UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Daniel Dias de Carvalho

UM METAMODELO PARA A REPRESENTAÇÃO DE LINHA DE PROCESSOS DE SOFTWARE

Belém
2015

Daniel Dias de Carvalho

UM METAMODELO PARA A REPRESENTAÇÃO DE LINHA DE PROCESSOS DE SOFTWARE

Dissertação de Mestrado apresentada para obtenção do grau de Mestre em Ciência da Computação. Programa de Pós-Graduação em Ciência da Computação. Instituto de Ciências Exatas e Naturais. Universidade Federal do Pará. Área de concentração Engenharia de Software. Orientador: Prof. Dr. Cleidson Ronald Botelho de Souza.

Belém
2015

Dados Internacionais de Catalogação-na-Publicação (CIP)
Sistema de Bibliotecas da UFPA

Carvalho, Daniel Dias de, 1985-
Um metamodelo para a representação de linha de processos de software / Daniel Dias de Carvalho. - 2015.

Orientador: Cleidson Ronald Botelho de Souza.
Dissertação (Mestrado) - Universidade Federal do Pará, Instituto de Ciências Exatas e Naturais, Programa de Pós-Graduação em Ciência da Computação, Belém, 2015.

1. Engenharia de software. 2. Software-Reutilização. 3. Processos de software. 4. Revisão sistemática da literatura. I. Título.

CDD 22. ed. 005.1

Daniel Dias de Carvalho

UM METAMODELO PARA A REPRESENTAÇÃO DE LINHA DE PROCESSOS DE SOFTWARE

Dissertação de Mestrado apresentada para obtenção do grau de Mestre em Ciência da Computação. Programa de Pós-Graduação em Ciência da Computação. Instituto de Ciências Exatas e Naturais. Universidade Federal do Pará.

Data da aprovação: 27 de Outubro de 2015

Banca Examinadora

Programa de Pós-Graduação em Ciência da Computação – UFPA – Orientador

Prof. Dr. Cleidson Ronald Botelho de Souza

Programa de Pós-Graduação em Ciência da Computação – UFPA – Membro

Profª. Drª. Carla Alessandra Lima Reis

Programa de Pós-Graduação em Ciência da Computação – UFPA – Membro

Prof. Dr. Rodrigo Quites Reis

Dedico este trabalho aos meus pais,
Carlos e Raimunda.

AGRADECIMENTOS

Agradeço a Deus pelos dias passados, por hoje e pelos dias que virão; pelos dias de chuva e pelos dias de sol; pelo milagre da vida.

Aos meus pais, Carlos e Raimunda, por sempre me incentivarem e me ensinarem a dar valor à educação.

Aos meus irmãos, Rafael e Gabriel, pelo companheirismo.

À Larissa Chagas por ter feito parte da minha vida por todos esses anos, sempre demonstrando dedicação, companheirismo e respeito; por ter contribuído para a realização deste trabalho, ajudando em todos os aspectos.

À Luciana Nascimento pela atenção e dedicação, sempre me fazendo refletir sobre os assuntos abordados na pesquisa.

Ao Anderson Costa por ter sido meu coorientador informal no início, pelo auxílio nas apresentações e pelas conversas descontraídas.

Ao Adailton Lima, que contribuiu diretamente com este trabalho, sendo coautor de artigos científicos publicados com resultados parciais da pesquisa e apresentando-os em conferências internacionais.

Aos meus colegas do LABES/UFPA por proporcionarem um ambiente muito agradável nos laboratórios, aliando estudos, conversas sobre temas variados e confraternizações. Ter estado em um ambiente de pesquisas científicas foi muito importante para conhecer relações entre diversas áreas e perceber diferentes visões sobre um determinado tema.

À professora Carla Lima pelos conhecimentos repassados, conselhos e, principalmente, por ter me oferecido a oportunidade de ingressar no mestrado.

Ao professor Rodrigo Quites, que sempre acompanhou o desenvolvimento do trabalho, participando de reuniões, apresentações e sugerindo melhorias.

Ao professor Cleidson de Souza por ter aceitado orientar este trabalho e ter revisado artigos publicados.

Aos professores do PPGCC/UFPA pelos conhecimentos e experiências compartilhados durante as aulas, contribuindo para a fundamentação teórica necessária para o trabalho. Muitos

conhecimentos adquiridos não foram utilizados diretamente na pesquisa, mas serão extremamente úteis para minha vida acadêmica e profissional.

À CAPES pelo apoio financeiro, que foi fundamental para a realização deste trabalho.

À Universidade Federal do Pará por ter me proporcionado estudo gratuito e de qualidade desde a graduação, especialização até o mestrado.

RESUMO

A Reutilização de processos de software é essencial para a melhoria da qualidade e diminuição dos esforços necessários para a definição de processos, além de promover a disseminação de conhecimentos dentro de uma organização de software. O paradigma de Linha de Processos de Software vem se destacando como uma alternativa viável para a reutilização de processos através da representação de similaridades e variações em uma única estrutura de processos. Neste sentido, esta dissertação de mestrado apresenta uma abordagem para a representação em linha de processos de software. A abordagem é composta por: um metamodelo para a representação de variações, características e dependências entre os elementos de processo que constituem uma arquitetura de linha de processos de software; uma notação gráfica, baseada em uma linguagem de modelagem de processos existente; e um processo de engenharia de linha de processos de software, composto por atividades de engenharia de domínio, engenharia de aplicação e de gerência, que guiam a utilização dos conceitos referentes ao metamodelo. Para fundamentar a pesquisa, foi realizada uma revisão sistemática da literatura, em que limitações na área foram identificadas e requisitos para a representação de linha de processos foram coletados. A abordagem proposta neste trabalho foi avaliada das seguintes formas: primeiro, através da publicação de artigos científicos com resultados parciais da pesquisa; e, finalmente, pela utilização da abordagem por outra dissertação de mestrado, no qual uma linha de processos foi modelada com a utilização da abordagem proposta neste trabalho e avaliada por especialistas.

Palavras-chave: Engenharia de Software, Reutilização de Processos de Software, Linha de Processos de Software, Modelo de Características, Revisão Sistemática da Literatura.

ABSTRACT

Software Process Reuse is essential for improving the quality and decrease the effort required to define processes and promote the dissemination of knowledge within a software organization. The Software Process Line paradigm has emerged as a feasible way of process reuse, by representing similarities and variations in a core process structure. In this regard, this work presents an approach to support the representation of software process lines. The approach comprises: a metamodel for representing variations, features and dependencies between process elements that constitute a software process line architecture; a graphical notation, based on an existing software processes modeling language; and a software process line engineering process, composed by domain engineering, application engineering and managing activities, guiding the use of the concepts related to the metamodel. In order to support the research, we conducted a systematic literature review, which enabled the identification of limitations in the area and the collection of requirements for representation of software process lines. The approach was evaluated through the following ways: publication of scientific papers with partial results of the research; and the use of the approach by another masters dissertation, in which a process line was modeled using the approach proposed in this work and assessed by experts.

Keywords: Software Engineering, Software Process Reuse, Software Process Line, Feature Model, Systematic Literature Review.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ALPrS	Arquitetura de Linha de Processos de Software
CAPES	Coordenadoria de Aperfeiçoamento de Pessoal de Nível Superior
EA	Engenharia de Aplicação
ED	Engenharia de Domínio
ELPrS	Engenharia de Linha de Processos de Software
LABES	Laboratório de Engenharia de Software
LPrS	Linha de Processos de Software
LPS	Linha de Produtos de Software
MEC	Ministério da Educação
PSEE	*Process-Centred Software Engineering Environment*
RBC	Raciocínio Baseado em Casos
SPEM	*Software Process Engineering Metamodel*
TCC	Trabalho de Conclusão de Curso
UFPA	Universidade Federal do Pará

SUMÁRIO

1. Introdução

Este Capítulo apresenta a motivação, o problema, os objetivos, o método de pesquisa, o contexto e a organização do texto da dissertação.

1.1. Motivação

A demanda por produtos de software vem aumentando e estes, por sua vez, são cada vez mais complexos (BORGES; FALBO, 2001). O desenvolvimento de produtos de software de qualidade, dentro do cronograma e dos custos estimados, sempre foi um desafio para organizações desenvolvedoras de software (BERTOLLO; FALBO, 2003). A Engenharia de Software (ES) tem como um de seus propósitos fornecer meios para melhorar o desenvolvimento e produzir software de alta qualidade (PRESSMAN, 2011).

Diversos autores argumentam que a definição de processos é essencial para produzir software de qualidade (BERTOLLO; FALBO, 2003; FUGGETTA, 2000; ROMBACH, 2005; SUTTON; OSTERWEIL, 1996). Um processo de software bem definido deve indicar as atividades a serem executadas, os recursos requeridos, os produtos de trabalho consumidos e produzidos, os procedimentos a serem adotados (como métodos, técnicas e modelos de documentos) e critérios para execução de atividades (BARRETO, 2011).

Entretanto, a definição de processos de software é uma atividade complexa que requer muita experiência e conhecimento de diversas áreas e disciplinas da engenharia de software (ALEIXO *et al.*, 2010). Uma das abordagens usadas para auxiliar na definição de processos de software é a reutilização de processos (BARRETO, 2011).

Diversas técnicas de reutilização de processos vêm sendo transferidas a partir de técnicas de reutilização de software (BARRETO, 2011; BARRETO *et al.*, 2009; KELLNER, 1996; REIS, 2002; WASHIZAKI, 2006b). Kellner (1996) afirma que técnicas de reutilização de software (como arquiteturas populadas com componentes reutilizáveis, gerência de ativos reutilizáveis e gerência de configuração de ativos reutilizáveis) são aplicáveis para a reutilização de processos de software. Segundo Barreto *et al.* (2009), conceitos como componentes, arquiteturas, linha de

produtos e padrões também têm sido adaptados para serem utilizados no contexto de processos de software.

Diversos pesquisadores (ALEIXO *et al.*, 2010; BARRETO, 2011; TEIXEIRA, 2011; TERNITÉ, 2009) sugerem abordagens de Linha de Processos de Software (LPrS) a partir da aplicação de conceitos de Linha de Produtos de Software (LPS) a processos, fornecendo uma estratégia sistemática para a reutilização de processos de software. O paradigma de LPrS vem ganhando cada vez mais destaque no campo das pesquisas científicas e na indústria de software, tendência essa que se reflete no aumento do número de artigos científicos publicados a cada ano sobre o assunto, como pode ser observado na seção 3.3.5.1.

Diante deste contexto, o paradigma de linha de processos de software será investigado em detalhes no contexto desta dissertação de mestrado como uma forma de reutilização de processos de software.

1.2. Problema

Para a efetividade da adoção do paradigma de LPrS, é de essencial importância a expressividade do metamodelo (SIMMONDS *et al.*, 2013). Um metamodelo é um modelo explícito que incorpora conceitos, relacionamentos, construtores e regras para a construção de entidades em um determinado domínio de interesse (HEIDARI *et al.*, 2013). Cada instância de um metamodelo é um modelo que está em conformidade com o próprio metamodelo.

Após levantar requisitos para a representação de variações em metamodelos de LPrS, foram identificadas limitações das abordagens existentes, como:

- **Especificação de variações em poucos tipos de elementos de processo de software:** A grande maioria dos trabalhos trata apenas de atividades (e conceitos equivalentes) como elementos que sofrem variações;
- **Falta de um estudo detalhado sobre requisitos para a representação de variações:** A maioria dos trabalhos relacionados não seguiu um método rigoroso, com etapas e critérios predefinidos, de coleta de requisitos para a especificação de uma abordagem bem fundamentada;
- **Dependência de uma linguagem de modelagem de processos específica:** Similarmente, a maioria dos trabalhos identificados especifica uma abordagem para a modelagem de variações dependente de uma linguagem específica, ou seja, os conceitos de variações são fortemente acoplados aos conceitos da linguagem de modelagem de processos.

1.3. Objetivos

O objetivo geral desta dissertação de mestrado é apresentar uma abordagem para auxiliar organizações e pesquisadores na representação de variações em processos de software utilizando o paradigma de linha de processos de software. Como objetivos específicos, este trabalho pretende:

- Estabelecer requisitos para a representação de variações;
- Comparar metamodelos existentes em relação aos requisitos estabelecidos;
- Elaborar um metamodelo que satisfaça as limitações dos metamodelos atuais;
- Elaborar uma notação gráfica para representar variações existentes em modelos de processo de software;
- Elaborar um processo de engenharia de linha de processos de software para auxiliar na utilização do metamodelo e da notação gráfica definidos neste trabalho.

1.4. Método de Pesquisa

No contexto deste trabalho, foi seguido um processo composto pelas etapas especificadas na Figura 1.

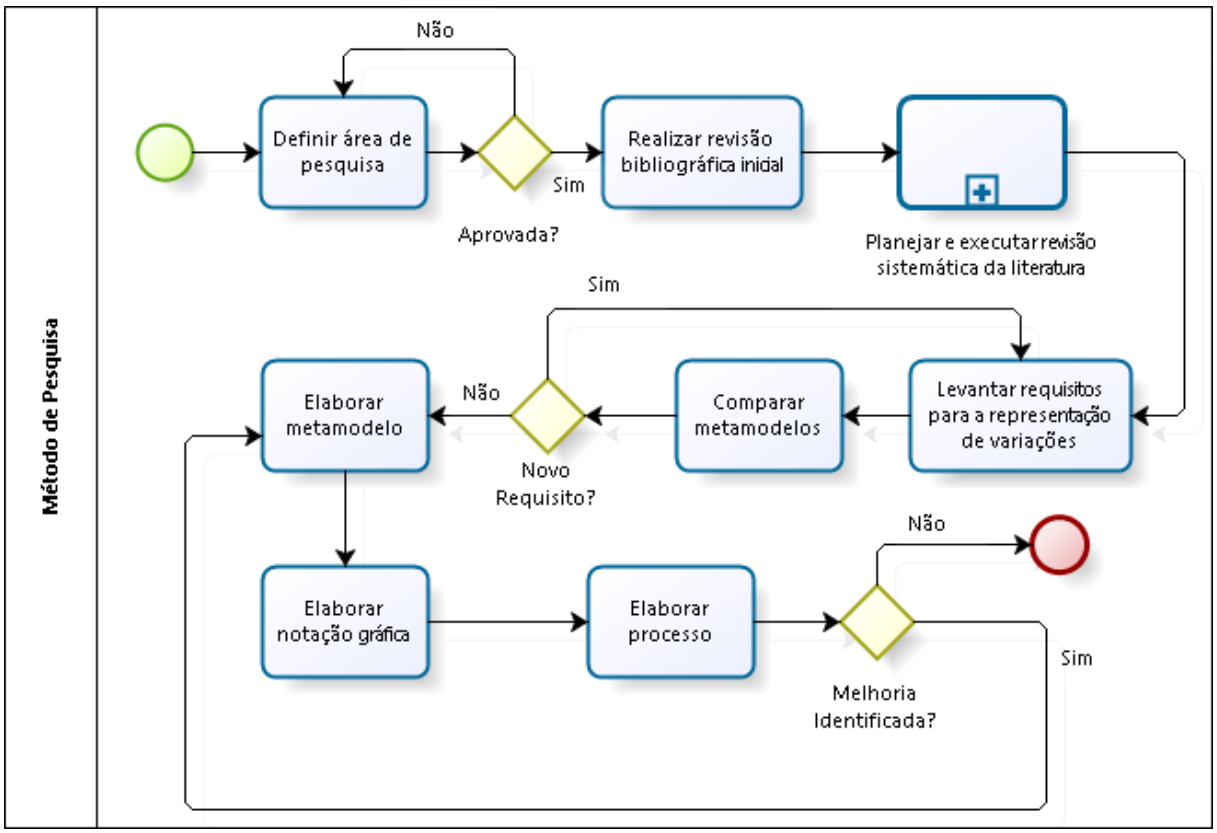


Figura 1 – Método de Pesquisa.

Definir a área de pesquisa: nessa etapa, foi definida que a área de pesquisa seria a reutilização de processos de software. A escolha foi influenciada por trabalhos anteriores (CARVALHO, 2011; COSTA *et al.*, 2007; COSTA, 2010; COSTA; SALES, 2007; REIS, 2002; SALES *et al.*, 2006) realizados no contexto do Laboratório de Engenharia de Software (LABES) da Universidade Federal do Pará (UFPA). A seção 1.5 apresenta o contexto e outros trabalhos que foram realizados antes da pesquisa apresentada nesta dissertação.

Realizar revisão bibliográfica inicial: tendo a área sido escolhida, foi realizada uma revisão inicial da literatura buscando publicações científicas, dissertações de mestrado e teses de doutorado que tratam de aspectos relacionados à área de reutilização de processos de software. Nessa etapa, foi identificada a emergência do paradigma de linha de processos de software como uma abordagem promissora para auxiliar na reutilização de processos, aplicando conceitos de linha de produtos de software a processos.

Planejar e executar revisão sistemática da literatura: a partir da identificação do paradigma de linha de processos de software, foi verificada a necessidade de uma investigação mais detalhada com o propósito de identificar limitações e novas oportunidades sobre o tema. Para isso, foi planejada e executada uma revisão sistemática da literatura seguindo um processo rigoroso, repetível e transparente para a exploração do tema, seguindo o método definido em Kitchenham (2007). O Capítulo 3 descreve de forma detalhada o planejamento, a execução e a análise dos resultados da revisão sistemática da literatura realizada neste trabalho.

Levantar requisitos para a representação de variações: essa etapa consistiu no levantamento de requisitos para a representação de variações em linha de processos de software a partir da análise das publicações retornadas pela revisão sistemática da literatura, complementadas pela análise de outros trabalhos sobre o tema. A seção 4.2 apresenta os requisitos levantados.

Comparar metamodelos: nessa etapa, os metamodelos identificados durante a revisão sistemática da literatura, complementados por outros trabalhos, foram comparados em relação aos requisitos estabelecidos, permitindo a identificação de lacunas e limitações de abordagens existentes. A seção 4.3 descreve em detalhes a comparação entre os metamodelos.

Elaborar metamodelo: essa etapa consistiu na elaboração de um metamodelo para a representação de variações, dependências e características em linha de processos de software que satisfaça aos requisitos levantados. O metamodelo elaborado é descrito em detalhes na seção 5.2 e no APÊNDICE A.

Elaborar notação gráfica: essa etapa consistiu na elaboração de uma notação gráfica, através da adaptação de uma linguagem de modelagem de processos existente para a visualização

de conceitos inerentes ao metamodelo. A seção 5.3 apresenta a notação gráfica elaborada neste trabalho.

Elaborar processo: por fim, foi elaborado um processo de engenharia de linha de processos de software, que engloba as fases de engenharia de domínio, engenharia de aplicação e de gerência. Este processo especifica a sequência de atividades para a utilização adequada dos conceitos presentes no metamodelo. A seção 5.4 apresenta o processo elaborado neste trabalho.

1.5. Contexto

O trabalho apresentado nesta dissertação de mestrado está inserido no contexto de pesquisas realizadas por colaboradores do LABES/UFPA. Uma das linhas de pesquisa do grupo envolve o tema reutilização de processos de software e diversos trabalhos de pesquisa já foram realizados sobre o assunto. Por exemplo, a tese de doutorado de Reis (2002) especificou um metamodelo, denominado de APSEE-Reuse, que teve como objetivo principal aumentar o nível de automação para a reutilização de processos, fornecendo uma série de construtores sintáticos e um formalismo para a modelagem de processos reutilizáveis de forma integrada a outras etapas do ciclo de vida de processos.

O metamodelo APSEE-Reuse foi evoluído no Trabalho de Conclusão de Curso (TCC) de Costa e Sales (2007). Costa *et al.* (2007) descrevem uma implementação do metamodelo APSEE-Reuse, englobando a modelagem de processos reutilizáveis, adaptação e generalização de processos.

Um mecanismo para a recuperação de processos reutilizáveis, através da utilização da técnica de Raciocínio Baseado em Casos (RBC), foi especificado em Sales *et al.* (2006). Tal mecanismo, denominado WebAPSEE-SearchEngine, recupera processos reutilizáveis de um repositório calculando a similaridade com contextos específicos.

A dissertação de mestrado de Costa (2010) e o artigo de Carvalho *et al.* (2011) apresentam um mecanismo para auxiliar na adaptação de processos de software de forma semiautomática a partir da utilização das técnicas de Raciocínio Baseado em Casos e Lógica *Fuzzy*. O mecanismo de adaptação auxilia na escolha de um processo reutilizável adequado para características específicas de um projeto, além de sugerir a remoção e adição de fragmentos de processo.

Complementarmente, os resultados desta dissertação foram utilizados em uma dissertação de mestrado de uma aluna do LABES/UFPA para a modelagem de uma linha de processos de software para a integração de métodos ágeis e modelos de maturidade (CHAGAS, 2015). De forma recíproca, as lições aprendidas de Chagas (2015) foram levadas em consideração para a melhoria contínua dos resultados deste trabalho.

Portanto, o trabalho aqui apresentando se iniciou a partir do estudo de trabalhos anteriores e teve como uma das diretrizes contribuir com as pesquisas de reutilização de processos de software desenvolvidas por alunos, professores e colaboradores do LABES/UFPA.

1.6. Organização do Texto

Neste Capítulo introdutório, foram apresentadas as ideias gerais desta dissertação de mestrado, descrevendo-se a motivação, o problema, os objetivos, o método de pesquisa e o contexto. Além desta introdução, o texto é composto pelos seguintes Capítulos:

- **Capítulo 2:** descreve os conceitos relacionados à disciplina de reutilização de processos de software e ao paradigma de linha de processos de software. O Capítulo também apresenta o ciclo de vida da engenharia de linha de processos de software, que é composto pelas seguintes fases: Engenharia de Domínio, Engenharia de Aplicação e Gerência;
- **Capítulo 3:** apresenta a revisão sistemática da literatura realizada no contexto deste trabalho;
- **Capítulo 4:** estabelece requisitos para a representação de variações, dependências e características em linha de processos de software, a partir da análise de abordagens existentes, e compara metamodelos existentes em relação aos requisitos estabelecidos;
- **Capítulo 5:** nesse Capítulo, a abordagem proposta é descrita em detalhes, sendo composta de um metamodelo, uma notação gráfica e um processo para a engenharia de linha de processos. Por fim, é apresentado um exemplo de modelagem de uma linha de processos de software;
- **Capítulo 6:** esse Capítulo apresenta como a abordagem proposta foi avaliada através da publicação de artigos científicos e pela utilização por outra dissertação de mestrado;
- **Capítulo 7:** conclui o trabalho desenvolvido nesta dissertação, apresentando as considerações finais, as principais contribuições, as limitações e os trabalhos futuros que podem contribuir para a evolução da abordagem proposta.

2. Reutilização de Processos e Linha de Processos de Software

Este Capítulo apresenta conceitos relacionados à reutilização de processos de software e de linha de processos de software, como os elementos estruturais que fazem parte e o método de engenharia seguido para criar processos reutilizáveis e reutilizar processos no contexto desse paradigma.

2.1. Introdução

Um processo de software pode ser conceituado como um conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos necessários para conceber, desenvolver, implantar e manter um produto de software (FUGGETTA, 2000). Segundo Reis (2002), um processo de software é formado por um conjunto de passos parcialmente ordenados, relacionados com conjuntos de artefatos, pessoas, recursos, estruturas organizacionais e restrições tendo como objetivo produzir e manter os produtos de software finais requeridos.

A reutilização de processos de software define uma ampla área de estudo e utilização prática relacionada aos diferentes aspectos envolvidos com a reutilização do conhecimento adquirido na condução de projetos anteriores (REIS, 2002). A Linha de Processos de Software (LPrS)[1] surgiu como uma aplicação do conceito de linha de produtos a processos de desenvolvimento de software (ARMBRUST *et al.*, 2009, p. 6; TERNITÉ, 2009, p. 173; WASHIZAKI, 2006b, p. 1301).

[1] Nesta dissertação, os termos "linha de processos de software" e "linha de processos" e a sigla "LPrS" serão utilizados indistintamente para se referir ao mesmo conceito. A sigla LPrS será utilizada no lugar de LPS para designar Linha de Processos de Software porque a segunda forma já é utilizada em larga escala na literatura para representar o conceito de Linha de Produtos de Software.

O paradigma de LPrS se baseia na reutilização de modelos de processo através da representação de partes similares e variações, que podem ser elementos opcionais, obrigatórios, pontos de variação e variantes.

A literatura relata diversos benefícios alcançados com a representação de variações em modelos de processos de software, dentre os quais podemos citar:

- **Auxílio à tomada de decisões:** a representação explícita de variações em modelos é uma importante estratégia porque, frequentemente, os usuários desses modelos não têm ciência das possíveis opções e, portanto, não podem tomar boas decisões. A representação explícita de variações comunica os usuários sobre as possíveis alternativas de escolha e auxilia na tomada de decisões (PETERSEN *et al.*, 2006).
- **Diminuição dos custos:** uma abordagem sistemática para gerenciar variações entre processos similares pode reduzir drasticamente os custos de implantação e manutenção de modelos de processos de software. Atualizações em processos similares podem ser gerenciadas de forma eficiente, evitando erros humanos e mitigando os riscos envolvidos em tais mudanças (NAKAMURA *et al.*, 2009, p. 71). Segundo Boffoli *et al.* (2009), a abordagem de linha de processos reduz os esforços para a modelagem e manutenção de processos de software porque a modelagem de uma pequena quantidade de ativos reutilizáveis permite a obtenção de muitas instâncias de processos, adequados para diferentes contextos específicos.
- **Aumento da consistência:** permitir qualquer combinação de elementos de processo pode atrapalhar a adaptação e causar confusão, visto que nem todas as possíveis combinações contribuem para alcançar os objetivos dos processos. Portanto, é importante a representação controlada de variações (uma das características principais de linha de processos). Esse controle é alcançado através de variações e dependências especificadas no metamodelo proposto neste trabalho.

2.2. Linha de Processos de Software

Uma linha de processos de software pode ser definida como um conjunto de processos que compartilham características comuns, e características variáveis de forma controlada, dentro de um domínio ou para um propósito particular (ALEIXO *et al.*, 2010; MARTÍNEZ-RUIZ *et al.*, 2008, p. 120; ROMBACH, 2005, p. 87; TERNITÉ, 2009; WASHIZAKI, 2006b).

A diferença fundamental entre os conceitos de linha de processos de software e conceitos tradicionais de adaptação de processos é que, no contexto de linha de processos, os processos de uma organização são ativamente preparados para várias necessidades futuras, enquanto que na adaptação clássica de processos, os processos são adaptados individualmente (e reativamente) para um projeto específico (ARMBRUST *et al.*, 2008).

Em uma linha de processos, as similaridades são partes bem definidas e as variações são antecipadas (KULKARNI; BARAT, 2011, p. 317). Segundo Aiello *et al.* (2010), variabilidade se refere à possibilidade de mudanças em processos, indicando que parte desse processo é variável, ou apenas parcialmente definida, de modo a possibilitar diferentes versões do mesmo processo dependendo do uso pretendido e do contexto de execução.

Variabilidade permite o adiamento de uma decisão para um momento específico no processo de desenvolvimento (DAIZHONG; SHANHUI, 2009, p. 257). No contexto de linha de produtos de software, Bachmann e Bass (2001, p. 1) afirmam que, algumas vezes, a pessoa que realiza a adaptação não é a mesma que desenhou a arquitetura, portanto, para permitir alta qualidade na adaptação, é necessário documentar explicitamente os locais das variações e quais tipos de variações são permitidas. Esse princípio é adequado para processos e é importante porque não é possível prever antecipadamente todo o desenvolvimento de software (LIMA REIS, 2003). Além disso, variações nos requisitos dos projetos implicam em variações também nos processos utilizados para desenvolvê-los (ROUILLÉ, *et al.*, 2012, p. 148).

Em uma linha de processos, as similaridades e variações são representadas através de uma estrutura reutilizável central denominada de Arquitetura de Linha de Processos de Software (ALPrS)[2] (WASHIZAKI, 2006b, p. 1301). A arquitetura da linha de processos representa uma única estrutura reutilizável e que age como um processo de referência a partir do qual todos os possíveis processos que compõem a linha de processos podem ser instanciados (SCHNIEDERS; PUHLMANN, 2006, p. 584). Esse conjunto formado por todos os processos que podem ser instanciados a partir de uma arquitetura de linha de processos é conceituado como uma linha de processos de software propriamente dita (WASHIZAKI, 2006b, p. 1302).

A Figura 2 apresenta a relação entre uma arquitetura de linha de processos e o conjunto de todos os processos que podem ser instanciados a partir dela, formando uma linha de processos. Nesse exemplo, a arquitetura de linha de processos (a) representa similaridades, variações e dependências, a partir das quais diversos processos podem ser instanciados. A linha

[2] Nesta dissertação de mestrado, os termos "arquitetura de linha de processos de software" e "arquitetura de linha de processos" e a sigla ALPrS serão utilizados indistintamente para representar o mesmo conceito.

de processos (b) é formada pelo "Processo 1", "Processo 2", e assim por diante, até o "Processo n".

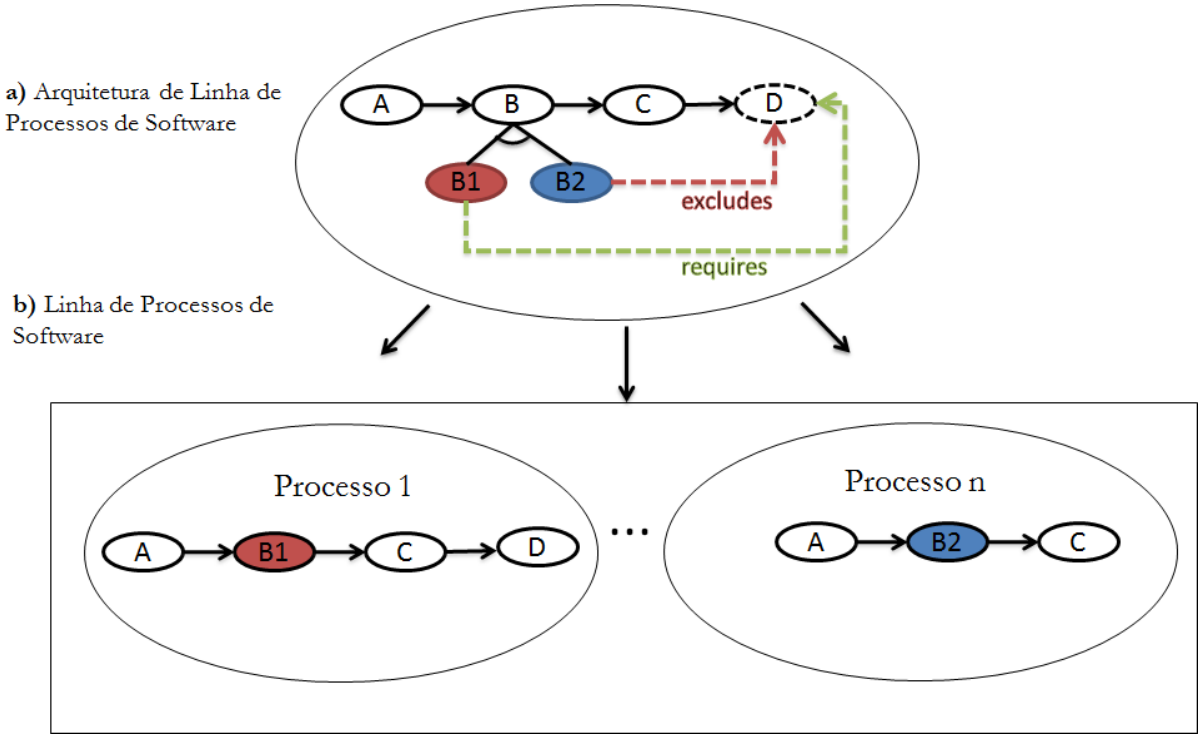


Figura 2 – Arquitetura de Linha de Processos e Linha de Processos (Adaptado de Washizaki 2006b).

A ideia de instanciar uma diversidade de processos a partir de uma arquitetura de processo central implica em eficiência na criação de uma linha de processos, visto que isso facilita a adaptação de grandes processos através da escolha de componentes dependendo das circunstâncias de cada projeto (SIMIDCHIEVA *et al.*, 2007).

As similaridades representam o núcleo da arquitetura da linha de processos, ou seja, as partes que não variam de uma instância de processo para outra. Por outro lado, variações são partes que se diferem entre os processos e são representadas por elementos opcionais, pontos de variação e variantes.

Um elemento obrigatório deve estar presente em todas as instâncias de processos pertencentes à uma linha de processos; já um elemento opcional pode ou não estar presente. Um ponto de variação representa um lugar em um modelo de processo onde variações ocorrem (MARTÍNEZ-RUIZ *et al.*, 2008). Isso significa que uma decisão tem que ser tomada em termos da escolha de alternativas (ARMBRUST *et al.*, 2009). As diversas alternativas de um ponto de variação, os chamados variantes, são elementos concretos que os preenchem, sendo que cada variante realiza o ponto de variação de uma forma diferente (MARTÍNEZ-RUIZ *et al.*, 2008).

Além das variações, uma linha de processos de software possui dependências entre os elementos, que garantem a consistência dos processos gerados (MARTÍNEZ-RUIZ *et al.*, 2008).

A utilização do mecanismo de dependências permite, por exemplo, que elementos complementares sejam inseridos mutuamente e que elementos conflitantes não possam ser inseridos ao mesmo tempo nos processos instanciados a partir de uma linha de processos de software.

2.3. Engenharia de Linha de Processos de Software

A Engenharia de Linha de Processos de Software (ELPrS) é definida como um sistema de estratégias relacionadas e abordagens sistemáticas, que envolvem modelos, procedimentos, arquitetura e tecnologia para construir, instanciar e gerenciar linha de processos de software (NUNES *et al.*, 2010; WASHIZAKI, 2006a, p. 416). Rombach (2005) separa a ELPrS nas seguintes fases: engenharia de domínio, engenharia de aplicação e gerência. A terceira fase, relacionada com o gerenciamento de ativos reutilizáveis da linha de processos, é prevista também em modelos de maturidade como o MR-MPS-SW (SOFTEX, 2013). A Figura 3 apresenta as três fases do processo de ELPrS.

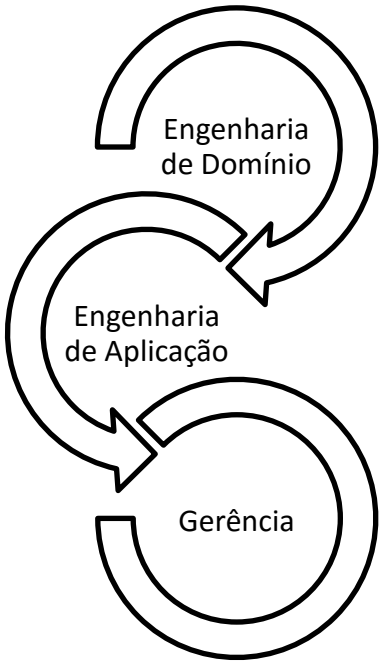


Figura 3 – Fases da Engenharia de Linha de Processos de Software.

A Engenharia de Domínio (ED) é o processo pelo qual processos reutilizáveis genéricos são criados, através da representação de similaridades e variações controladas em um determinado domínio (ROMBACH, 2005). Existem duas abordagens de execução da ED: *bottom-up* e *top-down*. Na abordagem *bottom-up*, processos reutilizáveis são criados a partir da análise de similaridades e variações em processos específicos de projetos já executados (ROMBACH, 2005). Ou seja, os processos específicos dos projetos são generalizados. Essa é uma abordagem indutiva

em que processos concretos são utilizados como insumo para a criação de processos em um nível mais alto de abstração.

Por outro lado, a abordagem *top-down* cria processos reutilizáveis sem a análise de processos preexistentes, partindo de requisitos que os processos devem satisfazer.

A arquitetura de linha de processos é elaborada durante a fase de engenharia de domínio (GIESE *et al.*, 2007). Já na Engenharia de Aplicação (EA), processos específicos de um projeto são instanciados (ROMBACH, 2005). Nesta etapa, um subconjunto dos elementos presentes na arquitetura da linha de processos é escolhido para a formação de um processo.

A gerência da linha de processos avalia os processos específicos de cada contexto para decidir se um comportamento fora do padrão deve ser generalizado para um processo genérico (ROMBACH, 2005). Além disso, garante a qualidade dos processos reutilizáveis.

2.4. Considerações Finais

Este Capítulo apresentou uma visão geral de conceitos de reutilização de processos e do paradigma de Linha de Processos de Software, conceituando os principais elementos que constituem uma LPrS e mostrando as fases do processo de engenharia envolvido na construção e utilização de processos reutilizáveis. Esses conceitos apresentados serão de fundamental importância para o entendimento dos demais Capítulos desta dissertação e, principalmente, da abordagem proposta.

3. Revisão Sistemática da Literatura

Este Capítulo apresenta os dados do planejamento e da condução da revisão sistemática da literatura executada no contexto desta dissertação de mestrado.

3.1. Introdução

Revisão Sistemática da Literatura (RSL)[3] é um tipo de estudo secundário que segue um método bem definido, repetível e imparcial para identificar, analisar e interpretar todas as evidências relacionadas a uma determinada área de pesquisa (KITCHENHAM, 2007).

Ainda segundo Kitchenham (2007), a menos que uma revisão informal da literatura seja conduzida de forma completa e criteriosa, os resultados obtidos têm pouco valor científico. Budgen e Brereton (2006) complementam afirmando que a execução de uma revisão sistemática da literatura traz resultados muito mais convincentes para a elaboração de um contexto de pesquisa. Essa é a principal motivação para a realização desse tipo de estudo.

No contexto desta dissertação de mestrado, uma revisão sistemática da literatura foi planejada e executada tendo como diretrizes o processo seguido no trabalho de Souza (2008) e o processo definido por Kitchenham (2007), sendo composta pelas fases especificadas na Figura 4.

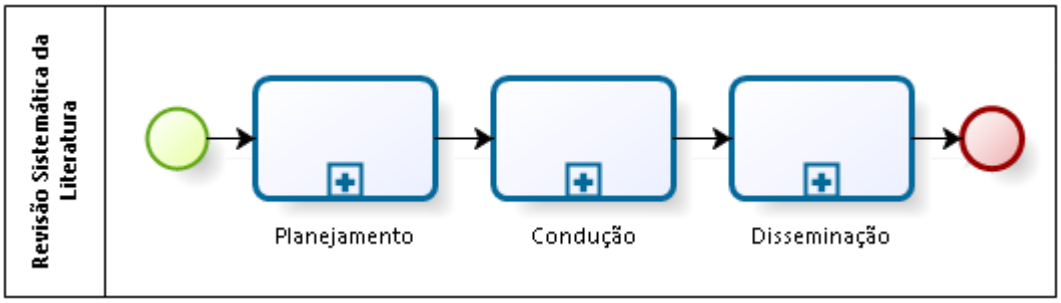


Figura 4 – Fases da Revisão Sistemática da Literatura.

[3] Neste documento, usaremos o termo "revisão sistemática" ou a sigla RSL para se referir a revisão sistemática da literatura.

Planejamento: essa etapa consiste no planejamento, definindo-se a necessidade para a realização do estudo, o contexto e o objeto a ser pesquisado. Durante o planejamento, é elaborado um documento denominado protocolo, onde se define as questões de pesquisa que serão investigadas, as bibliotecas digitais onde as publicações serão buscadas, os critérios de seleção de bibliotecas digitais e de publicações e que dados serão extraídos das publicações selecionadas. A Figura 5 apresenta as atividades do planejamento da revisão sistemática e a Figura 6 mostra as atividades para o desenvolvimento do protocolo.

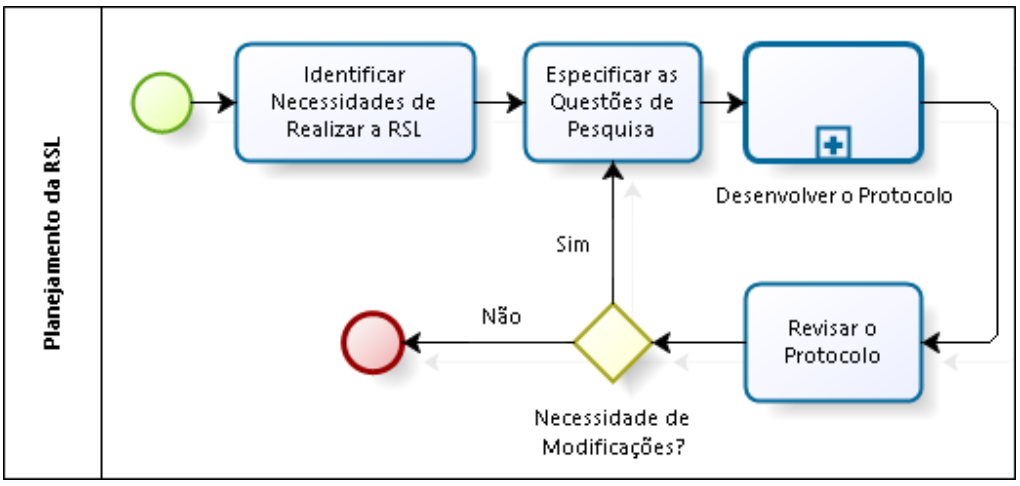


Figura 5 – Atividades do Planejamento da Revisão Sistemática.

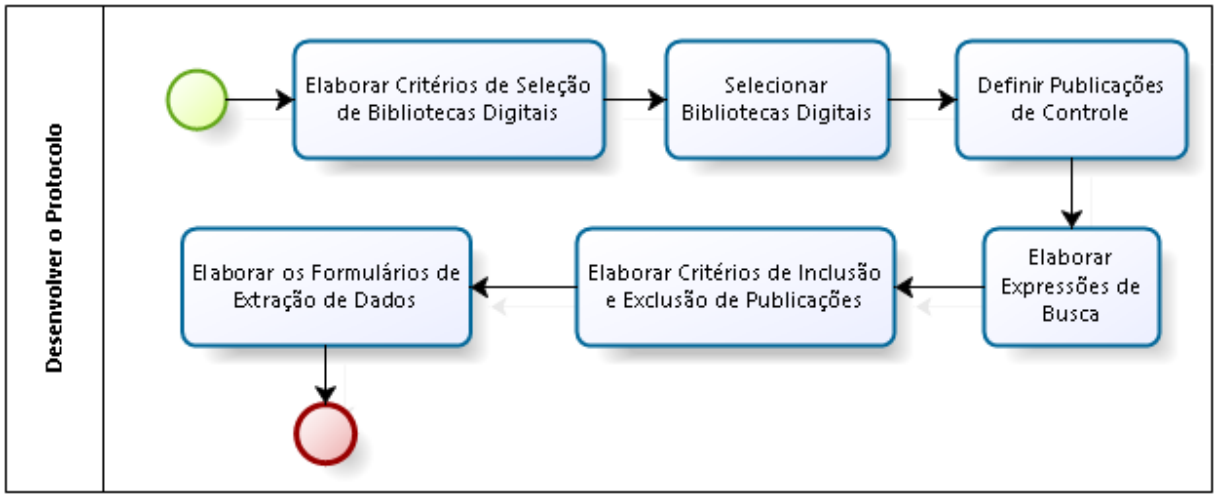


Figura 6 – Atividades de Desenvolver o Protocolo.

Condução: nessa etapa, as ações planejadas são executadas efetivamente. As expressões de busca são executadas nas bibliotecas digitais e as publicações retornadas são selecionadas (excluídas ou incluídas) segundo os critérios previamente definidos. Após a seleção, os dados das publicações remanescentes são extraídos para que sejam analisados em detalhes. Por fim, informações são cruzadas e comparadas, tendências e *gaps* na área de interesse são encontrados. A Figura 7 apresenta as atividades da condução da revisão sistemática.

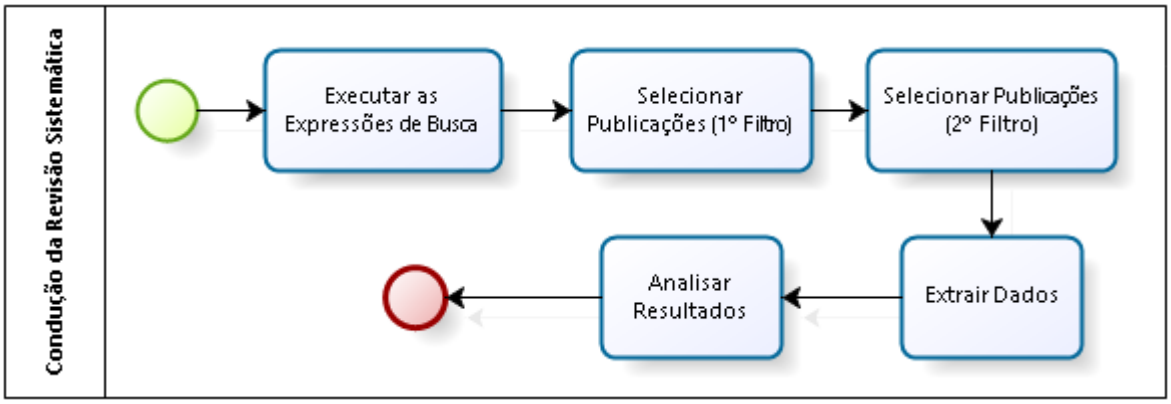


Figura 7 – Atividade da Condução da Revisão Sistemática.

Disseminação: essa etapa consiste em disseminar os resultados para que partes potencialmente interessadas possam acessá-los. Dentre os veículos de publicação, podemos citar periódicos, conferências e a própria dissertação de mestrado.

3.2. Protocolo do Estudo

O protocolo define um plano para a execução do estudo (KITCHENHAM, 2007). Nele são especificados os critérios, as informações a serem extraídas e os procedimentos básicos para a condução do estudo de forma criteriosa, repetível, objetiva e sem viés do(s) pesquisador(es).

Esta seção define o protocolo da revisão sistemática da literatura planejada e executada no contexto desta dissertação de mestrado. Elementos do modelo de protocolo do trabalho de Souza (2008) foram reutilizados, com algumas adaptações necessárias. Além disso, algumas informações foram acrescentadas baseadas em Kitchenham (2007).

A ferramenta StArt (*State of the Art through Systematic Review*) (HERNANDES *et al.*, 2012) auxiliou na etapa de definição do protocolo permitindo a especificação do objetivo do estudo, questões de pesquisa, critérios de inclusão e exclusão de publicações e formulário de extração de dados.

3.2.1. Contexto

A demanda por produtos de software vem aumentando e estes, por sua vez, são cada vez mais complexos (BORGES; FALBO, 2001). Para desenvolver software de qualidade, diversos autores argumentam que a definição de processo de software é essencial (BERTOLLO; FALBO, 2003; FUGGETTA, 2000; ROMBACH, 2005; SUTTON; OSTERWEIL, 1996).

Entretanto, a definição de um processo de software é uma atividade complexa que requer muita experiência e conhecimento de diversas áreas e disciplinas da engenharia de software

(ALEIXO *et al.*, 2010). Uma das abordagens usadas para auxiliar na definição de processo de software é a reutilização de processo (BARRETO, 2011).

Linha de Processos de Software (LPrS) é um paradigma que vem se mostrando adequado para auxiliar na definição de processos de software (HURTADO *et al.*, 2013). Entretanto, uma vez que ainda não existe um estudo que investigue detalhadamente os construtores de variabilidades em linha de processos de software, os metamodelos existentes são incompletos e isso vem dificultando a modelagem de processos com variabilidades.

A revisão sistemática da literatura que será executada no contexto desta dissertação visa identificar as características das abordagens de representação de variabilidades em linha de processos de software. Com essas informações, deseja-se investigar em profundidade os tipos de variabilidades que estão presentes em metamodelos existentes, levantar um conjunto de requisitos para que essas variabilidades sejam representadas, comparar os metamodelos em relação a esses requisitos e propor um metamodelo que os incorpore.

3.2.2. Objetivo do Estudo

Analisar publicações científicas sobre representação de variabilidades em linha de processos de software.

Com o propósito de caracterizar as abordagens existentes em relação aos tipos de variabilidades representadas, elementos de processos alvos de variação, linguagens utilizadas e ferramentas utilizadas como apoio.

Com relação ao paradigma de linha de processos de software.

Do ponto de vista do pesquisador.

No contexto acadêmico e industrial.

3.2.3. Questões de Pesquisa

Nesta RSL, foram definidas as seguintes questões de pesquisa a serem respondidas como resultado do estudo:

- **Q1:** Quais são as principais contribuições das abordagens propostas?
- **Q2:** Como representar variabilidades?
- **Q2.1:** Que tipos de variabilidades são fornecidas?
- **Q2.2:** Que tipos de dependências são fornecidas?
- **Q2.3:** Que tipos de elementos de processo de software são alvos de variabilidades?
- **Q3:** Que linguagens de modelagem de processos estão sendo usadas?

- **Q4:** Que modelos e padrões de processo estão sendo usados?
- **Q5:** Que paradigmas de engenharia de software estão sendo aplicados em conjunto com LPrS?
- **Q6:** Que conceitos de LPrS estão sendo usados?
- **Q7:** As abordagens são apoiadas por ferramentas?
- **Q8:** Como as propostas estão sendo validadas?

3.2.4. Intervenção

Paradigma de linha de processos de software.

3.2.5. Comparação

Não se aplica.

3.2.6. População

Trabalhos publicados em conferências e periódicos relatando abordagens para representação de variabilidades em linha processos de software.

3.2.7. Resultados

A partir dos dados extraídos das publicações retornadas pelas expressões de busca e que não forem excluídos pelos critérios de exclusão, pretende-se responder as questões de pesquisa e identificar requisitos para a representação de variabilidades em modelos de processos, que devem servir de base para a elaboração de um metamodelo.

3.2.8. Estratégia de Busca

Esta seção define os critérios para a seleção das bibliotecas digitais a serem consultadas, o grupo que forma as publicações de controle, e os critérios de exclusão e inclusão das publicações encontradas.

3.2.8.1. Critérios para a Seleção das Bibliotecas Digitais

Para a escolha das bibliotecas digitais a serem consultadas, todos os seguintes requisitos devem ser satisfeitos:

- Possuir engenho de busca que permita a busca automática de publicações a partir de expressões de busca fornecidas pelo pesquisador;
- Ser acessível via navegador de Internet;
- Permitir restringir a busca pelo ano de início ou início e fim;
- Permitir restringir a busca pelos títulos e resumos das publicações;
- Garantir resultados únicos para uma mesma expressão de busca.

3.2.8.2. Publicações de Controle

As publicações de controle formam o conjunto mínimo de publicações que devem ser retornadas pela expressão de busca, exceto quando não é indexada pela biblioteca digital. Dessa forma, são úteis para calibrar os termos e os conectores lógicos utilizados. As publicações de controle utilizadas neste estudo foram as seguintes:

1. ALEGRÍA, J.; BASTARRICA, M.; QUISPE, A.; OCHOA, S. An MDE Approach to Software Process Tailoring. In: INTERNATIONAL CONFERENCE ON SOFTWARE AND SYSTEMS PROCESS, ICSSP, 2011, Honolulu, EUA. **Proceedings...** Nova York: ACM, 2011. p. 43-52. ISBN: 978-1-4503-0730-7.
2. BARRETO, A.; NUNES, E.; ROCHA, A. Supporting the Definition of Software Processes at Consulting Organizations via Software Process Lines. In: INTERNATIONAL CONFERENCE ON THE QUALITY OF INFORMATION AND COMMUNICATIONS TECHNOLOGY, QUATIC, 7., 2010, Oporto, Portugal**. Proceedings…** Washington, USA: IEEE Computer Society, 2010. p. 15-24. ISBN: 978-0-7695-4241-6.
3. MARTÍNEZ-RUIZ, T.; GARCÍA, F.; PIATTINI, M. Towards a SPEM v2.0 Extension to Define Process Lines Variability Mechanisms. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, RESEARCH, MANAGEMENT AND APPLICATIONS, SERA, 6., 2008, Praga, República Checa. **Proceedings…** Berlin: Springer, 2008. p. 115-130.
4. MARTÍNEZ-RUIZ, T.; GARCÍA, F.; PIATTINI, M.; MÜNCH, J. Applying AOSE Concepts to Model Crosscutting Variability in Variant-Rich Processes. In: EUROMICRO CONFERENCE ON SOFTWARE ENGINEERING AND ADVANCED APPLICATIONS, SEAA, 37., 2011c, Oulu, Finlândia. **Proceedings…** Washington: IEEE Computer Society, 2011c. p. 334-338.

5. SUTTON, S.; OSTERWEIL, L. Product Families and Process Families. In: INTERNATIONAL SOFTWARE PROCESS WORKSHOP, ISPW, 10., 1996, Dijon, França. **Proceedings…**. Los Alamitos: IEEE Computer Society, 1996. p. 109-111.
6. TERNITÉ, T. Process Lines: A Product Line Approach Designed for Process Model Development. In: EUROMICRO CONFERENCE ON SOFTWARE ENGINEERING AND ADVANCED APPLICATIONS, SEAA, 35., 2009, Patras, Greece. **Proceedings…** Patras, Greece: IEEE Computer Society, 2009. p. 173-180. ISBN: 978-0-7695-3784-9.
7. WASHIZAKI, H. Building software process line architectures from bottom up. In: INTERNATIONAL CONFERENCE ON PRODUCT-FOCUSED SOFTWARE PROCESS IMPROVEMENT, PROFES, 7., 2006a, Amsterdam, Netherlands. **Proceedings…** Berlin/Heidelberg, Germany: Springer-Verlag, 2006a. p. 415-421. ISBN: 978-3-540-34682-1.

As expressões de busca foram elaboradas de forma que retornassem todas as publicações de controle indexadas em cada biblioteca digital. Assim, caso uma determinada publicação fosse indexada por uma biblioteca digital, mas não fosse retornada, a expressão de busca era refinada. A Tabela 1 contém informações sobre as publicações de controle indexadas e retornadas por cada biblioteca digital consultada.

Tabela 1 – Indexação das Publicações de Controle.

#		Scopus		IEEE		Eng. Village	
		Indexada	Retornada	Indexada	Retornada	Indexada	Retornada
1	(ALEGRÍA *et al.*, 2011)	✓	✓	✗	✗	✓	✓
2	(BARRETO *et al.*, 2010)	✓	✓	✓	✓	✓	✓
3	(MARTÍNEZ-RUIZ *et al.*, 2008)	✓	✓	✗	✗	✗	✗
4	(MARTÍNEZ-RUIZ *et al.*, 2011c)	✓	✓	✓	✓	✓	✓
5	(SUTTON; OSTERWEIL 1996)	✓	✓	✓	✓	✓	✓
6	(TERNITÉ, 2009)	✓	✓	✓	✓	✓	✓
7	(WASHIZAKI, 2006a)	✓	✓	✗	✗	✓	✓

3.2.8.3. Busca Manual

Para este estudo, não serão realizadas buscas manuais.

3.2.8.4. Expressão de Busca

A seguinte expressão de busca foi utilizada como base para as buscas nas bibliotecas digitais:

("process line" **OR** "process lines" **OR** "process family" **OR** "process families" **OR** "variant rich process" **OR** "variant-rich process") **OR**

(("business process" **OR** "business processes" **OR** "software process" **OR** "software processes" **OR** "process model" **OR** "process models") **AND**

(variability **OR** variabilities **OR** variation **OR** variations **OR** variant **OR** variants **OR** commonality **OR** commonalities **OR** similarity **OR** similarities **OR** "feature model" **OR** "feature models" **OR** "feature modeling" **OR** "feature-based"))

3.2.9. Critérios de Inclusão de Publicações

Durante a execução do primeiro filtro, devem ser incluídas as publicações que se enquadrem em pelo menos um dos seguintes critérios:

- **CI-01**: A publicação trata de variabilidade, similaridade ou flexibilidade em modelos de processos;
- **CI-02**: A publicação trata de linha de produtos de software;
- **CI-03**: A publicação trata de reutilização ou adaptação de processos;
- **CI-04**: A publicação trata de linha de processos de software.

Os critérios de inclusão são aplicados somente no primeiro filtro. Portanto, servem apenas para sinalizar um indício de que as publicações tratam de linha de processos de software.

O critério de inclusão CI-01 indica que a publicação aparentemente trata de variabilidade em modelos de processo. E, apesar dos autores não terem explicitamente citado linha de processos de software no título, resumo e palavras-chave a publicação é aceita para o segundo filtro para uma análise mais detalhada, visto que o tema pode ter sido abordado no restante da publicação. O mesmo raciocínio se aplica para os critérios de inclusão CI-02, CI-03 e CI-04.

O critério de inclusão CI-02 foi criado pelo fato de alguns autores utilizarem o termo linha de produtos para processos, em vez de linha de processos. Portanto, foi decidido aceitar, no primeiro filtro, todas as publicações que tratem de linha de produtos para que elas sejam analisadas em mais detalhes posteriormente.

O critério de inclusão CI-04 é o mais direto e é utilizado quando há evidências de que a publicação trata do tema já no primeiro filtro. Entretanto, em vez de aceitar definitivamente, a publicação ainda precisar ser analisada com a leitura completa no segundo filtro.

3.2.10. Critérios de Exclusão de Publicações

Durante a aplicação do primeiro filtro, serão excluídas as publicações que se enquadrem em pelo menos um dos seguintes critérios de exclusão:

- **CE-01**: A publicação representa anais de conferências ou de workshops, relatórios técnicos, livros, análises, tutoriais, painéis de discussão, *minitrack* ou qualquer tipo de publicação não-científica;
- **CE-02**: A publicação não está relacionada com a área de Engenharia de Software (se enquadra em outra área do conhecimento, como Inteligência Artificial, Eletrônica, Mecatrônica, Matemática, Engenharia Nuclear, Enfermagem, Medicina, Geofísica, Geologia, Física, Astronomia, Biologia, Engenharia Agrônoma, Engenharia Naval, Engenharia Elétrica, Meteorologia, Psiquiatria etc.);
- **CE-03**: A publicação trata reutilização somente em produtos de software;
- **CE-04**: A publicação trata variabilidade/similaridade a outras entidades (como: software, requisitos, características do projeto, empresas, culturas organizacionais, tempo, resultados esperados, desempenho, algoritmos, critérios, métricas, causas, funcionalidades, padrões, riscos, técnicas, fatores críticos de sucesso, localizações geográficas, protocolos);
- **CE-05**: A publicação emprega variabilidade/similaridade apenas no sentido genérico de comparação, cálculo de similaridade estrutural, mineração de processos, controle estatístico de processo, desvio em relação ao planejado ou imprevisibilidade;
- **CE-06**: A publicação usa o termo "*feature*" no sentido de funcionalidade;
- **CE-07**: A publicação trata de processo de software, mas sem considerar variabilidade;

Durante a aplicação do segundo filtro, serão excluídas as publicações que se enquadrem em pelo menos um dos seguintes critérios:

- **CE-08**: A publicação não foi encontrada disponível gratuitamente;
- **CE-09**: A publicação não foi encontrada para *download* nem em forma impressa;
- **CE-10**: A publicação trata de reutilização de processos de software, mas sem a aplicação de conceitos de linha de processos;
- **CE-11**: A publicação não está escrita em inglês;
- **CE-12**: A publicação trata somente de linha de produtos;
- **CE-13**: A publicação representa uma revisão ou mapeamento sistemático.

3.2.11. Escopo da Pesquisa

Nesta seção, são delimitados os idiomas em que as publicações podem estar escritas, os critérios de seleção das bibliotecas digitais e as restrições de datas de publicação.

3.2.11.1. Idiomas

Foram consideradas somente publicações escritas em **inglês**. Esse idioma foi escolhido por ser o mais utilizado na escrita de publicações científicas no tema pesquisado.

3.2.11.2. Restrições de Datas

Para a execução das buscas, foi estabelecido **1996** como ano de início. Esse ano foi escolhido por ser o ano de publicação do artigo de Sutton e Osterweil (1996), intitulado "*Product Families and Process Families*". Tal artigo foi a publicação mais antiga encontrada sobre o tema linha/família de processos e, apesar de abordar o tema de forma bastante abstrata, levantou uma série direções de pesquisa e questões que precisavam ser investigadas em mais profundidade.

3.2.12. Extração de Dados

A extração de dados das publicações deve ocorrer mantendo-se os termos originais utilizados pelos autores.

Os dados serão extraídos apenas de publicações que sejam aceitas após a aplicação do primeiro e do segundo filtro. Para cada publicação, serão extraídos os dados gerais (não relacionados diretamente com as questões de pesquisa) e dados relativos às questões de pesquisa, como segue:

- **Dados Gerais**
  - **Título:** título da publicação.
  - **Autores:** nomes dos autores da publicação.
  - **Ano de publicação**: ano em que o trabalho foi publicado.
  - **Tipo de veículo de publicação**: classificação do veículo em que o trabalho foi publicado. Possíveis valores: *workshop*, conferência/simpósio[4] ou periódico.
  - **Critério de inclusão:** critério de inclusão em que a publicação se enquadra.

[4] Conferência e simpósio foram agrupados por serem tratados como sinônimos pelos anais de alguns eventos científicos.

- **Critério de exclusão**: critério de exclusão em que a publicação se enquadra.

- **Dados Relativos às Questões de Pesquisa**
  - **Principais Contribuições:** principais contribuições e produtos do trabalho de pesquisa.
  - **Fases da Engenharia**: caso método de engenharia de linha de processos seja uma das contribuições do trabalho, é verificada que fases são cobertas pelo método. Possíveis valores: engenharia de domínio, engenharia de aplicação ou ambas.
  - **Paradigmas de Engenharia de Software Utilizados:** paradigmas de Engenharia de Software utilizados para complementar conceitos de linha de processos.
  - **Tipos de elementos de processo:** tipos de elementos de processo em que as variabilidades se aplicam. Exemplos: atividade, papel etc.
  - **Tipos de variabilidades:** tipos de variabilidades tratados nas publicações. Possíveis valores: opcional, obrigatório, ponto de variação etc.
  - **Modelos/Normas/Guias:** normas, modelos de qualidade, modelos de ciclo de vida, *frameworks* de processos e guias utilizados para a modelagem de processos utilizando conceitos de linha de processos. Exemplos: CMMI-DEV (SEI, 2010), ISO/IEC 12207, Scrum, Modelo em Espiral etc.
  - **Linguagens de Modelagem:** linguagens criadas, estendidas ou utilizadas para representar diretamente variabilidades em modelos de processos ou para representar qualquer tipo de informação relacionada. Exemplos: SPEM, UML etc.
  - **Tipos de Dependências:** tipos de dependências entre as características possíveis de serem definidas. Exemplos: exclusão-mútua etc.
  - **Método de Avaliação:** descrição de como a abordagem proposta foi avaliada. Exemplos: estudo de caso, estudo experimental, exemplo etc.
  - **Ferramentas:** se a abordagem possui ferramentas de apoio para auxiliar qualquer das etapas.

3.3. Condução

Esta seção apresenta os dados de execução do protocolo da revisão sistemática da literatura.

3.3.1. Seleção das Bibliotecas Digitais

Por satisfazerem os critérios estabelecidos na seção 3.2.8.1 (p. 36), foram selecionadas como fontes de pesquisa as bibliotecas digitais Scopus, IEEE Xplore Digital Library e Engineering Village. Como a expressão de busca de cada uma dessas bibliotecas exige uma sintaxe ligeiramente diferente, a expressão de busca base, definida na seção 3.2.8.4 (p. 39), precisou ser adaptada. Além disso, para cada biblioteca digital, foi necessário selecionar o modo de operação, opções e valores, como segue:

- **Scopus**
  - **Endereço:** http://www.scopus.com
  - **Expressão de Busca:** TITLE-ABS-KEY(("process line" **OR** "process lines" **OR** "process family" **OR** "process families" **OR** "variant rich process" **OR** "variant-rich process") **OR** (("business process" **OR** "business processes" **OR** "software process" **OR** "software processes" **OR** "process model" **OR** "process models") **AND** (variability **OR** variabilities **OR** variation **OR** variations **OR** variant **OR** variants **OR** commonality **OR** commonalities **OR** similarity **OR** similarities **OR** "feature model" **OR** "feature models" **OR** "feature modeling" **OR** "feature-based"))) **AND** (PUBYEAR > 1995) **AND** (PUBYEAR < 2014) **AND** (LIMIT-TO(SUBJAREA, "COMP"))
  - **Observações:** Modo "*advanced search*".
- **IEEE Xplore Digital Library**
  - **Endereço:** http://ieeexplore.ieee.org
  - **Expressão de Busca:** ("process line" **OR** "process lines" **OR** "process family" **OR** "process families" **OR** "variant rich process" **OR** "variant-rich process") **OR** (("business process" **OR** "business processes" **OR** "software process" **OR** "software processes" **OR** "process model" **OR** "process models") **AND** (variability **OR** variabilities **OR** variation **OR** variations **OR** variant **OR** variants **OR** commonality **OR** commonalities **OR** similarity **OR** similarities **OR** "feature model" **OR** "feature models" **OR** "feature modeling" **OR** "feature-based"))

- **Observações:** Modo "*command search*" em "*advanced search*". Opção "*metadata only*" selecionada. "*Content Type*" com os valores "*Conference Publications*", "*Journals & Magazines*" e "*Early Access Articles*" selecionados. "*Publication year*" com a opção "*Range*" selecionada e com os valores 1996 e 2013.

- **Engineering Village**
  - **Endereço:** http://www.engineeringvillage.org
  - **Expressão de Busca:** ((((("process line" **OR** "process lines" **OR** "process family" **OR** "process families" **OR** "variant rich process" **OR** "variant-rich process") **OR** (("business process" **OR** "business processes" **OR** "software process" **OR** "software processes" **OR** "process model" **OR** "process models") **AND** (variability **OR** variabilities **OR** variation **OR** variations **OR** variant **OR** variants **OR** commonality **OR** commonalities **OR** similarity **OR** similarities **OR** "feature model" **OR** "feature models" **OR** "feature modeling" **OR** "feature-based"))) WN KY) **AND** (723* WN CL) **AND** ({ca} **OR** {ja} **OR** {ip}) WN DT)
  - **Observações:** Modo "*expert search*". "*Limit to*" com valores 1996 e 2013.

3.3.2. Execução das Expressões de Buscas

Nessa etapa, as expressões de busca específicas de cada biblioteca digital foram executadas. As datas das buscas e os números de publicações retornadas são apresentados na Tabela 2.

Tabela 2 – Publicações Retornadas pelas Expressões de Busca.

Biblioteca	Quantidade	Porcentagem	Data de Busca
Scopus	1376	41%	12/02/2014
Engineering Village	1195	36%	12/02/2014
IEEE Xplore	778	23%	12/02/2014
Total	3349	100%	-

Como se pode perceber, a biblioteca digital Scopus retornou o maior número de publicações (1376), seguida pela Engineering Village (1195) e pela IEEE Xplore (778), totalizando 3349 publicações retornadas.

Dessas 3349 publicações retornadas, 1133 foram retornadas em mais de uma biblioteca digital e, portanto, foram consideradas duplicadas. Dessa forma, as expressões de busca retornaram 2216 publicações únicas.

3.3.3. Seleção de Publicações

A execução das expressões de busca não garante que todas as publicações retornadas abrangem de forma adequada o tema de pesquisa. Para refinar a escolha de publicações, foi executada a seleção de publicações, que consistiu na leitura de cada publicação e aplicação dos critérios de inclusão e exclusão. A seleção de publicações foi dividida em duas etapas: 1º filtro e 2º filtro.

Durante a etapa de seleção de publicações, a ferramenta StArt foi utilizada para aceitar ou rejeitar as publicações, selecionando-se os critérios de inclusão e de exclusão aplicáveis a cada uma delas. A Figura 8 apresenta o formulário de seleção de publicações, onde os critérios de inclusão e exclusão são aplicados para cada trabalho retornado.

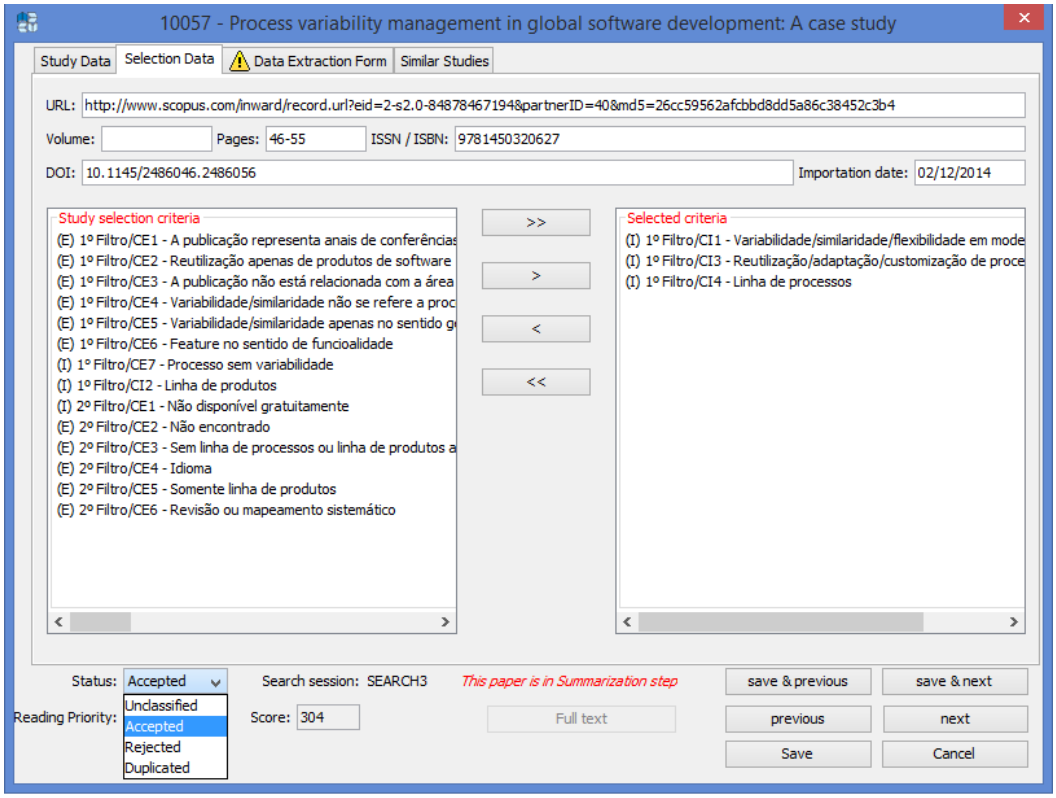


Figura 8 – Formulário de Seleção de Publicações.

Na medida em que o estudo vai sendo executado, gráficos de acompanhamento vão sendo gerados sobre a quantidade de publicações aceitas, rejeitadas e duplicadas, o que permite se ter uma visão geral do andamento do estudo (Figura 9).

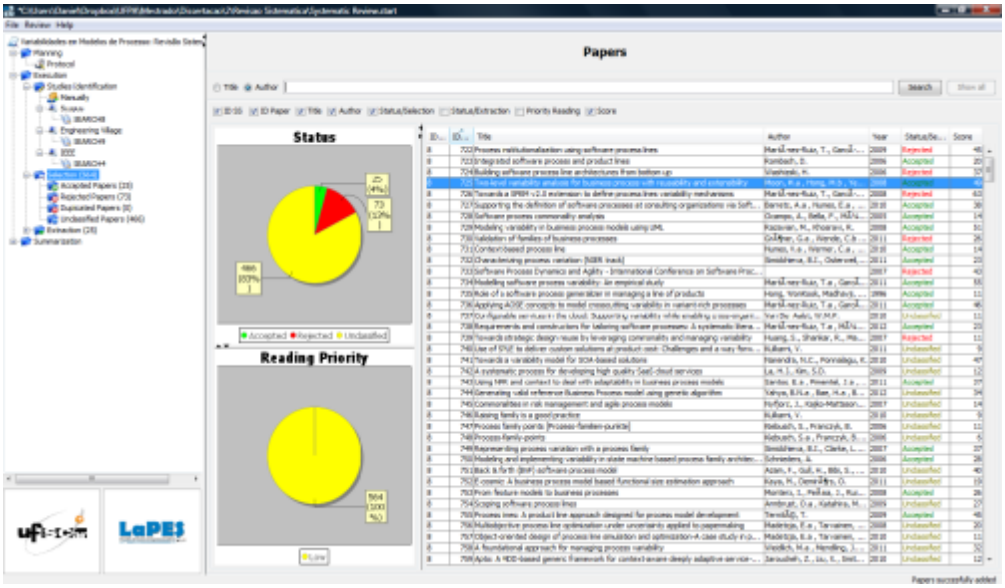


Figura 9 – Gráficos de Acompanhamento da Ferramenta StArt.

3.3.3.1. Seleção de Publicações (1º Filtro)

No primeiro filtro de seleção, os títulos, resumos e palavras-chaves das 3349 publicações foram lidos e avaliados de acordo com os critérios de inclusão CI-01, CI-02, CI-03 e CI-04 (definidos na seção 3.2.9, p. 39) e com os critérios de exclusão CE-01, CE-02, CE-03, CE-04, CE-05, CE-06 e CE-07 (definidos na seção 3.2.10, p. 40). Para que uma publicação possa seguir para o segundo filtro, deve satisfazer pelo menos um critério de inclusão e nenhum dos critérios de exclusão. Ao fim dessa etapa, 463 publicações foram aceitas. A Figura 10 apresenta o número de publicações rejeitadas, duplicadas e aceitas.

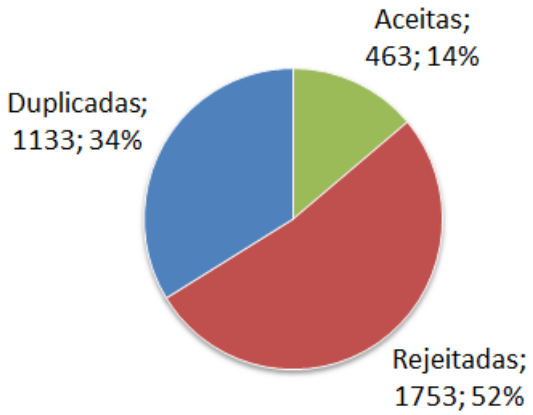


Figura 10 – Execução do 1º Filtro.

3.3.3.2. Seleção de Publicações (2º Filtro)

No segundo filtro de seleção, o texto completo das 463 publicações aceitas na etapa anterior foi lido e foram aplicados os critérios de exclusão CE-08, CE-09, CE-10, CE-11, CE-12 e CE-13 (definidos na seção 3.2.10, p. 40). Para que uma publicação possa seguir para a extração

de dados, não deve satisfazer nenhum desses critérios de exclusão. Ao fim dessa etapa, 40 publicações foram aceitas. A Figura 11 apresenta o número de publicações rejeitadas e aceitas ao final do segundo filtro de seleção.

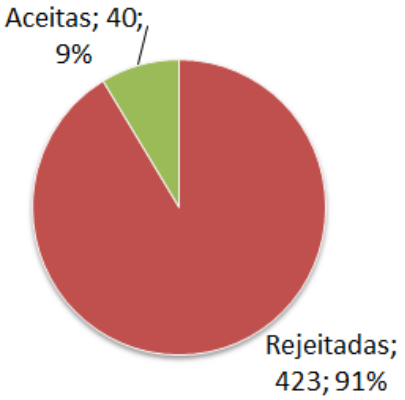


Figura 11 – Execução do 2º Filtro.

3.3.4. Extração de Dados

A extração dos dados foi realizada nas 40 publicações remanescentes após a aplicação dos filtros. A ferramenta StArt permitiu a seleção dos valores para cada atributo definido. Por exemplo, as contribuições de cada publicação. A ferramenta permite a definição de três tipos de atributos: com um único valor, valores múltiplos ou texto livre. A Figura 12 apresenta parte do formulário de extração de dados definido com a ferramenta.

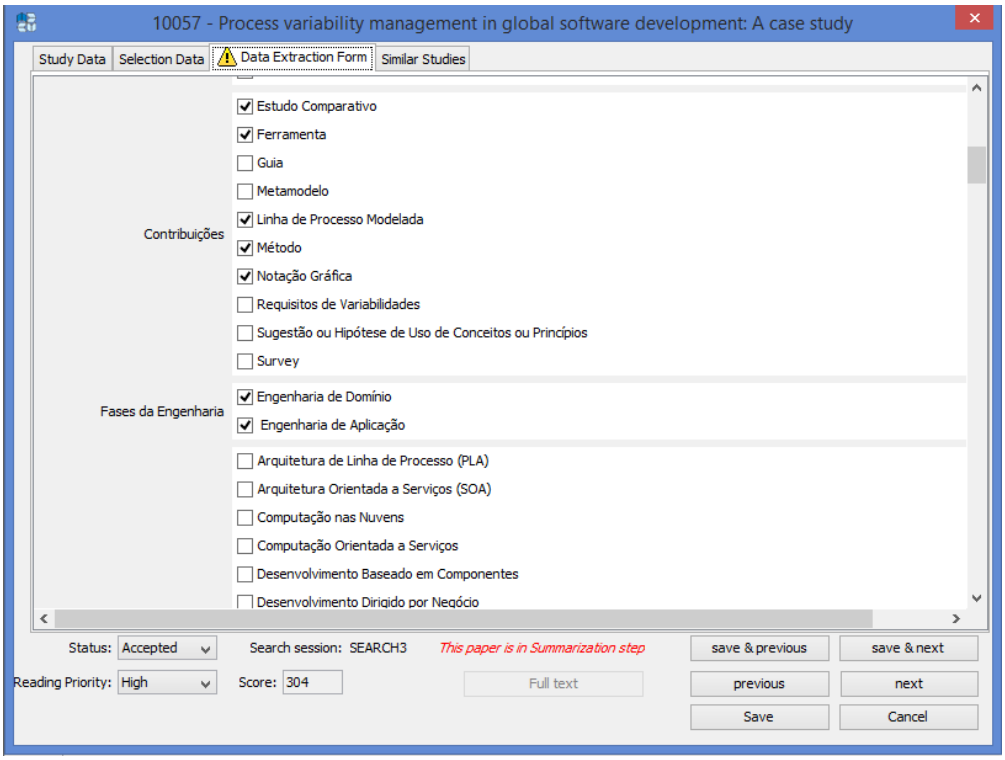


Figura 12 – Formulário de Extração de Dados da Ferramenta StArt.

Após o preenchimento do formulário de extração de dados para cada publicação aceita, os dados foram exportados para o formato XLS (planilha eletrônica), o que permitiu a geração de gráficos para a análise dos resultados.

3.3.5. Análise dos Resultados

A análise dos resultados envolve o agrupamento, resumo e interpretação dos dados extraídos das publicações aceitas. Para responder cada questão de pesquisa, foram analisados os dados quantitativos coletados durante a extração de dados.

3.3.5.1. Estudos Primários

A Figura 13 mostra a distribuição temporal em que os estudos foram publicados, considerando todos os tipos de veículos de publicação (periódico, conferência/simpósio e *workshop*). Os 40 estudos primários aceitos foram publicados em uma faixa de 18 anos (de 1996 até 2013). Existe evidência de que o interesse na área está crescendo, visto que entre 1996 e 2003 apenas 2 (5%) dos estudos haviam sido publicados; e 26 (65%) deles foram publicados nos últimos 4 anos analisados (entre 2010 e 2013).

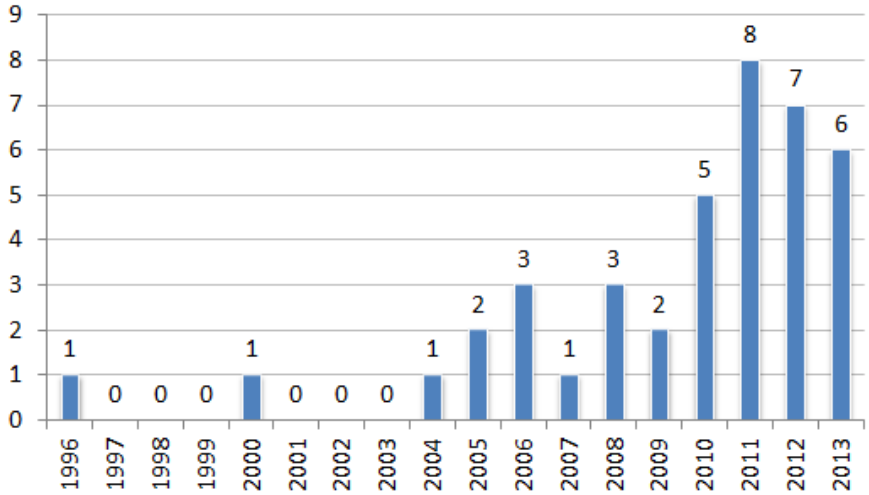


Figura 13 – Publicações por Ano.

3.3.5.2. Tipos de Veículos de Publicação

Apesar do número baixo de publicações, há evidência de que as pesquisas na área estão amadurecendo, visto que, até 2005, nenhum estudo havia sido publicado em periódicos, e, de 2006 até 2013, 5 estudos foram publicados nesse tipo de veículo de publicação. A Figura 14 mostra o número de publicações por tipo de veículo de publicação.

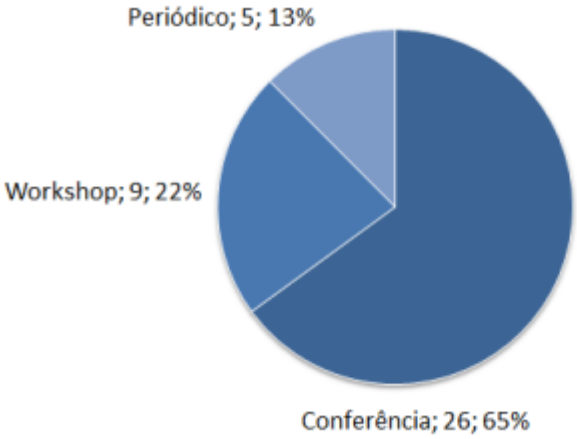


Figura 14 – Tipos de Veículos de Publicação.

3.3.5.3. Veículos de Publicação

Os 40 estudos primários foram publicados em 28 diferentes veículos de publicação (Figura 15). Em 21 desses veículos de publicação (que representa 75% deles), somente um estudo foi publicado. A conferência ICSSP (*International Conference on Software and Systems Process*) foi o veículo de publicação em que mais estudos primários foram publicados, com um total de 4.

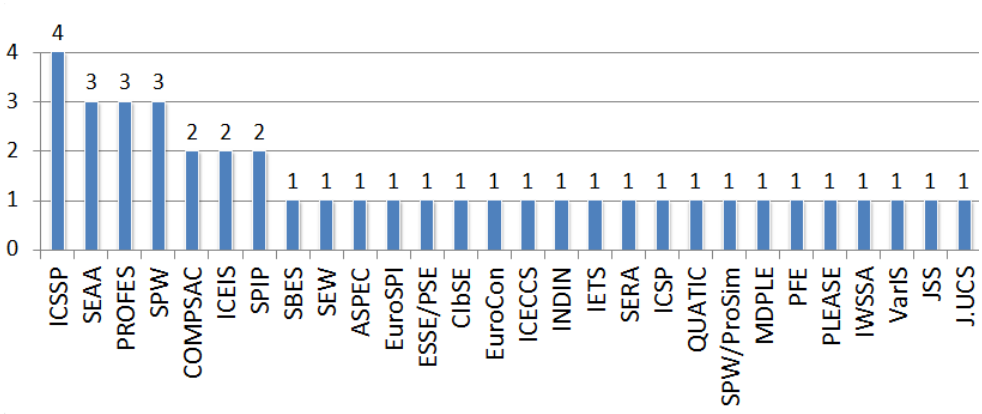


Figura 15 – Número de Publicações por Veículo de Publicação.[5]

[5] ICSSP - International Conference on Software and Systems Process; SEAA - Euromicro Conference on Softare Engineering and Advanced Applications; PROFES - International Conference on Product-Focused Software Process Improvement; SPW - International Software Process Workshop; COMPSAC - Annual IEEE International Computer Software and Applications Conference; ICEIS - International Conference on Enterprise Information Systems; SPIP - Software Process: Improvement and Practice; SBES - Brazilian Symposium on Software Engineering; SEW - Annual IEEE Software Engineering Workshop; ASPEC - Asia-Pacific Software Engineering Conference; EuroSPI - European Conference on Systems, Software and Services Process Improvement; ESSE/PSE - European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering; CIbSE - Ibero-American Conference on Software Engineering; EuroCon - IEEE International Conference on Computer as a Tool; ICECCS - IEEE International Conference on Engineering of Complex Computer Systems; ICECCS - IEEE International Conference on Industrial Informatics; IETS - IET Software; SERA - International Conference on Software Engineering; Research; Management and Applications;

3.3.5.4. Principais Contribuições (Q1)

Q1: Quais são as principais contribuições das abordagens?

Para responder a essa questão de pesquisa, foram extraídas das publicações suas principais contribuições. Para ser considerada uma contribuição, as informações precisavam ser julgadas como úteis para que outros pesquisadores que leem as publicações as utilizem para evoluir a área de linha de processos.

É importante salientar que uma mesma publicação pode conter várias contribuições. A Figura 16 mostra quais foram as principais contribuições encontradas nas publicações.

A contribuição mais comumente encontrada foi "método", que foi encontrado em 28 publicações (70%). No contexto desta RSL, um método foi conceituado como um conjunto de atividades, produtos de trabalho e papeis com o objetivo de auxiliar na engenharia de linha de processos.

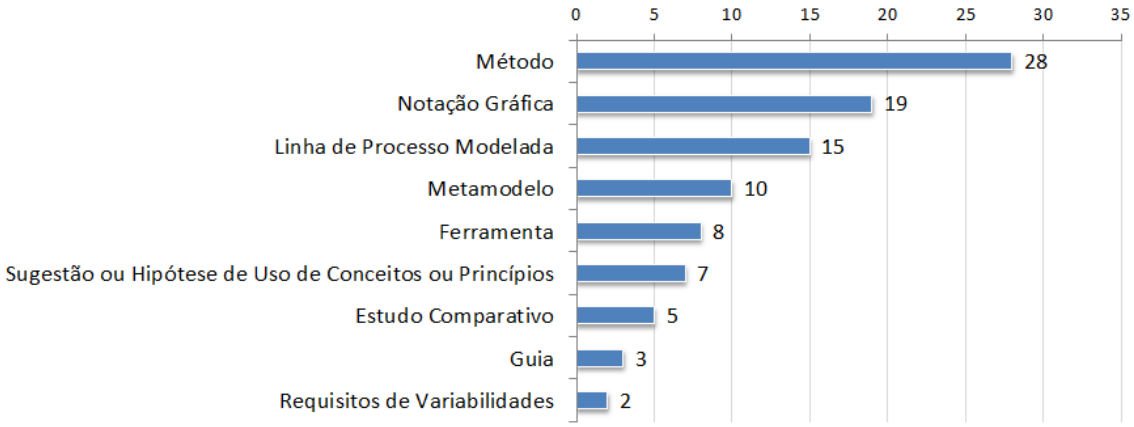


Figura 16 – Principais Contribuições.

Como a Figura 17 mostra, das 28 publicações que propõem métodos de engenharia de linha de processos, todas elas cobrem a fase de engenharia de domínio e 25 cobrem a fase de engenharia de aplicação. Isso pode indicar que mais esforços estão sendo conduzidos em pesquisas sobre como projetar processos reutilizáveis do que em como reutilizar processos a partir de linha de processos.

ICSP - International Conference on Software Process; QUATIC - International Conference on the Quality of Information and Communications Technology; SPW/ProSim - International Software Process Workshop and International Workshop on Software Process Simulation and Modeling; MDPLE - International Workshop on Model-Driven Product Line Engineering; PFE - International Workshop on Product Family Engineering; PLEASE - International Workshop on Product Line Approaches in Software Engineering; IWSSA - International Workshop on System/Software Architectures; VarIS - International Workshop on Variability Support in Information Systems; JSS - Journal of Systems and Software; J.UCS - Journal of Universal Computer Science.

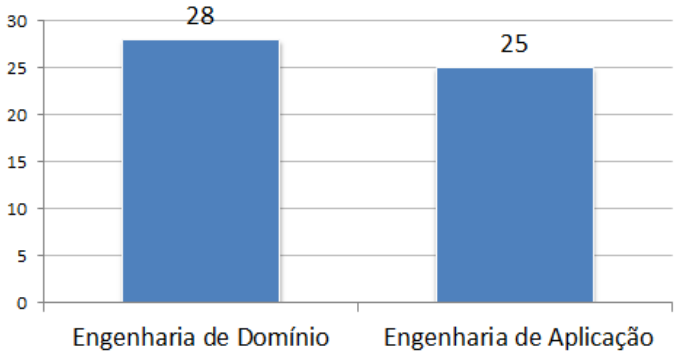


Figura 17 – Fases da Engenharia de Linha de Processos.

3.3.5.5. Tipos de Variabilidades (Q2.1)

Q2.1: Que tipos de variabilidades são fornecidas?

Como estabelecido por Simmonds *et al.* (2013), a expressividade da notação para especificar variabilidade é um dos fatores mais relevantes para o sucesso na adoção de linha de processos de software. Assim, o propósito da questão Q2.1 é investigar como variabilidade pode ser expressa nas abordagens propostas para linha de processos.

Inicialmente, foram considerados os tipos mais usados de variabilidades, investigados por Martínez-Ruiz *et al.* (2012): "alternativo/ponto de variação", "opcional" e "obrigatório". Entretanto, outros valores surgiram durante a condução da revisão sistemática: "invariante" e "transversal". A Figura 18 apresenta os tipos de variabilidades encontradas nas publicações.

O tipo de variabilidade transversal se refere a variações que podem ocorrer em diversos locais durante um processo. Um exemplo em que sua aplicação pode ser adequada é para a representação variabilidades em atividades de garantia da qualidade. Porém, essa possibilidade ainda não foi explorada pela literatura e pode ser uma direção de pesquisa promissora.

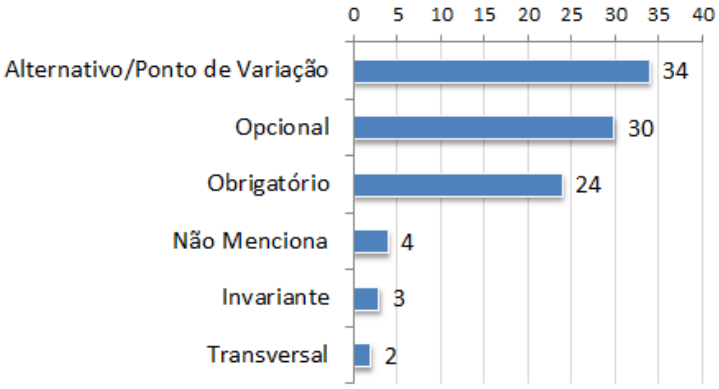


Figura 18 – Tipos de Variabilidades.

3.3.5.6. Tipos de Dependências (Q2.2)

Q2.2: Que tipos de dependências são fornecidas?

Dependências formam um importante mecanismo para apoiar as resoluções de variabilidades e para garantir a consistência dos processos gerados. Essas dependências definem regras para ações que devem ser tomadas em conjunto de forma a facilitar a definição do processo.

Os dois tipos de dependências consideradas foram "requisito" (e.g.: se A é incluído, então B também deve ser incluído) e "exclusão/exclusão mútua"[6] (e.g.: se A é incluído, então B não deve ser incluído). A Figura 19 mostra os tipos de dependências e o número de publicações relacionadas.

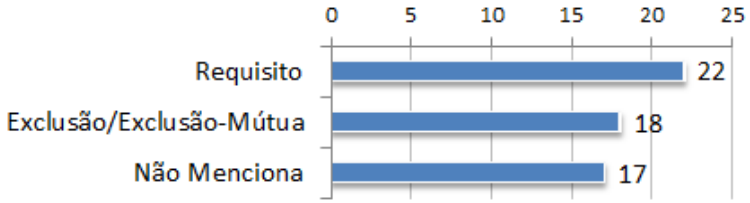


Figura 19 – Tipos de Dependências.

3.3.5.7. Elementos de Processo de Software (Q2.3)

Q2.3: Que tipos de elementos de processo de software são alvos de variabilidades?

Esta questão de pesquisa buscou quais são os elementos de processo em que as variabilidades se aplicam. Por exemplo, se a abordagem define que uma atividade pode ser opcional, contabilizava-se o tipo de elemento atividade.

Primeiramente tentou-se estabelecer um conjunto fechado de tipos (e.g., considerando como base os tipos de elementos de uma linguagem amplamente conhecida como o SPEM), Entretanto, percebeu-se que essa é uma tarefa muito difícil, devido à variedade de sinônimos e significados conflituosos para elementos de processos utilizados por diferentes autores. Além disso, muitos autores não entravam em detalhes sobre o significado dos elementos, e simplesmente utilizavam uma determinada nomenclatura sem utilizar uma definição formal ou referenciar um modelo ou linguagem que os definisse. Assim, os nomes originalmente utilizados pelos autores foram mantidos. A Figura 20 apresenta os tipos de elementos de processo encontrados e o número de publicações em que cada um foi encontrado.

[6] Inicialmente, os tipos "exclusão" e "exclusão-mútua" eram contabilizados como tipos separados. Porém, muitas publicações não especificavam explicitamente a qual desses dois tipos estavam se referindo, o que tornou impossível a contabilização dessa forma. Assim, foi decidido pela unificação.

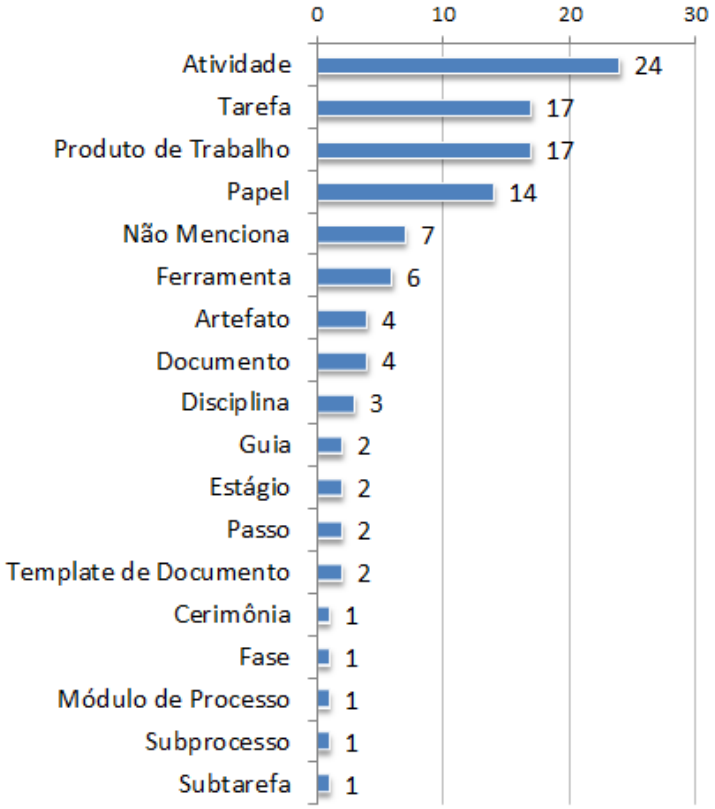


Figura 20 – Elementos de Processo de Software.

3.3.5.8. Linguagens de Modelagem (Q3)

Q3: Que linguagens de modelagem de processo estão sendo usadas?

A linguagem SPEM (*Software Process Engineering Metamodel*) foi a mais utilizada para a representação de linha de processos de software, totalizando 17 publicações (que representa 42%). Destas, 6 publicações utilizaram o SPEM em sua forma original e 11 publicações utilizaram o SPEM com algumas modificações dessa linguagem. Das 11 publicações que utilizaram formas modificadas do SPEM, 7 utilizaram o vSPEM, 2 utilizaram o eSPEM (Experimental SPEM) e 2 utilizaram modificações do SPEM sem no entanto utilizar uma denominação própria.

Merece destaque, também, a linguagem de modelagem de características, encontrada em 13 publicações (32%).

Em 5 publicações (que representa 12%) os autores afirmaram que suas abordagens são independentes de linguagens, ou seja, são genéricas o suficiente para serem aplicadas em diversas linguagens. Ainda assim, alguns desses autores utilizaram alguma linguagem com o objetivo de demostrar os conceitos de variabilidades, como no trabalho de Rouillé *et al.* (2012), que utilizaram a linguagem SPEM para tal.

A linguagem BPMN (*Business Process Model and Notation*), apesar de ser largamente utilizada para a modelagem de processos, inclusive com variabilidades, foi encontrada em apenas 1

publicação. Isso pode ser explicado pelo fato da RSL ter incluído no escopo somente processos de software, deixando de fora trabalhos que trataram de processos de negócio. A Figura 21 apresenta as linguagens encontradas como resultado da extração de dados.

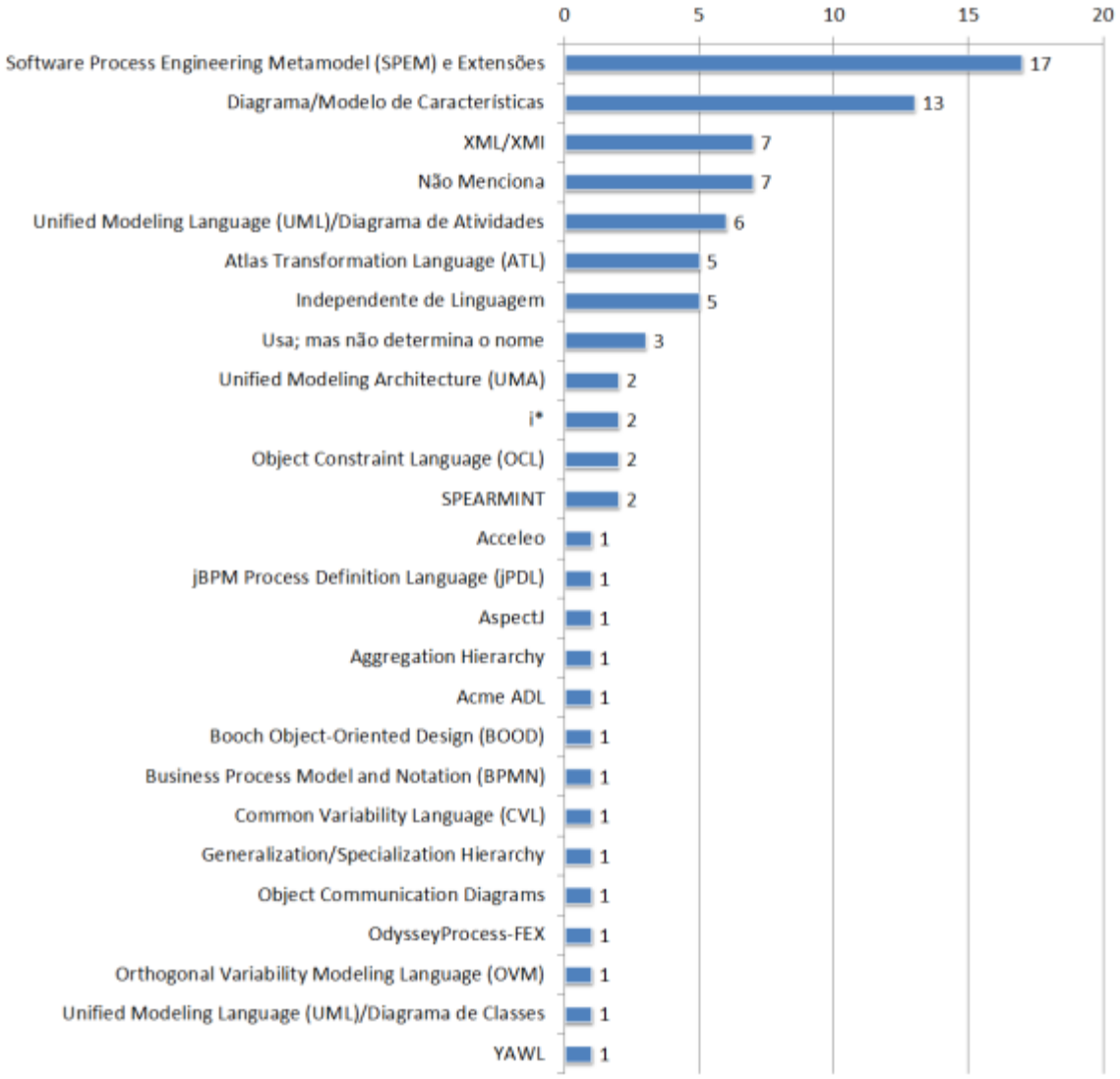


Figura 21 – Linguagens de Modelagem.

3.3.5.9. Modelos de Qualidade e Padrões de Processo (Q4)

Q4: Que modelos e padrões de processo estão sendo usados?

Esta pergunta teve como objetivo encontrar quais são os modelos de maturidade, padrões, modelos de ciclo de vida, normas, *frameworks* de processo e entidades similares que vêm sendo modelados utilizando conceitos de linha de processos de software.

Esse interesse partiu da afirmação de alguns autores de que linha de processos de software é um paradigma que vem se mostrando adequado para a definição de processos de software que precisam estar em conformidade com modelos de maturidade (BARRETO *et al.*, 2010; BIFFL *et al.*, 2006; HURTADO *et al.*, 2013) e de métodos ágeis (ALEIXO *et al.*, 2010;

JAFARINEZHAD; RAMSIN, 2012; MARTÍNEZ-RUIZ *et al.*, 2013a). Portanto, linhas de processos de software baseadas nesses processos poderiam trazer uma série de benefícios a engenheiros de processos que necessitam defini-los.

Entretanto, foi verificado que poucos trabalhos modelaram processos utilizando conceitos de linha de processos de software. Além disso, a definição dos processos não ocorreu com o rigor necessário e os processos definidos foram bastante incompletos, até porque esse não foi o objetivo principal de nenhum deles. Na maioria das publicações, a definição de processos foi realizada apenas em forma de exemplos para demostrar conceitos.

Foram encontrados 18 modelos diferentes, sendo que os mais frequentes foram o V-Modell-XT, que foi encontrado em 4 publicações, e o Scrum, encontrado em 3 publicações. Dos modelos encontrados, 12 foram modelados em apenas uma publicação e 4 foram modelados em apenas 2 publicações. Em 21 publicações (52%), nenhum modelo de processo é mencionado.

Existe, portanto, uma lacuna de pesquisa que pode indicar direções de futuros esforços na área de linha de processos, que é a modelagem de linhas de processos baseadas em modelos de processos largamente utilizados, como o Scrum, CMMI-DEV (SEI, 2010) e MR-MPS-SW. A Figura 22 apresenta os modelos de processo encontrados.

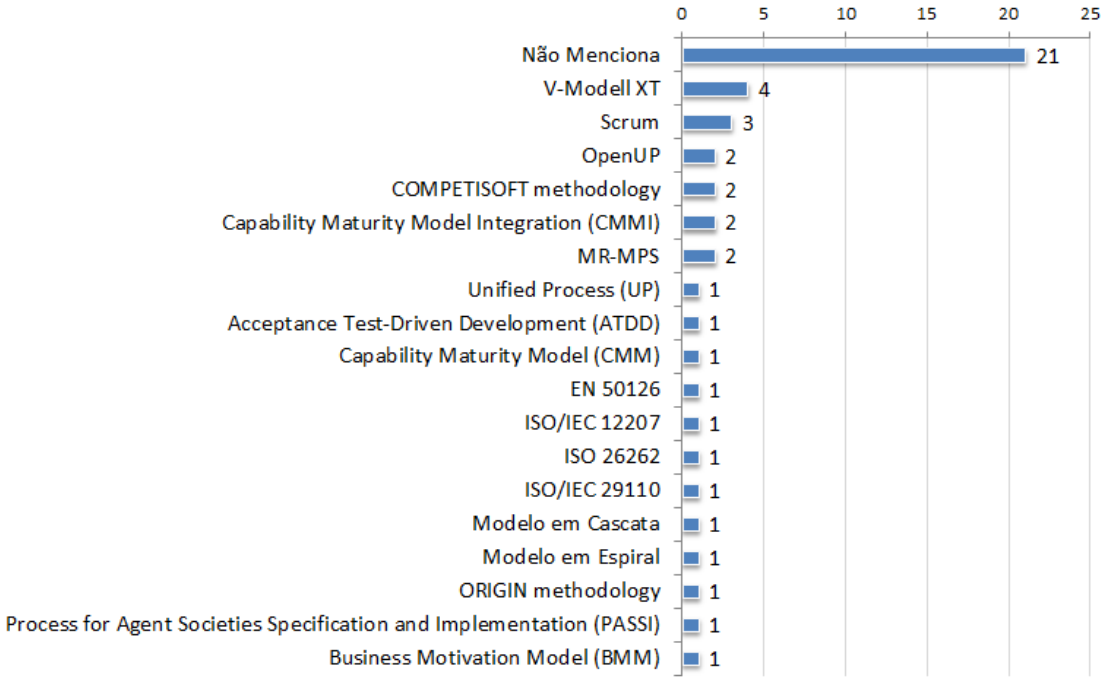


Figura 22 – Modelos de Qualidade e Padrões de Processo.

3.3.5.10. Paradigmas de Engenharia de Software (Q5)

Q5: Que paradigmas de engenharia de software estão sendo aplicados em conjunto com LPrS?

Existem vários conceitos e paradigmas de engenharia de software que podem complementar e beneficiar o paradigma de linha de processos de software (HURTADO *et al.*, 2013; MARTÍNEZ-RUIZ *et al.*, 2011c). Partindo desse princípio, foi investigado no contexto dessa revisão sistemática como outros paradigmas são utilizado em conjunto com linha de processos.

"Engenharia Dirigida por Modelo" (*Model-Driven Engineering* - MDE) foi o paradigma mais frequentemente aplicado em conjunto com linha de processos de software. Tal paradigma está relacionado com transformações entre modelos para automatizar alguns passos da engenharia de linha de processos de software. Essas transformações podem auxiliar, por exemplo, na geração automática de um processo a partir da verificação das variabilidades selecionadas em um modelo de características.

"Gerência de Raciocínio" foi o segundo paradigma mais aplicado. Isso pode ser explicado porque linha de processos de software é baseada na representação de variabilidades e decisões devem ser tomadas sobre incluir ou não certos elementos ou decidir qual é a alternativa mais adequada de uma variedade de possibilidades. Assim, diversos pesquisadores vêm aplicando *rationale management* para sistematicamente resolver questões de variabilidades.

"Engenharia Orientada a Aspectos" foi o terceiro paradigma mais encontrado. Alguns autores utilizaram tal paradigma com o objetivo de representar variabilidades transversais. Entretanto, esse é um tipo de variabilidade muito pouco explorado pela literatura e pode ser uma área de pesquisas futuras. Um exemplo em que variabilidades transversais podem ser interessantes é na representação de variabilidades em processos de garantia de qualidade.

Além desses, os paradigmas "Arquitetura Orientada a Serviços" (*Service-Oriented Architecture* - SOA), "Gerência de Processo de Negócio" (*Business Process Management* - BPM) e "*Situational Method Engineering*" (SME) foram encontrados em duas publicações cada um; "Desenvolvimento Global de Software" (*Global Software Development* - GSD) e "Engenharia Orientada a Agentes" foram encontrados em uma publicação cada um. A Figura 23 apresenta os paradigmas encontrados.

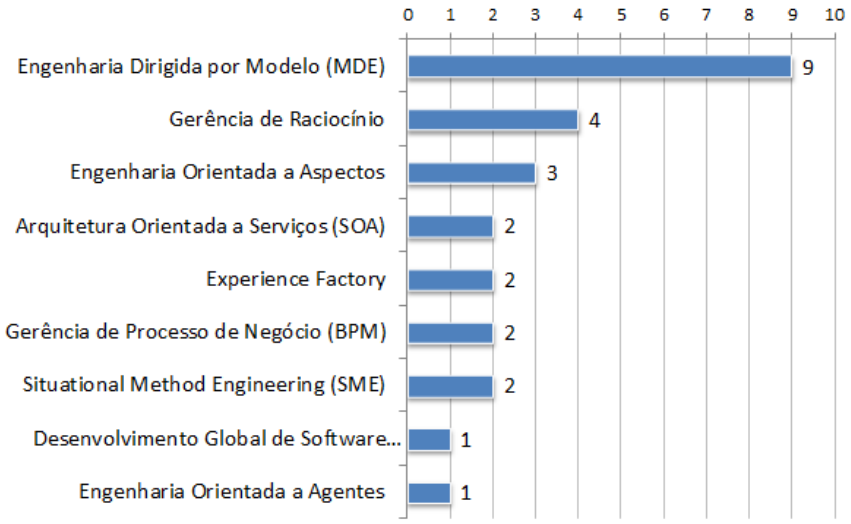


Figura 23 – Paradigmas de Engenharia de Software.

3.3.5.11. Conceitos de Linha de Processos de Software (Q6)

Q6: Que conceitos de LPrS estão sendo usados?

Para responder a esta questão de pesquisa, foram considerados os conceitos de linha de processos que veem sendo utilizados. As próprias denominações para linha de processos foram consideradas.

A maioria dos autores utiliza o termo "linha de processos" para denominar o paradigma, encontrado em 29 publicações (72%). Entretanto, outros termos são utilizados também, como "família de processos" (37%) e "*variant rich processes*" (10%).

O segundo conceito mais utilizado pelos autores é "linha de produtos" e "modelagem de características", encontrado cada um em 16 publicações (40%). À primeira vista, pode parecer estranho o uso do termo linha de produtos. Isso pode ser explicado pelo fato do paradigma linha de processos ter sido inspirado em linhas de produtos de software e, portanto, muitos autores utilizam a nomenclatura linha de produtos para processos, deixando explícito que estão aplicando esse paradigma de reutilização de software a processos de software. Outros termos sinônimos de linha de produtos também foram encontrados, como "família de produtos" e "família de sistemas".

Um fato que evidencia a falta de padronização da nomenclatura da área é o conceito de "arquitetura de linha de processo", encontrado em 12 publicações (30%). Alguns autores utilizam tal conceito como uma arquitetura de processo representada como um processo genérico reutilizável a partir do qual todos os processos que fazem parte da linha de processos são instanciados. Entretanto, outros autores utilizam esse conceito simplesmente como um sinônimo de linha de processos.

A Figura 24 apresenta os conceitos relacionados a linha de processos encontrados nas publicações.

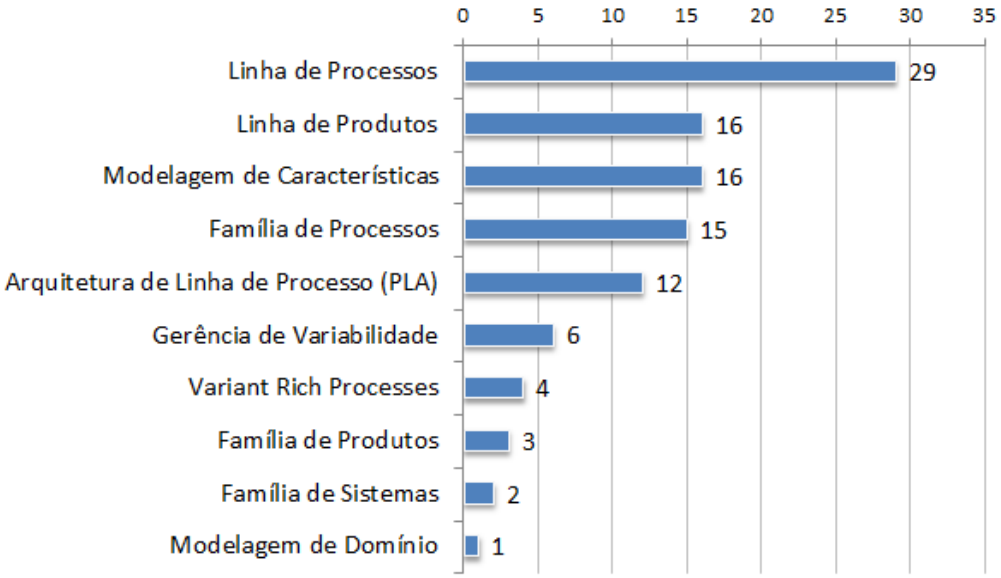


Figura 24 – Conceitos de Linha de Processos de Software.

3.3.5.12. Ferramentas de Apoio (Q7)

Q7: As abordagens são apoiadas por ferramentas?

Para responder a essa pergunta de pesquisa, verificou-se se os autores citavam a existência de um apoio ferramental para auxiliar qualquer das atividades de engenharia de linha de processos de software. Tais ferramentas poderiam ser: desenvolvidas por terceiros e apenas utilizadas pelos autores; desenvolvidas por terceiros e modificadas pelos autores; ou totalmente desenvolvidas pelos autores. O critério para a contabilização era apenas apoiar as abordagem em alguma das etapas.

No contexto dessa RSL, não foi realizada um análise detalhada sobre as funcionalidades dessas ferramentas. Pode ser uma direção de pesquisa o detalhamento das funcionalidades e o mapeamento de que etapas da engenharia de linha de processos podem estar desamparadas por apoio ferramental. A Figura 25 mostra a proporção de trabalhos citaram ou não a existência de ferramentas de apoio.

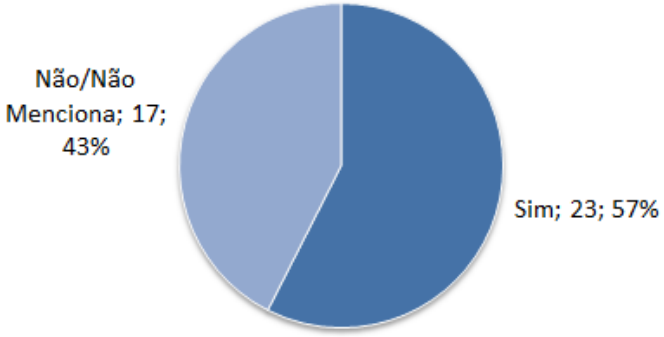


Figura 25 – Uso de Ferramentas de Apoio.

3.3.5.13. Métodos de Validação (Q8)

Q8: Como as propostas estão sendo validadas?

O propósito dessa questão é investigar quão rigorosamente as abordagens propostas têm sido validadas. Para estabelecer a faixa de métodos possíveis, as seguintes estratégias de estudos empíricos, definidas por Wohlin *et al.* (2012), foram consideradas: *survey*, estudo de caso e estudo experimental. Além disso, foram consideradas estratégias não-científicas como: avaliação de especialistas, comparação, exemplo e relato de uso. Para contabilização, foi possível classificar um mesmo estudo em mais do que uma dessas estratégias.

Foi possível notar que a grande maioria das abordagens foi validada através dos métodos menos rigorosos (exemplo, relato de uso, comparação e avaliação de especialistas). Além disso, 4 publicações (10% do total) nem sequer validaram as abordagens propostas.

Apenas 3 publicações (representando 7%) utilizaram métodos rigorosos de validação (estudo experimental e *survey*) e nenhum trabalho foi validado através de estudo de caso. Esses resultados evidenciam uma enorme lacuna e a necessidade de métodos mais rigorosos para a validação das propostas, o que pode indicar uma imaturidade nas pesquisas sobre linha de processos de software. A Figura 26 apresenta os métodos de avaliação das abordagens.

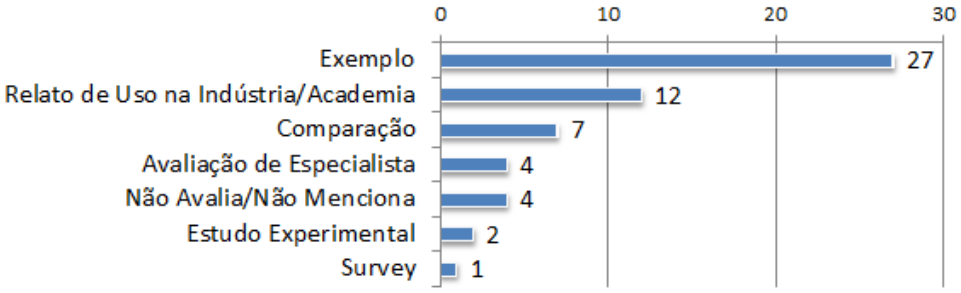


Figura 26 – Métodos de Avaliação.

3.4. Disseminação dos Resultados

Essa etapa foi alcançada pela publicação do artigo científico "*Software Process Lines: A Systematic Literature Review*" (CARVALHO *et al.*, 2014b), na conferência internacional denominada *International Conference on Software Process Improvement and Capability Determination* (SPICE). Desta forma, pessoas potencialmente interessadas nos resultados podem acessá-los. Além disso, a própria escrita e publicação desta dissertação de mestrado contribui para a disseminação.

3.5. Ameaças à Validade

Para analisar as ameaças à validade desta RSL, alguns fatores são tratados separadamente como segue:

Descobrir o maior número de estudos primários: para esse propósito, as expressões de busca foram revisadas por um segundo pesquisador, estudante de mestrado em Ciência da Computação. Além disso, as expressões de busca foram elaboradas de modo a possuir sinônimos e tanto o singular quanto o plural dos principais termos. Quando um determinado sinônimo ou plural não tinha impacto no número de publicações retornadas, era omitido. Entretanto, o número de bibliotecas digitais usadas (apenas três) pode ser considerado pequeno em relação a outras revisões sistemáticas com tema similar - Martínez-Ruiz *et al.* (2012) usaram cinco e Rocha e Fantinato (2013) usaram sete. Outro risco foi a falta de acesso a algumas publicações que eram pagas ou que não foram encontradas. Por fim, somente publicações escritas em inglês eram lidas. Por todos esses fatores, esta RSL pode ter ignorado alguns estudos primários potencialmente relevantes.

Evitar a rejeição precoce de estudos primários potencialmente relevantes: para evitar tal rejeição, no primeiro filtro do processo se seleção, publicações que tratavam de temas similares eram aceitas mesmo quando conceitos de linha de processos não eram evidentes à primeira vista. Além disso, em caso de dúvida as publicações também eram aceitas. Entretanto, a falta de claridade e inconsistências encontradas em algumas publicações, somando com restrições de tempo (que impossibilitava a consulta de outros pesquisadores) pode ter conduzido à rejeição de algum estudo primário potencialmente relevante.

Evitar a classificação errônea de estudos primários: a extração de dados foi uma tarefa complicada, visto que havia muitos conceitos com definições conflituosas ou ambíguas. Não existe uma padronização entre os autores para a utilização de termos e conceitos relacionados a linha de processos. Além disso, muitas informações estavam incompletas ou

mesmo incorretas nas publicações. Como uma medida para mitigar o risco de classificação incorreta, outros pesquisadores eram consultados em caso de dúvidas; entretanto, isso não era sempre possível. Por esses fatores, alguma classificação incorreta pode ter ocorrido e não é garantido que os resultados da extração de dados realizada por outros pesquisadores sejam idênticos aos resultados apresentados neste trabalho.

Aumentar a transparência e capacidade de repetição: esses fatores são melhorados pela documentação e disseminação das questões de pesquisa, expressões de busca, critérios de inclusão e exclusão, bibliotecas digitais usadas, processo seguido e as referências de todas as publicações aceitas.

4. Requisitos para a Representação de Variações

Este Capítulo apresenta requisitos para a representação de variações em linha de processos de software. Em seguida, metamodelos existentes são comparados em relação aos requisitos estabelecidos.

4.1. Introdução

Segundo Simmonds *et al.* (2013), a expressividade do metamodelo escolhido é um dos fatores mais críticos para a adoção com sucesso do paradigma linha de processos de software por empresas. Essa foi a principal motivação para a coleta de requisitos realizada neste trabalho. A expressividade diz respeito aos tipos de variações e restrições que podem ser expressas pelo metamodelo e aos tipos de elementos de processo que podem sofrer variações.

Os requisitos para a representação de variações em linha de processos de software foram coletados a partir da análise de abordagens existentes. Para isso, foram aproveitadas as publicações retornadas pela execução da revisão sistemática da literatura apresentada no Capítulo 3. Além disso, foi realizada a análise de uma dissertação de mestrado (TEIXEIRA, 2011) e uma tese de doutorado (BARRETO, 2011) que também tratam do assunto.

Todos os trabalhos analisados especificaram metamodelos e/ou notações gráficas para linha de processos de software, servindo de fontes para os requisitos apresentados neste Capítulo.

4.2. Requisitos

Por motivo de organização e facilidade de compreensão, os requisitos foram classificados de acordo com as categorias apresentadas na Tabela 3.

Tabela 3 – Categorias de Requisitos.

Categoria	Descrição
Requisitos de Variações	Agrupamento com os requisitos que estabelecem os tipos de variações que podem ser expressas pelo metamodelo.
Requisitos de Cardinalidades	Agrupamento com os requisitos que estabelecem os tipos de cardinalidades que podem ser expressas no metamodelo. Uma cardinalidade determina o número mínimo (cardinalidade mínima) e máximo (cardinalidade máxima) de elementos de uma relação.
Requisitos de Elementos de Processo	Agrupamento com os requisitos que estabelecem os tipos de elementos de processo que podem ser alvos de variações.
Requisitos de Dependências	Agrupamento com os requisitos que estabelecem os tipos de dependências entre elementos que podem ser expressas pelo metamodelo.

A Tabela 4 apresenta os requisitos para a representação de variações, de acordo com as categorias. Nas próximas seções, cada um desses requisitos será abordado com mais detalhes.

Tabela 4 – Requisitos para a Representação de Variações.

Categoria	Requisito	
	Id.	**Nome**
Requisitos de Variações	R1.1	Elementos opcionais e obrigatórios
	R1.2	Pontos de variação e variantes
	R1.3	Ponto de variação aberto ou fechado
	R1.4	*Binding time* de um ponto de variação
	R1.5	*Binding time* de elementos opcionais
	R1.6	Variante(s) selecionado(s) por padrão
	R1.7	Raciocínio para escolha de variante
	R1.8	Raciocínio para escolha de elemento opcional
	R1.9	Variações transversais
Requisitos de Cardinalidades	R2.1	Cardinalidades entre ponto de variação e variantes
	R2.2	Cardinalidades de instâncias
	R2.3	Cardinalidades entre elementos
Requisitos de Elementos de Processo	R3.1	Atividade
	R3.2	Papel
	R3.3	Produto de trabalho
	R3.4	Ferramenta
	R3.5	Conexão
Requisitos de Dependências	R4.1	Requerimento
	R4.2	Exclusão
	R4.3	Sugestão
	R4.4	Substituição

4.2.1. Requisitos de Variações

Os requisitos de variações se referem aos tipos de variações que podem ser expressas pelo metamodelo. Os seguintes requisitos foram identificados:

- **R.1.1 – Elementos opcionais e obrigatórios:** permitir expressar que um elemento é opcional, ou seja, pode ser selecionado ou não; e permitir expressar que um elemento é obrigatório, ou seja, deve ser selecionado compulsoriamente para compor o processo instanciado.
- **R1.2 – Pontos de variação e variantes:** permitir expressar pontos de variação e seus possíveis variantes.
- **R.1.3 – Ponto de variação aberto ou fechado:** permitir expressar que os variantes de um ponto de variação fazem parte de um conjunto predeterminado especificado durante a modelagem da linha de processos (ponto de variação fechado). Nesse caso, é possível selecionar somente variantes previamente modelados na engenharia de domínio. A outra possibilidade é permitir expressar que um ponto de variação pode ser satisfeito por um variante que não estava previsto inicialmente durante a engenharia de domínio (ponto de variação aberto).
- **R1.4 – *Binding time* de um ponto de variação:** permitir expressar que os variantes de um ponto de variação devem ser escolhidos em tempo de adaptação ou em tempo de execução. No último caso, a escolha pode ser postergada até que a atividade seja executada. Restrições: não deve ser possível instanciar uma linha de processos até que todos os pontos de variação de tempo de adaptação tenham seus variantes selecionados.
- **R1.5 – *Binding time* de elementos opcionais:** permitir expressar que a decisão de selecionar ou não um elemento opcional deve ser tomada durante a adaptação do processo ou pode ser postergada para a fase de execução do processo. Restrições: não deve ser possível instanciar uma linha de processos até que todos os elementos opcionais de tempo de adaptação sejam selecionados ou deselecionados.
- **R1.6 – Variante(s) selecionado(s) por padrão:** permitir expressar um ou mais variantes que devem ser selecionados por padrão (pré-selecionados) caso não seja tomada uma decisão. Uma aplicabilidade desse conceito é pré-selecionar o(s) variante(s) mais comumente utilizado(s) para que o processo de instanciação seja

agilizado. Restrições: o número de variantes selecionados por padrão não pode ser maior que a cardinalidade máxima do ponto de variação.

- **R1.7 – Raciocínio para escolha de variante:** permitir expressar o raciocínio para a escolha de um variante, ou seja, informações relevantes sobre um determinado variante de tal modo que possa auxiliar na tomada de decisão sobre quando a sua escolha é mais adequada.
- **R1.8 – Raciocínio para escolha de elemento opcional:** permitir expressar o raciocínio para a escolha de um elemento opcional, ou seja, informações relevantes sobre um determinado elemento de tal modo que possa auxiliar na tomada de decisão sobre quando a sua escolha é mais adequada.
- **R1.9 – Variações transversais:** permitir expressar que variações são transversais. Esse tipo de variação utiliza conceitos de engenharia de software orientada a aspectos para a representação de variações que afetam o processo como um todo, e não são apenas pontuais.

4.2.2. Requisitos de Cardinalidades

Os requisitos de cardinalidades se referem às cardinalidades mínima e máxima que podem ser expressas pelo metamodelo. Os seguintes requisitos foram identificados:

- **R2.1 – Cardinalidades entre ponto de variação e variantes:** permitir expressar a cardinalidade mínima, isto é, o número mínimo de variantes que devem ser selecionados em um ponto de variação; especificar a cardinalidade máxima, isto é, o número máximo de variantes que devem ser selecionados em um ponto de variação.
- **R2.2 – Cardinalidades de instâncias:** permitir expressar a cardinalidade mínima de instâncias, isto é, o número mínimo de instâncias que devem existir no processo instanciado; permitir expressar a cardinalidade máxima de instâncias, isto é, o número máximo de instâncias que podem existir no processo instanciado.
- **R2.3 – Cardinalidades entre elementos:** permitir expressar a cardinalidade mínima de relação, isto é, o número mínimo de relações entre elementos; permitir expressar a cardinalidade máxima de relações, isto é, o número máximo de relações que podem existir entre elementos. Exemplo: uma atividade (de programação) deve estar associada a pelo menos duas instâncias do papel programador (para satisfazer à prática de programação por pares).

4.2.3. Requisitos de Elementos de Processo

Requisitos que se referem a que tipos de elementos de processo podem ser representados com variações. Os tipos de variações considerados são: opcionais, obrigatórios e pontos de variação e variantes. Para compor o grupo de tipos de elementos de processo, foram considerados os tipos mais comumente encontrados como resultado da revisão sistemática da literatura:

- **R3.1 – Atividade:** permitir expressar atividades com variações. Ou seja, permite representar atividades obrigatórias, opcionais ou como ponto de variação e variantes. Além de atividades, são considerados qualquer tipo de elemento de processo comportamental com significado similar, como tarefa, passo, disciplina, cerimônia ou fase.
- **R3.2 – Papel:** permitir expressar papéis com variações. Ou seja, permite representar papéis obrigatórios, opcionais ou como ponto de variação e variantes.
- **R3.3 – Produto de trabalho:** permitir expressar produtos de trabalho com variações. Ou seja, permite representar produtos de trabalhos obrigatórios, opcionais ou como ponto de variação e variantes. São considerados também elementos com significado similar, como artefato, documento ou *template* de documento.
- **R3.4 – Ferramenta:** permite expressar ferramentas com variações. Ou seja, permite representar ferramentas como elementos obrigatórios, opcionais ou como ponto de variação e variantes.
- **R3.5 – Conexão:** permitir expressar variações em uma conexão entre elementos, ou seja, como obrigatória, opcional ou ponto de variação e variantes. Essas relações podem existir entre elementos de processo do mesmo tipo ou, ainda, entre diferentes tipos. Exemplo: conectores entre atividades ou uma relação entre um papel e uma atividade.

4.2.4. Requisitos de Dependências

Um dos princípios de um metamodelo robusto para processos de software é a possibilidade de especificação de restrições entre os elementos (SIMMONDS *et al.*, 2013, p. 33). A importância desse princípio se deve ao fato de que somente um subconjunto de todas as combinações de elementos de processo são corretas e consistentes. Assim, a arquitetura da linha

de processos deve especificar dependências de forma a permitir apenas a instanciação de membros válidos da linha de processos.

Portanto, esta seção estabelece os requisitos com os tipos de dependências entre elementos que podem ser expressas pelo metamodelo. As dependências em um metamodelo garantem que os modelos gerados são consistentes em relação às regras de boa formação. Os seguintes requisitos, baseados no trabalho de Aiello *et al.* (2010), foram identificados:

- **R4.1 – Requerimento:** permitir expressar dependências do tipo requerimento. Exemplo: se o elemento A for inserido, então o elemento B deve ser inserido.
- **R4.2 – Exclusão:** permitir expressar dependências do tipo exclusão: Exemplo: se o elemento A for inserido, então o elemento B não deve ser inserido.
- **R4.3 – Sugestão:** permitir expressar dependências do tipo sugestão. Exemplo: se o elemento A for inserido, então é recomendável que o elemento B seja inserido. Observação: enquanto que dependência do tipo requerimento obriga a seleção de outro elemento, uma dependência do tipo sugestão apenas recomenda.
- **R4.4 – Substituição:** permitir expressar dependências do tipo substituição. Exemplo: se o elemento A não for inserido, então o elemento B deve ser inserido.

4.3. Comparação

Esta seção apresenta a comparação entre os trabalhos, segundo os requisitos estabelecidos na seção 4.2 (p. 62). A comparação foi realizada de forma completa, ou seja, todos os trabalhos foram comparados em relação a todos os requisitos. Entretanto, nesta seção será apresentado apenas o atendimento ou não de cada um dos trabalhos em relação a cada requisito. As evidências que levaram o autor desta dissertação a chegar a esses resultados são apresentadas na seção 4.4 (p. 71).

Para a melhor visualização da comparação, a Tabela 5 apresenta os símbolos que demonstram o atendimento total, parcial ou o não atendimento aos requisitos estabelecidos.

Tabela 5 – Relação entre Símbolos e Satisfação aos Requisitos.

Símbolo	Descrição
✔	• Quando se referir à categoria de requisito, indica que todos os requisitos da categoria foram satisfeitos. • Quando se referir a requisito, indica que o requisito foi satisfeito.
⚠	• Indica que pelo menos um requisito da categoria foi satisfeito, entretanto, pelo menos um requisito não foi satisfeito. Ou seja, o número de requisitos satisfeitos na categoria é maior que zero e menor que o número total de requisitos da categoria.

Símbolo	Descrição
	• Não se aplica a requisitos.
✗	• Quando se referir à categoria de requisito, indica que nenhum dos requisitos da categoria foi satisfeito. • Quando se referir a requisito, indica que o requisito não é satisfeito ou não existem informações suficientes para avaliar sua satisfação.

A Tabela 6 apresenta o resultado da comparação dos trabalhos em relação à satisfação de requisitos por categoria. Esta visão tem como objetivo apresentar o resultado da comparação em um alto nível de abstração.

Tabela 6 – Satisfação de Requisitos por Categoria.

	Requisitos de Variações	Requisitos de Cardinalidades	Requisitos de Elementos de Processo	Requisitos de Dependências
Alegría e Bastarrica (2012)	⚠	⚠	⚠	⚠
Alegría *et al.* (2011)	⚠	⚠	⚠	✗
Barreto (2011)	⚠	⚠	⚠	⚠
Barreto *et al.* (2010)	⚠	⚠	⚠	⚠
Barreto *et al.* (2011)	⚠	⚠	⚠	⚠
Costache *et al.* (2011)	⚠	✗	⚠	⚠
Golpayegani *et al.* (2013)	⚠	✗	⚠	⚠
Hurtado *et al.* (2013)	⚠	⚠	⚠	⚠
Martínez-Ruiz *et al.* (2008)	⚠	⚠	⚠	⚠
Martínez-Ruiz *et al.* (2011a)	⚠	✗	⚠	✗
Martínez-Ruiz *et al.* (2011b)	⚠	⚠	⚠	⚠
Martínez-Ruiz *et al.* (2011c)	⚠	✗	⚠	⚠
Martínez-Ruiz *et al.* (2013a)	⚠	⚠	⚠	⚠
Rouillé *et al.* (2012)	⚠	⚠	✓	⚠
Teixeira (2011)	⚠	⚠	⚠	⚠
Ternité (2009)	⚠	✗	⚠	✗
Washizaki (2006a)	⚠	✗	✓	⚠
Washizaki (2006b)	⚠	✗	✓	⚠

A Tabela 7 apresenta a comparação dos trabalhos em relação aos requisitos de variações.

Tabela 7 – Satisfação de Requisitos de Variações.

	R1.1	R1.2	R1.3	R1.4	R1.5	R1.6	R1.7	R1.8	R1.9
Alegría e Bastarrica (2012)	✓	✓	✗	✗	✗	✗	✓	✓	✓
Alegría *et al.* (2011)	✓	✓	✗	✗	✗	✗	✓	✓	✓
Barreto (2011)	✓	✓	✗	✗	✗	✗	✗	✗	✗
Barreto *et al.* (2010)	✓	✓	✗	✗	✗	✗	✗	✗	✗
Barreto *et al.* (2011)	✓	✓	✗	✗	✗	✗	✗	✗	✗
Costache *et al.* (2011)	✓	✓	✗	✗	✗	✗	✗	✗	✗
Golpayegani *et al.* (2013)	✓	✓	✗	✗	✗	✗	✗	✗	✓
Hurtado *et al.* (2013)	✓	✓	✗	✗	✗	✗	✓	✓	✓
Martínez-Ruiz *et al.* (2008)	✓	✓	✗	✗	✗	✗	✗	✗	✗
Martínez-Ruiz *et al.* (2011a)	✗	✓	✗	✗	✗	✗	✗	✗	✗
Martínez-Ruiz *et al.* (2011b)	✓	✓	✗	✗	✗	✗	✓	✓	✓
Martínez-Ruiz *et al.* (2011c)	✗	✓	✗	✗	✗	✗	✗	✗	✓
Martínez-Ruiz *et al.* (2013a)	✓	✓	✗	✗	✗	✗	✗	✗	✓
Rouillé *et al.* (2012)	✓	✓	✗	✗	✗	✗	✗	✗	✗
Teixeira (2011)	✓	✓	✗	✗	✗	✗	✗	✗	✗
Ternité (2009)	✓	✓	✗	✗	✗	✗	✗	✗	✗
Washizaki (2006a)	✓	✓	✗	✗	✗	✗	✗	✗	✗
Washizaki (2006b)	✓	✓	✗	✗	✗	✗	✗	✗	✗

A Tabela 8 apresenta a comparação dos trabalhos em relação aos requisitos de cardinalidades.

Tabela 8 – Satisfação de Requisitos de Cardinalidades.

	R2.1	R2.2	R2.3
Alegría e Bastarrica (2012)	✓	✗	✗
Alegría *et al.* (2011)	✓	✗	✗
Barreto (2011)	✓	✗	✗
Barreto *et al.* (2010)	✓	✗	✗
Barreto *et al.* (2011)	✓	✗	✗
Costache *et al.* (2011)	✗	✗	✗
Golpayegani *et al.* (2013)	✗	✗	✗
Hurtado *et al.* (2013)	✓	✗	✗
Martínez-Ruiz *et al.* (2008)	✓	✗	✗
Martínez-Ruiz *et al.* (2011a)	✗	✗	✗
Martínez-Ruiz *et al.* (2011b)	✗	✓	✗
Martínez-Ruiz *et al.* (2011c)	✗	✗	✗
Martínez-Ruiz *et al.* (2013a)	✓	✗	✗
Rouillé *et al.* (2012)	✓	✗	✗
Teixeira (2011)	✓	✗	✗
Ternité (2009)	✗	✗	✗
Washizaki (2006a)	✗	✗	✗
Washizaki (2006b)	✗	✗	✗

A Tabela 9 apresenta a comparação dos trabalhos em relação aos requisitos de elementos de processo.

Tabela 9 – Satisfação de Requisitos de Elementos de Processo.

	R3.1	R3.2	R3.3	R3.4	R3.5
Alegría e Bastarrica (2012)	✓	✓	✓	✗	✗
Alegría *et al.* (2011)	✓	✓	✓	✗	✗
Barreto (2011)	✓	✗	✗	✗	✓
Barreto *et al.* (2010)	✓	✗	✗	✗	✗
Barreto *et al.* (2011)	✓	✗	✗	✗	✗
Costache *et al.* (2011)	✓	✗	✓	✗	✗
Golpayegani *et al.* (2013)	✓	✗	✗	✗	✗
Hurtado *et al.* (2013)	✓	✓	✓	✗	✗
Martínez-Ruiz *et al.* (2008)	✓	✓	✓	✗	✗
Martínez-Ruiz *et al.* (2011a)	✓	✓	✓	✓	✗
Martínez-Ruiz *et al.* (2011b)	✓	✓	✓	✗	✗
Martínez-Ruiz *et al.* (2011c)	✓	✓	✓	✗	✗
Martínez-Ruiz *et al.* (2013a)	✓	✓	✓	✗	✗
Rouillé *et al.* (2012)	✓	✓	✓	✓	✓
Teixeira (2011)	✓	✓	✓	✗	✓
Ternité (2009)	✗	✓	✓	✗	✓
Washizaki (2006a)	✓	✓	✓	✓	✗
Washizaki (2006b)	✓	✓	✓	✓	✗

A Tabela 10 apresenta a comparação dos trabalhos em relação aos requisitos de dependências.

Tabela 10 – Satisfação de Requisitos de Dependências.

	R4.1	R4.2	R4.3	R4.4
Alegría e Bastarrica (2012)	✓	✓	✗	✗
Alegría *et al.* (2011)	✗	✗	✗	✗
Barreto (2011)	✓	✓	✗	✗
Barreto *et al.* (2010)	✓	✓	✗	✗
Barreto *et al.* (2011)	✓	✓	✗	✗
Costache *et al.* (2011)	✗	✓	✗	✗
Golpayegani *et al.* (2013)	✓	✓	✗	✗
Hurtado *et al.* (2013)	✓	✗	✗	✗
Martínez-Ruiz *et al.* (2008)	✓	✓	✗	✗
Martínez-Ruiz *et al.* (2011a)	✗	✗	✗	✗
Martínez-Ruiz *et al.* (2011b)	✓	✗	✗	✗
Martínez-Ruiz *et al.* (2011c)	✓	✗	✗	✗
Martínez-Ruiz *et al.* (2013a)	✓	✓	✗	✗
Rouillé *et al.* (2012)	✓	✗	✗	✗
Teixeira (2011)	✓	✓	✗	✗
Ternité (2009)	✗	✗	✗	✗
Washizaki (2006a)	✓	✓	✗	✗
Washizaki (2006b)	✓	✓	✗	✗

4.4. Justificativas de Comparação

Esta seção apresenta a comparação entre os metamodelos, justificando, através de evidências encontradas nas publicações, como os metamodelos comparados na seção 4.3 satisfazem ou não aos requisitos estabelecidos na seção 4.2.

Para caracterizar um requisito como atendido, foi utilizada como critério a afirmação explícita dos autores ou a apresentação de partes da notação gráfica ou do metamodelo que justifique a satisfação. Caso a informação não seja encontrada, o requisito foi considerado como não satisfeito.

Por motivo de organização, as publicações foram analisadas individualmente, cada uma em uma das subseções seguintes. Além disso, os requisitos foram agrupados de acordo com as categorias. Os conceitos apresentados nos metamodelos são escritos entre aspas para facilitar a sua identificação. As próximas subseções apresentam cada um dos trabalhos analisados.

4.4.1. Alegría e Bastarrica (2012)

O trabalho de Alegría e Bastarrica (2012) apresenta uma linguagem, denominada Experimental SPEM (eSPEM), baseada na linguagem SPEM (OMG, 2008), para a representação de variações em linha de processos.

Requisitos de variações: a representação de elementos opcionais é possível através do modelo de características (Figura 27) e da notação gráfica (Figura 28), satisfazendo ao requisito "R1.1 – Elementos opcionais e obrigatórios". Também é possível registrar o contexto em que um elemento opcional é adequado, através do metamodelo de contexto de processo de software, satisfazendo ao requisito "R1.8 – Raciocínio para escolha de elemento opcional".

Entretanto, não é possível especificar o momento do ciclo de vida em que os elementos opcionais devem ser selecionados (não satisfazendo ao requisito "R1.5 – *Binding time* de elementos opcionais").

Figura 27 – Modelo de Características (ALEGRÍA; BASTARRICA, 2012).

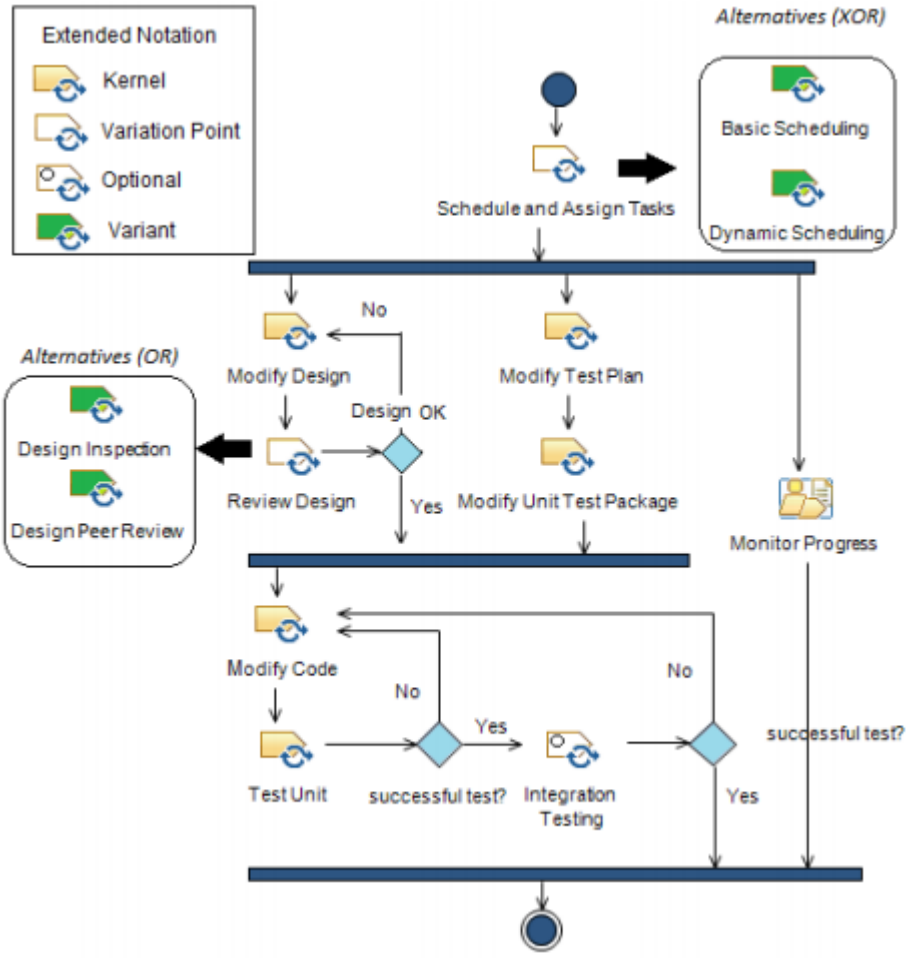


Figura 28 – Notação Gráfica (ALEGRÍA; BASTARRICA, 2012).

Pontos de variação e variantes são representados através modelo de características (Figura 27) e da notação gráfica (Figura 28), satisfazendo ao requisito "R1.2 – Pontos de variação e variantes".

Porém, não existem informações no metamodelo sobre ponto de variação aberto ou fechado (não satisfazendo ao requisito "R1.3 – Ponto de variação aberto ou fechado"), sobre o momento do ciclo de vida em que o(s) variante(s) deve(m) ser escolhido(s) (não satisfazendo ao requisito "R1.4 – *Binding time* de um ponto de variação") e nem a especificação de variantes selecionados por padrão (não satisfazendo ao requisito "R1.6 – Variante(s) selecionado(s) por padrão").

O metamodelo de configuração de processo de software define classes a partir das quais é possível registrar o raciocínio que auxilie na seleção do variante mais adequado (atendendo ao requisito "R1.7 – Raciocínio para escolha de variante").

Não é possível representar variações transversais, não sendo satisfeito o requisito "R1.9 – Variações transversais".

Requisitos de cardinalidades: segundo os autores, a abordagem utiliza o modelo de características proposto por Czarnecki *et al.* (2005), que permite expressar cardinalidades entre pontos de variação e variantes, como pode ser verificado na Figura 27 (satisfazendo ao requisito "R2.1 – Cardinalidades entre ponto de variação e variantes"). Entretanto, não são tradadas cardinalidades de instâncias e entre elementos (não satisfazendo aos requisitos "R2.2 – Cardinalidades de instâncias" e "R2.3 – Cardinalidades entre elementos").

Requisitos de elementos de processo: a notação gráfica permite expressar variações em atividades e tarefas (satisfazendo ao requisito "R3.1 – Atividade"); em papéis (satisfazendo ao requisito "R3.2 – Papel"); e produtos de trabalho (satisfazendo ao requisito "R3.3 – Produto de trabalho"). Entretanto, não foram encontradas informações sobre a representação de variações em ferramentas e conexões de elementos de processo (não satisfazendo aos requisitos "R3.4 – Ferramenta" e "R3.5 – Conexão").

Requisitos de dependências: foi explicitamente definida a possibilidade de definir dependências dos tipos requerimento e exclusão (satisfazendo aos requisitos "R4.1 – Requerimento" e "R4.2 – Exclusão"). Porém, não existe a possibilidade de definir outros tipos de dependências (não satisfazendo aos requisitos "R4.3 – Sugestão" e "R4.4 – Substituição").

4.4.2. Alegría *et al.* (2011)

O trabalho de Alegría *et al.* (2011) apresenta uma linguagem, denominada Experimental SPEM (eSPEM), baseada na linguagem SPEM (OMG, 2008), para a representação de variações em linha de processos.

Requisitos de variações: a representação de elementos opcionais é possível através do atributo booleano "isOptional", presente nas classes "WorkBreakDownElement" e suas subclasses (Figura 29), satisfazendo ao requisito "R1.1 – Elementos opcionais e obrigatórios". Também é possível registrar o contexto em que um elemento opcional é adequado, através do metamodelo de contexto de processo de software (Figura 32), satisfazendo ao requisito "R1.8 – Raciocínio para escolha de elemento opcional". Esse raciocínio pode ser definido através da classe "ContextAttributeConfiguration", que relaciona um "ContextAttribute" com um dos possíveis "ContextAttributeValues".

Entretanto, não é possível especificar o momento do ciclo de vida em que os elementos opcionais devem ser selecionados (não satisfazendo ao requisito "R1.5 – *Binding time* de elementos opcionais").

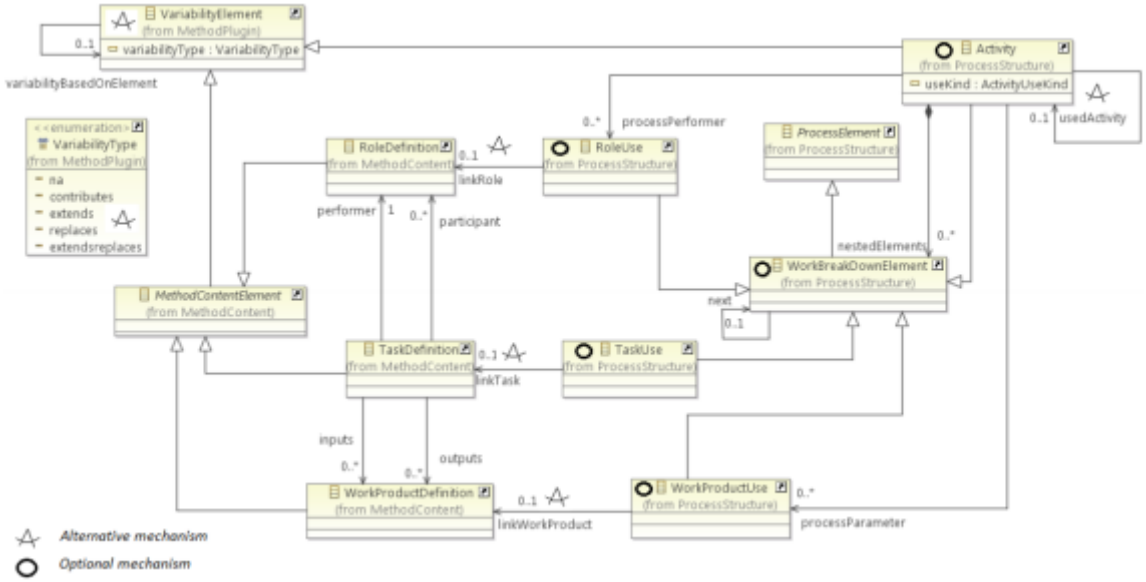


Figura 29 – Metamodelo eSPEM (ALEGRÍA *et al.*, 2011).

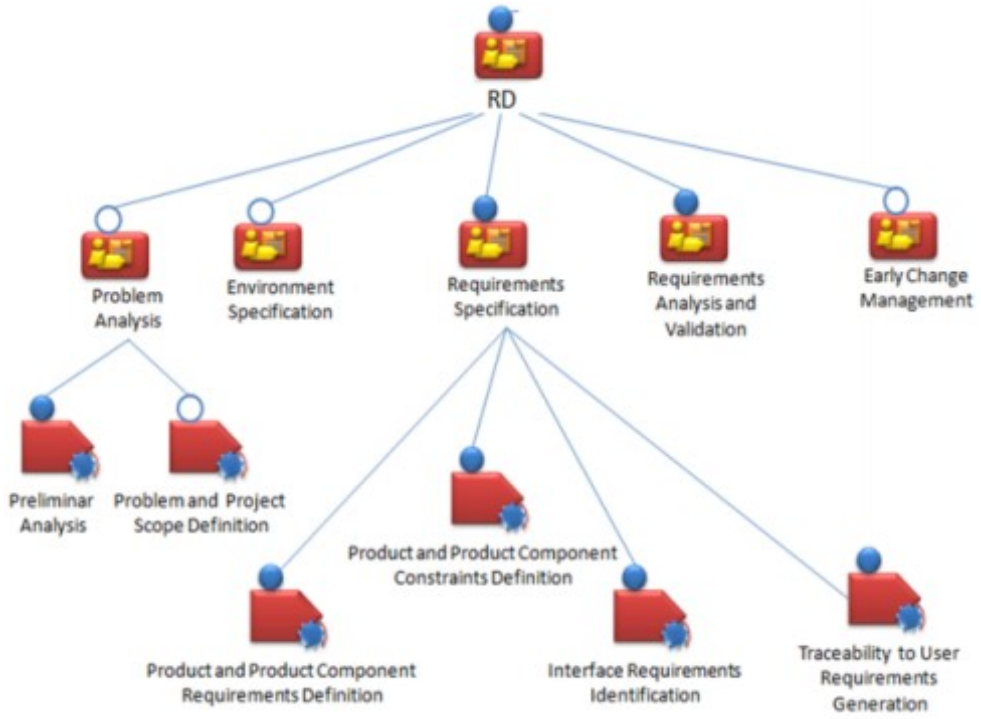


Figura 30 – Processo de Desenvolvimento de Requisitos (ALEGRÍA *et al.*, 2011).

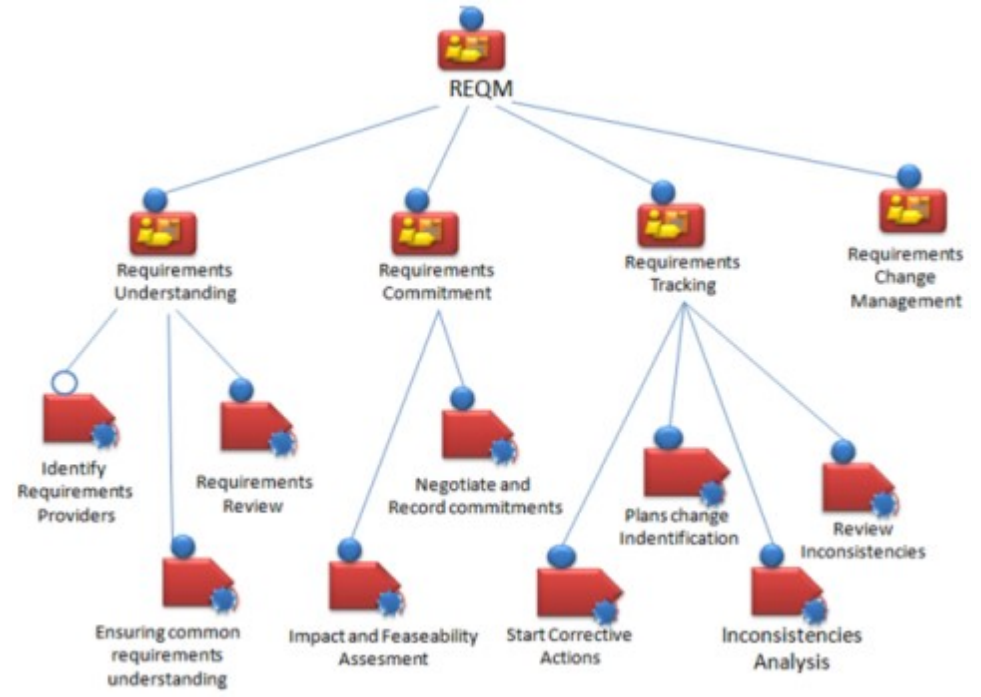


Figura 31 – Processo de Gerência de Requisitos (ALEGRÍA *et al.*, 2011).

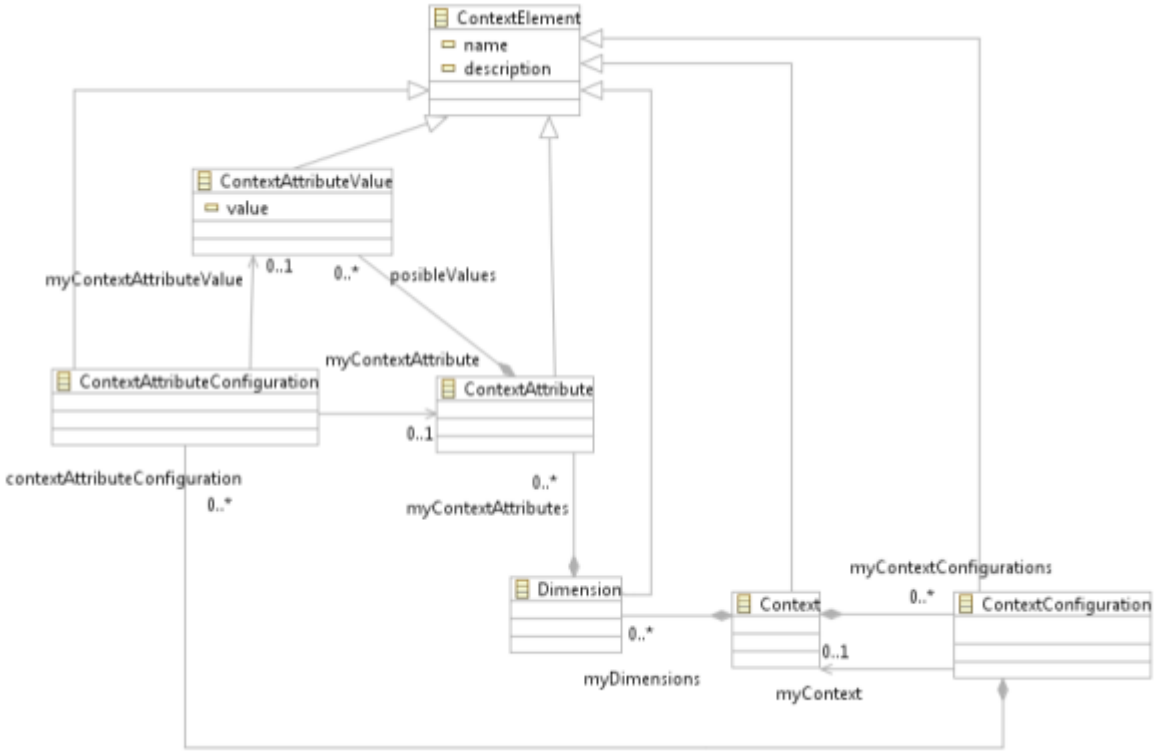


Figura 32 – Metamodelo de Contexto de Processo (ALEGRÍA *et al.*, 2011).

Pontos de variação e variantes são representados através de relações entre subclasses de "VariabilityElement" (ponto de variação) e subclasses de "ProcessElement" (variante) (Figura 29), satisfazendo ao requisito "R1.2 – Pontos de variação e variantes".

Porém, não existem informações no metamodelo sobre ponto de variação aberto ou fechado (não satisfazendo ao requisito "R1.3 – Ponto de variação aberto ou fechado"), sobre o momento do ciclo de vida em que o(s) variante(s) deve(m) ser escolhido(s) (não satisfazendo ao requisito "R1.4 – *Binding time* de um ponto de variação") e nem a especificação de variantes selecionados por padrão (não satisfazendo ao requisito "R1.6 – Variante(s) selecionado(s) por padrão").

O metamodelo de configuração de processo de software define classes a partir das quais é possível registrar o raciocínio que auxilie na seleção do variante mais adequado (atendendo ao requisito "R1.7 – Raciocínio para escolha de variante").

Não é possível representar variações transversais, não sendo satisfeito o requisito "R1.9 – Variações transversais".

Requisitos de cardinalidades: segundo os autores, a abordagem utiliza o modelo de características proposto por Czarnecki *et al.* (2005), que permite expressar cardinalidades entre pontos de variação e variantes (satisfazendo ao requisito "R2.1 – Cardinalidades entre ponto de variação e variantes"). Entretanto, não são tradadas cardinalidades de instâncias e entre elementos

(não satisfazendo aos requisitos "R2.2 – Cardinalidades de instâncias" e "R2.3 – Cardinalidades entre elementos").

Requisitos de elementos de processo: o metamodelo eSPEM permite expressar variações em atividades, através das classes "TaskUse", "TaskDefinition" e "Activity" (satisfazendo ao requisito "R3.1 – Atividade"); em papéis, através das classes "RoleUse" e "RoleDefinition" (satisfazendo ao requisito "R3.2 – Papel"); e produtos de trabalho, através das classes "WorkProductUse" e "WorkProductDefinition" (satisfazendo ao requisito "R3.3 – Produto de trabalho"). Entretanto, não foram encontradas informações sobre a representação de variações em ferramentas e conexões de elementos de processo (não satisfazendo aos requisitos "R3.4 – Ferramenta" e "R3.5 – Conexão"). A Figura 29 apresentam os tipos de elementos de processo com variações definidas no metamodelo.

Requisitos de dependências: os autores não apresentaram informações sobre a possibilidade de representar dependências (não satisfazendo aos requisitos "R4.1 – Requerimento", "R4.2 – Exclusão", "R4.3 – Sugestão" e "R4.4 – Substituição").

4.4.3. Barreto (2011)

Esse trabalho é uma tese de doutorado que apresenta uma evolução do metamodelo apresentado nos artigos de Barreto *et al.* (2010) e Barreto *et al.* (2011).

Requisitos de variações: o metamodelo representa elementos opcionais pela propriedade booleana "opcional" da classe "Item Elemento de Processo de Arquitetura de Processo" (Figura 34) (satisfazendo ao requisito "R1.1 – Elementos opcionais e obrigatórios"). Porém, não é possível definir o momento em que a escolha de um elemento opcional deve ser realizada (não satisfazendo ao requisito "R1.5 – *Binding time* de elementos opcionais") e nem o raciocínio para a sua seleção (não satisfazendo ao requisito "R1.8 – Raciocínio para escolha de elemento opcional").

Os conceitos de ponto de variação e variantes através das classes "Componente de Processo Abstrato" e "Componente de Processo Concreto". Um "Componente de Processo Abstrato" admite variações e pode ser implementado de diversas formas. Essa escolha é representada pelo relacionamento "implementa" entre um "Componente de Processo Abstrato" (ponto de variação) e vários possíveis "Componentes de Processo Concreto" (variantes) (Figura 33). Portanto, o requisito "R1.2 – Pontos de variação e variantes" é satisfeito.

Porém, apesar de representar pontos de variação e variantes, não foram encontradas informações sobre: a representação de pontos de variação abertos ou fechados (não satisfazendo ao requisito "R1.3 – Ponto de variação aberto ou fechado"); o momento em que os variantes

devem ser escolhidos (não satisfazendo ao requisito "R1.4 – *Binding time* de um ponto de variação"); especificação de variante(s) que é(são) selecionado(s) por padrão (não satisfazendo ao requisito "R1.6 – Variante(s) selecionado(s) por padrão"); e nem a definição do raciocínio para a escolha de cada variante (não satisfazendo ao requisito "R1.7 – Raciocínio para escolha de variante"). Além disso, não é possível representar variações transversais (não satisfazendo ao requisito "R1.9 – Variações transversais").

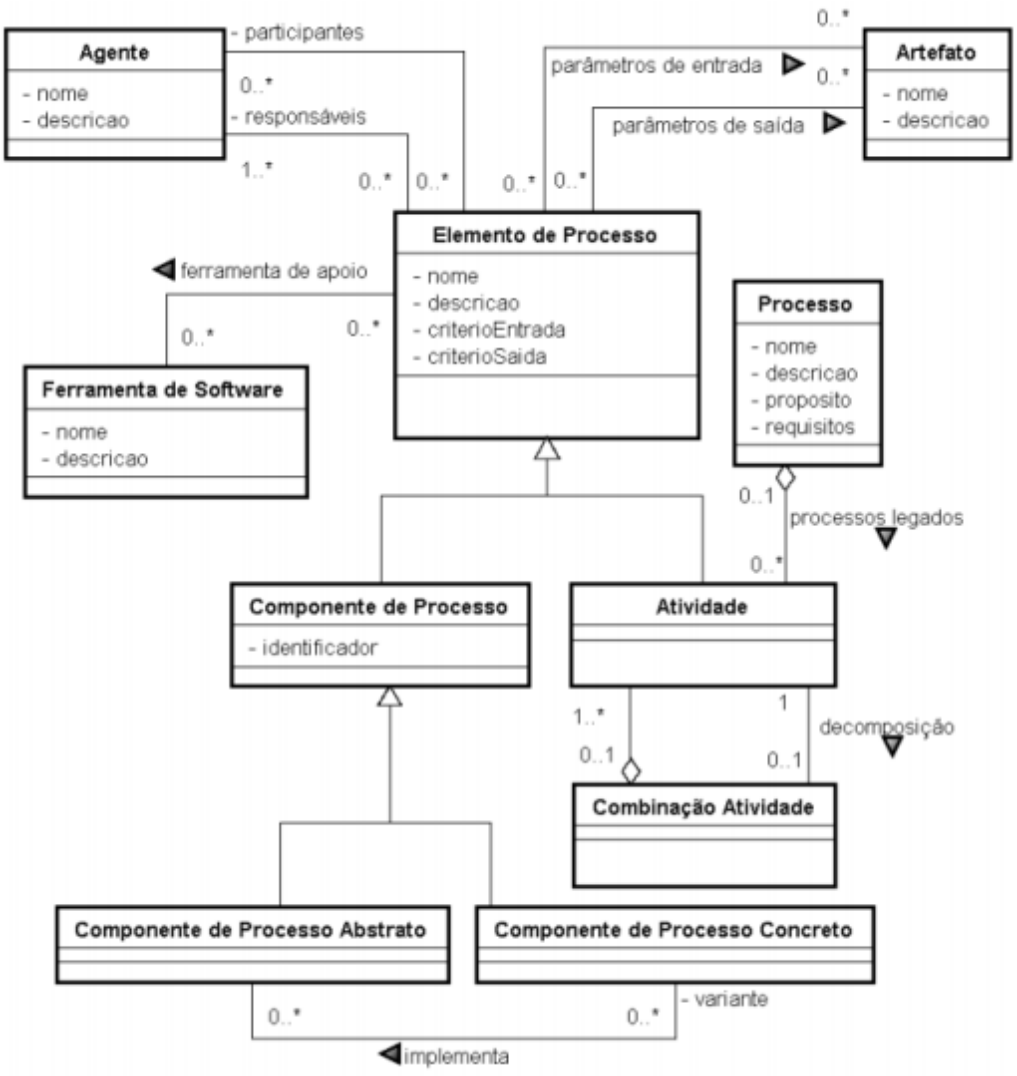


Figura 33 – Elementos de Processo (BARRETO, 2011).

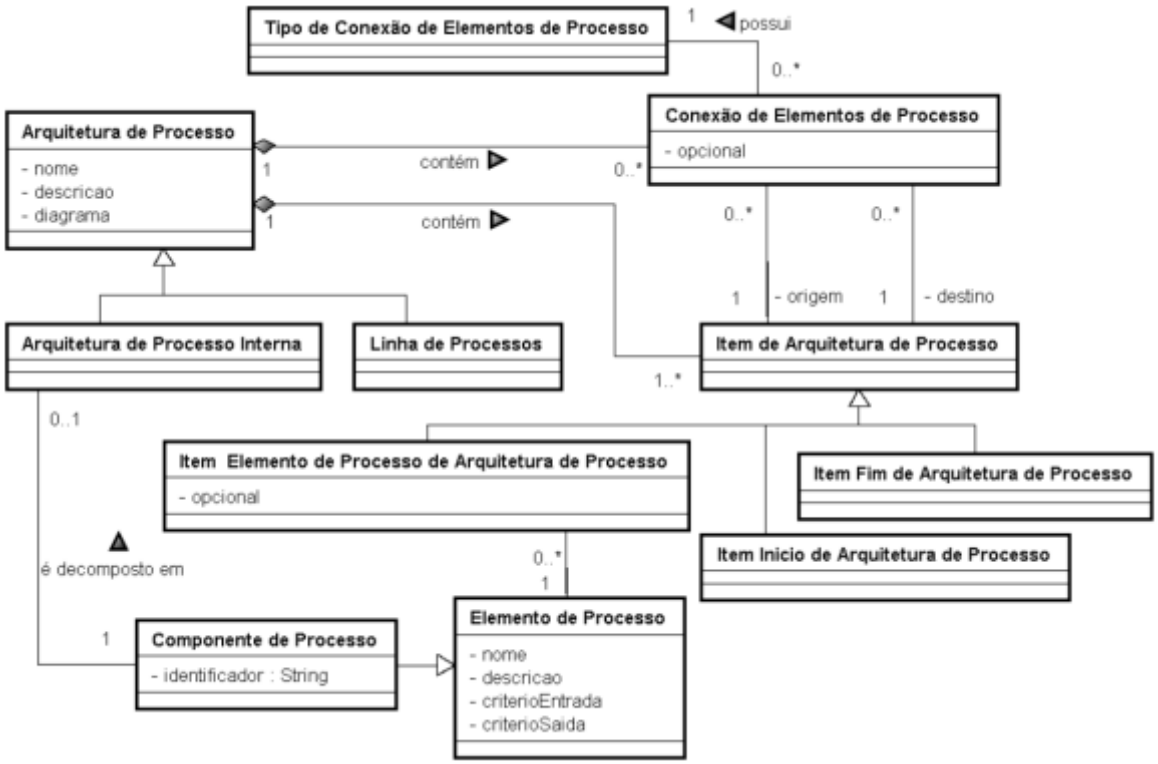


Figura 34 – Linha de Processos (BARRETO, 2011).

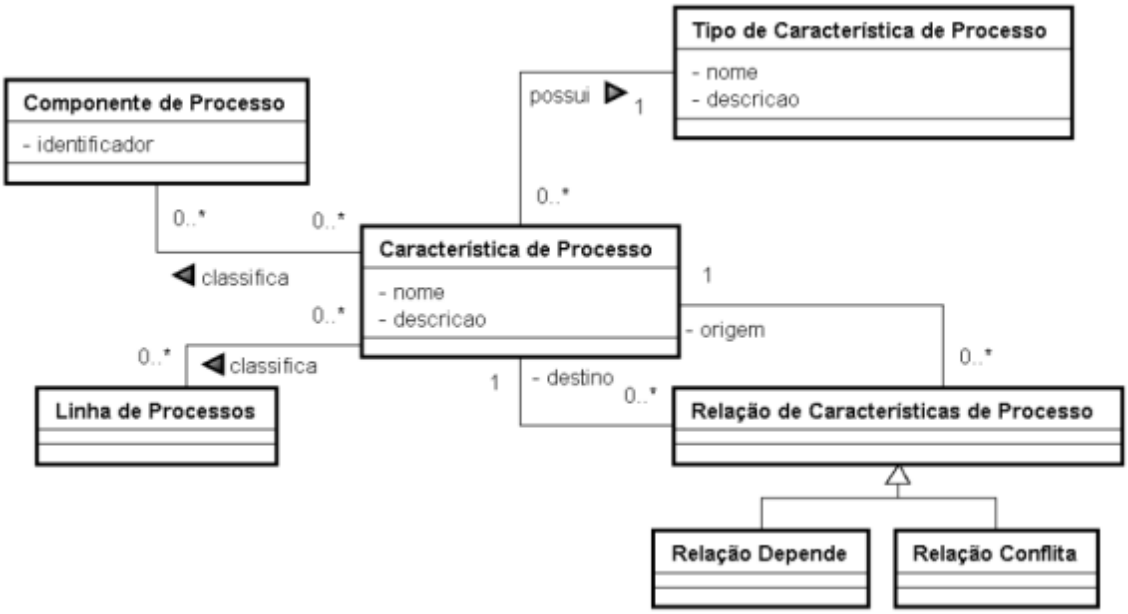


Figura 35 – Características de Processo (BARRETO, 2011).

Requisitos de cardinalidades: apesar de permitir a representação de pontos de variação e variantes, não é possível definir as cardinalidades mínima e máxima entre eles (não satisfazendo ao requisito "R2.1 – Cardinalidades entre ponto de variação e variantes"). Adicionalmente, também não é possível expressar as cardinalidades de instâncias e nem de relações entre elementos de processo, não satisfazendo aos requisitos "R2.2 – Cardinalidades de instâncias" e "R2.3 – Cardinalidades entre elementos".

Requisitos de elementos de processo: é possível representar variações em componentes de processo (Figura 33), que possuem um conceito semelhante a atividades, em atividades propriamente ditas (satisfazendo ao requisito "R3.1 – Atividade"), e em "Conexão de Elementos de Processo" (satisfazendo ao requisito "R3.5 - Conexão"). Entretanto, não é possível definir variações em elementos dos tipos papel, produto de trabalho e ferramenta, não sendo satisfeitos os requisitos "R3.2 – Papel", "R3.3 – Produto de trabalho" e "R3.4 – Ferramenta". Vale ressaltar que, apesar do metamodelo representar os tipos de elemento "Agente", "Artefato" e "Ferramenta de Software" (Figura 33), não é possível representar variações nesses tipos de elemento.

Requisitos de dependências: dependências são representadas pela hierarquia entre a superclasse "Relação de Características de Processo" e suas duas subclasses "Relação Dependente" (requerimento) e "Relação Conflita" (exclusão) (Figura 35). Dependências dos tipos sugestão e substituição não podem ser representadas, não satisfazendo aos requisitos "R4.3 – Sugestão" e "R4.4 – Substituição".

4.4.4. Barreto *et al.* (2010)

O trabalho apresenta um metamodelo para a representação de variações em linha de processos.

Requisitos de variações: o metamodelo representa elementos opcionais através do relacionamento "optional elements" entre as classes "ProcessArchitecture" e "ProcessElement" (Figura 36) (satisfazendo ao requisito "R1.1 – Elementos opcionais e obrigatórios"). Porém, não é possível definir o momento em que a escolha de um elemento opcional deve ser realizada (não satisfazendo ao requisito "R1.5 – *Binding time* de elementos opcionais") e nem o raciocínio para a sua seleção (não satisfazendo ao requisito "R1.8 – Raciocínio para escolha de elemento opcional").

Os autores utilizam os conceitos de ponto de variação e variantes através das classes "AbstractComponent" e "ConcreteComponent". Um "AbstractComponent" admite variações e pode ser implementado de diversas formas. Essa escolha é representada pelo relacionamento "implements" entre um "AbstractComponent" (ponto de variação) e vários possíveis "ConcreteComponent" (variantes) (Figura 36). Portanto, o requisito "R1.2 – Pontos de variação e variantes" é satisfeito.

Porém, apesar de representar pontos de variação e variantes, não foram encontradas informações sobre: a representação de pontos de variação abertos ou fechados (não satisfazendo ao requisito "R1.3 – Ponto de variação aberto ou fechado"); o momento em que os variantes

devem ser escolhidos (não satisfazendo ao requisito "R1.4 – *Binding time* de um ponto de variação"); especificação de variante(s) que é(são) selecionado(s) por padrão (não satisfazendo ao requisito "R1.6 – Variante(s) selecionado(s) por padrão"); e nem a definição do raciocínio para a escolha de cada variante (não satisfazendo ao requisito "R1.7 – Raciocínio para escolha de variante"). Além disso, não é possível representar variações transversais (não satisfazendo ao requisito "R1.9 – Variações transversais").

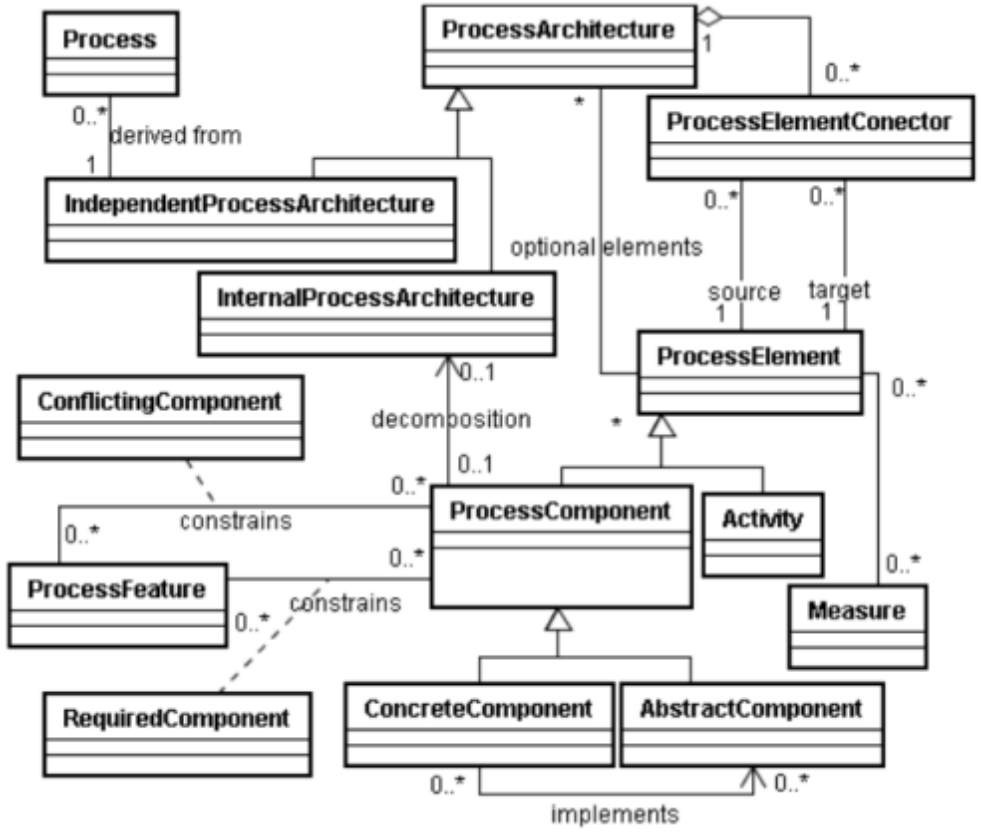


Figura 36 – Metamodelo (BARRETO *et al.*, 2010).

Requisitos de cardinalidades: apesar de permitir a representação de pontos de variação e variantes, não é possível definir as cardinalidades mínima e máxima entre eles (não satisfazendo ao requisito "R2.1 – Cardinalidades entre ponto de variação e variantes"). Adicionalmente, também não é possível expressar as cardinalidades de instâncias e nem de relações entre elementos de processo, não satisfazendo aos requisitos "R2.2 – Cardinalidades de instâncias" e "R2.3 – Cardinalidades entre elementos".

Requisitos de elementos de processo: é possível representar variações em componentes de processo (Figura 36), que possuem um conceito semelhante a atividades, e em atividades propriamente ditas (satisfazendo ao requisito "R3.1 – Atividade"). Entretanto, não é possível definir variações em elementos dos tipos papel, produto de trabalho, ferramenta e conexão, não sendo satisfeitos os requisitos "R3.2 – Papel", "R3.3 – Produto de trabalho" e "R3.4 – Ferramenta" e "R3.5 – Conexão".

Requisitos de dependências: existem dois tipos de dependências que podem ser representadas: requerimento e exclusão (satisfazendo aos requisitos "R4.1 – Requerimento" e "R4.2 – Exclusão"). Dependências são representadas pelas classes associativas "RequiredComponent" (requerimento) e "ConflictingComponent" (exclusão) (Figura 36). Dependências dos tipos sugestão e substituição não podem ser representadas, não satisfazendo aos requisitos "R4.3 – Sugestão" e "R4.4 – Substituição".

4.4.5. Barreto *et al.* (2011)

O trabalho apresenta um metamodelo para a representação de variações em linha de processos, apresentando uma evolução de Barreto *et al.* (2010).

Requisitos de variações: o metamodelo representa elementos opcionais através do relacionamento "optional elements" entre as classes "ProcessArchitecture" e "ProcessElement" (Figura 37) (satisfazendo ao requisito "R1.1 – Elementos opcionais e obrigatórios"). Porém, não é possível definir o momento em que a escolha de um elemento opcional deve ser realizada (não satisfazendo ao requisito "R1.5 – *Binding time* de elementos opcionais") e nem o raciocínio para a sua seleção (não satisfazendo ao requisito "R1.8 – Raciocínio para escolha de elemento opcional").

Os conceitos de ponto de variação e variantes através das classes "AbstractComponent" e "ConcreteComponent". Um "AbstractComponent" admite variações e pode ser implementado de diversas formas. Essa escolha é representada pelo relacionamento "implements" entre um "AbstractComponent" (ponto de variação) e vários possíveis "ConcreteComponent" (variantes) (Figura 37). Portanto, o requisito "R1.2 – Pontos de variação e variantes" é satisfeito.

Porém, não foram encontradas informações sobre: a representação de pontos de variação abertos ou fechados (não satisfazendo ao requisito "R1.3 – Ponto de variação aberto ou fechado"); o momento em que os variantes devem ser escolhidos (não satisfazendo ao requisito "R1.4 – *Binding time* de um ponto de variação"); especificação de variante(s) que é(são) selecionado(s) por padrão (não satisfazendo ao requisito "R1.6 – Variante(s) selecionado(s) por padrão"); e nem a definição do raciocínio para a escolha de cada variante (não satisfazendo ao requisito "R1.7 – Raciocínio para escolha de variante"). Além disso, não é possível representar variações transversais (não satisfazendo ao requisito "R1.9 – Variações transversais").

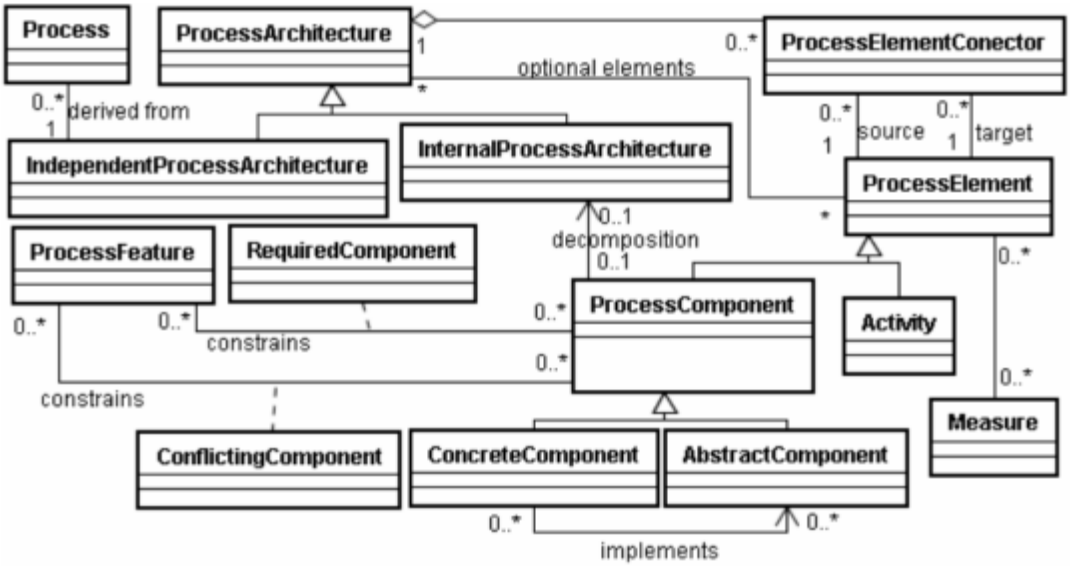


Figura 37 – Metamodelo (BARRETO *et al.*, 2011).

Requisitos de cardinalidades: apesar de permitir a representação de pontos de variação e variantes, não é possível definir as cardinalidades mínima e máxima entre eles (não satisfazendo ao requisito "R2.1 – Cardinalidades entre ponto de variação e variantes"). Adicionalmente, também não é possível expressar as cardinalidades de instâncias e nem de relações entre elementos de processo, não satisfazendo aos requisitos "R2.2 – Cardinalidades de instâncias" e "R2.3 – Cardinalidades entre elementos".

Requisitos de elementos de processo: é possível representar variações em componentes de processo (Figura 37), que possuem um conceito semelhante a atividades e em atividades propriamente ditas (satisfazendo ao requisito "R3.1 – Atividade").

Entretanto, não é possível definir variações em elementos dos tipos papel, produto de trabalho, ferramenta e conexão, não sendo satisfeitos os requisitos "R3.2 – Papel", "R3.3 – Produto de trabalho", "R3.4 – Ferramenta" e "R3.5 – Conexão".

Requisitos de dependências: existem dois tipos de dependências que podem ser representadas: requerimento e exclusão (satisfazendo aos requisitos "R4.1 – Requerimento" e "R4.2 – Exclusão"), que são representadas pelas classes associativas "RequiredComponent" (requerimento) e "ConflictingComponent" (exclusão) (Figura 37). Entretanto, dependências dos tipos sugestão e substituição não podem ser representadas, não satisfazendo aos requisitos "R4.3 – Sugestão" e "R4.4 – Substituição".

4.4.6. Costache *et al.* (2011)

O trabalho apresenta uma ferramenta, denominada PDE, para a modelagem de linha de processos de software. Uma notação de modelo de características é utilizada como linguagem para a representação de variações nos processos.

Requisitos de variações: o modelo de características utilizado pelos autores para a modelagem de variações é capaz de representar características opcionais (círculo branco) e obrigatórias (círculo preto) (satisfazendo ao requisito "R1.1 – Elementos opcionais e obrigatórios"). Porém, não é possível definir quando elementos opcionais devem ser selecionados e nem o raciocínio para a seleção (não satisfazendo aos requisitos "R1.5 – *Binding time* de elementos opcionais" e "R1.8 – Raciocínio para escolha de elemento opcional"). A Figura 38 apresenta a notação gráfica do modelo de características utilizado.

Pontos de variação e variantes são representados pela relação "*alternative-child*" entre características (satisfazendo ao requisito "R1.2 – Pontos de variação e variantes").

Apesar de representar pontos de variação e variantes, não existem informações sobre ponto de variação aberto ou fechado (não satisfazendo ao requisito "R1.3 – Ponto de variação aberto ou fechado"); o momento da escolha dos variantes (não satisfazendo ao requisito "R1.4 – *Binding time* de um ponto de variação"); o variante selecionado por padrão (não satisfazendo ao requisito "R1.6 – Variante(s) selecionado(s) por padrão") e o raciocínio para a escolha de variantes (não satisfazendo ao requisito "R1.7 – Raciocínio para escolha de variante").

Os autores não mencionam a possibilidade de representar variações transversais (não satisfazendo ao requisito "R1.9 – Variações transversais").

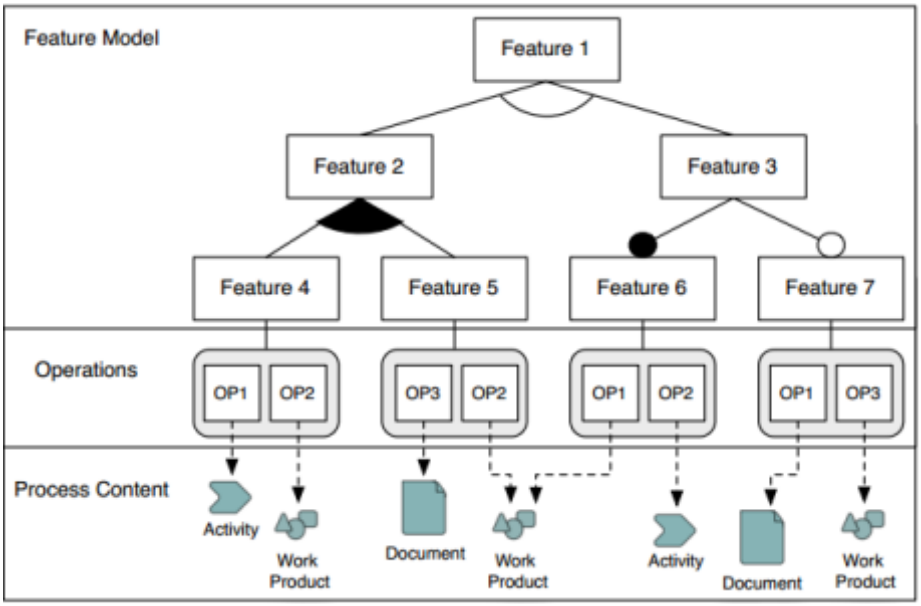


Figura 38 – Modelo de Características e Elementos de Processo (COSTACHE *et al.*, 2011).

Requisitos de cardinalidades: os autores não especificam a possibilidade de definir cardinalidades no metamodelo (não satisfazendo aos requisitos "R2.1 – Cardinalidades entre ponto de variação e variantes", "R2.2 – Cardinalidades de instâncias" e "R2.3 – Cardinalidades entre elementos").

Requisitos de elementos de processo: cada característica-folha no modelo de características está associada com um conjunto de operações para manipular diversos tipos de elementos de processo. Os autores não detalham explicitamente todos os tipos de elementos de processo que podem ser referenciados pelo modelo de características, mas, como pode ser visto na Figura 38, é possível referenciar atividades (satisfazendo ao requisito "R3.1 – Atividade"); produtos de trabalho e documentos (satisfazendo aos requisitos "R3.3 – Produto de trabalho").

Não foram encontradas informações sobre outros tipos de elementos de processo (não satisfazendo aos requisitos "R3.2 – Papel", "R3.4 – Ferramenta" e "R3.5 – Conexão").

Requisitos de dependências: o único tipo de dependência mencionado pelos autores é do tipo exclusão (satisfazendo o requisito "R4.2 – Exclusão" e não satisfazendo aos requisitos "R4.1 – Requerimento", "R4.3 – Sugestão" e "R4.4 – Substituição").

4.4.7. Golpayegani *et al.* (2013)

O trabalho apresenta uma abordagem para a modelagem de linha de processo de software para o processo de gerência de requisitos. A abordagem é composta por um método de engenharia de linha de processos de software que apresenta os passos necessários para os engenheiros definirem e instanciarem diversas linhas de processos. Para a modelagem das variações, os autores utilizaram o modelo de características da ferramenta FeatureIDE[7].

Requisitos de variações: o modelo de características utilizado suporta a modelagem de características obrigatórias e opcionais (Figura 39) (satisfazendo ao requisito "R1.1 – Elementos opcionais e obrigatórios"). Entretanto, não é possível especificar quando os elementos opcionais devem ser selecionados (não satisfazendo ao requisito "R1.5 – *Binding time* de elementos opcionais") e nem registrar o raciocínio para a escolha (não satisfazendo ao requisito "R1.8 – Raciocínio para escolha de elemento opcional").

Existem dois tipos de pontos de variação representados pela abordagem: *Encapsulated Variation Points* (EVP) e *Free Variation Points* (FVP) (satisfazendo ao requisito "R1.2 – Pontos de variação e variantes"). Porém, o trabalho não apresenta informações sobre a especificação de pontos de variação abertos e fechados, o momento em que os variantes podem ser escolhidos, os

[7] http://wwwiti.cs.uni-magdeburg.de/iti_db/research/featureide/

variantes selecionados por padrão ou o raciocínio para a escolha dos variantes (não satisfazendo aos requisitos "R1.3 – Ponto de variação aberto ou fechado", "R1.4 – *Binding time* de um ponto de variação", "R1.6 – Variante(s) selecionado(s) por padrão" e "R1.7 – Raciocínio para escolha de variante").

Variações transversais são representadas através de pontos de variação do tipo FVP, que afetam toda a linha de processos de software (satisfazendo ao requisito "R1.9 – Variações transversais").

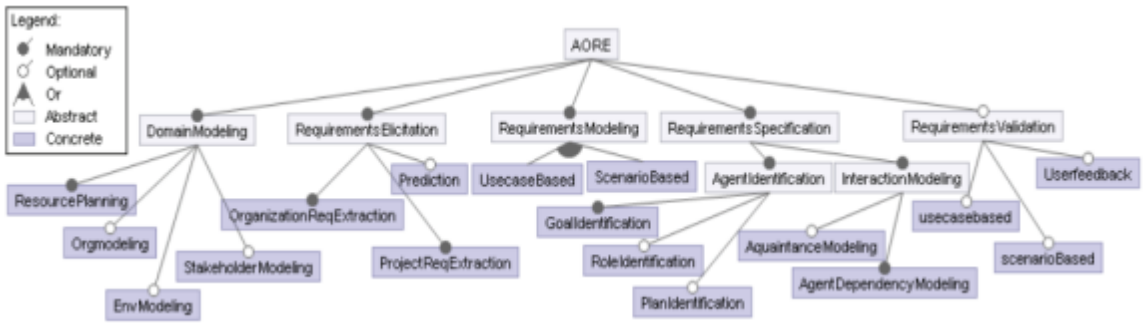


Figura 39 – Modelo de Características (GOLPAYEGANI *et al.*, 2013).

Requisitos de cardinalidades: não foram encontradas informações sobre a definição de cardinalidades (não satisfazendo aos requisitos "R2.1 – Cardinalidades entre ponto de variação e variantes", "R2.2 – Cardinalidades de instâncias" e "R2.3 – Cardinalidades entre elementos")

Requisitos de elementos de processo: o modelo de características definido pelos autores apresenta variações em tarefas e estágios (Figura 39) (satisfazendo ao requisito "R3.1 – Atividade"). Não foram encontradas informações sobre a representação de variações em outros tipos de elementos de processo (não satisfazendo aos requisitos "R3.2 – Papel", "R3.3 – "Produto de trabalho", "R3.4 – Ferramenta" e "R3.5 – Conexão").

Requisitos de dependências: uma das etapas do processo de engenharia se denomina "Análise de Similaridades e Dependências", em que são identificadas dependências do tipo requerimento entre as características de processo (satisfazendo ao requisito "R4.1 – Requerimento").

Similarmente, uma das etapas do processo de engenharia se denomina "Análise de Conflitos de Características", em que são identificados conflitos entre as características de processo (satisfazendo ao requisito "R4.2 – Exclusão").

Não foram encontradas informações sobre outros tipos de dependências (não satisfazendo aos requisitos "R4.3 – Sugestão" e "R4.4 – Substituição").

4.4.8. Hurtado *et al.* (2013)

Assim como os trabalhos anteriores do mesmo grupo de pesquisa (ALEGRÍA *et al.*, 2011; ALEGRÍA; BASTARRICA, 2012), a publicação apresenta uma linguagem, denominada Experimental SPEM (eSPEM), baseada na linguagem SPEM (OMG, 2008), para a representação de variações em linha de processos.

Requisitos de variações: a representação de elementos opcionais é possível através do modelo de características (Figura 40), satisfazendo ao requisito "R1.1 – Elementos opcionais e obrigatórios". Também é possível registrar o contexto em que um elemento opcional é adequado, através do metamodelo de contexto de processo de software (Figura 42), satisfazendo ao requisito "R1.8 – Raciocínio para escolha de elemento opcional". Esse raciocínio pode ser definido através da classe "ContextAttributeConfiguration", que relaciona um "ContextAttribute" com um dos possíveis "ContextAttributeValues".

Entretanto, não é possível especificar o momento do ciclo de vida em que os elementos opcionais devem ser selecionados (não satisfazendo ao requisito "R1.5 – *Binding time* de elementos opcionais").

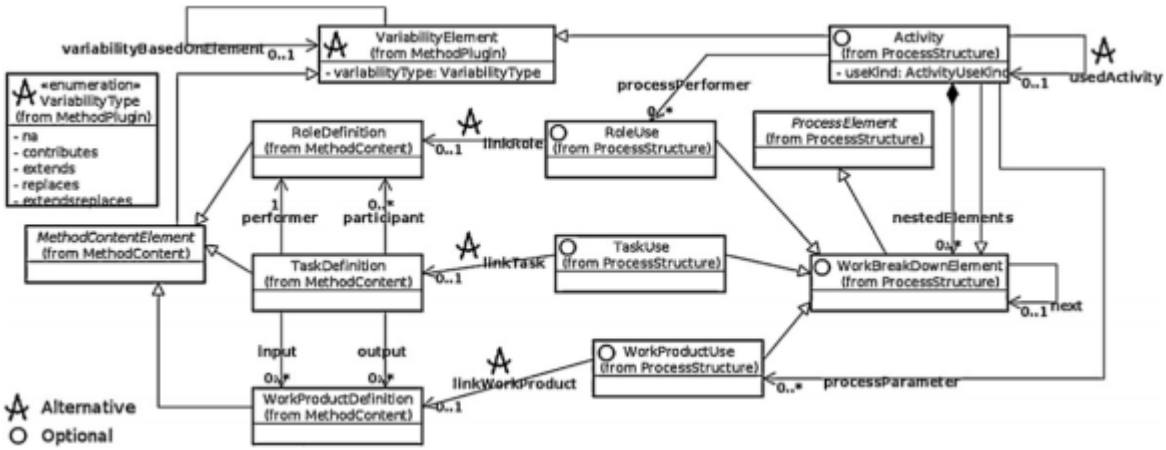


Figura 40 – Metamodelo eSPEM (HURTADO *et al.*, 2013).

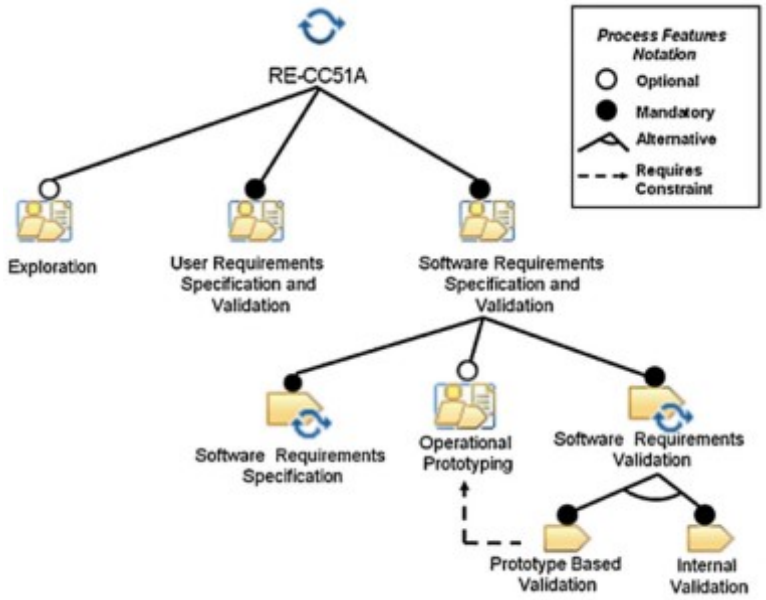


Figura 41 – Modelo de Características (HURTADO *et al.*, 2013).

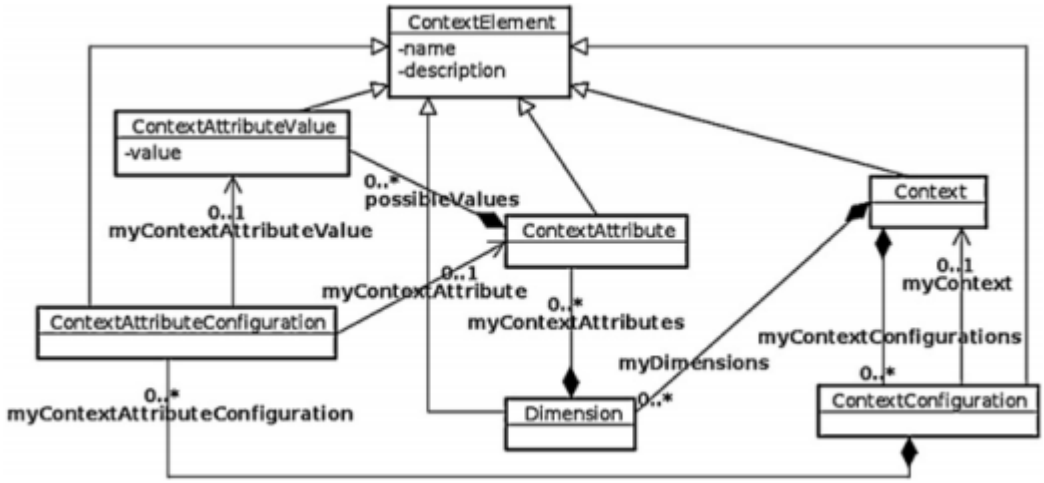


Figura 42 – Modelo de Contexto (HURTADO *et al.*, 2013).

Pontos de variação e variantes são representados através modelo de características (Figura 41), satisfazendo ao requisito "R1.2 – Pontos de variação e variantes".

Porém, não existem informações no metamodelo sobre ponto de variação aberto ou fechado (não satisfazendo ao requisito "R1.3 – Ponto de variação aberto ou fechado"), sobre o momento do ciclo de vida em que o(s) variante(s) deve(m) ser escolhido(s) (não satisfazendo ao requisito "R1.4 – *Binding time* de um ponto de variação") e nem a especificação de variantes selecionados por padrão (não satisfazendo ao requisito "R1.6 – Variante(s) selecionado(s) por padrão").

O metamodelo de configuração de processo de software define classes a partir das quais é possível registrar o raciocínio que auxilie na seleção do variante mais adequado (atendendo ao requisito "R1.7 – Raciocínio para escolha de variante").

Não é possível representar variações transversais, não sendo satisfeito o requisito "R1.9 – Variações transversais".

Requisitos de cardinalidades: segundo os autores, a abordagem utiliza o modelo de características proposto por Czarnecki *et al.* (2005), que permite expressar cardinalidades entre pontos de variação e variantes (satisfazendo ao requisito "R2.1 – Cardinalidades entre ponto de variação e variantes"). Entretanto, não são tradadas cardinalidades de instâncias e entre elementos (não satisfazendo aos requisitos "R2.2 – Cardinalidades de instâncias" e "R2.3 – Cardinalidades entre elementos").

Requisitos de elementos de processo: o metamodelo eSPEM permite expressar variações em atividades, através das classes "TaskUse", "TaskDefinition" e "Activity" (satisfazendo ao requisito "R3.1 – Atividade"); em papéis, através das classes "RoleUse" e "RoleDefinition" (satisfazendo ao requisito "R3.2 – Papel"); e produtos de trabalho, através das classes "WorkProductUse" e "WorkProductDefinition" (satisfazendo ao requisito "R3.3 – Produto de trabalho"). Entretanto, não foram encontradas informações sobre a representação de variações em ferramentas e conexões de elementos de processo (não satisfazendo aos requisitos "R3.4 – Ferramenta" e "R3.5 – Conexão"). A Figura 40 apresenta os tipos de elementos de processo com variações definidos no metamodelo.

Requisitos de dependências: no trabalho de Hurtado *et al.* (2013), foram encontradas informações apenas de dependências do tipo requerimento (satisfazendo ao requisito "R4.1 – Requerimento" e não satisfazendo aos requisitos "R4.2 – Exclusão", "R4.3 – Sugestão" e "R4.4 – Substituição").

4.4.9. Martínez-Ruiz *et al.* (2008)

Martínez-Ruiz *et al.* (2008) propõem uma extensão do metamodelo e da notação gráfica do SPEM (*Software Process Engineering Metamodel*) (OMG, 2008), versão 2.0, para acrescentar a capacidade de representação de variações em linha de processos de software. A ideia dos autores era aplicar conceitos de linhas de produtos de software para a modelagem de processos.

Requisitos de variações: a representação de elementos opcionais e obrigatórios é possível através das classes abstratas "Optional" e "Mandatory" (Figura 43) (satisfazendo ao requisito "R1.1 – Elementos opcionais e obrigatórios"), que são subclasses da classe abstrata "LProcelement", adicionada ao metamodelo pelos autores para a representação de todos os elementos relacionados às variações em linha de processo de software.

Entretanto, não existe a possibilidade de especificar o momento em que os elementos opcionais devem ser selecionados (não satisfazendo ao requisito "R1.5 – *Binding time* de

elementos opcionais") e o raciocínio para a sua escolha (não satisfazendo ao requisito "R1.8 – Raciocínio para escolha de elemento opcional").

Uma das capacidades acrescentadas foi a representação de pontos de variação e variantes, através das classes "VarPoint" e "Variant" (Figura 44), respectivamente (satisfazendo ao requisito "R1.2 – Pontos de variação e variante"). Para que um tipo de elemento seja representado como um ponto de variação, deve haver uma classe no metamodelo que seja, simultaneamente, uma subclasse de tal tipo de elemento e de da classe "VarPoint". Similarmente, para que um tipo de elemento possa ser representado como um variante, deve haver uma classe que seja uma subclasse do tipo de elemento e da classe "Variant". A Figura 45 apresenta as classes "VPActivity" e "VActivity", que representam, respectivamente, ponto de variação e variante da classe "Activity".

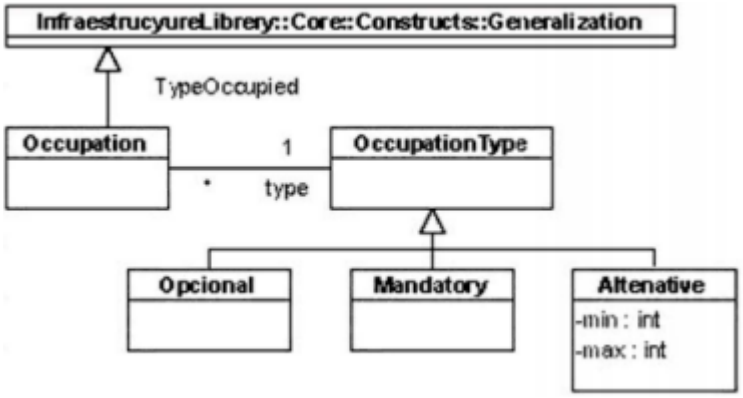


Figura 43 – Tipos de Variações (MARTÍNEZ-RUIZ *et al.* 2008).

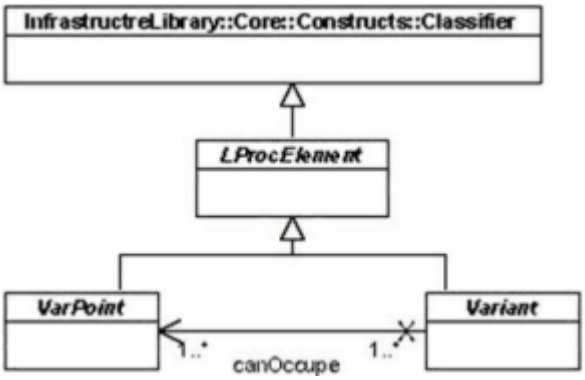


Figura 44 – Pontos de Variação e Variantes (MARTÍNEZ-RUIZ *et al.* 2008).

Apesar de representar pontos de variação e variantes, não é possível especificar: se são abertos ou fechados, o momento em que os variantes devem ser escolhidos, os variantes selecionados por padrão, nem o raciocínio para a escolha dos variantes (não satisfazendo aos requisitos "R1.3 – Ponto de variação aberto ou fechado", "R1.4 – *Binding time* de um ponto de variação", "R1.6 – Variante(s) selecionado(s) por padrão" e "R1.7 – Raciocínio para escolha de variante").

Não é possível representar variações transversais (não satisfazendo ao requisito "R1.9 – Variações transversais").

Requisitos de cardinalidades: a classe "Alternative" possui os atributos "min" e "max", permitindo expressar as cardinalidades mínima e máxima entre um ponto de variação e seus variantes (satisfazendo ao requisito "R2.1 – Cardinalidades entre ponto de variação e variantes"). Porém, não é possível representar cardinalidades de instâncias e entre elementos de processo (não satisfazendo aos requisitos "R2.2 – Cardinalidades de instâncias" e "R2.3 – Cardinalidades entre elementos").

Requisitos de elementos de processo: o metamodelo permite expressar atividades, tarefas, produtos de trabalho e papéis com variações (satisfazendo aos requisitos "R3.1 – Atividade" e "R3.2 – Papel", "R3.3 – Produto de trabalho"). Pontos de variação são representados pelas classes "VPActivity", "VPWorkProductUse", "VPRoleUse" e "VPTaskUse". Variantes são representados pelas classes "VActivity", "VWorkProductUse", "VRoleUse" e VTaskUse. A Figura 46 apresenta os tipos de elementos que em que variações podem ser representadas no metamodelo.

Porém, não é possível representar ferramentas e conexões com variações (não satisfazendo aos requisitos "R3.4 – Ferramenta" e "R3.5 – Conexão".

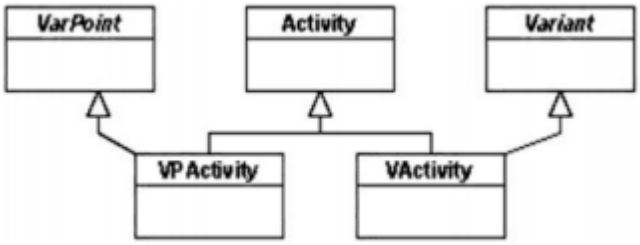


Figura 45 – Pontos de Variação e Variantes (MARTÍNEZ-RUIZ *et al.* 2008).

	Activity	WorkProductUse	RoleUse	TaskUse
Base Element	Activity	WorkProductUse	RoleUse	TaskUse
VarPoint	VPActivity	VPWorkProductUse	VPRoleUse	VPTaskUse
Variant	VActivity	VWorkProductUse	VRoleUse	VTaskUse

Figura 46 – Tipos de Pontos de Variação e Variantes (MARTÍNEZ-RUIZ *et al.* 2008).

Requisitos de dependências: dependências são introduzidas através da classe "VariabilityDependency", que podem representar relações entre dois variantes (subclasse

"VariantToVariant"), entre dois pontos de variação (subclasse "VarPointToVarPoint") ou entre um variante e um ponto de variação (subclasse "VariantToVarPoint"). Os tipos de dependências são representados pelas seguintes subclasses de "VariabilityDependencyType": "Inclusion" e "Exclusion", que representam requerimento e exclusão, satisfazendo respectivamente aos requisitos "R4.1 – Requerimento" e "R4.2 – Exclusão").

Os tipos de dependências sugestão e substituição não são cobertos pelo metamodelo (não satisfazendo aos requisitos "R4.3 – Sugestão" e "R4.4 – Substituição"). A Figura 47 apresenta as classes relacionadas às dependências.

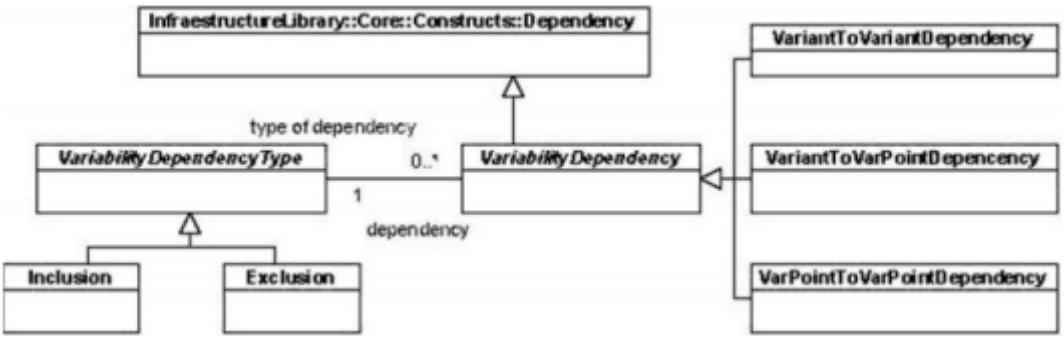


Figura 47 – Dependências e Relações (MARTÍNEZ-RUIZ *et al.* 2008).

4.4.10. Martínez-Ruiz *et al.* (2011a)

Os autores apresentam uma linguagem denominada vSPEM, que estende o metamodelo e a notação gráfica do SPEM (*Software Process Engineering Metamodel*) (OMG, 2008), versão 2.0, para a representação de conceitos de variações específicos de linha de processos de software, como pontos de variação e variantes. O foco do artigo é descrever um estudo experimental sobre a comparação entre as linguagens SPEM e vSPEM em relação à capacidade de compreensão da notação gráfica e ao poder de representar processos de software com variações.

Requisitos de variações: os autores não apresentam nenhum elemento no metamodelo ou na notação gráfica que indiquem a representação de elementos opcionais e obrigatórios (não satisfazendo ao requisito "R1.1 – Elementos opcionais e obrigatórios". Consequentemente, também não foram encontradas informações sobre a possibilidade de especificar o momento em que os elementos opcionais devem ser selecionados (não satisfazendo ao requisito "R1.5 – *Binding time* de elementos opcionais") e o raciocínio para a sua escolha (não satisfazendo ao requisito "R1.8 – Raciocínio para escolha de elemento opcional").

A representação de variações foi acrescentada através da classe abstrata "ProcLElement", que possui duas subclasses, também abstratas, "VarPoint" e "Variant", que representam, respectivamente, pontos de variação e variantes (a Figura 48 apresenta as classes no metamodelo e a Figura 49 apresenta a notação gráfica correspondente) (satisfazendo ao requisito "R1.2 –

Pontos de variação e variantes"). Um ponto de variação concreto deve herdar simultaneamente de uma classe de elemento de processo e da classe "VarPoint" e um variante concreto deve herdar simultaneamente de uma classe de elemento de processo e da classe "Variant". A Figura 48 apresenta as classes "VPActivity" e "VActivity", que representam, respectivamente, ponto de variação e variante do tipo de elemento atividade.

A classe "VarPoint" possui o atributo booleano "Open" (Figura 48). Entretanto, os autores não apresentam a semântica de tal atributo (não satisfazendo ao requisito "R1.3 – Ponto de variação aberto ou fechado"). Também não é possível determinar o momento em que os variantes devem ser selecionados, especificar os variantes selecionados por padrão e nem especificar o raciocínio para a escolha de variantes (não satisfazendo aos requisitos "R1.4 – *Binding time* de um ponto de variação", "R1.6 – Variante(s) selecionado(s) por padrão" e "R1.7 – Raciocínio para escolha de variante".

Apesar de representar pontos de variação e variantes, não é possível expressar variações transversais (não satisfazendo ao requisito "R1.9 – Variações transversais").

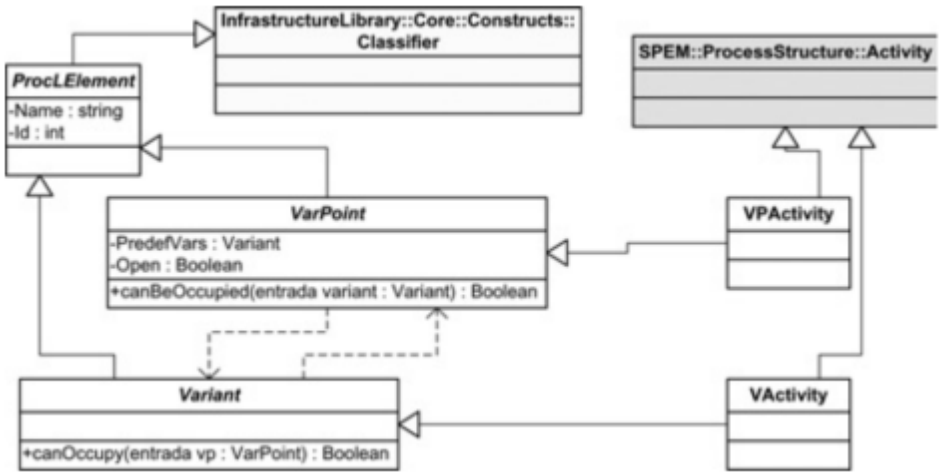


Figura 48 – Metamodelo (MARTÍNEZ-RUIZ *et al.*, 2011a).

Requisitos de cardinalidades: o trabalho não menciona nenhuma informação sobre cardinalidades. Portanto, os requisitos "R2.1 – Cardinalidades entre ponto de variação e variantes", "R2.2 – Cardinalidades de instâncias" e "R2.3 – Cardinalidades entre elementos" não são satisfeitos.

Requisitos de elementos de processo: os tipos de elementos de processo que podem ser representados com variações pelo metamodelo são: atividade (classes "VPActivity" para pontos de variação e "VActivity" para variantes) e tarefa (classes "VPTask" para pontos de variação e "VTask" para variantes), satisfazendo ao requisito "R3.1 – Atividade"; papel (classes "VPRole" para pontos de variação e "VRole" para variantes), satisfazendo ao requisito "R3.2 – Papel"; produto de trabalho (classes "VPWorkP" para pontos de variação e "VWorkP" para

variantes), satisfazendo ao requisito "R3.3 – Produto de trabalho"; e ferramenta (classes "VPTool" para pontos de variação e "VTool" para variantes), satisfazendo ao requisito "R3.3 – Produto de trabalho". Entretanto, o metamodelo não representa variações em conexões, não satisfazendo ao requisito "R3.5 – Conexão". A Figura 49 apresenta os tipos de elementos de processo em que variações podem ser representadas pelo metamodelo.

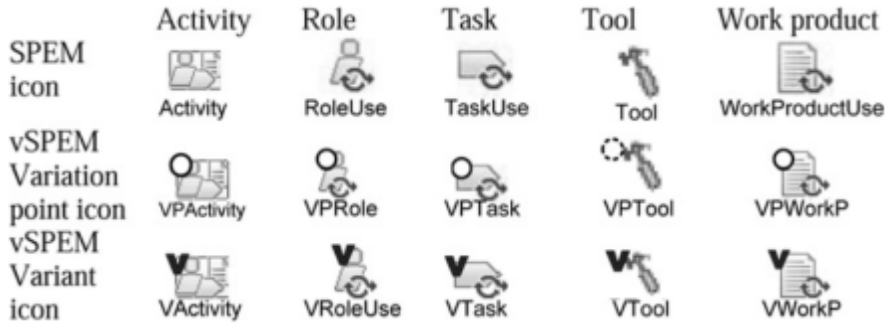


Figura 49 – Tipos de Elementos de Processo (MARTÍNEZ-RUIZ *et al.*, 2011a)

Requisitos de dependências: nenhuma informação sobre dependências é fornecida pelos autores (não satisfazendo aos requisitos "R4.1 – Requerimento", "R4.2 – Exclusão", "R4.3 – Sugestão" e "R4.4 – Substituição".

4.4.11. Martínez-Ruiz *et al.* (2011b)

O trabalho apresenta um metamodelo para a gerência de raciocínio no contexto de linha de processos de software com o objetivo de apoiar a tomada de decisão sobre a resolução de variações e para manter a rastreabilidade sobre os motivos de tais decisões. Os conceitos de gerência de raciocínio foram incluídos no metamodelo vSPEM, baseado na versão 2.0 do SPEM (OMG, 2008).

Requisitos de variações: apesar de não apresentar informações no metamodelo sobre a representação de elementos opcionais e obrigatórios, no Capítulo de estudo de aplicação, onde é apresento o uso em um processo real, os autores apresentam uma tabela com a identificação de atividades opcionais (satisfazendo ao requisito "R1.1 – Elementos opcionais e obrigatórios").

As classes "VElement" e "Variation" representam os elementos variáveis presentes na linha de processos, que podem ser pontos de variação, variantes e aspectos de processo (variações transversais) (satisfazendo ao requisito "R1.2 – Pontos de variação e variantes").

Entretanto, não é possível especificar outras informações sobre pontos de variação e variantes (não satisfazendo aos requisitos "R1.3 – Ponto de variação aberto ou fechado", "R1.4 – *Binding time* de um ponto de variação", "R1.5 – *Binding time* de elementos opcionais" e "R1.6 – Variante(s) selecionado(s) por padrão").

Um dos principais objetivos do trabalho foi acrescentar o raciocínio para a resolução de variações (satisfazendo aos requisitos "R1.7 – Raciocínio para escolha de variante" e "R1.8 – Raciocínio para escolha de elemento opcional"). Diversos elementos do metamodelo são utilizados para esse fim. A classe "Tailoring Issue" representa um problema de adaptação de processo. A classe "Alternative" representa as possíveis soluções para um problema de adaptação, onde pode-se especificar as vantagens (atributo "Advantages") e desvantagens (atributo "Disadvantages") de cada uma das alternativas em relação à resolução do problema. A classe "Arguments" representa os critérios para a decisão de quão boa cada alternativa é para a resolução do problema. A classe "Assessment" representa as avaliações de cada argumento para cada alternativa. Por fim, a classe "Resolution" representa a alternativa selecionada para a resolução do problema, mantendo a rastreabilidade sobre as discussões (atributo "discussion") e justificativa (atributo "justification") que levaram à sua seleção.

Os autores citam que, em trabalhos anteriores, foi adicionada ao metamodelo a capacidade de representar variações transversais (satisfazendo ao requisito "R1.9 – Variações transversais"). Esse tipo de variação foi elaborado seguindo o paradigma de Engenharia de Software Orientada a Aspectos (*Aspect Oriented Software Engineering* – AOSE) e é representado através das classes "VElement" e "Variation" (Figura 50). Entretanto, os autores não apresentam em detalhes tais conceitos.

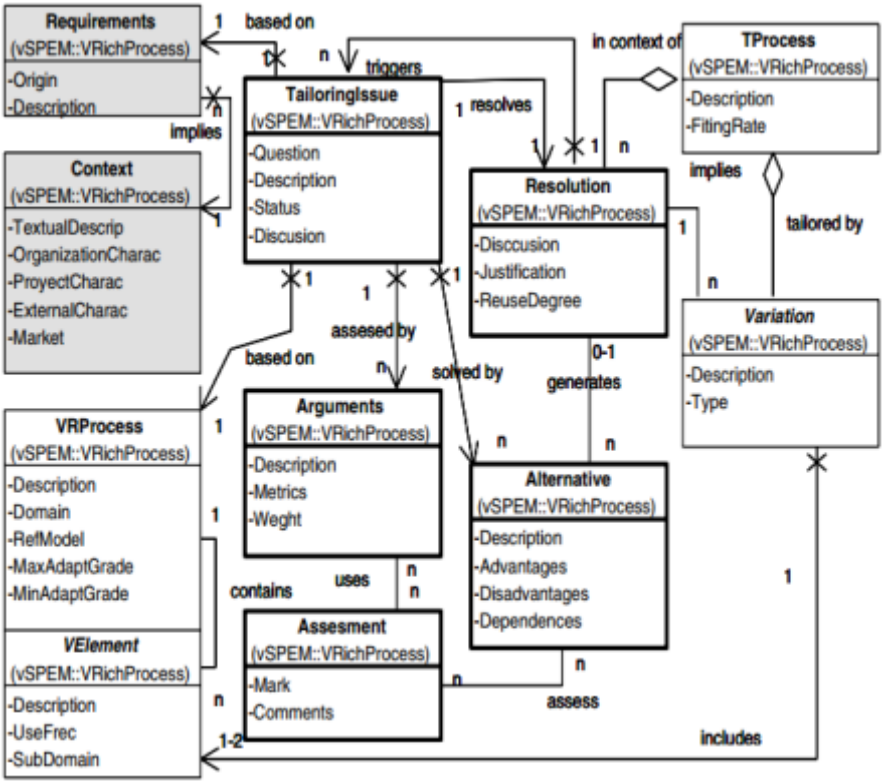


Figura 50 – Metamodelo (MARTÍNEZ-RUIZ *et al.*, 2011b).

Requisitos de cardinalidades: não foram apresentadas informações sobre cardinalidades entre pontos de variação e variantes (não satisfazendo ao requisito "R2.1 – Cardinalidades entre ponto de variação e variantes").

A classe "VRProcess" possui os atributos "MaxAdaptGrade" e "MinAdaptGrade" (Figura 50). O primeiro corresponde ao número máximo de pontos de variação somado com os aspectos de processo (variações transversais) que podem ser usados na linha de processos. O segundo corresponde ao número obrigatório de pontos de variação incluídos na linha de processos (satisfazendo ao requisito "R2.2 – Cardinalidades de instâncias").

Entretanto, não é possível especificar a cardinalidade entre elementos (não satisfazendo ao requisito "R2.3 – Cardinalidades entre elementos").

Requisitos de elementos de processo: apesar de não apresentar informações no metamodelo sobre os tipos de elementos de processo em que variações podem ocorrer, no Capítulo de estudo de aplicação, onde é apresento o uso em um processo real, os autores apresentam uma tabela com os seguintes elementos de processos representados com variações: papel, produto de trabalho e atividade (satisfazendo aos requisitos "R3.1 – Atividade", "R3.2 – Papel" e "R3.3 – Produto de trabalho").

Entretanto, não são apresentadas informações sobre a representação de outros tipos de elementos de processo (não satisfazendo aos requisitos "R3.4 – Ferramenta" e "R3.5 – Conexão").

Requisitos de dependências: é possível representar dependências do tipo inclusão através do atributo "Dependences", da classe "Alternative" (Figura 50) (satisfazendo ao requisito "R4.1 – Requerimento").

Não foram encontradas informações sobre outros tipos de dependências (não satisfazendo aos requisitos "R4.2 – Exclusão", "R4.3 – Sugestão" e "R4.4 – Substituição").

4.4.12. Martínez-Ruiz *et al.* (2011c)

Os autores apresentam uma linguagem denominada vSPEM, que estende o metamodelo e a notação gráfica do SPEM (*Software Process Engineering Metamodel*) (OMG, 2008), versão 2.0, para a representação de conceitos de variações específicos de linha de processos de software. O foco do artigo é descrever os tipos de variações, especialmente variações transversais, ou aspectos de processo, utilizando conceitos de Engenharia de Software Orientada a Aspectos (Aspect Oriented Software Engineering – AOSE).

Requisitos de variações: não foram apresentadas informações sobre a representação de elementos opcionais e obrigatórios (não satisfazendo ao requisito "R1.1 – Elementos opcionais e

obrigatórios"). Consequentemente, os requisitos "R1.5 – *Binding time* de elementos opcionais" e "R1.8 – Raciocínio para escolha de elemento opcional" também não são satisfeitos.

O metamodelo classifica as variações em pontuais (*on-point variations*) e transversais (*crosscutting variations*). Variações pontuais são representadas por pontos de variação, classe "VarPoint", e variantes, classe "Variant" (Figura 51) (satisfazendo ao requisito "R1.2 – Pontos de variação e variantes"). O atributo booleano "isImplicit", da classe "VarPoint", diferencia pontos de variação convencionais (valor "falso") de variações transversais (valor "verdadeiro").

Conceitos de orientação a aspectos para representar variações transversais são implementados pelas classes "ProcessPointCut", "ProcessAspect" e "ProcessAdvice" (Figura 51) (satisfazendo ao requisito "R1.9 – Variações transversais").

Entretanto, apesar de representar pontos de variação e variantes, não é possível expressar que pontos de variação são abertos ou fechados (não satisfazendo ao requisito "R1.3 – Ponto de variação aberto ou fechado"), o momento do ciclo de vida em que os variantes devem ser escolhidos (não satisfazendo ao requisito "R1.4 – *Binding time* de um ponto de variação"), especificar variantes selecionados por padrão (não satisfazendo ao requisito "R1.6 – Variante(s) selecionado(s) por padrão") e nem registrar o raciocínio para a escolha dos variantes mais adequados (não satisfazendo ao requisito "R1.7 – Raciocínio para escolha de variante").

Requisitos de cardinalidades: não foram encontradas informações sobre cardinalidades (não satisfazendo aos requisitos "R2.1 – Cardinalidades entre ponto de variação e variantes", "R2.2 – Cardinalidades de instâncias" e "R2.3 – Cardinalidades entre elementos").

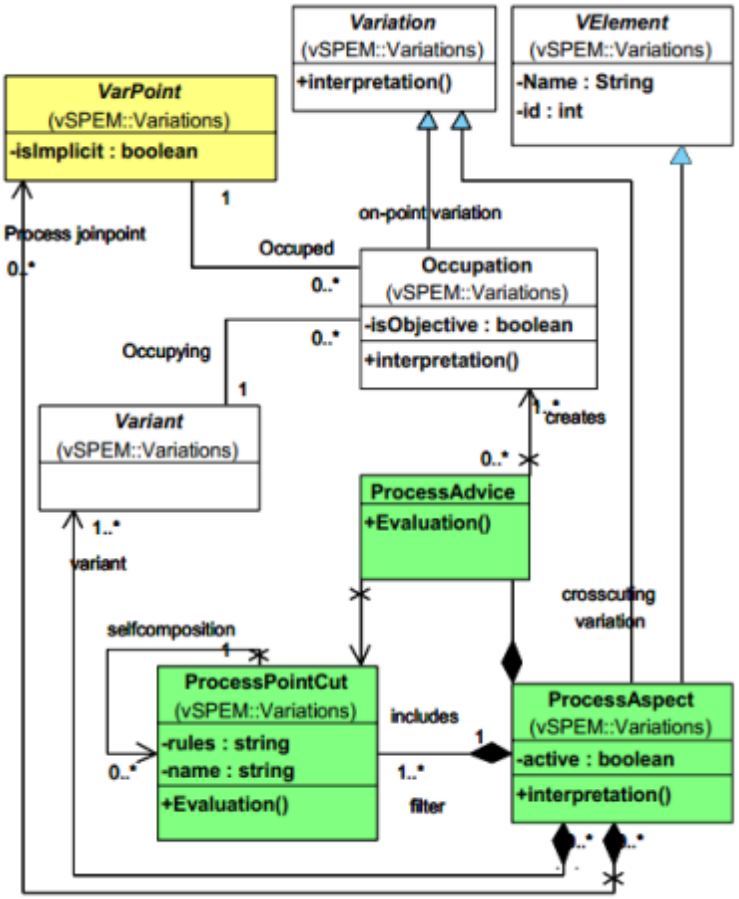


Figura 51 – Metamodelo (MARTÍNEZ-RUIZ *et al.*, 2011c).

Requisitos de elementos de processo: apesar de não apresentar diretamente construtores no metamodelo, os autores afirmam que é possível representar variações em atividades, tarefas e produto de trabalhos (satisfazendo aos requisitos "R3.1 – Atividade", "R3.2 – Papel" e "R3.3 – Produto de trabalho").

Os requisitos "R3.4 – Ferramenta" e "R3.5 – Conexão" não são satisfeitos.

Requisitos de dependências: apesar de não apresentar classes no metamodelo, os autores afirma que é possível definir dependências do tipo inclusão entre dois variantes ("variant2variant") (satisfazendo ao requisito "R4.1 – Requerimento").

Entretanto, não foram encontradas informações sobre outros tipos de dependências (não satisfazendo aos requisitos "R4.2 – Exclusão", "R4.3 – Sugestão" e "R4.4 – Substituição").

4.4.13. Martínez-Ruiz *et al.* (2013a)

O trabalho relata a utilização de conceitos de linha de processos de software para auxiliar na adaptação de processos no contexto de desenvolvimento global de software (*Global Software Development* - GSD). Para a modelagem de processos de software, foi utilizada a notação vSPEM,

uma modificação do metamodelo e da notação gráfica da linguagem SPEM (OMG, 2008), versão 2.0.

Requisitos de variações: a notação gráfica do vSPEM é capaz de representar elementos opcionais. Na seção que apresenta o processo modelado para o desenvolvimento global de software, foram apresentados dois elementos opcionais: "Retrospective of Retrospectives" e "Scrum of Scrums" (satisfazendo ao requisito "R1.1 – Elementos opcionais e obrigatórios").

Entretanto, não foram encontradas informações sobre o momento em que elementos opcionais devem ser selecionados e nem o raciocínio para tal escolha (não satisfazendo aos requisitos "R1.5 – *Binding time* de elementos opcionais" e "R1.8 – Raciocínio para escolha de elemento opcional").

Na Figura 52, os elementos "2. Variation Point DS1" e "4. Variation Point DS2" são pontos de variação e os elementos "V1. Analysis and Design" e "V4. Test and Integration" são variantes (satisfazendo ao requisito "R1.2 – Pontos de variação e variantes"). Além disso, Figura 53 apresenta o formulário de cadastro de ponto de variação.

O formulário de cadastro de ponto de variação (Figura 53) apresenta o campo booleano "Is Open". Entretanto, os autores não apresentam a semântica de tal campo (não satisfazendo ao requisito "R1.3 – Ponto de variação aberto ou fechado").

Complementarmente, também não foram encontradas informações sobre a possibilidade de expressar que pontos de variação são abertos ou fechados o momento do ciclo de vida em que os variantes devem ser escolhidos (não satisfazendo ao requisito "R1.4 – *Binding time* de um ponto de variação"), especificar variantes selecionados por padrão (não satisfazendo ao requisito "R1.6 – Variante(s) selecionado(s) por padrão") e nem registrar o raciocínio para a escolha dos variantes mais adequados (não satisfazendo ao requisito "R1.7 – Raciocínio para escolha de variante").

Existe a possibilidade de representar variações transversais através do uso de conceitos de *Aspect Oriented Software Engineeriong* (AOSE) (satisfazendo ao requisito "R1.9 – Variações transversais").

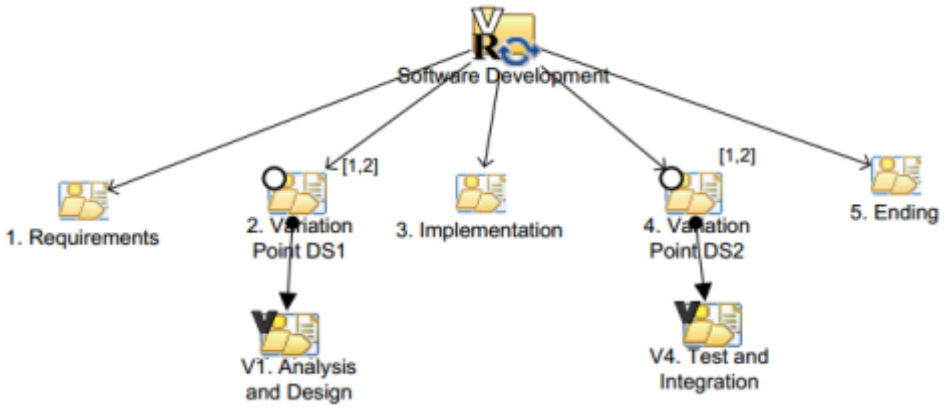


Figura 52 – Notação Gráfica (MARTÍNEZ-RUIZ *et al.*, 2013a).

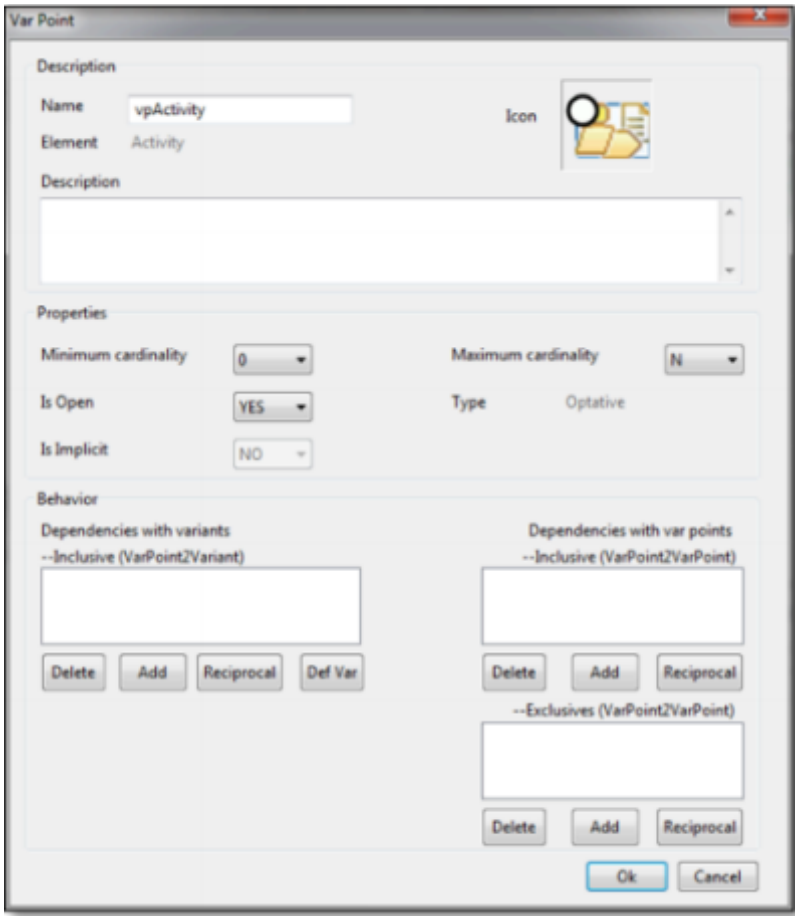


Figura 53 – Formulário de Cadastro de Ponto de Variação (MARTÍNEZ-RUIZ *et al.*, 2013a).

Requisitos de cardinalidades: existe a possibilidade de especificar a cardinalidade mínima e a cardinalidade máxima entre pontos de variação e variantes, campos "Minimum cardinality" e "Maximum cardinality" no formulário de cadastro de ponto de variação (Figura 53) (satisfazendo ao requisito "R2.1 – Cardinalidades entre ponto de variação e variantes").

Entretanto, não foram encontradas informações sobre cardinalidades de instâncias e entre elementos de processo (não satisfazendo aos requisitos "R2.2 – Cardinalidades de instâncias" e "R2.3 – Cardinalidades entre elementos").

Requisitos de elementos de processo: é possível representar variações em atividades, tarefas, produto de trabalhos e papéis, como mostra o formulário de adaptação de processos (Figura 54) (satisfazendo aos requisitos "R3.1 – Atividade", "R3.2 – Papel" e "R3.3 – Produto de trabalho").

Não foram encontradas informações sobre a representação de variações em ferramentas e conexões (não satisfazendo aos requisitos "R3.4 – Ferramenta" e "R3.5 – Conexão").

Vale ressaltar que, apesar de não ter sido considerado um requisito de comparação, também é possível representar variações em fases, iterações e áreas de processo.

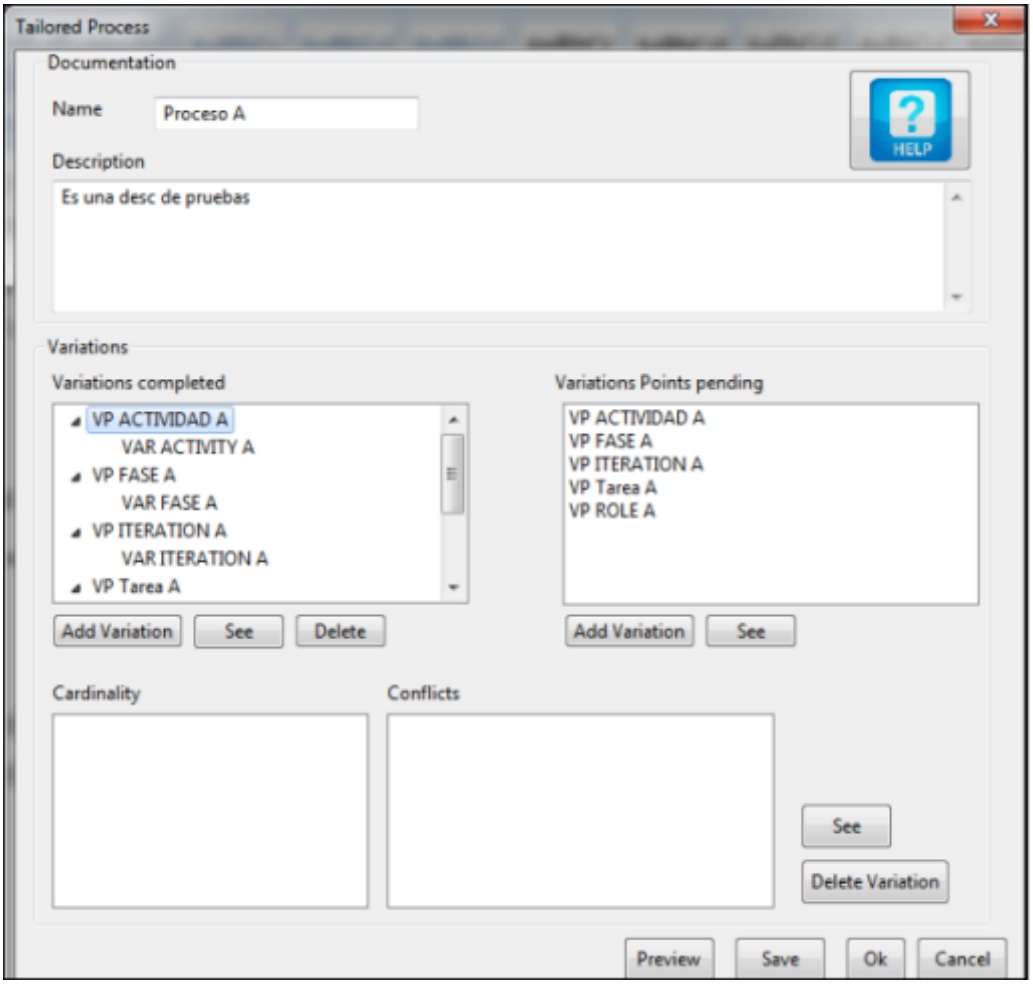


Figura 54 – Formulário de Adaptação de Processos (MARTÍNEZ-RUIZ *et al.*, 2013a).

Requisitos de dependências: Conforme o formulário de cadastro de ponto de variação (Figura 53) é possível definir dependências do tipo requerimento, entre pontos de variação e variantes ou entre pontos de variação, e exclusão, entre pontos de variação (satisfazendo aos requisitos "R4.1 – Requerimento" e "R4.2 – Exclusão").

Entretanto, não foram encontradas informações sobre outros tipos de dependências (não satisfazendo aos requisitos "R4.3 – Sugestão" e "R4.4 – Substituição").

4.4.14. Rouillé *et al.* (2012)

O trabalho apresenta a proposta de utilização da linguagem SPEM 2.0 para a modelagem de processos de software e a linguagem CVL (*Common Variability Language*) (OMG, 2012) para a modelagem de variações em linha de processos de software. A ideia dos autores é promover o desacoplamento entre as linguagens de modelagem de processo de software e mecanismos de variações, com a alegação de que representar variações no próprio metamodelo da linguagem de modelagem de processos dificulta a legibilidade e limita o poder de reutilização dos mecanismos de variações para outros contextos.

Requisitos de variações: elementos opcionais são representados no metamodelo CVL através da classe "ObjectExistence" (Figura 56), um tipo de ponto de variação que indica a possibilidade de incluir ou não um elemento de processo. Além disso, o atributo booleano "isImpliedByParent" da classe "Choice", indica elementos obrigatórios (satisfazendo ao requisito "R1.1 – Elementos opcionais e obrigatórios"). Porém, não é possível especificar o momento do ciclo de vida em que elementos opcionais devem ser escolhidos e nem registrar o raciocínio para auxiliar na escolha de elementos opcionais (não satisfazendo aos requisitos "R1.5 – *Binding time* de elementos opcionais" e "R1.8 – Raciocínio para escolha de elemento opcional").

Pontos de variação são representados pela classe "VariationPoint" e suas subclasses ("ChoiceVariationPoint", "LinkAssignment", "ObjectSubstitution", "Existence" e "ObjectExistence") (satisfazendo ao requisito "R1.2 – Pontos de variação e variantes"). A Figura 56 apresenta os tipos de ponto de variação.

Entretanto, apesar de representar pontos de variação, o metamodelo CVL não possibilita definir se um ponto de variação é aberto ou fechado (não satisfazendo ao requisito "R1.3 – Ponto de variação aberto ou fechado"); o momento em que os variantes devem ser selecionados (não satisfazendo ao requisito "R1.4 – *Binding time* de um ponto de variação"); os variantes selecionados por padrão (não satisfazendo ao requisito "R1.6 – Variante(s) selecionado(s) por padrão"); e nem registrar o raciocínio para a auxiliar na escolha dos variantes (não satisfazendo ao requisito "R1.7 – Raciocínio para escolha de variante").

Variações transversais não são tratadas no metamodelo CVL (não satisfazendo ao requisito "R1.9 – Variações transversais").

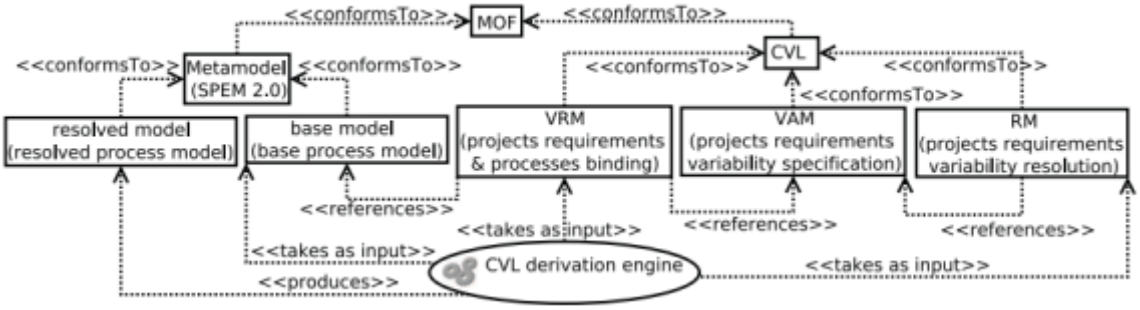


Figura 55 – CVL para Linha de Processos de Software (ROUILLÉ *et al.*, 2012).

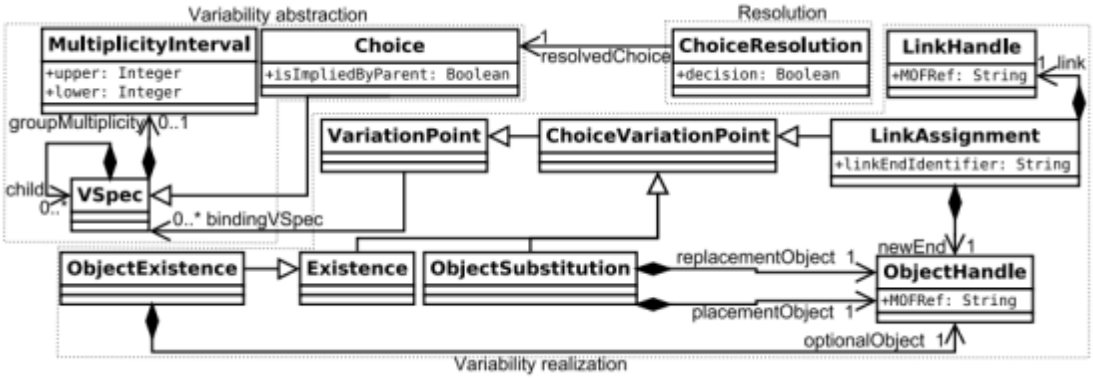


Figura 56 – Metamodelo CVL (ROUILLÉ *et al.*, 2012).

Requisitos de cardinalidades: a classe "MultiplicityInterval" possui os atributos "upper" e "lower", que significam, respectivamente, o número mínimo e máximo de filhos que um ponto de variação pode ter (satisfazendo ao requisito "R2.1 – Cardinalidades entre ponto de variação e variantes").

Porém, não é possível definir cardinalidades de instâncias e entre elementos (não satisfazendo aos requisitos "R2.2 – Cardinalidades de instâncias" e "R2.3 – Cardinalidades entre elementos").

Requisitos de elementos de processo: segundo os autores, a linguagem CVL é capaz de representar variações em qualquer modelo de processo de software, independente da linguagem de modelagem de processo utilizada. A associação entre as variações e os elementos de processo é representada através das classes "ObjectHandle" e "LinkHandle" (Figura 56), que possuem o atributo texto "MOFRef", capazes referenciar qualquer elemento de processo presente no metamodelo SPEM 2.0 (Figura 57) (satisfazendo aos requisito "R3.1 – Atividade", "R3.2 – Papel", "R3.3 – Produto de trabalho", "R3.4 – Ferramenta" e "R3.5 – Conexão"). Apesar de tal flexibilidade, os autores alertam para algumas questões como a dificuldade de garantir a validade dos modelos gerados.

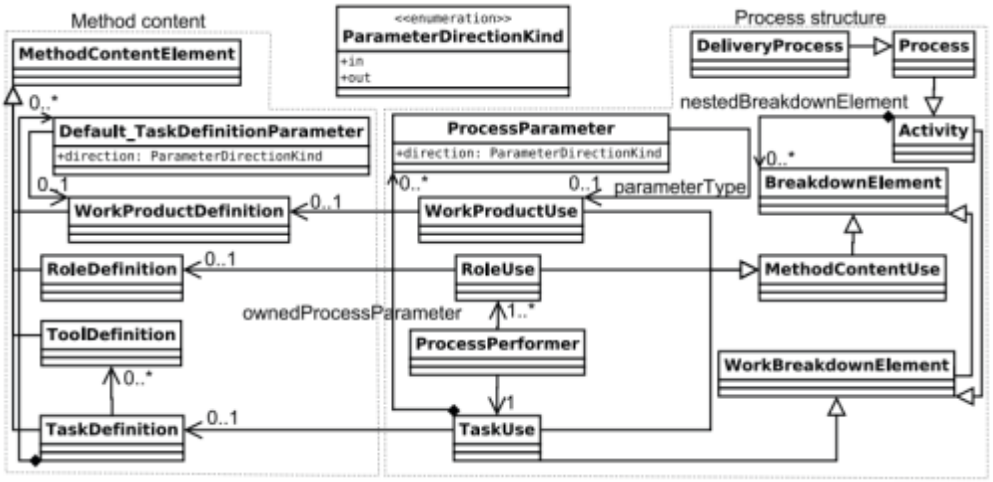


Figura 57 – Metamodelo SPEM 2.0 (ROUILLÉ *et al.*, 2012).

Requisitos de dependências: é possível representar dependências do tipo requerimento através do atributo "isImpliedByParent" da classe "Choice", que requer a seleção de um variante se o ponto de variação for selecionado (satisfazendo ao requisito "R4.1 – Requerimento").

Não existem informações sobre outros tipos de dependências (não satisfazendo aos requisitos "R4.2 – Exclusão", "R4.3 – Sugestão" e "R4.4 – Substituição").

4.4.15. Teixeira (2011)

Na dissertação de mestrado de Teixeira (2011), é proposto um metamodelo e uma notação gráfica para a representação de variabilidades em linha de processos de software.

Requisitos de variações: o metamodelo representa elementos opcionais e obrigatórios através do atributo booleano "ehMandatoria" da classe "Característica" (Figura 58), satisfazendo ao requisito "R1.1 – Elementos opcionais e obrigatórios". Porém, não é possível representar o momento do ciclo de vida em que as características opcionais devem ser selecionadas (não satisfazendo ao requisito "R1.5 – *Binding time* de elementos opcionais") e nem registrar a situação em que sua escolha é recomendável (não satisfazendo ao requisito "R1.8 – Raciocínio para escolha de elemento opcional").

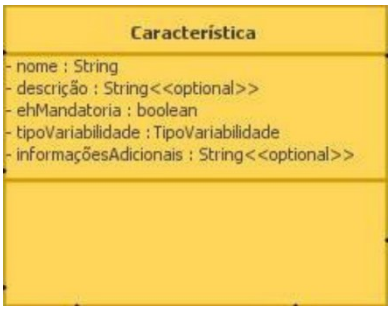


Figura 58 – Classe "Característica" (TEIXEIRA, 2011).

A classe "Alternativo" (Figura 59) representa um ponto de variação e seus variantes (satisfazendo ao requisito "R1.2 - Pontos de variação e variantes"). As possíveis combinações entre os tipos de características referenciadas como pontos de variação e variantes estão especificadas na Tabela 11.

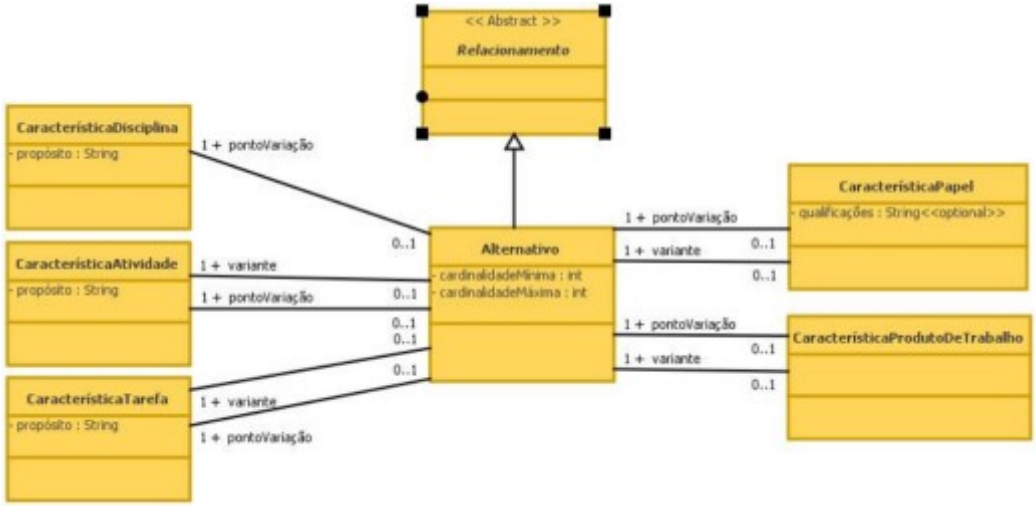


Figura 59 – Classe "Alternativo" (Ponto de Variação e Variantes) (TEIXEIRA, 2011).

Tabela 11 – Restrições de Combinações de Pontos de Variação (TEIXEIRA, 2011).

Tipo de Relacionamento	Categoria de Característica Origem: Ponto de Variação	Categoria de Característica Destino: Variante
Alternativo	CaracterísticaDisciplina	CaracterísticaAtividade
	CaracterísticaAtividade	CaracterísticaAtividade
	CaracterísticaAtividade	CaracterísticaTarefa
	CaracterísticaTarefa	CaracterísticaTarefa
	CaracterísticaPapel	CaracterísticaPapel
	CaracterísticaProdutoDeTrabalho	CaracterísticaProdutoDeTrabalho

Entretanto, o metamodelo não permite expressar: se o ponto de variação é aberto ou fechado (não satisfazendo ao requisito "R1.3 – Ponto de variação aberto ou fechado"); em que momento do ciclo de vida a escolha do(s) variante(s) deve ser realizada (não satisfazendo ao requisito "R1.4 – *Binding time* de um ponto de variação"); o(s) variante(s) selecionados por padrão (não satisfazendo ao requisito "R1.6 – Variante(s) selecionado(s) por padrão"); e nem registrar informações que auxiliem a decidir em que contexto um determinado variante é mais adequado (não satisfazendo ao requisito "R1.7 – Raciocínio para escolha de variante").

Apesar de representar características opcionais, obrigatórias e pontos de variação, não é possível expressar variações transversais (não satisfazendo ao requisito "R1.9 – Variações transversais").

Requisitos de cardinalidades: A classe "Alternativo" possui os atributos "cardinalidadeMínima" e "cardinalidadeMáxima" (satisfazendo ao requisito "R2.1 - Cardinalidades entre ponto de variação e variantes"), que determinam o número mínimo e máximo, respectivamente, de variantes que podem ser selecionados em um ponto de variação.

Contudo, os requisitos "R2.2 – Cardinalidades de instâncias" e "R2.3 – Cardinalidades entre elementos" não são satisfeitos.

Requisitos de elementos de processo: os tipos de elementos de processo que podem ser representados com variações pelo metamodelo são: disciplina (classe "CaracterísticaDisciplina"), atividade (classe "CaracterísticaAtividade"), tarefa (classe "CaracterísticaTarefa"), satisfazendo ao requisito "R3.1 – Atividade"; papel (classe "CaracterísticaPapel"), satisfazendo ao requisito "R3.2 – Papel"; e produto de trabalho (classe "CaracterísticaProdutoDeTrabalho"), satisfazendo ao requisito "R3.3 – Produto de trabalho".

O requisito "R3.4 – Ferramenta" não é satisfeito. Entretanto, o metamodelo também representa variações em tipos de elementos não cobertos pelos requisitos, como prática (classe "CaracterísticaPrática") e conceito (classe "CaracterísticaConceito"). A Figura 60 apresenta os tipos de elementos de processo em que variações podem ser representadas pelo metamodelo.

A classe "Relacionamento" (Figura 61) possui subclasses que representam relações entre diferentes tipos de elementos de processo ("LigaçãoPapelProdutoDeTrabalho", "LigaçãoPapelTarefa" e "LigaçãoProdutoDeTrabalhoTarefa"). Cada um desses relacionamentos possui o atributo "ehOpcional", satisfazendo ao requisito "R3.5 – Conexão".

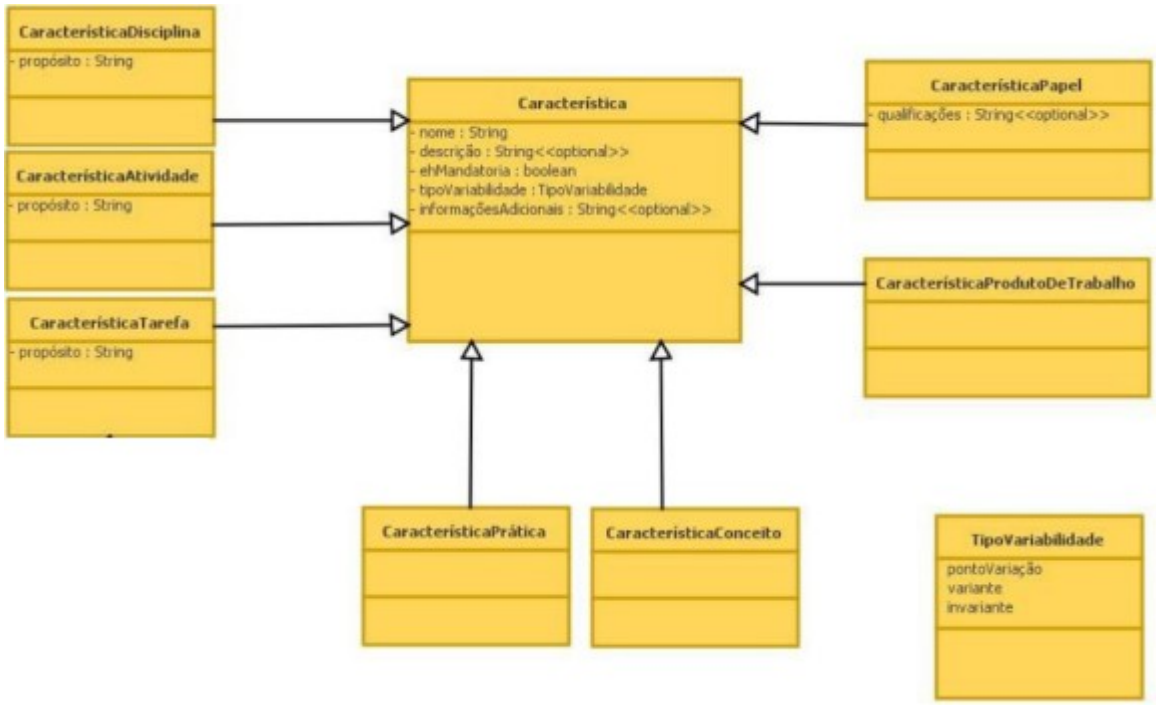


Figura 60 – Tipos de Elementos de Processo (TEIXEIRA, 2011).

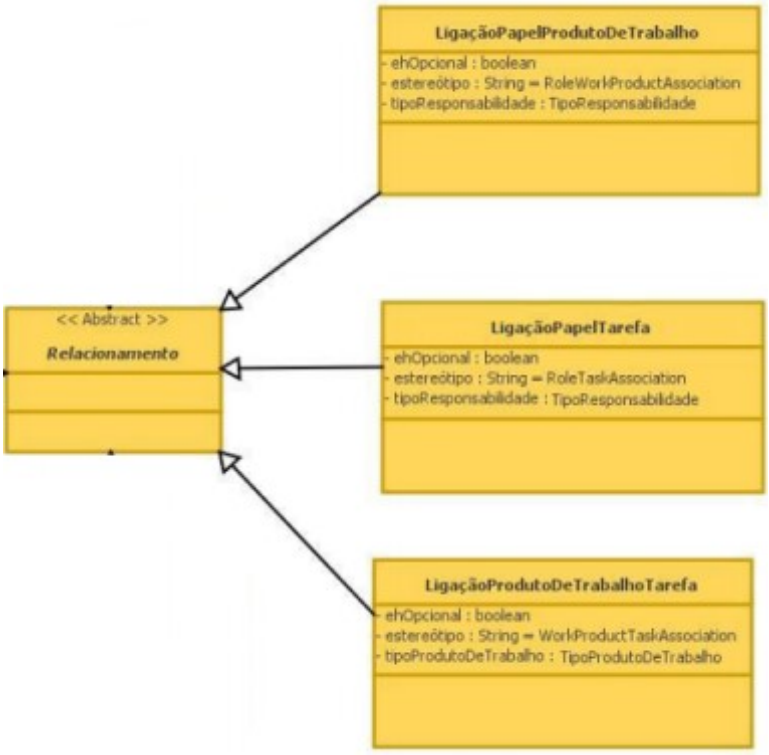


Figura 61 – Relacionamentos entre Elementos de Processo (TEIXEIRA, 2011).

Requisitos de dependências: quanto aos tipos de dependências, é possível expressar: requerimento, através da classe "RegraComposiçãoInclusiva", satisfazendo ao requisito "R4.1 – Requerimento"; e exclusão, através da classe "RegraComposiçãoExclusiva", satisfazendo ao requisito "R4.2 – Exclusão". Entretanto, os requisitos "R4.3 – Sugestão" e "R4.4 – Substituição" não são satisfeitos. A Figura 62 apresenta as classes que formam as regras de composição do metamodelo, sendo possível expressar dependências entre as características.

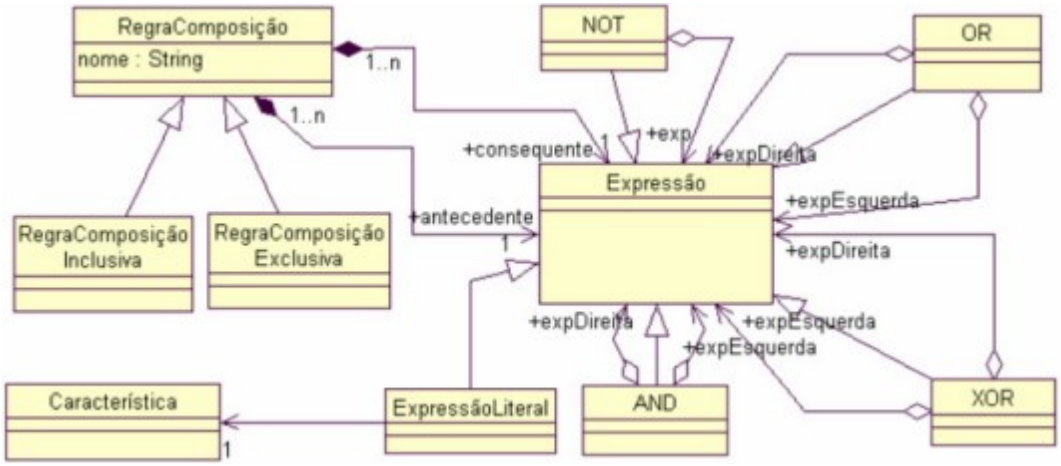


Figura 62 – Regras de Composição de Dependências (TEIXEIRA, 2011).

4.4.16. Ternité (2009)

O trabalho apresenta um metamodelo, baseado no V-Modell XT 1.3, para a modelagem de linha de processos de software. O foco da pesquisa é manter a rastreabilidade das modificações realizadas nos modelos de processo.

Requisitos de variações: o metamodelo proposto por Ternité (2009) representa características opcionais e obrigatórias, através das classes "OptionalFeature" e "MandatoryFeature" (Figura 63), respectivamente, satisfazendo ao requisito "R1.1 – Elementos opcionais e obrigatórios". Porém, não é possível determinar o momento do ciclo de vida em que as características opcionais devem ser selecionadas (não satisfazendo ao requisito "R1.5 – *Binding time* de elementos opcionais") e nem registrar o raciocínio para auxiliar na sua seleção (não satisfazendo ao requisito "R1.8 – Raciocínio para escolha de elemento opcional").

Apesar de representar características opcionais, obrigatórias e pontos de variação, não é possível expressar variações transversais (não satisfazendo ao requisito "R1.9 – Variações transversais").

Pontos de variação e variantes são representados pelo relacionamento "replaces" entre a classe abstrata "Feature" e a classe "AlternativeFeature" (Figura 63), atendendo ao requisito "R1.2 – Pontos de variação e variantes". Apesar de representar ponto de variação, não é possível expressar que é aberto ou fechado, o momento do ciclo de vida em que os variantes devem ser selecionados e nem especificar o(s) variante(s) selecionados por padrão (não sendo satisfeitos os requisitos "R1.3 – Ponto de variação aberto ou fechado", "R1.4 – *Binding time* de um ponto de variação" e "R1.6 – Variante(s) selecionado(s) por padrão").

Não é possível registrar o raciocínio para auxiliar na tomada de decisão sobre que variante(s) deve(m) ser selecionado(s), não satisfazendo ao requisito "R1.7 – Raciocínio para escolha de variante".

Requisitos de cardinalidades: apesar de representar pontos de variação e variantes, não existem mecanismos para especificar a cardinalidade mínima e máxima entre esses elementos, o que não satisfaz ao requisito "R2.1 – Cardinalidades entre ponto de variação e variantes". Também não é possível representar cardinalidades de instâncias e entre elementos de processo. Portanto, os requisitos "R2.2 – Cardinalidades de instâncias" e "R2.3 – Cardinalidades entre elementos" não são satisfeitos.

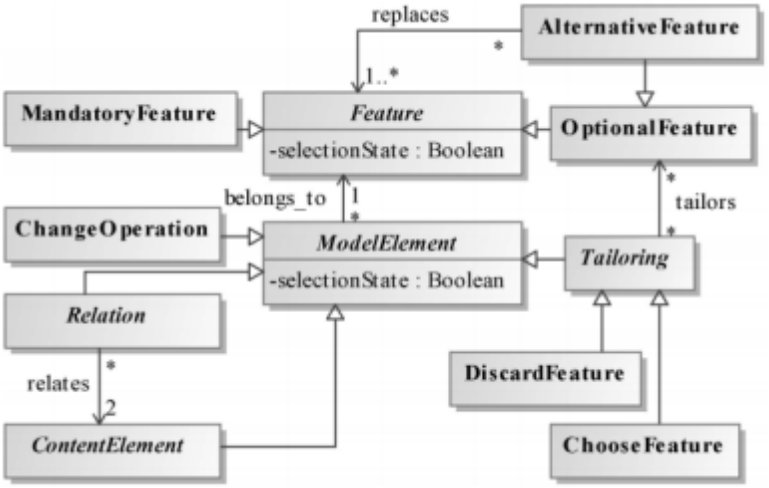


Figura 63 – Metamodelo (TERNITÉ, 2009).

Requisitos de elementos de processo: segundo os autores, elementos de processo são representados como subclasses da classe abstrata "ContentElement" e cabe às organizações definirem os elementos que atendam às suas necessidades específicas. Porém, os próprios autores definiram, com o propósito de exemplificar a aplicação do metamodelo, os elementos papel e produto de trabalho, através das classes "Role" e "WorkProduct", respectivamente, satisfazendo aos requisitos "R3.2 – Papel" e "R3.3 – Produto de trabalho". Já os requisitos "R3.1 – Atividade" e "R3.4 – Ferramenta" não foram satisfeitos, pois não foram definidas classes equivalentes a esses conceitos. A Figura 64 apresenta os tipos de elementos de processo definidos no metamodelo.

Como pode ser verificado na Figura 63, o metamodelo define a classe "Relation" como uma subclasse de "ModelElement", permitindo, assim, expressar relações entre elementos com variações. A Figura 64 apresenta um exemplo através da classe "accounts_for", que representa uma relação entre "Role" e "WorkProduct", satisfazendo ao requisito "R3.5 – Conexão".

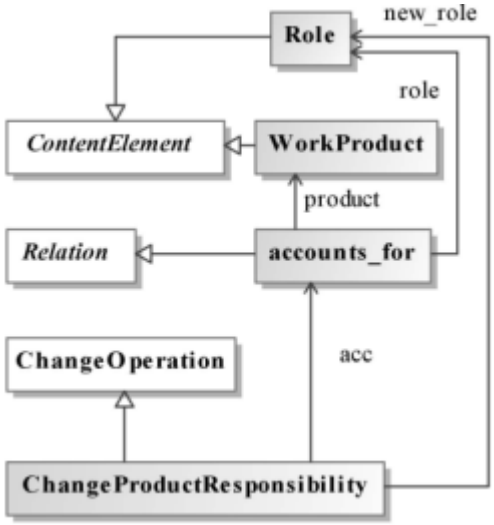


Figura 64 – Instâncias de "ContetElement" e "Relation" (TERNITÉ, 2009).

Requisitos de dependências: nenhum tipo de dependência foi definido no metamodelo, não satisfazendo aos requisitos "R4.1 – Requerimento", "R4.2 – Exclusão", "R4.3 – Sugestão" e "R4.4 – Substituição".

4.4.17. Washizaki (2006a)

O trabalho apresenta um *framework* em que é possível representar similaridades e variações em linha de processos. Apesar de não definir diretamente um metamodelo, o autor define conceitos e elaborou uma notação gráfica a partir da qual é possível inferir elementos de um metamodelo para a representação das variações. A Figura 65 apresenta uma visão geral da abordagem de Washizaki (2006a).

Requisitos de variações: as variações podem ser representadas por elementos opcionais e pontos de variação. Elementos opcionais são identificados através do estereótipo <<optional>> (satisfazendo ao requisito "R1.1 – Elementos opcionais e obrigatórios"), como pode ser visto na Figura 66. Mas o momento de escolha de um elemento opcional não é tratado no trabalho (não satisfazendo ao requisito "R1.5 – *Binding time* de elementos opcionais"), tampouco é possível definir o raciocínio para a seleção ou não de elementos opcionais (não satisfazendo ao requisito "R1.8 – Raciocínio para escolha de elemento opcional").

Pontos de variação e variantes são representados pelos estereótipos <<variationPoint>> e <<variant>>, respectivamente (satisfazendo ao requisito "R1.2 – Pontos de variação e variantes") (Figura 66). Apesar de representar pontos de variação e variantes, não foram encontradas informações sobre: a representação de pontos de variação abertos ou fechados (não satisfazendo ao requisito "R1.3 – Ponto de variação aberto ou fechado"); o momento em que os variantes devem ser escolhidos (não satisfazendo ao requisito "R1.4 – *Binding time* de um ponto de variação"); especificação de variante(s) que é(são) selecionado(s) por padrão (não satisfazendo ao requisito "R1.6 – Variante(s) selecionado(s) por padrão"); e nem a definição do raciocínio para a escolha de cada variante (não satisfazendo ao requisito "R1.7 – Raciocínio para escolha de variante").

Além disso, não é possível representar variações transversais (não satisfazendo ao requisito "R1.9 – Variações transversais").

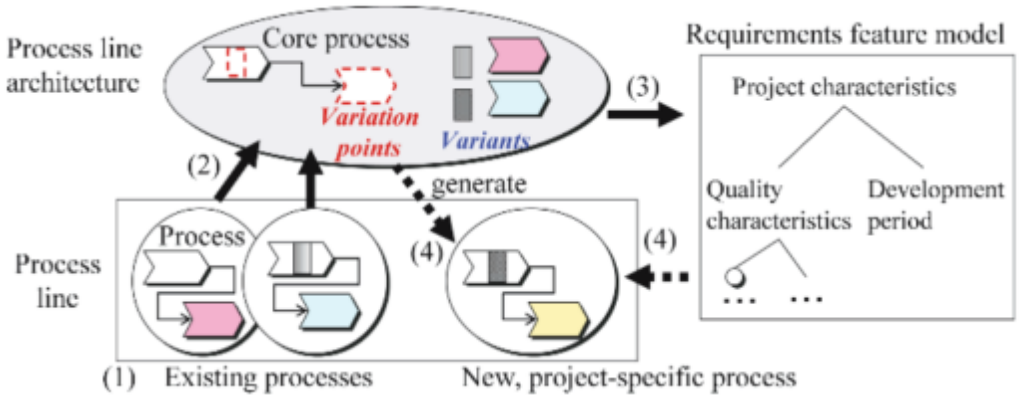


Figura 65 – Abordagem de Linha de Processos (WASHIZAKI, 2006a).

Requisitos de cardinalidades: nenhum dos requisitos de cardinalidades é tratado. Desse modo, não é possível: estabelecer as cardinalidades entre um ponto de variação e seus variantes (não satisfazendo ao requisito "R2.1 – Cardinalidades entre ponto de variação e variantes"); especificar o número de instâncias de um elemento (não satisfazendo ao requisito "R2.2 – Cardinalidades de instâncias"); e nem especificar a cardinalidade na relação entre elementos (não satisfazendo ao requisito "R2.3 – Cardinalidades entre elementos").

Requisitos de elementos de processo: segundo o autor, é possível representar variações em atividades, papéis, produtos de trabalho e ferramentas (satisfazendo, respectivamente, aos requisitos "R3.1 – Atividade", "R3.2 – Papel", "R3.3 – Produto de trabalho" e "R3.4 – Ferramenta"). Entretanto, apenas elementos do tipo "atividade" são apresentados nos modelos e diagramas dos artigos, como pode ser verificado na Figura 66. Não é possível representar conexões com variações (não satisfazendo ao requisito "R3.5 – Conexão").

Requisitos de dependências: dependências do tipo requerimento são expressas através de elementos com o estereótipo <<requires>> (satisfazendo ao requisito "R4.1 – Requerimento") e dependências do tipo exclusão são representadas pelo estereótipo <<conflict>> (satisfazendo ao requisito "R4.2 – Exclusão") (Figura 66). Entretanto, não são tratadas dependências do tipo sugestão (não satisfazendo ao requisito "R4.3 – Sugestão") e substituição (não satisfazendo ao requisito "R4.4 – Substituição").

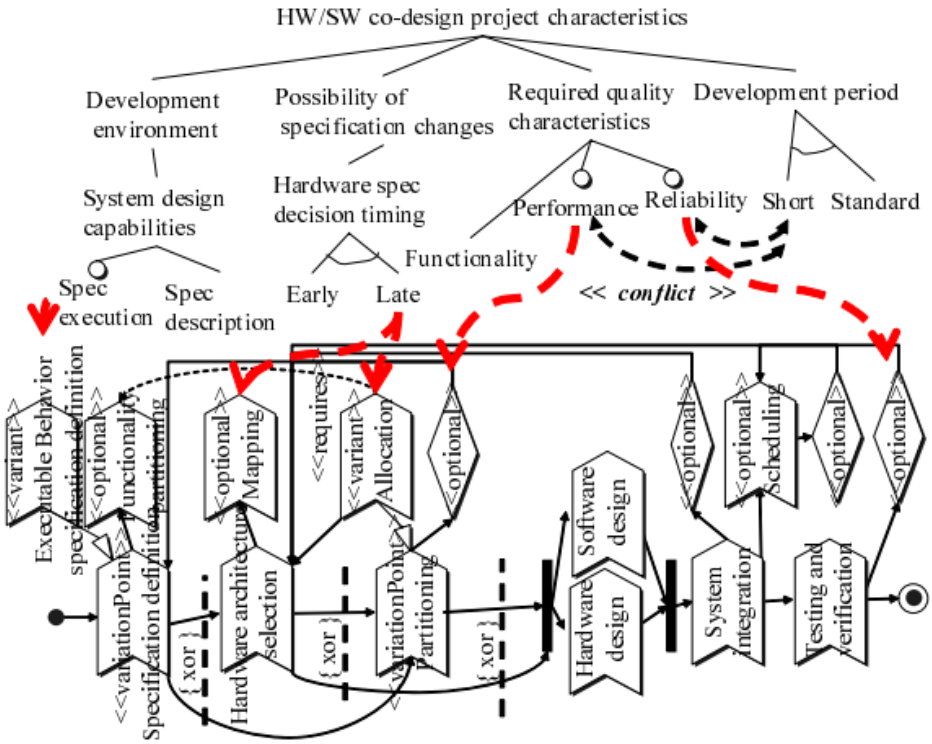


Figura 66 – Variações e Dependências (WASHIZAKI, 2006a).

4.4.18. Washizaki (2006b)

A publicação apresenta um *framework* para a representação de similaridades e variações em linha de processos. O autor define conceitos e elaborou uma notação gráfica, possibilitando inferir elementos de um metamodelo para a representação das variações.

Requisitos de variações: elementos opcionais são identificados através do estereótipo <<optional>> (satisfazendo ao requisito "R1.1 – Elementos opcionais e obrigatórios") (Figura 67 e Figura 68). Mas o momento de escolha de um elemento opcional não é tratado no trabalho (não satisfazendo ao requisito "R1.5 – *Binding time* de elementos opcionais"), tampouco é possível definir o raciocínio para a seleção ou não de elementos opcionais (não satisfazendo ao requisito "R1.8 – Raciocínio para escolha de elemento opcional").

Pontos de variação e variantes são representados pelos estereótipos <<variationPoint>> e <<variant>>, respectivamente (satisfazendo ao requisito "R1.2 – Pontos de variação e variantes") (Figura 67 e Figura 68). Entretanto, não foram encontradas informações sobre: a representação de pontos de variação abertos ou fechados (não satisfazendo ao requisito "R1.3 – Ponto de variação aberto ou fechado"); o momento em que os variantes devem ser escolhidos (não satisfazendo ao requisito "R1.4 – *Binding time* de um ponto de variação"); especificação de variante(s) que é(são) selecionado(s) por padrão (não satisfazendo ao requisito "R1.6 –

Variante(s) selecionado(s) por padrão"); e nem a definição do raciocínio para a escolha de cada variante (não satisfazendo ao requisito "R1.7 – Raciocínio para escolha de variante").

Também não foram encontradas informações sobre a possibilidade de representar variações transversais (não satisfazendo ao requisito "R1.9 – Variações transversais").

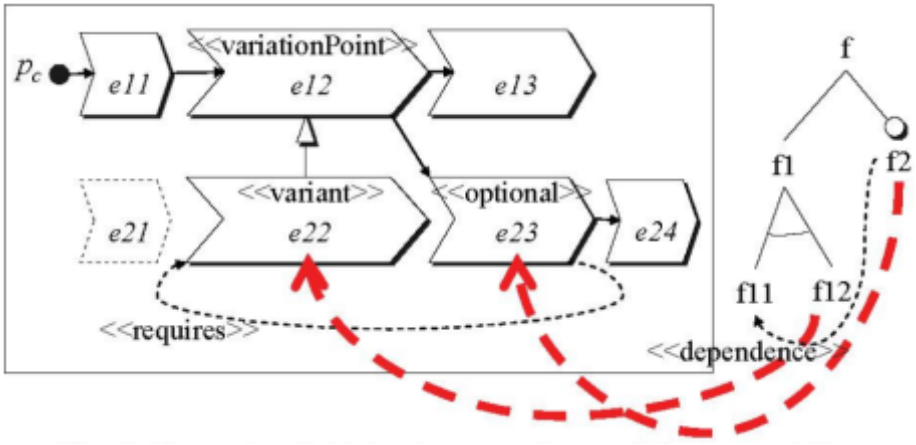


Figura 67 – Variações e Dependências (WASHIZAKI, 2006b).

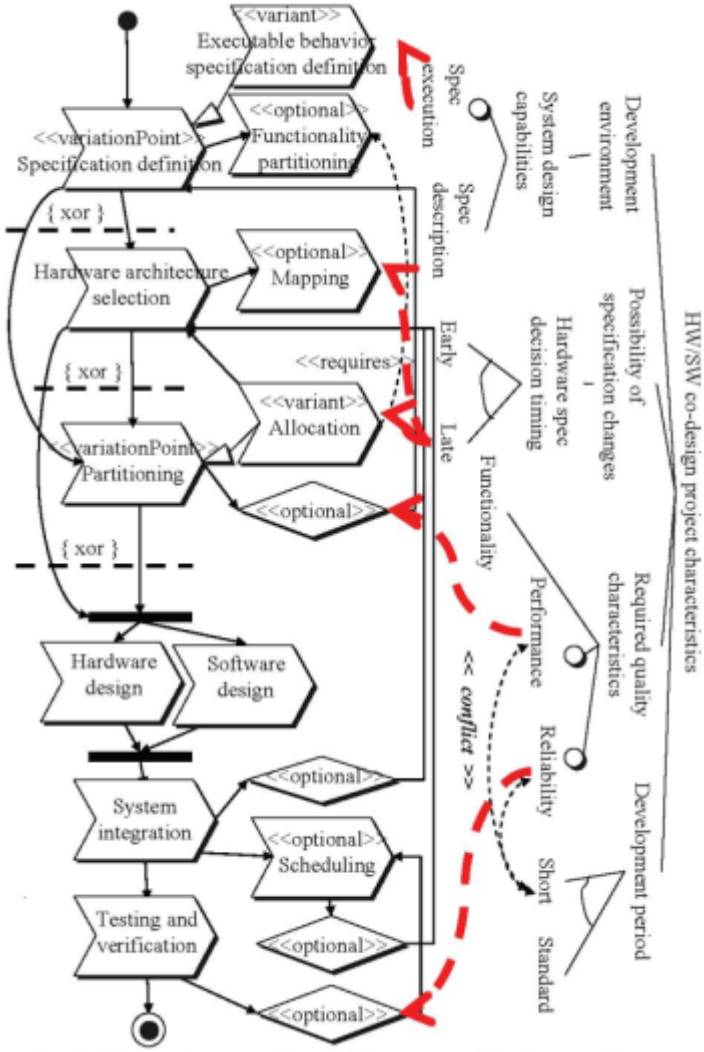


Figura 68 – Variações e Dependências (WASHIZAKI, 2006b).

Requisitos de cardinalidades: nenhum dos requisitos de cardinalidades é tratado. Desse modo, não é possível: estabelecer as cardinalidades entre um ponto de variação e seus variantes (não satisfazendo ao requisito "R2.1 – Cardinalidades entre ponto de variação e variantes"); especificar o número de instâncias de um elemento (não satisfazendo ao requisito "R2.2 – Cardinalidades de instâncias"); e nem especificar a cardinalidade na relação entre elementos (não satisfazendo ao requisito "R2.3 – Cardinalidades entre elementos").

Requisitos de elementos de processo: segundo o autor, é possível representar variações em atividades, papéis, produtos de trabalho e ferramentas (satisfazendo, respectivamente, aos requisitos "R3.1 – Atividade", "R3.2 – Papel", "R3.3 – Produto de trabalho" e "R3.4 – Ferramenta"). Entretanto, apenas elementos do tipo "atividade" são apresentados nos modelos e diagramas dos artigos, como pode ser verificado na Figura 67 e na Figura 68.

Não foram encontradas informações sobre a representação de conexões com variações (não satisfazendo ao requisito "R3.5 – Conexão").

Requisitos de dependências: dependências do tipo requerimento são expressas através de elementos com o estereótipo <<requires>> (satisfazendo ao requisito "R4.1 – Requerimento") (Figura 67) e dependências do tipo exclusão são representadas pelo estereótipo <<conflict>> (satisfazendo ao requisito "R4.2 – Exclusão") (Figura 68). Entretanto, não são tratadas dependências do tipo sugestão (não satisfazendo ao requisito "R4.3 – Sugestão") e substituição (não satisfazendo ao requisito "R4.4 – Substituição").

4.5. Considerações Finais

No presente Capítulo, foram estabelecidos requisitos para a representação de variações em linha de processos de software. Os requisitos foram coletados a partir de trabalhos que definiram metamodelos e/ou notações gráficas para a representação de variações. Em seguida, metamodelos existentes foram comparados para que seja avaliado o atendimento a tais requisitos.

Os requisitos para a representação de variações estabelecidos nesta dissertação não são definitivos, significando que requisitos importantes para atender às necessidades de organizações ou contextos específicos podem ter sido ignorados. Além disso, não faz parte do escopo deste trabalho avaliar a real utilidade, os benefícios e desvantagens de um metamodelo atender a cada um dos requisitos estabelecidos.

Adicionalmente, a avaliação do atendimento aos requisitos pelos metamodelos pode ter sofrido o viés do julgamento do pesquisador (essa possibilidade é minimizada pelas justificativas apresentadas na seção 4.4). Portanto, outros pesquisadores podem considerar um conjunto

diferente de requisitos ou descrever um mesmo requisito com semântica diferente; além de julgarem de forma diferente o atendimento aos requisitos estabelecidos. Finalmente, informações importantes para a avaliação do atendimento ou não dos requisitos podem ter sido omitidas nas publicações.

5. Abordagem Proposta

Este Capítulo apresenta a proposta de um metamodelo para a representação de variações, além de uma notação gráfica e um processo de engenharia de linha de processos de software. Por fim, é apresentado um exemplo de modelagem de uma linha de processos utilizando a abordagem proposta neste trabalho.

5.1. Introdução

Um metamodelo robusto e formal fornece uma base sólida para a representação de linha de processos, propiciando a utilização de ferramentas para a definição e execução de processos (RAUSCH; KUHRMANN, 2011, p. 232). Seguindo esse princípio, o principal foco desta dissertação de mestrado é a elaboração de um metamodelo.

Apesar da importância do metamodelo, também devem ser levados em consideração a notação gráfica e o processo de engenharia utilizado para a construção e instanciação de linhas de processos. Este Capítulo apresenta, nas próximas seções, o metamodelo, a notação gráfica e o processo propostos nesta dissertação de mestrado.

5.2. Metamodelo

No contexto deste trabalho, um metamodelo foi elaborado para a representação de variações no domínio de processos de software, através da utilização de conceitos do paradigma de linha de processos de software. Assim, o metamodelo engloba os tipos de variações que podem ocorrer, os tipos de dependências e as relações com os elementos de processo de software. Os nomes das classes e atributos do metamodelo foram definidos em inglês visando a publicação do trabalho em eventos científicos internacionais.

Para a elaboração do metamodelo, primeiramente foram identificadas publicações científicas cujos resultados foram a especificação de metamodelos e/ou notações gráficas para a

representação de linha de processos de software. Em seguida, foram extraídos os conceitos presentes em tais publicações, gerando um conjunto de requisitos. Finalmente, foi elaborado um metamodelo com o objetivo de satisfazer a alguns dos requisitos estabelecidos.

Um dos princípios seguidos durante a elaboração do metamodelo foi a separação entre conceitos de variações e os elementos que constituem o modelo de processos, permitindo que as variações possam ser representadas de forma independente da linguagem de modelagem de processos utilizada. Segundo Rouillé *et al.* (2012), essa estratégia é benéfica, visto que o metamodelo de variações pode ser utilizado sem a necessidade de modificar o metamodelo de processos de software. Assim, ferramentas existentes para a modelagem e execução de processos não necessitam de ajustes para a adequação aos conceitos de variações.

Os conceitos presentes no metamodelo foram separados nos seguintes componentes, apresentados na Figura 69:

- **Modelo de características:** representa um modelo de características contendo uma estrutura em árvore de requisitos de processo.
- **Arquitetura de linha de processos:** representa uma arquitetura de linha de processos. Este componente é subdividido nos seguintes subcomponentes:
    - **Variabilidades:** contém os conceitos relacionados aos tipos de variações que podem ser representados.
    - **Dependências:** representa os tipos de dependências que podem ser representados.
    - **Modelo de resolução:** representa os conceitos relacionados com as possíveis resoluções de variações em uma arquitetura de linha de processos.
    - **Configuração padrão:** representa as possíveis configurações que podem ser utilizadas para diminuir os esforços na engenharia de domínio e engenharia de aplicação.
- **Elementos de processo:** contém os conceitos relacionados aos tipos de elementos de processo de software, como atividades, papéis, produtos de trabalho e conectores em que as variações podem ser aplicadas.

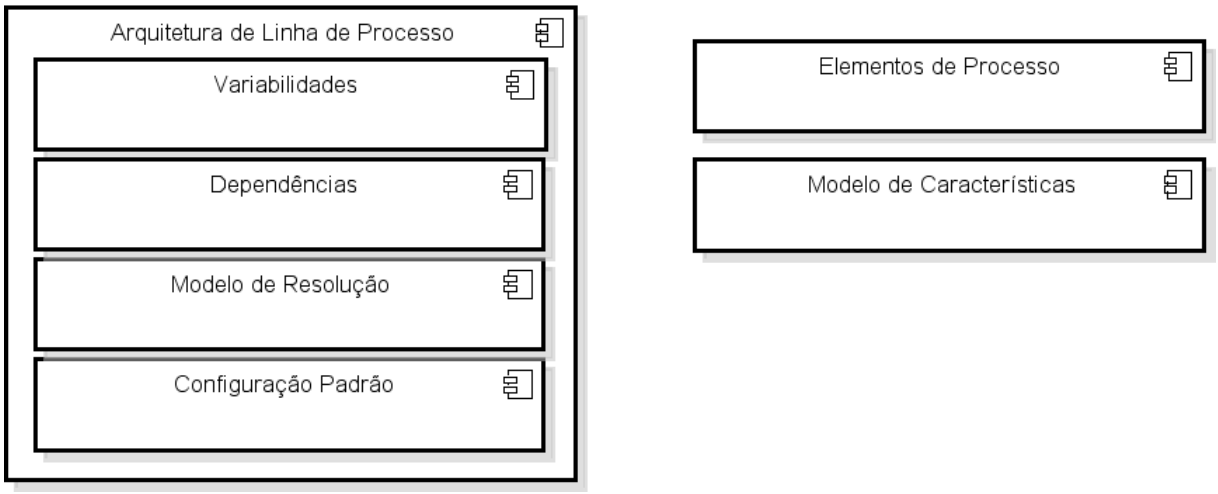


Figura 69 – Componentes do Metamodelo.

A Figura 70 apresenta a visão geral do metamodelo proposto nesta dissertação de mestrado.

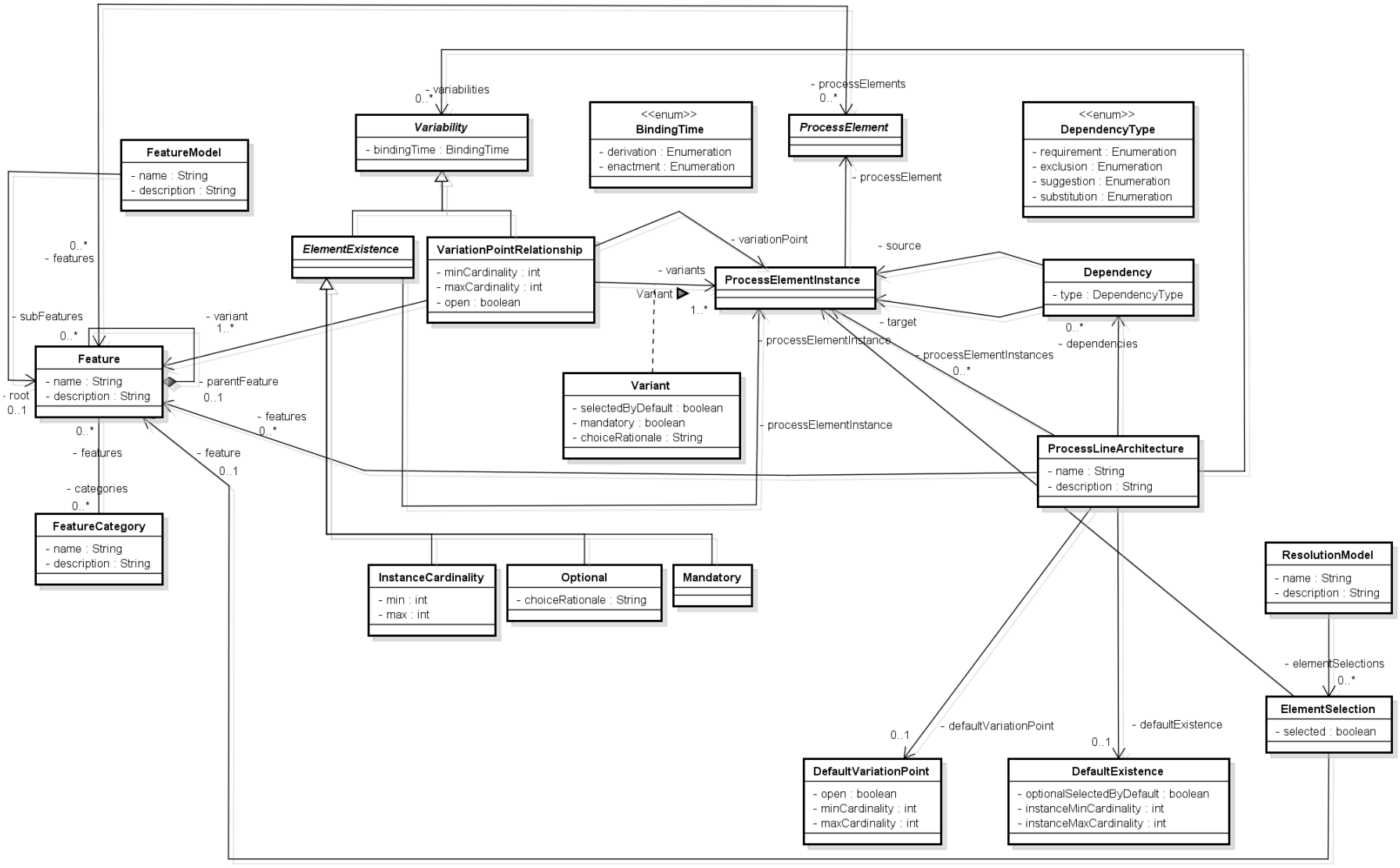


Figura 70 – Visão Geral do Metamodelo.

5.2.1. Modelo de Características

Este componente do metamodelo representa os conceitos relacionados aos modelos de características (Figura 71).

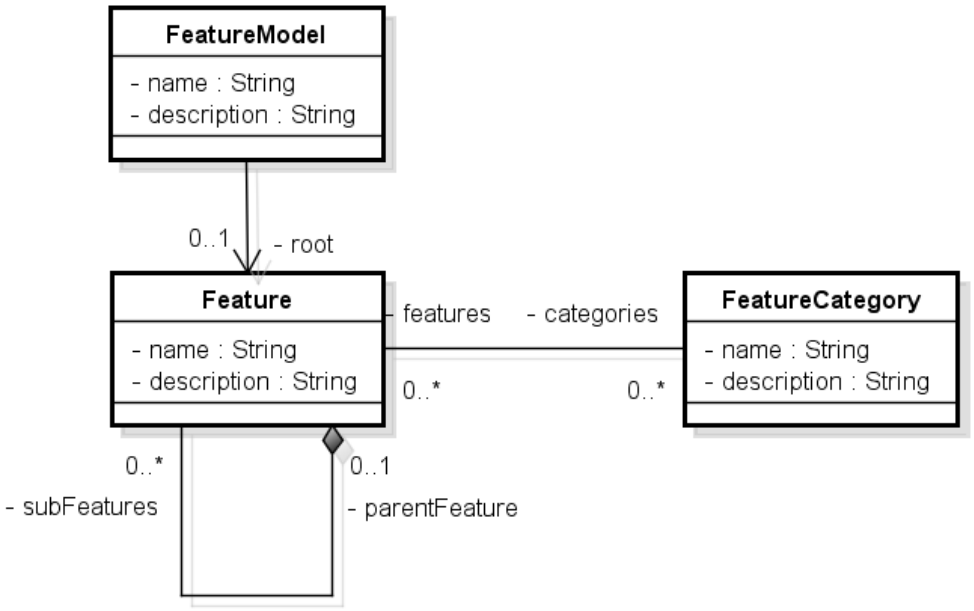


Figura 71 – Modelo de Características.

Um modelo de características no metamodelo proposto representa um conjunto de características em um determinado domínio. Um modelo de características possui os atributos textuais "name" e "description", que representam, respectivamente, seu nome e descrição. O nome e a descrição devem ser utilizados para registrar o escopo e o domínio do modelo de características e são importantes para aumentar a sua possibilidade de recuperação e reutilização por arquiteturas de linha de processos. Além disso, existe a associação "root", que aponta para a característica raiz do modelo de características, que é ancestral de todas as demais características que compõem o modelo de características e representa um conceito em alto nível de abstração. Em um determinado modelo de características, pode existir no máximo uma característica raiz.

Uma característica (classe "Feature") representa um requisito que um processo deve satisfazer. Além dos atributos nome ("name") e descrição ("description"), uma característica pode possuir uma característica pai (associação "parentFeature") e várias subcaracterísticas (associação "subFeatures"). A característica pai representa o ascendente direto de uma característica e as subcaracterísticas representam os seus descendentes diretos. Somente a característica raiz de um modelo de características não possui uma característica pai. Essa estrutura de subcaracterísticas e característica pai permite a modelagem de características em

árvore, propiciando o agrupamento de características similares e a representação hierárquica em vários níveis de abstração.

Uma categoria de característica (classe "FeatureCategory") representa um agrupamento de características por categoria. Esse é um tipo de relacionamento em um alto nível de abstração e pode agrupar, por exemplo, características que representam modelos de maturidade. Assim, características como "CMMI-DEV" (SEI, 2010) e "MR-MPS-SW" (SOFTEX, 2012) podem ambas ser classificadas na categoria "Modelos de Maturidade", facilitando, dentre outras coisas, a sua localização e recuperação. Existe uma associação muitos-para-muitos entre as classes "Feature" e "FeatureCategory", representando que uma característica pode se enquadrar em várias categorias e uma categoria pode possuir várias características. Este conceito não está restrito ao escopo de um modelo de características, portanto, características de diferentes modelos de características podem ser classificadas na mesma categoria.

A decisão de incluir no metamodelo o conceito de categoria de característica se deve ao seguinte motivo: em um modelo de características, pode-se modelar uma característica raiz em um alto nível de abstração, denominada "Modelos de Maturidade", representando diversos modelos de maturidade. Desse modo, as subcaracterísticas diretas poderiam ser "CMMI-DEV" e "MR-MPS-SW". Entretanto, essa estrutura poderia tornar a modelagem de características muito complexa e, para que o esquema de classificação funcionasse adequadamente, todos os modelos de maturidade deveriam ser modelados como subcaracterísticas de tal característica, aumentando exponencialmente o tamanho do modelo de características. Além disso, não seria possível classificar uma mesma característica em mais do que um grupo de interesse, tendo em vista que uma característica possui somente uma característica pai. Por outro lado, a criação do conceito de categoria de característica permite a classificação de características pertencentes a modelos de características diferentes em uma mesma categoria (aumentando a flexibilidade e desacoplamento) e permite a classificação de uma mesma característica em vários grupos de interesse (aumentando a acurácia da classificação).

Na maioria dos trabalhos que utilizam a modelagem de características, as variações e dependências são representadas diretamente nos modelos de características (ALEGRÍA; BASTARRICA, 2012; ALEIXO *et al.*, 2010; ALEIXO *et al.*, 2012; COSTACHE *et al.*, 2011; GOLPAYEGANI *et al.*, 2013; HURTADO *et al.*, 2013; JAFARINEZHAD; RAMSIN, 2012; MAGDALENO *et al.*, 2012; TEIXEIRA, 2011; WASHIZAKI, 2006a; WASHIZAKI, 2006b). Dessa forma, se uma característica for modelada como obrigatória, será obrigatória em todos os contextos em que for reutilizada. Tendo isso em vista, um modelo de características no metamodelo proposto nesta dissertação não representa diretamente variações, mas apenas um

conjunto de características (requisitos de processos) em um determinado domínio estruturado em forma de árvore. Esse desacoplamento teve como objetivo aumentar o grau de reutilização das características. Dessa forma, um modelo de características que representa, por exemplo, um modelo de maturidade poderá ser reutilizado por várias linhas de processos de software, diminuindo o retrabalho e propiciando o compartilhamento de conhecimento entre diferentes contextos.

5.2.2. Arquitetura de Linha de Processos de Software

Este componente do metamodelo representa uma arquitetura de linha de processos e os tipos de variações e dependências que podem ser modelados entre elementos de processo. A Figura 72 apresenta as classes de uma arquitetura de linha de processos.

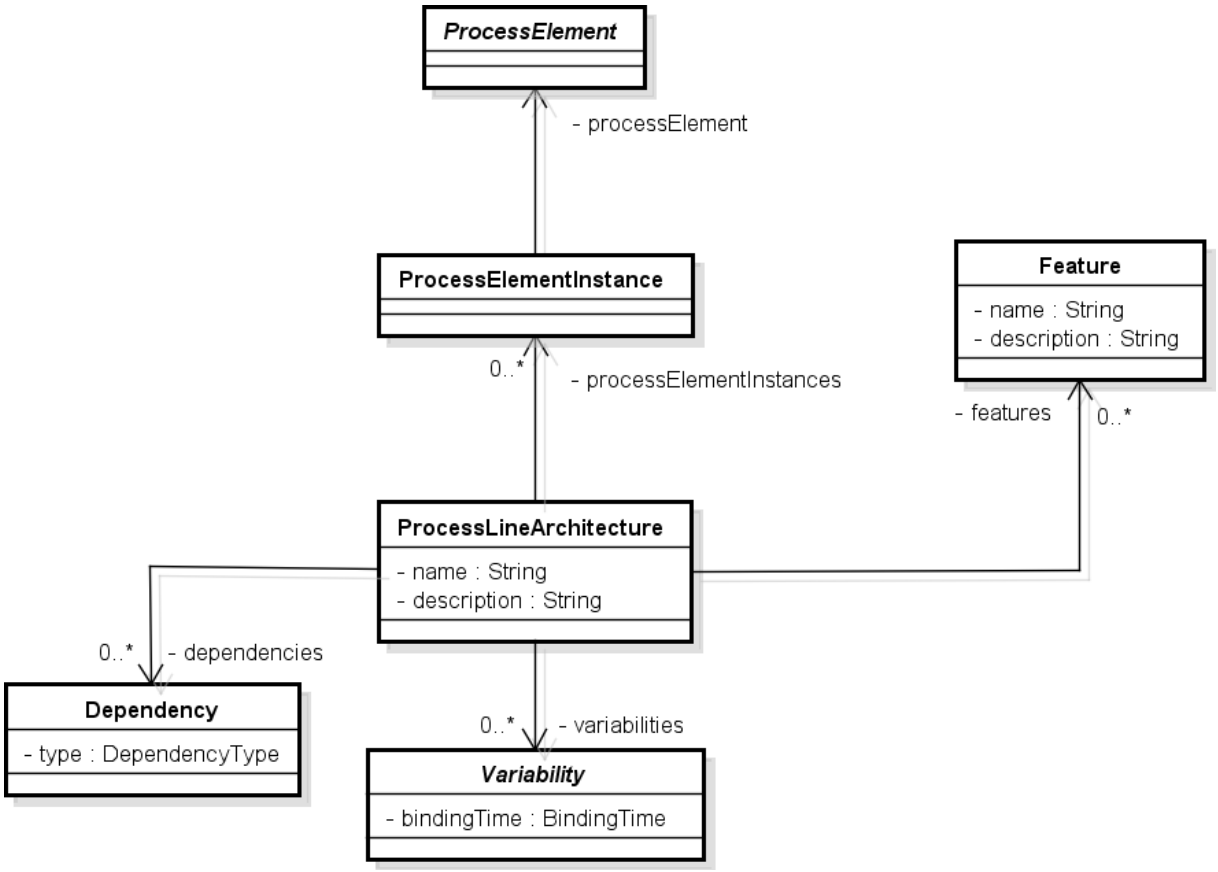


Figura 72 – Arquitetura de Linha de Processos de Software.

Uma arquitetura de linha de processos representa uma estrutura central, que incorpora variações, similaridades e dependências a partir das quais todos os processos de software que compõem uma linha de processos podem ser instanciados.

As variações e dependências representadas pelo metamodelo estão no escopo de uma arquitetura de linha de processos e não do modelo de características. Ou seja, é em uma

arquitetura de linha de processos que os elementos de processo são definidos como opcionais ou obrigatórios, os pontos de variação e variantes são determinados e as dependências são definidas.

Uma arquitetura de linha de processos pode referenciar várias características, através da associação "features", que representa as características que são associadas ao domínio. Uma mesma característica pode ser associada à várias arquiteturas de linha de processos.

5.2.2.1. Variabilidades

A Figura 73 apresenta os conceitos de variações contidos em uma ALPrS.

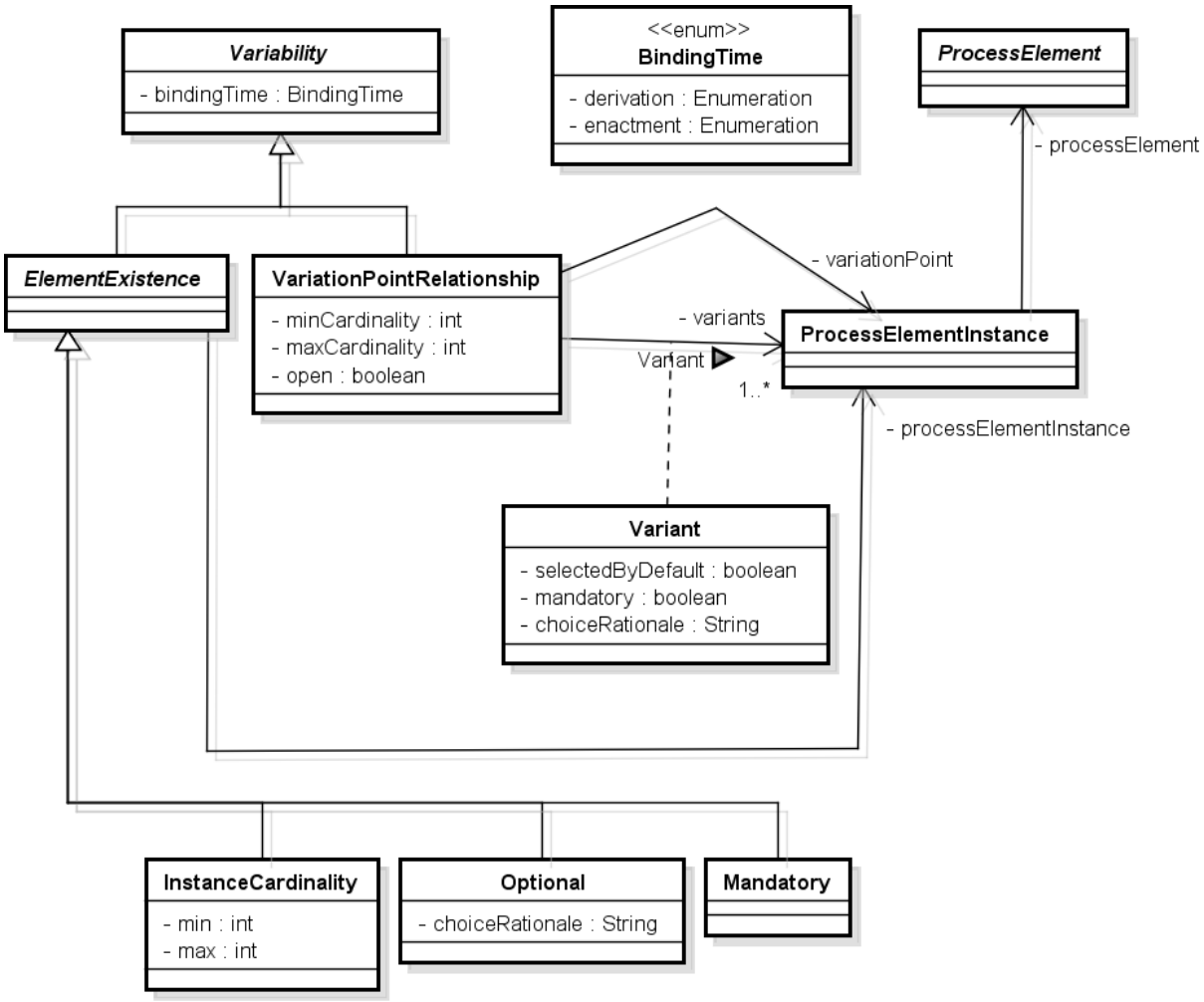


Figura 73 – Variabilidades

Os tipos de variações permitidas para uma arquitetura de linha de processos são representados pela classe "Variability" e suas subclasses: "ElementExistence", "VariationPointRelationship".

A classe "ProcessElementInstance" representa uma instância de um elemento de processo e as variações associadas a ela em uma arquitetura de linha de processos. As subclasses de "Variability" sempre referenciam indiretamente um elemento de processo, através da classe "ProcessElementInstance". Um mesmo elemento de processo (classe "ProcessElement") pode

ser adicionado diversas vezes à uma ALPrS, sendo que, em cada referência, um tipo de variação diferente pode ser associado ao elemento. Isso permite, por exemplo, que uma atividade como "Testar Software" possa ser adicionada várias vezes em uma mesma ALPrS, sempre após um atividade de implementação, mas somente seja especificada como obrigatória quando um componente executável é implementado, e opcional, caso o componente não seja executável. Nesse caso, cada instância de "Testar Software" será representada por um objeto da classe "ProcessElementInstance".

A classe "ElementExistence" especifica o tipo de variação de uma instância de elemento de processo, através da associação "processElementInstance" com a classe "ProcessElementInstance", especificando uma relação existencial que pode ter diferentes significados, de acordo com as três subclasses de "ElementExistence": "Mandatory", "Optional" e "InstanceCardinality".

A classe "Mandatory" especifica uma instância de elemento de processo como obrigatória no contexto da linha de processos sendo modelada. Ou seja, a instância do elemento de processo deve estar presente em todos os processos da linha de processos.

Por outro lado, a classe "Optional" especifica uma instância de elemento de processo como opcional no contexto da linha de processos. O atributo "choiceRationale", presente na classe "Optional", tem como objetivo especificar uma descrição textual com o contexto em que uma instância opcional de elemento de processo deve ser selecionada, de forma a auxiliar na tomada de decisão sobre a inclusão ou não do elemento.

Uma restrição do metamodelo é que uma instância de elemento de processo em uma ALPrS deve ser sempre especificada como opcional ou obrigatória, ou seja, deve haver pelo uma instância de "Optional" ou "Mandatory" referenciando cada instância de elemento de processo. Assim, quando um elemento de processo é adicionado à ALPrS, automaticamente uma instância de "ElementExistence" é criada para o elemento, especificando se tal elemento é obrigatório ou opcional na linha de processos de software em questão. Entretanto, uma mesma instância de elemento de processo não pode ser especificada simultaneamente como opcional e obrigatória.

A classe "InstanceCardinality" especifica o número mínimo e máximo, através dos atributos "min" e "max", respectivamente, de instâncias que podem existir de um determinado elemento de processo em uma linha de processo de software. Caso não exista uma instância de "InstanceCardinality" referenciando uma instância de elemento de processo, o número de instâncias se torna indefinido e não é controlado pela linha de processos.

A classe "VariationPointRelationship" representa um relacionamento entre instâncias de elementos de processo, que podem exercer o papel de ponto de variação e de variante. O ponto

de variação é referenciado pela associação "variationPoint" e os variantes pela classe associativa "Variant". Uma instância de elemento de processo que faz o papel de variante em um relacionamento pode fazer o papel de ponto de variação em outro relacionamento, permitindo a especificação de vários níveis de abstração nos relacionamentos entre pontos de variação e variantes.

Existe a possibilidade de especificar se um variante é selecionado por padrão (valor verdadeiro) ou não (valor falso), através do atributo "selectedByDefault" da classe associativa "Variant". Um variante selecionado por padrão é previamente selecionado durante a engenharia de aplicação, entretanto, isso não significa que a instância de elemento de processo seja obrigatória, pois poderá ser desmarcada posteriormente. Quando um variante especificado como selecionado por padrão se referir a uma instância obrigatória de elemento de processo, o valor deve necessariamente ser verdadeiro. Ou seja, um variante obrigatório é sempre selecionado por padrão em um ponto de variação. Adicionalmente, não pode haver mais variantes obrigatórios do que a cardinalidade máxima de um ponto de variação. Nesse caso, a cardinalidade máxima deve ser igual ou maior do que o número de variantes obrigatórios.

A classe "VariationPointRelationship" possui os atributos "minCardinality" e "maxCardinality", que especificam, respectivamente, a cardinalidade mínima e a cardinalidade máxima de um ponto de variação. A cardinalidade mínima especifica o número mínimo de variantes que devem ser selecionados em um ponto de variação. Por outro lado, a cardinalidade máxima especifica o número máximo de variantes que podem ser selecionados.

Alguns metamodelos não permitem a seleção de mais do que um variante em um ponto de variação. Entretanto, podem ocorrer situações em que essa possibilidade pode ser útil. Por exemplo, existem várias formas de realizar a atividade de levantamento de requisitos, como através de entrevista, análise de sistemas similares e conversa informal. Essa situação poderia ser representada da seguinte forma: a atividade "levantamento de requisitos" seria um ponto de variação e as atividades "entrevista", "análise de sistemas similares" e "conversa informal" seriam modeladas como variantes, em que um ou mais variantes poderiam ser escolhidos.

O atributo booleano "open" especifica se o ponto de variação é aberto (valor verdadeiro) ou fechado (valor falso). Em um ponto de variação fechado, somente os variantes previamente atrelados a ele na engenharia de domínio poderão ser selecionados durante a engenharia de aplicação. Por outro lado, se um ponto de variação for aberto, um variante diferente, que não foi previsto inicialmente, poderá ser selecionado para implementá-lo.

Especificar um ponto de variação como fechado aumenta o controle e a consistência do processo derivado, porém pode deixar a engenharia de aplicação pouco flexível. Por outro lado,

especificar um ponto de variação como aberto pode aumentar a flexibilidade, porém pode dificultar a garantia da consistência do processo derivado. Além disso, devem ser especificadas regras para a inclusão de um variante fora do conjunto predeterminado de variantes. Vale ressaltar que, caso um ponto de variação seja aberto, pode haver a necessidade de especificar restrições que indicam os requisitos que um variante deve satisfazer para poder ser selecionado nesse tipo de ponto de variação. Entretanto, o metamodelo proposto neste trabalho não tratou desse assunto e, portanto, qualquer elemento poderá ser selecionado em um ponto de variação aberto.

O atributo "bindingTime", da classe "Variability", especifica o momento do ciclo de vida em que a resolução de variações deve ocorrer: durante a derivação da linha de processos ou durante a execução de processo. Por estar no escopo da classe "Variability", se aplica tanto a elementos opcionais quanto a pontos de variação, porém não é aplicável aos elementos obrigatórios, visto que estes estarão sempre presentes no processo derivado. Quando se referir a um elemento opcional, especifica quando a decisão de inclui-lo ou não deve ser realizada e quando se referir a um ponto de variação indica quando os variantes relacionados devem ser escolhidos. A escolha durante a derivação deve ser obrigatoriamente resolvida antes da instanciação do modelo de processo e a escolha durante a execução implica na possibilidade de instanciar o modelo de processo mesmo antes de realizar as escolhas necessárias. Esse segundo caso visa flexibilizar a execução de processos, tendo em vista que processos são entidades dinâmicas e nem todas as decisões podem ser tomadas antes do início da execução (LIMA REIS, 2003). Apesar de aumentar a flexibilidade, a resolução de variações durante a execução aumenta a complexidade, pois regras de consistências específicas devem ser definidas. Por exemplo, se um elemento A previamente selecionado e executado requer um elemento B (escolhido em tempo de execução) opcional, o elemento B, apesar de opcional, não poderá ser removido, visto que o elemento A já foi executado no processo de software e, portanto, não pode mais ser removido. Outro exemplo que pode ocorrer é quando um elemento B (escolhido em tempo de execução) requer um elemento A que já foi previamente removido no processo executado. Nesse caso, não é mais possível satisfazer à dependência existente. Entretanto, apesar de necessária, as regras para a resolução de variações em tempo de execução não foram definidas no metamodelo proposto neste trabalho, sendo este um dos trabalhos futuros previstos.

Para auxiliar na tomada de decisão sobre o contexto em que cada variante é mais adequado, a classe associativa "Variant" possui o atributo "choiceRationale". Além disso, a classe "Variant" possui o atributo "selectedByDefault", que especifica se o variante é selecionado (valor verdadeiro) ou não (valor falso) por padrão. É recomendado que um variante seja especificado

como selecionado por padrão quando está presente na maioria das instâncias da linha de processos de software, poupando esforços para removê-lo. De forma similar, se o variante normalmente não é selecionado pelo ponto de variação, é recomendado que seja modelado como não selecionado por padrão.

5.2.2.2. Dependências

Este componente do metamodelo representa as possíveis dependências que podem ser modeladas entre os elementos de processo de uma ALPrS (Figura 74).

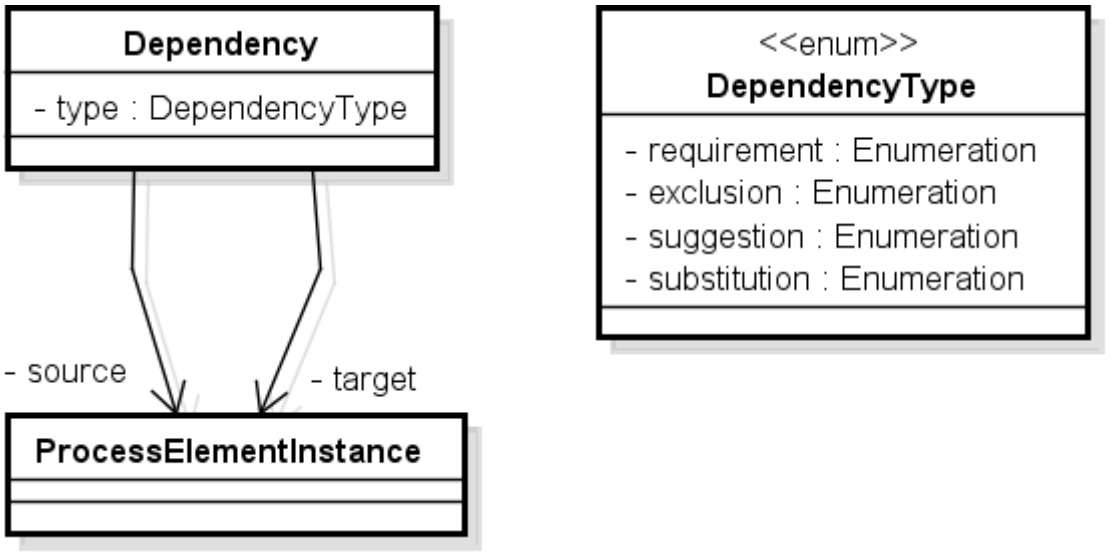


Figura 74 – Dependências.

No metamodelo proposto, dependências representam conexões entre dois elementos de processo no escopo de uma arquitetura de linha de processos, que têm como objetivo aumentar o grau de consistência dos processos gerados, evitando conflitos entre elementos e garantindo que elementos complementares sejam inseridos concomitantemente.

Dependências são representadas pela classe "Dependency" (dependência), que podem ser de quatro tipos diferentes: "requirement" (requerimento),"exclusion" (exclusão), "suggestion" (sugestão) e "substitution" (substituição). Os tipos de dependências são representados pela enumeração "DependencyType".

Independentemente do tipo, dependências representam relacionamentos binários e unidirecionais entre instâncias de elementos de processo, no qual uma instância faz o papel de origem e uma instância faz o papel de destino, através, respectivamente, das associações "source" e "target" da classe "Dependency". O significado que as instâncias de elemento de processos de origem e destino exercem na relação de dependência depende do tipo de dependência, como segue:

- **Requerimento:** se a instância de elemento que faz o papel de origem for selecionada, então a instância de elemento que faz o papel de destino também deve ser selecionada. Nesse caso, não é possível remover a instância de elemento de destino até que a instância de elemento de origem também seja removida.
- **Exclusão:** se a instância de elemento que faz o papel de origem for selecionada, então a instância de elemento que faz o papel de destino não deve ser selecionada. Nesse caso, não é possível selecionar a instância de elemento de destino até que a instância de elemento de origem seja desmarcada.
- **Sugestão:** se a instância de elemento que faz o papel de origem for selecionada, então é sugerido que a instância de elemento que faz o papel de destino seja selecionada. Nesse caso, é possível desmarcar a instância de elemento de destino mesmo que a instância de elemento de origem seja selecionada, visto que a dependência significa apenas uma sugestão.
- **Substituição:** se a instância de elemento que faz o papel de origem não for selecionada, então a instância de elemento que faz o papel de destino deve ser selecionada. Ou seja, a instância de elemento de destino é substituta da instância de elemento de origem da dependência. Nesse caso, não é possível desmarcar a instância de elemento de destino até que a instância de elemento de origem seja selecionada.

Por ser uma relação binária, somente é possível definir dependências entre duas instâncias de elemento. Por exemplo, para uma dependência do tipo requerimento, é possível definir que, se a instância de elemento A for selecionada, então a instância de elemento B também deve ser selecionada. Entretanto, não é possível definir uma relação no formato: se a instância de elemento A for selecionada, então as instâncias de elemento B e C também devem ser selecionadas. Essa não foi considerada como uma limitação, pois é possível contornar essa situação definindo duas dependências complementares: (i) se a instância de elemento A for selecionada, então a instância de elemento B também deve ser selecionada; e (ii) se a instância de elemento A for selecionada, então a instância de elemento C também deve ser selecionada.

O aspecto unidirecional de uma dependência implica, por exemplo, que se a instância de elemento A requer a instância de elemento B, não significa que a instância de elemento B requer a instância de elemento A. Nesse exemplo, para que a dependência seja recíproca, é necessária a definição de duas dependências: (i) a instância de elemento A requer a instância de elemento B; e (ii) a instância de elemento requer a instância de elemento A. Essa decisão de definir somente dependências unidirecionais foi tomada devido ao fato de ser possível definir um conjunto de

dependências unidirecionais equivalentes a uma dependência bidirecional. Portanto, essa também não foi considerada como uma limitação do metamodelo. Alguns trabalhos permitem a definição de dependências bidirecionais (BARRETO, 2011; TEIXEIRA, 2011).

As dependências são definidas entre as instâncias de elementos de processo de uma arquitetura de linha de processos e não entre as características de um modelo de características. Essa decisão arquitetural teve como objetivo aumentar o grau de reutilização dos modelos de características. Por exemplo, sejam duas características A e B. Se fosse definida uma dependência do tipo exclusão entre A e B no modelo de características, então tal dependência estaria presente em todas as arquiteturas de linha de processos que reutilizassem o modelo de características, mesmo que a coexistência das duas características faça sentido em uma determinada linha de processos, o que pode diminuir a flexibilidade e o grau de reutilização do modelo de características. Portanto, no contexto deste metamodelo, foi decidida que as dependências sejam definidas entre as instâncias de elementos de processo no escopo de uma arquitetura de linha de processos.

5.2.2.3. Modelo de Resolução

Este componente do metamodelo representa o modelo de resolução. A Figura 75 apresenta as classes que constituem uma modelo de resolução.

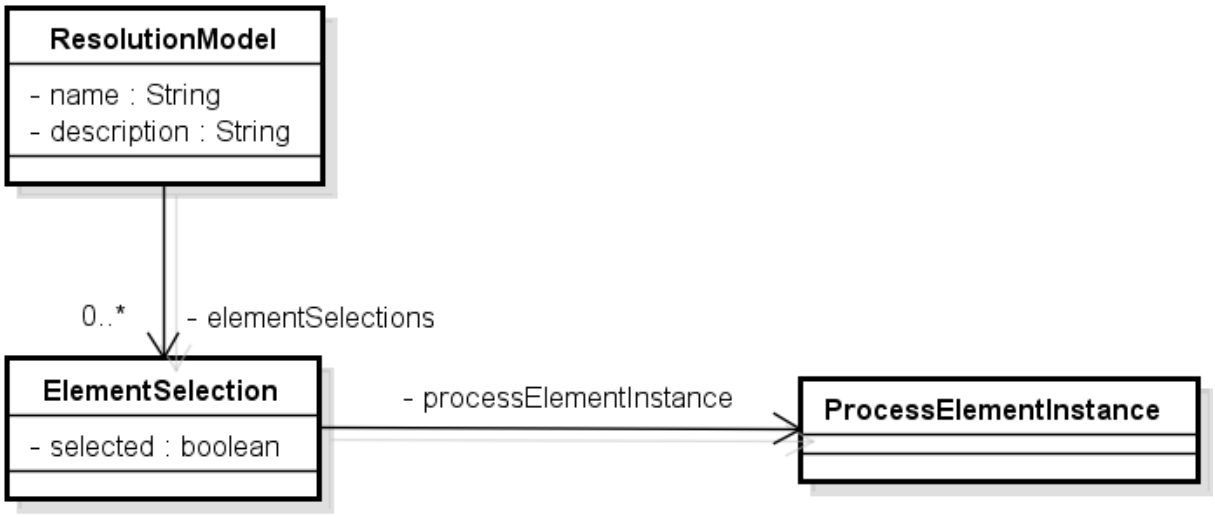


Figura 75 – Modelo de Resolução.

Um modelo de resolução (classe "ResolutionModel") define as resoluções sobre a seleção ou não dos elementos de processo no contexto de uma linha de processos de software. Ou seja, esse modelo armazena escolhas como: a inclusão ou não de elementos opcionais e quais variantes de um ponto de variação devem ser selecionados na fase de engenharia de aplicação.

As escolhas resolvidas em um modelo de resolução são representadas pela associação "elementSelections", que referenciam a seleção de instâncias de elementos (classe

"ElementSelection"). O atributo "selected", da classe "ElementSelection", pode receber o valor verdadeiro, quando a instância de elemento de processo referenciada deve ser selecionada; ou falso, quando a instância de elemento de processo referenciada não deve ser selecionada. A associação "processElementInstance", da classe "ElementoSelection", referencia uma instância do elemento de processo, que deve ser selecionado ou não.

Para os mesmos elementos de processo de uma arquitetura de linha de processos, podem existir vários modelos de resolução associados, cada um refletindo um contexto diferente, sendo que, em cada um desses, um conjunto diferente de elementos pode estar selecionado. Por exemplo, pode-se modelar uma arquitetura de linha de processos contendo elementos de processo que representam os vários níveis de maturidade de um determinado modelo de maturidade de processos. Nesse caso, pode-se modelar um modelo de resolução em que os elementos referentes ao primeiro nível de maturidade são selecionados e os níveis mais altos são desmarcados. Adicionalmente, pode-se modelar outro modelo de resolução em que todos os elementos de processo são selecionados, poupando esforços em contextos em que processos de alta maturidade devem ser instanciados.

Vale ressaltar que os diferentes modelos de resolução não alteram o tipo de variação especificado para os elementos de processo, mas somente modificam o estado de seleção destes. Assim, elementos opcionais permanecem sempre opcionais, mas podem estar selecionados em um modelo de resolução e desmarcados em outro. Esse mecanismo pode agilizar o processo de engenharia de aplicação, além de permitir a reutilização do raciocínio de adaptação de um determinado contexto. Mas a reutilização de um modelo de resolução preexistente não é obrigatória, pois elementos ainda podem ser selecionados manualmente.

5.2.2.4. Configuração Padrão

Este componente do metamodelo representa as configurações padrão que podem ser estabelecidas no escopo de em uma linha de processos de software. A Figura 76 apresenta as classes que constituem uma configuração padrão.

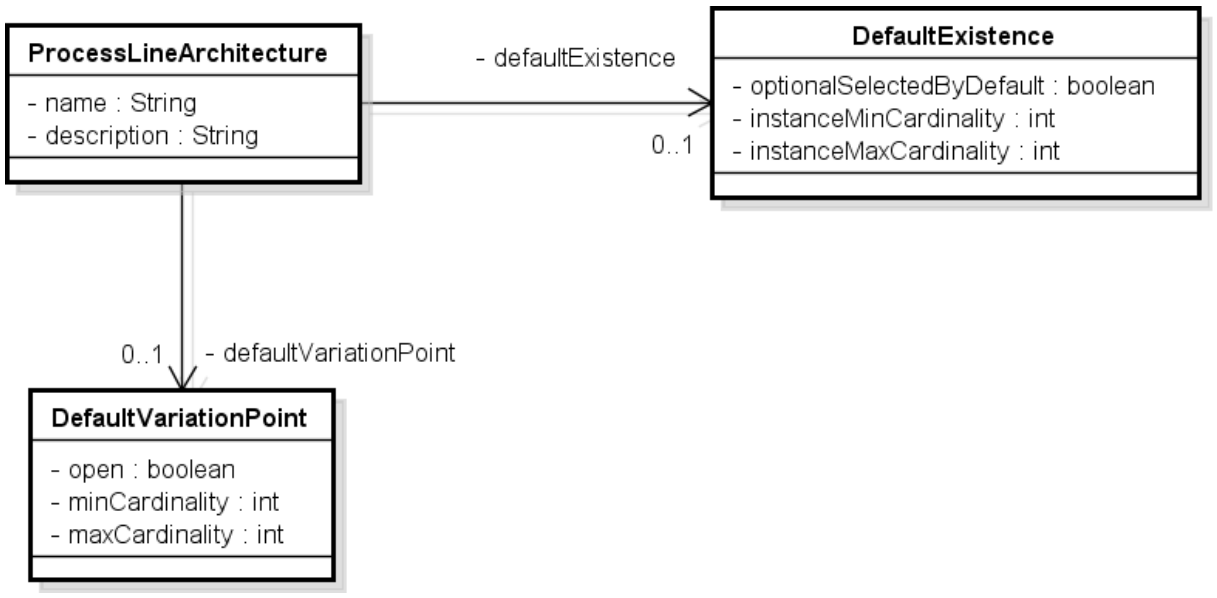


Figura 76 – Configuração Padrão.

As configurações padrão representam valores padrão que serão atribuídos durante os processos de engenharia de domínio e engenharia de aplicação. Esses conceitos somente fazem sentido com a implementação de um apoio ferramental em que os valores podem ser registrados e utilizados para a automação do processo de engenharia de linha de processos.

A classe "DefaultExistence" representa os valores padrão referentes às variações de elementos de processo. Essa classe possui o atributo booleano "optionalSelectedByDefault". Se o valor desse atributo for verdadeiro, então toda instância de elemento opcional inserida será selecionada por padrão no momento da criação de um modelo de resolução. Vale ressaltar que isso não torna a instância de elemento de processo obrigatória, pois ainda poderá ser removida. Por outro lado, se o valor do atributo for falso, então toda instância de elemento de processo opcional inserida será desmarcada por padrão no modelo de resolução.

A classe "DefaultVariationPoint" representa os valores padrão que serão atribuídos durante a criação de pontos de variação. Essa classe possui o atributo booleano "open", que especifica se um ponto de variação criado será aberto (valor verdadeiro) ou fechado (valor falso) por padrão. O atributo "minCardinality" especifica o valor para a cardinalidade mínima padrão de um ponto de variação. Por fim, o atributo "maxCardinality" especifica o valor padrão para a cardinalidade máxima de um ponto de variação.

5.2.3. Elementos de Processo

Este componente do metamodelo representa os elementos de processo específicos das linguagens de modelagem de processos.

A classe abstrata "ProcessElement" (Figura 77) representa uma superclasse para todos os elementos de processo presentes na linguagem de modelagem de processo utilizada como modelo base. A priori, não é possível determinar quais são os tipos de elementos de processo que serão referenciados, pois é uma informação dependente de cada linguagem.

Para referenciar os elementos de processo específicos de uma linguagem, é necessário criar subclasses de "ProcessElement" no metamodelo, cada uma representando um tipo de elemento de processo presente na linguagem de modelagem de processos original. Assim, para cada linguagem, deve existir uma estrutura de subclasses de "ProcessElement". A seção 5.2.3.1 apresenta uma extensão do metamodelo para englobar os elementos de processo da linguagem WebAPSEE-PML.

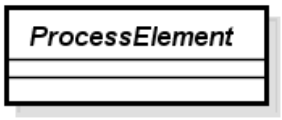


Figura 77 – Elementos de Processo.

Apesar da decisão de desacoplar a representação de variações do metamodelo de processos trazer algumas vantagens, existem questões que devem ser tratadas. A experiência de Rouillé *et al.* (2012) demonstrou que é difícil manter a integridade do processo gerado em relação ao metamodelo de processos utilizado, visto que as regras de boa formação não se encontram no metamodelo de variações e cada metamodelo de processo possui regras específicas que não podem ser generalizadas. Por exemplo, em uma linhagem de modelagem de processos, pode ser permitida a ligação entre um papel e um produto de trabalho. No entanto, tal ligação pode não ser permitida em uma segunda linguagem. Além disso, conceitos específicos podem existir em um metamodelo sem um correspondente em outros metamodelos.

Outra dificuldade relatada por Rouillé *et al.* (2012) é a referência a objetos não mais existentes no modelo de processos. Assim, é necessário verificar periodicamente se os elementos de processo referenciados pelas variações ainda existem no modelo de processos para garantir a consistência dos processos gerados a partir da linha de processos.

Para contornar essas situações, é necessário especificar um conjunto de restrições específicas para cada metamodelo de processo utilizado como modelo base.

5.2.3.1. Extensão do metamodelo WebAPSEE-PML

Esta seção apresenta uma extensão do metamodelo para a definição dos elementos de processo específicos da linguagem de modelagem de processo WebAPSEE-PML (LIMA *et al.*, 2006). Essa linguagem é utilizada pelo ambiente WebAPSEE para a modelagem de processos. O ambiente WebAPSEE é um Ambiente de Engenharia de Software Centrado em Processos

(*Process-Centered Software Engineering Environment - PSEE*), desenvolvido pelo Laboratório de Engenharia de Software (LABES) da Universidade Federal do Pará (UFPA), que tem como objetivo prover maior flexibilidade na gerência e desenvolvimento de processos de software (LIMA REIS; REIS, 2007).

Todas as classes apresentadas nesta seção devem ser subclasses de "ProcessElement", devendo haver uma classe no metamodelo para cada classe que representa um elemento de processo na WebAPSEE-PML.

A classe "Activity" (atividade) representa um comportamento em um processo de software e pode ser classificada em simples (subclasse "Plain") e decomposta (subclasse "Decomposed"). Uma atividade simples pode ser classificada em automática (subclasse "Automatic") e normal (subclasse "Normal"). A Figura 78 apresenta a hierarquia de classes de atividades da lingaguagem WebAPSEE-PML.

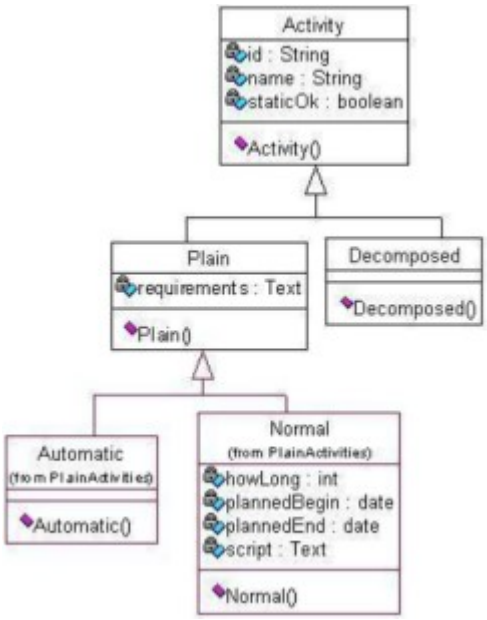


Figura 78 – Tipos de Atividades do Metamodelo WebAPSEE-PML (LABES, 2006).

Atividades podem consumir e produzir artefatos (classe "Artifact") e utilizar recursos (classe "Resource") para a sua execução (Figura 79). Além disso, podem estar envolvidos entidades humanas, como agentes (classe "Agent"), grupos (classe "Group") e papéis (classe "Role").

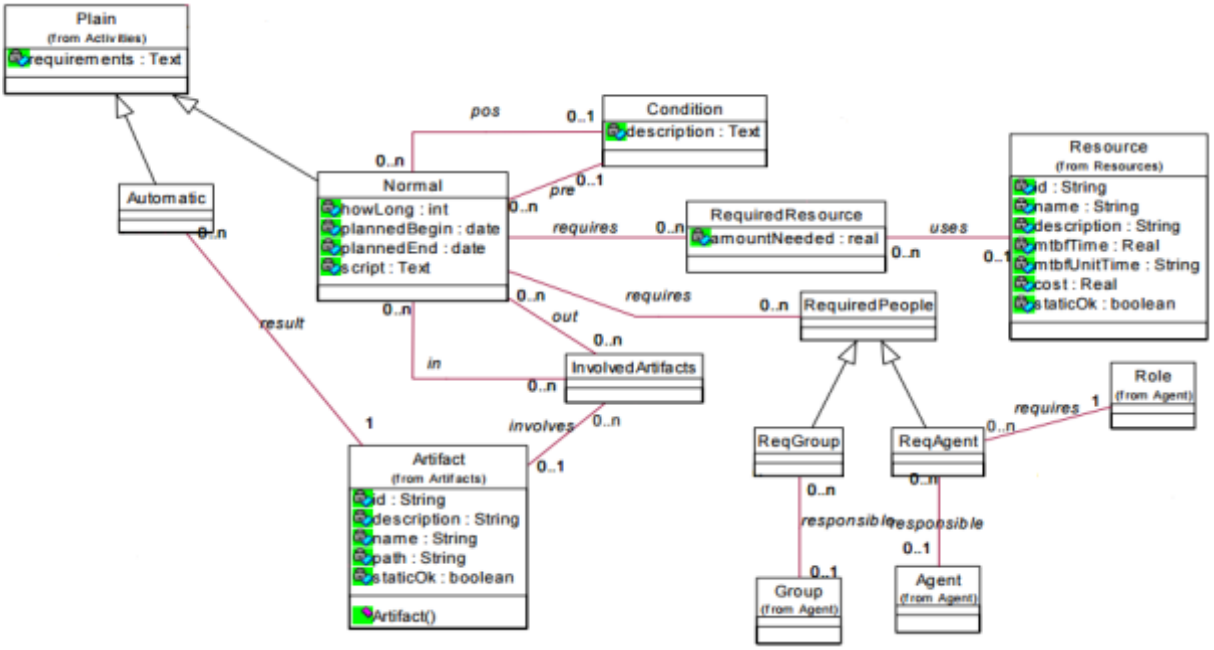


Figura 79 – Metamodelo WebAPSEE-PML – Adaptado de LABES (2006).

A Figura 80 apresenta as conexões entre os elementos de processo. Uma conexão (classe "Connection") pode ser dos tipos simples (classe "SimpleCon"), múltipla (classe "MultipleCon") ou de artefato (classe "ArtifactCon"). As conexões simples e múltiplas especificam o fluxo de trabalho entre atividades, enquanto que as conexões de artefatos associam atividades aos artefatos de entrada e saída. Uma conexão simples pode ser de *feedback* (classe "Feedback") ou sequencial (classe "Sequence"). Uma conexão múltipla pode ser um *branch* (classe "Branch") ou *join* (classe "Join").

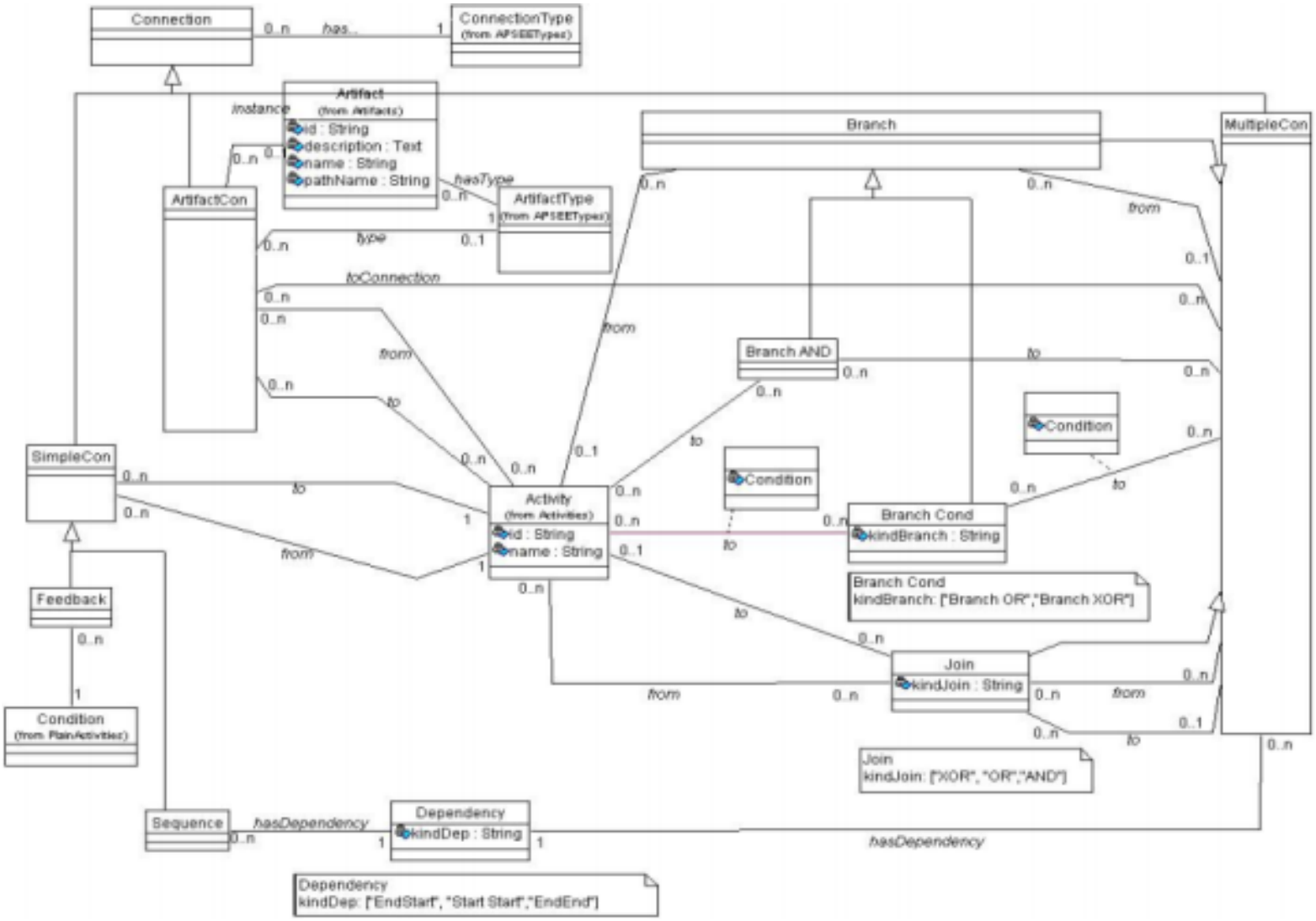


Figura 80 – Tipos de Conexões do Metamodelo WebAPSEE-PML – Adaptado de LABES (2006).

A Tabela 12 apresenta a descrição dos principais elementos de processo da linguagem WebAPSEE-PML.

Tabela 12 – Descrição dos Elementos de Processo da Linguagem WebAPSEE-PML.

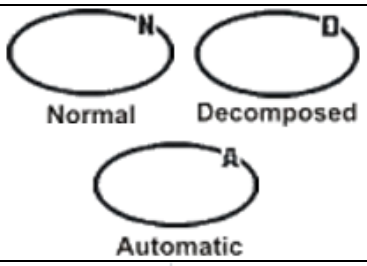
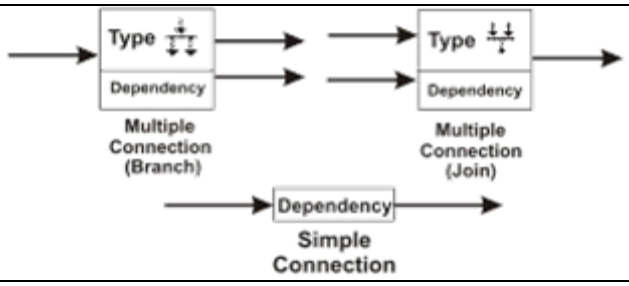
Elemento de Processo	Descrição
Atividade	Elemento que representa as ações realizadas em um processo. Podem ser de três tipos: normal, decomposta e automática.
Atividade Automática	Atividade realizada sem intervenção humana. Apesar de especificada no metamodelo da linguagem WebAPSEE-PML, ainda não existe uma implementação disponível na ferramenta.
Atividade Normal	Atividade indivisível, realizada por papéis humanos.
Atividade Decomposta	Um tipo de atividade que possui um subprocesso, composto por outras atividades.
Artefato	Produto de trabalho que pode servir de entrada ou saída em atividades.
Papel	Uma função exercida por um agente humano responsável pela execução de atividades
Grupo	Um agrupamento de agentes humanos responsável pela execução de atividades.
Recurso	Qualquer recurso necessário para a execução das atividades, como computador, ferramenta ou passagem aérea.
Conexão	Uma conexão representa o fluxo de controle e informações entre as atividades em um processo. Uma conexão entre atividades especifica um tipo de dependência, que define quando as atividades podem ser iniciadas e finalizadas.
Conexões Simples (*Simple Connection*)	Interconecta uma atividade de origem a uma atividade de destino formando um fluxo de controle.
Conexão de *Feedback* (*Feedback Connection*)	Reativa uma ou mais atividades já realizadas, de acordo com condições lógicas associadas. Este tipo de conexão está fora do escopo do metamodelo de LPrS definido neste trabalho.
Conexão de Artefato (*Artifact Connection*)	Conecta artefatos de entrada e saída a atividades.
Conexão Múltipla do Tipo *Branch* (*Multiple Branch Connection*)	Conecta uma atividade de origem a mais de uma atividade de destino, formando uma bifurcação no fluxo de controle.
Conexão Múltipla do Tipo *Join* (*Multiple Join Connection*)	Une uma ou mais atividades de origem a uma atividade de destino.
Dependência end-start	Atividade de destino apenas pode ser iniciada quando a atividade de origem for finalizada
Dependência end-end	Atividade de destino apenas pode ser finalizada quando a atividade de origem for finalizada.
Dependência start-start	Atividade de destino apenas pode ser iniciada quando a atividade de origem for iniciada.

5.3. Notação Gráfica

Esta seção apresenta a notação gráfica elaborada para a visualização das variações representadas pelo metamodelo.

Como o metamodelo pretende ser independente de linguagem de modelagem de processo, é necessária a elaboração de uma notação gráfica específica para cada linguagem utilizada. Neste trabalho, foi elaborada uma notação gráfica que estende a linguagem WebAPSEE-PML (LIMA *et al.*, 2006). A notação gráfica original é apresentada na Tabela 13.

Tabela 13 – Linguagem WebAPSEE-PML - Adaptado de LIMA *et al.* (2006).

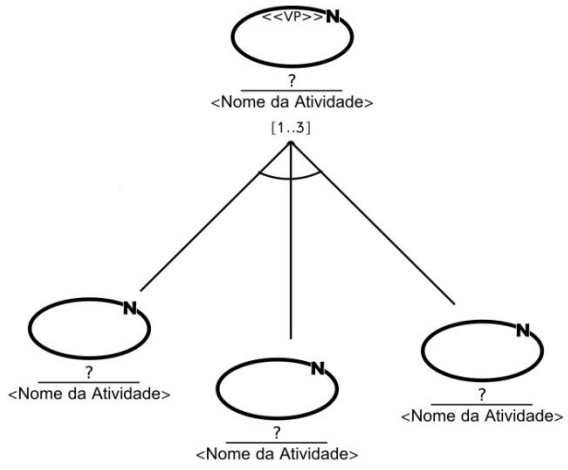
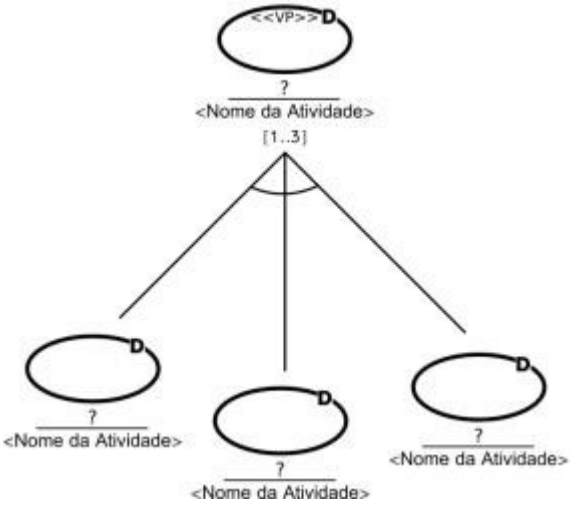
Elemento	Notação Gráfica
Atividade	
Papel	
Grupo	
Artefato	
Recurso	
Conexões	

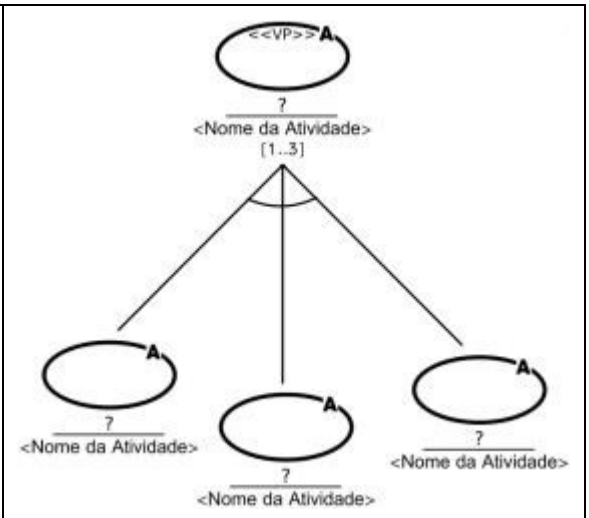
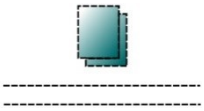
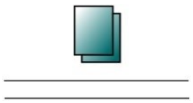
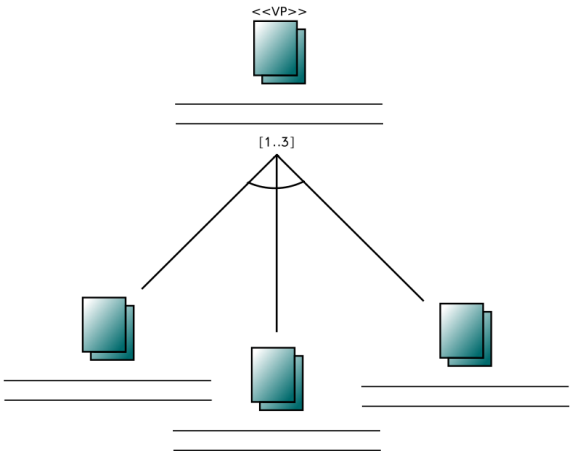
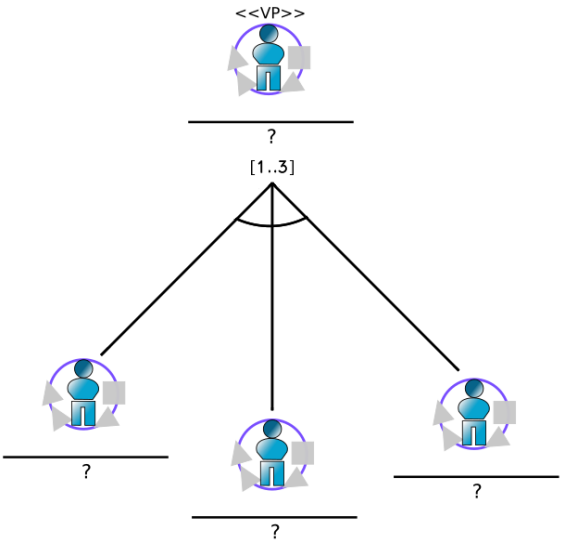
Componentes visuais para a representação de elementos opcionais e obrigatórios, pontos de variação e variantes foram acrescentados à notação gráfica, que foi definida inicialmente em Carvalho *et al.* (2014a) através da utilização dos estereótipos <<Opt>> para designar elementos opcionais e <<VP>> para identificar pontos de variação. Elementos obrigatórios não possuíam estereótipos e não havia notação gráfica para variantes. Além disso, apenas variações em atividades eram representadas.

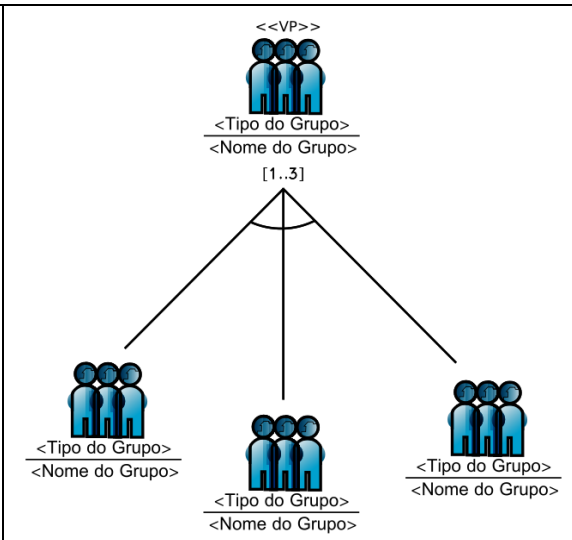
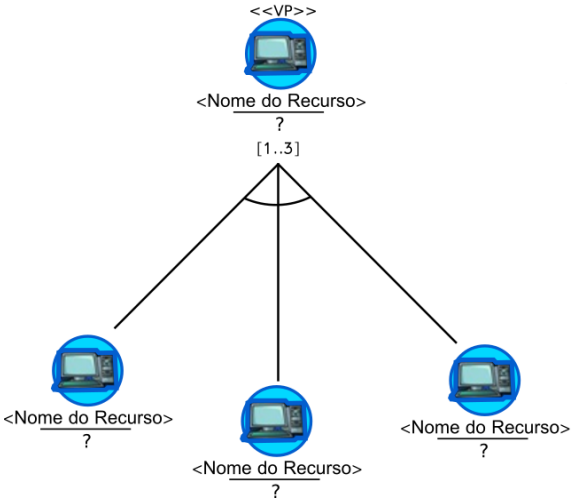
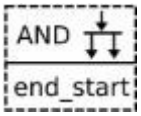
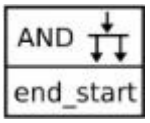
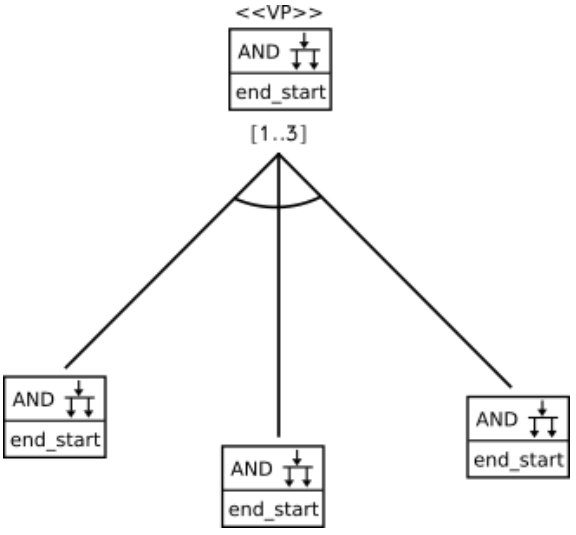
Nesta dissertação, a notação gráfica definida em Carvalho *et al.* (2014a) foi evoluída com a adição e modificação de construtores visuais e a possibilidade de representação visual de variações em um conjunto mais amplo de elementos de processo.

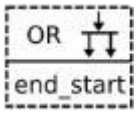
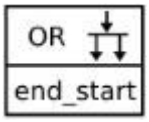
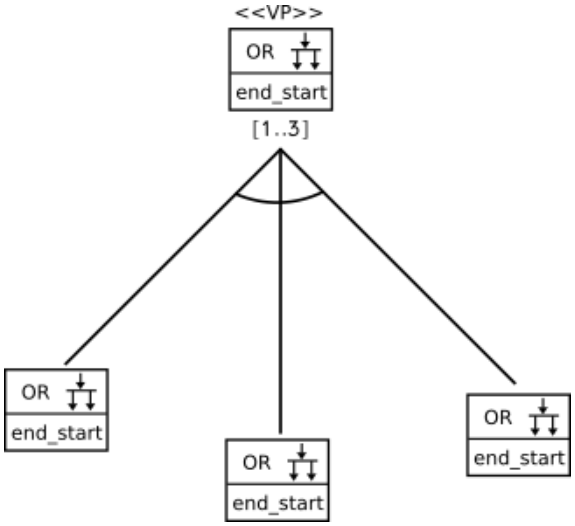
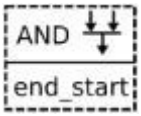
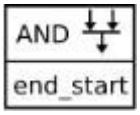
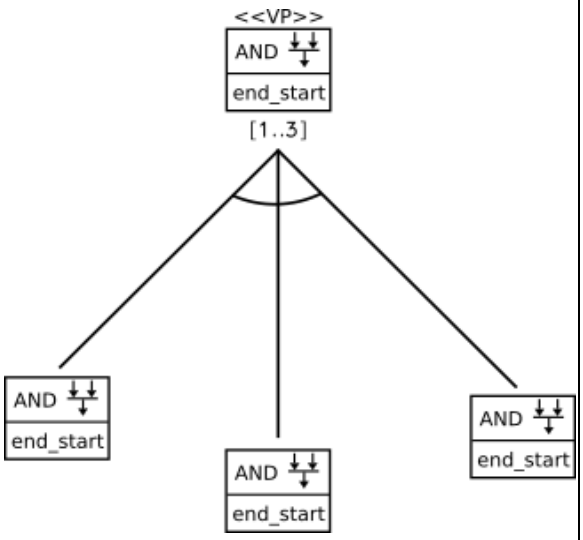
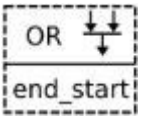
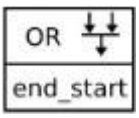
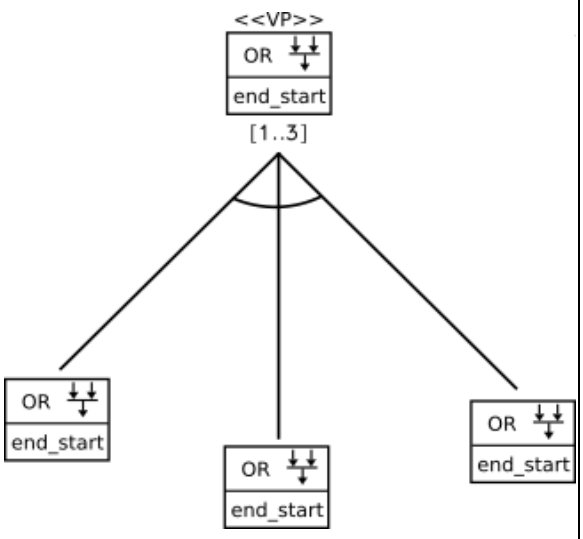
A Tabela 14 apresenta a notação gráfica proposta. Elementos opcionais são desenhados com o contorno tracejado e elementos obrigatórios possuem o contorno contínuo. Para representar pontos de variação, pode-se utilizar a notação resumida ou a notação estendida. Na notação resumida, os variantes não são exibidos explicitamente. Na notação estendida, a relação é representada por linhas, cortadas por um arco sem preenchimento, que ligam o ponto de variação aos variantes. Nesse caso, os variantes são representados graficamente de forma explícita e a cardinalidade mínima e a cardinalidade máxima são especificadas entre colchetes abaixo do ponto de variação. Em ambos os casos, o estereótipo <<VP>> é utilizado para designar que um elemento é um ponto de variação.

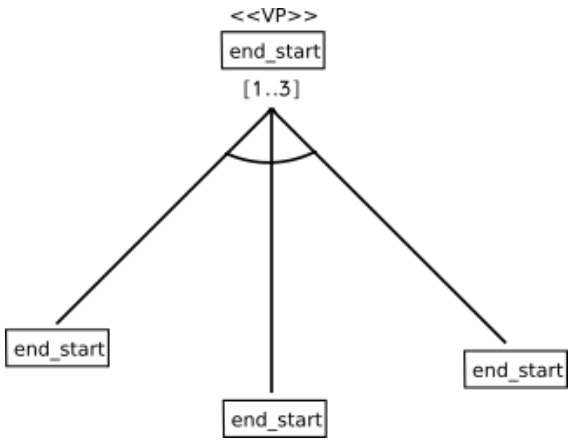
Tabela 14 – Notação Gráfica de Variações em Elementos de Processo.

Elemento/ Variação	Opcional	Obrigatório	Ponto de Variação e Variantes
Atividade Normal			
Atividade Decomposta			

Atividade Automática			
Artefato			
Papel			

Grupo			
Recurso			
Conexão AND Banch			

Conexão OR ***Banch***			
Conexão AND ***Join***			
Conexão OR ***Join***			

Conexão Sequencial	end_start	end_start	

5.4. Processo de Engenharia de Linha de Processos de Software

Esta seção apresenta o processo de engenharia de linha de processos de software recomendado para a utilização do metamodelo e da notação gráfica propostos.

O processo especificado neste trabalho segue a abordagem *top-down*. Ou seja, a modelagem de uma linha de processos de software é realizada a partida da identificação dos requisitos de processo. O processo é composto pelas seguintes fases: Engenharia de Domínio, Engenharia de Aplicação e Gerência da Linha de Processos, que serão abordadas em mais detalhes nas próximas subseções. A Figura 81 apresenta as fases do processo proposto neste trabalho.

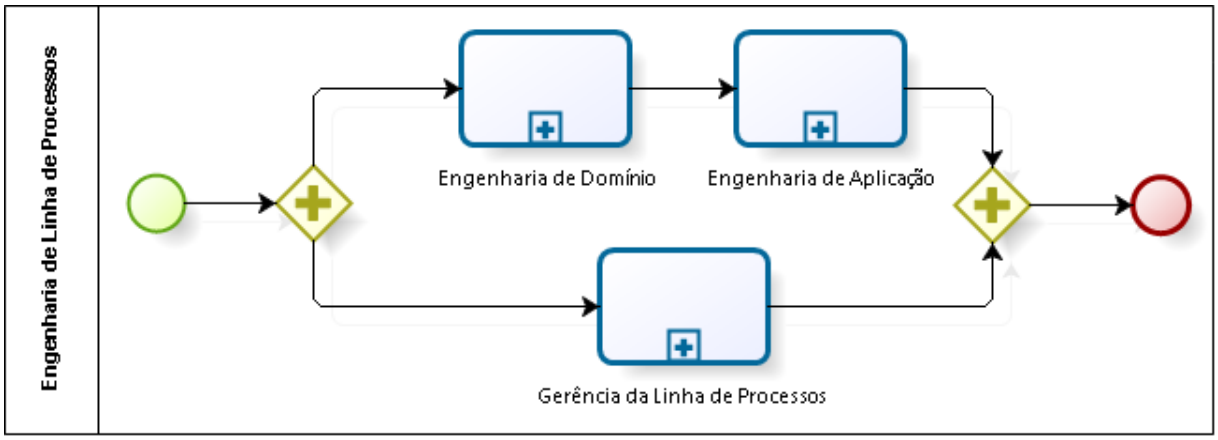


Figura 81 – Processo Proposto de Engenharia de Linha de Processos de Software.

5.4.1. Engenharia de Domínio

A fase de Engenharia de Domínio (Figura 82) consiste na definição de processo para reutilização, gerando ativos reutilizáveis de processo. Essa fase é composta pelas seguintes etapas: modelagem de características, modelagem de elementos de processo, mapeamento entre características e elementos de processo e modelagem de arquitetura de linha de processos.

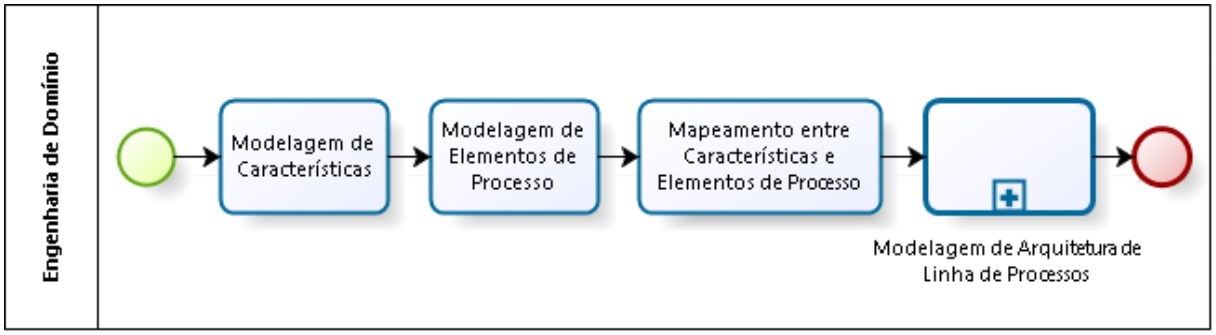


Figura 82 – Processo Proposto de Engenharia de Domínio.

Modelagem de Características: consiste na modelagem de características através de modelos de características, que são armazenados em um repositório. Um modelo de características consiste em uma estrutura em árvore com requisitos de processo. É importante manter a coesão dos modelos de características para que estes possam ser reutilizados de forma eficiente por diversas linhas de processos.

Modelagem de Elementos de Processo: nesta etapa, elementos de processo reutilizáveis são modelados e armazenados em um repositório. Vale ressaltar que os elementos não possuem variações nesta etapa. Ou seja, o metamodelo não define que um elemento de processo é obrigatório, opcional ou um ponto de variação, mas apenas no contexto de uma arquitetura de linha de processo. Além disso, nesta etapa somente são modelados aspectos estruturais, como as atividades, papéis, produtos de trabalho e ferramentas que podem ser reutilizados pelas linhas de processos. Os fluxos entre as atividades ainda não são modelados.

Mapeamento entre Características e Elementos de Processo: nesta etapa, cada elemento de processo é classificado em relação às características no qual possui associações. Esse mapeamento é fundamental para avaliar se o processo atende aos requisitos de alto nível representados pelas características.

Modelagem de Arquitetura de Linha de Processos: nesta etapa, a arquitetura da linha de processos é modelada através das seguintes etapas:

- **Seleção de Elementos de Processo e Características:** esta etapa da modelagem de uma arquitetura de linha de processos consiste na seleção dos elementos de processo e características que serão incluídos.
- **Modelagem de Variabilidades:** nesta etapa, os elementos selecionados para compor a arquitetura de linha de processos são especificados como opcionais ou obrigatórios e os pontos de variação e variantes são definidos;
- **Modelagem de Dependências:** nesta etapa, as dependências entre as características são definidas.

A Figura 83 apresenta o subprocesso de modelagem de arquitetura de linha de processos de software.

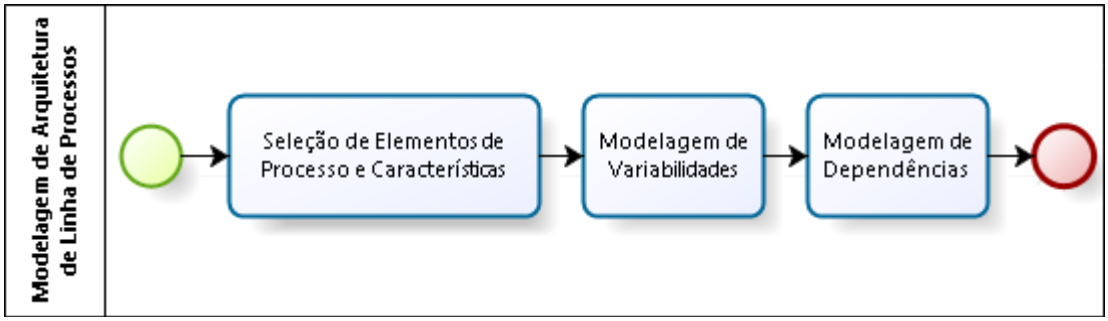


Figura 83 – Subprocesso de Modelagem de Arquitetura de Linha de Processos.

5.4.2. Engenharia de Aplicação

A fase de Engenharia de Aplicação consiste na definição de processo com reutilização, ou seja, os ativos reutilizáveis de processo são instanciados de acordo com um contexto específico.

- **Resolução de Variações:** nesta etapa, a resolução de variações pode ser realizada de forma manual ou através da escolha de um modelo de resolução preexistente. A seleção manual consiste na escolha de que elementos opcionais devem ser incluídos e que variante(s) deve(m) ser selecionado(s) em cada ponto de variação. Nesse caso, todas as escolhas devem ser realizadas manualmente, caso a caso e, no final, é gerado um novo modelo de resolução que poderá ser reutilizado posteriormente. Vale ressaltar que, caso o modelo de resolução seja disponibilizado para reutilização, é importante descrevê-lo detalhadamente para facilitar na sua recuperação e decisão de reutilizá-lo ou não, de acordo com o contexto de cada projeto. Por outro lado, um modelo de resolução previamente existente com escolhas previamente realizadas pode ser selecionado. Nesse caso, os elementos opcionais já são incluídos ou excluídos previamente, assim como os variantes dos pontos de variação. Uma mesma arquitetura de linha de processos pode ter vários modelos de resolução associados, cada um com decisões específicas.
- **Derivação do Processo:** a etapa de derivação consiste na geração do modelo de processo que representa uma instância da linha de processos de software. Para isso, elementos de processo são incluídos no processo de acordo com as escolhas realizadas no modelo de resolução.

5.4.3. Gerência da Linha de Processos

Para que os processos reutilizáveis desenvolvidos durante a engenharia de domínio possam ser reutilizados na engenharia de aplicação, é necessário que estes sejam estejam disponíveis para a busca e recuperação (SOFTEX, 2013). Os processos devem estar armazenados em um repositório, onde são classificados de acordo com o contexto em que são adequados para a utilização. Entretanto, antes do armazenamento de um processo reutilizável, estes precisam ser avaliados em relação à sua qualidade. Essas avaliações são compostas das seguintes etapas: (i) avaliação de aceitação, (ii) avaliação de certificação, (iii) avaliação de utilização e (iv) avaliação de descontinuidade, descritas a seguir com mais detalhes:

- **Avaliação de aceitação:** as arquiteturas de linha de processos definidas na ED precisam ser valiosas para a organização, portanto é necessário o cumprimento de critérios de aceitação para avaliar se o conceito referente ao processo está de acordo com as necessidades da organização e agregará valor ao processo de desenvolvimento de software.
- **Avaliação de certificação:** quando uma arquitetura de linha de processos é aceita, ainda precisa passar pelos critérios de certificação, que avaliam se os serviços propostos pelo ativo são atendidos. Os critérios de certificação avaliam se os conteúdos estão de acordo com a qualidade esperada pela organização.
- **Avaliação de utilização:** quando as arquiteturas de linha de processos são disponibilizadas para a reutilização, existe ainda a possibilidade dos usuários avaliarem a sua utilização.
- **Avaliação de descontinuidade:** por fim, as arquiteturas de linha de processos devem ser continuamente avaliadas em relação a critérios de descontinuidade, que avaliam se os mesmos ainda devem fazer parte do repositório de processos reutilizáveis da organização. Um repositório repleto de processos sem valor para a organização pode dificultar a busca e recuperação de processos mais alinhados com as metas da organização.

5.5. Exemplo de Linha de Processos

Esta seção tem como objetivo apresentar um exemplo da modelagem de uma linha de processos de software com o propósito demonstrar a viabilidade de utilizar o metamodelo, a notação gráfica e o processo propostos nesta dissertação de mestrado. Para isso, será apresentada uma linha de processos definida por Hurtado *et al.* (2013), que foi publicada no periódico "The

Journal of Systems and Software", classificado no extrato A2 no sistema Qualis Ciência da Computação, da Coordenadoria de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), do Ministério da Educação (MEC). Essa abordagem utiliza o metamodelo Experimental Software & System Engineering Metamodel (eSPEM) e foi aplicada na prática para a modelagem de linha de processos na empresa Amisoft, avaliada no nível 2 do CMMI-DEV. Os autores do artigo de Hurtado *et al.* (2013) já haviam publicado uma série de artigos na área de linha de processos de software (ALEGRÍA *et al.*, 2011; ALEGRÍA; BASTARRICA, 2012; SIMMONDS *et al.*, 2013), evidenciando que os mesmos são especialistas na área.

Vale ressaltar que a modelagem da linha de processos nesta seção não tem como objetivo justificar os elementos de processo e as relações entre eles. Por exemplo, não é um objetivo a especificação dos papéis, grupos, ferramentas e produtos de trabalho de cada atividade de forma a manter conformidade com um modelo de maturidade, norma, guia, modelo ou *framework* de processo específico.

5.5.1. Linha de Processos para Engenharia de Requisitos de Hurtado *et al.* (2013)

Hurtado *et al.* (2013) definiram uma linha de processos para o processo de Engenharia de Requisitos, composto pelos processos de Desenvolvimento de Requisitos ("Requirements Development") e Gerência de Requisitos ("Requirements Management") (Figura 84). Para o propósito deste exemplo, somente o processo de gerência de requisitos será detalhado nesta seção. A linha de processos foi modelada com a utilização da linguagem de modelagem de processos *Experimental Software Process Engineering Metamodel* (eSPEM).

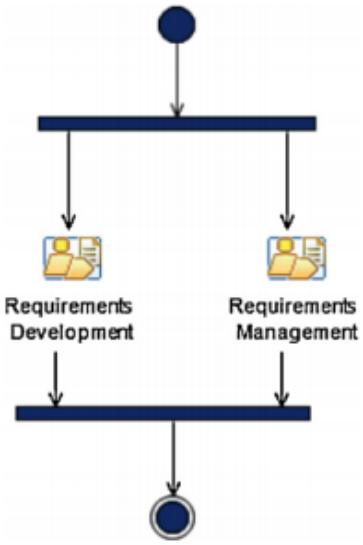


Figura 84 – Linha de Processos para Engenharia de Requisitos (HURTADO *et al.*, 2013).

A notação gráfica da abordagem de Hurtado *et al.* (2013) representa variações através de modelos de características e do próprio modelo de processo na linguagem eSPEM. As variações em um modelo de processo identificam atividades opcionais envolvendo-as por um retângulo com um fundo amarelo e cantos arredondados. Atividades obrigatórias não possuem esse fundo.

A Figura 85 apresenta o processo de gerência de requisitos. Esse processo é composto por quatro atividades obrigatórias ("Requirements Understanding", "Requirements Commitment", "Requirements Tracking" e "Requirements Change Management").

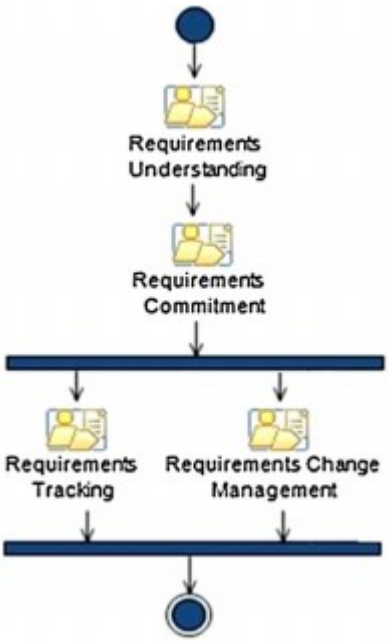


Figura 85 – Processo de Gerência de Requisitos (HURTADO *et al.*, 2013).

A atividade "Requirements Understanding" é composta por uma tarefa opcional ("Identify Requirements Providers") e duas tarefas obrigatórias ("Requirements Review" e "Ensuring Common Requirements Understanding"). A Figura 86 apresenta as tarefas da atividade "Requirements Understanding".

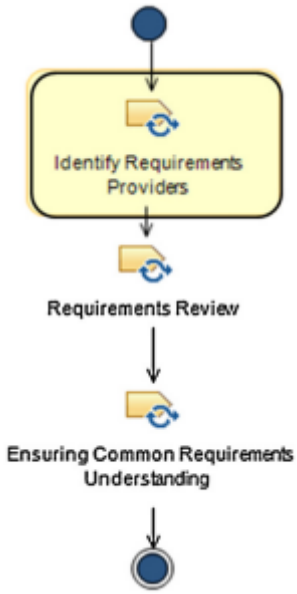


Figura 86 – Atividade "Requirements Understanding" (HURTADO *et al.*, 2013).

A Figura 87 apresenta o modelo de características referente ao processo de gerência de requisitos. Tal modelo de características representa variações do tipo opcional e obrigatória. Características opcionais são identificadas por um círculo com contorno azul e fundo branco. Por outro lado, características obrigatórias são identificadas por um círculo com contorno e fundo azuis. Além de variações, é possível representar dependências ("constraints"). Entretanto, nenhuma dependência foi definida nesses exemplos específicos.

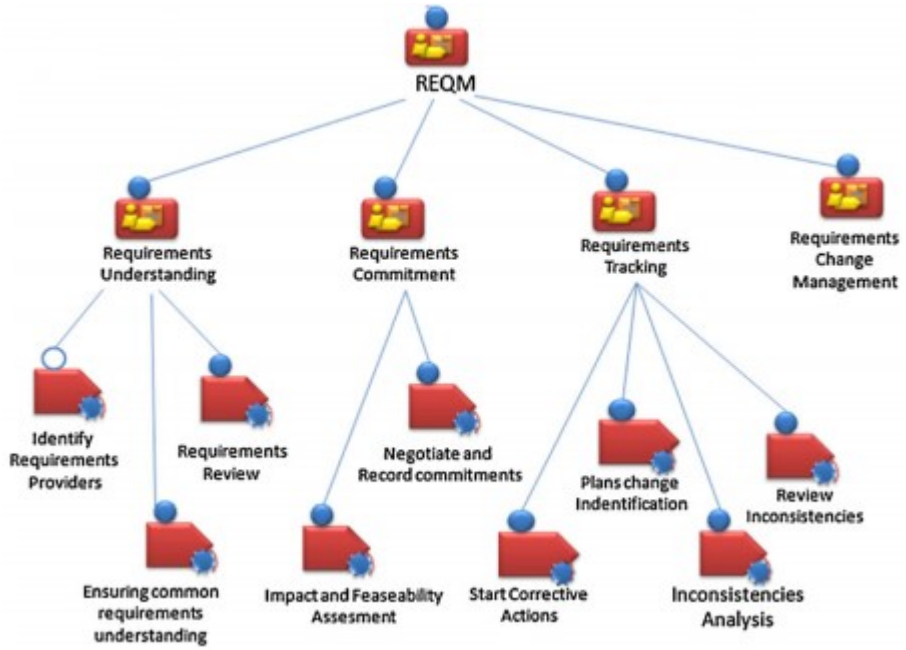


Figura 87 – Modelo de Características para o Processo Gerência de Requisitos (HURTADO *et al.*, 2013).

Antes da modelagem da linha de processos de engenharia de requisitos de Hurtado *et al.* (2013) com a utilização do metamodelo, da notação gráfica e do processo propostos neste

trabalho, foi realizado um mapeamento entre os conceitos presentes nas duas abordagens (Tabela 15). O símbolo ✖ indica que os autores não informaram um conceito equivalente na publicação.

Tabela 15 – Mapeamento entre os Conceitos.

Conceitos	
Este Metamodelo	**Metamodelo de Hurtado *et al.* (2013)**
Decomposed Activity	Activity
Normal Activity	TaskDefinition, TaskUse
Feature	Feature
Dependency	Constraint
Artifact	WordProductDefinition, WorkProductUse
Role	RoleDefinition, RoleUse
Group	✖
Resource	✖

Apesar de definir no metamodelo os conceitos de "WordProductDefinition", "WorkProductUse", "RoleDefinition", "RoleUse" e "Constraint", a parte da linha de processos definida por Hurtado *et al.* (2013) e apresentada neste exemplo não fez uso de tais conceitos.

Um conflito de conceitos encontrado se refere à definição de dependência. Uma dependência ("dependency") no metamodelo proposto neste trabalho se refere a duas instâncias de elementos de processo, enquanto que o conceito equivalente ("constraint") no metamodelo de Hurtado *et al.* (2013) representa uma associação entre duas características.

Para a modelagem da linha de processos de exemplo, o processo de engenharia de domínio especificado na seção 5.4 foi seguido, contendo as seguintes etapas:

- Engenharia de Domínio:
  - Modelagem de Características;
  - Modelagem de Elementos de Processo;
  - Mapeamento entre Características e Elementos de Processo;
  - Modelagem de Arquitetura de Linha de Processos.

As fases de engenharia de aplicação e de gerência da linha de processos não estão no escopo deste exemplo.

5.5.2. Engenharia de Domínio para a Linha de Processos de Engenharia de Requisitos

Nesta etapa, a linha de processos de exemplo foi modelada através do processo de engenharia de domínio. Os conceitos foram modelados em inglês para manter conformidade com a linha de processos original.

5.5.2.1. Modelagem de Características

No metamodelo proposto neste trabalho, um modelo de características é apenas uma estrutura contendo uma árvore de características, sem a especificação de variações e dependências entre as mesmas. As variações e dependências são especificadas somente nos elementos de processo. A Figura 88 apresenta o modelo de características[8] do processo de gerência de requisitos. Esse modelo de características possui as mesmas estruturas do modelo de características definido por Hurtado *et al.* (2013), apresentado na Figura 87.

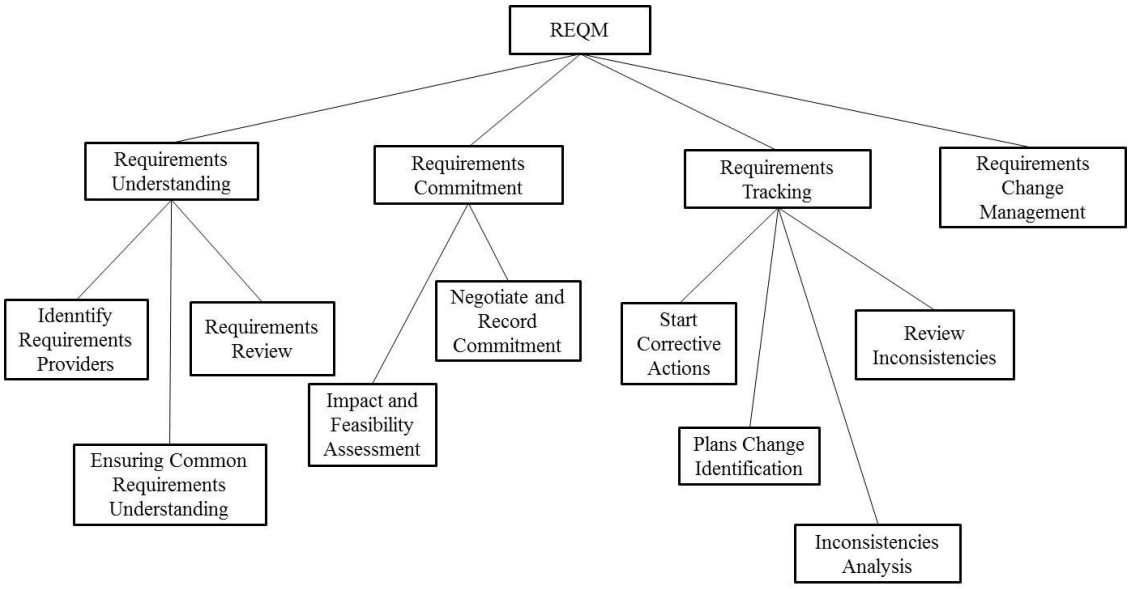


Figura 88 – Modelo de Características de Gerência de Requisitos de Exemplo.

5.5.2.2. Modelagem de Elementos de Processo

Nessa etapa, foram definidos os elementos de processo que comporão a arquitetura de linha de processos. A linha de processos de Hurtado *et al.* (2013) definiu somente atividades e tarefas. Entretanto, neste exemplo serão definidos elementos de processo dos tipos atividade, papel, grupo, ferramenta e produto de trabalho do processo de gerência de requisitos.

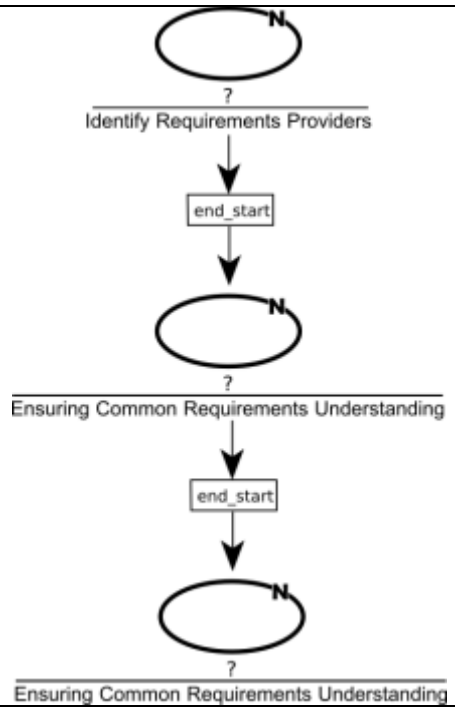
Atividades: todas as atividades de gerência de requisitos modeladas neste exemplo já estavam presentes na linha de processos original definida por Hurtado *et* al. (2013). Foram modeladas as seguintes atividades decompostas: "Requirements Understanding", "Requirements Commitment", "Requirements Tracking" e "Requirements Change Management". A atividade "Requirements Understanding" é composta pelas seguintes atividades normais: "Identify Requirements Providers", "Ensuring Common Requirements Understanding" e "Requirements

[8] O nome da característica "Impact and Feaseability Assesment" do modelo de características de Hurtado *et al.* (2013) foi modificado para "Impact and Feasibility Assessment" por motivo de adequação ortográfica da língua inglesa.

Review". A atividade "Requirements Commitment" é composta pelas seguintes atividades normais: "Impact and Feasibility Assessment" e "Negotiate and Record Commitment". A atividade "Requirements Tracking" é composta pelas seguintes atividades normais: "Start Corrective Actions", "Plans Change Identification", "Inconsistencies Analysis" e "Review Inconsistencies". Por fim, a atividade "Requirements Change Management" também foi definida como decomposta, entretanto não possui subatividades.

Para a modelagem das atividades, foram utilizados os formulários de cadastro de atividades apresentados na seção B.5 do APÊNDICE B. Entretanto, os campos "Descrição", "Critérios de Entrada", "Critérios de Saída", "Participantes" e "Status" não foram preenchidos por não serem relevantes para o propósito do exemplo.

Tabela 16 – Atividade Decomposta "Requirements Understanding".

Requirements Understanding	
Nome	Requirements Understanding
Descrição	-
Tipo	Decomposta
Critérios de Entrada	-
Critérios de Saída	-
Responsável	Requirements Responsible, Product Owner, Requirements Analyst
Participantes	-
Produtos de Trabalho de Entrada	Requirements Document, Use Case Document, Product Backlog
Produtos de Trabalho de Saída	Requirements Document, Use Case Document, Product Backlog
Ferramentas	Text Processor
Diagrama	

Características Associadas	REQM, • Requirements Understanding
Status	-

Tabela 17 – Atividade Normal "Identify Requirements Providers".

Identify Requirements Providers	
Nome	Identify Requirements Providers
Descrição	-
Tipo	Normal
Critérios de Entrada	-
Critérios de Saída	-
Responsável	Requirements Analyst
Participantes	-
Produtos de Trabalho de Entrada	Requirements Document, Use Case Document, Product Backlog
Produtos de Trabalho de Saída	Requirements Document, Use Case Document, Product Backlog
Ferramentas	Text Processor
Características Associadas	REQM, • Requirements Understanding, o Identify Requirements Providers
Status	-

Tabela 18 – Atividade Normal "Ensuring Common Requirements Understanding".

Ensuring Common Requirements Understanding	
Nome	Ensuring Common Requirements Understanding
Descrição	-
Tipo	Normal
Critérios de Entrada	-
Critérios de Saída	-
Responsável	Requirements Responsible, Product Owner, Requirements Analyst
Participantes	-
Produtos de Trabalho de Entrada	Requirements Document, Use Case Document, Product Backlog
Produtos de Trabalho de Saída	Requirements Document, Use Case Document, Product Backlog
Ferramentas	Text Processor
Características Associadas	REQM, • Requirements Understanding, o Ensuring Common Requirements Understanding
Status	-

Tabela 19 – Atividade Normal "Requirements Review".

Requirements Review	
Nome	Requirements Review
Descrição	-

Tipo	Normal
Critérios de Entrada	-
Critérios de Saída	-
Responsável	Reviewer
Participantes	-
Produtos de Trabalho de Entrada	Requirements Document, Use Case Document, Product Backlog
Produtos de Trabalho de Saída	Requirements Review
Ferramentas	Text Processor
Características Associadas	REQM, • Requirements Understanding, ○ Requirements Review
Status	-

Tabela 20 – Atividade Decomposta "Requirements Commitment".

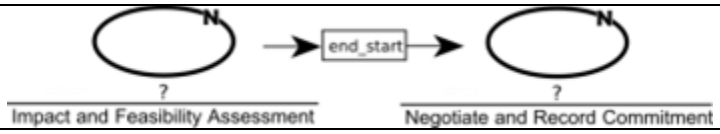
Requirements Commitment	
Nome	Requirements Commitment
Descrição	-
Tipo	Decomposta
Critérios de Entrada	-
Critérios de Saída	-
Responsável	Project Team
Participantes	-
Produtos de Trabalho de Entrada	-
Produtos de Trabalho de Saída	-
Ferramentas	Text Processor
Diagrama	
Características Associadas	REQM, • Requirements Commitment
Status	-

Tabela 21 – Atividade Normal "Impact and Feasibility Assessment".

Impact and Feasibility Assessment	
Nome	Impact and Feasibility Assessment
Descrição	-
Tipo	Normal
Critérios de Entrada	-
Critérios de Saída	-
Responsável	Project Team
Participantes	-

Produtos de Trabalho de Entrada	Project Plan, Requirements Document, Use Case Document, Product Backlog
Produtos de Trabalho de Saída	Impact Analysis Document
Ferramentas	Text Processor
Características Associadas	REQM, • Requirements Commitment, o Impact and Feasibility Assessment
Status	-

Tabela 22 – Atividade Normal "Negotiate and Record Commitment".

Negotiate and Record Commitment	
Nome	Negotiate and Record Commitment
Descrição	-
Tipo	Normal
Critérios de Entrada	-
Critérios de Saída	-
Responsável	Project Leader, Project Manager
Participantes	-
Produtos de Trabalho de Entrada	Requirements Document, Use Case Document, Product Backlog
Produtos de Trabalho de Saída	Requirements Document, Use Case Document, Product Backlog
Ferramentas	Text Processor
Características Associadas	REQM, • Requirements Commitment, o Negotiate and Record Commitment
Status	-

Tabela 23 – Atividade Decomposta "Requirements Tracking".

Requirements Tracking	
Nome	Requirements Tracking
Descrição	-
Tipo	Decomposta
Critérios de Entrada	-
Critérios de Saída	-
Responsável	Requirements Responsible, Product Owner, Requirements Analyst
Participantes	-
Produtos de Trabalho de Entrada	Requirements Document, Use Case Document, Product Backlog
Produtos de Trabalho de Saída	Traceability Matrix
Ferramentas	Text Processor
Diagrama	

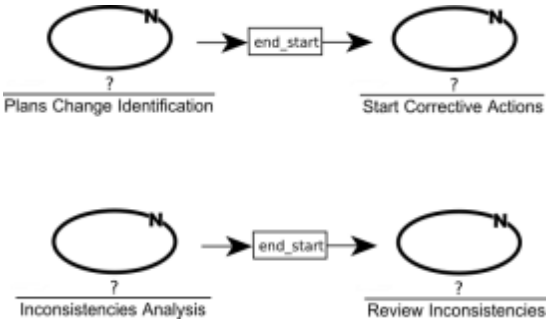
	
Características Associadas	REQM, • Requirements Tracking
Status	-

Tabela 24 – Atividade Normal "Start Corrective Actions".

Start Corrective Actions	
Nome	Start Corrective Actions
Descrição	-
Tipo	Normal
Critérios de Entrada	-
Critérios de Saída	-
Responsável	Requirement Responsible
Participantes	-
Produtos de Trabalho de Entrada	Requirement Specification, Use Case Document, Product Backlog
Produtos de Trabalho de Saída	Requirement Specification, Use Case Document, Product Backlog
Ferramentas	Text Processor
Características Associadas	REQM, • Requirements Tracking, o Start Corrective Actions
Status	-

Tabela 25 – Atividade Normal "Plans Change Identification".

Plans Change Identification	
Nome	Plans Change Identification
Descrição	-
Tipo	Normal
Critérios de Entrada	-
Critérios de Saída	-
Responsável	Project Leader, Project Manager
Participantes	-
Produtos de Trabalho de Entrada	Requirements Document, Use Case Document, Product Backlog
Produtos de Trabalho de Saída	Requirements Document, Use Case Document, Product Backlog
Ferramentas	Text Processor

Características Associadas	REQM, • Requirements Tracking, o Plans Change Identification
Status	-

Tabela 26 – Atividade Normal "Inconsistencies Analysis".

Inconsistencies Analysis	
Nome	Inconsistencies Analysis
Descrição	-
Tipo	Normal
Critérios de Entrada	-
Critérios de Saída	-
Responsável	Requirement Responsible, Product Owner, Requirements Analyst
Participantes	-
Produtos de Trabalho de Entrada	Requirements Document, Use Case Document, Product Backlog
Produtos de Trabalho de Saída	Requirements Document, Use Case Document, Product Backlog
Ferramentas	Text Processor
Características Associadas	REQM, • Requirements Tracking, o Inconsistencies Analysis
Status	-

Tabela 27 – Atividade Normal "Review Inconsistencies".

Review Inconsistencies	
Nome	Review Inconsistencies
Descrição	-
Tipo	Normal
Critérios de Entrada	-
Critérios de Saída	-
Responsável	Reviewer
Participantes	-
Produtos de Trabalho de Entrada	Requirement Specification, Use Case Document, Product Backlog
Produtos de Trabalho de Saída	Requirement Specification, Use Case Document, Product Backlog
Ferramentas	Text Processor
Características Associadas	REQM, • Requirements Tracking, o Review Inconsistencies
Status	-

Tabela 28 – Atividade Decomposta "Requirements Change Management".

Requirements Change Management	
Nome	Requirements Change Management

Descrição	-
Tipo	Decomposta
Critérios de Entrada	-
Critérios de Saída	-
Responsável	Project Team
Participantes	-
Produtos de Trabalho de Entrada	-
Produtos de Trabalho de Saída	-
Ferramentas	Text Processor
Diagrama	
Características Associadas	REQM, • Requirements Change Management
Status	-

Produtos de Trabalho: a linha de processos original definida por Hurtado *et* al. (2013) não considerou os produtos de trabalho para cada atividade/tarefa, portanto os produtos de trabalho modelados neste exemplo são elementos adicionais em relação à linha de processo original. Foram modelados os seguintes produtos de trabalho: "Requirements Document", "Requirements Specification", "Use Case Document", "Product Backlog", "Impact Analysis Document", "Project Plan" e "Traceability Matrix".

Para a modelagem dos produtos de trabalho, foram utilizados os formulários de cadastro apresentados na seção B.8 do APÊNDICE B. Entretanto, os campos "Descrição" e "Status" não foram preenchidos por não serem relevantes para o propósito do exemplo.

Tabela 29 – Produto de Trabalho "Requirements Document".

Requirements Document	
Nome	Requirements Document
Descrição	-

Características Associadas	REQM,
	• Requirements Understanding,
	  o Identify Requirements Providers,
	  o Ensuring Common Requirements Understanding,
	• Requirements Commitment,
	  o Impact and Feasibility Assessment,
	  o Negotiate and Record Commitment,
	• Requirements Tracking,
	  o Plans Change Identification,
	  o Inconsistencies Analysis

Status	-

Tabela 30 – Produto de Trabalho "Requirements Review".

Requirements Review	
Nome	Requirements Review
Descrição	-
Características Associadas	REQM, • Requirements Understanding, o Requirements Review
Status	-

Tabela 31 – Produto de Trabalho "Requirements Specification".

Requirements Specification	
Nome	Requirements Specification
Descrição	-
Características Associadas	REQM, • Requirements Tracking, o Start Corrective Actions o Review Inconsistencies
Status	-

Tabela 32 – Produto de Trabalho "Use Case Document".

Use Case Document	
Nome	Use Case Document
Descrição	-
Características Associadas	REQM, • Requirements Understanding, o Identify Requirements Providers, o Ensuring Common Requirements Understanding, • Requirements Commitment, o Impact and Feasibility Assessment, o Negotiate and Record Commitment, • Requirements Tracking, o Plans Change Identification, o Inconsistencies Analysis
Status	-

Tabela 33 – Produto de Trabalho "Product Backlog".

Product Backlog	
Nome	Product Backlog
Descrição	-
Características Associadas	REQM, • Requirements Understanding, o Identify Requirements Providers, o Ensuring Common Requirements Understanding, • Requirements Commitment, o Impact and Feasibility Assessment, o Negotiate and Record Commitment, • Requirements Tracking, o Plans Change Identification, o Inconsistencies Analysis

Status	-

Tabela 34 – Produto de Trabalho "Impact Analysis Document".

Impact Analysis Document	
Nome	Impact Analysis Document
Descrição	-
Características Associadas	REQM, • Requirements Commitment, o Impact and Feasibility Assessment
Status	-

Tabela 35 – Produto de Trabalho "Project Plan".

Project Plan	
Nome	Project Plan
Descrição	-
Características Associadas	-
Status	-

Tabela 36 – Produto de Trabalho "Traceability Matrix".

Traceability Matrix	
Nome	Traceability Matrix
Descrição	-
Características Associadas	REQM, • Requirements Tracking
Status	-

Papéis: a linha de processos original definida por Hurtado *et* al. (2013) não considerou os papéis para cada atividade/tarefa, portanto esses elementos são considerados adicionais em relação à linha de processos original. Foram modelados os seguintes papéis: "Project Leader", "Scrum Master", "Project Manager", "Requirements Responsible", "Product Owner", "Requirements Analyst" e "Reviewer".

Para a modelagem dos papéis, foram utilizados os formulários de cadastro apresentados na seção B.6 do APÊNDICE B. Entretanto, os campos "Descrição" e "Status" não foram preenchidos por não serem relevantes para o propósito do exemplo.

Tabela 37 – Papel "Project Leader".

Project Leader	
Nome	Project Leader
Descrição	-
Características Associadas	REQM, • Requirements Commitment, o Negotiate and Record Commitment • Requirements Tracking, o Plans Change Identification
Status	-

Tabela 38 – Papel "Project Manager".

Project Manager	
Nome	Project Manager
Descrição	-

Características Associadas	REQM,
	• Requirements Commitment,
	  o Negotiate and Record Commitment
	• Requirements Tracking,
	  o Plans Change Identification

Status	-

Tabela 39 – Papel "Requirements Responsible".

Requirements Responsible	
Nome	Requirements Responsible
Descrição	-
Características Associadas	REQM, • Requirements Understanding o Ensuring Common Requirements Understanding • Requirements Tracking
Status	-

Tabela 40 – Papel "Product Owner".

Scrum Master	
Nome	Product Owner
Descrição	-

Características Associadas	REQM,
	• Requirements Understanding
	  o Ensuring Common Requirements Understanding
	• Requirements Tracking
	  o Inconsistencies Analysis

Status	-

Tabela 41 – Papel "Requirements Analyst".

Scrum Master	
Nome	Requirements Analyst
Descrição	-

Características Associadas	REQM,
	• Requirements Understanding
	  o Identify Requirements Providers
	  o Ensuring Common Requirements Understanding
	• Requirements Tracking
	  o Inconsistencies Analysis

Status	-

Tabela 42 – Papel "Reviewer".

Reviewer	
Nome	Reviewer
Descrição	-
Características	REQM,

Associadas	• Requirements Understanding,
	  o Requirements Review
	• Requirements Tracking,
	  o Review Inconsistencies

Status	-

Grupos: a linha de processos original definida por Hurtado *et* al. (2013) não considerou os grupos para cada atividade/tarefa, portanto esses elementos foram adicionados. Foi modelado o seguinte grupo: "Project Team".

Para a modelagem do grupo, foi utilizado o formulário de cadastro apresentado na seção B.7 do APÊNDICE B. Entretanto, os campos "Descrição" e "Status" não foram preenchidos por não serem relevantes para o propósito do exemplo.

Tabela 43 – Grupo "Project Team".

Project Team	
Nome	Project Team
Descrição	-
Papéis	Project Leader, Scrum Master, Project Manager, Requirements Responsible, Product Owner, Requirements Analyst e Reviewer
Características Associadas	REQM, • Requirements Commitment o Impact and Feasibility Assessment REQM, • Requirements Change Management
Status	-

Ferramentas: a linha de processos original definida por Hurtado *et* al. (2013) não considerou as ferramentas para a execução das atividades/tarefas, portanto esses elementos são considerados adicionais em relação à linha de processos original. Foi modelada a seguinte ferramenta: "Text Processor".

Para a modelagem da ferramenta, foi utilizado o formulário de cadastro apresentado na seção B.9 do APÊNDICE B. Entretanto, os campos "Descrição" e "Status" não foram preenchidos por não serem relevantes para o propósito do exemplo.

Tabela 44 – Ferramenta "Text Processor".

Text Processor	
Nome	Text Processor
Descrição	-

Características Associadas	REQM,
	• Requirements Understanding,
	  o Identify Requirements Providers
	  o Ensuring Common Requirements Understanding
	  o Requirements Review

	• Requirements Commitment
	  o Impact and Feasibility Assessment
	  o Negotiate and Record Commitment
	• Requirements Tracking,
	  o Start Corrective Actions
	  o Plans Change Identification
	  o Inconsistencies Analysis
	  o Review Inconsistencies
	• Requirements Change Management

Status	-

5.5.2.3. Mapeamento entre Características e Elementos de Processo

Nesta etapa, foram especificadas as características associadas a cada elemento de processo que compõem a arquitetura de linha de processos. As características associadas a cada elemento de processos foram preenchidas no campo "Características Associadas" dos formulários apresentados na seção 5.5.2.2. Para facilitar a visualização da hierarquia de características, as mesmas foram preenchidas em listas de vários níveis.

Os seguintes critérios foram utilizados para associar as características a cada um dos elementos de processo:

- Para as atividades, foram associadas as características correspondentes no modelo de características e todas as características superiores. Por exemplo, a atividade "Identify Requirements Providers" foi associada à característica de mesmo nome "Identify Requirements Providers" e às características superiores, "Requirements Understanding" e "REQM";
- Para os produtos de trabalho, foram associadas as mesmas características das atividades nas quais tais produtos de trabalho foram definidos como saída. Os produtos de trabalho de entrada foram desconsiderados por representar uma relação mais fraca com as atividades;
- Para os papéis e grupos, foram associadas as mesmas características das atividades nas quais tais papéis/grupos foram definidos como responsáveis. Os papéis/grupos definidos somente como participantes foram desconsiderados por representar uma relação mais fraca com as atividades;
- Para as ferramentas, foram associadas as mesmas características das atividades nas quais tais ferramentas foram definidas como apoio.

5.5.2.4. Modelagem de Arquitetura de Linha de Processos

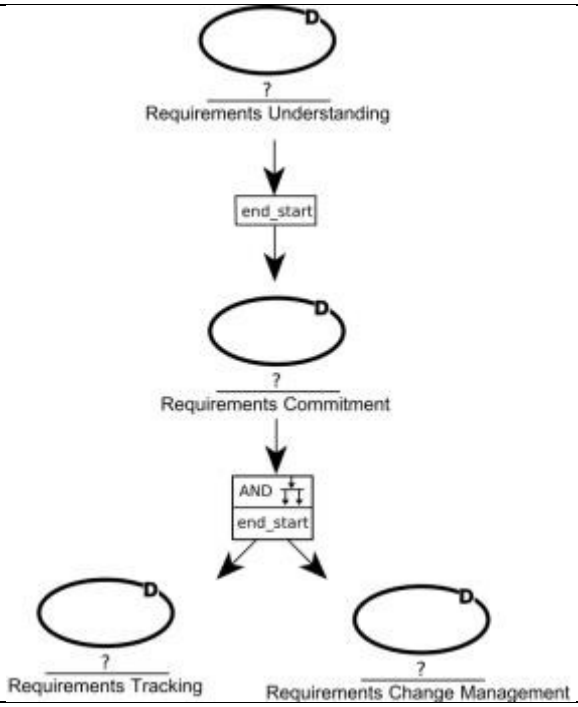
Nesta etapa, foram selecionados todos os elementos de processo modelados na seção 5.5.2.2 para compor a Arquitetura de Linha de Processos de Software (ALPrS). Além disso, foram definidos os fluxos entre as atividades e as variações e dependências entre os elementos de processo.

Na abordagem proposta neste trabalho, as variações e dependências se referem aos elementos de processos em uma determinada ALPrS. Ou seja, os elementos não possuem intrinsecamente variações, mas somente quando são inseridos em uma ALPrS. Além disso, essa abordagem permite que diferentes ALPrSs definam variações e dependências diferentes entre os mesmos elementos de processo. A Tabela 45 apresenta a ALPrS de exemplo. Os campos "Descrição" e "Status" não foram preenchidos por não serem relevantes para o propósito do exemplo.

Tabela 45 – Arquitetura de Linha de Processos de Gerência de Requisitos.

Requirements Management Software Process Line Architecture	
Nome	Requirements Management Software Process Line Architecture
Descrição	-

Características Associadas	REQM,
	• Requirements Understanding,
	  o Identify Requirements Providers
	  o Ensuring Common Requirements Understanding
	  o Requirements Review
	• Requirements Commitment
	  o Impact and Feasibility Assessment
	  o Negotiate and Record Commitment
	• Requirements Tracking,
	  o Start Corrective Actions
	  o Plans Change Identification
	  o Inconsistencies Analysis
	  o Review Inconsistencies
	Requirements Change Management

Diagrama	
Dependências	-
Status	-

A Figura 89 apresenta as subatividades da atividade "Requirements Understanding". A atividade normal "Identify Requirements Providers" foi definida como opcional no contexto da ALPrS. As demais atividades ("Requirements Review" e "Ensure Common Requirements Understanding") foram definidas como obrigatórias.

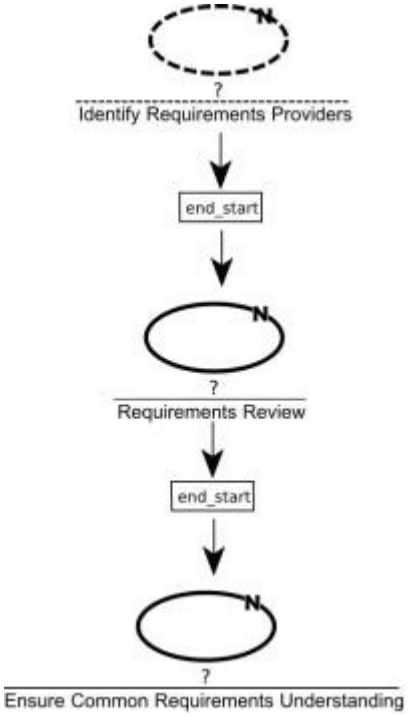


Figura 89 – Requirements Understanding.

Para facilitar a visualização, a Figura 89 não apresenta os papéis, grupos, produtos de trabalho e ferramentas das atividades. Esses elementos serão apresentados isoladamente para cada atividade.

A Figura 90 apresenta a atividade opcional "Identify Requirements Providers". Essa atividade é executada pelo papel "Requirements Analyst", obrigatório; com o apoio da ferramenta "Text Processor", obrigatória; e possui o produto de trabalho "Requirements Document" como entrada e saída, que é um ponto de variação obrigatório com dois possíveis variantes, "Use Case Document" e "Product Backlog". O ponto de variação "Requirements Document" possui cardinalidade mínima e máxima iguais a 1, portanto somente um dos variantes pode ser escolhido.

Vale ressaltar que o papel, os produtos de trabalho e a ferramenta são obrigatórios somente no contexto da atividade. Assim, caso a atividade opcional "Identify Requirements Providers" não seja selecionada para compor o processo gerado, esses elementos também não serão adicionados. Entretanto, caso a atividade seja selecionada, esses elementos devem obrigatoriamente estar conectados à atividade.

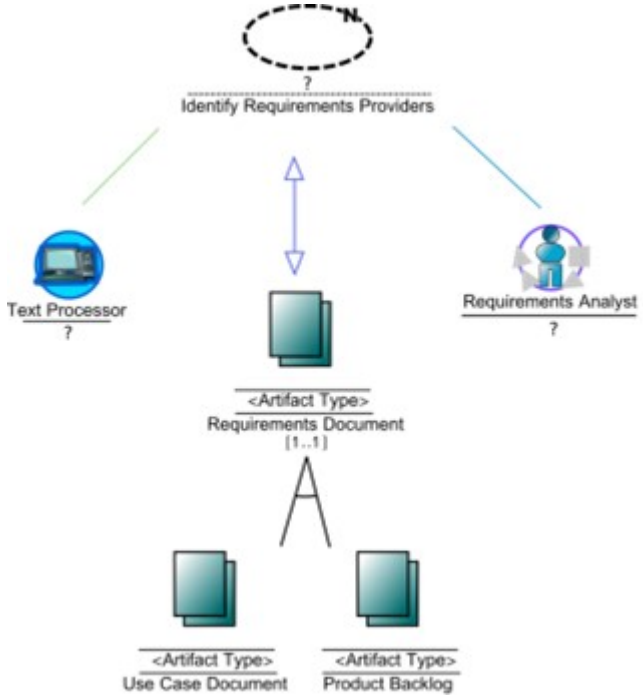


Figura 90 – Atividade "Identify Requirements Providers".

A Figura 91 apresenta a atividade obrigatória "Requirements Review". Essa atividade é executada pelo papel "Reviwer", obrigatório; com o apoio da ferramenta "Text Processor", obrigatória; e possui o produto de trabalho "Requirements Document" como entrada e saída, que é um ponto de variação obrigatório com cardinalidade máxima e mínima iguais a 1 e com dois possíveis variantes, "Use Case Document" e "Product Backlog".

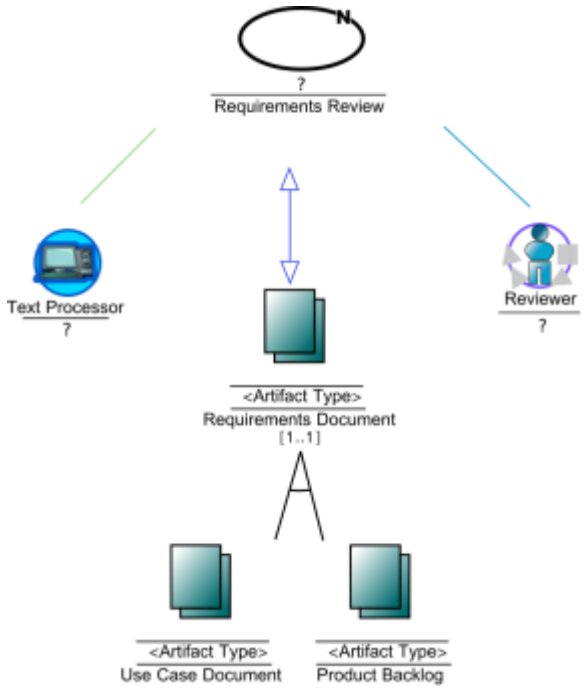


Figura 91 – Atividade "Requirements Review".

A Figura 92 apresenta a atividade obrigatória "Ensuring Common Requirements Understanding". Essa atividade é executada pelo papel "Requirements Responsible", que é um ponto de variação obrigatório com dois possíveis variantes, "Product Owner" e "Requirements Analyst"; com o apoio da ferramenta "Text Processor", obrigatória; e possui o produto de trabalho "Requirements Document" como entrada e saída, que é um ponto de variação obrigatório com cardinalidade máxima e mínima iguais a 1 e com dois possíveis variantes, "Use Case Document" e "Product Backlog".

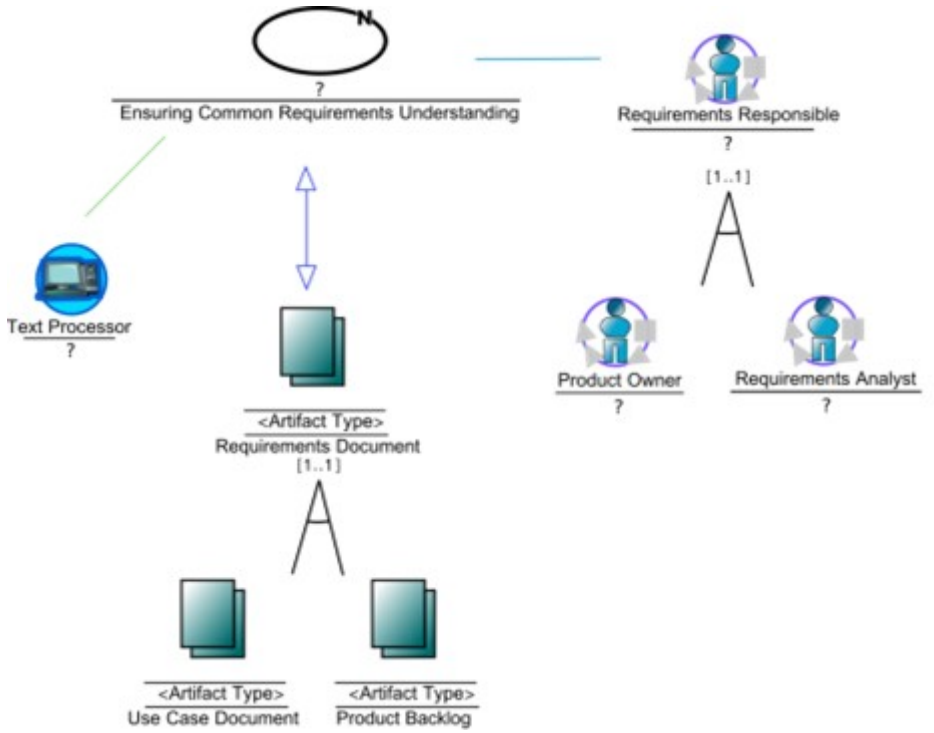


Figura 92 – Atividade "Ensuring Common Requirements Understanding".

O exemplo modelado nesta seção atendeu a todos os conceitos da linha de processos original e ainda acrescentou mais informações, como um maior número de tipos de variações, de tipos de dependências e de tipos de elementos de processo com variações compondo a linha de processos de software. Assim, o exemplo demostrou a viabilidade da utilização da abordagem proposta nesta dissertação para a modelagem de uma linha de processos publicada na literatura.

6. Avaliação da Abordagem Proposta

Este Capítulo apresenta a avaliação da abordagem para a representação de variações em linha de processos proposta nesta dissertação de mestrado.

6.1. Introdução

A abordagem proposta nesta dissertação foi avaliada das seguintes maneiras: (i) publicação de artigos científicos; (ii) e utilização da abordagem por outra dissertação de mestrado.

6.2. Publicação de Artigos Científicos

O artigo de Carvalho *et al.* (2014b) publicou os resultados da revisão sistemática da literatura, avaliando o método que foi seguido e os resultados da mesma, que serviram de base para o levantamento dos requisitos para a representação de variações apresentados neste trabalho. Além dos requisitos, os resultados da revisão sistemática guiaram a elaboração do metamodelo e do processo de engenharia de linha de processos.

A modelagem de uma linha de processos de software para a aplicação conjunta de modelos de maturidades e métodos ágeis, utilizando o metamodelo proposto nesta dissertação, foi publicada por Carvalho *et al.* (2014a).

A publicação de artigos serviu como uma forma de avaliação da relevância dos resultados desta dissertação de mestrado, visto que cada artigo foi submetido a um processo de revisões por pares por três especialistas da área.

6.3. Utilização da Abordagem por outra Dissertação de Mestrado

O metamodelo, a notação gráfica e o processo de engenharia de linha de processos especificados nesta dissertação serviram de base para a definição de uma linha de processos de software na dissertação de mestrado de Chagas (2015) e a colaboração entre os dois trabalhos resultou em algumas sugestões de melhoria, dentre as quais podemos destacar:

- (i) a especificação de variações em elementos de processos e não em características. Esta medida teve como objetivo aumentar o controle sobre os elementos de processo que devem fazer parte de uma arquitetura de linha de processos. A representação de variações nas características implicaria na seguinte situação: como cada característica pode referenciar vários elementos de processo, então caso ela seja opcional/obrigatória, todos os elementos referenciados também seriam opcionais/obrigatórios. Transferindo a representação de variações para os elementos de processo, torna possível a especificação mais refinada de quais elementos devem ser obrigatórios ou opcionais;
- (ii) a possibilidade de especificar instâncias de um mesmo elemento de processo em um ALPrS com tipos de variações diferentes. Em uma versão anterior do metamodelo, todas as referências a um elemento de processo em uma ALPrS possuíam o mesmo tipo de variação. Por exemplo, uma atividade especificada como obrigatória teria todas as suas instâncias obrigatórias. Para permitir que cada instância tenha um tipo de variação diferente, foi acrescentada a classe "ProcessElementInstance";
- (iii) a possibilidade de uma ALPrS referenciar características avulsas e não modelos de características, tendo como objetivo aumentar a flexibilidade e o grau de reutilização das características, visto que, se um modelo de características como um todo for referenciado, todas as característica constituintes seriam referenciadas, diminuindo a coesão. Por exemplo, se fosse definido um modelo de características para a representação do modelo de maturidade CMMI-DEV (SEI, 2010) em conjunto com o Scrum, todas as características seriam reutilizadas, o que pode dificultar o entendimento e os esforços de adaptação de processos. Por outro lado, sendo possível à uma ALPrS referenciar características diretamente, é possível reutilizar somente as características que que enquadrem no escopo do domínio que está sendo modelado;

- e (iv) a possibilidade de classificar as características por categoria. Essa foi uma necessidade identificada durante a modelagem de linha de processos que facilitava a etapa inicial do processo, pois permite a especificação de um requisito de processo em um alto nível de abstração e, em seguida, a modelagem de características mais específicas a partir dessas categorias. Por exemplo, especificar uma categoria "Métodos Ágeis" e, a partir de então, diversos modelos de características com métodos específicos, como Scrum, *Lean* e XP.

A utilização da abordagem na dissertação de Chagas (2015) demonstrou a viabilidade da utilização dos conceitos propostos neste trabalho para a modelagem de uma linha de processos de software e forneceu uma série de sugestões de melhorias para a adequação da abordagem em uma utilização prática.

6.4. Considerações Finais

Este Capítulo apresentou a avaliação da abordagem proposta. A publicação de artigos com resultados parciais evidenciou a relevância científica dos mesmos. A utilização da abordagem por outra dissertação de mestrado demonstrou a viabilidade da modelagem de linha de processos e proporcionou a identificação de oportunidades de melhoria para os conceitos.

7. Conclusão

Este Capítulo apresenta as considerações finais, contribuições para a área, limitações e trabalhos futuros que possam evoluir, complementar e melhorar os resultados do trabalho apresentado nesta dissertação.

7.1. Considerações Finais

Este trabalho teve como objetivo principal elaborar uma abordagem para auxiliar organizações de software e pesquisadores na representação de variações em processos de software utilizando o paradigma de linha de processos de software.

Para o embasamento teórico e direcionamento inicial da pesquisa, foi realizada uma revisão sistemática da literatura a partir da qual foi possível visualizar o estado atual e limitações da área de pesquisa. Em seguida, foram elaborados requisitos para a representação de variações em linha de processos de software através da análise de publicações retornadas pela revisão sistemática, complementadas por teses de doutorado e dissertações de mestrado. Além dos requisitos, foram identificados metamodelos e notações gráficas existentes, que foram comparados em relação ao atendimento aos requisitos. Por fim, foram elaborados um metamodelo e uma notação gráfica que satisfizeram alguns dos requisitos levantados.

A abordagem proposta deste trabalho foi validada através da publicação de artigos científicos com resultados parciais (CARVALHO *et al.*, 2014b) e da utilização do metamodelo, notação gráfica e processo por outra dissertação de mestrado (seção 6.3, p. 169).

7.2. Contribuições

As principais contribuições resultantes desta dissertação de mestrado são:

- **Requisitos para a representação de variações:** a partir da análise de metamodelos e notações gráficas encontradas em publicações existentes, foram coletados requisitos para

a representação de variações em linha de processos de software. Esses requisitos podem ser interessantes para a elaboração, comparação e melhoria de metamodelos, notações gráficas, ferramentas e métodos de engenharia de linha de processos de software.

- **Estudo comparativo:** metamodelos e notações gráficas existentes para a representação de variações em linha de processos de software foram comparadas em relação ao atendimento aos requisitos levantados. Tal estudo comparativo pode servir de guia para a elaboração e melhoria de metamodelos, notações gráficas e ferramentas para o apoio a linhas de processos de software. Além disso, pode servir como uma oportunidade para analisar lacunas a partir das quais pesquisas mais aprofundadas podem ser realizadas.
- **Metamodelo:** uma das principais contribuições desta dissertação foi o metamodelo elaborado, que buscou atender aos requisitos levantados e contribuir para o avanço na área de linha de processos de software. Como o metamodelo incorpora conhecimentos encontrados através de uma análise ampla da literatura, pode contribuir para a elaboração e melhoria de metamodelos, notações gráficas e ferramentas para o apoio a linhas de processos de software.
- **Notação Gráfica:** uma notação gráfica foi elaborada com o objetivo de representar graficamente variações em processos de software. Tal notação apresenta alguns conceitos inerentes ao metamodelo proposto e foi elaborada a partir da extensão de uma linhagem de modelagem de processos preexistente.
- **Processo:** um processo de engenharia de linha de processos foi elaborado, contento as etapas de engenharia de domínio e engenharia de aplicação, auxiliando na melhor utilização dos conceitos englobados pela metamodelo.
- **Revisão sistemática da literatura:** durante a elaboração deste trabalho, foi realizada uma revisão sistemática da literatura para a análise do estado da arte e identificação de oportunidades de pesquisas na área de linha de processos de software. Tal estudo seguiu um método formal e utilizou critérios objetivos para a identificação e avaliação de publicações existentes, servindo como um embasamento sólido para a elaboração da pesquisa. Os resultados da revisão sistemática forneceram fontes bibliográficas, conceitos, hipóteses, resultados de estudos empíricos, relatos de uso na indústria e academia, métodos de engenharia, metamodelos, notações gráficas, ferramentas e diversas outras informações que podem ser utilizadas para o embasamento teórico e para o direcionamento de novas pesquisas na área.
- **Artigos publicados:** durante a elaboração desta dissertação de mestrado, foram publicados três artigos que servem de validação desta pesquisa e podem contribuir para o

embasamento teórico de novas pesquisas. Nas etapas inicias da pesquisa, foi publicado um artigo sobre reutilização de processos de software (CARVALHO *et al.*, 2011). Os resultados da revisão sistemática foram publicados em Carvalho *et al.* (2014b). Por fim, o trabalho de Carvalho *et al.* (2014a) apresenta a modelagem de uma linha de processos de software para a aplicação conjunta de modelos de maturidades e métodos ágeis.

7.3. Limitações

As principais limitações desta dissertação de mestrado são:

- **Falta de especificação mais detalhada sobre as regras de um ponto de variação aberto:** o metamodelo prevê a possibilidade de especificar se um ponto de variação é aberto ou fechado. Entretanto, não foram especificados conceitos, regras e restrições detalhados para o tratamento de pontos de variação abertos.
- **Falta de uma especificação mais detalhada sobre raciocínios para a resolução de variações:** o metamodelo suporta a especificação do raciocínio de resolução de variações apenas através de um atributo de texto livre, dificultado um maior grau de automação para o mecanismo.
- **Falta de uma especificação mais detalhada sobre a resolução de variações em tempo de execução:** o metamodelo permite especificar o momento do ciclo de vida da engenharia de linha de processo em que a resolução das variações deve ocorrer (durante a derivação ou execução do processo). Entretanto, não foram definidas regras que garantem a consistência e a flexibilidade de tais escolhas durante a execução do processo.
- **Falta de especificação mais detalhada sobre variações transversais:** durante a análise das publicações retornadas pela revisão sistemática, foram identificados trabalhos que utilizaram conceitos de variações transversais, ou seja, que afetam os processos como um todo e não em pontos específicos. Entretanto, esse requisito não foi elaborado no metamodelo especificado nesta dissertação de mestrado.
- **Falta de avaliação empírica para o metamodelo:** como resultados deste trabalho, foram levantados requisitos para a representação de variações em linha de processos de software e foi proposto um metamodelo que os incorpora. A viabilidade da modelagem de linha de processos com a abordagem proposta neste trabalho foi avaliada através da modelagem de uma linha de processos de engenharia de requisitos publicada na literatura (HURTADO *et al.*, 2013). Entretanto, os requisitos para a representação de variações incorporados ao metamodelo não foram avaliados empiricamente de modo avaliar a

utilidade, os benefícios e as dificuldades que cada um deles implica na modelagem de linha de processos.

7.4. Trabalhos Futuros

Algumas questões ficaram em aberto e oportunidades de melhorias e evoluções a este trabalho foram identificados, com destaque para:

- **Resolução das limitações:** um dos trabalhos futuros identificados foi resolver as limitações identificadas na seção 7.3, como segue:
    - **Especificação mais detalhada sobre as regras de um ponto de variação aberto:** esse tipo de flexibilidade pode especificar simplesmente a permissão para a inclusão de qualquer variante não previamente associado ao ponto de variação ou pode haver regras específicas que um variante deve satisfazer para poder ser escolhido. Essas questões ficaram em aberto e são possíveis áreas de investigações futuras para a evolução do metamodelo.
    - **Especificação mais detalhada sobre raciocínios para a resolução de variações:** conceitos mais elaborados, como foram especificados nos trabalhos de Alegría *et al.* (2011), Alegría e Bastarrica (2012), Hurtado *et al.* (2013) e Martínez-Ruiz *et al.* (2011b) podem servir de inspiração para a evolução do metamodelo proposto nesta dissertação no sentido de incorporar conceitos e regras para a gerência de raciocínio. Existe ainda a possibilidade de integração com o mecanismo de adaptação de processos baseado em contexto definido na dissertação de mestrado de Costa (2010).
    - **Especificação mais detalhada sobre resoluções de variações em tempo de execução:** a possibilidade de resolver variações na etapa de execução do processo pode melhorar a flexibilidade, mas é necessário especificar as regras que garantam a consistência do processo sendo executado. Como o metamodelo proposto nesta dissertação pretende ser independente de linguagem de modelagem de processo, não foram especificadas quais são essas regras. Portanto, é necessário definir regras específicas para cada mecanismo de execução associado à linguagem de modelagem de processo utilizada. Um trabalho futuro identificado foi a elaboração de um mecanismo que estenda as regras de execução de uma ferramenta existente de forma a tratar os conceitos de variações durante a execução do processo derivado a partir da linha de processos.

  - **Especificação mais detalhada sobre variações transversais:** a possibilidade de especificar variações que afetam os processos como um todo precisa de mais investigações. Esse tipo de variação envolve regras específicas, como os tipos de elementos que podem ser alvos de variações transversais, a forma de inserir tais elementos e a especificação das situações ou eventos que disparam a inserção. Por exemplo, a possibilidade de especificar que uma atividade de garantia da qualidade deve ser inserida sempre que um tipo de produto de trabalho sob um controle rígido de qualidade é gerado ou atualizado.

- **Atualização da revisão sistemática da literatura:** a revisão sistemática identificou artigos científicos publicados até o ano de 2013. Novos trabalhos podem ter sido publicados e evoluções significativas em trabalhos anteriores podem ter ocorrido desde então. Além disso, novas questões de pesquisa podem ser incluídas ou questões existentes podem ser investigadas com mais profundidade. Por exemplo, a revisão sistemática realizada neste trabalho investigou quais publicações citavam a existência de ferramental para apoio às suas abordagens e uma investigação mais profunda pode ser realizada para identificar quais são as funcionalidades e as etapas da engenharia de linha de processos cobertas por tais ferramentas. Portanto, um trabalho futuro poderá atualizar a revisão sistemática com novas informações.
- **Implementação do metamodelo:** o metamodelo elaborado incorporou alguns dos requisitos levantados, mas ainda está em um estágio conceitual, dificultando uma possível avaliação e utilização prática dos conceitos. Ainda não existe a possibilidade de modelar e instanciar uma linha de processos com o auxílio de uma ferramenta especializada, o que dificulta a utilização do metamodelo e da notação gráfica em um projeto real. Portanto, a implementação de uma ferramenta de apoio é uma das oportunidades de evoluções futuras da pesquisa.
- **Avaliação empírica:** uma das principais direções futuras da pesquisa é a realização de estudos empíricos, como estudo experimental e estudo de caso, para a avaliação dos requisitos, do metamodelo e da notação gráfica. Os estudos empíricos investigariam cada um dos requisitos levantados, evidenciando a real utilidade e os possíveis benefícios e dificuldades que eles trazem para a abordagem. Adicionalmente, a notação gráfica precisar ser avaliada em relação à sua legibilidade e expressividade.
- **Integração com um metamodelo de contexto:** o presente metamodelo não trata em detalhes a representação de contextos de adaptação. Portanto, um dos trabalhos futuros seria a inclusão de conceitos ao metamodelo para acrescentar a capacidade de representar

informações de contextos do projeto e da organização de software que forneçam guias para a instanciação de processos. Apesar de não tratar diretamente de linha de processos de software, o tema adaptação baseada em contextos foi abordado em uma dissertação de mestrado do grupo LABES/UFPA (COSTA, 2010) e pode-se realizar um estudo para a integração dos conceitos.

Referências Bibliográficas

AIELLO, M.; BULANOV, P.; GROEFSEMA, H. Requirements and Tools for Variability Management. In: ANNUAL IEEE COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE WORKSHOPS, COMPSACW, 34., 2010, Seul, Coreia do Sul. **Proceedings…** Washington: IEEE Computer Society, 2010. p. 245-250. ISBN: 978-0-7695-4105-1.

BACHMANN, F.; BASS, L. Managing Variability in Software Architectures. In: SYMPOSIUM ON SOFTWARE REUSABILITY: PUTTING SOFTWARE REUSE IN CONTEXT, SSR, 2001, Toronto, Canadá. **Proceedings…** New York: ACM, 2001.

BARRETO, A. **Uma Abordagem para Definição de Processos Baseada em Reutilização Visando à Alta Maturidade em Processos.** 2011. 327 f. Tese (Doutorado em Engenharia de Sistemas e Computação)–Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia (COPPE), Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2011.

BARRETO, A.; MURTA, L.; ROCHA, A. Componentizando Processos Legados de Software Visando a Reutilização de Processos. In: SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE, SBQS, 8., 2009, Ouro Preto, Brasil. **Anais...** [s.l.]: [s.n.], 2009.

BERTOLLO, G.; FALBO, R. Apoio Automatizado à Definição de Processos de Software em Níveis. In: SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE, SBQS, 2., 2003, Fortaleza, Brasil. **Anais...** [s.l.]: [s.n.], 2003.

BOFFOLI, N.; CIMITILE, M.; MAGGI, F. Managing Business Process Flexibility and Reuse through Business Process Lines. In: INTERNATIONAL CONFERENCE ON SOFTWARE AND DATA TECHNOLOGIES, ICSOFT, 4., 2009, Sófia, Bulgária. **Proceedings…** Berlin: Springer, 2009. p. 61-68.

BORGES, L.; FALBO, R. Gerência de Conhecimento sobre Processos de Software. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, SBES, 15.,v2001, Rio de Janeiro, Brasil. **Anais…** Rio de Janeiro: [s.n.], 2001.

Budgen, D.; Brereton, P. Performing Systematic Literature Reviews in Software

Engineering. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 28., 2006, Xangai, China. **Proceedings…** New York: ACM, 2006. p. 1051-1052.

CARVALHO, D.; COSTA, A.; SALES, E.; LIMA, A.; REIS, R. Apoio à Reutilização de Processos de Software em um Ambiente de Engenharia de Software Centrado em Processo. In: SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE, SBQS, 10., 2011, Curitiba, Brasil. **Anais...** Curitiba: [s.n.], 2011.

CARVALHO, D.; CHAGAS, L.; LIMA REIS, C. Definition of Software Process Lines for Integration of Scrum and CMMI. In: CONFERENCIA LATINOAMERICANA EN INFORMÁTICA, CLEI, 40., 2014a, Montevidéu, Uruguai. **Proceedings...** [s.l.]: IEEE Computer Society, 2014a. p. 1-12.

CARVALHO, D.; CHAGAS, L.; LIMA, A.; LIMA REIS, C. Software Process Lines: A Systematic Literature Review. In: International Conference on Software Process Improvement and Capability Determination, SPICE, 14., 2014b, Vilnius, Lituânia. **Proceedings...** Suíça: Springer International Publishing, 2014b. p. 118-130.

CHAGAS, L. **Linha de Processos de Software para Integração entre práticas do Scrum e de Modelos de Maturidade**. 2015. Dissertação (Mestrado em Ciência da Computação)–Instituto de Ciências Exatas e Naturais, Universidade Federal do Pará, Belém, 2015.

COSTA, A. **Um Mecanismo de Adaptação de Processos de Software**. 2010. 103 f. Dissertação (Mestrado em Ciência da Computação)–Instituto de Ciências Exatas e Naturais, Universidade Federal do Pará, Belém, 2010.

COSTA, A; SALES, E. **Uma Proposta para Reutilização de Processos de Software para o Ambiente WebAPSEE**. 2007. 87 f. Trabalho de Conclusão de Curso (Bacharel em Ciência da Computação)–Centro de Ciências Exatas e Naturais, Universidade Federal do Pará, Belém, 2007.

COSTA, A.; SALES, E.; REIS, R.; LIMA REIS, C. Apoio a Reutilização de Processos de Software através de Templates e Versões. In: SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE, SBQS, 6., 2007, Porto de Galinhas, Brasil. **Anais...** Porto de Galinhas: [s.n.], 2007.

DAIZHONG, L.; SHANHUI, D. Feature Dependency Modeling for Software Product Line. In: INTERNATIONAL CONFERENCE ON COMPUTER ENGINEERING AND TECHNOLOGY - VOLUME 2, ICCET, 2009, SINGAPURA. **Proceeding...** Los Alamitos: IEEE Computer Society, 2009. p. 256-260.

FUGGETTA, A. Software Process: A Roadmap. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 22., 2000, Limerick, Irlanda. **Proceedings…** ACM: New York, NY, USA, p. 2000. p. 25-34.

GOMAA, H. Domain Modeling of Software Process Models. In: IEEE INTERNATIONAL CONFERENCE ON ENGINEERING OF COMPLEX COMPUTER SYSTEMS, ICECCS, 6., 2000, Tóquio, Japão. **Proceedings…** Los Alamitos: IEEE Computer Society, 2000.

GIESE, C.; SCHNIEDERS, A.; WEILAND, J. A Practical Approach for Process Family Engineering of Embedded Control Software. In: ANNUAL IEEE INTERNATIONAL CONFERENCE ON THE ENGINEERING OF COMPUTER BASED SYSTEMS, ECBS, 14., 2007, Tucson, EUA. **Proceedings...** Los Alamitos: IEEE Computer Society, 2007. p. 229-240.

HEIDARI, F.; LOUCOPOULOS, P.; BRAZIER, F.; BARJIS, J. A Meta-Meta-Model for Seven Business Process Modeling Languages. In: IEEE CONFERENCE ON BUSINESS INFORMATICS, CBI, 15., 2013,Viena, Áustria. **Proceedings…** Washington: IEEE Computer Society, 2013. p. 216-221.

HERNANDES, E.; ZAMBONI, A.; FABBRI, S.; THOMMAZO, A. Using GQM and TAM to evaluate StArt – a tool that supports Systematic Review. **CLEI Electronic Journal**, v. 15, n. 1, p. 13-25, 2012.

KELLNER, M. Connecting Reusable Software Process Elements and Components. In: INTERNATIONAL SOFTWARE PROCESS WORKSHOP, 10., 1996, Dijon, França. **Proceedings...** Los Alamitos: IEEE Computer Society, 1996. p. 8-11.

KITCHENHAM, Barbara. **Guidelines for performing Systematic Literature Review in Software Engineering**. Technical Report, EBSE-2007-1, Version 2.3. 2007.

KULKARNI, V.; BARAT, S. Business Process Families Using Model-Driven Techniques. In: Business Process Management Workshops, BPM, 2010, Hoboken, EUA. **Proceedings...** Berlin: Springer, 2011. p. 314-325.

LABES – LABORATÓRIO DE ENGENHARIA DE SOFTWARE, 2006, **Documentação de Referência do Sistema WebAPSEE 1.0 (beta)**. Disponível em: < http://ufpa.br/webapsee/images/documentacaoWebAPSEE/doc%20de%20referencia.pdf>.

LIMA, A.; FRANÇA, B.; SCHLEBBE, H.; SILVA, M.; REIS, R.; LIMA REIS, C. WebAPSEE: Um Ambiente Livre e Flexível Para Gerência de Processos de Software. In: WORKSHOP DE SOFTWARE LIVRE, WSL, 7., 2006, Porto Alegre, Brasil. **Anais do Fórum Internacional de Software Livre 7.0**... Porto Alegre, Brasil: [s.n.], 2006. p. 163-168.

LIMA REIS, C. **Uma Abordagem Flexível para Execução de Processos de Software Evolutivos**. 2003. 277 f. Tese (Doutorado em Ciência da Computação)–Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2002.

LIMA REIS, C.; REIS, R. Laboratório de Engenharia de Software e Inteligência Artificial: Construção do ambiente WebAPSEE. **Revista ProQuality - Qualidade na Produção de Software**, Lavras, Brasil, , v. 3, p. 43-48, 2007.

MARTÍNEZ-RUIZ, T.; MÜNCH, J.; GARCÍA, F.; PIATTINI, M. Requirements and Constructors for Tailoring Software Processes: A Systematic Literature Review. **Software Quality Journal**, Hingham, EUA, v. 20, Issue 1, 2012, p. 229-260.

MARTÍNEZ-RUIZ, T.; RUIZ, F.; PIATTINI, M. Towards Understanding Software Process Variability from Contextual Evidence of Change. In: INTERNATIONAL WORKSHOP ON VARIABILITY SUPPORT IN INFORMATION SYSTEMS, VarIS, 1., 2013b, Valência, Espanha. **Proceedings…**. Berlin: Springer, 2013b. p. 417-431.

NAKAMURA, M.; KUSHIDA, T.; BHAMIDIPATY, A.; CHETLUR, M. A Multi-layered Architecture for Process Variation Management. In: WORLD CONFERENCE ON SERVICES - PART II, SERVICES-2, 5., 2009, Bangalore, Índia. **Proceedings...** Los Alamitos: IEEE Computer Society, 2009. p. 71-78.

PETERSEN, K.; BRAMSIEPE, N.; POHL, K. Applying Variability Modeling Concepts to Support Decision Making for Service Composition. In: International Workshop on Service-Oriented Computing: Consequences for Engineering Requirements, SOCCER, 2., 2006, Minneapolis, EUA. **Proceedings...** Washington: IEEE Computer Society, 2006.

ROCHA, R., FANTINATO, M.: The use of software product lines for business process management: A systematic literature review. **Information and Software Technology**, Volume 55, Issue 8, August 2013, pp. 1355-1373. (2013)

SALES. E.; FREITAS, S;. QUITES, R. Uma Ferramenta para Recuperação de Modelos de Processo de Software Reutilizáveis. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, SBES, 20., 2006, Florianópolis, Brasil. **Anais da Sessão de Ferramentas...** Florianópolis: [s.n.], 2006.

SCHNIEDERS, A.; PUHLMANN, F. Variability Mechanisms in E-Business Process Families. In: INTERNATIONAL CONFERENCE ON BUSINESS INFORMATION SYSTEMS, BIS, 9., 2006, Klagenfurt, Áustria. **Proceedings...** [s.l.]: [s.n], 2006.

SIMIDCHIEVA, B.; CLARKE, L.; OSTERWEIL, L. Representing Process Variation with a Process Family. In: INTERNATIONAL CONFERENCE ON SOFTWARE PROCESS,

ICSP, 2007, Minneapolis, USA. **Proceedings…** Heidelberg, Germany: Springer-Verlag, 2007. p. 109-120. ISBN: 978-3-540-72425-4.

SOUZA, G. Ambientes **de Engenharia de Software Orientados a Corporação**. 2011. 319 f. Tese (Doutorado em Engenharia de Sistemas e Computação)–COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2011.

OMG – OBJECT MANAGEMENT GROUP, 2008. **SPEM - Software & Systems Process Engineering Meta-Model Specification v2.0**. Disponível em: <http://www.omg.org/spec/SPEM/2.0/PDF>.

OMG – OBJECT MANAGEMENT GROUP, 2012. **Common Variability Language Wiki**. Disponível em: <http://www.omgwiki.org/variability>.

REIS, R. **APSEE-Reuse:** Um Meta-Modelo para Apoiar a Reutilização de Processos de Software. 2002. 215 f. Tese (Doutorado em Ciência da Computação)–Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2002.

SEI – SOFTWARE ENGINEERING INSTITUTE, 2010. **CMMI for Development, Version 1.3, CMU/SEI-2010-TR-033**. Disponível em <http://www.sei.cmu.edu/reports/10tr033.pdf>.

SOFTEX – ASSOCIAÇÃO PARA PROMOÇÃO DA EXCELÊNCIA DO SOFTWARE BRASILEIRO, 2012. **Guia Geral MPS de Software**. Disponível em: <http://www.softex.br/wp-content/uploads/2013/07/MPS.BR_Guia_Geral_Software_2012-c-ISBN-1.pdf>

SOFTEX – ASSOCIAÇÃO PARA PROMOÇÃO DA EXCELÊNCIA DO SOFTWARE BRASILEIRO, 2013. **Guia de Implementação – Parte 3: Fundamentação para Implementação do Nível E do MR-MPS-SW:2012**. Disponível. em: <http://www.softex.br/wp-content/uploads/2013/07/MPS.BR_Guia_de_Implementacao_Parte_3_20131.pdf>

PRESSMAN, R. **Engenharia de Software**: Uma Abordagem Profissional. Tradução: Ariovaldo Griese. 7. ed. Porto Alegre: AMGH, 2011.

TEIXEIRA E. **OdysseyProcess-FEX**: Uma Abordagem para Modelagem de Variabilidades de Linha de Processos de Software. 2011. 161 f. Dissertação (Mestrado em Engenharia de Sistemas e Computação)–Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia (COPPE), Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2011.

WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M.; REGNELL, B., WESSLÉN, A. **Experimentation in Software Engineering**. Berlin: Springer-Verlag, 2012.

Referências Bibliográficas (Estudos Primários da RSL)

ADAM, S.; DOERR, J. The Role of Service Abstraction and Service Variability and Its Impact on Requirements Engineering for Service-Oriented Systems. In: ANNUAL IEEE INTERNATIONAL COMPUTER SOFTWARE AND APPLICATIONS, COMPSAC, 32., 2008, Turku, Finlândia. **Proceedings…** Los Alamitos: IEEE Computer Society, 2008. p. 631-634.

ALEGRÍA, J.; BASTARRICA, M. Building Software Process Lines with CASPER. In: INTERNATIONAL CONFERENCE ON SOFTWARE AND SYSTEM PROCESS, ICSSP, 2002, Zurique, Suíça. **Proceedings...** [s.l.]: IEEE Computer Society, 2012. p. 170-179.

ALEGRÍA, J.; BASTARRICA, M.; QUISPE, A.; OCHOA, S. An MDE Approach to Software Process Tailoring. In: INTERNATIONAL CONFERENCE ON SOFTWARE AND SYSTEMS PROCESS, ICSSP, 2011, Honolulu, EUA. **Proceedings...** Nova York: ACM, 2011. p. 43-52. ISBN: 978-1-4503-0730-7.

ALEIXO, F.; FREIRE, M.; SANTOS, W.; KULESZA, U. A Model-driven Approach to Managing and Customizing Software Process Variabilities. In: INTERNATIONAL CONFERENCE ON ENTERPRISE INFORMATION SYSTEMS, ICEIS, 12., 2010, Funchal, Portugal. **Proceedings...** [s.l.]: Springer, 2010. p. 92-100.

ALEIXO, F.; FREIRE, M.; ALENCAR, D.; CAMPOS, E.; KULESZA, U. A Comparative Study of Compositional and Annotative Modelling Approaches for Software Process Lines. In: BRAZILIAN SYMPOSIUM ON SOFTWARE ENGINEERING, SBES, 26., 2012, Natal, Brasil. **Anais...** [s.l.]: IEEE Computer Society, 2012. p. 51-60.

ARMBRUST, O.; KATAHIRA, M.; MIYAMOTO, Y.; MÜNCH, J.; NAKAO, H.; OCAMPO, A. Scoping Software Process Models – Initial Concepts and Experience from Defining Space Standards. In: INTERNATIONAL CONFERENCE ON SOFTWARE PROCESS, ICSP, 2008, Leipzig, Alemanha. **Proceedings...** Berlin: Springer 2008. p. 160-172.

ARMBRUST, O.; KATAHIRA, M.; MIYAMOTO, Y.; MÜNCH, J.; NAKAO, H.; OCAMPO, A. Scoping Software Process Lines. In: **Software Process: Improvement and**

Practice - Examining Process Design and Change, Volume 14, Issue 3, May 2009, p. 181-197. John Wiley & Sons, New York (2009).

AZANZA, M.; SOSA, J.; TRUJILLO, S.; DÍAZ, O. Towards a Process-line for MDPLE. In: INTERNATIONAL WORKSHOP ON MODEL-DRIVEN PRODUCT LINE ENGINEERING, MDPLE, 2., 2010, Paris, França. **Proceedings...** [s.n.]: [s.l.], 2010. p. 3-12.

BARRETO, A.; MURTA, L.; ROCHA, A. Software Process Definition: A Reuse-based Approach. In: **Journal of Universal Computer Science**, v. 17, issue 13, p. 1765-1799 (2011).

BARRETO, A.; NUNES, E.; ROCHA, A.; MURTA, L. Supporting the Definition of Software Processes at Consulting Organizations via Software Process Lines. In: INTERNATIONAL CONFERENCE ON THE QUALITY OF INFORMATION AND COMMUNICATIONS TECHNOLOGY, QUATIC, 7., 2010, Oporto, Portugal. **Proceedings…** Washington: IEEE Computer Society, 2010. p. 15-24.

BIFFL, S.; WINKLER, D.; HÖHN, R.; WETZEL, H. Software Process Improvement in Europe: Potential of the New V-Modell XT and Research Issues. In: **Software Process: Improvement and Practice**, v. 3, Issue 11 (2006)

BOUCHER, Q.; PERROUIN, G.; DEPREZ, J.; HEYMANS, P. Towards Configurable ISO/IEC 29110-Compliant Software Development Processes for Very Small Entities. In: EUROPEAN CONFERENCE, EuroSPI, 19., 2012, Viena, Áustria. **Proceedings...** Berlin: Springer, 2012. p. 169-180.

CASTRO, J.; PIMENTEL, J.; LUCENA, M.; SANTOS, E.; DERMEVAL, D. F-STREAM: A Flexible Process for Deriving Architectures from Requirements Models. In: INTERNATIONAL WORKSHOP ON SYSTEM/SOFTWARE ARCHITECTURES, IWSSA, 9., 2011, Londres, Inglaterra. **Proceedings...** Berlin: Springer, 2011. p. 342-353.

COSTACHE, D.; KALUS, G.; KUHRMANN, M. Design and Validation of Feature-Based Process Model Tailoring - A Sample Implementation of PDE. In: 19TH ACM SIGSOFT SYMPOSIUM AND THE 13TH EUROPEAN CONFERENCE ON FOUNDATIONS OF SOFTWARE ENGINEERING, ESEC/FSE, Szeged, Hungria. **Proceedings...** Nova York: ACM, 2011. p. 464-467.

DURÁN, A.; BENAVIDES, D.; BERMEJO, J. Applying System Families Concepts to Requirements Engineering Process Definition. In: INTERNATIONAL WORKSHOP SOFTWARE PRODUCT-FAMILY ENGINEERING, PFE, 5., 2003, Siena, Itália. **Proceedings...** Berlin: Springer, 2004. p. 140-151.

GALLINA, B.; SLJIVO, I.; JARADAT, O. Towards a Safety-Oriented Process Line for Enabling Reuse in Safety Critical Systems Development and Certification. In: 35th ANNUAL

IEEE SOFTWARE ENGINEERING WORKSHOP, SEW, 35., 2012, Heraklion, Grécia. **Proceedings...** Los Alamitos: IEEE Computer Society, 2012. p. 148-157.

GOLPAYEGANI, F.; AZADBAKHT, K.; RAMSIN, R. Towards Process Lines for Agent-Oriented Requirements Engineering. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER AS A TOOL, EUROCON, 14., 2013, Zagreb, Croácia. **Proceedings...** [s.l.]: IEEE Computer Society, 2013. p. 550-557.

GOMAA, H.; KERSCHBERG, L.; FARRUKH, G. Domain Modeling of Software Process Models. In: IEEE INTERNATIONAL CONFERENCE ON ENGINEERING OF COMPLEX COMPUTER SYSTEMS, ICECCS, 6., 2000, Tóquio, Japão. **Proceedings…** Los Alamitos: IEEE Computer Society, 2000. p. 50-60.

HURTADO, J.; BASTARRICA, M.; OCHOA, S.; SIMMONDS, J. MDE Software Process Lines in Small Companies. In: **The Journal of Systems and Software**, v. 86, Issue 5, p. 1153-1171, 2013.

JAFARINEZHAD, O.; RAMSIN, R. Development of Situational Requirements Engineering Processes: A Process Factory Approach. In: IEEE ANNUAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE, COMPSAC, 36., 2012, Izmir, Turquia. **Proceedings...** Washington: IEEE Computer Society, 2012. p. 279-288.

JAUFMAN, O.; MÜNCH, J. Acquisition of a Project-Specific Process. In: INTERNATIONAL CONFERENCE ON PRODUCT-FOCUSED SOFTWARE PROCESS IMPROVEMENT, PROFES, 6., 2005, Oulu, Finlândia. **Proceedings...** Berlin: Springer, 2005. p. 328-342.

KIEBUSCH, S.; FRANCZYK, B.; SPECK, A. Process-Family-Points. In: INTERNATIONAL SOFTWARE PROCESS WORKSHOP AND INTERNATIONAL WORKSHOP ON SOFTWARE PROCESS SIMULATION AND MODELING, SPW/ProSim, 2006, Xangai, China. **Proceedings...**. Berlin: Springer, 2006. p. 314-321.

MARTÍNEZ-RUIZ, T.; GARCÍA, F.; PIATTINI, M. Towards a SPEM v2.0 Extension to Define Process Lines Variability Mechanisms. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, RESEARCH, MANAGEMENT AND APPLICATIONS, SERA, 6., 2008, Praga, República Checa. **Proceedings…** Berlin: Springer, 2008. p. 115-130.

MARTÍNEZ-RUIZ, T.; GARCÍA, F.; PIATTINI, M.; MÜNCH, J. Modelling Software Process Variability: An Empirical Study. In: **IET Software**, Volume 5, Issue 2, p. 172-187 (2011a)

MARTÍNEZ-RUIZ, T.; GARCÍA, F.; PIATTINI, M. Managing Process Diversity by Applying Rationale Management in Variant Rich Processes. In: INTERNATIONAL

CONFERENCE ON PRODUCTFOCUSED SOFTWARE PROCESS IMPROVEMENT, PROFES, 12., 2011b, Torre Canne, Itália. **Proceedings...** Berlin: Springer, 2011b. p. 128-142.

MARTÍNEZ-RUIZ, T.; GARCÍA, F.; PIATTINI, M.; MÜNCH, J. Applying AOSE Concepts to Model Crosscutting Variability in Variant-Rich Processes. In: EUROMICRO CONFERENCE ON SOFTWARE ENGINEERING AND ADVANCED APPLICATIONS, SEAA, 37., 2011c, Oulu, Finlândia. **Proceedings…** Washington: IEEE Computer Society, 2011c. p. 334-338.

MARTÍNEZ-RUIZ, T.; GARCÍA, F.; PIATTINI, M.; LUCAS-CONSUEGRA, F. Process Variability Management in Global Software Development: A Case Study. In: INTERNATIONAL CONFERENCE ON SOFTWARE AND SYSTEM PROCES, ICSSP, 2013a, São Francisco, EUA. **Proceedings…** Nova York: ACM, 2013a. p. 46-55.

MAGDALENO, A.; ARAUJO, R.; WERNER, C. COMPOOTIM: An Approach to Software Processes Composition and Optimization. In: CONGRESSO IBERO-AMERICANO EM ENGENHARIA DE SOFTWARE, CIbSE, 15., Buenos Aires, Argentina. **Proceedings...** [s.l.]: [s.n.], 2012. p. 1-14.

MOSER, T.; BIFFL, S.; WINKLER, D. Process-Driven Feature Modeling for Variability Management of Project Environment Configurations. In: INTERNATIONAL CONFERENCE ON PRODUCT FOCUSED SOFTWARE DEVELOPMENT AND PROCESS IMPROVEMENT, PROFES, 11., Limerick, Irlanda. **Proceedings…** Nova York: ACM, 2010. p. 47-50.

NUNES, V.; WERNER, C.; SANTORO, F. Context-Based Process Line. In: INTERNATIONAL CONFERENCE ON ENTERPRISE INFORMATION SYSTEMS, ICEIS, Funchal, Portugal. **Proceedings…** [s.l]: [s.n.], 2010. p. 277-282.

RAUSCH, A.; KUHRMANN, M. A Proposal for Principles and Values from the Perspective of the German Standard IT-Development Process V-Modell XT. In: INTERNATIONAL CONFERENCE ON SOFTWARE AND SYSTEMS PROCESS, ICSSP, Honolulu, EUA. **Proceedings…** Nova York: ACM, 2011. p. 230-233.

ROMBACH, D. Integrated Software Process and Product Lines. In: INTERNATIONAL SOFTWARE PROCESS WORKSHOP, SPW, Pequim, China. **Proceedings…** Berlin: Springer, 2005. p. 83-90.

ROUILLÉ, E.; COMBEMALE, B.; BARAIS, O.; TOUZET, D.; JÉZÉQUEL, J. Leveraging CVL to Manage Variability in Software Process Lines. In: ASIA-PACIFIC SOFTWARE ENGINEERING CONFERENCE, APSEC, 19., Hong Kong, China. **Proceedings…** Washington: IEEE Computer Society, 2012. p. 148-157.

ROUILLÉ, E.; COMBEMALE, B.; BARAIS, O.; TOUZET, D.; JÉZÉQUEL, J. Improving Reusability in Software Process Lines. In: EUROMICRO CONFERENCE ON SOFTWARE ENGINEERING AND ADVANCED APPLICATIONS, SEAA, 39., Santander, Espanha. **Proceedings…** Washington: IEEE Computer Society, 2013. p. 90-93.

SIMMONDS, J.; BASTARRICA, M.; SILVESTRE, L.; QUISPE, A. Variability in Software Process Models: Requirements for Adoption in Industrial Settings. In: INTERNATIONAL WORKSHOP ON PRODUCT LINE APPROACHES IN SOFTWARE ENGINEERING, PLEASE, 4., São Francisco. **Proceedings…** [s.l.]: IEEE, 2013. p. 33-36.

SUTTON, S.; OSTERWEIL, L. Product Families and Process Families. In: INTERNATIONAL SOFTWARE PROCESS WORKSHOP, ISPW, 10., 1996, Dijon, França. **Proceedings…**. Los Alamitos: IEEE Computer Society, 1996. p. 109-111.

TERNITÉ, T. Process Lines: A Product Line Approach Designed for Process Model Development. In: EUROMICRO CONFERENCE ON SOFTWARE ENGINEERING AND ADVANCED APPLICATIONS, SEAA, 35., Patras, Grécia. **Proceedings…** Washington: IEEE Computer Society, 2009. p. 173-180.

WASHIZAKI, H. Building Software Process Line Architectures from Bottom Up. In: 7th INTERNATIONAL CONFERENCE ON PRODUCT-FOCUSED SOFTWARE PROCESS IMPROVEMENT, PROFES, 7., 2006a, Amsterdã, Holanda. **Proceedings…** Berlin: Springer, 2006a. p. 415-421.

WASHIZAKI, H. Deriving Project-Specific Processes from Process Line Architecture with Commonality and Variability. In: IEEE INTERNATIONAL CONFERENCE ON INDUSTRIAL INFORMATICS, INDIN, Singapura, 2006b. **Proceedings...** [s.l.]: IEEE Computer Society, 2006b. p. 1301-1306.

APÊNDICE A. METAMODELO

Este Apêndice apresenta os conceitos e regras que constituem o metamodelo elaborado no contexto desta dissertação de mestrado.

A.1. Introdução

Para a descrição de cada conceito presente no metamodelo, foi utilizada uma estrutura de cláusulas, apresentadas na Tabela 46.

Tabela 46 – Cláusulas para a Descrição de cada Conceito.

Cláusula	**Descrição**
Generalizações	Lista todas as generalizações diretas do conceito. Por generalização direta, entende-se estar imediatamente acima na hierarquia de classes.
Especializações	Lista todas as especializações diretas do conceito. Por especialização direta, entende-se estar imediatamente abaixo na hierarquia de classes.
Descrição	Uma descrição textual do significado do conceito.
Atributos	Lista todos os atributos que são definidos para o conceito. Cada atributo é descrito pelo seu nome, tipo, multiplicidade e uma descrição textual do significado do atributo.
Associações	Lista todas as associações entre o conceito e outros conceitos. Cada associação é descrita pelo seu nome, classe associada, multiplicidade e uma descrição textual do significado da associação.
Restrições	Lista todas as restrições que definem as regras de boa formação aplicadas ao conceito.
Comportamentos	Descreve os comportamentos específicos do conceito.
Notação	Apresenta a notação gráfica associada ao conceito.
Exemplos	Apresenta informações adicionais sobre a aplicação do conceito.

A.2. Classes

A.2.1. Modelo de Características

A.2.1.1. FeatureModel

Generalizações

Sem generalizações.

Especializações

Sem especializações.

Descrição

Representa um modelo de características contendo o conjunto de todas as características que fazem parte de uma determinada linha de processos de software. Modelo de características é uma estrutura em árvore composta por um conjunto de características, sendo que uma delas é a raiz.

Atributos

- **name** : String
  - Nome que identifica o modelo de características.
- **description** : String
  - Descrição textual do modelo de características.

Associações

- **root** : Feature[0..1]
  - Representa a característica raiz do modelo de características. A característica raiz é ascendente de todas as demais características do modelo de características e deve ser abrangente o suficiente para representá-lo como um todo.

Restrições

Sem restrições adicionais.

Comportamentos

Sem comportamentos.

Notação

Sem notação.

Exemplos

Um modelo de características pode representar a hierarquia de elementos de modelos de maturidade, como o MR-MPS-SW e CMMI-DEV. Nesses casos, o próprio nome do modelo de

características poderia ser a característica raiz e as demais características poderiam formar os diversos níveis de maturidades e áreas de processo.

A.2.1.2. Feature

Generalizações

Sem generalizações.

Especializações

Sem especializações.

Descrição

Em termos genéricos, uma característica representa um requisito em um determinado domínio (GOOMA *et al.*, 2000, p. 51). No contexto de processos de software, uma característica representa um requisito, em um determinado nível de abstração, que um processo de software pode satisfazer.

Atributos

- **name** : String
  - Nome que identifica a característica.
- **description** : String
  - Descrição textual da característica.

Associações

- **parentFeature** : Feature[0..1]
  - Representa a característica pai da característica. A característica pai é ascendente direta da característica em questão.
- **subFeatures** : Feature[0..*]
  - Representa as subcaracterísticas da característica. As subcaracterísticas são descendentes diretas da característica em questão.
- **processElements** : ProcessElement[0..*]
  - Representa os elementos de processo referenciados pela característica. A seleção (ou não) de uma característica implica que os elementos de processo referenciados por ela devem (ou não) estar presentes no modelo de processo derivado.
- **categories** : FeatureCategoty[0..*]
  - Representa as categorias nas quais a característica é associada.

Restrições

- [1] Uma característica não pode ter ela mesma como subcaracterística.
- [2] Uma característica não pode ter ela mesma como característica pai.
- [3] Se a característica for raiz no modelo de características, então ela não pode ter uma característica pai.
- [4] Se uma característica for selecionada, então todas as suas características ascendentes diretas e indiretas também devem ser selecionadas.
- [5] Se uma característica não for selecionada, então todas as suas características descendentes também não devem ser selecionadas.

Comportamentos

Sem comportamentos.

Notação

Sem notação.

Exemplos

Exemplos de requisitos de processo que podem ser representados por características de processo são níveis de maturidade e modelos de ciclo de vida (ALEGRÍA *et al.*, 2011).

A.2.1.3. FeatureCategory

Generalizações

Sem generalizações.

Especializações

Sem especializações.

Descrição

Uma categoria de característica representa um agrupamento de características.

Atributos

- **name** : String
  - Nome que identifica a categoria de característica.
- **description** : String
  - Descrição textual da categoria de característica.

Associações

- **features** : Feature[0..*]
  - Representa as características que pertencem à categoria.

Restrições

Sem restrições adicionais.

Notação

Sem notação.

Exemplos

Exemplos de categorias de características poderiam ser "Modelos de Maturidade" e "Métodos Ágeis".

A.2.2. Arquitetura de Linha de Processos de Software

A.2.2.1. ProcessLineArchitecture

Generalizações

Sem generalizações.

Especializações

Sem especializações.

Descrição

Representa uma arquitetura de linha de processos de software, que é uma estrutura central de processo que incorpora as similaridades e variabilidades a partir da qual todos os processos que constituem a linha de processos são derivados (WASHIZAKI, 2006a).

Atributos

- **name** : String
  - Nome que identifica a arquitetura de linha de processos.
- **description** : String
  - Descrição textual da arquitetura de linha de processos.

Associações

- **processElementInstances** : ProcessElementInstance[0..*]
  - Instâncias de elementos de processo que fazem parte da arquitetura da linha de processos.
- **dependencies** : Dependency[0..*]
  - Dependências entre instâncias de elementos de processo.
- **features** : Feature[0..*]
  - Características referenciadas.
- **variabilities** : Variability[0..*]
  - Variabilidades especificadas para as instâncias de elementos de processo.

Restrições

[1] Deve existir pelo menos uma instância de elemento de processo para cada característica adicionada à uma ALPrS.

[2] Deve existir pelo menos uma instância de "Variability" para cada instância de elemento de processo adicionada à uma ALPrS.

Comportamentos

[1] Quando uma característica é adicionada à ALPrS, todos os elementos de processo que atendem à característica são adicionados à arquitetura.

[2] Quando uma característica é adicionada à ALPrS, todas as subcaracterísticas são adicionadas à arquitetura.

[3] Quando uma instância de um elemento de processo é adicionada à ALPrS, todas as características associadas ao elemento de processo são adicionadas à arquitetura.

[4] Quando uma característica é removida da ALPrS, todas as instâncias de elementos que atendem à essa característica e que não atendem a nenhuma outra característica, são removidos da arquitetura.

[5] Quando uma característica é removida da ALPrS, todas as subcaracterísticas são removidas da arquitetura.

[6] Quando uma instância de um elemento é removida da ALPrS, todas as característica atendidas pelo elemento são removidas, exceto quando existir outra instância do mesmo elemento ou quando existir uma instância de outro elemento de processo que também atende à característica.

Notação

Sem notação.

Exemplos

Sem exemplos.

A.2.2.2. Variabilidades

A.2.2.2.1. Variability

Generalizações

Sem generalizações.

Especializações

ElementExistence (seção A.2.2.2.3, p. 194).

VariationPointRelationship (seção A.2.2.2.7, p. 198).

Descrição

Classe abstrata que representa os tipos de variabilidades que podem ser representadas em uma arquitetura de linha de processos de software. O metamodelo parte do princípio de que um elemento de processo não possui variabilidades intrínsecas. Cabe à uma arquitetura de linha de processos especificar se cada instância de um elemento é opcional, obrigatória, ponto de variação ou variante.

Atributos

- **bindingTime** : BindingTime[1]
    - Especifica o momento do ciclo de vida da engenharia de linha de processos em que a(s) instância(s) variante(s) ou opcional(ais) deve(m) ser selecionada(s). Pode assumir os valores *derivation*, que especifica que a escolha sobre a seleção ou não da(s) instância(s) deve ser feita durante a derivação do processo; e *enactment*, que especifica que as escolhas podem ser feitas na execução do processo.

Associações

Sem associações adicionais.

Restrições

Sem restrições adicionais.

Comportamentos

Sem comportamentos.

Notação

Sem notação.

Exemplos

Sem exemplos.

A.2.2.2.2. BindingTime

Generalizações

Sem generalizações.

Especializações

Sem especializações.

Descrição

Enumeração que especifica o momento de realizar as resoluções das variações. Os valores podem ser:

- *derivation*, que especifica que a escolha sobre a seleção ou não da(s) instância(s) deve ser feita durante a derivação do processo;

- e *enactment*, que especifica que, embora a escolha sobre a seleção ou não da(s) instância(s) possa ser feita durante a derivação, pode, também, ser postergada para a fase de execução do processo. Essa estratégia de permitir a escolha durante a fase de execução promove flexibilidade à execução do processo de software. Segundo Lima Reis (2003), não é possível prever antecipadamente todo o processo de desenvolvimento de software. Deste modo, deve ser possível construir o processo aos poucos e o mecanismo de processos deve lidar com processos incompletos. Complementarmente, processos de software envolvem incerteza e não-determinismo e escolhas entre soluções alternativas devem ser permitidas durante a execução. Entretanto, tal flexibilidade implica em um nível maior de complexidade do mecanismo de execução.

Atributos

Sem atributos adicionais.

Associações

Sem associações adicionais.

Restrições

Sem restrições adicionais.

Comportamentos

Sem comportamentos.

Notação

Sem notação.

Exemplos

Sem exemplos.

A.2.2.2.3. ElementExistence

Generalizações

Variability (seção A.2.2.2.1, p. 192).

Especializações

InstanceCardinality (seção A.2.2.2.4, p. 195).

Optional (seção A.2.2.2.5, p. 196).

Mandatory (seção A.2.2.2.6, p. 197).

Descrição

Esta classe abstrata representa uma variação existencial de uma instância de elemento de processo. Através de uma de suas subclasses, especifica se uma instância de elemento é

obrigatória ou opcional, além de poder especificar o número mínimo e máximo de instâncias que podem existir.

Atributos

Sem atributos adicionais.

Associações

- **processElementInstance :** ProcessElementInstance[1]
    - Instância de elemento de processo na qual a variação se aplica.

Restrições

Sem restrições adicionais.

Comportamentos

Sem comportamentos.

Notação

Sem notação.

Exemplos

Sem exemplos.

A.2.2.2.4. InstanceCardinality

Generalizações

ElementExistence (seção A.2.2.2.3, p. 194).

Especializações

Sem especializações.

Descrição

Esta classe especifica o número mínimo e máximo de instâncias de um elemento de processo que podem existir nos processos instanciados a partir da arquitetura de linha de processos.

Atributos

- **min** : int
    - Número mínimo de instâncias.
- **max** : int
    - Número máximo de instâncias.

Associações

Sem associações adicionais.

Restrições

- [1] O valor do atributo "min" deve ser maior ou igual ao valor do atributo "max".

- [2] O valor do atributo "max" deve ser maior do que 0 (zero).
- [3] O valor do atributo "min" não pode ser maior do que 0 (zero) se a instância de elemento de processo for definida como opcional.
- [4] O valor do atributo "min" deve ser maior do que 0 (zero) se a instância de elemento de processo for definida como obrigatória.
- [5] Se o valor do atributo "min" for igual a 0 (zero), não pode haver uma instância de "Mandatory" referenciando a instância de elemento de processo.

Comportamentos

Sem comportamentos.

Notação

Sem notação.

Exemplos

Sem exemplos.

A.2.2.2.5. Optional

Generalizações

ElementExistence (seção A.2.2.2.3, p. 194).

Especializações

Sem especializações.

Descrição

Especifica que uma instância de um elemento de processo é opcional. Uma instância opcional pode ou não ser selecionada. Nesse caso, é necessário tomar a decisão de incluí-la ou não dependendo das necessidades específicas.

Atributos

- **choiceRationale :** String
    - Descrição textual que especifica o contexto em que seria adequada a seleção da instância de elemento de processo, auxiliando, assim, na tomada de decisão.

Associações

Sem associações adicionais.

Restrições

- [1] Não pode haver uma instância de "Optional" referenciando uma instância de elemento de processo definida previamente como obrigatória.

Comportamentos

Sem comportamentos.

Notação

A notação gráfica depende do tipo de elemento de processo, conforme apresentada na seção 5.3.

Exemplos

Sem exemplos.

A.2.2.2.6. Mandatory

Generalizações

ElementExistence (seção A.2.2.2.3, p. 194).

Especializações

Sem especializações.

Descrição

Especifica que uma instância de um elemento de processo é obrigatória. Uma instância obrigatória deve sempre ser selecionada.

Atributos

Sem atributos adicionais.

Associações

Sem associações adicionais.

Restrições

- [1] Não pode haver uma instância de "Mandatory" referenciando uma instância de elemento de processo definida previamente como opcional.
- [2] Não pode haver uma instância de "Mandatory" referenciando uma instância de elemento de processo com cardinalidade mínima igual a 0 (zero).

Comportamentos

Sem comportamentos.

Notação

A notação gráfica depende do tipo de elemento de processo, conforme apresentada na seção 5.3.

Exemplos

Sem exemplos.

A.2.2.2.7. VariationPointRelationship

Generalizações

Variability (seção A.2.2.2.1, p. 192).

Especializações

Sem especializações.

Descrição

Representa a relação entre um ponto de variação e seus variantes. Em termos genéricos, pontos de variações são lugares onde variabilidades ocorrem (SCHINIEDERS; PUHLMANN, 2006, p. 588). Para cada ponto de variação, existe um conjunto de alternativas, os chamados variantes, que podem implementá-lo de diferentes maneiras e é necessário realizar a escolha entre um ou mais variantes. No contexto de processos de software, pontos de variação são elementos de processo que podem mudar de acordo com características específicas do projeto (WASHIZAKI, 2006a, p. 416).

Atributos

- **minCardinality** : int
  - Número mínimo de variantes que podem ser selecionados.
- **maxCardinality** : int
  - Número máximo de variantes que podem ser selecionados.
- **open** : boolean
  - Especifica se o ponto de variação é aberto (valor verdadeiro), ou seja, podem ser acrescentados variantes fora do conjunto de variantes especificados no relacionamento; ou fechado (valor falso), ou seja, o ponto de variação poderá ser preenchido somente pelo conjunto de variantes que fazem parte do relacionamento.

Associações

- **variantionPoint** : ProcessElementInstance[1]
  - Referencia a instância de elemento de processo que faz o papel de ponto de variação.
- **variants** : ProcessElementInstance[1..*]
  - Referencia as instâncias de elemento de processo que fazem os papéis de variantes.

Restrições

- [1] Uma mesma instância de elemento de processo não pode ser referenciada como ponto de variação e variante.
- [2] A instância de elemento de processo referenciada como ponto de variação e as instâncias de elemento de processo referenciadas como variantes devem apontar para elementos de processo do mesmo tipo. Exemplo: se um ponto de variação é do tipo atividade, então os variantes também devem ser atividades.
- [3] A cardinalidade mínima e a cardinalidade máxima devem ser maior ou igual a 0 (zero).
- [4] A cardinalidade máxima deve ser igual ou maior que a cardinalidade mínima.
- [5] Não é possível adicionar um variante que não foi previamente modelado na associação "variants" em um ponto de variação fechado.

Comportamentos

Sem comportamentos.

Notação

A notação gráfica depende do tipo de elemento de processo, conforme especificado na seção 5.3, p. 137.

Exemplos

Sem exemplos.

A.2.2.2.8. Variant

Generalizações

Sem generalizações.

Especializações

Sem especializações.

Descrição

Representa uma instância de elemento de processo que faz o papel de variante em relação a um determinado ponto de variação. Variantes são os candidatos concretos que podem ser aplicados aos pontos de variação (WASHIZAKI, 2006a).

Atributos

- **selectedByDefault :** boolean
    - Especifica se o variante é selecionado por padrão (valor verdadeiro) ou não (valor falso). Quando o valor for verdadeiro, o variante é automaticamente selecionado durante a instanciação, mas ainda é possível desmarca-lo.
- **choiceRationale :** String

- Descrição textual que especifica o contexto em que seria adequada a seleção da instância de elemento de processo, auxiliando, assim, na tomada de decisão.

Associações

Sem associações adicionais.

Restrições

- [1] A quantidade de instâncias de elemento de processo selecionadas por padrão não pode ser maior que a cardinalidade máxima do ponto de variação.
- [2] O valor do atributo "selectedByDefault" deve ser verdadeiro, caso a instância de elemento de processo for definida como obrigatória.

Comportamentos

Sem comportamentos.

Notação

A notação gráfica depende do tipo de elemento de processo, conforme especificado na seção 5.3, p. 137.

Exemplos

Sem exemplos.

A.2.2.2.9. ProcessElementInstance

Generalizações

Sem generalizações.

Especializações

Sem especializações.

Descrição

Representa uma instância de um elemento de processo em uma ALPrS.

Atributos

Sem atributos adicionais.

Associações

Sem associações adicionais.

Restrições

Sem restrições adicionais.

Comportamentos

Sem comportamentos.

Notação

A notação gráfica depende do tipo de elemento de processo, conforme especificado na seção 5.3, p. 137.

Exemplos

Sem exemplos.

A.2.2.3. Dependências

A.2.2.3.1. Dependency

Generalizações

Sem generalizações.

Especializações

Sem generalizações.

Descrição

Representa o relacionamento de dependência entre duas instâncias de elemento de processo.

Atributos

Sem atributos adicionais.

Associações

- **source** : Feature[1]
    - Referencia a instância de elemento de processo de origem da dependência O significado dessa associação depende do tipo de dependência.
- **target** : Feature[1]
    - Referencia a instância de elemento de processo de destino da dependência. O significado dessa associação depende do tipo de dependência.
- **type** : DependencyType[1]
    - Referencia o tipo de dependência, que pode ser: requisito, exclusão, sugestão e substituição.

Restrições

- [1] A instância de elemento de processo de origem deve ser diferente da instância de elemento de processo de destino.
- [2] Não pode haver duas dependências entre as mesmas instâncias de elemento de processo A e B quando A é origem e B é destino, ou vice-versa, nas duas dependências.

- [3] Se existe dependência do tipo requerimento entre as instâncias de elemento de processo A e B e entre as instâncias A e C, não pode haver uma dependência do tipo exclusão entre as instâncias B e C ou entre C e B.
- [4] Se existe dependência do tipo exclusão entre instâncias de elemento de processo A e B e entre as instâncias A e C, não pode haver uma dependência do tipo substituição entre B e C ou entre C e B.
- [5] Não pode haver uma dependência do tipo requerimento quando a instância de elemento de processo de destino é obrigatória.
- [6] Não pode haver uma dependência do tipo exclusão quando instância de elemento de processo de destino é obrigatória.
- [7] Não pode haver uma dependência do tipo sugestão quando a instância de elemento de processo de destino é obrigatória.
- [8] Não pode haver uma dependência do tipo substituição quando a instância de elemento de processo de destino é obrigatória.
- [9] Não pode haver uma dependência do tipo requerimento quando a instância de elemento de processo de origem é obrigatória.
- [9] Não pode haver uma dependência do tipo exclusão quando a instância de elemento de processo de origem é obrigatória.
- [10] Não pode haver uma dependência do tipo substituição quando a instância de elemento de processo de origem é obrigatória.
- [11] Não pode haver uma dependência do tipo exclusão entre um ponto de variação e um variante.
- [12] Não pode haver uma dependência do tipo substituição entre um ponto de variação e um variante.

Comportamentos

Sem comportamentos.

Notação

Sem notação.

Exemplos

Sem exemplos.

A.2.2.3.2. DependencyType

Generalizações

Sem generalizações.

Especializações

Sem especializações.

Descrição

Representa os tipos de dependências que podem ser definidas entre duas instâncias de elementos de processo, como segue:

- Uma dependência do tipo requisito (“requirement”) indica que, se instância de elemento de processo de origem (associação “source”) for selecionada, então a instância de elemento de processo de destino (associação “target”) também deve ser selecionada.
- Uma dependência do tipo exclusão (“exclusion”) indica que, se a instância de elemento de processo de origem (associação “source”) for selecionada, então a instância de elemento de processo de destino (associação “target”) não deve ser selecionada.
- Uma dependência do tipo sugestão (“suggestion”) indica que, se a instância de elemento de processo de origem (associação “source”) for selecionada, então é recomendável que a instância de elemento de processo de destino (associação “target”) seja selecionada. A diferença entre os tipos de dependências requerimento e sugestão é que, enquanto a primeira implica na seleção da instância de elemento de processo, a segunda apenas sugere.
- Uma dependência do tipo substituição (“substitution”) indica que, se a instância de elemento de processo de origem (associação “source”) não for selecionada, então a instância de elemento de processo de destino (associação “target”) deve ser selecionada.

Atributos

Sem atributos adicionais.

Associações

Sem associações adicionais.

Restrições

Sem restrições adicionais.

Comportamentos

Sem comportamentos.

Notação

Sem notação.

Exemplos

Sem exemplos.

A.2.2.4. Modelo de Resolução

A.2.2.4.1. ResolutionModel

Generalizações

Sem generalizações.

Especializações

Sem especializações.

Descrição

Um modelo de resolução define as decisões sobre a seleção ou não das instâncias de elemento de processo de uma arquitetura de linha de processos. Esse modelo armazena escolhas como: a inclusão ou não de instâncias de elemento de processo opcionais e quais variantes de um ponto de variação foram selecionados. Uma mesma ALPrS pode estar associado a vários modelos de resolução, cada um refletindo um contexto diferente, sendo que em cada um desses um conjunto diferente de instâncias de elemento de processo pode estar selecionada.

Esse mecanismo pode agilizar o processo de engenharia de aplicação, além de permitir a reutilização do raciocínio de adaptação de um determinado contexto. Mas o uso do modelo de resolução não é obrigatório, pois as instâncias de elemento de processo ainda podem ser selecionadas manualmente.

Atributos

- **name** : String
  - Nome que identifica o modelo de resolução.
- **description** : String
  - Descrição textual do modelo de resolução.

Associações

- **elementSelections** : ElementSelection[0..*]
  - Conjunto de decisões sobre a inclusão ou não de instâncias de elementos de processo.

Restrições

- [1] Para cada instância de elemento de processo presente na ALPrS, deve haver uma instância de "ElementSelection" associada.

Comportamentos

Sem comportamentos.

Notação

Sem notação.

Exemplos

Sem exemplos.

A.2.2.4.2. ElementSelection

Generalizações

Sem generalizações.

Especializações

Sem especializações.

Descrição

Representa o estado de seleção de uma instância de elemento de processo de uma determinada ALPrS.

Atributos

- **selected** : boolean
    - Especifica se a instância de elemento de processo referenciada deve ser selecionada (valor verdadeiro) ou não (valor falso) em um processo instanciado.

Associações

- **processElementInstance** : ProcessElementInstance[1]
    - Instância de elemento de processo referenciada, que pode ser selecionada ou não, dependendo do valor do atributo "selected".

Restrições

- [1] Se a instância de elemento de processo referenciada for obrigatória, então o valor do atributo "selected" deve ser verdadeiro.
- [2] Se a instância de elemento de processo referenciada for opcional e existe uma instância de elemento de processo selecionada com dependência do tipo requerimento referenciando-a, então o valor do atributo "selected" deve ser verdadeiro.
- [3] Se a instância de elemento de processo referenciada for opcional e existe uma instância de elemento de processo desmarcada com dependência do tipo substituição referenciando-a, então o valor do atributo "selected" deve ser verdadeiro.
- [4] Se a instância de elemento de processo referenciada for opcional e existe uma instância de elemento de processo selecionada com dependência do tipo exclusão referenciando-a, então o valor do atributo "selected" deve ser falso.

Comportamentos

Sem comportamentos.

Notação

Sem notação.

Exemplos

Sem exemplos.

A.2.2.5. Configuração Padrão

A.2.2.5.1. DefaultExistence

Generalizações

Sem generalizações.

Especializações

Sem especializações.

Descrição

Especifica valores padrão para atributos de instâncias de elementos opcionais e obrigatórios.

Atributos

- **optionalSelectedByDefault** : boolean
    - Especifica se uma instância de elemento de processo opcional deve ser selecionada (valor verdadeiro) ou não (valor falso) por padrão durante a criação de um novo modelo de resolução. Se o valor for verdadeiro, então toda instância de elemento de processo opcional criada durante a engenharia de domínio será previamente selecionada durante a derivação. Vale ressaltar que uma instância de elemento de processo selecionada por padrão não implica que ela se torne obrigatória, pois, apesar de ser previamente selecionada, pode ser removida posteriormente. Por outro lado, se o valor for falso, então toda instância de elemento de processo opcional criada durante a engenharia de domínio será previamente não-selecionada.
- **instanceMinCardinality** : int
    - Especifica a cardinalidade mínima padrão de uma instância de elemento de processo. Esse valor somente é aplicável quando existe um objeto "InstanceCardinality" referenciando a instância de elemento de processo em questão.
- **instanceMaxCardinality** : int

- Especifica a cardinalidade máxima padrão de uma instância de elemento de processo. Esse valor somente é aplicável quando existe um objeto "InstanceCardinality" referenciando a instância de elemento de processo em questão.

Associações

Sem associações adicionais.

Restrições

Sem restrições adicionais.

Comportamentos

Sem comportamentos.

Notação

Sem notação.

Exemplos

Sem exemplos.

A.2.2.5.2. DefaultVariationPoint

Generalizações

Sem generalizações.

Especializações

Sem especializações.

Descrição

Especifica valores padrão para atributos do relacionamento entre ponto de variação e variantes.

Atributos

- **open** : boolean
  - Especifica se um ponto de variação deve ser aberto (valor verdadeiro) ou fechado (valor falso) por padrão. Se o valor for verdadeiro, então todo novo ponto de variação criado durante a engenharia de domínio será previamente aberto. Por outro lado, se o valor for falso, então todo novo ponto de variação criado durante a engenharia de domínio será previamente fechado. Vale ressaltar que um ponto de variação aberto pode ser modificado para se tornar fechado e vice-versa.
- **minCardinality** : int

  - Especifica a cardinalidade mínima padrão de um ponto de variação. O valor especificado nesse atributo será utilizado para estabelecer a cardinalidade mínima de todo novo ponto de variação criado durante a engenharia de domínio.

- **maxCardinality** : int
  - Especifica a cardinalidade máxima padrão de um ponto de variação. O valor especificado nesse atributo será utilizado para estabelecer a cardinalidade máxima de todo novo ponto de variação criado durante a engenharia de domínio.

Associações

Sem associações adicionais.

Restrições

[1] A valor do atributo "minCardinality" deve ser menor ou igual ao valor do atributo "maxCardinality".

[2] O valor do atributo "maxCardinality" deve ser maior do que 0 (zero).

Comportamentos

Sem comportamentos.

Notação

Sem notação.

Exemplos

Sem exemplos.

A.2.3. Elementos de Processo

A.2.3.1. ProcessElement

Generalizações

Sem generalizações.

Especializações

As especializações dependem da linguagem de modelagem de processos estendida.

Descrição

Classe abstrata que representa um elemento de processo. Todos os elementos de processo específicos de uma linguagem de modelagem de processos devem ser subclasses diretas ou indiretas de "ProcessElement".

Atributos

Sem atributos adicionais.

Associações

Sem associações adicionais.

Restrições

Sem restrições adicionais.

Comportamentos

Sem comportamentos.

Notação

Sem notação.

Exemplos

Elementos de processo comumente presentes em linguagens de modelagem de processos são atividades, tarefas e produtos de trabalho.

APÊNDICE B. FORMULÁRIOS DE CADASTRO

Este Apêndice apresenta os formulários para o cadastro de modelos e categorias de características, arquiteturas de linha de processos de software e os diversos tipos de elementos de processo.

B.1. Introdução

Os formulários definidos neste Apêndice são utilizados para o cadastro dos conceitos envolvidos com o metamodelo. Cada formulário possui uma série de campos, na primeira coluna, e valores, na segunda coluna. Os valores definidos entre os símbolos "<" e ">" são apenas informativos e devem ser substituídos por valores reais dependentes do contexto que está sendo modelado. Entretanto, os valores que não estão envoltos em tais símbolos são constantes e devem ser mantidos.

B.2. Formulário de Cadastro de Modelo de Características

Para o cadastro de modelos de características, as informações apresentadas na Tabela 47 devem ser preenchidas. O campo "Diagrama" apresenta um diagrama de característica com informações visuais da estrutura do modelo de características. Como não faz parte do escopo deste trabalho especificar uma notação gráfica para o modelo de características, qualquer linguagem pode ser utilizada. O campo "Status" indica a estado do ativo reutilizável e envolve as atividades de gerência da linha de processos.

Tabela 47 – Formulário de Cadastro de Modelo de Características.

<Nome do modelo de características>	
Nome	<Nome do modelo de características>
Descrição	<Descrição do modelo de características>
Diagrama	<Notação gráfica do modelo de características>
Status	<Rascunho, ativo ou descontinuado>

B.3. Formulário de Cadastro de Categoria de Característica

Para o cadastro de categorias de características, as informações apresentadas na Tabela 48 devem ser preenchidas. O campo "Características Associadas" deve conter uma lista de todas as características associadas à categoria em questão, separadas por vírgula.

Tabela 48 – Formulário de Cadastro de Categoria de Característica.

<Nome da categoria de característica>	
Nome	<Nome da categoria de característica>
Descrição	<Descrição da categoria de característica>
Características Associadas	<Lista de características classificadas na categoria>

B.4. Formulário de Cadastro de Arquitetura de Linha de Processos

A Tabela 49 apresenta o formulário para o cadastro de arquitetura de linha de processos de software.

Tabela 49 – Formulário de Cadastro de Arquitetura de Linha de Processos.

<Nome da arquitetura de linha de processos de software>	
Nome	<Nome da arquitetura de linha de processos de software>
Descrição	<Descrição da arquitetura de linha de processos de software>
Características Associadas	<Nomes das características associadas>
Diagrama	<Diagrama gráfico com os elementos de processo utilizados e as variações associadas>
Dependências	<Nome do elemento> <tipo de dependência> <nome do elemento>
Status	<Rascunho, ativo ou descontinuado>

B.5. Formulário de Cadastro de Atividade

Para cadastrar atividades reutilizáveis, as informações sugeridas em SOFTEX (2013) foram reutilizadas com algumas adaptações necessárias. As atividades foram separadas em três tipos: atividades normais, que são indivisíveis; atividades automáticas, que são realizadas sem intervenção humana; e atividades decompostas, que possuem um subprocesso com um conjunto de atividades. O subprocesso de uma atividade composta pode ser composto por atividades normais ou outras atividades decompostas. A Tabela 50 apresenta o formulário para o cadastro de atividade normal.

Tabela 50 – Formulário de Cadastro de Atividade Normal.

<Nome que identifica a atividade>	
Nome	<Nome que identifica a atividade>

Descrição	<Descrição da atividade>
Tipo	Normal
Critérios de Entrada	<Critérios que devem ser atendidos para que a atividade possa ser iniciada>
Critérios de Saída	<Critérios que devem ser atendidos para que a atividade possa ser finalizada>
Responsável	<Responsável pela execução da atividade>
Participantes	<Envolvidos na execução da atividade>
Produtos de Trabalho de Entrada	<Lista dos produtos de trabalho que servem de insumo para a execução da atividade>
Produtos de Trabalho de Saída	<Lista dos produtos de trabalho que são gerados ou modificados com a execução da atividade>
Ferramentas	<Lista das ferramentas que são utilizadas para auxiliar na execução da atividade>
Características Associadas	<Lista de características associadas à atividade>
Status	<Rascunho, ativo ou descontinuado>

A Tabela 51 apresenta o formulário para o cadastro de atividade automática.

Tabela 51 – Formulário de Cadastro de Atividade Automática.

<Nome que identifica a atividade>	
Nome	<Nome que identifica a atividade>
Descrição	<Descrição da atividade>
Tipo	Automática
Critérios de Entrada	<Critérios que devem ser atendidos para que a atividade possa ser iniciada>
Critérios de Saída	<Critérios que devem ser atendidos para que a atividade possa ser finalizada>
Produtos de Trabalho de Saída	<Lista dos produtos de trabalho que são gerados ou modificados com a execução da atividade>
Ferramentas	<Lista das ferramentas que são utilizadas para auxiliar na execução da atividade>
Características Associadas	<Lista de características associadas à atividade>
Status	<Rascunho, ativo ou descontinuado>

Para o cadastro de atividades decompostas, foi acrescentado no formulário o campo "Diagrama", que apresenta o desenho das subatividades que compõem a atividade decomposta, através de uma linguagem de modelagem de processos. A Tabela 52 apresenta o formulário para o cadastro de atividades decompostas.

Tabela 52 – Formulário de Cadastro de Atividade Decomposta.

<Nome que identifica a atividade>	
Nome	<Nome que identifica a atividade>
Descrição	<Descrição da atividade>

Tipo	Decomposta
Critérios de Entrada	<Critérios que devem ser atendidos para que a atividade possa ser iniciada>
Critérios de Saída	<Critérios que devem ser atendidos para que a atividade possa ser finalizada>
Responsável	<Papel responsável pela execução da atividade>
Participante	<Papéis envolvidos na execução da atividade>
Produtos de Trabalho de Entrada	<Lista dos produtos de trabalho que servem de insumo para a execução da atividade>
Produtos de Trabalho de Saída	< Lista dos produtos de trabalho que são gerados ou modificados com a execução da atividade>
Ferramentas	<Lista das ferramentas que são utilizadas para auxiliar na execução da atividade>
Diagrama	<Diagrama gráfico com as subatividades da atividade decomposta>
Características Associadas	<Lista de características associadas à atividade>
Status	<Rascunho, ativo ou descontinuado>

B.6. Formulário de Cadastro de Papel

A Tabela 53 apresenta o formulário para o cadastro de papéis.

Tabela 53 – Formulário de Cadastro de Papel.

<Nome que identifica o papel>	
Nome	<Nome que identifica o papel>
Descrição	<Descrição do papel>
Características Associadas	<Lista de características associadas ao papel>
Status	<Rascunho, ativo ou descontinuado>

B.7. Formulário de Cadastro de Grupo

A Tabela 54 apresenta o formulário para o cadastro de grupos.

Tabela 54 – Formulário de Cadastro de Grupo.

<Nome que identifica o grupo>	
Nome	<Nome que identifica o grupo>
Descrição	<Descrição do grupo>
Papéis	<Lista de papéis pertencentes ao grupo>
Características Associadas	<Lista de características associadas ao grupo>
Status	<Rascunho, ativo ou descontinuado>

B.8. Formulário de Cadastro de Produto de Trabalho

A Tabela 55 apresenta o formulário para o cadastro de produtos de trabalho.

Tabela 55 – Formulário de Cadastro de Produto de Trabalho.

<Nome que identifica o produto de trabalho>	
Nome	<Nome que identifica o produto de trabalho>
Descrição	<Descrição do produto de trabalho>
Características Associadas	<Lista de características associadas ao produto de trabalho>
Status	<Rascunho, ativo ou descontinuado>

B.9. Formulário de Cadastro de Ferramenta

A Tabela 56 apresenta o formulário para o cadastro de ferramentas.

Tabela 56 – Formulário de Cadastro de Ferramenta.

<Nome que identifica a ferramenta>	
Nome	<Nome que identifica a ferramenta>
Descrição	<Descrição da ferramenta>
Características Associadas	<Lista de características associadas à ferramenta>
Status	<Rascunho, ativo ou descontinuado>

APÊNDICE C. FORMULÁRIOS DE AVALIAÇÃO

Este Apêndice apresenta os formulários para a avaliação de modelos e categorias de características, arquiteturas de linha de processos e os diversos tipos de elementos de processo.

C.1. Introdução

Os formulários definidos neste Apêndice são utilizados para a avaliação da qualidade dos processos reutilizáveis, sendo parte das atividades da Gerência da Linha de Processos especificadas na seção 5.4.3, p. 145.

C.2. Formulário de Avaliação de Modelo de Características

Para a avaliação de modelo de características, as informações apresentadas na Tabela 57 devem ser preenchidas.

Tabela 57 – Formulário de Avaliação de Modelo de Características.

Modelo de Características	<Nome do modelo de características>	
1. Propósito e Reusabilidade		
Questão	**Descrição**	**Resposta**
1.1. O conceito referente ao modelo de características é útil para a organização?	Verificar se o modelo de características é útil e agregará valor.	
1.2. O modelo de características está duplicado?	Verificar já se existe, no repositório, outro modelo de características com o mesmo propósito do que está sendo avaliado.	
1.3. O modelo de características é adequado para ser reutilizado em diversos contextos?	Verificar se o modelo de características é genérico o suficiente para ser reutilizado em diversos contextos.	
2. Identificação		
Questão	**Descrição**	**Resposta**
2.1. O nome do modelo de características é adequado?	Verificar se o nome do modelo de características é adequado para o seu	

	propósito, facilita a sua identificação e está de acordo com o padrão de nomenclatura definido.	
2.2. A descrição do modelo de características é adequada?	Verificar se a descrição do modelo de características é adequada para o seu propósito e contém todas as informações necessárias.	
3. Estrutura de Características		
Questão	**Descrição**	**Resposta**
3.1. Os nomes e a descrições das características são adequados?	Verificar se os nomes e descrições das características são adequados para os seus propósitos.	
3.2. A hierarquia de características está adequadamente representada?	Verificar se a hierarquia de características reflete adequadamente a relação de composição entre as mesmas.	
3.3. Existe alguma característica que pode ser removida do modelo?	Verificar se alguma característica está fora do escopo do modelo de características e deveria ser removida.	
3.4. Existe alguma característica que pode ser adicionada ao modelo?	Verificar se alguma característica poderia ser inserida no escopo do modelo de características.	
3.5. As características foram associadas às categorias adequadas?	Verificar se cada característica foi classificada nas categorias adequadas.	

C.3. Formulário de Avaliação de Categoria de Característica

Para a avaliação de categorias de características, as informações apresentadas na Tabela 58 devem ser preenchidas.

Tabela 58 – Formulário de Avaliação de Categoria de Característica.

Categoria de Característica	<Nome da categoria de características>	
1. Propósito e Reusabilidade		
Questão	**Descrição**	**Resposta**
1.1. O conceito referente à categoria de característica é útil para a organização?	Verificar se a categoria de característica é útil e agregará valor.	
1.2. A categoria de característica está duplicada?	Verificar já se existe, no repositório, outra categoria de característica com o mesmo propósito do que está sendo	

	avaliado.	
1.3. A categoria de característica é adequada para ser reutilizada em diversos contextos?	Verificar se a categoria de característica é genérica o suficiente para ser reutilizada em diversos contextos.	
2. Identificação		
Questão	**Descrição**	**Resposta**
2.1. O nome da categoria de característica é adequado?	Verificar se o nome da categoria de característica é adequado para o seu propósito, facilita a sua identificação e está de acordo com o padrão de nomenclatura definido.	
2.2. A descrição da categoria de característica é adequada?	Verificar se a descrição da categoria de característica é adequada para o seu propósito e contém todas as informações necessárias.	

C.4. Formulário de Avaliação de Arquitetura de Linha de Processos

Para a avaliação de arquiteturas de linha de processos, as informações apresentadas na Tabela 59 devem ser preenchidas.

Tabela 59 – Formulário de Avaliação de Arquitetura de Linha de Processos.

Arquitetura de Linha de Processos	<Nome da arquitetura de linha de processos>	
1. Propósito e Reusabilidade		
Questão	**Descrição**	**Resposta**
1.1. O conceito referente à arquitetura de linha de processos é útil para a organização?	Verificar se a arquitetura de linha de processos é útil e agregará valor.	
1.2. A arquitetura de linha de processos está duplicada?	Verificar já se existe, no repositório, outra arquitetura de linha de processos com o mesmo propósito da que está sendo avaliada.	
1.3. A arquitetura de linha de processos é adequada para ser reutilizada em diversos contextos?	Verificar se a arquitetura de linha de processos é genérica o suficiente para ser reutilizada em diversos contextos.	
2. Identificação		
Questão	**Descrição**	**Resposta**
2.1. O nome da arquitetura de	Verificar se o nome da	

linha de processos é adequado?	arquitetura de linha de processos é adequado para o seu propósito, facilita a sua identificação e está de acordo com o padrão de nomenclatura definido.	
2.2. A descrição arquitetura de linha de processos é adequada?	Verificar se a descrição da arquitetura de linha de processos é adequada para o seu propósito e contém todas as informações necessárias.	
3. Estrutura de Características		
Questão	**Descrição**	**Resposta**
3.1. Existe alguma característica que pode ser removida da arquitetura?	Verificar se alguma característica está fora do escopo da arquitetura de linha de processos.	
3.2. Existe alguma característica que pode ser adicionada à arquitetura de linha de processos?	Verificar se alguma característica poderia ser inserida no escopo da arquitetura de linha de processos.	
4. Elementos de Processo		
Questão	**Descrição**	**Resposta**
4.1. Existe algum elemento de processo que pode ser removido da arquitetura de linha de processos?	Verificar se algum elemento de processo está fora do escopo da arquitetura de linha de processos.	
4.2. Existe algum elemento de processo que pode ser adicionado à arquitetura de linha de processos?	Verificar se algum elemento de processo poderia ser inserido no escopo da arquitetura de linha de processos.	
4.3. As conexões entre os elementos são adequadas?	Verificar se o fluxo entre as atividades e as conexões entre atividades, papéis, produtos de trabalho e outros elementos são adequados.	
4.4. As dependências entre os elementos são adequadas?	Verificar se as dependências dos tipos requerimento, exclusão, substituição e sugestão foram especificadas de forma adequada.	
5. Variações		
Questão	**Descrição**	**Resposta**
5.1. Os elementos foram adequadamente especificados como obrigatório ou opcional?	Verificar se todos os elementos obrigatórios devem realmente estar presentes em todos os processos da linha de processos e se os elementos opcionais foram classificados adequadamente.	

5.2. Os pontos de variação foram definidos com seus respectivos variantes?	Verificar se existe a necessidade de incluir ou excluir variantes em cada ponto de variação.	

C.5. Formulário de Avaliação de Atividade

Para a avaliação de atividades, as informações apresentadas na Tabela 60 devem ser preenchidas.

Tabela 60 – Formulário de Avaliação de Atividade.

Atividade	<Nome da atividade>	
1. Propósito e Reusabilidade		
Questão	**Descrição**	**Resposta**
1.1. O conceito referente à atividade é útil para a organização?	Verificar se a atividade é útil e agregará valor.	
1.2. A atividade está duplicada?	Verificar já se existe, no repositório, outra atividade com o mesmo propósito da que está sendo avaliada.	
1.3. A atividade é adequada para ser reutilizada em diversos contextos?	Verificar se a atividade é genérica o suficiente para ser reutilizada em diversos contextos.	
2. Identificação		
Questão	**Descrição**	**Resposta**
2.1. O nome da atividade é adequado?	Verificar se o nome da atividade é adequado para o seu propósito, facilita a sua identificação e está de acordo com o padrão de nomenclatura definido.	
2.2. A descrição da atividade é adequada?	Verificar se a descrição da atividade é adequada para o seu propósito e contém todas as informações necessárias.	
2.3. A atividade foi especificada com o tipo adequado?	Verificar se a atividade foi especificada como normal ou decomposta adequadamente. Uma atividade normal pouco coesa pode ser especificada como uma decomposta com subatividades. Por outro lado, uma atividade decomposta sem necessidade de subatividades pode ser	

	especificada como uma atividade normal	
2.4. Os critérios de entrada são adequados?	Verificar se todos os critérios de entrada foram identificados e se especificam as condições para que a atividade possa ser executada.	
2.5. Os critérios de saída são adequados?	Verificar se todos os critérios de saída foram identificados e se especificam as condições para que a atividade possa ser finalizada.	
2.6. O papel do responsável é adequado?	Verificar se o papel especificado como responsável realmente tem a responsabilidade pela atividade. (Não aplicável para atividade automática)	
2.7. Os participantes são adequados?	Verificar se os papéis definidos como participantes realmente devem executar a atividade. (Não aplicável para atividade automática)	
2.8. Os produtos de trabalho de entrada são adequados?	Verificar se todos os produtos de trabalho de entrada foram definidos para a atividade. (Não aplicável para atividade automática)	
2.9. Os produtos de trabalho de saída são adequados?	Verificar se todos os produtos de trabalho de saída foram definidos para a atividade (Não aplicável para atividade automática).	
2.10. As ferramentas são adequadas?	Verificar se as ferramentas especificadas para auxiliar na execução da atividade são adequadas.	
2.11. O diagrama é adequado?	Verificar se o diagrama está adequado. O diagrama especifica as subatividades, e os fluxos entre elas, de uma atividade decomposta (Não aplicável para atividade automática ou normal).	
2.12. A atividade foi associada às características adequadas?	Verificar se todas as características associadas à atividade foram especificadas.	

C.6. Formulário de Avaliação de Papel

Para a avaliação de papéis, as informações apresentadas na Tabela 61 devem ser preenchidas.

Tabela 61 – Formulário de Avaliação de Papel.

Papel	<Nome do papel>	
1. Propósito e Reusabilidade		
Questão	**Descrição**	**Resposta**
1.1. O conceito referente ao papel é útil para a organização?	Verificar se o papel é útil e agregará valor.	
1.2. O papel está duplicado?	Verificar já se existe, no repositório, outro papel com o mesmo propósito do que está sendo avaliado.	
1.3. O papel é adequado para ser reutilizado em diversos contextos?	Verificar se o papel é genérico o suficiente para ser reutilizado em diversos contextos.	
2. Identificação		
Questão	**Descrição**	**Resposta**
2.1. O nome do papel é adequado?	Verificar se o nome do papel é adequado para o seu propósito, facilita a sua identificação e está de acordo com o padrão de nomenclatura definido.	
2.2. A descrição do papel é adequada?	Verificar se a descrição do papel é adequada para o seu propósito e contém todas as informações necessárias.	
2.3. O papel foi associado às características adequadas?	Verificar se todas as características associadas ao papel foram especificadas.	

C.7. Formulário de Avaliação de Grupo

Para a avaliação de grupos, as informações apresentadas na Tabela 62 devem ser preenchidas.

Tabela 62 – Formulário de Avaliação de Grupo.

Grupo	<Nome do grupo>	
1. Propósito e Reusabilidade		
Questão	**Descrição**	**Resposta**
1.1. O conceito referente grupo é útil para a	Verificar se o grupo é útil e agregará valor.	

organização?		
1.2. O grupo está duplicado?	Verificar já se existe, no repositório, outro grupo com o mesmo propósito do que está sendo avaliado.	
1.3. O grupo é adequado para ser reutilizado em diversos contextos?	Verificar se o grupo é genérico o suficiente para ser reutilizado em diversos contextos.	
2. Identificação		
Questão	**Descrição**	**Resposta**
2.1. O nome do grupo é adequado?	Verificar se o nome do grupo é adequado para o seu propósito, facilita a sua identificação e está de acordo com o padrão de nomenclatura definido.	
2.2. A descrição do grupo é adequada?	Verificar se a descrição do grupo é adequada para o seu propósito e contém todas as informações necessárias.	
2.3. O grupo foi associado às características adequadas?	Verificar se todas as características associadas ao grupo foram especificadas.	
2.4. Os papéis associados ao grupo são adequados?	Verificar se os papéis que compõem o grupo são adequados.	

C.8. Formulário de Avaliação de Produto de Trabalho

Para a avaliação de modelo de características, as informações apresentadas na Tabela 63 devem ser preenchidas.

Tabela 63 – Formulário de Avaliação de Produto de Trabalho.

Produto de Trabalho	<Nome do produto de trabalho>	
1. Propósito e Reusabilidade		
Questão	**Descrição**	**Resposta**
1.1. O conceito referente ao produto de trabalho é útil para a organização?	Verificar se o produto de trabalho é útil e agregará valor.	
1.2. O produto de trabalho está duplicado?	Verificar já se existe, no repositório, outro produto de trabalho com o mesmo propósito do que está sendo avaliado.	
1.3. O produto de trabalho é adequado para ser reutilizado em diversos contextos?	Verificar se o produto de trabalho é genérico o suficiente para ser reutilizado em diversos contextos.	

2. Identificação		
Questão	**Descrição**	**Resposta**
2.1. O nome do produto de trabalho é adequado?	Verificar se o nome do produto de trabalho é adequado para o seu propósito, facilita a sua identificação e está de acordo com o padrão de nomenclatura definido.	
2.2. A descrição do produto de trabalho é adequada?	Verificar se a descrição do produto de trabalho é adequada para o seu propósito e contém todas as informações necessárias.	
2.3 O produto de trabalho foi associado às características adequadas?	Verificar se todas as características associadas ao produto de trabalho foram especificadas.	

C.9. Formulário de Avaliação de Ferramenta

Para a avaliação de modelo de características, as informações apresentadas na Tabela 64 devem ser preenchidas.

Tabela 64 – Formulário de Avaliação de Ferramenta.

Ferramenta	<Nome da ferramenta>	
1. Propósito e Reusabilidade		
Questão	**Descrição**	**Resposta**
1.1. O conceito referente à ferramenta é útil para a organização?	Verificar se a ferramenta é útil e agregará valor.	
1.2. A ferramenta está duplicada?	Verificar já se existe, no repositório, outra ferramenta com o mesmo propósito do que está sendo avaliado.	
1.3. A ferramenta é adequada para ser reutilizada em diversos contextos?	Verificar se a ferramenta é genérica o suficiente para ser reutilizada em diversos contextos.	
2. Identificação		
Questão	**Descrição**	**Resposta**
2.1. O nome da ferramenta é adequado?	Verificar se o nome da ferramenta é adequado para o seu propósito, facilita a sua identificação e está de acordo com o padrão de nomenclatura definido.	

2.2. A descrição da ferramenta é adequada?	Verificar se a descrição da ferramenta é adequada para o seu propósito e contém todas as informações necessárias.	
2.3. A ferramenta foi associada às características adequadas?	Verificar se todas as características associadas à ferramenta foram especificadas.	