsável pela Medição:**	Será realizada de forma automatizada (sistema).
16	**Procedimento de Análise:**	- Representar em um gráfico os valores coletados para a medida comparando-a com outras execuções do mesmo processo, de forma a possibilitar a comparação destes. Assim, será possível analisar se a medida do processo em análise apresenta o mesmo comportamento da mesma medida em execuções anteriores. Caso isto não ocorra, será necessário

		investigar as causas da instabilidade no comportamento do processo e identificar ações corretivas, quando pertinente.
17	**Momento da Análise de Medição:**	Após a execução do processo.
18	**Periodicidade da Análise:**	Após a execução de cada instância do processo.
19	**Responsável pela Análise:**	Gerente do Projeto

Tabela 6.2: Definição da Medida de Tempo Médio de Execução do Processo (TMEP)

1	**Nome:**	Tempo Médio de Execução do Processo
2	**Definição**	Medida utilizada para quantificar o valor médio da duração da execução dos processos de Requisição de Mudanças.
3	**Mnemônico:**	TMEP
4	**Tipo de Medida:**	Medida Derivada
5	**Entidade Medida:**	Processo de Requisição de Mudanças (RDM)
6	**Propriedade Medida:**	Tempo
7	**Unidade de Medida:**	Horas
8	**Tipo de Escala:**	Absoluta
9	**Valores da Escala:**	Números reais positivos, utilizando-se uma precisão de duas casas decimais.
10	**Intervalo Esperado dos Dados:**	-
11	**Fórmula de Cálculo de Medida:**	TMEP = STEP / TPE Onde: STEP = Somatório do Tempo de Execução de Processo de todas as execuções do processo de RDM; TEP = total de execuções do processo de RDM.
12	**Procedimento de Medição:**	Calcular o tempo médio de duração da execução do processo, usando a fórmula de cálculo da medida.
13	**Momento da Medição:**	Após a execução do processo.
14	**Periodicidade de Medição**	Sempre que for finalizada uma execução do processo.
15	**Responsável pela Medição:**	Será realizada de forma automatizada (sistema).
16	**Procedimento de Análise:**	- Representar em um gráfico os valores coletados para a medida, exibindo a variação da mesma ao longo das execuções do processo de RDM, de forma a possibilitar o comportamento da mesma e identificar ações corretivas, quando pertinente.
17	**Momento da Análise de Medição:**	Após a execução do processo.
18	**Periodicidade da Análise:**	Após a execução de cada instância do processo.
19	**Responsável pela Análise:**	Gerente do Projeto

Tabela 6.3: Definição da Medida de Porcentagem de Desdobramentos (PD)

1	**Nome:**	Porcentagem de Desdobramentos
2	**Definição**	Medida utilizada para quantificar a porcentagem de execuções do processo que foram criadas a partir de uma RDM anterior.
3	**Mnemônico:**	PD

4	**Tipo de Medida:**	Medida Base
5	**Entidade Medida:**	Processo de Requisição de Mudanças que são desdobramentos
6	**Propriedade Medida:**	Desdobramento
7	**Unidade de Medida:**	Porcentagem de execuções que são desdobramento
8	**Tipo de Escala:**	Absoluta
9	**Valores da Escala:**	Números inteiros.
10	**Intervalo Esperado dos Dados:**	-
11	**Fórmula de Cálculo de Medida:**	MD = (SD / TPE)*100 Onde: SD = Somatório das execuções que são desdobramentos TPE = total de processos executados
12	**Procedimento de Medição:**	Calcular a porcentagem de execuções do processo que foram criadas a partir de uma RDM anterior, usando a fórmula de cálculo da medida.
13	**Momento da Medição:**	Ao iniciar qualquer execução de uma instância do processo.
14	**Periodicidade de Medição**	Sempre que for inicializado uma execução do processo.
15	**Responsável pela Medição:**	Será realizada de forma automatizada (sistema).
16	**Procedimento de Análise:**	- Representar em um gráfico os valores coletados para a medida, exibindo a variação da mesma ao longo das execuções do processo de RDM, de forma a possibilitar o comportamento da mesma e identificar ações corretivas, quando pertinente.
17	**Momento da Análise de Medição:**	Durante a atividade de análise das execuções do processo.
18	**Periodicidade da Análise:**	Semanal.
19	**Responsável pela Análise:**	Gerente do Projeto

6.2.1.2 Fase 2: Execução e monitoramento do processo

O processo apresentado nesta seção não foi executado e monitorado de acordo com esta proposta de tese, ou seja, não foram utilizados os mecanismos/ferramentas para execução e monitoramento do processo sugeridos no Capítulo 5. Porém, conseguiu-se junto a empresa responsável pelo processo descrito, dados referentes a execuções já finalizadas deste processo e, portanto, foram utilizados estes dados para a realização desta fase. Desta forma, na Fase 1 (Seção 6.2.1.1) foram criadas medidas que eram possíveis de serem obtidas ou calculadas com base nos dados fornecidos. Assim, a proveniência retrospectiva dos processos executados foi armazenada também de acordo com estes dados obtidos, utilizando a abordagem PROV-Process.

Deve-se ressaltar que, para este primeiro processo, não foram fornecidos os dados de execução de todo o processo e sim, apenas uma parte do mesmo. Destes dados obtidos, foram

analisadas 10 instâncias de execução do processo que foram completamente concluídas[9]. Com relação aos dados obtidos, foram utilizados os seguintes:

- Número da RDM;
- Informação se a RDM Aberta é um desdobramento de uma outra RDM anterior, ou seja, se ela foi criada a partir de uma RDM anterior;
- Data e Hora de Abertura da RDM;
- Tipo da RDM;
- Responsável pela Abertura da RDM (Origem);
- Módulos e Componentes alterados durante a tarefa de Implementação da Solução;
- Equipe responsável pela Implementação da Solução;
- Situação da RDM;
- Data de Conclusão da RDM; e
- Hora de Conclusão da RDM.

Estes dados foram obtidos através de uma planilha enviada pela empresa responsável pelo projeto. Um exemplo de como estes dados foram fornecidos pode ser visualizado nas Tabelas 6.4 e 6.5. Cada linha desta tabela representa uma execução distinta de uma instância do processo para requisição e gerenciamento de mudanças.

Tabela 6.4: Exemplo de Dados e execução – Processo 1 – Parte 1

NumRDM	Desd	DataAbertura	HoraAbertura	Tipo	Origem
30006	0	03/10/2013	14:54:00	Liberação de Módulos	Cliente
30006	1	11/06/2014	17:18:00	Liberação de Módulos	Cliente

Tabela 6.5: Exemplo de Dados e execução – Processo 1 – Parte 2

NumRDM	ModuloRDM	Modulo	Componente	Equipe	SituacaoRDM	DataConc	HoraConc
30006	Financeiro	DLL - ERP PDA	clsValidacao	VB6	Concluída	03/10/2013	22:06:00
30006	Financeiro	DLL - ERP PDA	clsValidacao	VB6	Concluída	12/06/2014	10:41:00

Estes dados foram importados para o repositório relacional da base PROV-Process de acordo com os seguintes critérios:

[9]Foi fornecido um número bem maior de execuções de instâncias do processo, porém, como toda a manipulação de dados de execução de processos por meio da abordagem PROV-Process ainda não está completamente automatizada, tornar-se-ia muito custosa a transformação manual de todos os dados de execução e posterior importação dos mesmos na ontologia desta abordagem. Todos os dados utilizados para este exemplo podem ser visualizados em http://gabriellacastro.com.br/dsc/ex1/ex1.xlsx.

- Para todas as execuções cujos dados foram analisados, foram criados três registros na tabela *Activity*, com os respectivos nomes:
    - Abertura da Requisição de Mudança;
    - Implementação da Solução;
    - Alterar RDM para Concluída.
- O número da RDM ao qual a execução estava associada foi inserido como um atributo de cada uma das atividades anteriores, utilizando as tabelas *Attribute* e *Activity*_Attribute.
- Se uma determinada execução corresponde ao desdobramento de uma RDM anterior, foi criado um registro na tabela *WasInfomedBy*.
- A data e a hora de abertura da RDM foram incluídas através do atributo *startTime* da atividade de Abertura da Requisição de Mudança.
- O tipo da RDM ao qual a execução estava associada foi inserido como um atributo da atividade Abertura da Requisição de Mudança, utilizando as tabelas *Attribute* e *Activity_Attribute*.
- O papel responsável pela atividade Abertura da Requisição de Mudança, campo Origem dos dados de execução recebidos, foram inseridos como registros da tabela *Agent*, utilizando o campo *Name* e o tipo de agente *Person*.
- O relacionamento entre a atividade Abertura da Requisição de Mudança e o responsável pela mesma foi inserido como registro da tabela *WasAssociatedWith*.
- Os valores fornecidos como Módulo, Módulo RDM e Componente foram incluídos como registros da tabela *Entity* e foram associados à tarefa Implementação da Solução criando-se registros na tabela *Used*.
- Os valores fornecidos como Módulo, Módulo RDM e Componente, incluídos como registros da tabela *Entity*também foram associados aos agentes que o manipularam criando-se registros na tabela *WasAttributedTo*.
- O papel responsável pela atividade Implementação da Solução, campo Equipe dos dados de execução recebidos, foram inseridos como registros da tabela *Agent*, utilizando o campo *Name* e o tipo de agente *Person*.
- O relacionamento entre a atividade Implementação da Solução e o responsável pela mesma (campo Equipe dos dados de execução recebidos) foi inserido como registro da tabela *WasAssociatedWith*.

- Os campos data e hora de conclusão da RDM, informados nos dados de execução recebidos, foram inseridos utilizando o atributo *endTime* da atividade Alterar RDM para Concluída.
- Como no modelo de fluxo deste exemplo de processo o papel responsável pela tarefa Alterar RDM para Concluída é a Qualidade, este papel foi inserido como registro da tabela *Agent* e associado a esta tarefa, inserindo-o e registrando-o na tabela *wasAssociatedWith*.
- De forma a identificar a qual instância da execução do processo uma determinada atividade está associada, foi inserido um atributo relacionado ao id da instância (*processInstanceId*) a todas as tarefas, utilizando as tabelas *Attribute* e *Activity_Attribute*.

Um esquema de como estes dados foram importados pode ser visualizado nas Figuras 6.5 a 6.7, que ilustram como os dados das Tabelas 6.4 e 6.5 foram inseridos na base de dados relacional da abordagem PROV-Process.

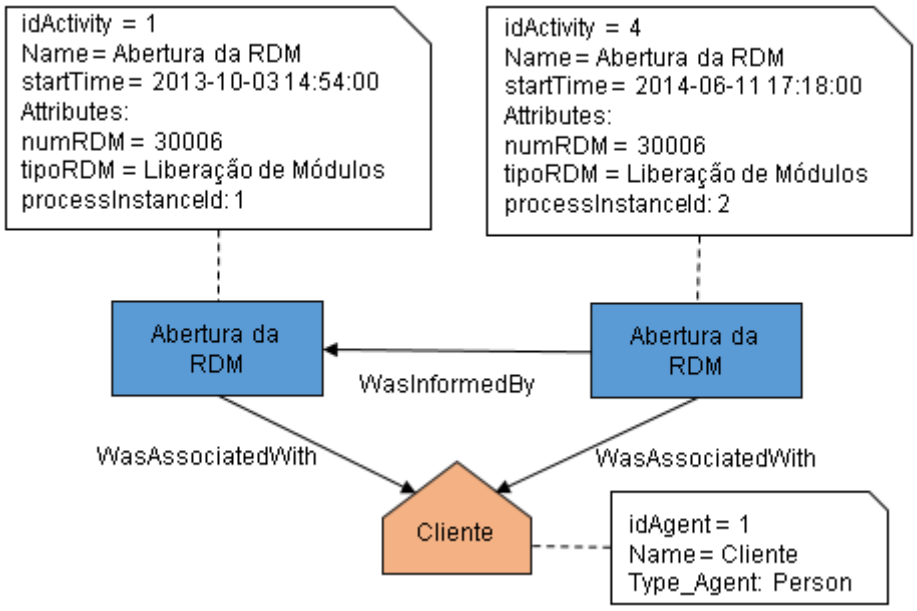


Figura 6.5: Esquema para Importação da Activity 'Abertura da RDM'

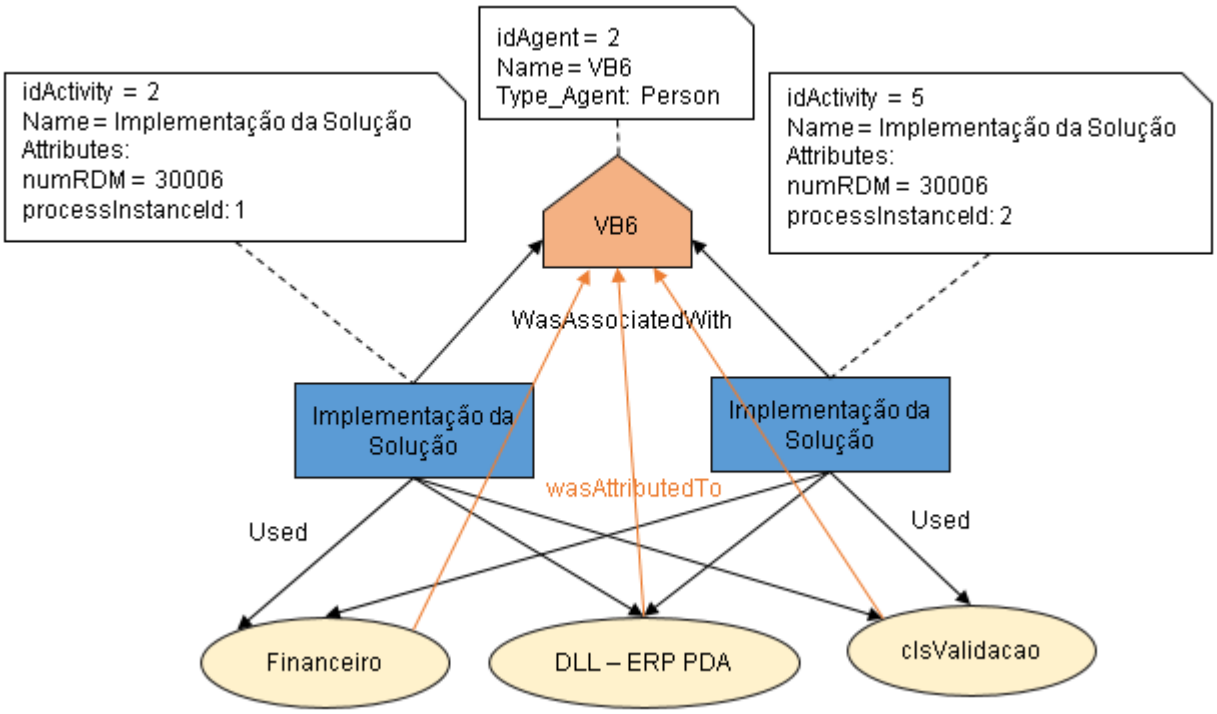


Figura 6.6: Esquema para Importação da Activity 'Implementação da Solução'

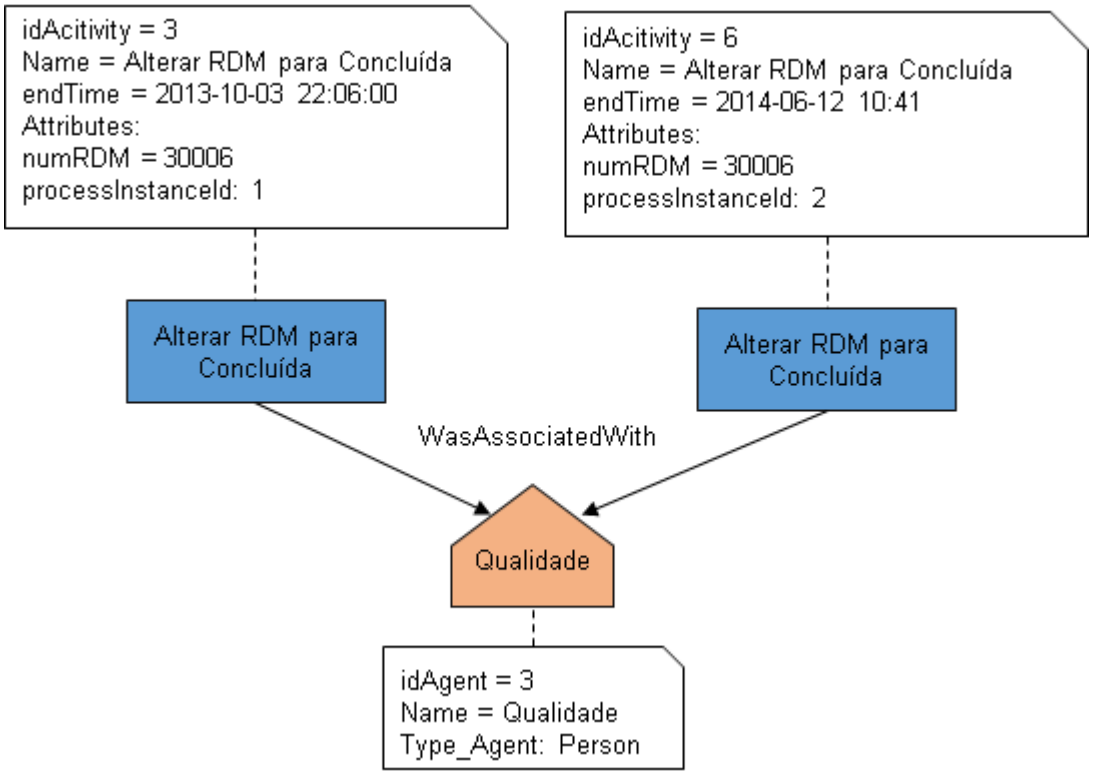


Figura 6.7: Esquema para Importação da Activity 'Alterar RDM para Concluída'

6.2.1.3 Fase 3: Análise da execução do processo

Durante esta etapa, as três medidas definidas na Fase 1 foram analisadas, com base nos dados de execução do processo, o que permite ao responsável pelo gerenciamento do processo a verificação dos valores obtidos e uma comparação destes com outras execuções do mesmo processo. Esta análise é descrita a seguir, para cada uma das três medidas:

- **Medida 1: Tempo de Execução do Processo**

Para obtenção desta medida, bastou-se subtrair do tempo final (campo endTime da última tarefa do processo, Alterar RDM para Concluída, o valor do campo startTime da primeira tarefa do processo, Abertura da RDM). Estes resultados podem ser obtidos pelas consultas realizadas na base de dados da abordagem PROV-Process. Os dados obtidos para esta medida, para as 10 instâncias executadas são listadas abaixo, assim como o gráfico gerado a partir das mesmas (Figura 6.8).

- Instância 1: 7,20 horas
- Instância 2: 17,38 horas
- Instância 3: 102,63 horas
- Instância 4: 101,98 horas
- Instância 5: 927,98 horas
- Instância 6: 23,85 horas
- Instância 7: 170,00 horas
- Instância 8: 72,25 horas
- Instância 9: 1321,60 horas
- Instância 10: 58,42 horas

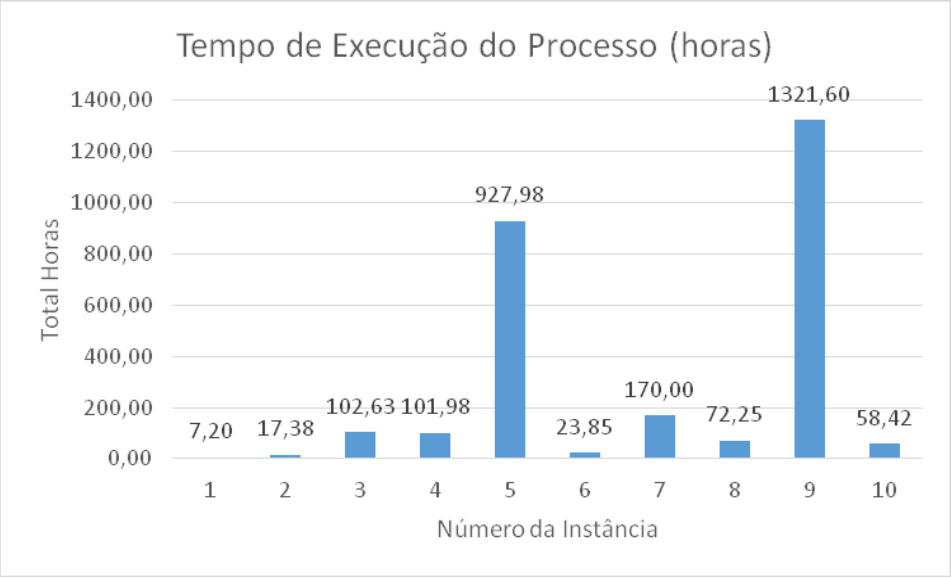


Figura 6.8: Gráfico da Medida de Tempo de Execução do Processo

- **Medida 2: Tempo Médio de Execução do Processo**

Para a obtenção desta medida, que é calculada após a execução de cada instância, foi realizado o somatório do tempo de execução de processo (Medida 1) de todas as execuções do processo de RDM, dividido pelo número de execuções deste processo. Os dados obtidos para esta medida, após cada uma das 10 instâncias executadas, são listados abaixo, assim como o gráfico gerado a partir das mesmas (Figura 6.9).

  - Instância 1: 7,20 horas
  - Instância 2: 12,29 horas
  - Instância 3: 42,40 horas
  - Instância 4: 57,30 horas
  - Instância 5: 231,43 horas
  - Instância 6: 196,84 horas
  - Instância 7: 193,00 horas
  - Instância 8: 177,91 horas
  - Instância 9: 304,99 horas
  - Instância 10: 280,33 horas

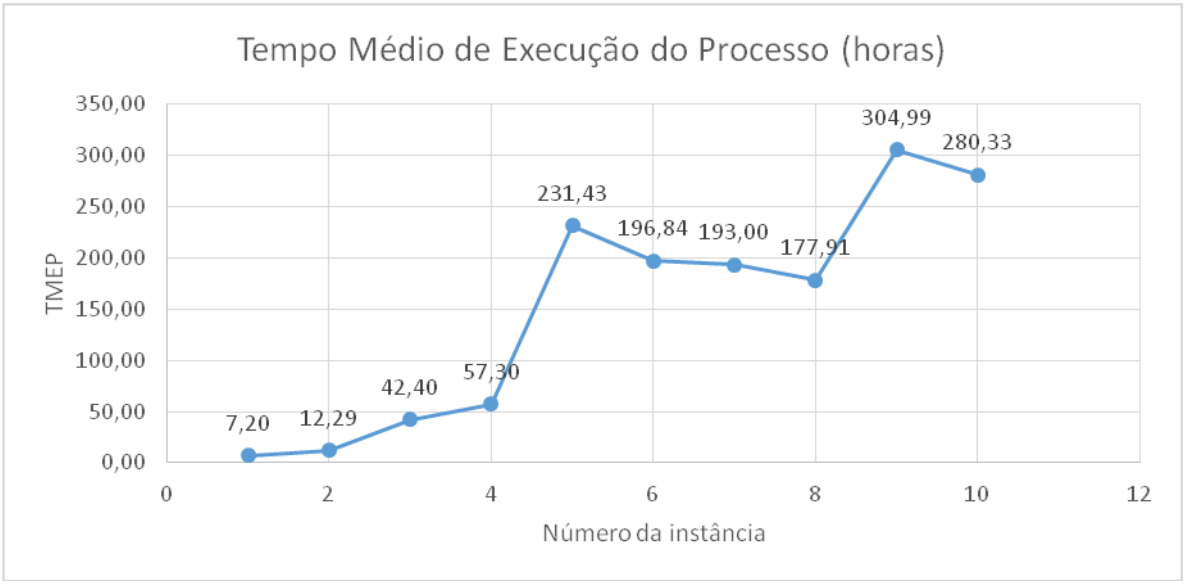


Figura 6.9: Gráfico da Medida de Tempo Médio de Execução do Processo

- **Medida 3: Porcentagem Desdobramentos**

Para obtenção desta medida, foram avaliadas as tarefas executadas e suas respectivas propriedades (*numRDM*e *processInstanceId*) e, tendo estas tarefas o mesmo *numRDM* e *processInstanceId* diferentes, adotou-se que a instância com maior valor para o campo *processInstanceId*corresponde a um desdobramento. Os dados obtidos para esta medida, após cada

uma das 10 instâncias executadas são listadas abaixo, assim como o gráfico gerado a partir das mesmas (Figura 6.10).

- Instância 1: (0/1)*100 = 0%
- Instância 2: (1/2)*100 = 50%
- Instância 3: (1/3)*100 = 33%
- Instância 4: (2/4)*100 = 50%
- Instância 5: (2/5)*100 = 40%
- Instância 6: (3/6)*100 = 50%
- Instância 7: (3/7)*100 = 43%
- Instância 8: (4/8)*100 = 50%
- Instância 9: (4/9)*100 = 44%
- Instância 10: (5/10)*100 = 50%

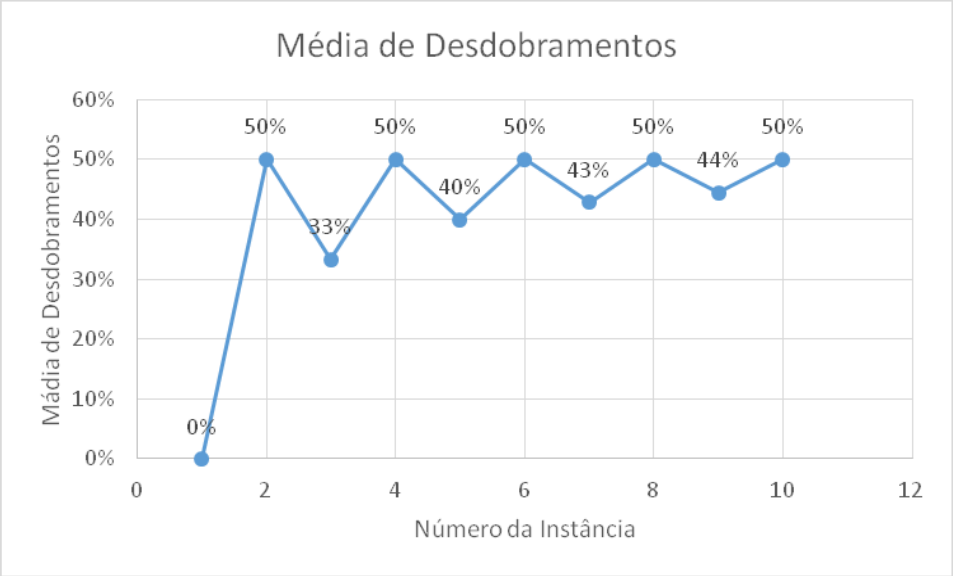


Figura 6.10: Gráfico da Medida de Média de Desdobramentos

Além dos valores obtidos para as medidas definidas na Fase 1 desta proposta, através do mecanismo de inferência utilizado na ontologia, que foi populada com indivíduos criados a partir dos dados de execução do processo, é possível a derivação de informações estratégicas, a respeito da execução do processo. Assim, para que isto fosse possível, os dados de execução do processo foram exportados da base de dados do PROV-Process e importados para a ontologia desta mesma abordagem[10]. Como informações inferidas a partir dos dados de proveniência retrospectiva deste processo, podem-se destacar quatro tipos, sendo elas:

[10] A ontologia gerada para este exemplo encontra-se disponível em: http://gabriellacastro.com.br/dsc/ex1/ex1.owl.

1) Tarefas que influenciaram a geração de outras tarefas, ou seja, conforme pode ser visualizado na primeira marcação em vermelho da Figura 6.11, a tarefa Abertura da Requisição de Mudança (id = 1) influenciou a tarefa Abertura da Requisição de Mudança (id = 4). Este mesmo tipo de informação também foi inferida para as tarefas deste mesmo tipo com ids iguais a 7, 13, 19 e 25.

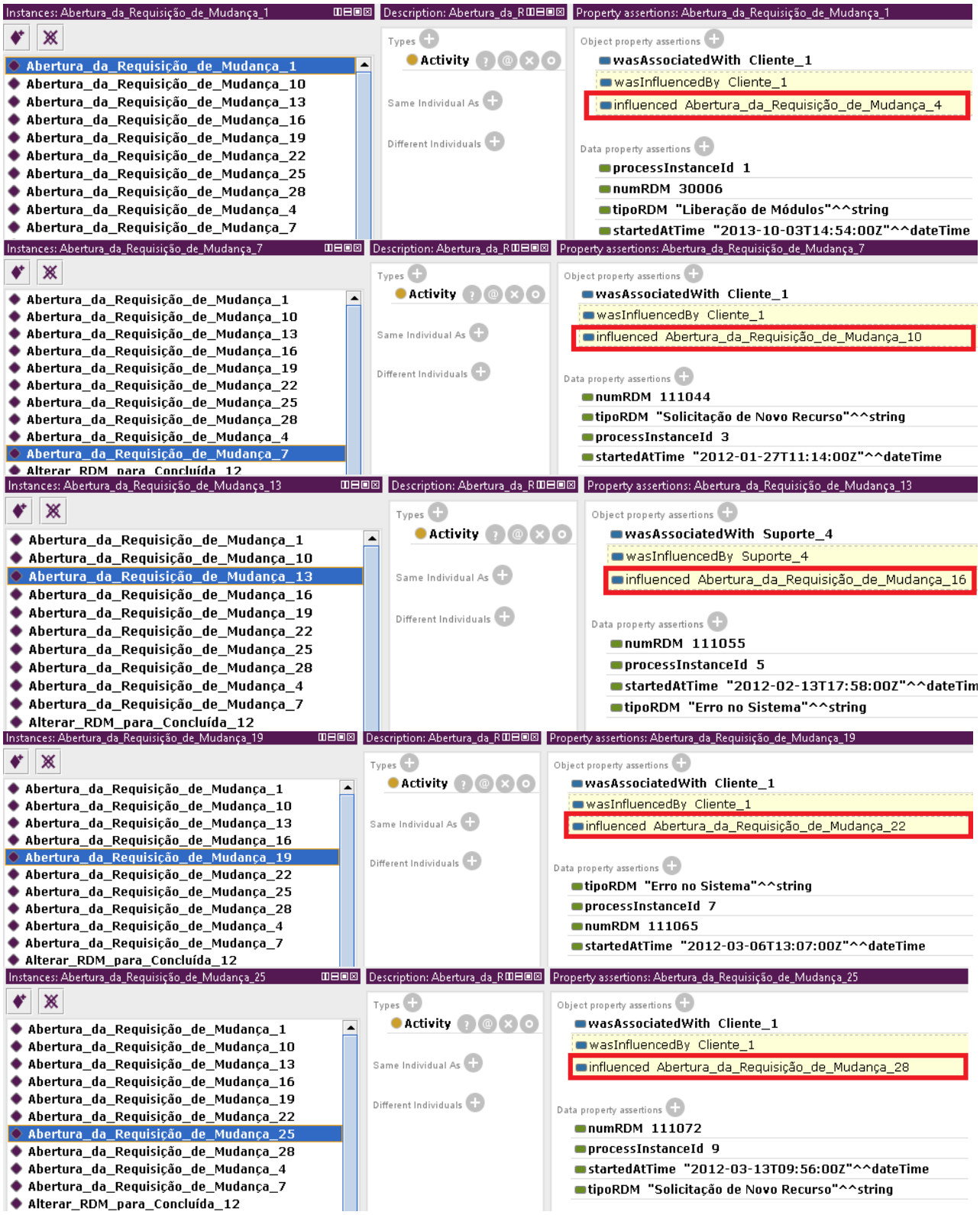


Figura 6.11: Tarefas que influenciaram outras tarefas

2) Agentes que poderiam estar associados à tarefa de Implementação da Solução, haja vista que já manipularam os artefatos envolvidos nesta tarefa em alguma outra execução do processo. A Figura 6.12, apresenta, por exemplo, que a atividade Implementação da Solução (id = 11) foi influenciada pelo agente DotNet (id = 5), haja vista que este agente manipulou artefatos comuns a esta tarefa em outras instâncias deste processo.

Este mesmo tipo de informação (agentes que poderiam estar associados à tarefa de Implementação da Solução) também ocorre para as tarefas de Implementação da Solução com id's iguais a 8, 20, 23, 26 e 29.

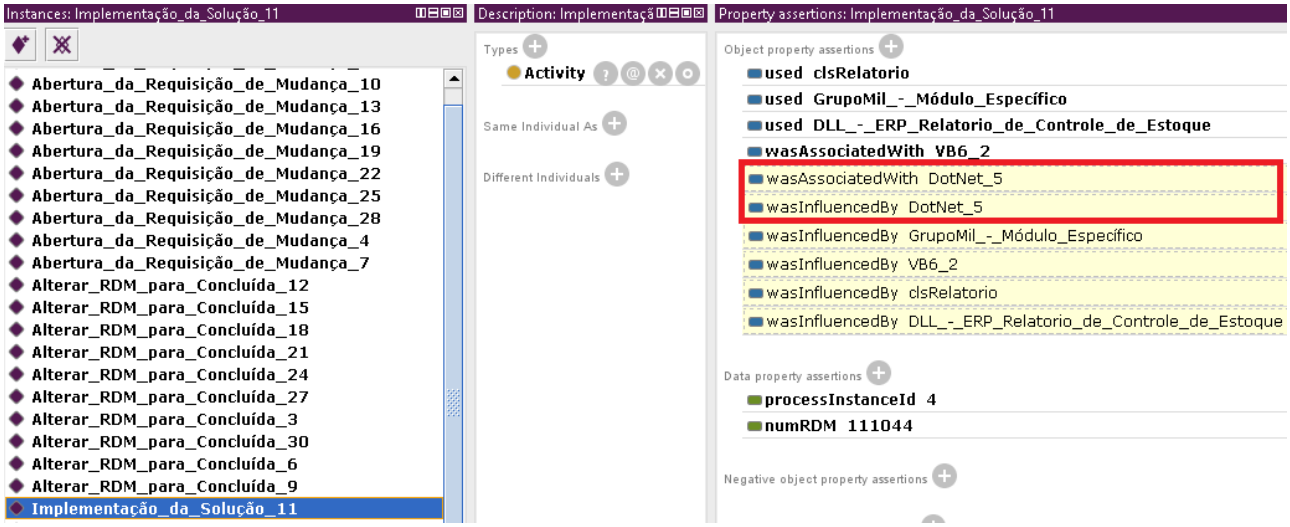


Figura 6.12: Agentes que influenciaram uma tarefa

3) Listagem de todas as tarefas em que um agente estava envolvido, bem como os artefatos manipulados pelo mesmo. Embora este tipo de informação possa ser obtido por consultas na própria base de dados da abordagem PROV-Process, com a utilização da ontologia e da máquina de inferência, este tipo de informação pode ser obtida mais facilmente (com uma simples busca em SPARQL), conforme pode ser visualizado na Figura 6.13.

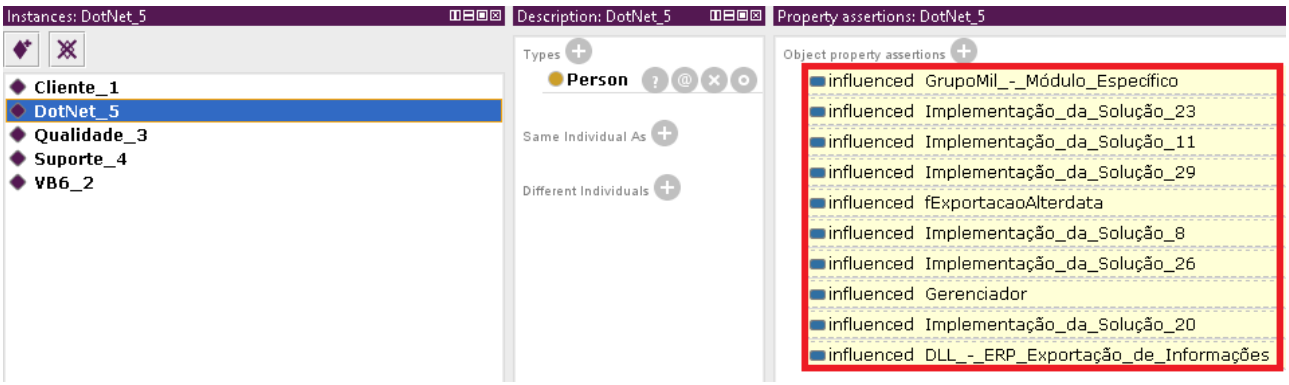


Figura 6.13: Tarefas e Artefatos manipulados pelo Agente DotNet

4) Listagem de todas as tarefas em que um determinado artefato foi utilizado. Embora este tipo de informação possa ser obtido por meio de consultas na própria base de dados da abordagem PROV-Process, com a utilização da ontologia e da máquina de inferência, este tipo de informação pode ser obtida mais facilmente (com uma simples busca em SPARQL), conforme pode ser visualizado na Figura 6.14.

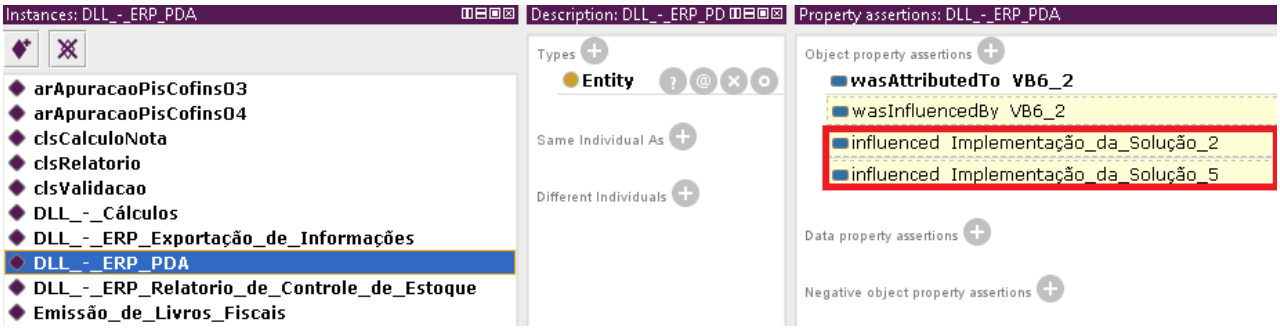


Figura 6.14: Tarefas em que um determinado artefato foi manipulado

6.2.1.4 Fase 4: Realimentação do Processo

As informações inferidas a partir do uso da ontologia em conjunto com um modelo de proveniência de dados poderiam auxiliar na melhoria do desempenho do processo em questão. Por exemplo, ao analisar-se os resultados obtidos para as medidas 1 e 2 (Tempo de Execução do Processo e Tempo Médio de Execução do Processo), observa-se que as execuções das instâncias 5 e 9, elevaram consideravelmente os valores obtidos até então, ou seja, apresentaram um tempo de execução muito superior quando comparado com as demais execuções do mesmo processo. Assim, estas duas execuções necessitariam de uma análise posterior de forma a evitar que novas execuções deste processo tenham tempo de execuções elevados como estes. Neste caso, ao analisar estas duas execuções por meio da ontologia e dos dados inferidos, poder-se-ia realimentar o processo com as seguintes informações:

1) Ao avaliar as atividades e agentes envolvidos na execução da instância 5, não foram inferidas nenhuma informação relevante, porém, ao analisar os artefatos manipulados durante a execução desta instância de processo, verificou-se que os artefatos ‘arApuracaoPisCofins03’ e ‘frelApuracaoPisCofins’ foram manipulados apenas pela tarefa Implementação da Solução (id=14) desta instância de execução do processo (vide Figuras 6.15 e 6.16). Assim, o processo poderia ser realimentado com uma anotação junto à tarefa de Implementação da Solução que a manipulação dos artefatos ‘arApuracaoPisCofins03’ e ‘frelApuracaoPisCofins’ nesta tarefa poderá acarretar num aumento considerável no tempo de execução do processo.

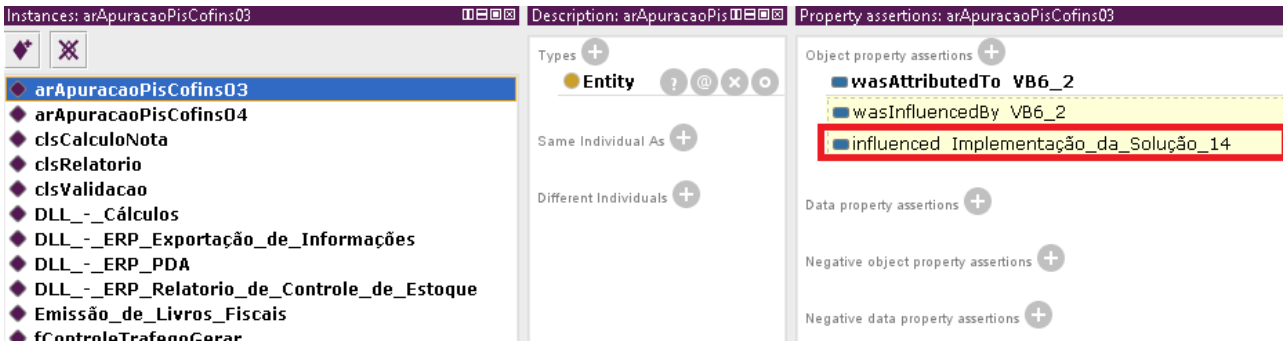


Figura6.15: Artefato 'arApuracaoPisCofins03'

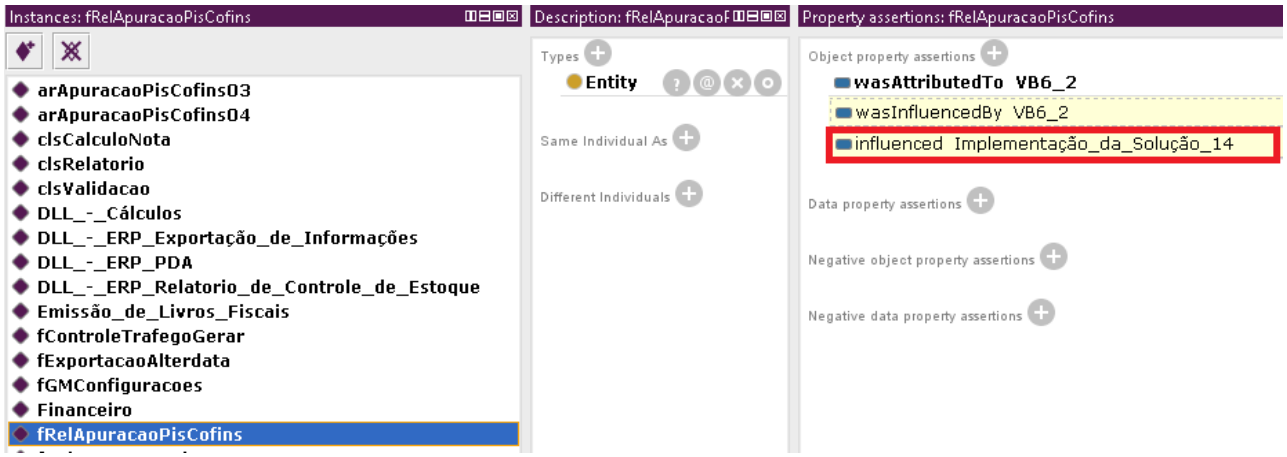


Figura 6.16: Artefato 'fRelApuracaoPisCofins'

2) Ao avaliar a atividade de Implementação da Solução realizada na execução da instância 9, foi inferida a informação de que o Agente VB_6 (id=2) influenciou esta tarefa (vide Figura 6.17), haja vista que já manipulou os artefatos envolvidos na mesma em alguma outra execução do processo, o que poderia, talvez, influenciar na redução do tempo de execução do processo. Assim, o processo poderia ser realimentado com anotações junto aos artefatos de agentes que já o manipularam. Desta forma, durante a execução de um processo, ao iniciar a manipulação de um determinado artefato, o executor da tarefa poderia incluir novos agentes para a solução daquela tarefa, uma vez que os mesmos já utilizaram aquele artefato em alguma execução anterior e, portanto, poderiam compartilhar algum conhecimento relativo ao mesmo, o que, possivelmente, poderia contribuir para a redução do tempo de execução da tarefa.

A partir da obtenção das informações para a realimentação do processo apresentadas anteriormente, sugere-se como trabalho futuro o desenvolvimento de uma aplicação inteligente, ou seja, um agente, que auxilie, de forma automatizada, tanto a realimentação como as adaptações ou possíveis evoluções do processo que possam auxiliar na melhoria do desempenho do mesmo, a partir do uso das métricas previamente estabelecidas.

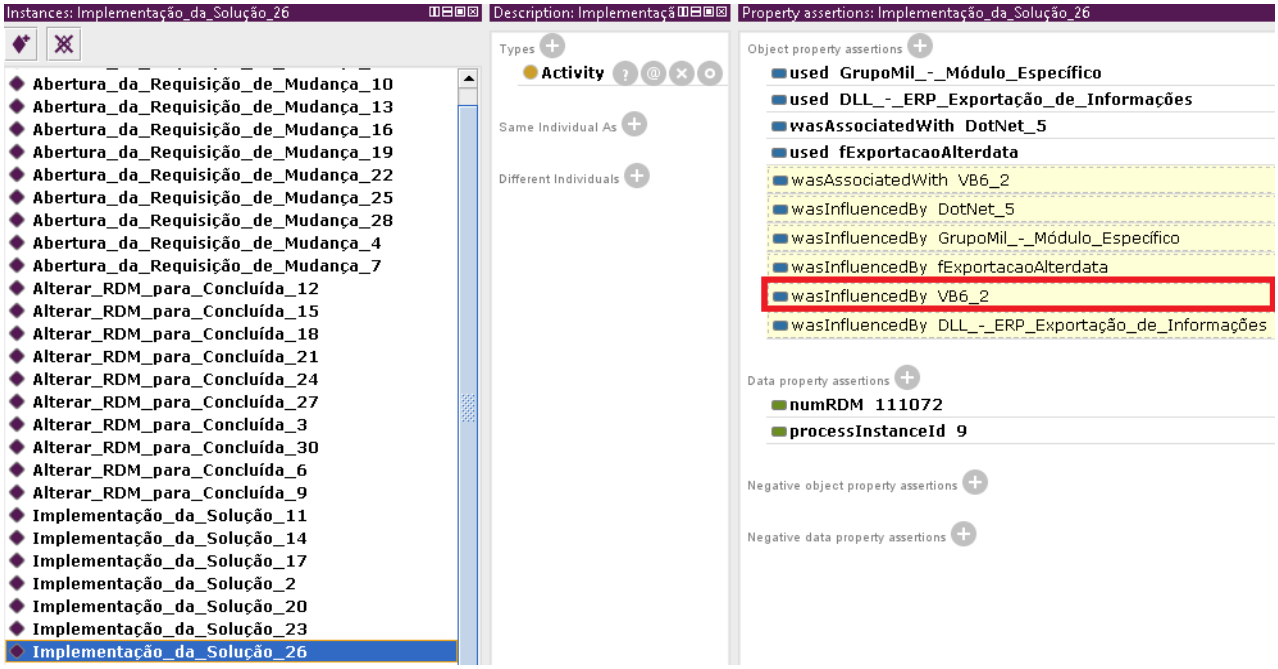


Figura 6.17: Tarefa implementação da Solução (id=26)

6.2.2 Processo 2

Com o objetivo de verificar se os tipos de informações obtidas a partir da aplicação das fases da proposta ao Processo 1 seriam as mesmas obtidos em outros processos, um estudo envolvendo outro processo foi realizado. Assim, esta subseção apresenta a execução da prova de conceito usando um processo para solicitação e implementação de novos recursos e tratamento de erros em um ERP Contábil da Projetus Tecnologia da Informação Ltda, uma empresa de desenvolvimento de software brasileira, de pequeno porte [PROJETUS, 2015].

Com base nas especificações deste processo, foi criado o modelo de fluxo do processo (Figura 6.18). Este processo é composto por 16 tarefas distintas, sendo estas executadas por 6 papéis diferentes (Cliente, Equipe de Teste, Suporte, Gerente de Suporte, Gerente de Desenvolvimento e Programador).

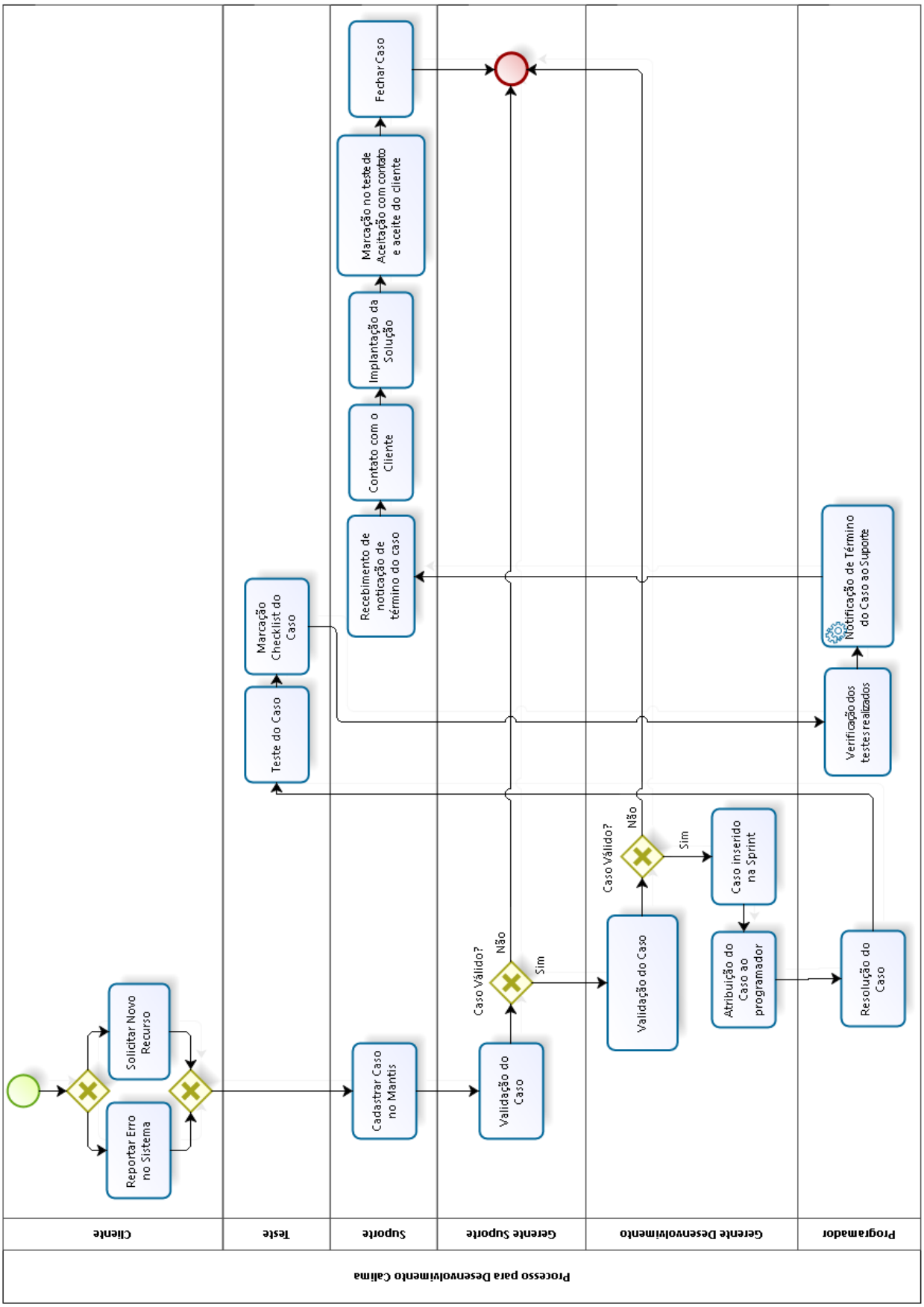


Figura 6.18: Modelo de Fluxo Processo 2

Após a definição do modelo apresentado anteriormente, foram desenvolvidas as fases propostas no Capítulo 5, conforme descrito nas subseções a seguir.

6.2.2.1 Fase 1: Definição das medidas do processo a serem monitoradas

Para o processo de solicitação/implementação de novos recursos e tratamento de erros em um ERP Contábil, de requisição e gerenciamento de mudanças nos softwares desenvolvidos nesta seção, foram estabelecidas quatro medidas a serem coletadas e armazenadas durante a fase de Execução e Monitoramento do Processo, sendo elas: (1) Porcentagem de erros por módulo do sistema; (2) Porcentagem de solicitações de novos recursos por módulo do sistema; (3) Tempo de duração da execução do processo, e (4) Tempo médio de duração das execuções do processo. Estas medidas foram definidas conforme proposto na Seção 5.1 e são detalhadas a seguir (vide Tabelas 6.6 a 6.9).

Tabela 6.6: Definição da Medida de Porcentagem de Erros por Módulo do Sistema (PEM)

1	**Nome:**	Porcentagem de Erros por Módulo do Sistema
2	**Definição**	Medida utilizada para quantificar a porcentagem de erros por módulo do sistema com relação ao total de erros reportados, através do processo de solicitação/implementação de novos recursos e tratamento de erros do ERP Contábil.
3	**Mnemônico:**	PEM
4	**Tipo de Medida:**	Medida Derivada
5	**Entidade Medida:**	Processo de solicitação/implementação de novos recursos e tratamento de erros do ERP Contábil
6	**Propriedade Medida:**	Erros Reportados
7	**Unidade de Medida:**	Número de erros
8	**Tipo de Escala:**	Absoluta
9	**Valores da Escala:**	Números reais positivos, utilizando-se uma precisão de duas casas decimais.
10	**Intervalo Esperado dos Dados:**	-
11	**Fórmula de Cálculo de Medida:**	PEM = TEM / TE Onde: TEM = total de erros por módulo do sistema; TE = total de erros reportados.
12	**Procedimento de Medição:**	Calcular a porcentagem de erros por módulo do sistema, usando a fórmula de cálculo da medida.
13	**Momento da Medição:**	Após a execução do processo.
14	**Periodicidade de Medição**	Sempre que for finalizada uma execução do processo.
15	**Responsável pela Medição:**	Será realizada de forma automatizada (sistema).
16	**Procedimento de Análise:**	- Representar em um gráfico os valores coletados para a medida, exibindo a variação da mesma ao longo das execuções do processo, de forma a possibilitar o comportamento da mesma e identificar

		ações corretivas, quando pertinente.
17	**Momento da Análise de Medição:**	Após a execução do processo.
18	**Periodicidade da Análise:**	Após a execução de cada instância do processo.
19	**Responsável pela Análise:**	Gerente do Projeto

Tabela 6.7: Definição da Medida de Porcentagem de Novos Recursos por Módulo do Sistema(PNRM)

1	**Nome:**	Porcentagem de Novos Recursos por Módulo do Sistema
2	**Definição**	Medida utilizada para quantificar a porcentagem de solicitações de novos recursos por módulo do sistema com relação ao total de novos recursos solicitados, pelo processo de solicitação/implementação de novos recursos e tratamento de erros do ERP Contábil.
3	**Mnemônico:**	PNRM
4	**Tipo de Medida:**	Medida Derivada
5	**Entidade Medida:**	Processo de solicitação/implementação de novos recursos e tratamento de erros do ERP Contábil
6	**Propriedade Medida:**	Novos Recursos Solicitados
7	**Unidade de Medida:**	Número de novos recursos
8	**Tipo de Escala:**	Absoluta
9	**Valores da Escala:**	Números reais positivos, utilizando-se uma precisão de duas casas decimais.
10	**Intervalo Esperado dos Dados:**	-
11	**Fórmula de Cálculo de Medida:**	PNRM = TNRM / TNR Onde: TNRM = total de novos recursos por módulo do sistema; TNR = total de novos recursos reportados.
12	**Procedimento de Medição:**	Calcular a porcentagem de novos recursos por módulo do sistema, usando a fórmula de cálculo da medida.
13	**Momento da Medição:**	Após a execução do processo.
14	**Periodicidade de Medição**	Sempre que for finalizada uma execução do processo.
15	**Responsável pela Medição:**	Será realizada de forma automatizada (sistema).
16	**Procedimento de Análise:**	- Representar em um gráfico os valores coletados para a medida, exibindo a variação da mesma ao longo das execuções do processo, de forma a possibilitar o comportamento da mesma e identificar ações corretivas, quando pertinente.
17	**Momento da Análise de Medição:**	Após a execução do processo.
18	**Periodicidade da Análise:**	Após a execução de cada instância do processo.
19	**Responsável pela Análise:**	Gerente do Projeto

Tabela 6.8: Definição da Medida de Tempo de Execução do Processo (TEP)

1	**Nome:**	Tempo de Execução do Processo
2	**Definição**	Medida utilizada para quantificar a duração da execução do processo de Requisição de Mudanças.
3	**Mnemônico:**	TEP
4	**Tipo de Medida:**	Medida Base
5	**Entidade Medida:**	Processo de solicitação/implementação de novos recursos e tratamento de erros do ERP Contábil
6	**Propriedade Medida:**	Tempo
7	**Unidade de Medida:**	Dias
8	**Tipo de Escala:**	Absoluta
9	**Valores da Escala:**	Números inteiros
10	**Intervalo Esperado dos Dados:**	-
11	**Fórmula de Cálculo de Medida:**	TEP = TF – TI Onde: TF = tempo final (data de finalização) do processo; TI = tempo inicial (data de inicialização) do processo.
12	**Procedimento de Medição:**	Calcular o tempo de duração da execução do processo, usando a fórmula de cálculo da medida.
13	**Momento da Medição:**	Ao iniciar e finalizar qualquer execução de uma instância do processo.
14	**Periodicidade de Medição**	Sempre que for inicializada e finalizada uma execução do processo.
15	**Responsável pela Medição:**	Será realizada de forma automatizada (sistema).
16	**Procedimento de Análise:**	- Representar em um gráfico os valores coletados para a medida comparando-a com outras execuções do mesmo processo, de forma a possibilitar a comparação destes. Assim, será possível analisar se a medida do processo em análise apresenta o mesmo comportamento da mesma medida em execuções anteriores. Caso isto não ocorra, será necessário investigar as causas da instabilidade no comportamento do processo e identificar ações corretivas, quando pertinente.
17	**Momento da Análise de Medição:**	Após a execução do processo.
18	**Periodicidade da Análise:**	Após a execução de cada instância do processo.
19	**Responsável pela Análise:**	Gerente do Projeto

Tabela 6.9: Definição da Medida de Tempo Médio de Execução do Processo (TMEP)

1	**Nome:**	Tempo Médio de Execução do Processo
2	**Definição**	Medida utilizada para quantificar o valor médio da duração da execução dos processos de Requisição de Mudanças.
3	**Mnemônico:**	TMEP
4	**Tipo de Medida:**	Medida Derivada
5	**Entidade Medida:**	Processo de solicitação/implementação de novos recursos e tratamento de erros do ERP Contábil

6	**Propriedade Medida:**	Tempo
7	**Unidade de Medida:**	Dias
8	**Tipo de Escala:**	Absoluta
9	**Valores da Escala:**	Números reais positivos, utilizando-se uma precisão de duas casas decimais.
10	**Intervalo Esperado dos Dados:**	-
11	**Fórmula de Cálculo de Medida:**	TMEP = STEP / TPE Onde: STEP = Somatório do Tempo de Execução de Processo de todas as execuções do processo de solicitação/implementação de novos recursos e tratamento de erros do ERP Contábil; TEP = total de execuções do processo de solicitação/implementação de novos recursos e tratamento de erros do ERP Contábil.
12	**Procedimento de Medição:**	Calcular o tempo médio de duração da execução do processo, usando a fórmula de cálculo da medida.
13	**Momento da Medição:**	Após a execução do processo.
14	**Periodicidade de Medição**	Sempre que for finalizada uma execução do processo.
15	**Responsável pela Medição:**	Será realizada de forma automatizada (sistema).
16	**Procedimento de Análise:**	- Representar em um gráfico os valores coletados para a medida, exibindo a variação da mesma ao longo das execuções do processo de solicitação/implementação de novos recursos e tratamento de erros do ERP Contábil, de forma a possibilitar o comportamento da mesma e identificar ações corretivas, quando pertinente.
17	**Momento da Análise de Medição:**	Após a execução do processo.
18	**Periodicidade da Análise:**	Após a execução de cada instância do processo.
19	**Responsável pela Análise:**	Gerente do Projeto

6.2.2.2 Fase 2: Execução e monitoramento do processo

O processo apresentado nesta seção não foi executado e monitorado de acordo com esta proposta de tese, ou seja, não foram utilizados os mecanismos/ferramentas para execução e monitoramento do processo sugeridos no Capítulo 5. O processo não foi executado e monitorado de acordo com o proposto na Fase 2, uma vez que o ferramental necessário para esta fase encontra-se em desenvolvimento. Assim, conseguiu-se junto ao gerente de desenvolvimento da empresa responsável pelo processo descrito, dados referentes a execuções deste processo e, portanto, foram utilizados estes dados para a realização desta fase. Destes dados, foram selecionadas instância de execução com estado (ou status) igual a resolvido, ou seja, cujo processo já havia sido finalizado. Desta forma, na Fase 1 (Seção 6.2.2.1), foram criadas medidas que eram possíveis de serem obtidas ou calculadas com base nos dados fornecidos. Assim como foi realizado com o Processo 1, a

proveniência retrospectiva das execuções do processo foi armazenada utilizando a abordagem PROV-Process.

Para este processo, não foram fornecidos dados de execução que contemplassem todas as tarefas modeladas no diagrama de fluxo do mesmo (Figura 6.18). Destes dados obtidos, foram analisadas 10 instâncias de execução do processo[11]. Com relação a estes dados obtidos, foram utilizados os seguintes:

- Número do caso;
- Cliente que reportou um erro ou solicitou um novo recurso;
- Responsável pela abertura do caso (Relator);
- Tipo do caso (erro ou implementação);
- Prioridade, gravidade, frequência e resumo do caso;
- Data de abertura do caso (data de envio);
- Programador ao qual foi atribuído o caso para sua resolução;
- Módulo e versão do produto que foram alterados; e
- Data de finalização do caso.

Estes dados foram obtidos através de uma planilha enviada pela empresa responsável pelo projeto. Um exemplo dos dados que foram fornecidos e utilizados pode ser visualizado nas Tabelas 6.10 e 6.11. Cada linha destas tabelas representa uma execução distinta de uma instância do processo.

Tabela 6.10: Exemplo de Dados e execução – Processo 2 – Parte 1

Nº	Cliente	Relator	Tipo de Caso	Prioridade	Gravidade	Frequência	Resumo
11946	Empresa _A	Antonio	ERRO	alta	grande	sempre	Erro ao importar NF Serviço - Marília
11697	Empresa _B	Bernardo	IMPLEMENTAÇÃO	normal	pequeno	sempre	Relatório que filtre por Histórico

Tabela 6.11: Exemplo de Dados e execução – Processo 2 – Parte 2

Nº	Data de Envio	Atribuído a	Módulo	Versão do Produto	Finalizado
11946	2015-02-04	Filipe	Fiscal	2.5.07	2015-02-05
11697	2014-12-18	Gustavo	Contabilidade	2.5.06	2015-02-05

[11]Foi fornecido um número bem maior de execuções de instâncias do processo, porém, como toda a manipulação de dados de execução de processos através da abordagem PROV-Process ainda não está completamente automatizada, tornar-se-ia muito custosa a transformação manual de todos os dados de execução e posterior importação dos mesmos na ontologia desta abordagem. Todos os dados utilizados para este exemplo podem ser visualizados em http://gabriellacastro.com.br/dsc/ex2/ex2.xlsx. Ressalta-se que a única intervenção feita nos mesmos foi o mascaramento dos nomes de pessoas envolvidas (clientes ou funcionários) na execução do processo.

Estes dados foram importados para o repositório relacional da base PROV-Process de acordo com os seguintes critérios:

- Para todas as execuções cujos dados foram analisados, foram criados quatro registros na tabela *Activity*, com os respectivos nomes:
    - Reportar Erro no Sistema ou Solicitar Novo Recurso;
    - Cadastrar Caso no Mantis;
    - Resolução do Caso; e
    - Fechar Caso.
- O número do caso ao qual a execução estava associada foi inserido como um atributo de cada uma das atividades anteriores, utilizando as tabelas *Attribute* e *Activity*_Attribute.
- O responsável pela atividade Reportar Erro no Sistema ou Solicitar Novo Recurso, campo Cliente dos dados de execução recebidos, foi inserido como registro da tabela *Agent*, utilizando o campo *Name* e o tipo de agente *Organization*.
- O relacionamento entre a atividade Reportar Erro no Sistema ou Solicitar Novo Recurso e o responsável pela mesma foram inseridos como registros da tabela *WasAssociatedWith*.
- O tipo, a prioridade, a gravidade, a frequência e o resumo do caso foram inseridos como atributos da atividade Cadastrar Caso no Mantis, utilizando as tabelas *Attribute* e *Activity_Attribute*.
- A data de abertura do caso (data de envio) foi incluída pelo atributo *startTime* da atividade de Cadastrar Caso no Mantis.
- O responsável pela atividade Cadastrar Caso no Mantis, campo 'Relator' dos dados de execução recebidos, foi inserido como registro da tabela *Agent*, utilizando o campo *Name* e o tipo de agente *Person*.
- O relacionamento entre a atividade Cadastrar Caso no Mantis e o responsável pela mesma foram inseridos como registros da tabela *WasAssociatedWith*.
- O responsável pela atividade Resolução do Caso, campo 'Atribuído a' dos dados de execução recebidos, foi inserido como registro da tabela *Agent*, utilizando o campo *Name* e o tipo de agente *Person*.
- O relacionamento entre a atividade Resolução do Caso e o responsável pela mesma foram inseridos como registros da tabela *WasAssociatedWith*.
- Os valores fornecidos como Módulo e Versão foram incluídos como registros da tabela *Entity* e foram associados à tarefa Resolução do Caso, criando-se registros na tabela *Used*.

- Os valores fornecidos como Módulo e Versão, incluídos como registros da tabela *Entity*também foram associados aos agentes que o manipularam, criando-se registros na tabela *WasAttributedTo*.
- O campo 'Finalizado' informado nos dados de execução recebidos, foi inserido utilizando o atributo *endTime* da atividade Fechar Caso.
- De forma a identificar a qual instância da execução do processo uma determinada atividade está associada, foi inserido um atributo relacionado ao id da instância (*processInstanceId*) a todas as tarefas, utilizando as tabelas *Attribute* e *Activity_Attribute*.

6.2.2.3 Fase 3: Análise da execução do processo

Durante esta etapa, as quatro medidas definidas na Fase 1 foram analisadas, com base nos dados de execução do processo. Esta análise é descrita a seguir, para cada uma das quatro medidas:

- **Medida 1: Porcentagem de erros por módulo do sistema**

Para obtenção desta medida, utilizou-se o total de tarefas *Cadastrar Caso no Mantis* cujo tipo da mesma era igual a ERRO e que estavam associadas a um mesmo módulo do sistema. O total de tarefas com este tipo foi dividido pelo total de execuções do processo cuja tarefa *Cadastrar Caso no Mantis* possuía tipo igual ERRO até o momento de coleta desta medida. Os dados obtidos para esta medida, após cada uma das 10 instâncias executadas, são listados abaixo, assim como o gráfico gerado a partir das mesmas (Figura 6.19).

- Instância 1: Erro no Módulo Fiscal
  - Módulo Fiscal 1/1 = 1 = 100,00%
- Instância 2:Não se tratava de um erro no sistema
  - Módulo Fiscal 1/1 = 1 = 100,00%
- Instância 3: Não se tratava de um erro no sistema
  - Módulo Fiscal 1/1 = 1 = 100,00%
- Instância 4: Erro no Módulo Folha de Pagamento
  - Módulo Fiscal: 1/2 = 0,5 = 50,00%
  - Módulo Folha de Pagamento: 1/2 = 0,5 = 50,00%
- Instância 5: Erro no Módulo Folha de Pagamento
  - Módulo Fiscal: 1/3 = 0,3333 = 33,33%
  - Módulo Folha de Pagamento: 2/3 = 0,6667 = 66,67%
- Instância 6: Erro no Módulo Fiscal
  - Módulo Fiscal: 2/4 = 0,5 = 50,00%

- Módulo Folha de Pagamento: 2/4 = 0,5 = 50,00%

- Instância 7: Não se tratava de um erro no sistema

  - Módulo Fiscal: 2/4 = 0,5 = 50,00%

  - Módulo Folha de Pagamento: 2/4 = 0,5 = 50,00%

- Instância 8: Erro no Módulo Folha de Pagamento

  - Módulo Fiscal: 2/5 = 0,4 = 40,00%

  - Módulo Folha de Pagamento: 3/5 = 0,6 = 60,00%

- Instância 9: Erro no Módulo Folha de Pagamento

  - Módulo Fiscal: 2/6 = 0,3333 = 33,33%

  - Módulo Folha de Pagamento: 4/6 = 0,6667 = 66,67%

- Instância 10: Erro no Módulo Folha de Pagamento

  - Módulo Fiscal: 2/7 = 0,2857 = 28,57%

  - Módulo Folha de Pagamento: 5/7 = 71,43 = 71,43%

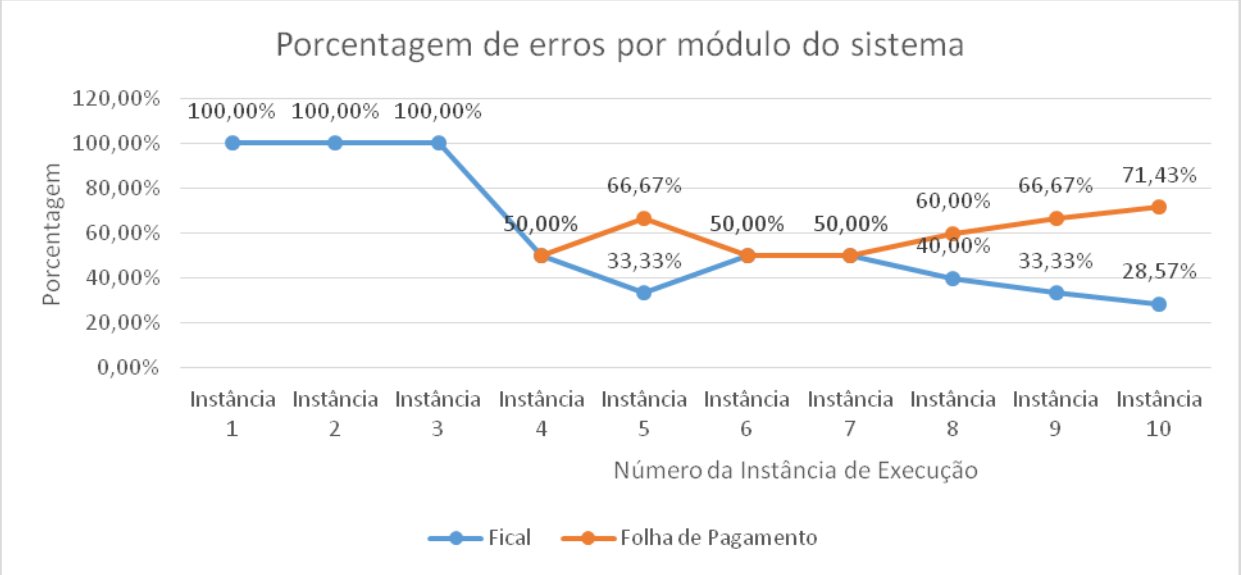


Figura 6.19: Gráfico da Porcentagem de erros por módulo do sistema

- **Medida 2: Porcentagem de solicitações de novos recursos por módulo do sistema**

Para obtenção desta medida, foram somadas as tarefas *Cadastrar Caso no Mantis* cujo tipo da mesma era igual a IMPLEMENTAÇÃO e que estavam associadas a um mesmo módulo do sistema. O total de tarefas com este tipo foi dividido pelo total de execuções do processo cuja tarefa *Cadastrar Caso no Mantis* possuía tipo igual IMPLEMENTAÇÃO até o momento de coleta desta medida. Os dados obtidos para esta medida, após cada uma das 10 instâncias executadas, são listados abaixo, assim como o gráfico gerado a partir das mesmas (Figura 6.20).

- Instância 1: Não se tratava de uma solicitação de novo recurso

- Instância 2: Implementação no Módulo Contabilidade

- Módulo Contabilidade 1/1 = 1 = 100,00%

- Instância 3: Implementação no Módulo Contabilidade
  - Módulo Contabilidade 2/2 = 1 = 100,00%
- Instância 4: Não se tratava de uma solicitação de novo recurso
  - Módulo Contabilidade 2/2 = 1 = 100,00%
- Instância 5: Não se tratava de uma solicitação de novo recurso
  - Módulo Contabilidade 2/2 = 1 = 100,00%
- Instância 6: Não se tratava de uma solicitação de novo recurso
  - Módulo Contabilidade 2/2 = 1 = 100,00%
- Instância 7: Implementação no Módulo Folha de Pagamento
  - Módulo Contabilidade 2/3 = 0,6667 = 66,67%
  - Módulo Folha de Pagamento: 1/3 = 0,3333 = 33,33%
- Instância 8: Não se tratava de uma solicitação de novo recurso
  - Módulo Contabilidade 2/3 = 0,6667 = 66,67%
  - Módulo Folha de Pagamento: 1/3 = 0,3333 = 33,33%
- Instância 9: Não se tratava de uma solicitação de novo recurso
  - Módulo Contabilidade 2/3 = 0,6667 = 66,67%
  - Módulo Folha de Pagamento: 1/3 = 0,3333 = 33,33%
- Instância 10: Não se tratava de uma solicitação de novo recurso
  - Módulo Contabilidade 2/3 = 0,6667 = 66,67%
  - Módulo Folha de Pagamento: 1/3 = 0,3333 = 33,33%

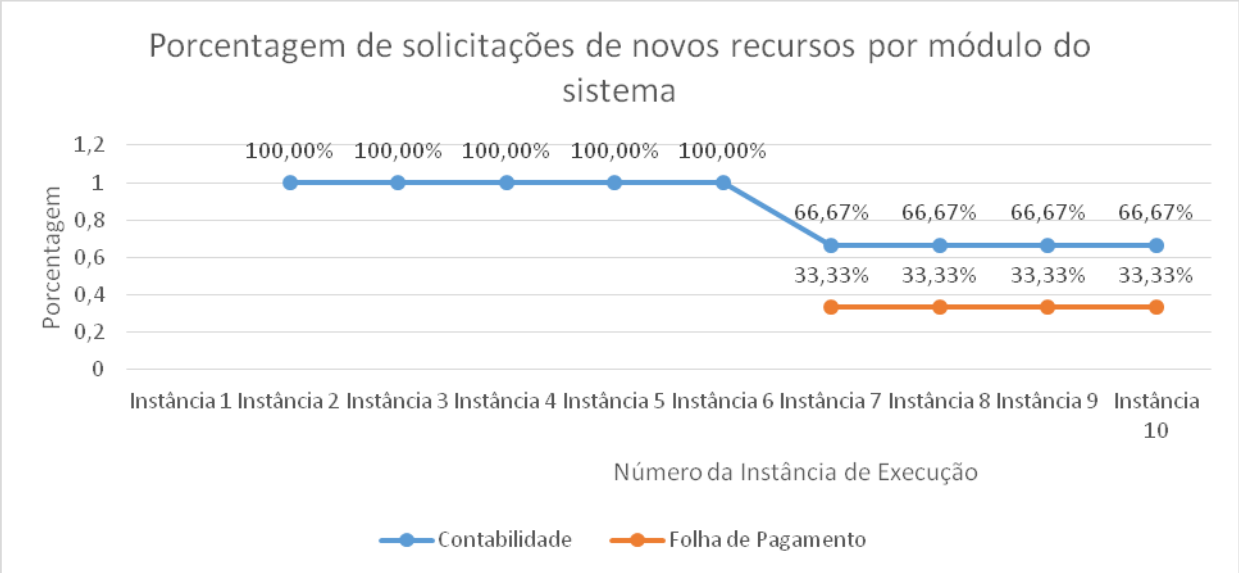


Figura 6.20: Gráfico da Porcentagem de solicitações de novos recursos por módulo do sistema

- **Medida 3: Tempo de duração da execução do processo**

Para obtenção desta medida, bastou-se subtrair da data final, campo *endTime* da atividade Fechar Caso, o valor do campo *startTime* da atividade de Cadastrar Caso no Mantis. Estes resultados podem ser obtidos por meio de consultas realizadas na base de dados da abordagem PROV-Process. Os dados obtidos para esta medida, para as 10 instâncias executadas, são listadas abaixo, assim como o gráfico gerado a partir das mesmas (Figura 6.21).

- Instância 1: 1 dia
- Instância 2: 49 dias
- Instância 3: 708 dias
- Instância 4: 21 dias
- Instância 5: 5 dias
- Instância 6: 64 dias
- Instância 7: 54 dias
- Instância 8: 27 dias
- Instância 9: 27 dias
- Instância 10: 5 dias

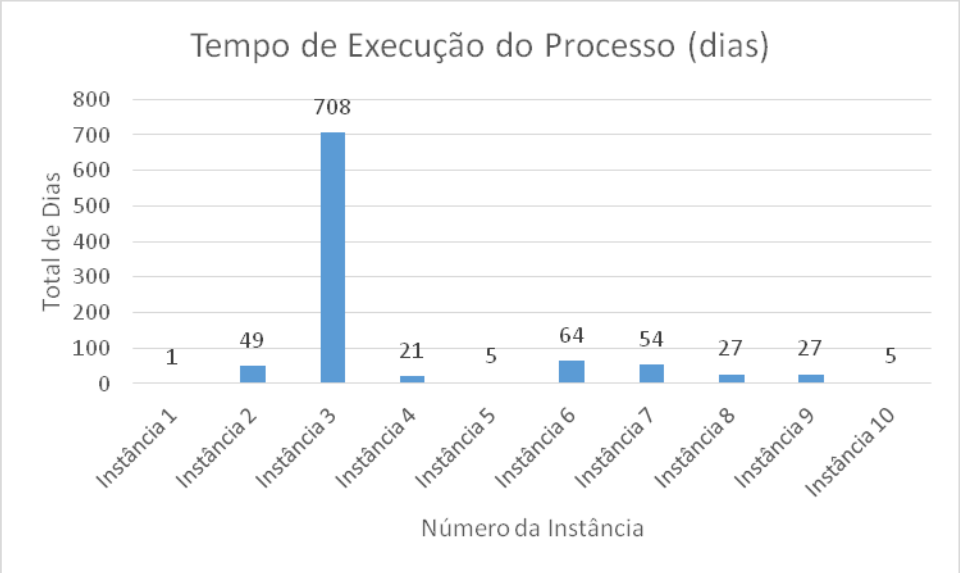


Figura 6.21: Gráfico da Medida de Tempo de Execução do Processo

- **Medida 4: Tempo Médio de Duração das Execuções do Processo**

Para a obtenção desta medida, que é calculada após a execução de cada instância, foi realizado o somatório do tempo de execução de processo (Medida 3) de todas as execuções do processo de RDM, dividido pelo número de execuções deste processo. Os dados obtidos para esta medida, após cada uma das 10 instâncias executadas, são listadas a seguir, assim como o gráfico gerado a partir das mesmas (Figura 6.22).

- Instância 1: 1,00 dia
- Instância 2: 25,00 dias
- Instância 3: 252,67 dias
- Instância 4: 194,75 dias
- Instância 5: 156,80 dias
- Instância 6: 141,33 dias
- Instância 7: 128,86 dias
- Instância 8: 116,13 dias
- Instância 9: 106,22 dias
- Instância 10: 96,10 dias

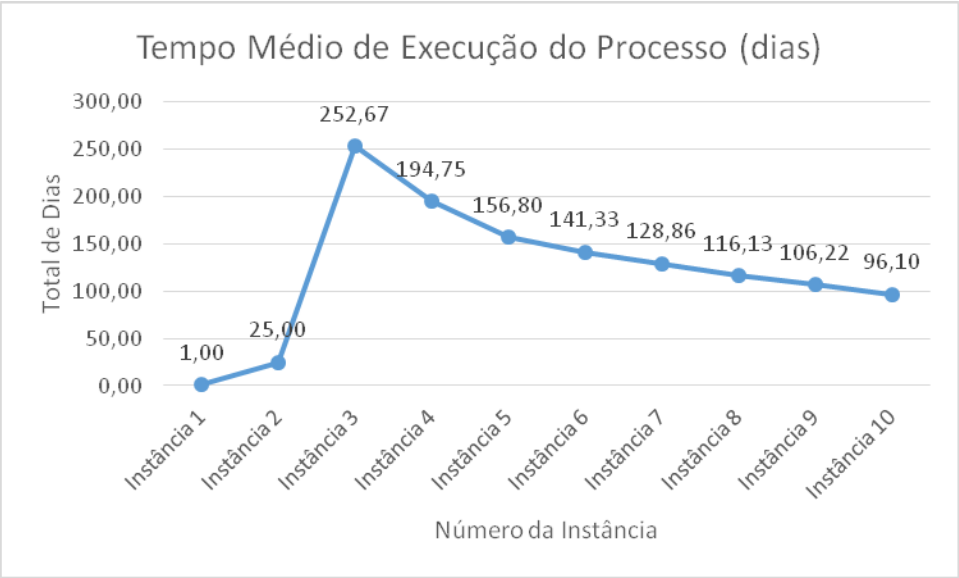


Figura 6.22: Gráfico da Medida de Tempo Médio de Execução do Processo

Foi também aplicado a este segundo processo o mecanismo de inferência à ontologia populada com os dados de execução do mesmo. Assim, foi possível a derivação de informações estratégicas, a respeito da execução do processo. Os dados de execução do processo foram exportados da base de dados do PROV-Process e importados para a ontologia desta mesma abordagem[12]. Como informações inferidas a partir dos dados de proveniência retrospectiva deste processo, podem-se destacar três tipos, sendo elas:

1) Agentes que poderiam estar associados à tarefa de Resolução do Caso, haja vista que já manipularam os artefatos envolvidos nesta tarefa em alguma outra execução do processo. A Figura 6.23, apresenta, por exemplo, que a atividade Resolução do Caso (id = 23) foi influenciada pelo agente Filipe, haja vista que este agente manipulou artefatos comuns a esta tarefa em outras instâncias deste processo.

[12] A ontologia gerada para este exemplo encontra-se disponível em: http://gabriellacastro.com.br/dsc/ex2/ex2.owl.

Este mesmo tipo de informação (agentes que poderiam estar associados à tarefa de Implementação da Solução) também ocorre para as tarefas de Resolução do Caso com id's iguais a 3, 7, 11, 15, 19, 27, 31, 35 e 39.

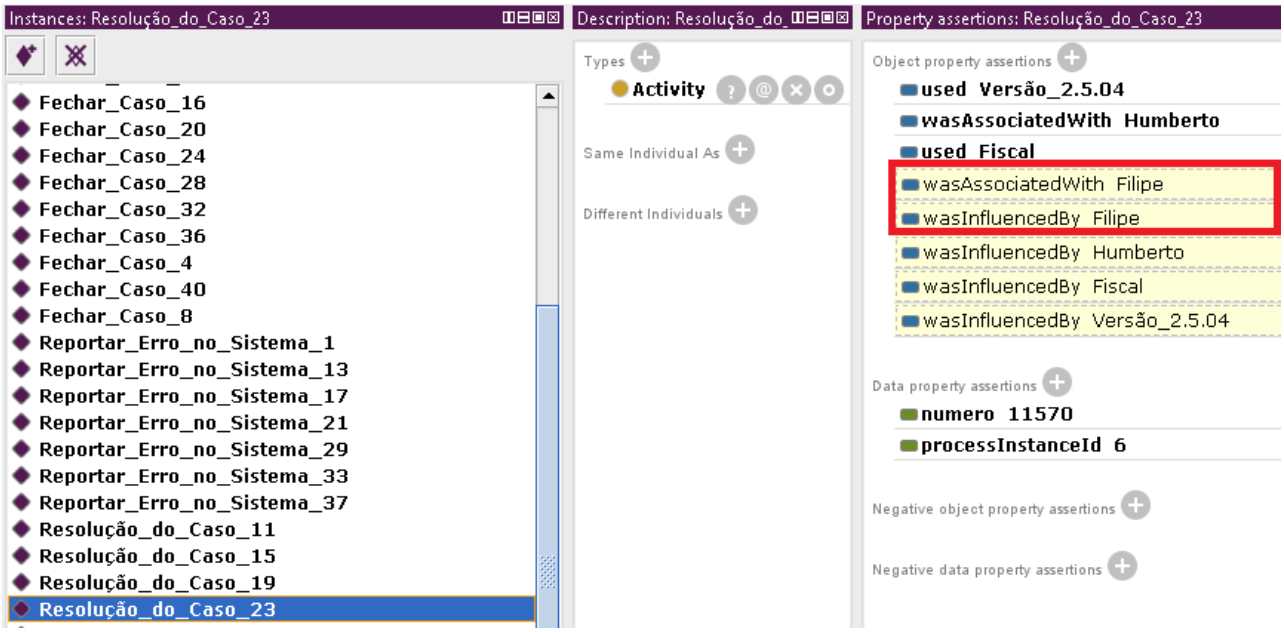


Figura 6.23: Agentes que influenciaram uma tarefa

2) Listagem de todas as tarefas em que um agente estava envolvido, bem como os artefatos manipulados pelo mesmo. Embora este tipo de informação possa ser obtido por meio de consultas na própria base de dados da abordagem PROV-Process, com a utilização da ontologia e da máquina de inferência, este tipo de informação pode ser obtido mais facilmente (com uma simples busca em SPARQL), conforme pode ser visualizado na Figura 6.24.

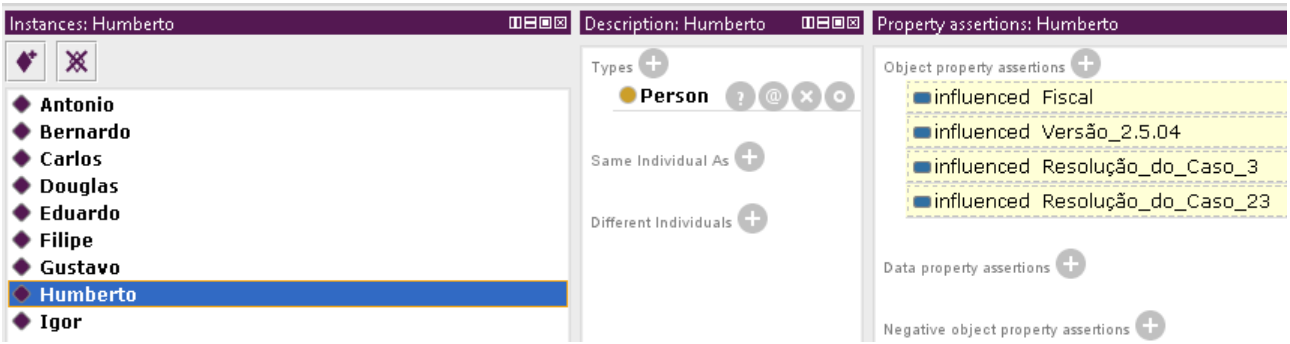


Figura 6.24: Tarefas e Artefatos manipulados pelo Agente Humberto

3) Listagem de todas as tarefas em que um determinado artefato foi utilizado. Embora este tipo de informação possa ser obtido por meio de consultas na própria base de dados da abordagem PROV-Process, com a utilização da ontologia e da máquina de inferência, este tipo de informação

pode ser obtido facilmente (com uma simples busca em SPARQL), conforme pode ser visualizado na Figura 6.25.

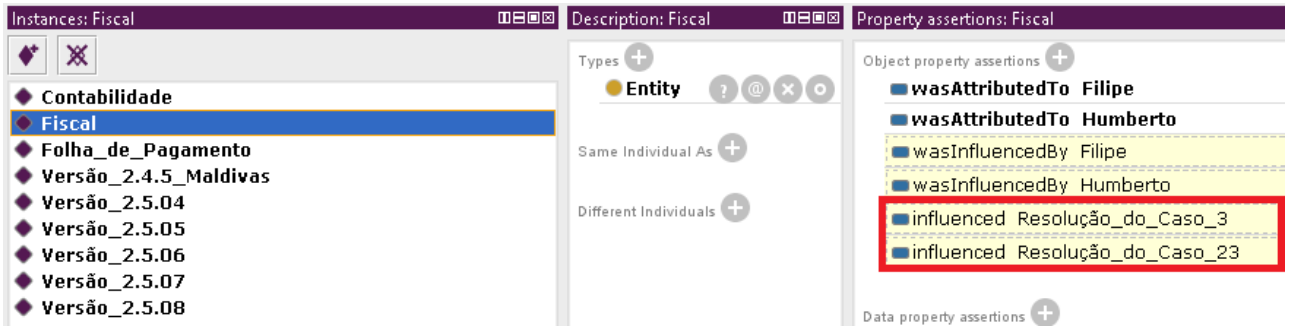


Figura 6.25: Tarefas em que um determinado artefato foi manipulado

6.2.2.4 Fase 4: Realimentação do Processo

No contexto deste processo, algumas informações inferidas a partir do uso da ontologia poderiam auxiliar na melhoria do desempenho do processo em questão. Por exemplo, conforme feito no Processo 1, ao analisar os resultados obtidos para as medidas 3 e 4 (Tempo de Duração da Execução do Processo e Tempo Médio de Duração das Execuções do Processo), observa-se que a execução da instância 3 elevou, consideravelmente, os valores obtidos até então. Assim, a execução desta instância necessitaria de uma análise posterior de forma a evitar que novas execuções deste processo tenham um tempo de execução elevado como este. Neste caso, ao analisar esta execução por meio da ontologia e dos dados inferidos, poder-se-ia realimentar o processo com as seguintes informações:

1) Ao avaliar a atividade de Resolução do Caso (id=11), realizada na execução da instância 3, foi inferida a informação de que o Agente Gustavo influenciou esta tarefa (vide Figura 6.26), haja vista que já manipulou os artefatos envolvidos em alguma outra execução do processo, o que poderia, talvez, influenciar a redução do tempo de execução do processo. Assim, o processo poderia ser realimentado com anotações junto aos artefatos de agentes que já o manipularam. Desta forma, durante a execução de um processo, ao iniciar a manipulação de um determinado artefato, o executor da tarefa poderia incluir novos agentes para a solução daquela tarefa, haja vista que os mesmos já utilizaram aquele artefato em alguma execução anterior e, portanto, poderia compartilhar algum conhecimento relativo ao mesmo, possivelmente, contribuindo para a redução do tempo de execução da tarefa.

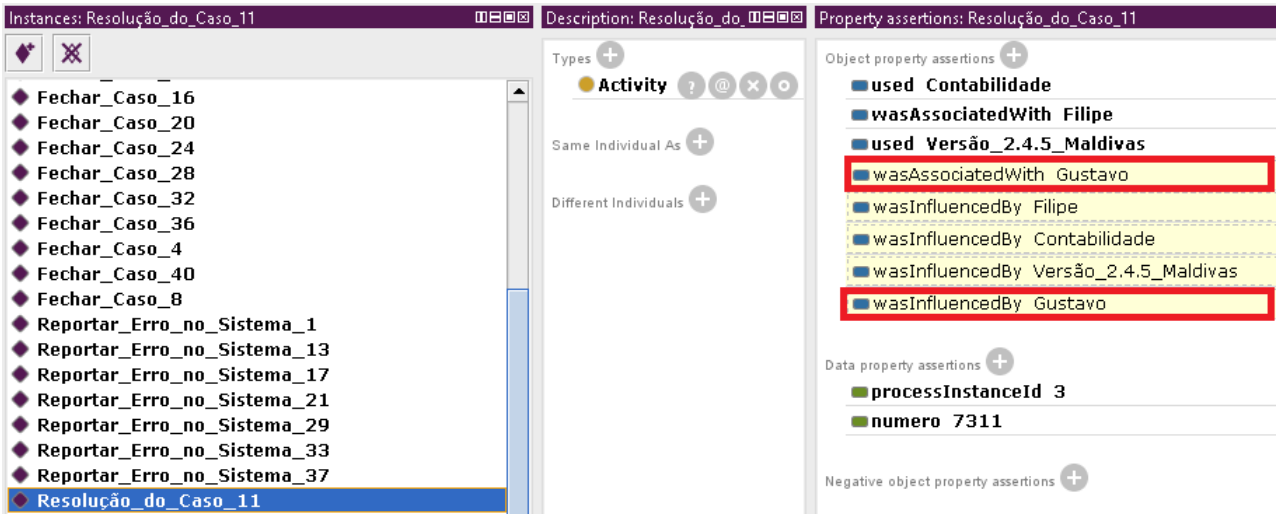


Figura 6.26: Inferência sobre a tarefa Resolução do Caso (id=11)

2) Ao analisar os artefatos manipulados durante a execução da instância 3 do processo, verificou-se que o artefato (ou entidade, de acordo com o modelo PROV) 'Versão 2.4.5 Maldivas' foi manipulado apenas pela tarefa Resolução do Caso (id=11) desta instância de execução do processo (vide Figura 6.27). Assim, o processo poderia ser realimentado com uma anotação junto à tarefa de Resolução do Caso que, a manipulação do artefato 'Versão 2.4.5 Maldivas' nesta tarefa poderá acarretar num aumento considerável no tempo de execução do processo.

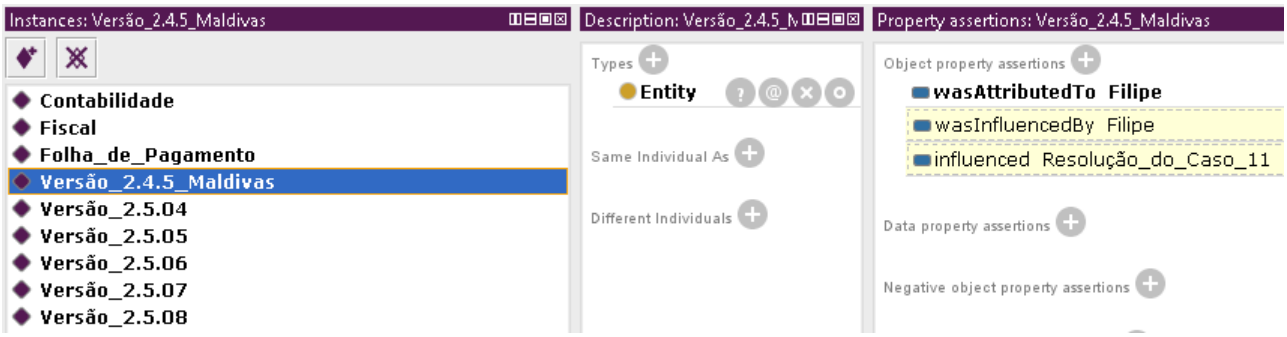


Figura 6.27: Artefato 'Versão 2.4.5 Maldivas'

6.3 Análise das Hipóteses

Ao executar a PoC proposta neste trabalho, considerando os dois processos da indústria utilizados, foi possível identificar a obtenção de quatro tipos distintos de informações:

1. **Tarefas que influenciaram a geração de outras tarefas;**
2. **Agentes que poderiam estar associados a uma tarefa;**
3. **Tarefas em que um agente estava envolvido e artefatos manipulados pelo mesmo;**
4. **Tarefas em que um determinado artefato foi utilizado.**

Entre estes quatro tipos de informações, todas elas foram obtidas no Processo 1 analisado. Já no Processo 2, o primeiro tipo de informação (tarefas que influenciaram a geração de outras tarefas) não foi obtido, haja vista que não havia nenhum tipo de informação nos dados de execução fornecidos pela organização que permitissem a inferência da mesma. Com o desenvolvimento do ferramental para apoio à Fase 2 da abordagem (Execução e monitoramento do processo), este tipo de informação de proveniência será capturada durante a execução do processo.

Ao utilizar-se as informações que podem ser obtidas através da abordagem proposta em conjunto com métricas previamente estabelecidas para o processo em questão, tem-se indícios de que é possível a derivação das informações que poderão contribuir com a melhoria do processo em execuções posteriores. As informações obtidas durante a análise dos exemplos de processos apresentados na Seção 6.2 podem ser classificadas em dois tipos:

1. **Informação junto aos artefatos que, se manipulados pelo processo, poderão acarretar num aumento considerável no tempo de execução de novas instâncias do processo.**
2. **Informação junto aos artefatos de agentes que já o manipularam. Desta forma, durante a execução de um processo, ao iniciar a manipulação de um determinado artefato, o executor da tarefa poderia incluir novos agentes para a solução daquela tarefa, haja vista que os mesmos já utilizaram aquele artefato em alguma execução anterior e, portanto, poderia compartilhar algum conhecimento relativo ao mesmo, o que, possivelmente, poderia contribuir para a redução do tempo de execução da tarefa.**

Os dois tipos de informações foram obtidos tanto para o Processo 1 quanto para o Processo 2 analisados.

Considerando as análises apresentadas acima, é possível provar a falsidade da hipótese H_{TO} e validar a hipótese $H_{T1,}$ haja vista que foram obtidas informações que podem ser utilizadas para auxiliar na melhoria do desempenho do processo, de acordo com métricas previamente definidas.

6.4 Limitações Encontradas

Ao executar o estudo dos processos obtidos junto às organizações, algumas limitações foram identificadas. Com relação ao Processo 1, pode-se citar:

- Não foram informados todos os dados de execução do processo, apenas uma planilha com alguns destes dados, o que pode interferir nos resultados apresentados na Fase 3, de análise da execução do processo.

- Nos dados de execução informados, não foram passadas as informações dos atores que, de fato, realizaram as tarefas. Foi informado apenas a equipe que realizou determinada tarefa.

Com relação ao Processo 2, para solicitação e implementação de novos recursos e tratamento de erros em um ERP Contábil, pode-se citar:

- Não foram informados todos os dados de execução do processo, apenas uma planilha com alguns destes dados, o que pode interferir nos resultados apresentados na Fase 3, de análise da execução do processo.

6.5 Ameaças à Validade

Há quarto tipos de validade dos resultados de um experimento: (1) **Validade de Conclusão**: relacionada à habilidade de se obter uma conclusão correta a respeito dos relacionamentos entre o tratamento e o resultado do experimento; (2) **Validade Interna**: define se o relacionamento observado entre o tratamento e o resultado é causal, e não é o resultado da influência de outro fator que não é controlado ou não foi medido; (3) **Validade de Construção**: considera dos relacionamentos entre a teoria e a observação, ou seja, se o tratamento reflete a causa e o resultado reflete o efeito e (4) **Validade Externa**: define as condições que limitam a habilidade de generalizar os resultados de um experimento para a prática industrial (TRAVASSOS *et al*., 2002).

Na PoC realizada, pode-se citar as seguintes ameaças à validade:

1. **Validade de Conclusão**: Tanto o total de processos (2) utilizados nesta PoC como o número de instâncias de execução utilizadas para a PoC (10 instâncias de um dos processos) não é o ideal do ponto de vista estatístico. Assim, os resultados obtidos devem ser considerados apenas como indícios.
2. **Validade Interna**: Os processos utilizados nas organizações foram remodelados utilizando a notação desta proposta de trabalho (BPMN), a partir de entrevistas/reuniões com conhecedores deste processo na empresa, porém, apesar dos modelos criados terem sido aprovados pelos respectivos gerentes do processo, não foi verificado se, de fato, na empresa, o processo ocorria exatamente como o especificado. Como uma ameaça à validade interna, pode-se citar também o fato de não terem sido fornecidos dados relativos à execução dos processos como um todo e sim, apenas de algumas tarefas.
3. **Validade de Construção**: Embora tenham sido apresentadas informações que podem ser utilizadas para auxiliar na melhoria do desempenho do processo, não foi possível avaliar se, a partir destas informações, o gerente do processo conseguiria, de fato,

observar a melhoraria no desempenho do processo, haja vista que não foram obtidas/realizadas execuções dos processos, posteriormente à análise dos dados iniciais de execução que foram fornecidos.

4. **Validade Externa**: Apenas com os dois processos utilizados, não foi possível representar todas as situações possíveis de um processo de software e as informações de proveniência que podem ser úteis após execuções do mesmo para melhorar seu desempenho. Apesar dos exemplos apresentados serem reais, ainda é necessário verificar se os objetivos da abordagem serão atingidos em outros processos de software da indústria.

6.6 Considerações sobre o Capítulo

Neste capítulo, foi apresentada uma prova de conceito com o objetivo de demostrar que dados de proveniência em conjunto com ontologias podem fornecer informações estratégicas a serem utilizadas para a melhoria do desempenho de processos, a partir de métricas previamente definidas. A partir do estudo realizado, foram obtidos quatro tipos de informações a partir de dados de execução de dois processos reais da indústria. Utilizando estas informações, em conjunto com métricas previamente estabelecidas e relacionadas ao tempo de execução do processo, puderam ser derivados dois tipos de informações que podem realimentar o processo, de forma a possibilitar uma melhoria em seu desempenho, com relação ao seu tempo de execução. Foram também apresentadas as limitações e as ameaças ao estudo realizado.

No capítulo seguinte, são apresentadas as considerações finais desta proposta, relacionando suas contribuições, atividades realizadas até o momento e atividades previstas, bem como o cronograma para realização das mesmas.

7. CONSIDERAÇÕES FINAIS

7.1 Contribuições

Esta proposta de tese tem como objetivo elaborar uma abordagem para definir, implementar e avaliar uma sistemática para apoiar a execução, monitoramento e realimentação de processos, visando melhorar o desempenho destes por meio da utilização de dados de proveniência e ontologias. Assim, esta abordagem é composta por quatro etapas distintas, sendo elas: (1) Definição das medidas do processo a serem monitoradas; (2) Execução e monitoramento do processo; (3) Análise da execução do processo e (4) Realimentação do Processo. Como principais contribuições, esta proposta de abordagem traz:

1. **Apoio ao gerente de processo na definição das medidas do processo** a serem coletadas e armazenadas durante a fase de execução e monitoramento, de acordo com os objetivos da organização;
2. **Mecanismo para execução do processo** e coleta dos dados necessários para o estabelecimento das medidas definidas pelo gerente de processo, além da captura dos dados relativos à proveniência retrospectiva do processo;
3. **Mecanismo para análise e visualização da execução do processo**, com base nas medidas definidas pelo gerente de processo;
4. **Mecanismo para derivação de informações estratégicas** relativas ao processo, pela manipulação dos dados de proveniência do processo em conjunto com ontologias e máquinas de inferência;
5. **Apoio à realimentação do processo definido**, com base nos dados de proveniência e medidas coletadas durante a execução do processo; e
6. **Avaliação da viabilidade de aplicação/utilização da abordagem** por meio da execução de estudos experimentais e definição de mecanismos de verificação dos diferentes mecanismos que compõem cada uma das quatro etapas da abordagem.

7.2 Atividades Realizadas

As atividades realizadas até o momento com relação a esta pesquisa são listadas a seguir:

1. Pesquisa bibliográfica geral sobre processos e processos de software;
2. Pesquisa bibliográfica geral sobre proveniência de dados e modelos utilizados para a representação da mesma;

3. Revisão bibliográfica inicial de publicações científicas que tratam sobre a execução e o monitoramento de processos de software, com o propósito de caracterizar abordagens (como ferramentas, métodos, processos e técnicas), com relação à execução de processos;
4. Busca estruturada com o objetivo de encontrar soluções ou ferramentas que têm sido utilizadas para execução, monitoramento e avaliação de processos de software;
5. Elaboração da proposta para apoiar a execução, monitoramento e realimentação de processos, visando melhorar o desempenho destes, composta por 4 quatro etapas bem definidas, sendo elas: (1) Definição das medidas do processo a serem monitoradas; (2) Execução e monitoramento do processo; (3) Análise da execução do processo e (4) Realimentação do Processo; e
6. Prova de conceito da abordagem, aplicando as três primeiras fases da mesma a dois processos da indústria, detalhando todos os passos e procedimentos seguidos.

Além das atividades anteriormente descritas, esta proposta de tese derivou um trabalho de mestrado voltado para a captura e análise dos dados de proveniência de processos de software (abordagem PROV-Process) que se encontra em andamento, em parceria com a Universidade Federal de Juiz de Fora.

Anteriormente à definição desta proposta de tese, foram realizadas duas publicações, a primeira a partir do trabalho realizado em uma disciplina do curso e a segunda decorrente do trabalho realizado em parceria com a Universidade Federal de Juiz de Fora.

- COSTA, G. C. B., SANTANA, F., MAGDALENO, A. M., WERNER, C. M. L., 2014, Monitoring Collaboration in Software Processes Using Social Networks. In: International Conference on Collaboration and Technology (CRIWG), v. 8658. pp. 89-96.
- DALPRA, H. L. O., COSTA, G. C. B., SIRQUEIRA, T. F. M., BRAGA, R., WERNER, C. M., CAMPOS, F., DAVID, J. M. N., 2015, Using Ontology and Data Provenance to Improve Software Processes. In: 7th Brazilian Conference on Ontologies (ONTOBRAS 2015).

7.3 Atividades Previstas

São propostas as seguintes atividades a serem realizadas após o exame de qualificação:

- Especificação e Desenvolvimento da infraestrutura / sistema para apoiar a execução, monitoramento e realimentação do processo, visando melhorar o desempenho de processos derivados da mesma;

- Aplicação da proposta e utilização da infraestrutura criada a exemplos da indústria;
- Estudos experimentais para avaliação da proposta;
- Submissões de artigos para eventos e periódicos (sugestões):
    - *International Conference on Software Engineering* (ICSE)
    - *Symposium on Applied Computing* (SAC)
    - *International Conference on Software Reuse* (ICSR)
    - Simpósio Brasileiro de Engenharia de Software (SBES)
    - *International Conference on Product-focused Software Development and Process Improvement* (PROFES)
    - *International Conference on Business Process Management* (BPM)
    - *European Conference on Software Process Improvement* (EuroSPI)
    - *The Journal of Systems and Software*
    - *Annals of Software Engineering*
    - *Information and Software Technology*
    - *Journal of Software Engineering Research and Development* (JSERD)
    - *Journal of Universal Computer Science* (JUCS)
- Escrita da tese de doutorado.

7.4 Cronograma Proposto

Para a obtenção dos resultados esperados neste trabalho, a Tabela 7.1 mostra a distribuição das atividades previstas para os próximos meses:

Tabela 7.1: Cronograma de atividades para esta proposta de tese

Atividade	**Ano**	**2015**				**2016**												**2017**		
	Mês	09	10	11	12	01	02	03	04	05	06	07	08	09	10	11	12	01	02	03
1. Exame de Qualificação																				
2. Construção do sistema para suporte à abordagem																				
2.1 Módulo para definição das medidas do processo																				
2.2 Módulo para execução e monitoramento do processo																				
2.3 Módulo para análise da execução do processo																				
2.4 Módulo para realimentação do processo																				
3. Aplicação da proposta a exemplos de processos																				
4. Planejamento e execução de experimento para avaliação																				
5. Reexecução da Busca Estruturada																				
6. Redação de Artigos																				
7. Escrita da tese de doutorado																				
8. Defesa da tese de doutorado																				

REFERÊNCIAS BIBLIOGRÁFICAS

ACUNA, S. T., DE ANTONIO, A., FERRE, X., et al., 2000, "The Software Process: Modelling, Evaluation and Improvement", *Handbook of Software Engineering and Knowledge Engineering*, pp. 1–35.

ALEIXO, F., FREIRE, M. A., SANTOS, W., KULESZA, U., 2010, "Uma Abordagem para Gerência e Customização de Variabilidades em Processos de Software". In: Proceedings of the XXIV Simpósio Brasileiro de Engenharia de Software (SBES´2010), Salvador, Brasil, pp.118-127.

ALEIXO, F. A., FREIRE, M., ALENCAR, D., CAMPOS, E., KULESZA, U., 2012, "A Comparative Study of Compositional and Annotative Modelling Approaches for Software Process Lines". In: 26th Brazilian Symposium on Software Engineering (SBES), Natal, RN, Brasil, pp. 51-60.

ALEIXO, F., 2013, "Uma Abordagem Anotativa para Gerência de Variabilidades em Linhas de Processos de Software: Concepção, Implementação e Avaliação". Tese de Doutorado, UFRN, Natal, Brasil.

AVERSANO, L., DE LUCIA, A., GAETA, M., RITROVATO, P., STEFANUCCI, S., VILLANI, M. L., 2004, "Managing coordination and cooperation in distributed software processes: the GENESIS environment". Software Process: Improvement and Practice, v. 9, n. 4, pp. 239-26.

AVRILIONIS, D., BELKHATIT, N., CUNIN, P.Y., 1996, "A Unified Framework for Software Process Enactment and Improvement". In: *Proceedings of the 4th International Conference on Software Process* (ICSP4), Brighton, UK, pp.102-111.

BALDASSARRE, M. T., CAIVANO, D., VISAGGIO, G., 2006, "Non Invasive Monitoring of a Distributed Maintenance Process". In: Instrumentation and Measurement Technology Conference, IMTC 2006. Proceedings of the IEEE. pp. 1098-1103.

BARCELLOS, M.P., 2009, "Uma estratégia para medição de software e avaliação de bases de medidas para controle estatístico de processos de software em organizações de alta maturidade", Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, Brasil.

BARRETO, A. S., 2011, "Uma Abordagem para Definição de Processos baseada em Reutilização Visando à Alta Maturidade em Processos". Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

BASILI, V. R., 1992, "Software modeling and measurement: The Goal/Question/Metric paradigm", Tech. Rep. CS-TR- 2956, Department of Computer Science, University of Maryland, College Park, MD 20742.

BELHAJJAME, K., DEUS, H., GARIJO, D., KLYNE, G., MISSIER, P., SOILAND-REYES, S., ZEDNIK, S., 2012, Prov model primer. Disponível em: http://www.w3.org/TR/prov-primer. Acesso em: abr 2015.

BELHAJJAME, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., Zhao, J., 2013, PROV-O: The PROV Ontology. Disponível em: http://www.w3.org/TR/prov-o. Acesso em: abr 2015.

BIVAR, B., SANTOS, L., KOHWALTER, T. C., MARINHO, A., MATTOSO, M., BRAGANHOLO, V., 2013. Uma Comparação entre os Modelos de Proveniência OPM e PROV, In: eScience Workshop, 2013, Maceió, pp. 1-8.

BIZAGI, 2015. Disponível em http://www.bizagi.com/. Acesso em: 10 jan 2015.

BONITASOFT, 2015. Bonita BPM, BPM de código aberto. Disponível em: http://br.bonitasoft.com/produtos/bonita-bpm-bpm-de-codigo-aberto. Acesso em: 10 jan 2015.

BUNEMAN, P., KHANNA, S., TAN, W. C, 2001, "Why and where: A characterization of data provenance". In: 8th International Conference on Database Theory, Londres, pp. 4-6.

CALLAHAN, S. P., FREIRE, J., SCHEIDEGGER, C. E., SILVA, C. T., VO, H. T., INC, V., 2008, Towards process provenance for existing applications. In Proceedings of the 2nd International Provenance and Annotation Workshop, pp.120–127.

CEOSOFTWARE, 2015. Soluções criativas e inovadoras. Disponível em: http://www.ceosoftware.com.br/. Acesso em: 1 jun. 2015.

COLOMBO, A., DAMIANI, E., FRATI, F., OLTOLINA, S., REED, K., RUFFATTI, G., 2008, "The use of a meta-model to support multi-project process measurement". In: Software Engineering Conference, APSEC'08, 15th Asia-Pacific, IEEE, pp. 503-510.

COULETTE B., CRÉGUT X., DONG T. B. T., TRAN D. T., 2000, "RHODES, a Process Component Centered Software Engineering Environment", In: Proceedings of the 2nd International Conference on Enterprise Information System (ICEIS), Stafford, UK, pp. 253-260.

COSTA, G. C. B., SANTANA, F., MAGDALENO, A. M., WERNER, C. M. L., 2014, "Monitoring Collaboration in Software Processes Using Social Networks". In: International Conference on Collaboration and Technology (CRIWG), 2014, Santiago, v. 8658. pp. 89-96.

COSTA, L., 2009, "Formulação de uma metodologia de modelagem de processos de negócio para implementação de workflow". Dissertação de mestrado em Engenharia de Produção, Universidade Tecnológica Federal do Paraná, Ponta Grossa, Brasil.

CRÉGUT, X., COULETTE, B., 1997, "PBOOL: an object-oriented language for definition and reuse of enactable processes". *Software-Concepts and Tools*, v. 18, n. 2, pp. 47-62.

CULVER-LOZO, K., 1994, "Rapid iteration in software process improvement: experience report". In: Software Process, 1994. Applying the Software Process, Proceedings, Third International Conference on the. IEEE, 1994. pp. 79-84.

DERNIAME, J. C., KABA, B. A., WASTELL, D., 1999, "Software Process: Principles, Methodology and Technology". Berlin: Springer-Verlag, Lecture Notes in Computer Science, 1500.

DAI, F., LI, T., 2007a, "Composing Software Evolution Process Component". In: Proceedings of the 7th International Symposium on Advanced Parallel Processing Technologies, Guangzhou, China, pp. 684-692.

DAI, F., LI, T., 2007b, "Tailoring Software Evolution Process". In: International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2007), Qingdao, China, pp. 782 – 787.

DAI, F., LI, T., ZHAO, N., YU, Y., HUANG, B., 2008, "Evolution Process Component Composition Based on Process Architecture". In: International Symposium on Intelligent Information Technology Application Workshops (IITAW '08), Shanghai, China, pp. 1097-110.

DUSTDAR, S., HOFFMANN, T., VAN DER AALST, W, 2004, Mining of ad-hoc business with TeamLog, Distributed Systems Group, Technical University of Vienna.

ECLARUS, 2015. eClarus Business Process Modeler for Business Analysts. Disponível em http://eclarus-business-process-modeler-for-bus.software.informer.com/. Acesso em: 23 jan 2015.

FACT++, 2015, FaCT++ Reasoner. Disponível em: http://owl.cs.manchester.ac.uk/tools/fact/. Acesso em: mai 2015.

FALBO, R. A., 1998, *Integração de Conhecimento em um Ambiente de Desenvolvimento de Software*. Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

FERREIRA, F. M. L. *et al*, 2008, A method for stochastic modeling the software development process in constrained resource environments. In: Systems and Information Engineering Design Symposium (SIEDS 2008), Charlottesville, VA, pp.129-134.

FLUIG, 2015. Disponível em http://www.fluig.com/. Acesso em: jan. 2015.

FREIRE, J., KOOP, D., SANTOS, E., SILVA, C. T., 2008, "Provenance for Computational Tasks: A Survey". Computing in Science and Engineering, vol. 10, no. 3, pp. 11-21.

FUGGETTA, A., 2000, “Software process: a roadmap”. In: *Proceedings of the Conference on The Future of Software Engineering*, pp. 25–34, Limerick, Ireland.

GARG, P. K., MI, P., PHAM, T., SCACCHI, W., THUNQUEST, G., 1994, “The SMART approach for software process engineering”. In Proceedings of the 16th international conference on Software engineering, IEEE Computer Society Press, pp. 341-350.

GARY, K., LINDQUIST, T., 1999a, “Distributed Architectures for Process Component Support”. In: Proceedings of the 5th International Conference on Information Systems Analysis and Synthesis (ISAS'99), Orlando, August.

GARY, K.A., LINDQUIST, T. E., 1999b, “Cooperating Process Components”. In: International Computer Software and Applications Conference (COMPSAC), pp. 218-223, Phoenix, United States, October.

GHOSHAL, D., PLALE, B, 2013, “Provenance from log files: a BigData problem”. In Proceedings of the Joint EDBT/ICDT 2013 Workshops (EDBT '13). ACM, New York, NY, USA, 290-297. DOI=10.1145/2457317.2457366 http://doi.acm.org/10.1145/2457317.2457366.

GROTH, P., MOREAU, L., 2013, “PROV - Overview”. Disponível em: <http://www.w3.org/TR/2013/NOTE-prov-overview-20130430/>. Acesso em: jul 2015.

GRUNDY, J., MUGRIDGE, W. B., HOSKING, J. G., 1997, “Utilising past event histories in a process-centred software engineering environment”. In: Software Engineering Conference, Proceedings., Australian. IEEE, pp. 127-136.

HALL C., HARMON P., MIERS D, 2007, The 2007 BPM Suites Report-2.1 – BPTrends.

HALLERBACH, A., BAUER, T., REICHERT, M., 2008, “Managing Process Variants in the Process Lifecycle”. In: 10th Int’l Conf. on Enterprise Information Systems (ICEIS’08). Barcelona, Spain, pp. 154–161.

HERMIT, 2015, HermiT OWL Reasoner. Disponível em: http://hermit-reasoner.com/. Acesso em: mai 2015.

HUO, M., ZHANG, H., JEFFERY, R., 2006, A systematic approach to process enactment analysis as input to software process improvement or tailoring. In: APSEC 2006: ASIA-PACIFIC SOFTWARE ENGINEERING CONFERENCE, 2006, Kanpur, pp. 401-410.

HUO, M., ZHANG, H., JEFFERY, R., 2008, Detection of consistent patterns from process enactment data. In *Making Globally Distributed Software Development a Success Story*, Springer Berlin Heidelberg, pp. 173-185.

INTALIO, 2015. Disponível em http://www.intalio.com/. Acesso em: jan. 2015.

ISO/IEC, 2004, ISO/IEC 15504 - Software Engineering - Process Assessment, The International Organization for Standardization and the International Electrotechnical Commission.

ISO/IEC, 2008, ISO/IEC 12207 – Systems and Software Engineering – Software Life Cycle Process, The International Organization for Standardization and the International Electrotechnical Commission.

JAUFMAN, O., MÜNCH, J., 2005, "Acquisition of a Project-Specific Process", In: Proceedings of the 6th International Conference on Product-Focused Software Process Improvement, pp. 328-342, Oulu, Finland, June.

JØRGENSEN, H.D., 2000, "Software Process Model Reuse and Learning". In: Process Support for Distributed Team-based Software Development, pp. 726-731, Orlando, Estados Unidos, Julho.

KEDJI, K. A., THAT, M. T. T., COULETTE, B., LBATH, R., TRAN, H. N., NASSAR, M., 2011, "Towards a tool-supported approach for collaborative process modeling and enactment". In: Software Engineering Conference (APSEC), 2011 18th Asia Pacific, IEEE, pp. 414-421.

KURNIAWATI, F., JEFFERY, R., 2006, The use and effects of an electronic process guide and experience repository: a longitudinal study, Information and Software Technology, Volume 48, Issue 7, July 2006, pp. 566-577.

LIM, C., LU, S., CHEBOTKO, A., FOTOUHI, F, 2010, "Prospective and Retrospective Provenance Collection in Scientific Workflow Environments". In Proceedings of the 2010 IEEE International Conference on Services Computing (SCC '10). IEEE Computer Society, Washington, DC, USA, pp. 449-456.

MACIEL, R. S. P., SILVA, B. C. D., MAGALHÃES, P. F., ROSA, N. S., 2009, "An integrated approach for model driven process modeling and enactment". In Software Engineering, 2009. SBES'09. XXIII Brazilian Symposium on, IEEE, pp. 104-114.

MAHNIC, V., VRANA, I., 2007, "Using stakeholder driven process performance measurement for monitoring the performance of a Scrum-based software development process". Electrotechnical Review, v. 74, n. 5, pp. 241-247.

MARINHO, A. S., 2011. PROVMANAGER: *Uma Abordagem para Gerenciamento de Proveniência de Experimentos Científicos*. Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

MARTÍNEZ-RUIZ, T., RUIZ, F., PIATTINI, M., 2013, "Towards Understanding Software Process Variability from Contextual Evidence of Change". In Advanced Information Systems Engineering Workshops, pp. 417-431. Springer Berlin Heidelberg.

MICROTOOL, 2014, In-STEP BLUE. Disponível em: http://www.microtool.de/en/in-step-blue-project-management-software/ Acesso em: nov. 2014.

MILES, S., GROTH, P., MUNROE, S., MOREAU, L., 2011, "PrIMe: A methodology for developing provenance-aware applications". ACM Trans. Softw. Eng. Methodol. 20, 3, Article 8 (August 2011), 42 pages. DOI=10.1145/2000791.2000792 http://doi.acm.org/10.1145/2000791.2000792

MINGHUI, W., JING, Y., CHUNYAN, Y., 2004, "A methodology and its support environment for benchmark-based adaptable software process improvement", Systems, Man and Cybernetics, 2004 IEEE International Conference, vol.6, pp.5183-5188 vol.6, 10-13 Oct. 2004.

MISSIER, P., DEY, S., BELHAJJAME, K., CUEVAS-VICENTTIN, V., LUDAESCHER, B., 2013, "D-prov: extending the prov provenance model with workflow structure". In: Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance, TaPP 2013, pp. 9:1–9:7.

MOREAU, L., FREIRE, J., FUTRELLE, J., MCGRATH, R. E., MYERS, J., PAULSON, P., 2008, "The Open Provenance Model: An Overview". In *Provenance and Annotation of Data and Processes*, Juliana Freire, David Koop, and Luc Moreau (Eds.). Lecture Notes In Computer Science, Vol. 5272. Springer-Verlag, Berlin, Heidelberg pp. 323-326.

MOREAU, L., CLIFFORD, B., FREIRE, J., FUTRELLE, J., GIL, Y., GROTH, P., KWASNIKOWSKA, N., MILES, S., MISSIER, P., MYERS, J. *et al.*, 2011, The Open Provenance Model core specification (v1.1). Future Generation Computer Systems, v.27 n. 6, pp.743 –756.

NGUYEN, M. N., CONRADI, R., 1994, "Classification of Meta-processes and their Models". In: Proceedings of Third International Conference on the Applying Software Process, pp. 167-175.

OMG, 2005, "Unified Modeling Language: Superstructure", UML Superstructure Specification v2.0, formal/05-07-04, Object Management Group.

OMG, 2008, "Software & Systems Process Engineering Metamodel (SPEM)". Disponível em: http://www.omg.org/spec/SPEM/. Acesso em: mar. 2015.

OMG, 2011, "Business Process Model and Notation (BPMN) Version 2.0". Disponível em: http://www.omg.org/spec/BPMN/2.0/. Acesso em: mar. 2015.

OWL, 2009, OWL 2 Web Ontology Language Primer: Disponível em: http://www.w3.org/TR/2009/WD-owl2-primer-20090421/#Property_Chains. Acesso em: mai 2015.

PAULK, M. C., 2009, A History of the Capability Maturity Model for Software, Technical Report, American Society for Quality (ASQ).

PILLAT, R. M., OLIVEIRA, T. C., ALENCAR, P. S., COWAN, D. D., 2015. BPMNt: A BPMN extension for specifying software process tailoring. *Information and Software Technology*, v.57, pp. 95-115.

PUNTAR, S., IENDRIKE, H. DOS S., MAGDALENO, A. M.; *et al.*, 2009. *Estudo Conceitual sobre BPMS*, Relatório Técnico 0007/2009, RelaTe-DIA, UNIRIO. Disponível em: http://seer.unirio.br/index.php/monografiasppgi.

PUTNAM, L. H., MYERS, W., 2003. *Five Core Metrics: The Intelligence Behind Successful Software Management*, Dorset House Publishing Company.

RED HAT, 2015. jBPM - Open Source Business Process Management - Process Engine. Disponível em: http://www.jbpm.org/ Acesso em: jan. 2015.

REICHERT, M., 2000, Dynamic Changes in Process Management Systems. Tese de Doutorado, University of Ulm, Faculty of Computer Science, July 2000 (in German).

REIS, C.A., 2003, Uma Abordagem Flexível para Execução de Processos de Software Evolutivos. Tese de Doutorado, PPGC/UFRGS, Porto Alegre, RS, Brasil.

RIBEIRO, T. V., 2014, Alinhando Perspectivas de Qualidade em Código Fonte a partir de Estudos Experimentais – Um Caso na Indústria. Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

ROCHA, A. R., SANTOS, G., BARCELLOS, M. P., 2012, Medição de Software e Controle Estatístico de Processos. Ministério da Ciência, Tecnologia e Inovação (MCTI), SEPIN/PBQP Software - Brasília - DF, Brasil. Disponível em http://nemo.inf.ufes.br/files/Livro_Medicao_CEP.pdf.

SANTANA, F., 2015, "Evoltrack-Process: Um Ambiente para a Visualização da Colaboração em Processos de Software". Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

SEI, 2010, Capability Maturity Model Integration (CMMI) for Development, version 1.3, Carnegie Mellon University, Software Engineering Institute, Technical Report CMU/SEI-2010-TR-033.

SIMMHAN, Y. L., 2007, "Provenance framework in support of data quality estimation". Tese de Doutorado, Indiana University, Indianapolis, USA.

SIMMONDS, J., BASTARRICA, M. C., SILVESTRE, L., QUISPE, A., 2012, "Modeling Variability in Software Process Models". Technical Report. Santiago, Chile, Universidad de Chile.

SIRIN, E., PARSIA, B., GRAU, B. C., KALYANPUR, A., KATZ, Y., 2007, Pellet: A practical owl-dl reasoner. Web Semantics: science, services and agents on the World Wide Web, v. 5, n. 2, p. 51-53.

SOFTEX, 2011a, “MPS.BR – Melhoria de Processo do Software Brasileiro – Guia Geral”. Disponível em: http://www.softex.br/mpsbr.

SOMMERVILLE, I., 2004, *Software Engineering*. 7 ed. Addison Wesley.

SOYLEMEZ, M., TARHAN, A, 2013, “Using Process Enactment Data Analysis to Support Orthogonal Defect Classification for Software Process Improvement”. In Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2013 Joint Conference of the 23rd International Workshop on, IEEE, pp. 120-125.

TANG, J., JIANG, M., ZHU, Q, 2012, “Towards Quantitative Assessment Model for Software Process Improvement in Small Organization”. *Information Technology Journal* v. 11, n. 1, pp. 49-57.

TARHAN, A., DEMIRORS, O., 2011, “Assessment of Software Process and Metrics to Support Quantitative Understanding: Experience from an Undefined Task Management Process”. In: Software Process Improvement and Capability Determination. Springer Berlin Heidelberg, pp. 108-120

TEIXEIRA, E. N., 2014, “ODYSSEYPROCESSREUSE: Uma Proposta de Sistemática de Engenharia de Linha Processo de Software Baseada em um Projeto Arquitetural de Componentes de Processos De Software”. Exame de Qualificação, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

TORTORELLA M., VISAGGIO G., 1997, “CREP - Characterising Reverse Engineering Process components methodology”, In: Proceedings of International Conference on Software Maintenance, Bari, Italy, October 1 - 3, IEEE Comp. Soc. Press, pp. 222-231.

TRAVASSOS, G. H.; GUROV, D.; AMARAL, E. A. G., 2002, Introdução à Engenharia de Software Experimental, Relatório Técnico ES-590/02, PESC-COPPE.

WASHIZAKI, H., 2006, “Building software process line architectures from bottom up”. In: *7th International Conference on Product-Focused Software Process Improvement, PROFES 2006, June 12 - June 14*, v. 4034 LNCS, p.415-421, Amsterdam, Netherlands.

WERNER, C. M. L.; ARAUJO, R. M. DE; SANTORO, F. M.; et al., 2011, CDSoft: Balanceando Colaboração e Disciplina em Processos de Desenvolvimento de Software. Projeto de Pesquisa, Conselho Nacional de Desenvolvimento Científico e Tecnológico, Rio de Janeiro, RJ, Brasil.

WFMC, 2002, Workflow Management Coalition Workflow Standard: Workflow Process Definition Interface – XML Process Definition Language (XPDL), Technical Report, Workflow Management Coalition, Lighthouse Point, Florida, USA.

APÊNDICE A – Trabalhos Analisados na Revisão Bibliográfica Inicial

Este apêndice tem como objetivo citar os 48 trabalhos apresentados na revisão proposta por TEIXEIRA (2014), aos quais foram aplicados os filtros definidos na Seção 3.1 para análise de trabalhos que pudessem estra relacionados a esta proposta de tese. Os trabalhos em destaque foram selecionados para leitura completa e extração de dados para a revisão bibliográfica inicial desta proposta. Embora tenham sido selecionados 16 artigos, pode-se dizer que apenas 13 propostas diferentes entre si foram apresentadas, haja vista que alguns trabalhos de mesmos autores apenas apresentam pequenas evoluções no trabalho exposto em artigo anterior. Estes trabalhos encontram-se listados na ordem em que aparecem na proposta de TEIXEIRA (2014).

1	SUTTON, S., OSTERWEIL, L., 1996, "Product families and process families". In: Proceedings of the 10th International Software Process Workshop (ISPW), pp. 109-111, Dijon, France.
2	DURÁN, A., BENAVIDES, D., BERMEJO, J., 2004, "Applying system families concepts to requirements engineering process definition". In: Software Product-Family Engineering, pp. 140-151. Springer Berlin Heidelberg.
3	JAUFMAN, O., MÜNCH, J., 2005, "Acquisition of a Project-Specific Process", In: Proceedings of the 6th International Conference on Product-Focused Software Process Improvement, pp. 328-342, Oulu, Finland, June.
4	ROMBACH, D., 2006, "Integrated Software Process and Product Lines". In: Unifying the Software Process Spectrum, Berlin/Heidelberg: Springer-Verlag, pp. 83-90.
5	ROMBACH, D., 2013, "Integrated Software Process and Product Lines". In Perspectives on the Future of Software Engineering, pp. 359-366. Springer Berlin Heidelberg.
6	WASHIZAKI, H., 2006a, "Building software process line architectures from bottom up". In: Proceedings of the 7th International Conference on Product-Focused Software Process Improvement, PROFES 2006, pp. 415-421, Amsterdam, Netherlands, June.
7	WASHIZAKI, H., 2006b, "Deriving project-specific processes from process line architecture with commonality and variability". In: 2006 IEEE International Conference on Industrial Informatics, INDIN'06, August 16 - August 18, pp. 1301-1306, Singapore, Singapore.
8	SIMIDCHIEVA, B.I., CLARKE, L.A., OSTERWEIL, L., 2007, "Representing Process Variation with a Process Family". In: International Conference on Software Process, Lecture Notes in Computer Science v. 4470, pp. 109-120, Minneapolis, Estados Unidos, Maio.
9	HALLERBACH, A., BAUER, T., REICHERT, M., 2008, "Managing Process Variants in the Process Lifecycle". In: 10th Int'l Conf. on Enterprise Information Systems (ICEIS'08). Barcelona, Spain, pp. 154–161.
10	ARMBRUST, O., KATAHIRA, M., MIYAMOTO, Y. et al., 2008, "Scoping Software Process Models - Initial Concepts and Experience from Defining Space Standards", In: Proceedings of the International Conference on Making Globally Distributed Software Development a Success Story, Berlin / Heidelberg: Springer, pp. 160-172.
11	ARMBRUST, O., KATAHIRA, M., MIYAMOTO, Y. et al., 2009, "Scoping Software Process Lines". In: Software Process: Improvement and Practice, v. 14, Issue 3, pp. 181-197, New York, NY, USA.

12	TERNITÉ, T., 2009, "Process Lines: A Product Line Approach Designed for Process Model Development". In: Software Engineering and Advanced Applications. SEAA '09. 35th Euromicro Conference on, pp. 173-180, 27-29 Aug.
13	SIMMONDS, J., BASTARRICA, M. C., SILVESTRE, L. E QUISPE, A., 2011, "Analyzing Methodologies and Tools for Specifying Variability in Software Processes". Santiago, Chile, Universidad de Chile.
14	SIMMONDS, J., BASTARRICA, M. C., 2011, "Modeling Variability in Software Process Lines". Santiago, Chile, Universidad de Chile.
15	SIMMONDS, J., BASTARRICA, M. C., SILVESTRE, L., QUISPE, A., 2012, "Modeling Variability in Software Process Models". Technical Report. Santiago, Chile, Universidad de Chile.
16	BARRETO, A. S., MURTA, L. G. P., ROCHA, A. R., 2009, "Componentizando Processos Legados de Software Visando a Reutilização de Processos", In: VIII Simpósio Brasileiro de Qualidade de Software, pp. 189-203, Ouro Preto, MG, Brasil, Junho.
17	BARRETO, A. S., NUNES, E., ROCHA, A. R. C., MURTA, L. G. P., 2010, "Supporting the Definition of Software Processes at Consulting Organizations via Software Process Lines". In: Proceedings of the International Conference on the Quality of Information and Communications Technology (QUATIC), pp. 15-24, Porto, Portugal.
18	BARRETO, A. S., 2011, "Uma Abordagem para Definição de Processos baseada em Reutilização Visando à Alta Maturidade em Processos". Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.
19	ALEIXO, F., FREIRE, M. A., SANTOS, W., KULESZA, U., 2010, "Uma Abordagem para Gerência e Customização de Variabilidades em Processos de Software". In: Proceedings of the XXIV Simpósio Brasileiro de Engenharia de Software (SBES´2010), Salvador, Brasil.
20	ALEIXO, F. A., FREIRE, M., ALENCAR, D., CAMPOS, E., KULESZA, U., 2012, "A Comparative Study of Compositional and Annotative Modelling Approaches for Software Process Lines". In: 26th Brazilian Symposium on Software Engineering (SBES), Natal, RN, Brasil, pp. 51-60.
21	ALEIXO, F., 2013, "Uma Abordagem Anotativa para Gerência de Variabilidades em Linhas de Processos de Software: Concepção, Implementação e Avaliação". Tese de Doutorado, UFRN, Natal, Brasil.
22	ALEGRÍA, J. A. H., BASTARRICA, M. C., QUISPE, A., OCHOA, S. F., 2011, "An MDE approach to software process tailoring". In: Proceedings of the 2011 International Conference on Software and Systems Process, Honolulu, Havaí, pp. 43-52.
23	ALEGRÍA, J.A.H., BASTARRICA, M.C., 2012, "Building software process lines with CASPER". In: Proceedings of International Conference on Software and System Process (ICSSP), pp. 170 –179.
24	ALEGRÍA, J. A. H., BASTARRICA, M. C., QUISPE, A., OCHOA, S. F., 2014, "MDE-based process tailoring strategy". Journal of Software: Evolution and Process, v. 26, n. 4, pp. 386-403.
25	MAGDALENO, A. M., DE ARAUJO, R. M., & WERNER, C., 2012, "COMPOOTIM: An Approach to Software Processes Composition and Optimization". In CIbSE, Buenos Aires, Argentina, pp. 42-55.
26	MARTÍNEZ-RUIZ, T., GARCÍA, F., PIATTINI, M., 2008, "Towards a SPEM v2. 0 extension to define process lines variability mechanisms". In: Software engineering research, management and applications, pp. 115-130. Springer Berlin Heidelberg.
27	MARTÍNEZ-RUIZ, T., GARCÍA, F., PIATTINI, M., 2011, "Managing process diversity by applying rationale management in variant rich processes". In: Product-Focused Software Process Improvement, pp. 128-142. Springer Berlin Heidelberg.
28	MARTÍNEZ-RUIZ, T., RUIZ, F., PIATTINI, M., 2013, "Towards Understanding Software Process Variability from Contextual Evidence of Change". In Advanced

	Information Systems Engineering Workshops, pp. 417-431. Springer Berlin Heidelberg.
29	GARY, K., DERNIAME, J.C., LINDQUIST, T., KOEHNEMANN, H., 1998, "Component-Based Software Process Support". In: Proceedings of the 13th Conference on Automated Software Engineering (ASE'98), Honolulu, Hawaii, October.
30	GARY, K., LINDQUIST, T., 1999a, "Distributed Architectures for Process Component Support". In: Proceedings of the 5th International Conference on Information Systems Analysis and Synthesis (ISAS'99), Orlando.
31	GARY, K.A., LINDQUIST, T. E., 1999b, "Cooperating Process Components". In: International Computer Software and Applications Conference (COMPSAC), Phoenix, United States, pp. 218-223.
32	RU-ZHI et al., 2003, "Research on matching algorithm for XML-based software component query", Journal of Software, July, pp.1195-1202.
33	RU-ZHI, X., TAO, H., DONG-SHENG, C., et al., 2005, "Reuse-Oriented Process Component Representation and Retrieval". In: International Conference on Computer and Information Technology, pp. 911-315, Shangai, China, September.
34	RU-ZHI, X., PEIGUANG L., ZHIKUN Z., LEQIU Q., 2010, "An Approach of Reuse-based Software Process Improvement". In: Journal of Computational Information Systems Volume 6 Number 6, pp. 1897-1906.
35	BHUTA, J.; BOEHM, B.; MEYERS, S., 2005, "Process Elements: Components of Software Process Architectures." In: Proceedings of the Software Process Workshop, pp. 332-346, Beijing, China.
36	DAI, F., LI, T., 2007a, "Composing Software Evolution Process Component". In: Proceedings of the 7th International Symposium on Advanced Parallel Processing Technologies (APPT'2007), Guangzhou, China, pp. 684-692.
37	DAI, F., LI, T., 2007b, "Tailoring Software Evolution Process". In: Proceedings of the 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, (SNPD'2007), Qingdao, China, pp. 782-787.
38	DAI, F., LI, T., ZHAO, N., YU, Y., HUANG, B., 2008, "Evolution Process Component Composition Based on Process Architecture". In: International Symposium on Intelligent Information Technology Application Workshops (IITAW'08), pp. 1097 – 110.
39	SEGRINI, B. M., 2009, "Definição de Processos Baseada em Componentes". Dissertação de mestrado, Universidade Federal do Espírito Santo, Vitória, ES, Brasil.
40	LANNA, A. L. P. M, 2009, "Reuso de Processos de Software Baseado na Componentização de Processos e Conhecimento". Dissertação de mestrado em Engenharia Elétrica, PUC-MG, Belo Horizonte, Brasil.
41	LANNA, A. L. P. M., PIETROBON, C. A. M., 2010, "Desenvolvimento de um Método e uma Ferramenta para a Reutilização de Processos de Desenvolvimento de Software". Qualidade e Produtividade em Software Projetos Ciclo.
42	OMG, 2008, "Software Process Engineering Metamodel". Disponível em: http://www.omg.org/technology/documents/formal/spem.htm. Acesso em: dez 2014.
43	AVRILIONIS, D., BELKHATIT, N., CUNIN, P.Y., 1996, "A Unified Framework for Software Process Enactment and Improvement". In: Proceedings of the 4th International Conference on Software Process (ICSP4), Brighton, UK, pp. 102-111.
44	COULETTE B., CRÉGUT X., DONG T. B. T., TRAN D. T., 2000, "RHODES, a Process Component Centered Software Engineering Environment", In: Proceedings of the 2nd International Conference on Enterprise Information System (ICEIS'2000), Stafford, UK, pp. 253-260.
45	COULETTE, B., CRÉGUT X., DONG T. B. T., TRAN D. T., 2001, "Managing process through a base of reusable components", In: Proceedings of the 3rd International Conference on Enterprise Information Systems (ICEIS'2001), Setubal, Portugal, pp. 608-618.
46	IIDA H., TANAKA Y., 2002, "A Compositional Process Pattern Framework for

	Component-based Process Modeling Assistance", In: Proceedings of the 1st Workshop on Software Development Process Patterns (SDPP'02), Seattle, Washington, USA, pp. 53-61.
47	TORTORELLA M., VISAGGIO G., 1997, "CREP - Characterising Reverse Engineering Process components methodology", In: Proceedings of International Conference on Software Maintenance, Bari, Italy, October 1 - 3, IEEE Comp. Soc. Press, pp. 222-231.
48	FUSARO, P., VISAGGIO, G., TORTORELLA, M., 1998, "REP - ChaRacterizing and Exploiting Process Components: Results of Experimentation". In: Working Conference on Reverse Engineering, pp. 20-29, Honolulu, United States, October.

APÊNDICE B – Informações dos Artigos Incluídos na Busca Estruturada

Este apêndice tem como objetivo apresentar todas as informações que foram extraídas dos artigos selecionados na busca estruturada, a partir do entendimento da pesquisadora. Foram adotados como campos de extração de dados os seguintes:

(C1) Descrição de como é a realizada a execução do processo, quais os procedimentos, ferramentas utilizadas e dados armazenados;

(C2) Descrição de como é realizado o monitoramento do processo, quais os procedimentos, ferramentas utilizadas e dados monitorados;

(C3) Descrição de como é a realizada a avaliação do processo, quais os procedimentos, ferramentas utilizadas e dados avaliados;

(C4) Descrição de como é a realizada a realimentação do processo, quais os procedimentos, ferramentas e dados utilizados.

ARTIGO 1 - Ferreira *et al.*, 2008	
Título:	A Method for Stochastic Modeling the Software Development Process in Constrained Resource Environments
Autores:	Fernando Machado Lima Ferreira, Eber Assis Schmitz, Antonio Juarez Alencar, Fabio Protti and Carlos Henrique Alves
Ano de Publicação:	2008
Resumo	
A melhoria de processo exige uma definição formal do mesmo e medidas associadas ao desempenho. Assim, os processos podem ser diagnosticados, a fim de formular hipóteses sobre onde devem ser feitas alterações. Posteriormente, novos processos podem ser concebidos e os desenvolvedores podem verificar se as melhorias desejadas foram efetivamente atingidas. A simulação é uma ferramenta de baixo custo e poderosa para diagnóstico e teste de várias alternativas de melhoria, antes de testes reais. Este artigo apresenta um método baseado em UML para obter a distribuição de probabilidade do tempo de execução de uma grande variedade de processos de negócios, incluindo desenvolvimento de software. Tal método, que se baseia na simulação de Monte Carlo, permite a identificação de fatores que influenciam mais fortemente o tempo de execução do processo, favorecendo mudanças que aumentam a eficiência do processo com considerável impacto as táticas e estratégias de negócios.	
C1	Não aborda.
C2	O processo em execução não é monitorado, haja vista que a abordagem proposta é baseada em dados previamente simulados.
C3	O método proposto faz uma descrição do processo usando UML. A simulação do processo baseia-se no método de Monte Carlo, com o objetivo de descobrir a duração do processo de desenvolvimento de software, tendo-se como princípio que processos de software podem ser considerados como processos estocásticos, haja vista que as métricas de processo podem ser caracterizadas como variáveis aleatórias. Assim, a duração de um processo, de acordo com a abordagem, é descrita usando uma função de distribuição de probabilidade (PDF - *probability distribution function*).

C4	Não aborda.
ARTIGO DE CONTROLE	
ARTIGOS 2 e 3 - Huo *et al*., 2006, 2008	
Título:	A Systematic Approach to Process Enactment Analysis as Input to Software Process Improvement or Tailoring (2006) Detection of Consistent Patterns from Process Enactment Data (2008)
Autores:	Ming Huo, He Zhang, Ross Jeffery
Ano de Publicação:	2006, 2008
Resumo	
A melhoria do processo de software tem sido um foco da indústria por muitos anos. Para auxiliar o processo e implementação de melhoria de processo, este trabalho apresenta um método para a recuperação de processo de software com base na mineração dos dados de execução do projeto. O objetivo do método proposto é revelar o processo real utilizado, visando proporcionar dados de entrada para melhorar a qualidade de um processo de software definido. O modelo de recuperação (ou padrões) está no mesmo nível de abstração do modelo de processo pré-definido. Isso fornece uma maneira fácil e clara para identificar a diferença entre o modelo de processo planejado e o que foi executado. Este trabalho investiga a execução de um processo de software definido a partir do ponto de vista dos gerentes de projeto, no contexto de uma pequena organização de desenvolvimento de software. Foram coletados dados de organizações e aos quais foi aplicado o método proposto para um estudo de caso-piloto. A principal contribuição do trabalho é proporcionar um método de recuperação de modelo de processo de software que suporta a mudança de processo de software e melhoria.	
C1	Não é citada nenhuma ferramenta ou máquina de execução específica. O autor cita apenas a utilização do EPG (*electronic process guide/experience repository*) (KURNIAWATI e JEFFERY, 2006) para controle dos processos de software, não deixando claro se este possui as funcionalidades necessárias para execução de processos de software. Como tipos de informações armazenadas sobre os processos no EPG, pode-se citar: data de início e término das tarefas; o executor da tarefa, comentários do executor a respeito da tarefa e a qual projeto específico aquela tarefa pertence.
C2	Os dados de execução do exemplo apresentado neste artigo são armazenados em um repositório chamado EPG (KURNIAWATI e JEFFERY, 2006). O próprio EPG fornece uma visualização dos modelos do processo de forma gráfica, utilizando notações UML.
C3	É feita uma mineração dos dados de execução extraídos do EPG e estes são formalmente representados através de redes de Petri. A partir deste ponto, as 'redes que representam as execuções do processo' são comparadas com o processo que foi definido a priori, permitindo a detecção de problemas e possibilidades de melhoria no mesmo.
C4	Não aborda.

ARTIGO DE CONTROLE	
ARTIGO 4 - Avrilionis *et al*., 1996	
Título:	A Unified Framework for Software Process Enactment and Improvement
Autores:	Denis Avrilionis, Noureddine Belkhatir, Pierre-Yves Cunin
Ano de Publicação:	1996
Resumo	

A maioria dos trabalhos realizados anteriormente em processos de software abordado o problema da modelagem do processo como uma descrição monolítica, com foco em uma perspectiva, por exemplo, produto, processo, papel, etc. Pouco foi feito para resolver o problema de melhoria de processo de software através da composição dinâmica de componentes consistentes e reutilizáveis. O objetivo deste trabalho consiste em propor um *framework* que integra estes aspectos de forma unificada. Este trabalho é o resultado da experiência de cinco anos dos autores com os sistemas do próprio grupo, sendo eles: Tempo/Adele e Opsis. O *framework* proposta conta com uma abordagem baseada em componentes para construir diferentes níveis de visões que podem ser refinados e expandidos durante a execução do processo.	
C1	Para a execução de processos o autor cita que a abordagem Pynode foi implementada como uma extensão do Opsis System, uma ferramenta própria, que permite a execução e evolução de modelos de processos.
C2	O presente trabalho não aborda claramente a etapa de monitoramento da execução do processo. O autor apenas sugere o uso de visões de observação (assim como foram criadas as visões de execução). Uma visão de observação reuniria informações sobre diferentes visões de execução, porém, sem nenhuma influência direta sobre estas. Diferentes ferramentas, tais como ferramentas de medição ou ferramentas de gerenciamento de configuração, poderiam explorar estas visões de observação, contudo, a definição destas visões encontra-se em desenvolvimento, segundo o autor.
C3	Não aborda.
C4	Não aborda.

ARTIGO 5 – Maciel *et al.*, 2009	
Título:	An Integrated Approach for Model Driven Process Modeling and Enactment
Autores:	Rita Suzana Pitangueira Maciel, Bruno Carreiro da Silva, Ana Patrícia Fontes Magalhães, Nelson Souto Rosa
Ano de Publicação:	2009
Resumo	
A adoção do MDA no desenvolvimento de software vem aumentado e esta abordagem vem sendo amplamente reconhecido como uma abordagem importante para a construção de sistemas de software. Porém, nota-se a falta de uma terminologia e uma notação padrão que aborde os aspectos de projeto (concepção) de um processo MDA. As ferramentas e ambientes MDA disponíveis são particularmente focadas na definição e execução de transformações de modelos, enquanto um processo de desenvolvimento envolve outras definições importantes que devem ser levadas em consideração durante a execução do processo. Este artigo apresenta uma abordagem integrada para modelagem e execução de processos MDA baseado em especializações dos conceitos do SPEM. Para apoiar e avaliar a abordagem, a ferramenta desenvolvida foi aplicada em dois estudos de caso.	
C1	Um ambiente chamado *Transforms* foi desenvolvido e utilizado para apoiar tanto a modelagem como a execução de processos MDA. Este ambiente é dividido em dois módulos principais: módulo para modelagem e módulo para execução de processos MDA. O módulo para execução possibilita o registo dos profissionais de uma equipe de software e os papéis atribuídos a cada um deles, além da possibilidade de visualizar todas as definições de processo (fases, iterações, artefatos etc.). Fornece também uma visão para gerenciamento das tarefas, seus respectivos estados e artefatos do processo durante a execução de projetos de software MDA. No final do processo de execução, o *Transforms* mostra todos os modelos criados pela equipe, códigos e modelos gerados pelas transformações e o histórico das tarefas (sejam estas

	executadas manualmente ou automaticamente).
C2	O ambiente *Transforms* permite a possibilidade de visualizar todas as definições de processo (fases, iterações, artefatos etc.), incluindo as tarefas e seu estado durante a execução de um processo MDA.
C3	Não aborda
C4	Não aborda

ARTIGO 6 - Tarhan e Demirors, 2011	
Título:	Assessment of Software Process and Metrics to Support Quantitative Understanding: Experience from an Undefined Task Management Process
Autores:	Ayca Tarhan, Onur Demirors
Ano de Publicação:	2011
Resumo	
O gerenciamento na engenharia de software exige a medição, avaliação e melhoria dos processos e produtos de software. No entanto, a utilização de medição e análise em engenharia de software não é muito simples. Ela exige conhecimento sobre os conceitos de medição, gestão de processos e estatísticas, bem como sobre as suas aplicações práticas. Este trabalho apresenta uma abordagem sistemática para avaliar a adequação de um processo de software e as suas medidas de análise quantitativa, aplicando a mesma em diversos contextos industriais. Este artigo explica a experiência de avaliar um processo de gerenciamento de tarefas e medidas relacionadas de uma agência de pesquisa do governo. A agência não tinha definido os processos de gerenciamento de tarefas e de medição, e os dados de desempenho foram recolhidas a partir de uma ferramenta de gerenciamento de mudanças. Foram gastos seis pessoas/dias avaliando e analisando 92 execuções de processo.	
C1	Não aborda.
C2	Não aborda.
C3	Para a avaliação do processo é proposta a abordagem A^2QPM (*Assessment Approach for Quantitative Process Management*) que avalia a 'capacidade de duração estimada de um processo', usando uma análise quantitativa e gráficos XmR.
C4	Não aborda.

ARTIGO 7 - Aversano *et al*., 2004	
Título:	Managing Coordination and Cooperation in Distributed Software Processes: the GENESIS Environment
Autores:	Lerina Aversano, Andrea De Lucia, Matteo Gaeta, Pierluigi Ritrovato, Silvio Stefanucci, Maria Luisa Villani.
Ano de Publicação:	2004
Resumo	
É apresentada a plataforma GENESIS (ambiente generalizado para o gerenciamento de processos de engenharia de software cooperativa), que foi o resultado de um projeto de pesquisa com o objetivo de projetar e desenvolver um sistema não invasivo e de código aberto para apoiar os processos de engenharia de software em um ambiente altamente distribuído. O sistema suporta a cooperação e a coordenação em processos de software e sua linguagem de modelagem de processos permite a decomposição de processos complexos em subprocessos que podem ser distribuídos e executados em diferentes locais da organização. Na plataforma GENESIS, tecnologias de gerenciamento de	

workflows foram integradas com gerenciamento de artefatos e serviços de comunicação para atender os requisitos necessários à gestão da cooperação entre equipes distribuídas. Seus pontos fortes são um poderoso gerenciamento de atividades, abrangendo todos os principais aspectos do ciclo de vida de uma atividade; um projeto de monitoramento eficiente e flexível, coletando métricas de produtividade e qualidade, mostrando, sob demanda, todo o processo e suas partes em diferentes níveis de detalhe, e uma consideração cuidadosa das questões relativas à evolução do processo, permitindo gerir adequadamente as exceções mais comuns que ocorrem durante a execução do processo, de uma forma simples e flexível.	
C1	A própria plataforma GENESIS possui um sistema de gerenciamento de workflow que permite a definição, instanciação e execução de processos de software. Este sistema é composto por quatro componentes principais (máquina de workflow, manipulador da lista de tarefas, ferramenta para a definição do processo e ferramenta para gerenciamento do projeto) que compartilham um banco de dados comum. A máquina de workflow lê as regras de transição do processo definido e as executa, permitindo assim o avanço da instância do processo. Além disso, a máquina de execução é a responsável pela criação de novas instâncias de atividades e de lhes atribuir aos trabalhadores especificados no modelo de processo concreto. A máquina de execução também interage com o mecanismo de evento para propagar eventos que são relevantes para a execução de processos cooperantes executados em outros sites.
C2	O suporte ao monitoramento do projeto é fornecido pela ferramenta de visualização do processo e pela ferramenta GEM (**GE**NESIS **M**easurements). Ambos os módulos interpretam os dados do processo gravadas através do componente 'manipulador da lista de tarefas' durante a execução. A ferramenta de visualização de processo possibilita a visualização em tempo real da instância de processo, mostrando o estado de cada atividade, os artefatos produzidos, os atores atuais, etc.
C3	A combinação dos dados de execução com os dados de estimativa sobre a duração do processo, o esforço para cada atividade, e artefato a ser produzido, armazenado anteriormente através da ferramenta de gerenciamento de projetos, permite um controle eficaz das restrições de agendamento usando um mecanismo de mensagens de alerta.
C4	Na plataforma GENESIS, a evolução dos processos pode ser efetivamente suportada pela política de operações de mudança, permitindo construir, de forma incremental, os modelos de processo. Ou seja, através da plataforma é possível fazer um modelo de processo inicial e executar as suas instâncias, e, em seguida, detalhar o modelo de processo assim que mais conhecimento e informações estiverem disponíveis, como, por exemplo, a adição de novas atividades e transições. As novas instâncias de processo correspondentes serão modificadas automaticamente após as alterações.

ARTIGO 8 – Baldassarre *et al*., 2006	
Título:	Non Invasive Monitoring of a Distributed Maintenance Process
Autores:	Maria Teresa Baldassarre, Danilo Caivano, Giuseppe Visaggio
Ano de Publicação:	2006
Resumo	
É bem conhecido que a medição de processos impacta na performance do desenvolvedor. Além disso, a fim de ser eficaz, ela deve estar em conformidade com as características do contexto. Ambos os	

aspectos são fundamentais para os processos de software distribuídos globalmente devido à distância e a heterogeneidade entre os sites monitorados. É ressaltado que a medição deve ser não invasiva e a interpretação dos resultados deve ser flexível, no que diz respeito a cada contexto local. Este artigo aborda que a qualidade dos processos primários pode ser medida, de forma não invasiva, através da interpretação dos dados recolhidos durante a execução do processo. Este trabalho aborda estas questões críticas com foco em processos de manutenção. O artigo apresenta uma abordagem de Controle Estatístico de Processos (SPC) não invasiva, para a medição de um processo primário (manutenção). A eficácia da abordagem é investigada através de uma simulação efetuada em dados legados coletados em um ambiente industrial.	
C1	Não aborda.
C2	Não aborda.
C3	Este trabalho apresenta o uso de Controle Estatístico de Processos (SPC) como uma técnica de análise dos dados de execuções anteriores do processo. Esta técnica utiliza dados do processo para o estabelecimento de limites de controle realistas. Os limites fixados tornam-se limites operacionais para apontar situações anômalas e variações nas performances do processo. Eles são utilizados para monitorar o desempenho real do processo e são automaticamente ajustados para se adaptar a cada vez que o processo começa a tornar-se instável. Essas análises permitem fazer previsões sobre o processo que está sendo monitorado e controlado.
C4	Não aborda

ARTIGO 9 – Culver-Lozo, 1994	
Título:	Rapid Iteration in Software Process Improvement: Experience Report
Autores:	Kathleen Culver-Lozo
Ano de Publicação:	1994
Resumo	
Organizações de desenvolvimento de software mudam seus processos para tirar proveito de novas tecnologias, corrigir erros, ou para produzir novos produtos. Este trabalho descreve as experiências de uma organização de desenvolvimento de software para melhorar o seu processo. Ao concentrar-se em criar definições de processos utilizáveis por aqueles que executam o processo e através do recolhimento de dados sobre as melhorias necessárias durante a execução do processo, a organização introduziu centenas de mudanças no processo e incorporou essas mudanças em suas operações.	
C1	Apenas são citados diagramas de PERT e diagramas de fluxo de dados (DFDs), para entendimento da execução do processo (quais os fluxos existentes no processo, que entradas são necessárias para cada tarefa e quais saídas são produzidas).
C2	O artigo apresenta que são armazenados dados qualitativos e quantitativos do processo executado. É citado, por exemplo, que os membros da equipe registram o tempo gasto em cada tarefa, porém, não são apresentadas quais ferramentas são usadas para dar suporte a esta etapa do ciclo de vida do processo.
C3	Não aborda.
C4	Não aborda.

ARTIGO 10 – Garg *et al.*, 1994	
Título:	The SMART Approach for Software Process Engineering
Autores:	Pankaj K. Garg, Peiwei Mi, Thuan Pham, Walt Scacchi, Gary Thunquest
Ano de Publicação:	1994
Resumo	
Este artigo descreve uma metodologia para engenharia de processo de software e um ambiente para apoiá-la, chamado SMART. Este ambiente oferece suporte ao ciclo de vida de um processo, incluindo a modelagem, análise e execução de processos de software. A funcionalidade de monitoramento de processos no ambiente SMART pode ser usada para fornecer *feedback* do processo de execução com relação ao modelo de processo. Este ambiente representa a integração de três mecanismos de processo que foram desenvolvidas separadamente e utiliza dois formalismos para modelagem do processo (representação de dados orientados a objetos e linguagens de programação imperativa) para preencher a lacuna entre a modelagem de processos, análise e execução. SMART é um meta-ambiente que aceita um modelo de processo como sua especificação de entrada, e, em seguida, gera um programa de processo que produz um executável PSEE (*Process-Centered Software Engineering Environments*). Além disto, o SMART oferece suporte a uma abordagem orientada para a equipe de modelagem de processos, análise e execução.	
C1	A execução de processos no ambiente SMART é suportada através da ferramenta de execução de processos da HP, chamada SynerVision. Esta ferramenta apresenta ao usuário uma lista de tarefas a serem executadas, possibilitando a alteração de seu status, além da inclusão de novas tarefas.
C2	O monitoramento do processo também é realizado através da ferramenta SynerVision, que permite que os usuários do processo visualizem uma estimativa / faixa de tempo para a conclusão da tarefa. Um mecanismo de relógio controla a quantidade de tempo gasto ao executar uma tarefa. A fim de assegurar a privacidade de dados de tempo, o tempo que um utilizador gastou em uma tarefa particular é visível apenas a esse utilizador. O usuário pode, a seu critério escolher compartilhar essas informações com outros membros da equipe. Além da informação do tempo gasto em uma atividade, a ferramenta SynerVision permite monitorar também as ferramentas invocadas, os usuários que trabalharam em tarefas diferentes e todas as notas de usuários ou feedback adicionados a uma determinada tarefa. Esses dados são disponibilizados para os usuários em diversos formatos de relatórios.
C3	Não aborda.
C4	Não aborda.

ARTIGO 11 – Colombo *et al.*, 2008	
Título:	The use of a meta-model to support multi-project process measurement
Autores:	Alberto Colombo, Ernesto Damiani, Fulvio Frati, Sergio Oltolina, Karl Reed, Gabriele Ruffatti
Ano de Publicação:	2008
Resumo	
Empresas de software estão envolvidas em vários projetos que utilizam plataformas heterogêneas e uma ampla quantidade de aplicações em domínios de aplicação distintos. Muitas vezes estes projetos envolvem o desenvolvimento de sistemas de software incluindo parceiros e clientes, bem como os	

próprios funcionários da empresa. Assim, é preciso apoiar o desenvolvimento de software através de múltiplos processos, enquanto busca-se garantir níveis uniformes de execução e qualidade dos produtos em todos eles. A abordagem apresentada neste trabalho fornece mecanismos para a medição do processo em um ambiente multi-processo. Baseia-se em uma meta-modelo, no qual métricas de vários projetos são estipuladas para permitir monitorar o cumprimento do CMMI. A ferramenta Spago4Q foi desenvolvida para apoiar a abordagem e é capaz de produzir as medições necessárias para o monitoramento de um conjunto de projetos de desenvolvimento em grande escala usando diferentes modelos de processos, em um ambiente industrial real na Europa. Os resultados apoiam a visão de que nem sempre é possível agregar o mesmo conjunto de métricas em modelos de processos distintos.	
C1	Não aborda.
C2	Não aborda.
C3	É apresentada a plataforma de código aberto chamada Spago4Q (SpagoBI for Quality), que pode ser adaptada para contextos organizacionais complexos, independentemente dos processos de desenvolvimento de software, ferramentas de infraestrutura, medição e quadros de avaliação utilizados. Esta ferramenta permite a avaliação da maturidade e eficácia do processo de desenvolvimento de software. Isto é alcançado através da avaliação de dados e medidas coletados a partir do gerenciamento de projetos e ferramentas de desenvolvimento com técnicas não-invasivas.
C4	Não aborda.

ARTIGO 12 – Kedji *et al.*, 2011	
Título:	Towards a tool-supported approach for collaborative process modeling and enactment
Autores:	Komlan Akpédjé Kedji, Minh Tu Ton That, Bernard Coulette, Redouane Lbath, Hanh Nhi Tran, Mahmoud Nassar
Ano de Publicação:	2011
Resumo	
Em engenharia de software, como em qualquer empreendimento coletivo, a compreensão e apoio à colaboração é uma grande preocupação. Infelizmente, os principais conceitos e formalismos de processos nem sempre são adequados para descrever a colaboração. Assim, foi criada uma extensão para o meta-modelo SPEM, através da introdução de conceitos necessários para expressar configurações de colaboração precisas e dinâmicas que os profissionais criam para enfrentar os desafios. Este trabalho tem como objetivo dar aos profissionais a capacidade de expressar a compreensão sobre a colaboração em um formalismo adequado para facilitar a representação e a assistência prestada pela ferramenta. Este trabalho é baseado em um meta-modelo de processo colaborativo desenvolvido pelos autores. Neste trabalho é apresentado um meta-processo para a modelagem de processos e execução, ao qual se aplica o meta-modelo de processo colaborativo. Em seguida, descreve-se a implementação de um editor de modelo de processo e um gerador de plano de projeto a partir de modelos de processos.	
C1	Esta meta-processo descrito destaca duas maneiras de permitir a execução do processo modelado: usando um PSEE (*Process centered Software Engineering Environment*) ou usando uma ferramenta de gerenciamento de projetos de prateleira, tais como MS Project e Gantt Project.
C2	Não aborda.
C3	Não aborda.

C4	Não aborda.

ARTIGO 13 – Tang *et al*., 2012	
Título:	Towards Quantitative Assessment Model for Software Process Improvement in Small Organization
Autores:	Jingfan Tang, Ming Jiang, Qin Zhu
Ano de Publicação:	2012
Resumo	
A Avaliação de Processos de Softwares (SPA - *Software Process Assessment*) é realizada antes da Melhoria do Processo de Software (SPI - Software Process Improvement), que possui atividades que permitem identificar as áreas do processo a serem melhoradas. Os métodos de avaliação de processos, tais como ISO / IEC 15504-2 e SCAMPI estão disponíveis para todas as empresas, porém, são considerados de difícil implementação para a maioria das pequenas organizações, devido à complexidade e ao grande investimento em termos de tempo e recursos. A fim de realizar as atividades de SPA com pouco esforço em pequenas organizações e de forma eficaz, este trabalho apresenta um modelo de avaliação quantitativa. É proposta uma abordagem eficiente para avaliar a qualidade da execução do processo nos projetos para a melhoria da maturidade da organização. As atividades do projeto e as entregas foram identificados bem como os itens de avaliação da organização que poderiam ser adaptados para cada projeto específico. Foi atribuído um peso aos itens de avaliação adaptados e às atividades de revisão correspondentes, de acordo com a prioridade destes. Através de uma abordagem quantitativa eficaz na avaliação orientada a 'marcos' (*milestones*), a pontuação relativa à conformidade pode ser obtida com base na caracterização dos resultados da avaliação do documento de inspeção e sessão de entrevista. Os resultados iniciais mostraram que as práticas do modelo de avaliação apresentado são apropriadas para pequenas organizações, apoiando a avaliação de processos e a atividades de melhoria.	
C1	Não aborda.
C2	Não aborda.
C3	É descrito um modelo de avaliação quantitativa que pode ser personalizado para ajudar pequenas organizações a realizar uma avaliação eficaz de seus processos a um custo razoável. Os itens de avaliação podem ser definidos de acordo com os requisitos específicos da organização, abrangendo as atividades de todo o ciclo de vida de desenvolvimento do software, visando garantir a qualidade dos produtos finais. Um peso é atribuído aos itens de avaliação e às atividades de revisão correspondentes, visando atender a meta de pontuação da avaliação. Utilizando documento de inspeção e sessão de entrevista com as regras de avaliação quantitativas, a pontuação final de conformidade pode ser obtidas para cada etapa do projeto, como pode ser usada para identificar as potenciais oportunidades de melhoria do processo de software na pequena organização.
C4	Não aborda.

ARTIGO DE CONTROLE	
ARTIGO 14 – Soylemez e Tarhan, 2013	
Título:	Using Process Enactment Data Analysis to Support Orthogonal Defect Classification for Software Process Improvement
Autores:	Mehmet Soylemez, Ayça Tarhan

Ano de Publicação:	2013
Resumo	
Defeitos ocorrem em muitos projetos de desenvolvimento de software. É importante extrair informação semânticas relativas aos defeitos para investigar as suas causas e melhorar o processo. Neste estudo, os dados de execução de um processo de desenvolvimento de software em que foram originados defeitos, foi usado para apoiar a classificação de defeitos de forma ortogonal ODC (*Orthogonal Defect Classification*). Um projeto de uma organização CMMI ML3 foi usado como exemplo para a aplicação da abordagem ODC e os dados de execução do processo auxiliaram na descoberta sobre as causas dos defeitos e decorrentes ações de melhoria. Os atributos dos defeitos foram analisados e comparados antes e após a aplicação de medidas de melhoria sugeridas. A comparação entre o defito inicial e a melhoria realizada mostraram que houve uma mudança positiva no processo de desenvolvimento de software.	
C1	Os dados de execução do processo, a partir dos quais os defeitos coletados, são obtidos por meio de formulários pré-definidos (formulários de atributos do processo), prenchidos por desenvolvedores durante o desenvolvimento do projeto.
C2	Não aborda.
C3	A técnica de ODC foi usada para avaliar e melhorar o processo de desenvolvimento de software, sendo as causas dos defeitos identificadas a partir da avaliação dos resultados dos dados de execução do processo, que mostram a origem dos defeitos. Sugestões de melhorias podem ser feitas para eliminar as causas básicas e aplicadas em uma nova fase do desenvolvimento.
C4	Não aborda.

ARTIGO 15 – Mahnic e Vrana, 2007	
Título:	Using stakeholder-driven process performance measurement for monitoring the performance of a Scrum-based software development process
Autores:	Viljan Mahnic, Ivan Vrana
Ano de Publicação:	2007
Resumo	
É descrito um plano de métricas para monitorar e melhorar o desempenho do processo de desenvolvimento de software com base no método ágil Scrum. Após uma breve introdução aos conceitos de Scrum, uma descrição detalhada das métricas propostas é fornecida. As métricas são definidas usando os princípios de medição de desempenho processo focada no *stakeholder*, que requer uma abordagem equilibrada, considerando os pontos de vista das diferentes partes interessadas. As metas de cada parte interessada são definidas primeiro, seguido pela escolha de indicadores de desempenho adequados. A avaliação de cada indicador é baseada em valores de métricas coletadas durante a execução do processo. O plano de medidas permite uma introdução gradual de métricas que podem ser incorporadas ao Scrum sem afetar a agilidade do processo de desenvolvimento.	
C1	Não aborda.
C2	Não aborda.
C3	É proposto um conjunto de métricas que fornecem uma visão abrangente do desempenho do projeto. A definição destas métricas é baseado no conceito de um sistema de medição do desempenho do processo que defende uma abordagem equilibrada, considerando pontos de vista das diferentes partes interessadas que participam no processo. O plano de métricas proposto considera os pontos de vista de três partes interessadas: os gerentes de TI, os

	membros da equipe e os clientes. As métricas são definidas através de uma abordagem *top-down* semelhante ao método GQM (Goal/Question/Metric). As metas de cada parte interessada são definidas primeiro, seguido pela escolha de indicadores de desempenho adequados. Finalmente, as métricas que permitem a avaliação de cada indicador são definidas. Os indicadores propostos descrevem o processo quantitativamente e qualitativamente, proporcionando assim uma visão holística do desempenho do processo. A fim de preservar a agilidade, todas as métricas propostas na secção anterior (com exceção do número de erros reportados pelo utilizador após a liberação do sistema) são definidas de tal maneira que possam ser recolhidas durante as reuniões já previstas pela Scrum, não requerendo um esforço adicional substancial da equipe.
C4	Não aborda.

ARTIGO 16 – Grundy *et al*., 1997	
Título:	Utilising Past Event Histories in a Process-Centred Software Engineering Environment
Autores:	John C. Grundy, Warwick B. Mugridge, John G. Hosking
Ano de Publicação:	1997
Resumo	
Ao trabalhar em sistemas de software complexos, muitas vezes é difícil coordenar o trabalho de vários desenvolvedores e o uso múltiplo de ferramentas/softwares no processo. Ambientes de engenharia de software voltados para o processo visam ajudar os desenvolvedores a gerenciar as complexidades envolvidas na coordenação de um processo de software codificando os passos e o contexto da atividade realizada pelo desenvolvedor (ou seja, os artefatos, ferramentas e colaboradores necessários ao desenvolvedor durante o seu trabalho). Infelizmente, a maioria dos ambientes centrados em processos não suportam adequadamente a coordenação do trabalho, a facilidade de utilização e a melhoria dos modelos de processos. Assim, este trabalho utiliza 'histórias' de acontecimentos passados do ambiente centrado no processo para dar aos desenvolvedores informações extras para orientar o desenvolvimento de software colaborativo. São descritas técnicas para a determinação das 'histórias de trabalho' e para suportar a visualização da coordenação de trabalho, execução do processo e melhoria do mesmo.	
C1	Para a execução do processo é utilizado o ambiente Serendipity, que é um ambiente de modelagem, execução e planejamento do trabalho em processos, que também suporta a manipulação de eventos, comunicação em grupo e facilidades voltadas ao *awareness* do grupo.
C2	O ambiente Serendipity também é utilizado para monitor o processo, através da gravação de histórias de trabalho, que contém as atividades de um desenvolvedor ao realizar uma tarefa no processo.
C3	O ambiente Serendipity fornece visualizações relativas ao contexto do processo executado que podem auxiliar o responsável pelo processo a determinar melhorias a serem implementadas no mesmo.
C4	Não aborda.